Manual

TF5100 TwinCAT 3 NC I

TwinCAT 3

Version: 1.2
Date: 2018-04-22

BECKHOFF

Table of contents

1 Introduction

The TwinCAT NCI stands for 'numerical control interpolation' and is the NC system for interpolated path movements.
TwinCAT NCI offers 3D interpolation (interpreter, setpoint generation, position controller), an integrated PLC with an NC interface and an I/O connection for axes via the fieldbus.

NCI can be used to drive 3 path axes and up to 5 auxiliary axes per channel. In addition, master/slave couplings can be formed. In combination with TwinCAT Kinematic Transformation (TS 511x), complex kinematic systems can be controlled via NCI.

Programming is done with a dedicated NC program, based on DIN 66025, with its own language extensions (cf. Interpreter (DIN 66025/G-Code) [▸ 107]) or directly from the PLC with the PLC Library: Tc2_PlcInterpolation [▸ 276].

Installation preconditions

TwinCAT NCI is integrated in the TwinCAT 3 installation.

Overview

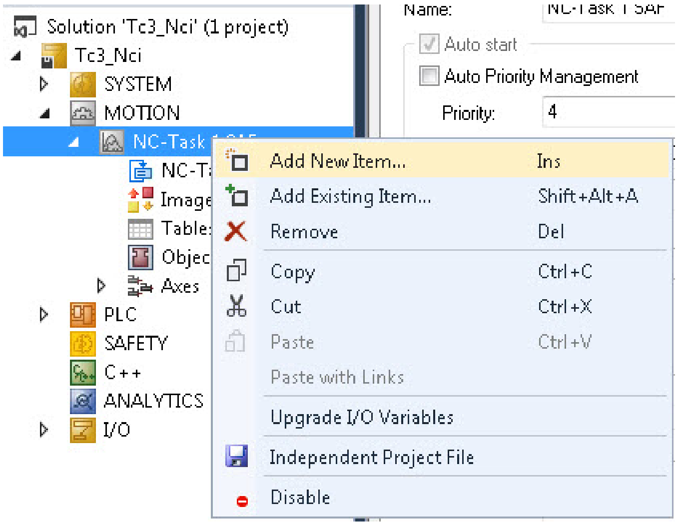
Chapter	Contents
XAE user interface [▸ 8]	Description of the parameters and functionalities for the interpreter in the TwinCAT 3 Engineering environment (XAE)
Interpreter [▸ 107]	Interpreter programming instructions.
PLC NCI Libraries [▸ 181]	Description of the special NCI libraries
Samples [▸ 293]	Samples for using TwinCAT NCI with PLC and parts program, and for direct motion control from the PLC with the Tc2_PlcInterpolation library
Appendix [▸ 301]	Parameterization, cyclic channel interface

2 User interface in the TwinCAT 3 Engineering environment

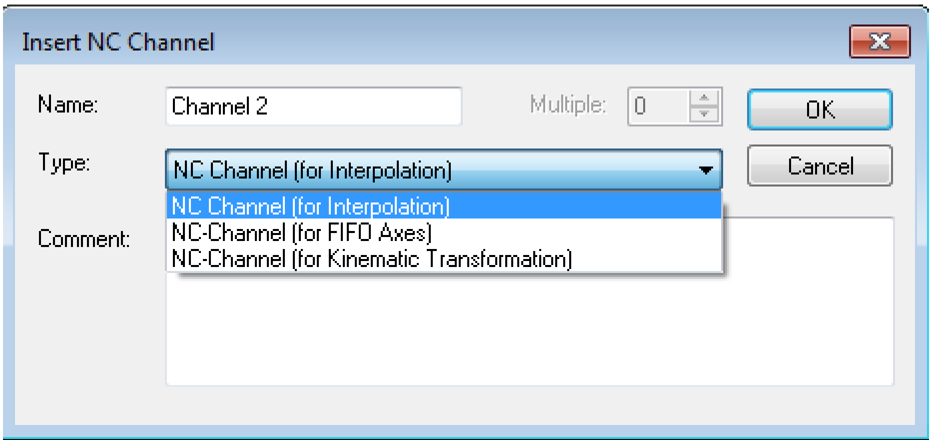
2.1 Outline

In order to be able to use the interpolation, add an interpolation channel in the XAE . This applies to the interpreter and the PLC Library: Tc2_PlcInterpolation [▶ 276].

1. Create an NC channel.



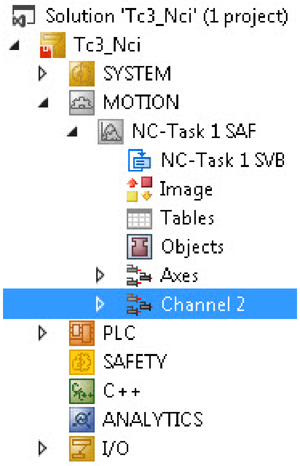
2. In the selection box select the NC channel for the interpolation.



3. Assign PTP axes to it from the PLC via a function block.

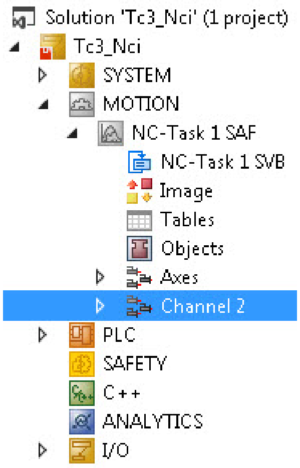
⇨ The created channel consists of the following elements:

Interpolation Channel [▶ 9]	Description of the properties pages embedded in the 'interpolation' element.
Interpreter Element [▶ 11]	Description of the properties pages embedded in the 'Interpreter' element
Group element [▶ 20]	Description of the properties pages embedded in the 'group' element



NOTE! Axis-specific parameters for NCI can be found in the axis parameterization under subitem 'NCI parameters'.

2.2 Interpolation Channel

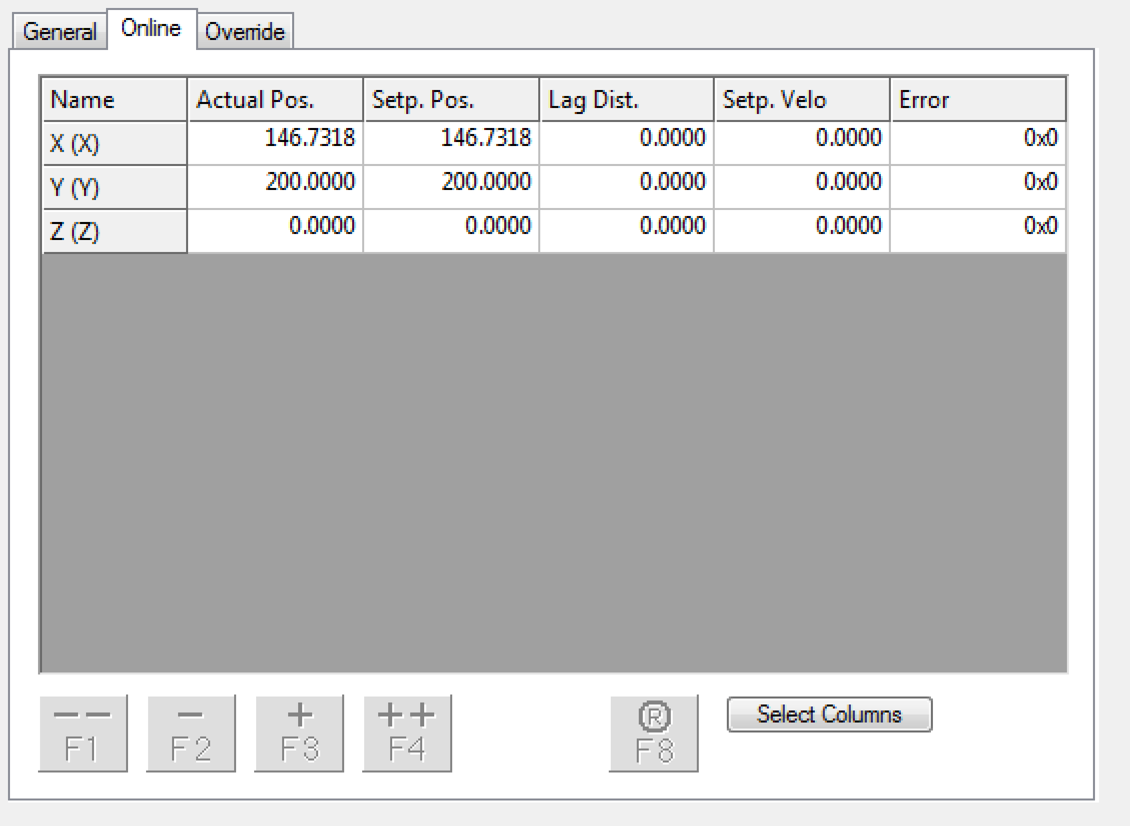


Click on the interpolation channel to display the following dialogs:

"Online" tab

All the axes in the current Interpolation Group [▸ 20] will be listed. Currently shown:

- Actual positions
- Set positions
- Following errors
- Set velocities and
- Error Codes

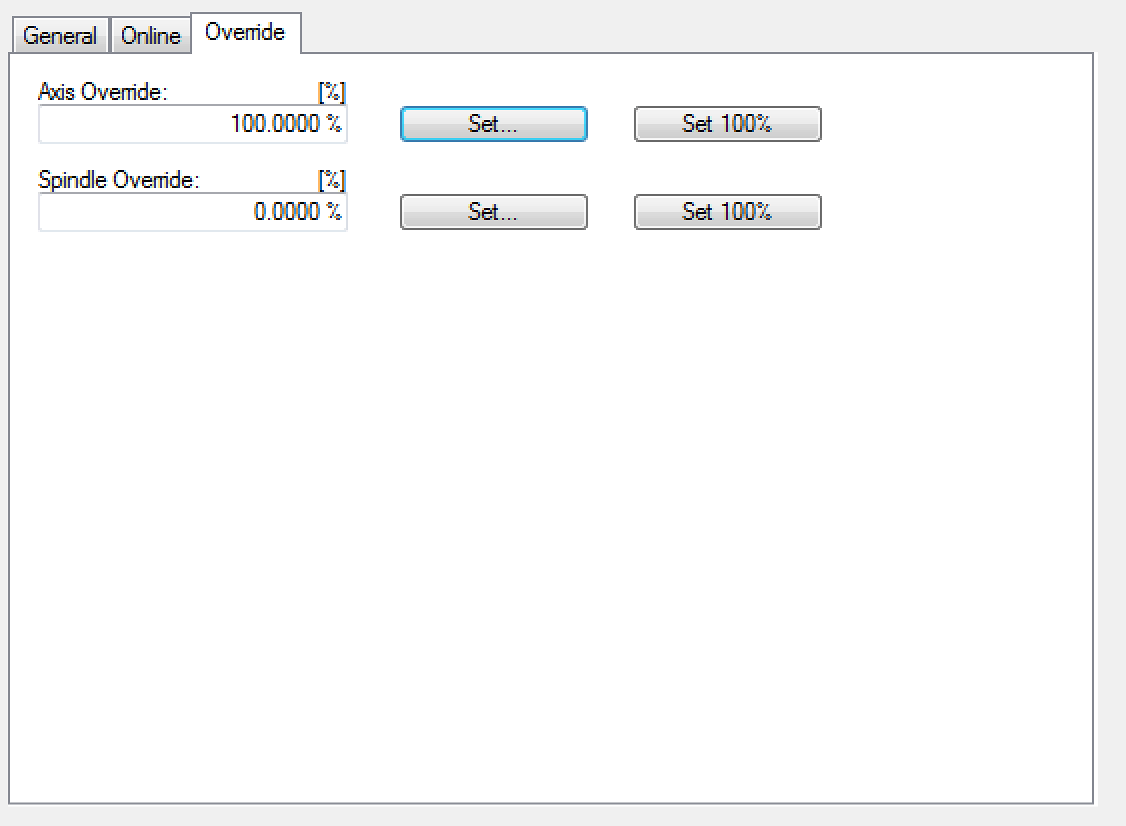


"Override" tab

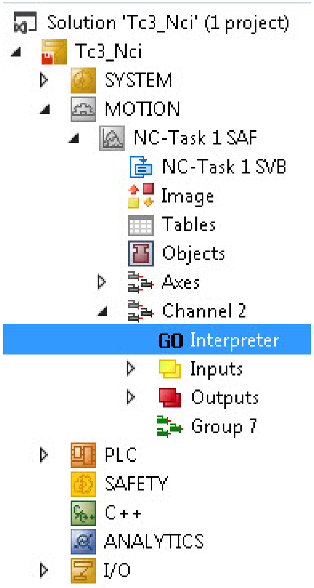
The channel override for the axes can be read and set on the 'Override' page. If PLC is running and the cyclical channel interface [▸ 306] is being written, the override set here will be overwritten by the PLC.

Further information on the override principle can be found under Path override (interpreter override types) [▸ 304].

The spindle override is described by the cyclic channel interface, although it is currently not supported.

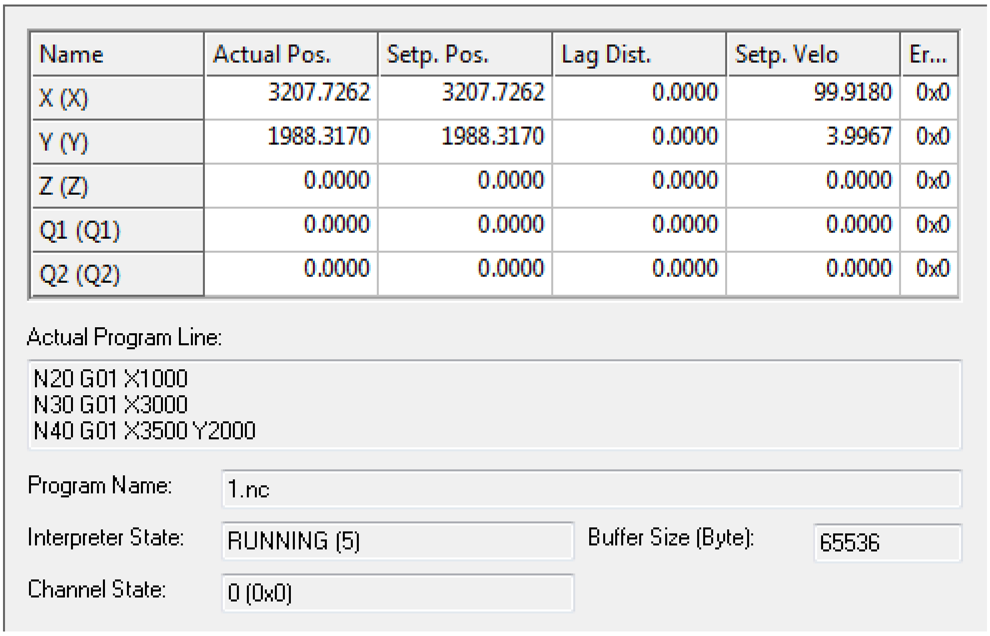


2.3 Interpreter element



Click on “Interpreter” to show the following property pages and the online window:

2.3.1 Interpreter online window



Axes

As on the "Online" properties page in the interpolation channel, this window lists all axes currently included in the interpolation group. Values for the following parameters are displayed:

- actual positions
- set positions
- following errors
- set velocities and
- current error codes

.

Actual Program Line

The Actual Program Line shows the current NC block to be processed in the block execution. The last row in the window is the current block.

As for nearly all the parameters, the program display can be read off via ADS. This can be used to display the current NC blocks in a Visual Basic application, for example (see ADS device documentation - ADS Interface NC [▶ 308]).

Program name

Displays the name of the currently loaded program. This does not necessarily have to be the program displayed in Editor.

Interpreter status

The interpreter status indicates the current status of the interpreter state machine. The complete list is given below. As PLC evaluation does not require all status information, only the most important parameters are explained.

Status	Description
ITP_STATE_IDLE	The interpreter is in idle state when there is no NC program loaded as yet or when a group reset is being executed. The interpreter also goes into idle state when a current program is stopped. In the case a group reset must be executed in order to prevent error 0x42C5. It is therefore recommended to execute a group reset after stopping via the PLC.
ITP_STATE_READY	After successful loading of an NC program, the interpreter is in ready state. After a program has been successfully processed and exited, the interpreter goes into ready state. In the meantime, however, other states are accepted.
ITP_STATE_ABORTED	If a runtime error occurs during the processing of an NC program, the interpreter goes into aborted state. The actual error code is given in the channel status
ITP_STATE_SINGLESTOP	This status is **only** accepted in Single Block Mode [▶ 114]. As soon as the entry has been sent from the interpreter to the NC core, the interpreter goes into this mode.

Note

Querying the interpreter status during program execution

Since the interpreter status may change between different states during program execution, we recommend querying it with a negative logic. During program execution the interpreter state is not necessarily ITP_STATE_RUNNING. If the program was executed successfully, the interpreter is subsequently always in Ready state (see also Samples [▶ 293]).

Interpreter status return values

```
0 ITP_STATE_INITFAILED
1 ITP_STATE_IDLE
2 ITP_STATE_READY
3 ITP_STATE_STARTED
4 ITP_STATE_SCANNING
5 ITP_STATE_RUNNING
6 ITP_STATE_STAY_RUNNING
7 ITP_STATE_WRITETABLE
8 ITP_STATE_SEARCHLINE
9 ITP_STATE_END
10 ITP_STATE_SINGLESTOP
11 ITP_STATE_ABORTING
12 ITP_STATE_ABORTED
13 ITP_STATE_FAULT
14 ITP_STATE_RESET
15 ITP_STATE_STOP
16 ITP_STATE_WAITFUNC
17 ITP_STATE_FLUSHBUFFERS
```

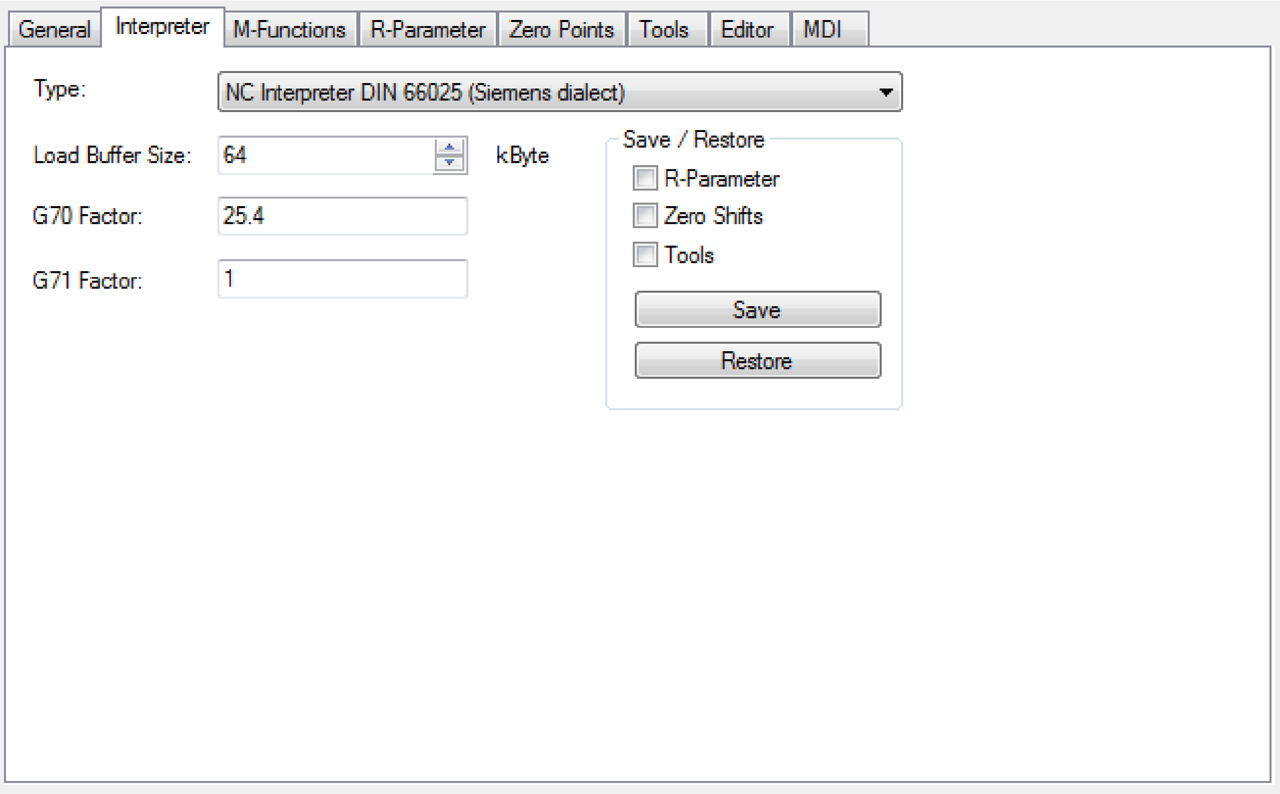
Channel status

The channel status indicates the current error state of the channel. If an error occurs during NC program loading or runtime, the corresponding error code [▶ 313] is displayed here. If, for example, an axis following error occurs during processing, the NC program is stopped and the channel status will have a value unequal 0. The channel status should therefore always be checked in the PLC, in order to be able to respond to errors. The channel status is always 0 during normal operation.

Loading buffer

The current size of the loading buffer for the interpreter is displayed here. Select the "Interpreter" tab to change the value.

2.3.2 "Interpreter" tab



Type

The interpreter type can be selected in the Type selection box. Available are

- the GST-interpreter. GST combines native DIN 66025 based G-code with programming extensions of Structured Text as a higher level language.
- The DIN 66025 based NC-interpreter [▶ 107] with @-command register function extensions.
- The selection of none if the PlcInterpolation [▶ 276] library is used.

As default setting the GST-interpreter is set. To employ the NC-interpreter with register function extensions you have to select it explicitly.

Loading Buffer Size

The loading buffer for the interpreter can be edited here. Note that the memory required in the interpreter is substantially greater than the size of the NC-file. The maximum permitted loading buffer size is limited to 64 MB.

Note	**Changing the Loading Buffer Size** If the size of the loading buffer is changed, it is absolutely necessary to execute a TwinCAT restart.

G70/G71 Factor

If a switch from G71 [▶ 113] (millimeters - default) to `G70` takes place in the parts program, the conversion factor is stored here. This conversion factor only has to be edited if the base reference system is not millimeters.

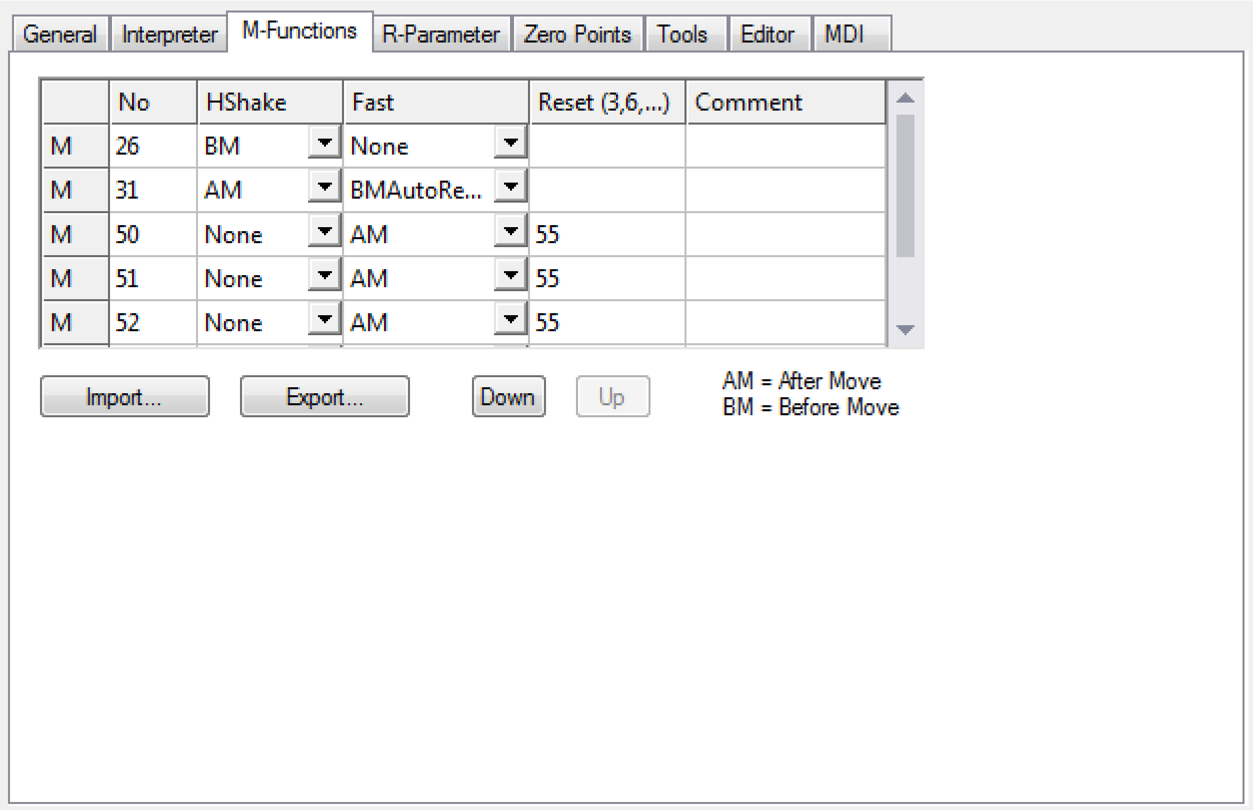
If for example the machine was calibrated based on inches and `G70` is activated in the parts program, the `G70` factor should be set to `1` and the `G71` factor should be set to `1/25.4`.

Save/Restore

At runtime the Save function can be used to save a “snapshot” of the current parameters. The checkboxes can be used to specify the parameters to be saved. The Save function generates the file ‘SnapShot.bin’ in the TwinCAT\CNC directory.

The Restore function loads the file saved with the Save function. This function is solely intended for debugging purposes.

2.3.3 "M-Functions" tab



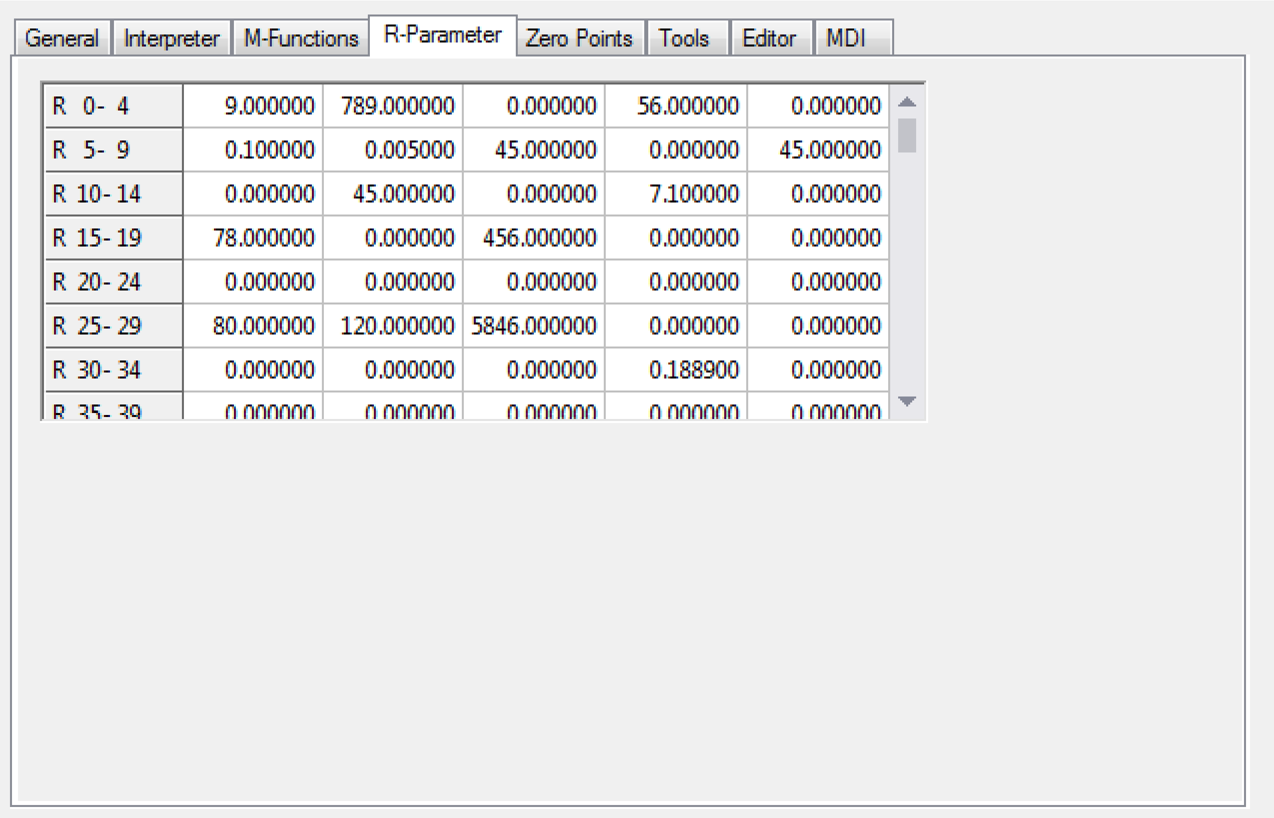
Note	**Use only with interpreter** This tab is irrelevant for operation with the library Tc2_PlcInterpolation.

Shows the currently parameterized M-functions. On this page new M-functions can be added, or existing ones modified.

A more detailed description of the available parameters can be found in the interpreter description under M-functions [▶ 145].

Note	**Parameterization of M-functions** If M-functions are re-parameterized, subsequent activation of the configuration and a Twin-CAT restart is required.

2.3.4 "R parameters" tab

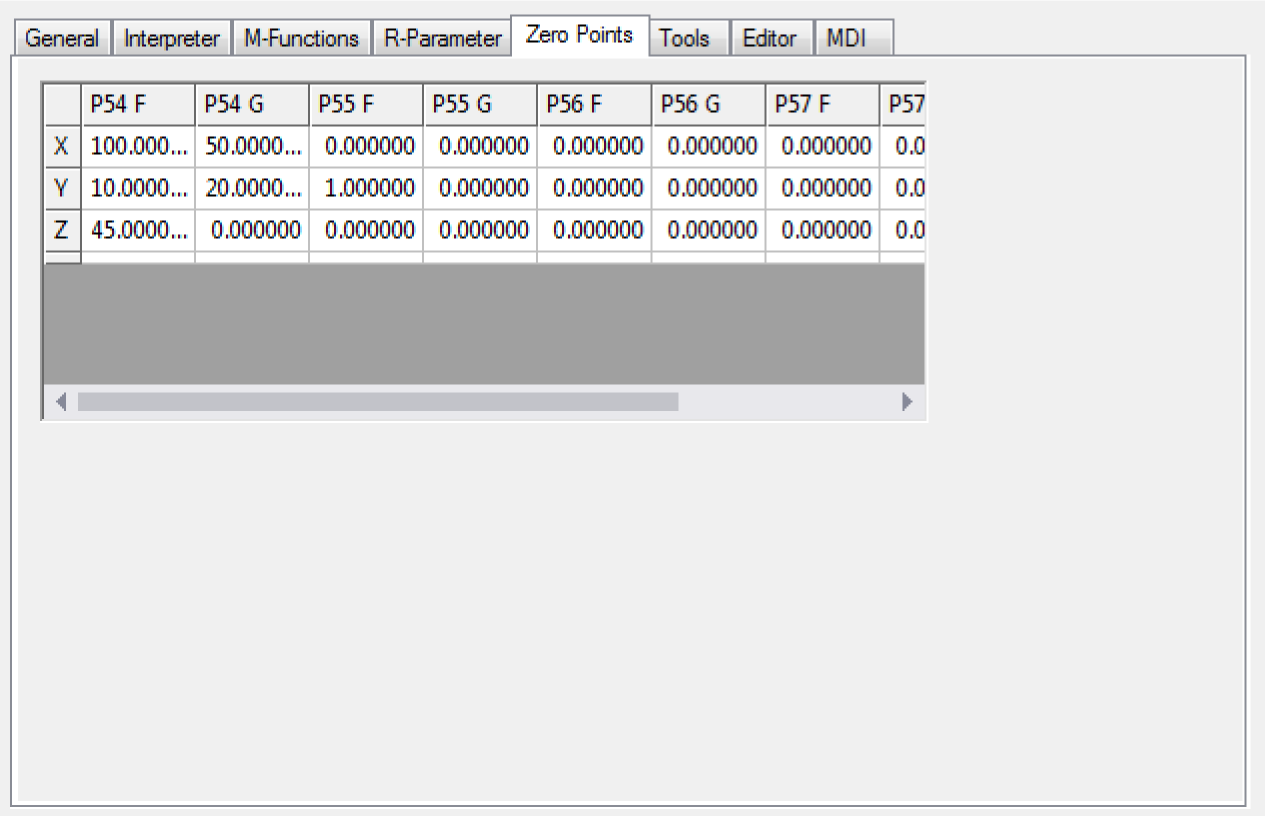


R 0- 4	9.000000	789.000000	0.000000	56.000000	0.000000
R 5- 9	0.100000	0.005000	45.000000	0.000000	45.000000
R 10- 14	0.000000	45.000000	0.000000	7.100000	0.000000
R 15- 19	78.000000	0.000000	456.000000	0.000000	0.000000
R 20- 24	0.000000	0.000000	0.000000	0.000000	0.000000
R 25- 29	80.000000	120.000000	5846.000000	0.000000	0.000000
R 30- 34	0.000000	0.000000	0.000000	0.188900	0.000000
R 35- 39	0.000000	0.000000	0.000000	0.000000	0.000000

The currently applicable R parameters are displayed on the 'R parameters' properties page. During the test phase it is possible to, for example, initialize or change R parameters here. R parameters are generally edited, however, from the NC program or if necessary, from the PLC.

You can find further information about R parameters in the interpreter description under R Parameters [▸ 115].

2.3.5 "Zero point" tab

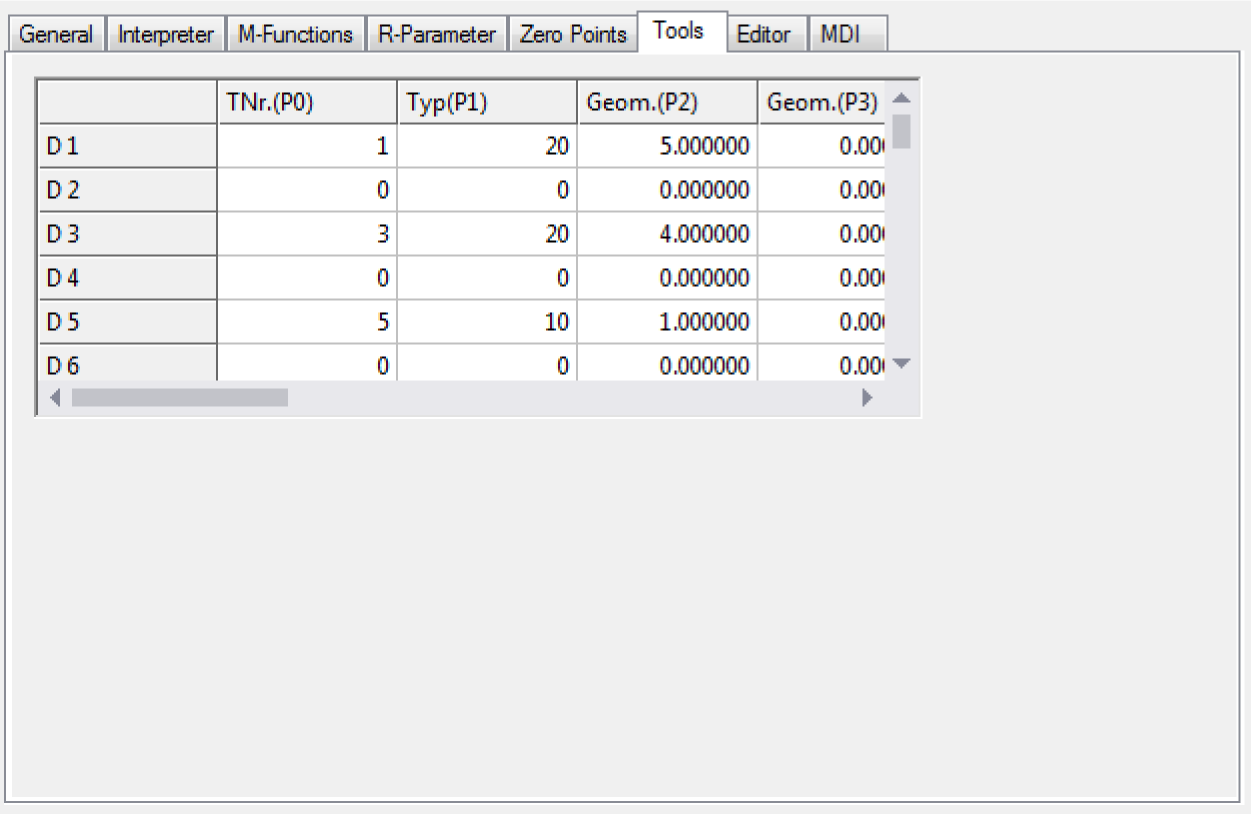


The current zero shift values for the axes within the interpolation group are displayed here. The parameters P54..P59 represent for the corresponding G code. As for the R parameters, the zero shift values can be edited from here.

NOTE! Columns F & G (e.g. P54 F & P54 G) exist for historical reasons and are added for each parameter.

You can find further details of the effects in the interpreter description under zero shifts [▸ 124].

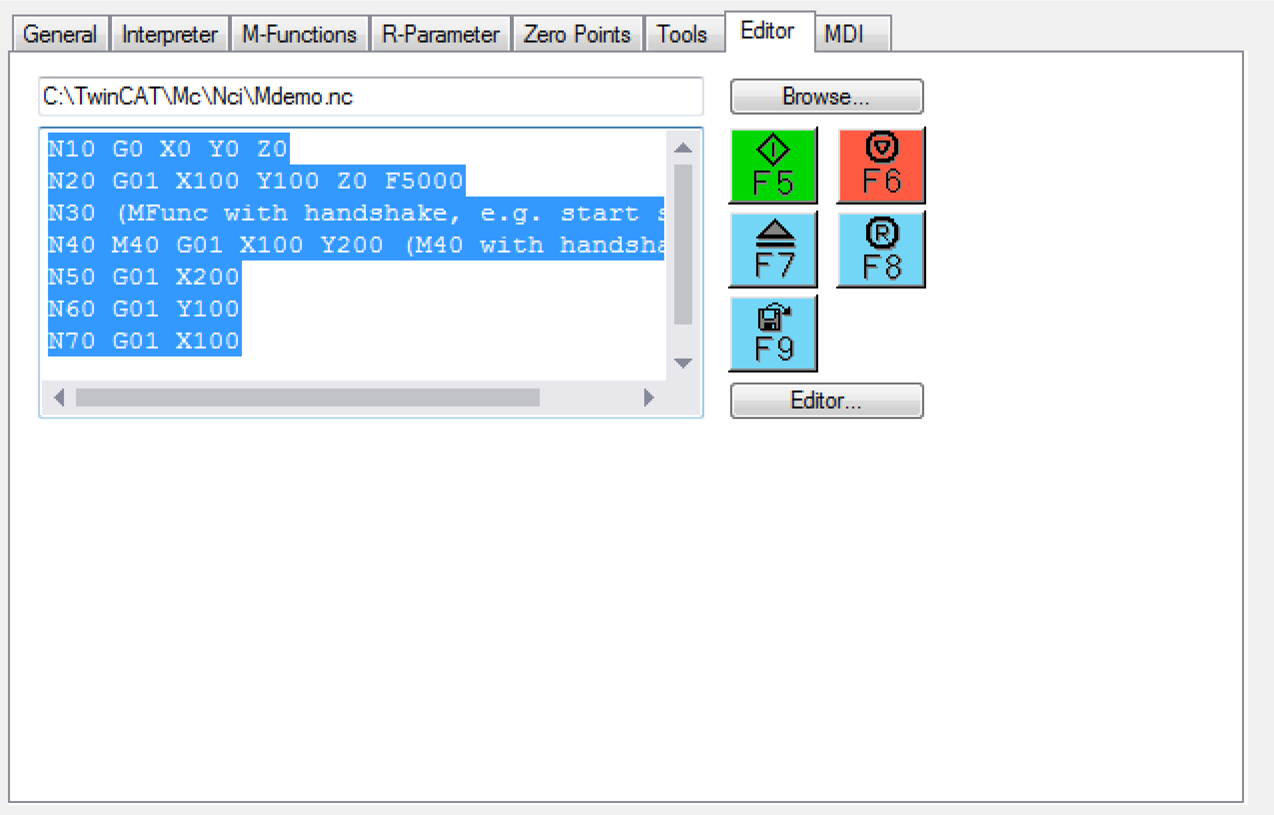
2.3.6 "Tools" tab



You can edit the data for the tool compensation on the "Tools" property page.

More detailed parameter descriptions can be found in the interpreter description under tool compensations [▸ 162].

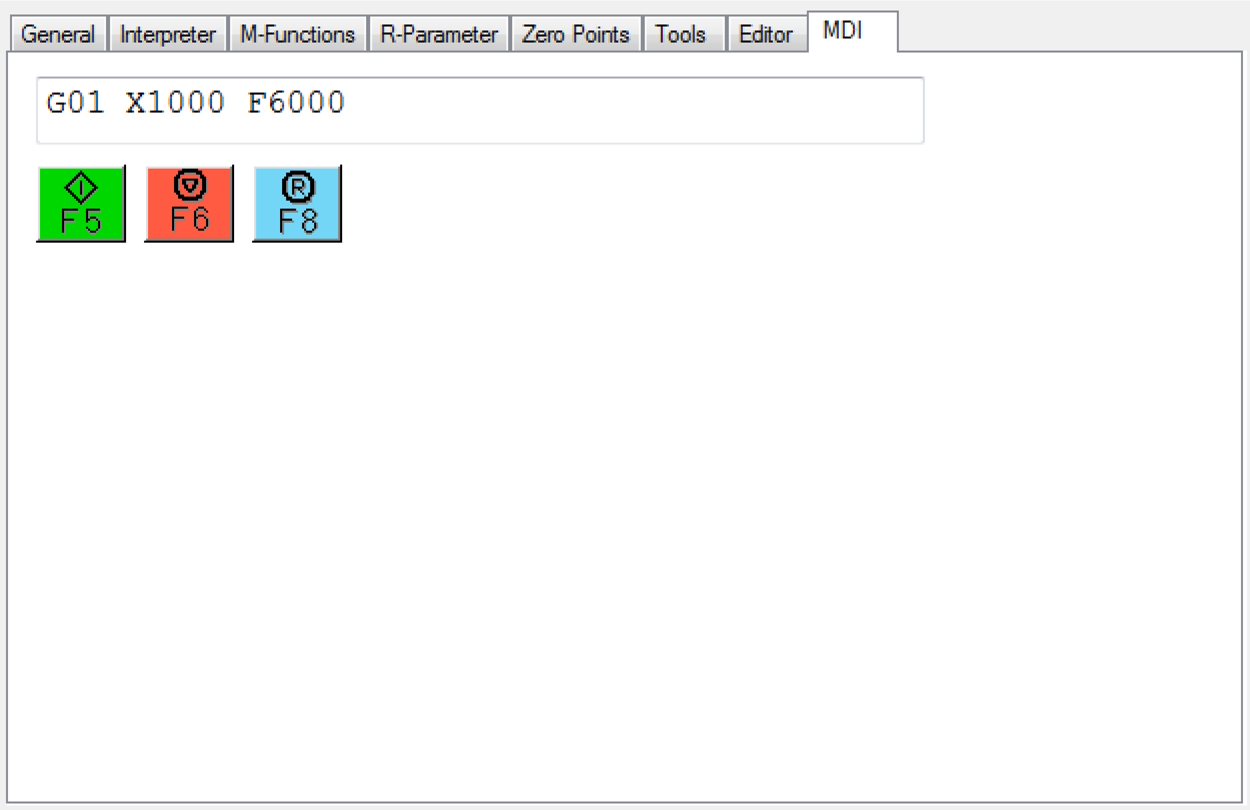
2.3.7 "Editor" tab



The editor is used to display and edit the NC programs.

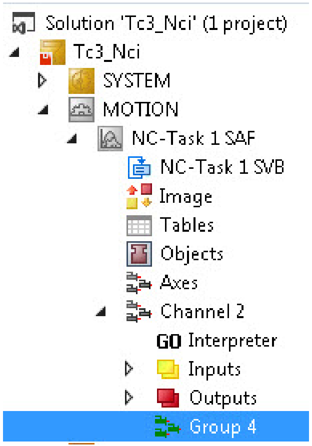
- **Browse...**
  Opens a dialog with which existing NC programs can be selected and displayed.
- **F5**
  Starts the currently loaded NC program
  NOTE! The NC program displayed in the editor need not be the currently loaded program
- **F6**
  Stops the currently running NC program
- **F7**
  Loads the NC program displayed in the editor
- **F8**
  Executes a group reset
- **F9**
  Saves the NC program currently displayed in the editor under the same name
- **Editor...**
  Opens a larger window in which the NC program is displayed

2.3.8 "MDI" tab

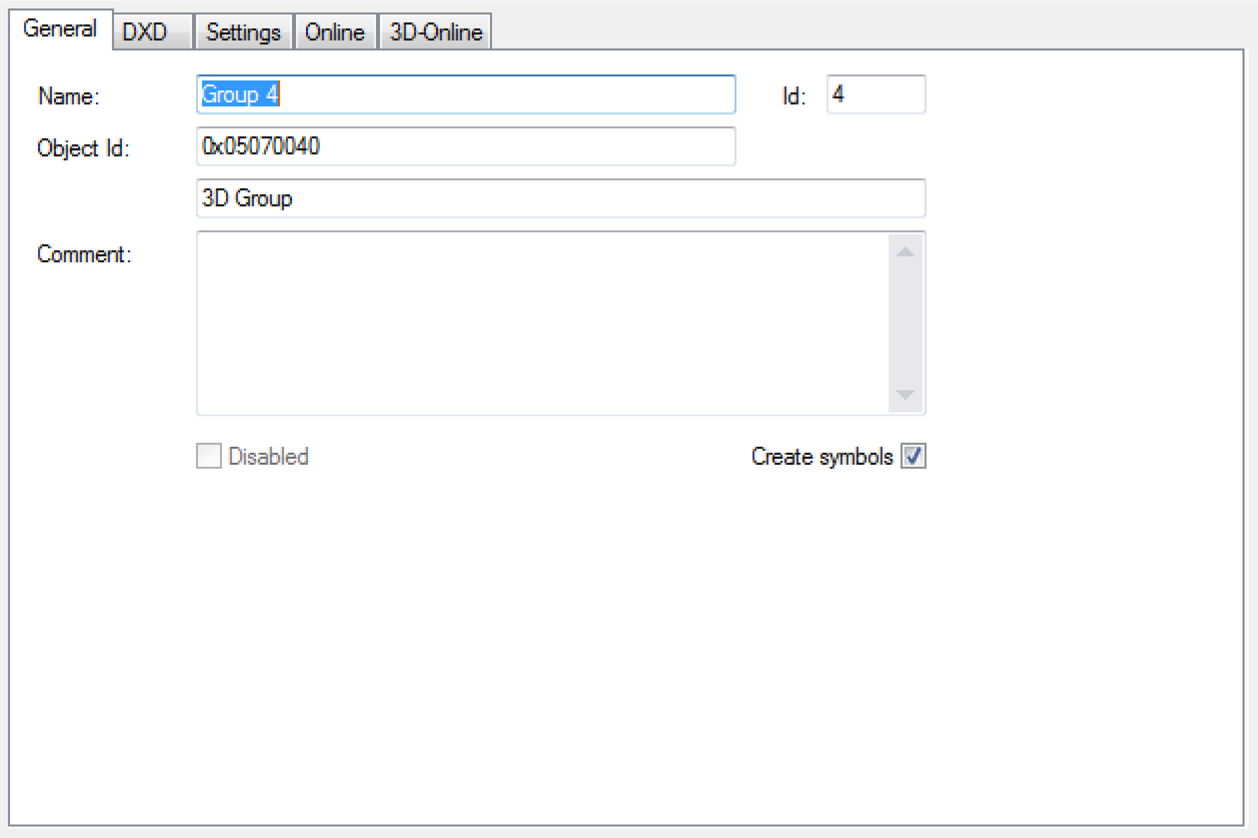


MDI stands for “Manual Data Interface”. It can be used to enter individual NC blocks directly from the TwinCAT 3 Engineering environment (XAE). Processing is started and stopped via F5 and F6 respectively.

2.4 Group element



2.4.1 "General" tab



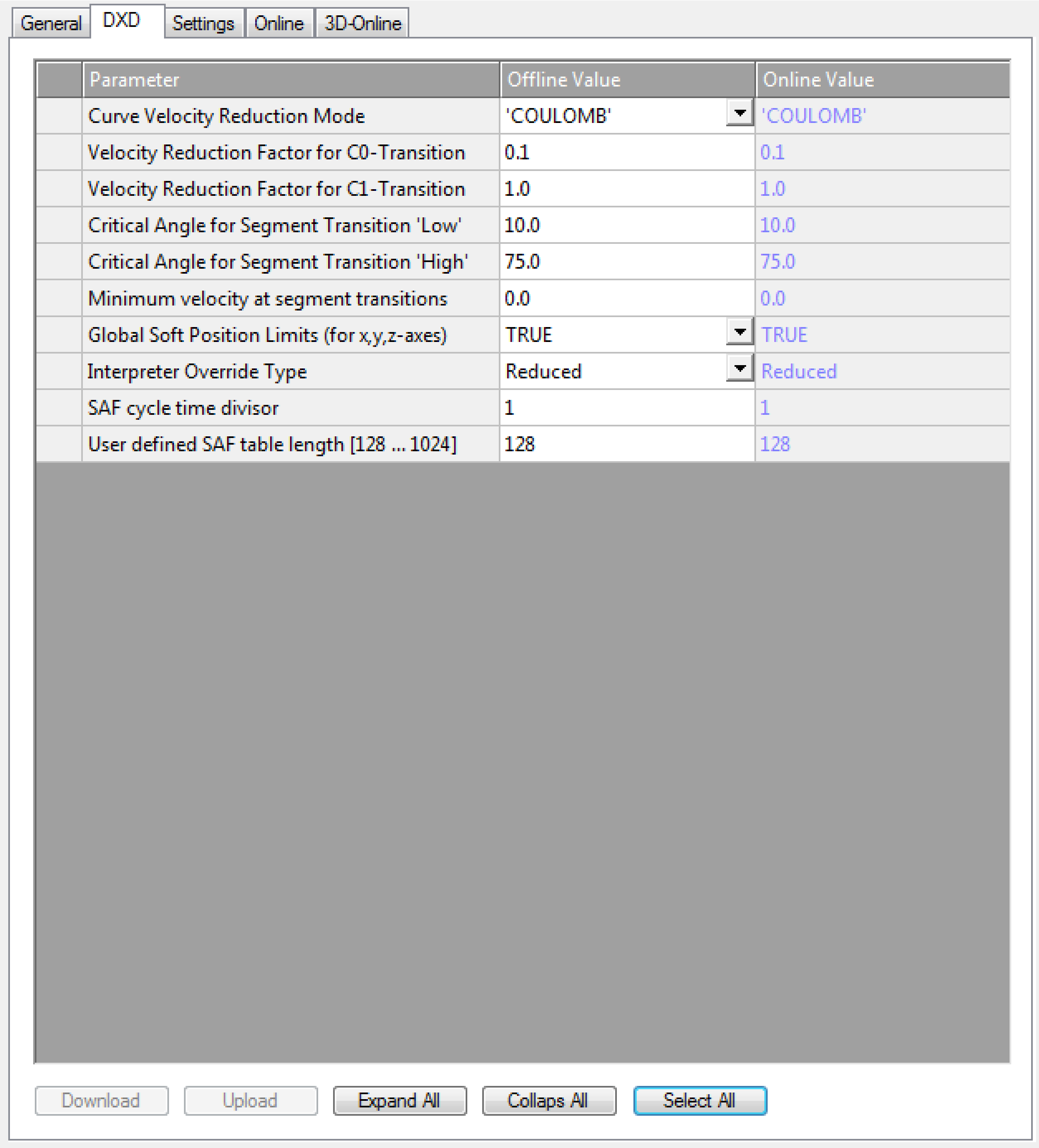
group ID

The group ID is shown on the "General" page. This is required for group-specific ADS commands.

Create symbols

In order to be able to access path variables symbolically, select symbol generation for the group here.

2.4.2 "DXD" tab



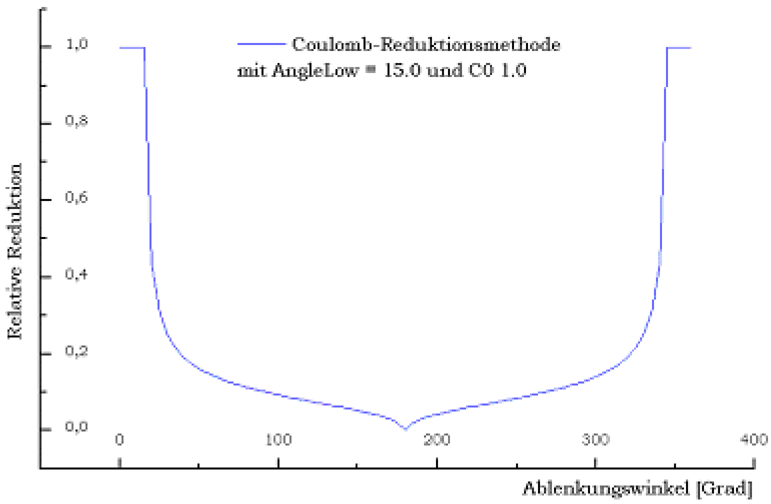
The NCI group parameters are written on the "DXD" properties page.

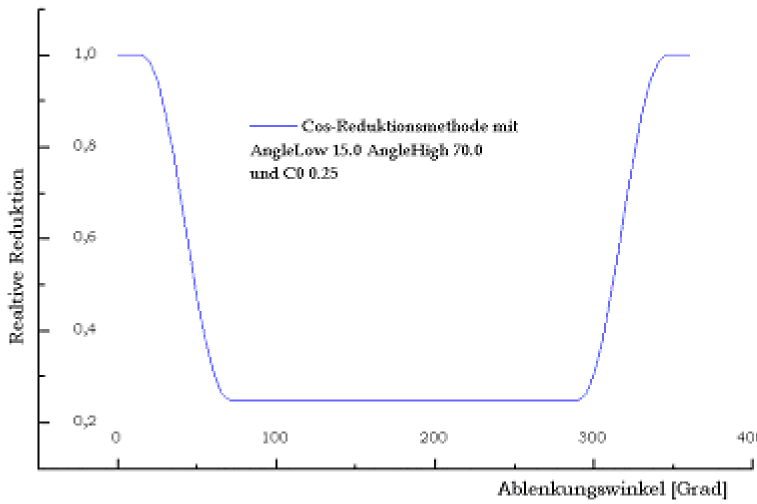
Curve velocity reduction method

The curve velocity reduction method is only effective for C0 transition (see Classification of Segment Transitions [▶ 301])

Defines of the curve velocity reduction method

```
0 Coulomb
1 Cosinus
2 VeloJump
3 DeviationAngle (not yet released)
```

Method	Description
Coulomb	The coulomb reduction method is a dynamic process analogous to the Coulomb scattering. The deflection angle φ in the transition point is the angle between the tangents of the path at the end of the segment S1 and the tangent of the path at the start of segment S2. The velocity is set to the velocity at infinity, in analogy to Coulomb scattering, $V_k \propto (\tan(0.5(\pi-\varphi)))^{1/2}$ and then reduced via the C0 factor. $V_k \leftarrow C0\ V_k$. In the case of a motion reversal (φ=180) the reduction is always V_k = C0. As the reduction in the case of small deflection angles is drastic, there is an angle $\varphi_{low} \in [0,180]$ from which full reduction takes effect. To avoid reduction, set φ_{low} = 180. For full reduction (down to φ = 0), set C0 = 0.0 and φ_{low} = 0. 

Method	Description
Cosine	The cosine reduction method is a purely geometrical process. It involves: • the C0 factor □ [0,1], • an angle φ_{low} □ [0,180], • an angle φ_{high} □ [0,180] with φ_{low} < φhigh Reduction scheme: • $_{\varphi < \varphi}$low: no reduction: $V_k \leftarrow V_{k,}$ • $_{\varphi}$high < φ: reduction by the C0 factor: $V_k \leftarrow$ C0 V_k • $\varphi_{low} < \varphi < \varphi_{high}$: partial reduction continuously interpolating between cases 1 and 2, proportional to the cos function in the range [0,π/2]. For full reduction (down to φ = 0), set C0 = 0.0 and φ_{low} = 0 and φ_{high} very small but not equal to 0 (e.g. 1.0E-10) 
VeloJump	It is a geometrical procedure for determining the segment transition velocity at a C0 transition. The procedure reduces the path velocity as required, so that the step change in velocity does not exceed the specified limit value. It is calculated based on the following formula: VeloJump factor * cycle time * min (acceleration; deceleration) Further information: [▸ 301]

Velocity reduction factor C0 transition

Reduction factor for C0 transitions. The effect depends upon the reduction method.

C0 ∈ [0.0, 1]

Velocity reduction factor C1 transition

First, *V_link* is set to the lower of the two segment target velocities:
V_link = min(*V_in,V_out*).

The geometrically induced absolute step change in acceleration *AccJump* in the segment transition is calculated depending on the geometry types *G_in* and *G_out,* and the plane selection *G_in* and *G_out* of the segments to be connected, at velocity *V_link*.

If this is greater than *C1* times the path acceleration/(absolute) deceleration *AccPathReduced* permissible for the geometries and planes, the velocity *V_link* is reduced until the resulting step change in acceleration is equal to *AccPathReduced.*

If this value is less than *V_min*, then *V_min* takes priority.

NOTE! When changing the dynamic parameters, the permissible path acceleration for the geometries and planes and thereby the reaction of the reduction changes automatically.

Reduction factor for *C1* transitions: *C1* ≥ *0.0*

Critical angle, segment transition 'low'

Parameters for φ_{low} (see curve velocity reduction method [▶ 22]).

Critical angle, segment transition 'high'

Parameters for φ_{high} (see curve velocity reduction method [▶ 22]).

Minimum velocity at segment transitions

Each NCI group has a minimum path velocity *V_min* ≥ *0.0.* The actual velocity should always exceed this value. User-specified exceptions are: programmed stop at segment transition, path end and override requests which lead to a velocity below the minimum value. A systemic exception is a motion reversal.

With the reduction method DEVIATIONANGLE the deflection angle is *φ* ≥ *φ_h*, in which case the minimum velocity is ignored. *V_min* must be less than the set value for the path velocity (F word) of each segment.

The minimum velocity can be set to a new value *V_min* ≥ *0.0* in the NC program at any time. The unit is *mm/sec*.

Global soft position limits (for x,y,z-axes)

Parameters for enabling the software end positions of the path (see: Parameterization [▶ 304]).

Interpreter override type

Parameter for selecting the path override type (see Path override (interpreter override types) [▶ 304]).

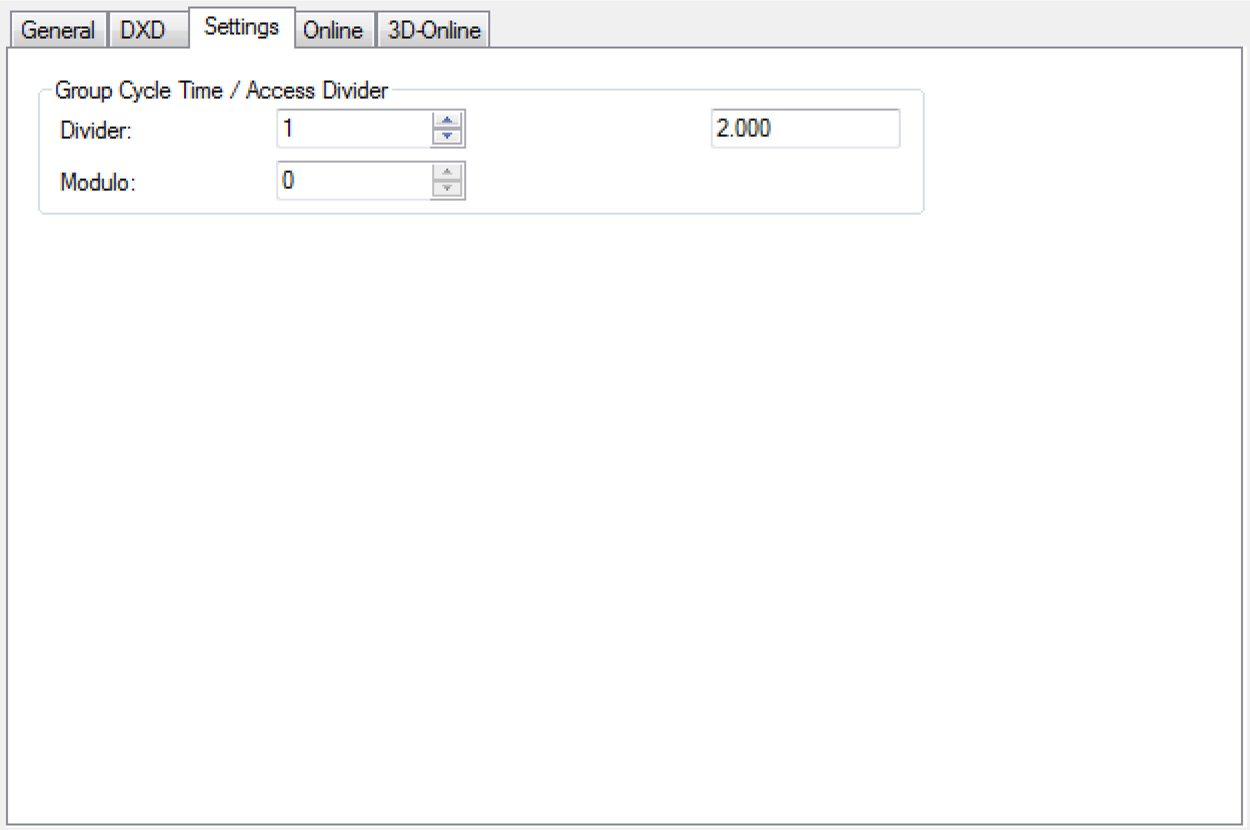
SAF cycle time divisor

The cycle time reduction ensures that the set value in the SAF is not calculated with the SAF cycle time, but with a time that is divided by the value specified here. For highly dynamic motions it may make sense to set the parameter to a value greater than 1, in order to minimize discretization inaccuracies. Increasing the SAF cycle time divisor results in the set value generator being called more frequently internally.

User-defined SAF table length

Parameter that defines the size of the SAF table and therefore the maximum number of cached SAF entries (look-ahead). If an NC program involves sequential movement of many very short segments, increasing this value can help to avoid an unintentional velocity reduction at the segment transitions.

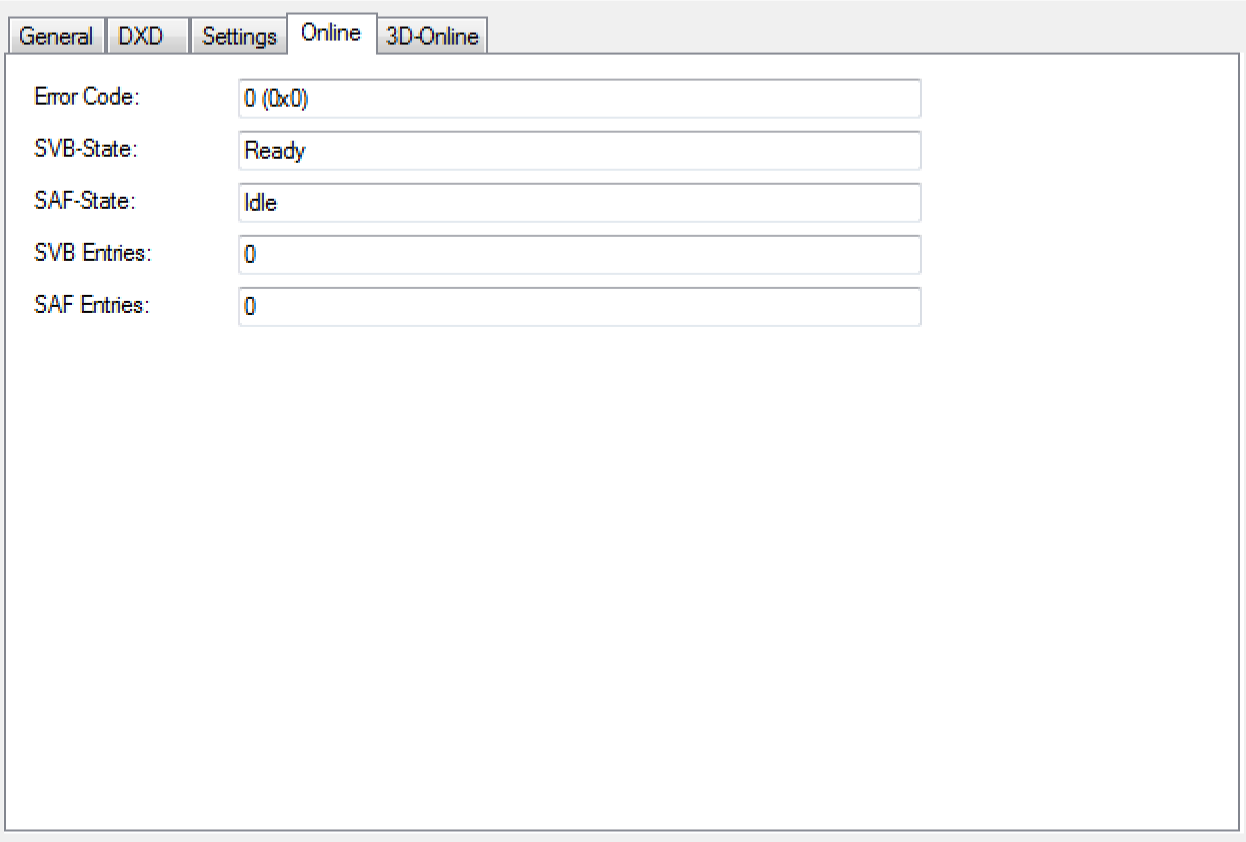
2.4.3 “Settings” tab



Under the "Settings" tab you can set the cycle time for the interpolation. The cycle time set here is a multiple of the cycle time of the SAF task.

Note	**Using the cycle time in the "Settings" tab** The cycle time setting can be used if you have to select a cycle time for the interpolation that differs from the SAF task. Generally, the cycle time of the SAF task should be adjusted to set the cycle time.

2.4.4 "Online" tab



Error code

The current error code for the channel is displayed here. The value is the same as the value displayed in the online window of the interpreter under 'channel status [▶ 11]'

SVB status

SVB status displays the current block preparation status (SVB = **S**atz**v**or**b**ereitung). Possible SVB states are:

```
ERROR
IDLE
READY
START
DRIVEOUT
CALIBRATE
MFUNC
SYNCREC
DELAY
MFUNCWAIT
SPINDLEWAIT
```

PLC evaluation of the SVB status is normally not necessary.

SAF status

SAF status displays the current block execution status (SAF = **S**atz**a**us**f**ührung). Possible SAF states are:

```
ERROR
IDLE
CONTROL
RUN
RUN_DRIVEOUT
WAIT
```

PLC evaluation of the SAF status is normally not necessary.

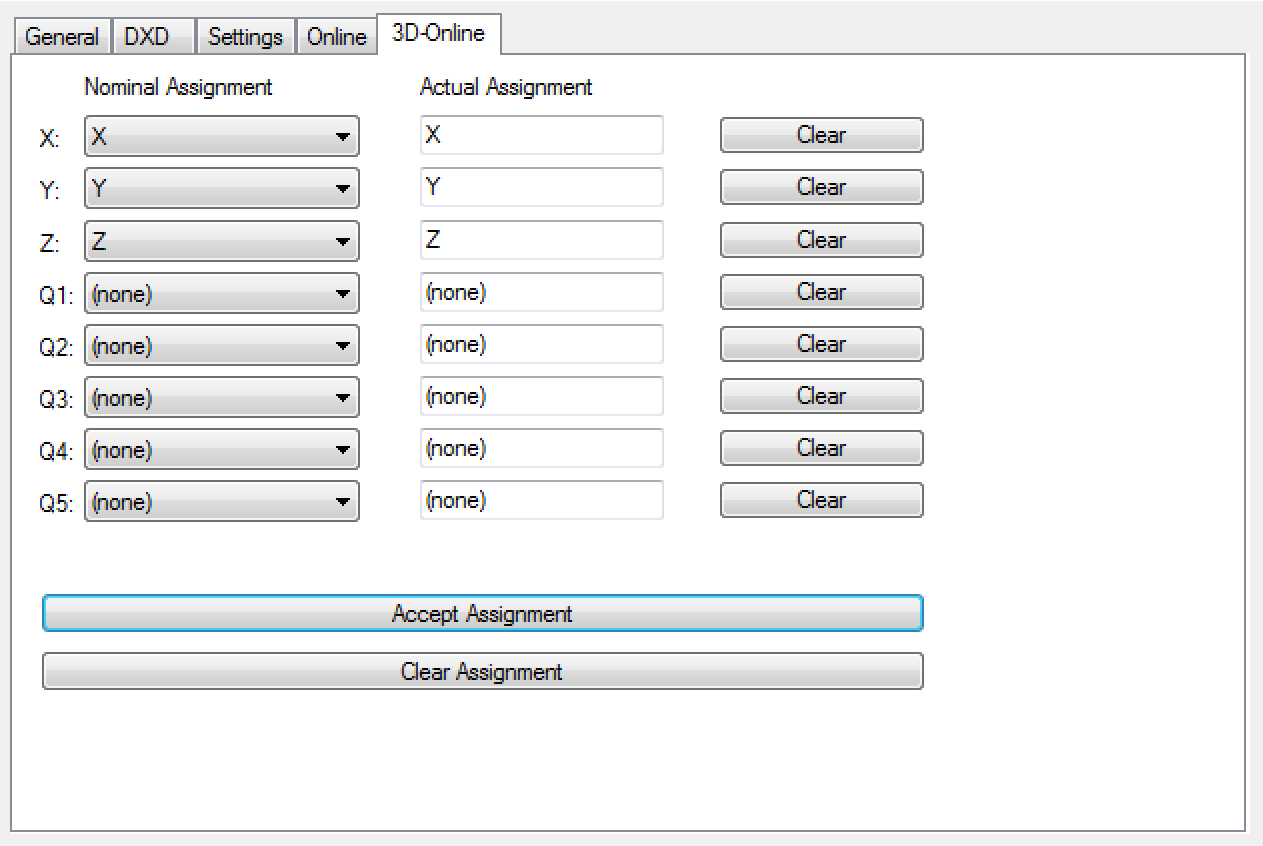
SVB entries

Number of current SVB entries.

SAF entries

Number of current SAF entries.

2.4.5 "3D-Online" tab



Target assignment

At this point the **interpolation group** is formed. The movement of the PTP axes, which are assigned to the path axes X, Y and Z, can then be based on interpolation.

Any PTP axes can be selected with the aid of the selection lists for the path axes X, Y and Z. Press the 'Apply' button to form the 3D group.

A comparably PLC function block is available in the PLC Library: Tc2_NCI [▸ 181]. (See CfgBuildExt3DGroup [▸ 182])

Actual assignment

The current path axis configurations are displayed here. Use 'Delete' to remove individual axes from the 3D group.

Delete whole configuration

Resolves the complete 3D group. Here, too, a corresponding PLC function block is available in the PLC Library: Tc2_NCI [▸ 181]. (See CfgReconfigGroup [▸ 184])

3 GST Reference Manual

Refer to

- TF5100 TC3 NC I

for the GST Documentation.

3.1 General Notes

All `GST`-examples in this documentation presuppose the following assumptions:

- Initially, the tool is located at `X0`, `Y0`, `Z0`.
- All state-variables of the interpreter are set to their default values, except that the velocity is set to a nonzero value.

3.2 Preprocessor

Include Directive

```
#include "<path>"
#include < <path> >
```

The `#include` directive inserts the contents of another file. The included file is referenced by its path. Typically, it is used to “import” commonly used code like e.g. libraries. Its behavior is similar to the `C`-Preprocessor.

Example:

In the following example file `a.gst` includes file `b.gst`. On execution of `a.gst` the interpreter internally replaces the include-line by the text of `b.gst`. Therefore, executing the program `a.gst` has the same effect as executing the program `c.gst`.

FILE a.gst:

```
G01 X0 Y0
#include "b.gst"
G01 X0 Y100
```

FILE b.gst:

```
G01 Z-2
G01 X100
G01 Z2
```

FILE c.gst:

```
G01 X0 Y0
G01 Z-2
G01 X100
G01 Z2
G01 X0 Y100
```

- If `path` is absolute, it is directly used to locate the included file. An absolute path must be surrounded by quotation marks.
- If `path` is relative and surrounded by quotation marks, it is appended to the directory of the including file to form the path of the included file.
- If `path` is enclosed in angle brackets, it is regarded to be relative to the paths in the `searchpath` list. The first entry in this list that leads to an existing file is used for inclusion. The `searchpath` list is supplied by the interpreter environment of the interpreter.

Example:

The following example assumes that the `searchpath` is set to the directories `c:\jjj` and `c:\kkk`. The file `aaa.gst` consists of a sequence of `#include`-directives that are explained in the following.

- The file `bbb.gst` is included using an absolute path. Therefore, its location is independent of the location of `aaa.gst`. Absolute referencing is useful for files that always reside at a fixed location on the filesystem.
- The file `ccc.gst` is referenced relative. It must reside in the directory of `aaa.gst` (the including file), which is `c:\mmm\`.
- The file `ddd.gst` is also referenced relative. It is expected to reside at `c:\mmm\ooo\ddd.gst`.
- The relative reference of `eee.gst` uses the sequence `'..'`, which refers to the parent directory. Therefore, the file `eee.gst` is expected in `c:\ppp\qqq\eee.gst`.
- The relative path of `fff.gst` is denoted in angle brackets. Therefore, the directories in the `searchpath` are considered, rather than the directory of `aaa.gst`. The file is expected in `c:\jjj\fff.gst` or `c:\kkk\fff.gst`. The first path that leads to an existing file is considered. If there is no file `fff.gst` in any directory of the `searchpath`, an error is reported.
- Finally, the file `ggg.gst` is expected in `c:\rrr\ggg.gst`. Both entries in the `searchpath` lead to this location.

FILE c:\mmm\aaa.gst:

```
#include "c:\nnn\bbb.gst"
#include "ccc.gst"
#include "ooo\ddd.gst"
#include "..\ppp\qqq\eee.gst"
#include <fff.gst>
#include <../rrr/ggg.gst>
```

- Each include-directive must be denoted on a dedicated line. Then, this entire line is replaced by the contents of the included file. An additional ‘newline’ character is appended.
- The include-directive may be used multiple times at arbitrary locations of the including file.
- The same file may be included multiple times.

Note	**TIP:** It is typically bad practice to include a file multiple times. Especially, if this feature is misused to factor out code. Instead, a function should be preferred to define code that is reused multiple times (See section Userdefined Functions [▶ 54].)

- If an included file does not exist, an error is reported.
- Include directives in included files are also subject to replacement.
- An infinite loop due to recursive inclusion (e.g. `A` includes `B`, `B` includes `C` and `C` includes `A`) is detected and reported as an error.

Example:

In the following example file `a.gst` includes file `b.gst` twice. The second inclusion is always expanded, independently of the enclosing condition by the `IF-THEN` expression. The included file `b.gst` itself includes file `c.gst`.

FILE a.gst:

```
G01 X100
#include "b.gst"
G01 Y100
! IF stVariable=47 THEN
#include "b.gst"
! END_IF;
```

FILE b.gst:

```
#include "c.gst"
G01 X0 Y0
```

FILE c.gst:

```
G01 Z0
```

Example:

File `x.gst` demonstrates a series of invalid include directives. The first three lines violate the rule that each include directive must be denoted on a dedicated line. In lines 4 and 5 the filename is not properly enclosed in quotation marks or angle brackets. In line 6 a nonexisting file is included. Line 7 includes the file `y.gst`, which itself includes file `x.gst`. This loop is reported as an error.

FILE x.gst:

```
#include "a.gst" G01 X100
! #include "a.gst"
#include "a.gst"    #include "b.gst"
#include a.gst
#include "a.gst>
#include "non_existing_file.gst"
#include y.gst
```

FILE y.gst:

```
#include "x.gst"
```

3.3 Combining G-Code and ST

A GST-Program

```
<g-code>
<g-code>
! <st-code>
<g-code>
<g-code>
{
<st-code>
<st-code>
! <g-code>
<st-code>
<st-code>
}
<g-code>
<g-code>
```

A GST-file consists of sequences of `G`-code and sequences of `ST`-code that can be interleaved as shown above. Each program starts in `G`-code mode. The mode can be switched to `ST` for one line using an exclamation mark (‘!’). The `ST`-mode ends at the end of line automatically.

As an alternative a block of `ST`-code can be defined using curly braces (‘{‘…’}’). This notation is more practical to define a long sequence of `ST`-code in a GST-program. Within the `ST`-block the `G`-code mode can be entered for one line using the exclamation mark. Thereby, the `G`-code mode ends at the end of line automatically.

G-Code Block

```
<address><value> <address>=<G-Expression> <address>{<ST-Expression>}
```

A line of `G`-code is called a **block**. It consists of a sequence of **words**. A word is a combination of an **address** (e.g. `G` or `X`) and a **value**. A value can be defined by a literal (e.g. `2.54`), by a `G`-expression (e.g. `2*foo+1`) or by an `ST`-expression (e.g. `sin(foo**2)-1`).

G-Code Expression

```
<address>=a+b-c*d/e
```

The result of the expression is used as the value of the word. The four basic arithmetic operations ('+', '–', '*', '/') can be used in a G-expression. They are evaluated as expected, i.e. all operations are left-associative and '*', '/' have a higher precedence than '+', '–'. Variables that have been declared in ST can also be used in a G-expression (with respect to their scope).

All computations are performed using type LReal (64-bit floating point according to IEEE 754). The value of an ST-variable is implicitly converted to type LReal according to the conversion rules of ST. If a type (e.g. STRING) cannot be converted, an error is reported.

Note	**RESTRICTION:** ST-variables that contain a number in their name (e.g. x0) cannot be used in a G-expression to avoid confusion with a G-Code like X0. This limitation does not apply to ST-expressions.

Note	**RESTRICTION:** Array variables, struct variables and objects cannot be used in a G-expression. This limitation does not apply to ST-expressions.

Note	**RESTRICTION:** Parentheses are not allowed in a G-expression as they are used to denote comments in G-Code. For the same reason function calls are not available. These limitations do not apply to ST-expressions.

Embedded ST-Expression

<address>{<ST-Expression>}

The result of the ST-expression is used as the value of the word. It must be convertible to LReal. Basically, an ST-expression is ST-Code that could be placed on the right hand side of an assignment. Other ST-Code (e.g. an ST-statement) is not allowed. However, extensive computations can be encapsulated in an ST-function that is then called in the ST-expression.

Note	**TIP:** An ST-expression should not have side effects, since the evaluation order of ST-expressions is generally undefined and may change in the future. Besides, this style of programming employing side effects is a bad programming style. For instance, an ST-expression should not call a function that contains G-Code.

Example:

- The following GST-program starts with a line of G-code that moves the tool rapidly to the origin.
- The line is followed by a line of ST-code that declares variable 'i'. The ST-mode is entered by the prefixed exclamation mark ('!'). After this line G-code mode resumes.
- The G-code in line 3 moves the tool down.
- Lines 4 to 8 define a block of ST-code that contains a FOR-loop. The code in this block is interpreted as ST-code, except for the G-code line in line 6. This line of G-code uses a G-expression to set the X-axis to 10*i. The value of the Y-axis is defined using an ST-expression that is enclosed in curly braces. This expression evaluates to 0 if 'i' is even and to 10 otherwise.
- The programmed path of the program is shown in Figure "ExampleExpressions".

```
G00 X0 Y0 Z0
! VAR   i : INT;   END_VAR
G01 Z-1
{
FOR i := 1 TO 5 DO
!G01 X=i*10 Y{ (i MOD 2) *10 }
END_FOR;
}
```

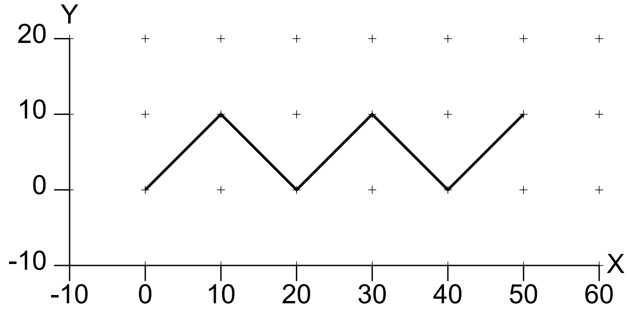


Figure “ExampleExpressions”.

Suppression of G-Code Blocks

/<n> <G-Code block>

The execution of a G-Code block can be suppressed conditionally. If ‘/<n>’ is prefixed and the n-th bit in an internal disable mask is set, the block is suppressed (not executed). The disable mask can be set by the PLC and by the ST-function `disableMaskSet`. If n is omitted, it has 0 value by default. [See section Suppression of G-Code Blocks [▶ 74].]

3.4 G-Code (DIN 66025)

3.4.1 Comments

DIN 66025 Comment

<g-code> (<comment>) <g-code>

Text that is enclosed in round parentheses is treated as comment in G-Code (according to DIN 66025). The comment must not include further parentheses.

Example:

The following example demonstrates the notation of comments in G-Code.

```
N10 G01 (activate linear interpolation) X10 (set X-coordinate to
10)
(the next block results in a semicircle with center point
X10 Y10)
N20 G02 (activate clockwise interpolation) Y20 U10 (radius is 10)
```

Line Comment

<g-code> // <comment>

Text between ‘//’ and the end of line is treated as a comment in G-Code.

Example:

The following example demonstrates the notation of line comments in G-Code.

```
N10 G01 X10 // perform a linear movement to X10 Y0
// the next block results in a semicircle with center point X10 Y10
N20 G02 Y20 U10
```

3.4.2 Codes

D

D<v>

Select tool v. The new tool applies to its own block and all succeeding blocks until a new tool is selected. Tool 0 is special. Its selection deactivates any tool compensation. Tool 0 can be regarded as tool where all tool parameters are set to zero. It is selected by default.

Example:

In the following example tool 1 is defined to have a Y-offset of 10 and tool 2 to have an Y-offset of 20. Block N10 and block N50 use tool 0. Tool 1 applies to block N20 and to block N30. In block N40 tool 2 is active. Figure “ExampleD” shows the resulting programmed path (dotted line) and the resulting tool center point path (solid line).

```
!toolSet(index:=1, nr:=1, offsetY:=10);
!toolSet(index:=2, nr:=2, offsetY:=20);
N10 G01 X10 Y0 F6000
N20 G01 X20 Y0 D1
N30 G01 X30 Y0
N40 G01 X40 Y0 D2
N50 G01 X50 Y0 D0
```

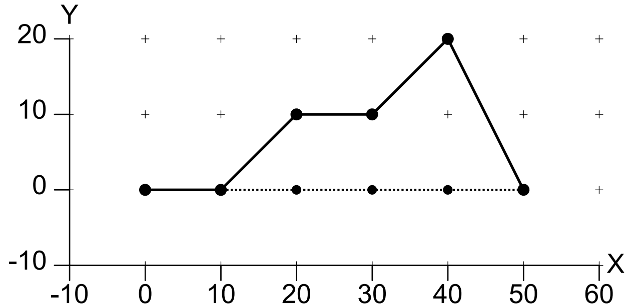


Figure “ExampleD”.

F

F<v>

Set velocity to v. Applies to the current block and all succeeding blocks until a new velocity is programmed. The unit for velocity selected currently is used. (See section unitVelocitySet [▶ 77] for details.) The default velocity is 0.

Note

NOTE:

The velocity must be set to a nonzero value before a movement is programmed. Otherwise, an error is issued.

Example:

The first two segments N10 and N20 are processed with a velocity of 6000 mm/min, and the last segment N30 is processed with a velocity of 3000 mm/min.

```
N10 G01 X100 F6000
N20 G01 X200
N30 G01 X300 F3000
```

G00

Set the interpolation mode to “rapid, linear”. The interpolation mode applies to this block and all succeeding blocks until it is reset by `G01`, `G02` or `G03`. `G00` is the default interpolation mode.

If `G00` is active, programming of a point (see `X`) will result in a linear geometry segment that is processed with maximum velocity. The programmed velocity is not considered. `G00` is typically used to position the tool. For machining `G01` should be used, which considers the programmed velocity.

Note	**NOTE:** `G01`, `G02`, `G03`, `G04`, `G58` and `G59` are mutually exclusive. They must not be programmed in a common block.

Example:

The resulting path of the following example is shown in Figure “ExampleG00”. The first block `N10` rapidly moves the tool to position `X20`, `Y10`, `Z30`. The resulting geometry segment is a line in space. The orientation remains unchanged. The second block `N20` performs a rapid movement to `X50`, `Y10`, `Z30`. There is no need to denote `G00` in this line, since interpolation is modal.

```
N10 G00 X20 Y10 Z30
N20 X50
```

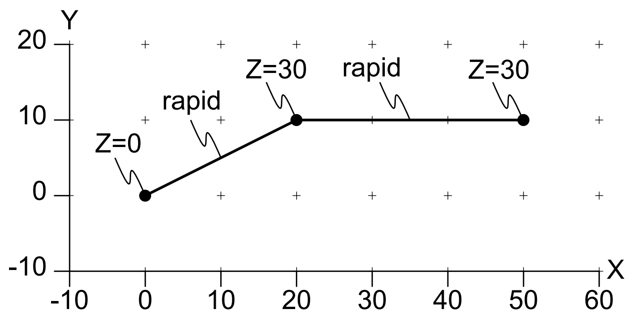


Figure “ExampleG00”.

G01

Set the interpolation mode to “linear”. This interpolation mode is like `G00`, except that the path is machined with the programmed velocity. (See `F`.) The interpolation mode applies to this block and all succeeding blocks until it is reset by `G00`, `G02` or `G03`.

G02

Set the interpolation mode to “circular/helical, clockwise”. The interpolation mode applies to this block and all succeeding blocks until it is reset by `G00`, `G01` or `G03`. If `G02` is active, programming of a point will result in a circular (or helical) arc that is machined with the current velocity. (See F [▸ 34].) In the following, a circular arc is regarded. The helical arc is covered later.

A circular arc starts at the current point and ends at the programmed point. It rotates around the working-plane normal (PCS, i.e. program coordinate system) in the center point. The center point can be defined using Centerpoint Programming [▸ 35] or using Radius Programming [▸ 36].

Centerpoint Programming

For Centerpoint Programming the center is defined relative to the starting-point using the `I,J,K` parameters. The center point is the sum of the starting-point and the vector `[I,J,K]`. The `I,J,K` parameters are optional and have `0` value by default. If the starting-point and the endpoint are equal with respect to the workingplane, a full circle will be emitted.

Note	**CONSTRAINT:** The radius at the starting-point and at the endpoint must be equal. However, small deviations are allowed and corrected automatically. (See centerpointCorrectionSet [▶ 69].)

Note	**CONSTRAINT:** The center point must not be equal to the starting-point or endpoint.

Radius Programming

For Radius Programming the center point is derived from the radius that is given by the `U` parameter. Typically, there are two arcs of a given radius that lead from the starting-point to the endpoint. If the radius is positive, the shorter one is used, otherwise the longer one is chosen. Apart from that, the absolute value of the radius is regarded by the interpreter.

Note	**CONSTRAINT:** Radius Programming can by its nature not be used to program a full circle. This curvature can be programmed by Centerpoint Programming.

Note	**CONSTRAINT:** The radius must not be zero.

Note	**CONSTRAINT:** The radius must not be smaller than half of the distance between starting-point and endpoint with respect to the workingplane.

Helical

If the starting-point and endpoint do not lie in a plane that is parallel to the workingplane, a *helical movement* is performed.

Note	**TIP: moveCircle3D** The `ST`-function `moveCircle3D` is a more powerful way to define a circle or helix. It covers 3D-arcs and multiturn circles.

Example:

The following example results in the path that is shown in Figure “ExampleG00G02”. The block `N10` uses Radius Programming to define a clockwise arc from `X0 Y0` to `X10 Y10` with radius `10`. Because the radius is positive, the center point `c1` of the shorter arc is chosen. In block `N30` the center point `c2` of the longer arc is used because the radius is negative. The block `N50` uses Centerpoint Programming, where the center `c3=[60,0,0]` is the sum of the starting-point `[50,0,0]` and `[I,J,K]=[10,0,0]`. The block `N70` defines a full circle with center point `C04` because the starting-point and endpoint are equal. The block `N90` defines a helical arc with center point `C05` and height `30` (in `Z`-direction).

```
N10 G02 X10 Y10 U10
N20 G00 X30 Y0
N30 G02 X40 Y10 U-10
N40 G00 X50 Y0
N50 G02 X60 Y10 I10
N60 G00 X80 Y0
N70 G02 J10
N80 G00 X110 Y0
N90 G02 J10 X120 Y10 Z30
```

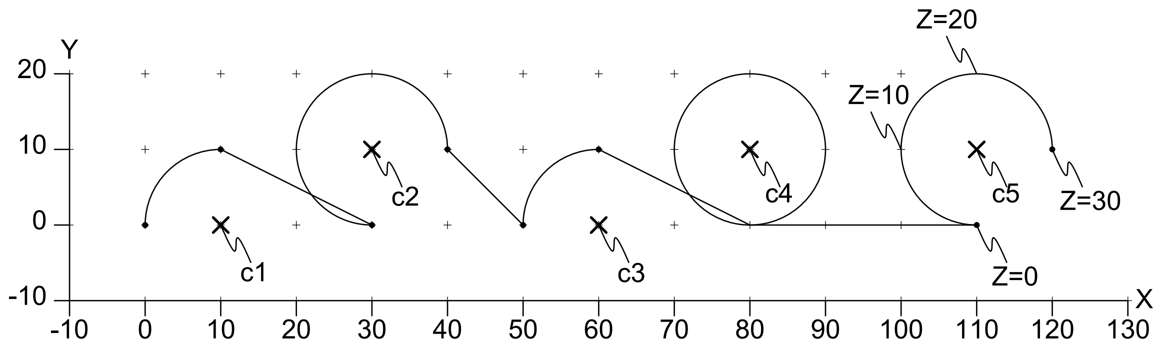


Figure “ExampleG00G02”.

G03

Set the interpolation mode to “circular/ helical, counterclockwise”. This interpolation behaves similar to `G02`. The interpolation mode applies to this block and all succeeding blocks until it is reset by `G00`, `G01` or `G02`.

G04

Suspend machining for a given duration. The duration is defined by either `X` or `F` in the current time unit. (See `unit` for details.)

Example:

The following example assumes that the current time unit is set to seconds. On one execution of the program the machine moves to `X10`, waits for `1.5` seconds and then moves to `X20`.

```
N10 G01 X10
N20 G04 F1.5
N30 G01 X20
```

G09

Accurate Stop - Nonmodal.

`G09` evokes an Accurate Stop.
`G09` is nonmodal.

G17

Select `XY`-plane as workingplane, i.e. the workingplane normal is set to `[0,0,1]`. See workingplaneSet(…) for details. This workingplane is the default workingplane.

G18

Select `ZX`-plane as workingplane, i.e. the workingplane normal is set to `[0,1,0]`. See workingplaneSet(…) for details.

G19

Select `YZ`-plane as workingplane, i.e. the workingplane normal is set to `[1,0,0]`. See GST_WorkingplaneSet for details.

G40

Deactivate Tool Radius Compensation (TRC). `G40` is equivalent to an invocation of `trcSet(…)` where the normal vector is set to the zero vector. See trcSet(…) for details.

G41

Activate tool radius compensation (TRC). After activation the programmed path is shifted left by the radius of the currently selected tool. (See D [▶ 34].)

More precisely, `G41` is equivalent to an invocation of `trcSet(…)`, where

- *normal* is set to the negated normal of the current working-plane, multiplied by the radius of the currently selected tool,
- *offset* is set to the value defined by `trcOffsetSet(…)`,
- *radius* and *angle* for approach and depart are set to the values defined by `trcApproachDepartSet(…)`,
- *limit* is set to the value defined by `trcLimitSet(…)`.

See trcSet(…) for details.

Note	NOTE: On activation, a tool with a nonzero index must be selected.

Example:

The following example demonstrates the activation and deactivation of tool radius compensation. The programmed path (dotted line) and the compensated path (solid/ dashed line) are shown in Figure “ExampleG40G41”.

- The first line of the GST program sets the `offset` parameter to `5 mm`. Therefore, the adjacent segments of a gap are extended by `5 mm`. The remaining gap is closed by a circular arc.
- The second line defines the approach and depart behavior to use a circular arc with a radius of `5 mm` and an angle of `90` degree.
- The third line defines tool `1` to have a radius of `10`.
- Block `N10` describes a linear movement to `[10,0,0]`.
- The next block `N20` selects tool `1` and activates tool radius compensation, where `D1` comes into effect before `G40` is processed and `G40` is active before `X20` is processed. Therefore, the end of segment `N20` is subject to TRC (tool radius compensation). The linear movement from the end of segment `N10` to the end of segment `N20` in the programmed path is substituted by an approach-segment (dotted line) from the end of `N10` to the end of `N20'` in the compensated path.
- In the next three lines a linear movement along `N30`, `N40` and `N50` is programmed. Since segment `N40` would result in a collision, it is eliminated from the compensated path.
- In the next line a circular arc along `N60` is programmed. The gap between the end of `N50'` and the beginning of `N60'` is closed as described earlier.
- The line along `N70` is the last segment that is subject to TRC (tool radius compensation), since its deactivation becomes active before the end of `N80`. The line along `N80` is replaced by the depart-segment `N80'`, similarly to the approach-segment.

```
!trcOffsetSet(offset:=5);
!trcApproachDepartSet(approachRadius:=5, approachAngle:=90, departRadius:=5, departAngle:=90);
!toolSet(index:=1, tooltype:=tooltypeMill, radius:=10);
N10 G01 X10
N20 X20 G41 D1
N30 X35
N40 X40
N50 Y20
N60 G02 X50 Y10 U10
N70 G01 X70
N80 X80 Y0 G40
N90 X90
```

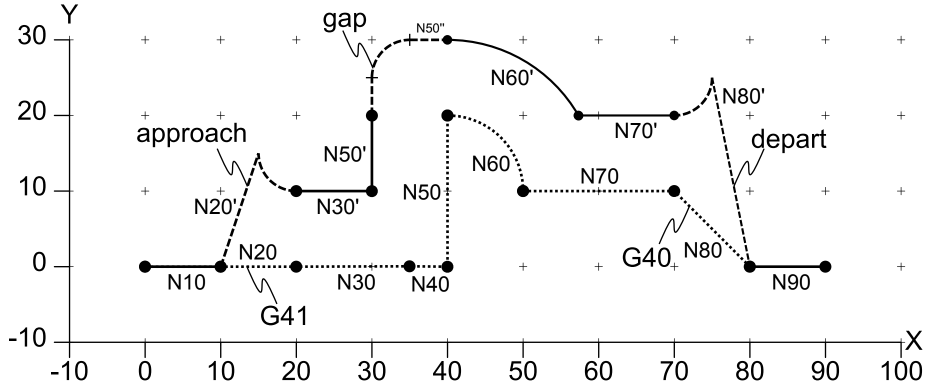


Figure “ExampleG40G41”.

G42

This function is the same as `G41`, except that the path is shifted to the right. The invocation of `trcSet(…)` is the same, except that the workingplane-normal is not negated. See G41 [▸ 38] and trcSet(…) for details.

G53

Deactivate any zero offset shift translation. This adjustment is the default. The deactivation becomes active also for the current block. See sections Zero Offset Shift [▸ 75] and G58, G59 [▸ 39] for details.

G54..G57

Activates the translation that is associated with the given `G`-Code (`TZ54…TZ57`). Also activates the translations of `G58` and `G59`. The translations apply to the current block and all succeeding blocks until changed. See section Zero Offset Shift [▸ 75] for details.

G58, G59

Set the translation that is associated with the given `G`-Code. The new translation value is given by the parameters `X,Y,Z`, which are mandatory. By default, the associated translations are zero. See section Zero Offset Shift [▸ 75] for details.

Example:

The resulting MCS (machine coordinate system) path and the applied translations of this example are shown in Figure “ExampleG54G58G59”.

- The first line sets the translation that is associated with `G54` to `[0,5,0]`.
- The next line sets the programmed translation of `G58` to `[0,10,0]`. Since zero-offset-shifts are still disabled (default `G53`), the PCS (program coordinate system) and MCS (machine coordinate system) match.
- Accordingly, the block `N20` results in a linear movement from MCS (machine coordinate system) coordinate `[0,0,0]` to `[20,0,0]`.
- The next line activates `G54` and programs a linear movement along `N30`, whereby `G54` becomes active before the movement. The programmed PCS (program coordinate system) coordinate `[40,0,0]` is mapped to the MCS (machine coordinate system) coordinate `[40,15,0]`.
- The next line sets the programmed transformation `G59` to `[0,5,0]`. Thereby, the effective translation changes from `[0,15,0]` to `[0,20,0]`. Since the current MCS (machine coordinate system) coordinate must not be affected by this change, the current PCS (program coordinate system) coordinate is set to `[40,-5,0]`, implicitly.
- The succeeding `ST`-function `frameGet` stores these coordinates in `[pcsX,pcsY,pcsZ]`.

- The next line merely programs the `X`-coordinate of the end of segment `N50`. Therefore, the PCS (program coordinate system) coordinate of the end of segment `N50` is `[60,-5,0]`, which is mapped to the MCS (machine coordinate system) coordinate `[60,15,0]`. In other words: The translation `G59` is active, but does not become apparent due to the adaption of the current PCS (program coordinate system) coordinate. (See section Applying Transformations [▶ 82] for details.)
- It becomes apparent by the last line, which sets the PCS (program coordinate system) coordinate of the end of segment `N60` to `[80,0,0]`. This coordinate is mapped to the MCS (machine coordinate system) coordinate `[80,20,0]`.

```
!zeroOffsetShiftSet(g:=54, x:=0, y:=5, z:=0);
N10 G58 X0 Y10 Z0
N20 G01 X20 Y0
N30 G54 X40 Y0
N40 G59 X0 Y5 Z0
!VAR pcsX, pcsY, pcsZ : LREAL; END_VAR
!frameGet(x=>pcsX, y=>pcsY, z=>pcsZ);
N50 X60
N60 X80 Y0
```

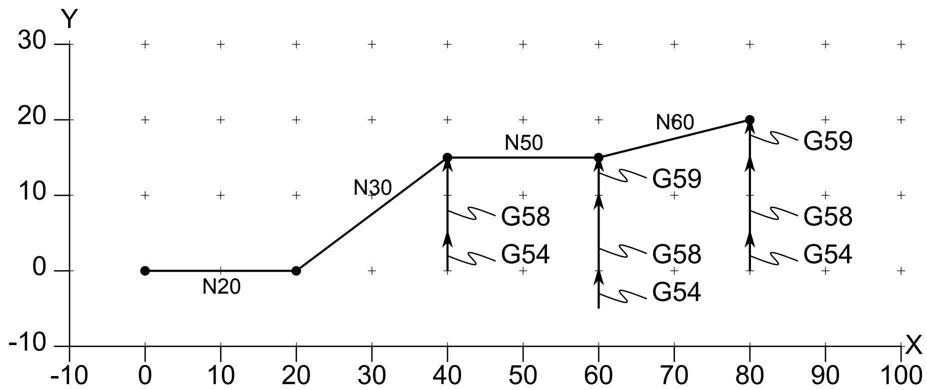


Figure “ExampleG54G58G59”.

G60

Accurate Stop - Modal.

`G60` evokes an Accurate Stop.
`G60` is modal. `G00` calls off `G60`.

G70

Set the unit for lengths to *inch*. The new unit also applies to the current block. `G70` is equivalent to the call `unitLengthSet(unitLengthInch)`. The unit for velocity is not affected. See UnitLength [▶ 77] and G71 [▶ 40] for details.

G71

Set the unit for lengths to *millimeter*. The new unit also applies to the current block. `G71` is equivalent to the call `unitLengthSet(unitLengthMillimeter)`. The unit for velocity is not affected. See UnitLength [▶ 77] for details.

Example:

In Figure “ExampleG70G71” the path of the following example is shown, which uses the unit *millimeter*.

- The first line of the program sets the unit for lengths to *inch*. This unit is used in the same line to interprete `X2` in inch. Thus, the path `N10` ends at position [`50.8` mm, `0` mm, `0` mm].
- Accordingly, the next line moves the tool along `N20` towards [`50.8` mm, `25.4` mm, `0` mm].

- The last line sets the unit to *millimeter*. Therefore, the path `N30` ends at position [`80` mm, `25.4` mm, `0` mm]. Accordingly, the segment `N30` is a horizontal line.

```
N10 G01 X2 G70
N20 G01 Y1
N30 G01 X80 Y25.4 G71
```

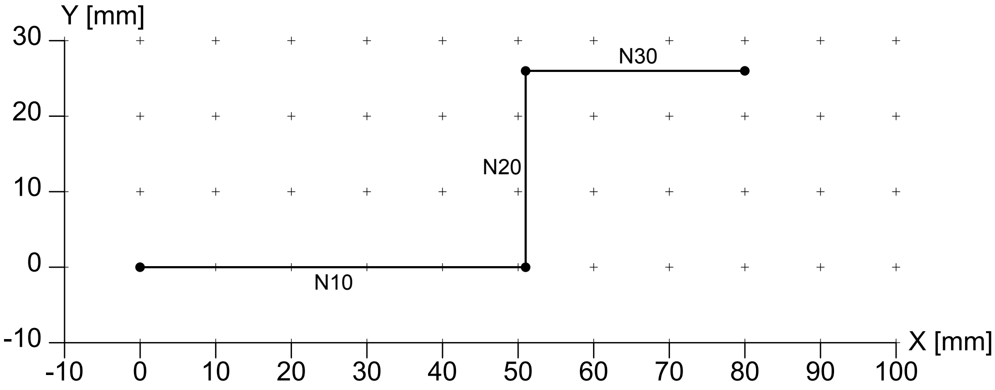


Figure “ExampleG70G71”.

G90

Switches to absolute coordinates. `X`, `Y`, `Z` are interpreted as absolute PCS (program coordinate system) coordinates. This adjustment is the default. The switch becomes active in its own block. See G91 [▶ 41] for details.

G91

Switches to relative coordinates. `X`, `Y`, `Z` are interpreted to be relative to the current point, i.e. the next point is computed as the sum of `[X,Y,Z]` and the current point. The switch has an effect for its own block.

Example:

The path of the following example is shown in Figure “ExampleG90G91”. The switch to `G90`/ `G91` takes effect immediately.

```
N10 G90 G01 X10 Y20
N20 X20 Y10
N30 G91 X10 Y10
N40 X10 Y-10
N50 G90 X50 Y20
```

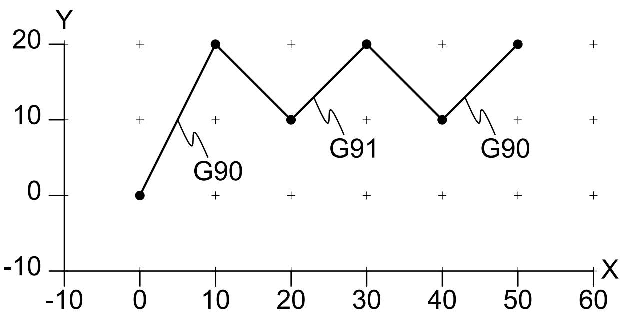


Figure “ExampleG90G91”.

G700

Like `G70`, but also applies to the interpretation of velocity. The new unit comes into effect in the current block. `G700` is equivalent to the calls `unitLengthSet(unitLengthInch)` and `unitVelocitySet(unitLengthInch,unitTimeMinute)`. See G710 [▶ 42] for an example.

G710

Like `G71`, but also applies to the interpretation of velocity. The new unit comes into effect in the current block. `G710` is equivalent to the calls `unitLengthSet(unitLengthMillimeter)` and `unitVelocitySet(unitLengthMillimeter,unitTimeMinute)`.

Example:

The path of the following example is shown in Figure “ExampleG700G710”.

- The first line defines a linear movement to `[1 in, 1 in, 0 in]` with a velocity of `100 in/min`.
- The second line sets the length unit to `mm`, but does not affect the velocity unit. It defines a movement to `[30 mm, 10 mm, 0 mm]` with a velocity of `50 in/min`.
- The last line also sets the velocity unit to `mm/min`. Therefore, there is a movement to `[40 mm, 20 mm, 0 mm]` with a velocity of `1000 mm/min`.

```
N10 G700 G01 X1 Y1 F100
N20 G71 G01 X50 Y10 F50
N30 G710 G01 X80 Y20 F1000
```

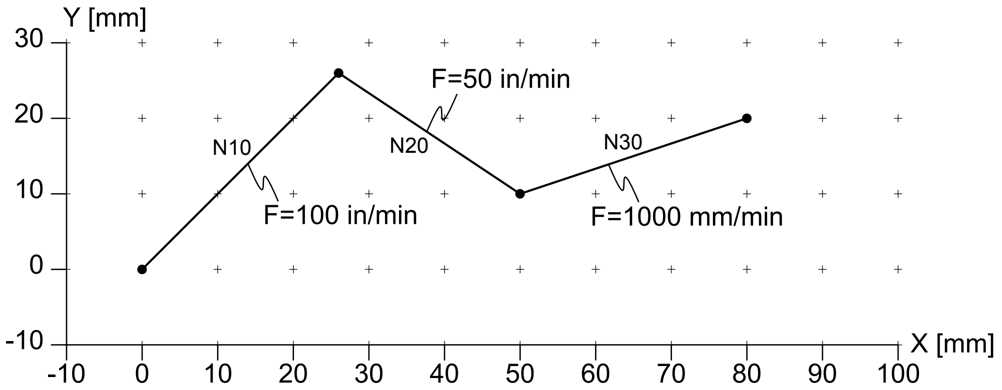


Figure “ExampleG700G710”.

IJK

`I<vx> J<vy> K<vz>`

Defines the center point for circular movements. See G02 [▶ 35], G03 [▶ 37] for details. The center point is defined as `currentPoint + [vx,vy,vz]`. The current length unit is used for `vx`, `vy`, `vz`. The parameters `I`, `J`, `K` are optional and have a `0` default value. The `I` parameter is also used by `G4` to define a duration.

M

`M<v>`

Triggers the `M`-function `v`. The timing and behavior depends on the definition of `v` in the System Manager of TwinCAT.

`M2` and `M30` are internally defined. Both functions trigger a synchronization with the NC-channel. (See wait [▶ 71]-function.) Both functions stop the execution of the GST-program. Due to this order the interpreter waits for the completion of the NC-channel before it stops.

In addition, `M30` also resets all fast `M`-functions and `H`, `S`, `T`.

NOTE:

There must not be more than one M-function of type handshake in a block.

Note

NOTE:

The M-functions M2 and M30 do not have to be defined by the user in the System Manager of TwinCAT.

Note

Example:

This example assumes the following definitions of M-functions:

M10:	Fast before move.
M11:	Fast after move.
M12:	Fast before move, auto-reset, reset M10, M11.
M20:	Handshake before move.
M21:	Handshake after move.

Figure “ExampleM10M11M12M20M21” visualizes the programmed path and the activation of M-functions. The fast M-functions M10, M11 are reset by M12, which itself is reset automatically.

```
N10 X10 G01
N20 X30 M10 M20
N30 X50 M11 M21
N40 X70
N50 X90 M12
```

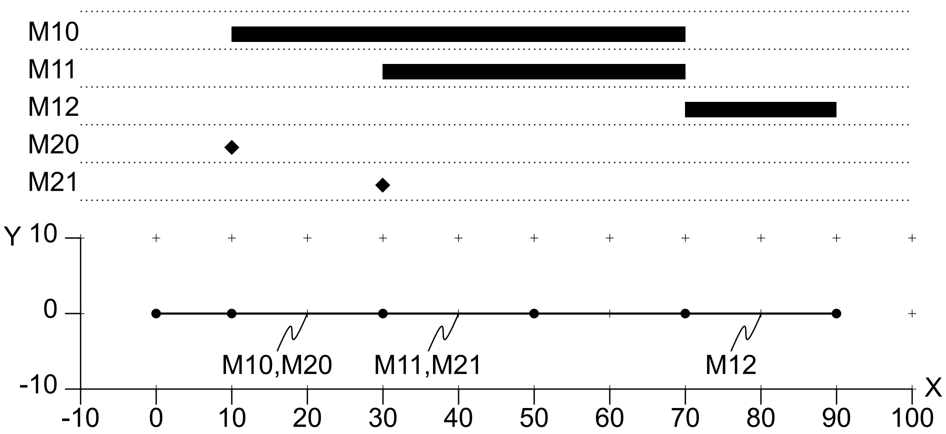


Figure “ExampleM10M11M12M20M21”.

P

P<v>

Switch tool orientation. The value of v must be 1 or -1. If v is negative, the tool points in the direction of the working plane normal. Otherwise, it points into the opposite direction.

Example:

The resulting MCS-path (MCS: machine coordinate system) of the following example is shown in Figure “ExampleP”. The first line of the program defines Tool `1` to have a length of `10`. `G18` activates the `XZ`-workingplane.

`N10`: The end of segment `N10` is not subject to any tool compensation as `D0` is active.

`N20`: For segment `N20` tool `1` is active with a positive tool orientation. To compensate the tool length the translation `[0,10,0]` is applied. (See section Transformations [▸ 80] for details.) Thereby, the PCS (program coordinate system) endpoint `[20,10,0]` of `N20` is mapped to the MCS (machine coordinate system) endpoint `[20,20,0]`. The MCS (machine coordinate system) point and the applied transformation are shown in Figure “ExampleP”.

`N30`: In block `N30` the tool orientation is switched, which sets the translation to `[0,-10,0]`. This translation is applied to the PCS (program coordinate system) endpoint of `N30` resulting in the MCS (machine coordinate system) endpoint `[30,0,0]`.

`N20..N90`: The blocks `N60..N90` are similar to `N20..N50`, except that the `Y`-coordinate is not programmed. Therefore, the tool length compensation does not become apparent, although it is active. That behavior happens because the current PCS (program coordinate system) point is always adapted on a changed transformation. (See section Applying Transformations [▸ 82] for details.)

```
!toolSet(index:=1, tooltype:=tooltypeDrill, length:=10);
G18
N10 X10 Y10 D0
N20 X20 Y10 D1
N30 X30 Y10 P-1
N40 X40 Y10 P1
N50 X50 Y10 D0
N60 X60 D1
N70 X70 P-1
N80 X80 P1
N90 X90 D0
```

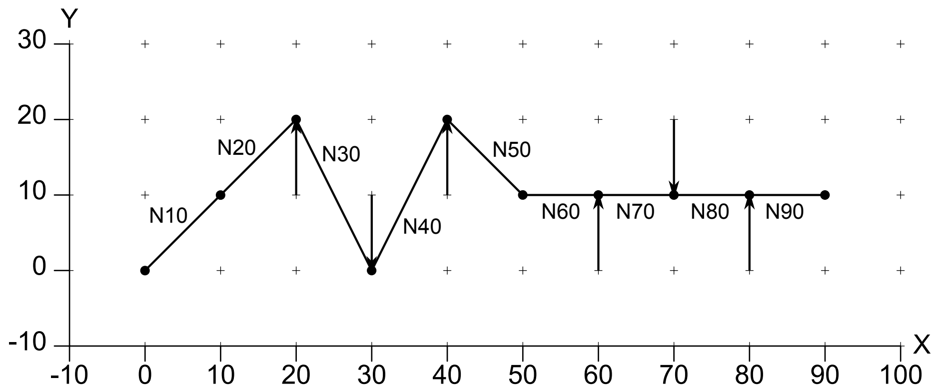


Figure “ExampleP”.

Q

`Q<i>=<v>`

Set the value of axis `Q<i>` to `v` where `i` must lie in the range `1` to `5`. The `Q`-axes use linear interpolation.

Note	**NOTE:** The address letters Q and R are handled in a special way for historical reasons.

Note	**CONSTRAINT:** The address Q<i> has to be followed by a G-expression or by an ST-expression. The G-word Q1100 is invalid. Use Q1=100, instead.

Example:

The path of the following example is shown in Figure “ExampleQ”. The Q-axes are interpolated linear with the interpolation of a movement. The last block (N40) results in a linear interpolation of a Q-axis without a concurrent movement.

```
N10 G01 X30 Y0 Q1=100 F6000
N20 G02 X50 Y20 I20 Q2=200
N30 G01 X60 Q1=300 Q2=300
N40 Q1=0
```

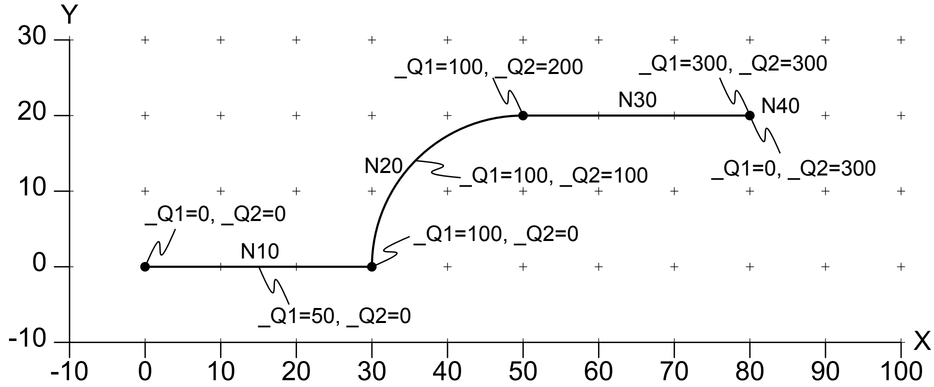


Figure “ExampleQ”.

N

N<v>

Set the block number to v. Typically, the block number is used to monitor the progress of the NC-program.

U

U<v>

In the context of G2 or G3 the radius is set to |v|. The current length unit is used for v. If v is positive, the shorter arc is used to interpolate between the current and the next point. If v is negative, the longer one is used. See G02 [▸ 35], G03 [▸ 37] for details.

X

X<v>

Sets the X-coordinate of the next point to v. The current length unit is used for v.

Y

Y<v>

Sets the Y-coordinate of the next point to v. The current length unit is used for v.

Z

Z<v>

Sets the Z-coordinate of the next point to v. The current length unit is used for v.

A

A<v>

Sets the A-angle of the next orientation to v. The current angle unit is used for v.

B

B<v>

Sets the B-angle of the next orientation to v. For v the current angle unit is used.

C

C<v>

Sets the C-angle of the next orientation to v. For v the current angle unit is used.

3.4.3 Execution Order

A block (line of G-code) consists of a sequence of words. The programmed order of words is not considered by the GST interpreter. Instead, the following execution order is obeyed that consists of 7 sequential and dependent steps.

1. Reference System	N*	Set block number.
	G17..G19	Selection of a workingplane.
	G70, G71, G700, G710	Selection of a unit.
	G90, G91	Selection of absolute/ incremental programming.
	D*, P*	Selection of a tool and its orientation.
2. Configuration	G40..G42	(De-)activation of Tool Radius Compensation.
	G53..G59	Selection and programming of zero offset shift.
	F*	Set velocity.
3. M-Function Pre	M*	M-functions that are configured as “before”.
4. Parameter to PLC	H*, S*, T*	

5. Movement	Q*, G00..G03	Movement to a point.
	G09, G60	Activation of accurate stop.
6. Wait	G04	Wait for a given duration.
7. M-Function Post	M*	M-functions that are configured as “after”.

The first step sets up the reference system. The second step configures following movements. Note that the second step may depend on the first one. E.g. the programmed velocity (F) considers a velocity unit (G700) that is programmed in the same block. Step three and the following steps perform actions like a movement.

3.4.4 Mutual Exclusive G-Codes

Certain combinations of G-Codes must not be programmed in the same block (line of G-Code). Such conflicting G-Codes typically set state variables to contradictory values (e.g. set length unit to mm and to inch). There are also combinations that use the same parameters and therefore must not be programmed in the same block (e.g. G58 and G59). Below is a list of groups of G-Codes. G-Codes that belong to the same group are in conflict.

- G00, G01, G02, G03, G04, G58, G59
  Interpolations and programmed zero-offset-shift.
- G70, G71, G700, G710
  Set unit for length and speed.
- G90, G91
  Set absolute/ relative programming.
- G53, G54, G55, G56, G57
  Deactivate/ select zero-offset-shift.
- G40, G41, G42
  Deactivate/ activate Tool Radius Compensation.
- G17, G18, G19
  Select workingplane.

3.5 ST - Structured Text (IEC 61131-3)

3.5.1 Comments

Line Comment

```
<st-code> // <comment>
```

Text between ‘//’ and the end of line is treated as comment in ST-Code.

Example:

```
{
VAR
    i : INT; // this variable is primarily used in FOR-loops for counting
END_VAR
}
```

/* */ Comment

```
<st-code> /* <comment>
<comment> */ <st-code>
```

Text between '/*' and '*/' is treated as comment in ST. This type of comment may be nested up to a depth of 3. The '/*…*/'-style comment may appear anywhere between literals, keywords, identifiers and special symbols. It may also contain G-Code lines.

Example:

The following example demonstrates the notation of comments in ST-Code. The first comment is placed within a variable declaration. The second comment encloses an entire ST-loop. The comment contains further comments and a G-Code line, which itself contains a G-Code comment.

```
{
VAR i /* used for counting */ : INT;   END_VAR

/* The following loop is commented out.
FOR i := 0 TO 10 DO
    /* zigzag pattern */
    ! G01 (linear interpolation) X=i Y{i MOD 2}
    // end of loop
END_FOR;
*/
}
```

(* *) Comment

```
<st-code> (* <comment>
<comment> *) <st-code>
```

Text between '(*' and '*)' is treated as comment in ST. This type of comment may be nested up to a depth of 3. It is similar to the /*…*/-style comment.

3.5.2 Literals

Integer Literals

Decimal	18
Binary	2#10010
Octal	8#22
Hexadecimal	16#12

The same integer value in decimal, binary, octal and hexadecimal notation.

Real Literals

Notation of real values

```
1.0
```

```
1.602E-19
```

Boolean Literals

Notation of Boolean values

0

1

TRUE

FALSE

Typed Literals

<typename>#<literal>

Typed literals where typename is a native type (e.g. Word or LReal) or an enumeration type (to avoid ambiguities).

Typing of literals is typically not necessary in GST, since the interpreter implements a decent typesystem that handles untyped literals properly. There are a few exceptions where the type of a literal is significant for semantics, like in the following example.

Example:

The first assignment assigns the value 16#80 to w, whereas the second one assigns the value 16#8000 to w.

```
{
VAR  w: word;  END_VAR
w :=  ror(BYTE#1,1);
w :=  ror(WORD#1,1);
}
```

String Literals

"abc"

'abc'

Notation of a 2-byte and a 1-byte string, respectively. Note that there is no implicit conversion between both types. The following escape-sequences can be used within both types of literals:

$L	line feed
$N	newline
$P	form feed
$R	carriage return
$t	tab
$' or $"	quotes
$<2 or 4 hexadecimal digits>	character of given code

.

Duration Literals

T#[+/-]<value><unit>[…]<value><unit>

TIME#[+/-] <value><unit>[…]<value><unit>

```
LT#[+/-]<value><unit>[…]<value><unit>
```

```
LTIME#[+/-]<value><unit>[…]<value><unit>
```

Time literals of type TIME or LTIME. The literal consists of an optional sign (+/-) and a sequence of value/unit pairs. Value must be an integer, except for the last one that may also be a floating point number. Values must not be negative and may be arbitraryly large. Units must appear in the following order.

d	day
h	hour
m	minute
s	second
ms	millisecond
us	microsecond
ns	nanosecond

An arbitrary subset of units may be used in a literal. For instance, the literal T#1d15ms1500.01us is valid.

Date Literals

```
DATE#<yyyy>-<mm>-<dd>
```

```
D#<yyyy>-<mm>-<dd>
```

```
LDATE#<yyyy>-<mm>-<dd>
```

```
LD#<yyyy>-<mm>-<dd>
```

Date literal of type DATE or LDATE. The literal is interpreted as UTC, i.e. timezone, daylight saving time and leap seconds are not considered. The year must not be smaller than 1970. The values yyyy, mm and dd have to be integer values, i.e. D#1980-20-10 is a valid date literal, for example.

Time-of-Day Literals

```
TIME_OF_DAY#<hh>:<mm>:<ss>
```

```
TOD#<hh>:<mm>:<ss>
```

```
LTIME_OF_DAY#<hh>:<mm>:<ss>
```

```
LTOD#<hh>:<mm>:<ss>
```

Time-of-day literal of type TOD or LTOD. The literal is interpreted as UTC, i.e. timezone, daylight saving time and leap seconds are not considered. hh and mm must be integer values. ss may be an integer or a floatingpoint number, i.e. TOD#7:30:3.1415 is a valid literal, for example.

Date-and-Time Literals

```
DATE_AND_TIME#<yyyy>-<mm>-<dd>-<hh>:<mm>:<ss>
```

```
DT#<yyyy>-<mm>-<dd>-<hh>:<mm>:<ss>
```

```
LDATE_AND_TIME#<yyyy>-<mm>-<dd>-<hh>:<mm>:<ss>
```

`LDT#<yyyy>-<mm>-<dd>-<hh>:<mm>:<ss>`

Date-and-time literal of type `DT` or `LDT`. The literal is interpreted as UTC, i.e. timezone, daylight saving time and leap seconds are not considered. This literal is a combination of the date literal and the time-of-day literal. Analogously, the corresponding rules for these two parts apply.

3.5.3 Native Data Types

Bitstring Types

`BOOL`, `BYTE`, `WORD`, `DWORD`, `LWORD`

Bitstring types of 1, 8, 16, 32 and 64 bit. Implicit conversion from left to right using zero extension.

Unsigned Integer Types

`USINT`, `UINT`, `UDINT`, `ULINT`

Unsigned integer types of 8, 16, 32 and 64 bit. Implicit conversion from left to right preserving the value.

Signed Integer Types

`SINT`, `INT`, `DINT`, `LINT`

Signed integer types of 8, 16, 32 and 64 bit. Implicit conversion from left to right preserving the value. An unsigned type of `n` bit is also implicitly converted to a signed type of `m` bit where the relation `m > n` must hold. There is no implicit conversion between bitstring types and integer types.

String Types

`string[<length>]`

`wstring[<length>]`

1-byte and 2-byte strings of given `length`. If `length` is omitted, it has `255` as default value.

Character Types

`char`

`wchar`

Single 1-byte and 2-byte character of a string. It can be implicitly converted to a string.

Time-Related Types

`TIME`, `LTIME`

`DATE`, `LDATE`

`TIME_OF_DAY`, `TOD`, `LTIME_OF_DAY`, `LTOD`

`DATE_AND_TIME`, `DT`, `LDATE_AND_TIME`, `LDT`

Datatypes for duration, date and time. Internally, all values of these types are represented with a granularity of 1 nanosecond. Values of date-related types represent the number of nanoseconds since `1.1.1970` (UTC). Leapseconds are ignored. Implicit conversion is allowed from a non-`L` type to an `L` type, e.g. from `TIME` to `LTIME`.

3.5.4 Userdefined Types

Derived Types

```
TYPE
<typeName>: <typeName> := <defaultValue>;
END_TYPE
```

Definition of a new type as an alias to an existing type. The default value is optional.

Enumeration Types

```
TYPE
<typeName> : (<enumValue>, …, <enumValue>) := <defaultValue>;
END_TYPE
```

Definition of an enumeration type. The default value is optional.

Enumeration Types with Defined Values

```
TYPE
<typeName> : (<enumValue>:=<integer value>, …,
<enumValue>:=<integer value>) := <defaultValue>;
END_TYPE
```

Definition of an enumeration type with user-defined values for each element. The default value is optional.

Array Types

```
TYPE
<typeName>: ARRAY [<from>..<to>,<from>..<to>] OF <typeName> :=
[<defaultValue>, <repetition>(<defaultValue>), …];
END_TYPE
```

Definition of an array type. The array may be multi-dimensional. The index range is defined for each dimension. At runtime the boundaries of the array are checked. A boundary violation leads to a runtime-error. The default values are defined in ascending order starting with the last dimension. A value can be repeated by placing it into parentheses prefixed with the number of repetitions. If the number of defined default values does not match the array size, initialization is truncated or padded with the default value of the element type. In either cases a compile-time warning is issued.

Structure Types

```
TYPE
<typeName>: STRUCT
    <memberName>: memberType;
    …
END_STRUCT := (<memberName> := <defaultValue>, …);
END_TYPE
```

Defines a structure type of the given members. Currently, the default value is placed after the type definition. This positional style is a difference to the `ST`-standard.

Pointer Types

```
TYPE
    <typeName>: REF_TO <basetypeName>;
END_TYPE
```

Defines a pointer type of the given base type.

3.5.5 Control Structures

IF-THEN-ELSIF-ELSE

```
IF <condition> THEN
    <statements>
ELSIF <condition> THEN
    <statements>
ELSE
    <statements>
END_IF;
```

Conditional statement. The `ELSIF`-branch and `ELSE`-branch are optional. `ELSIF` can be repeated arbitrarily.

CASE OF

```
CASE <expression> OF
    <value>, <value>, …, <value>: <statements>
ELSE
    <statements>
END_CASE;
```

The case-list consists of a comma-separated sequence of values or ranges. Only the first matching case is executed. The optional `ELSE`-branch is executed if no case matches.

FOR

```
FOR <variable> := <expression> TO <expression> BY <expression> DO
    <statements>
END_FOR;
```

Iterates over the given `variable` in the defined range (including) using the supplied step-size. If the latter is omitted, it has `1` as default value.

WHILE

```
WHILE <condition> DO
    <statements>
```

```
END_WHILE;
```

Pre-checked loop.

REPEAT

```
REPEAT
    <statement>
UNTIL <condition>
END_REPEAT;
```

Post-checked loop. The break condition is evaluated after performing the `<statements>` the loop includes.

EXIT

```
EXIT;
```

`EXIT` can be used within loops to leave the loop. If loops are nested, only the innermost loop is left. If there is no loop surrounding the `EXIT` keyword, a compile-time error is issued.

3.5.6 Userdefined Functions

Function Definition

```
FUNCTION <name> : <returntype>
VAR_INPUT
    <variable declarations>
END_VAR
VAR_OUTPUT
    <variable declarations>
END_VAR
VAR_IN_OUT
    <variable declarations>
END_VAR
VAR
    <variable declarations>
END_VAR
VAR_EXTERNAL
    <variable declarations>
END_VAR
    <statements>
END_FUNCTION
```

Declares a function. Thereafter, it is callable by its name. The declaration of the return type is optional. If it is supplied, the function returns a value of the given type. The return value is defined within the function body by an assignment to the function name.

The function may have input, output and in-out parameters. The order of declaration is significant. It is used for nonformal calls. Declared variables are only used within the function body. External variables are imported from global scope. Variables and parameters are not persistent, i.e. they do not retain their value between two calls. A function may call itself (recursion).

Nonformal Function Call

```
<functionname>(<expression>, …, <expression>)
```

Nonformal function call. The order of expressions must match the number and order of declared parameters.

Formal Function Call

```
<functionname>(
    <inputParamName>  := <expression>,
    <outputParamName> => <variableName>,
    <inputParamName>  := <variableName>)
```

Formal function call. Parameters are identified by their name. If a declared parameter is not listed, it is implicitly set to its default value.

3.5.7 Standard Functions

3.5.7.1 Type Conversion

Type Conversion (*_TO_*)

```
<nativeType>_to_<nativeType>(x)
```

```
to_<nativeType>(x)
```

Explicit conversion between the given native types. The second alternative is overloaded for any applicable type.

For conversion from floatingpoint to integer `x` is rounded.

3.5.7.2 Arithmetic and Trigonometric

ABS

```
ABS(x)
```

Returns the absolute value of `x`.

The function is overloaded for any integer type and floatingpoint type. The type of `x` is used as return type.

SQRT

```
SQRT(x)
```

Returns the square root of `x`.

The function is overloaded for any floatingpoint type. The type of `x` is used as return type.

Note	**RESTRICTION:** Variable `x` must not be negative.

LN

`LN(x)`

Returns the natural logarithm of `x`, i.e. the logarithm to the base `e`.

The function is overloaded for any floatingpoint type. The type of `x` is used as return type.

Note	**RESTRICTION:** Variable `x` must be larger than `0`.

LOG

`LOG(x)`

Returns the logarithm of `x` to the base `10`.

The function is overloaded for any floatingpoint type. The type of `x` is used as return type.

Note	**RESTRICTION:** Variable `x` must be larger than `0`.

EXP

`EXP(x)`

Returns `e` raised to the power of `x`.

The function is overloaded for any floatingpoint type. The type of `x` is used as return type.

SIN

`SIN(x)`

Returns the sine of `x` where `x` is expected to be in radians.

The function is overloaded for any floatingpoint type. The type of `x` is used as return type.

Confer the gSin [▸ 77] function.

COS

`COS(x)`

Returns the cosine of `x` where `x` is expected to be in radians.

The function is overloaded for any floatingpoint type. The type of `x` is used as return type.

Confer the gCos [▸ 77] function.

TAN

`TAN(x)`

Returns the tangent of `x` where `x` is expected to be in radians.

The function is overloaded for any floatingpoint type. The type of `x` is used as return type.

Confer the gTan [▸ 77] function.

ASIN

`ASIN(x)`

Returns the arc sine of `x` within the interval `[-PI/2,PI/2]` radians.

The function is overloaded for any floatingpoint type. The type of `x` is used as return type.

Confer the gASin [▶ 78] function.

Note	**RESTRICTION:** Variable `x` must lie within the interval `[-1,1]`.

ACOS

`ACOS(x)`

Returns the arc cosine of `x` within the interval `[0,PI]` radians.

The function is overloaded for any floatingpoint type. The type of `x` is used as return type.

Confer the gACos [▶ 78] function.

Note	**RESTRICTION:** Variable `x` must lie within the interval `[-1,1]`.

ATAN

`ATAN(x)`

Returns the arc tangent of `x` within the interval `[-PI/2,PI/2]` radians.

The function is overloaded for any floatingpoint type. The type of `x` is used as return type.

Confer the gATan [▶ 78] function.

ATAN2

`ATAN2(y,x)`

Returns the arc tangent of `y/x` within the interval `[-PI,PI]` radians.

The function is overloaded for any floatingpoint type. The smallest common type of `x` and `y` is used as return type.

Confer the gATan2 [▶ 78] function.

ADD

`ADD(x1, x2, …)`

Returns the sum of all parameters. The `ADD`-function can have an arbitrary number of parameters, but has to have at least one.

The function is overloaded for any integer and floatingpoint type. The smallest common type of all parameters is used as return type.

MUL

`MUL(x1, x2, …)`

Returns the product of all parameters. The `MUL`-function can have an arbitrary number of parameters, but has to have at least one. The infix-operator '`*`' can be used as an alternative.

The function is overloaded for any integer and floatingpoint type. The smallest common type of all parameters is used as return type.

SUB

```
SUB(x,y)
```

Returns the difference `x-y`. The infix-operator '-' can be used as an alternative.

The function is overloaded for any integer and floatingpoint type. The smallest common type of `x` and `y` is used as return type.

DIV

```
DIV(x,y)
```

Returns the quotient `x/y`. The infix-operator '`/`' can be used as an alternative.

The function is overloaded for any integer and floatingpoint type. The smallest common type of `x` and `y` is used as return type. If the return type is an integer type, the result is truncated towards zero.

Note	**RESTRICTION:** Variable `y` must not be zero.

MOD

```
MOD(x,y)
```

Returns the remainder of the integer division `x/y`. The infix-operator '`MOD`' can be used as an alternative.

The function is overloaded for any integer type. The smallest common type of `x` and `y` is used as return type. The result may also be negative. The equation `x = MUL(DIV(x,y),y) + MOD(x,y)` holds.

Note	**RESTRICTION:** Variable `y` must not be zero.

EXPT

```
EXPT(x,y)
```

Returns `x` raised to the power of `y`.

The function is overloaded such that `x` has a floatingpoint type and `y` has a floatingpoint type or integer type. The type of `x` is used as return type, i.e. returned is a `Real` or an `LReal` floating point type. The infix-operator '`**`' can be used as an alternative.

Note	**RESTRICTION:** If `x` is negative, then `y` must be an integer.

Note	**RESTRICTION:** If `x` is zero, then `y` must be larger than zero.

3.5.7.3 Shift and Rotation

SHL

`SHL(x,y)`

Returns the bitstring `x` shifted left by `y` bits. Zero-bits are inserted at the right side. The least significant bit is assumed to be rightmost.

The function is overloaded for any bitstring type for `x` and any integer type for `y`. The type of `x` is used as return type.

Note	**CONSTRAINT:** Variable `y` must not be negative.

SHR

`SHR(x,y)`

Returns the bitstring `x` shifted right by `y` bits. Zero-bits are inserted at the left side. The least significant bit is assumed to be rightmost.

The function is overloaded for any bitstring type for `x` and for any integer type for `y`. The type of `x` is used as return type.

Note	**CONSTRAINT:** Variable `y` must not be negative.

ROL

`ROL(x,y)`

Returns the bitstring `x` rotated left by `y` bits. Bits that are shifted out at the left side are inserted at the right side. The least significant bit is assumed to be rightmost.

The function is overloaded for any bitstring type for `x` and for any integer type for `y`. The type of `x` is used as return type.

Note	**CONSTRAINT:** Variable `y` must not be negative.

ROR

`ROR(x,y)`

Returns the bitstring `x` rotated right by `y` bits. Bits that are shifted out at the right side are inserted at the left side. The least significant bit is assumed to be rightmost.

The function is overloaded for any bitstring type for `x` and for any integer type for `y`. The type of `x` is used as return type.

Note	**CONSTRAINT:** Variable `y` must not be negative.

3.5.7.4 Logical Operations

AND

`AND(x1, x2, …)`

Returns the bitwise Logical And of all parameters. Bit `i` is set in the result if bit `i` is set in all parameters. The `AND` function can have an arbitrary number of parameters, but has to have at least one.

The function is overloaded for any bitstring type. The smallest common bitstring type is used as return type.

OR

`OR(x1, x2, …)`

Returns the bitwise Logical Or of all parameters. Bit `i` is set in the result if bit `i` is set in at least one of all parameters. The `OR` function can have an arbitrary number of parameters, but has to have at least one.

The function is overloaded for any bitstring type. The smallest common bitstring type is used as return type.

XOR

`XOR(x1, x2, …)`

Returns the bitwise Logical Exclusive Or of all parameters. Bit `i` is set in the result if bit `i` is set in an uneven number of all parameters. The `XOR` function can have an arbitrary number of parameters, but has to have at least one.

The function is overloaded for any bitstring type. The smallest common bitstring type is used as return type.

NOT

`NOT(x)`

Returns the bitwise complement of `x`. Bit `i` is set in the result if bit `i` is not set in `x`.

The function is overloaded for any bitstring type. The type of `x` is used as return type.

3.5.7.5 Selection (Conditional Expressions)

SEL

`SEL(cond,x1,x2)`

Returns `x1` if `cond` is false, and `x2` otherwise.

MUX

`MUX(select, x0, x1, …, xN)`

Returns `x<select>`. If `select` is `0`, `x0` is returned. If `select` is `1`, `x1` is returned and so forth. The `MUX` function can have an arbitrary number of parameters, but has to have at least two.

The function is overloaded for any type for `x<i>` and for any integer for `select`. The smallest common type of `x<i>` is used as return type.

Note	**RESTRICTION:** The variable `select` must lie within the interval `[0,N]`. Otherwise, an out-of-bounds error is issued at runtime.

3.5.7.6 Min, Max and Limit

MAX

`MAX(x1, x2, …)`

Returns the maximum of all parameters.

The function is overloaded for any integer and floatingpoint type. The smallest common type of all parameters is used as return type.

MIN

`MIN(x1, x2, …)`

Returns the minimum of all parameters.

The function is overloaded for any integer and floatingpoint type. The smallest common type of all parameters is used as return type.

LIMIT

`LIMIT(min,in,max)`

Returns `in` if it lies in the interval `[min,max]`. Otherwise, the violated bound (`min` or `max`) is returned.

The function is overloaded for any integer and floatingpoint type. The smallest common type of all parameters is used as return type.

Note	**CONSTRAINT:** The `min` boundary must be smaller than the `max` boundary.

3.5.7.7 Comparison

GT

`GT(x,y)`

Returns `TRUE` if `x` is larger than `y`. The smallest common type of `x` and `y` is used to perform the comparison.

The function is overloaded for all integer and floatingpoint types. The returntype is `BOOL`.

GE

`GE(x,y)`

Returns `TRUE` if `x` is not smaller than `y`. The smallest common type of `x` and `y` is used to perform the comparison.

The function is overloaded for all integer and floatingpoint types. The returntype is `BOOL`.

EQ

`EQ(x,y)`

Returns `TRUE` if `x` and `y` are equal. The smallest common type of `x` and `y` is used to perform the comparison.

The function is overloaded for all integer and floatingpoint types. The returntype is `BOOL`.

LE

`LE(x,y)`

Returns `TRUE` if `x` is not larger than `y`. The smallest common type of `x` and `y` is used to perform the comparison.

The function is overloaded for all integer and floatingpoint types. The returntype is `BOOL`.

LT

`LT(x,y)`

Returns `TRUE` if `x` is smaller than `y`. The smallest common type of `x` and `y` is used to perform the comparison.

The function is overloaded for all integer and floatingpoint types. The returntype is `BOOL`.

NE

NE(x,y)

Returns `TRUE` if `x` and `y` are not equal. The smallest common type of `x` and `y` is used to perform the comparison.

The function is overloaded for all integer and floatingpoint types. The returntype is `BOOL`.

3.6 CNC Functions

3.6.1 Strings and Messages

The toString-Function

```
toString(<arg0>, …, <argN>): STRING
```

Converts and concatenates the given arguments to one string. This string is limited to 255 characters, which is the default string length. The `toString`-function behaves like the print function, except that it yields a formatted string instead of printing.

Note	**NOTE:** The `toString`-function is especially useful to format a string for the `msg(…)`-function.

Msg

`msg(str:= String[81])`

Send the given message to the message list of TwinCAT. The message is processed by the NC-channel synchronously. It appears in the user-interface when all preceeding NC-commands are completed.

To send formatted strings this function can be combined with the `toString`-function.

Note	**NOTE:** The message is restricted to 81 characters. Text exceeding this restriction will be truncated.

Example:

The path of the following example is shown in Figure “ExampleMsg”. It is annotated with the emitted messages.

```
{
VAR
    x,y,z: LREAL;
    start: LDT;
END_VAR

!N10 G00 X0 Y0 F300
start := currentLdt();
!N20 G01 X30
msg('N20 completed');
!N30 X60 Y10
frameGet(x=>x,y=>y,z=>z); msg(toString('Current position: [',x,',',y,',',z,']'));
!N40 X90
sync();
msg(toString('Machining time: ', currentLdt()-start));
}
```

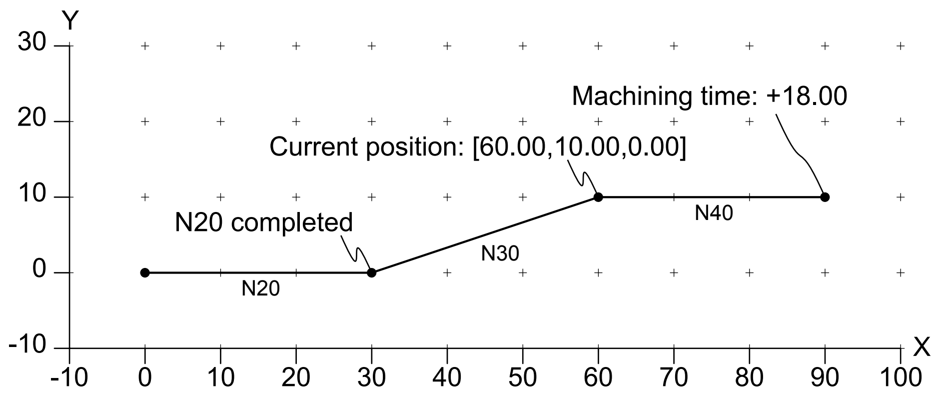


Figure “ExampleMsg”.

3.6.2 Transformations

transRotX/Y/Z

`transRotX(angle:= LReal)`

`transRotY(angle:= LReal)`

`transRotZ(angle:= LReal)`

Rotation around the respective axis by the given angle in the user-defined angle unit. The rotation is pushed onto the stack of transformations. The angle value is interpreted using the current angle-unit. See section Transformations [▶ 80] for details.

Example:

The resulting path of the following example is shown in Figure “ExampleTransRotZ”.

- `N10` is programmed with the PCS (program coordinate system) and the MCS (machine coordinate system) being equal.
- `N20` is programmed after a `45`-degree rotation around the `Z`-axis in `[0,0,0]` has been pushed onto the stack of transformations. Another rotation of `45` degrees is pushed onto the transformation stack such that the rotations add up to `90` degree.
- Therefore, the MCS (machine coordinate system) coordinate of the end of segment `N30` is `[0,30,0]`.

```
N10 G01 X30 Y0
!transRotZ(45);
N20 G01 X30 Y0
```

```
 !transRotZ(45);
N30 G01 X30 Y0
!transPop();
!transPop();
```

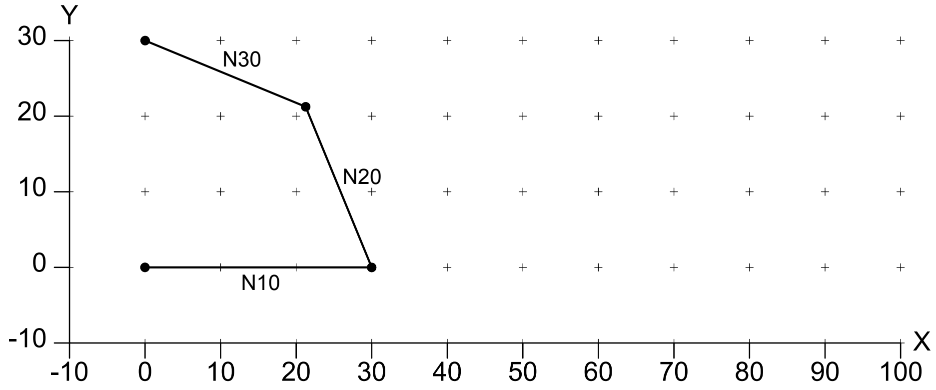


Figure “ExampleTransRotZ”.

transRotA

`transRotA(x:=LReal, y:=LReal, z:=LReal, angle:=LReal)`

Rotate around vector `[x,y,z]` by the given `angle`. The rotation is pushed onto the stack of transformations. The angle value is interpreted using the current angle-unit. See section Transformations [▶ 80] for details.

Note	NOTE: The vector `[x,y,z]` must not be the zero vector.

Example:

The resulting path of the following example is shown in Figure “ExampleTransRotA”. The first invocation of `transRotA` rotates the PCS (program coordinate system) around the positive `Z`-axis (right-hand rule) by `45` degree. The second invocation rotates around the negative `Z`-axis by the same angle, i.e. into the opposite direction. The combination of both rotations is the identity transformation.

```
!transRotA(0,0,1,45);
N10 G01 X30 Y0
!transRotA(0,0,-1,45);
N20 G01 X30 Y0
!transPop();
!transPop();
```

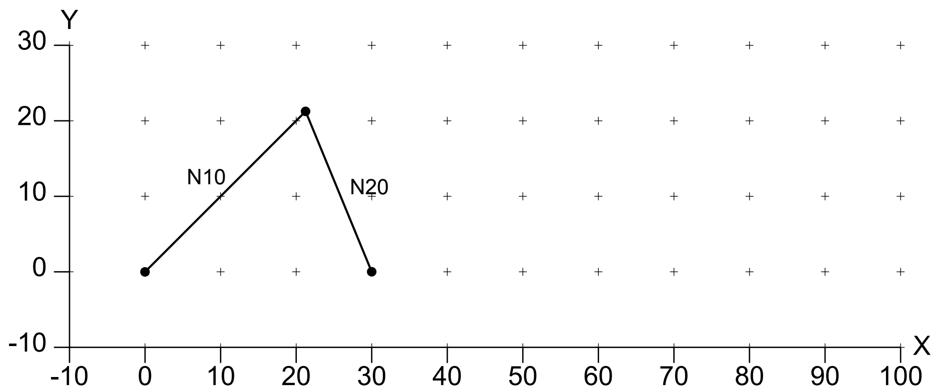


Figure “ExampleTransRotA”.

transMirrorX/Y/Z

`transMirrorX()`

`transMirrorY()`

`transMirrorZ()`

Mirror with respect to the `X`-direction, `Y`-direction or `Z`-direction relative to the origin of the current PCS (program coordinate system). The transformation is pushed onto the stack of transformations.

Note	**NOTE:** The invocation of a mirror function switches the orientation of the coordinate system from right-handed to left-handed or vice versa. Most notably, this behavior switches the rotation direction of circles and the compensation direction of tool radius compensation. By default, the coordinate system is right-handed.

Example:

The resulting path of the following example is shown in Figure “ExampleTransMirrorX”. The PCS (program coordinate system) is mirrored along the `X`-dimension. Thereby, the coordinate system becomes a left-handed system, within which the rotation direction of `G2` is (intentionally) swapped.

```
N10 G02 X20 Y20 U20
!transMirrorX();
N20 G02 X-40 Y0 U20
!transPop();
```

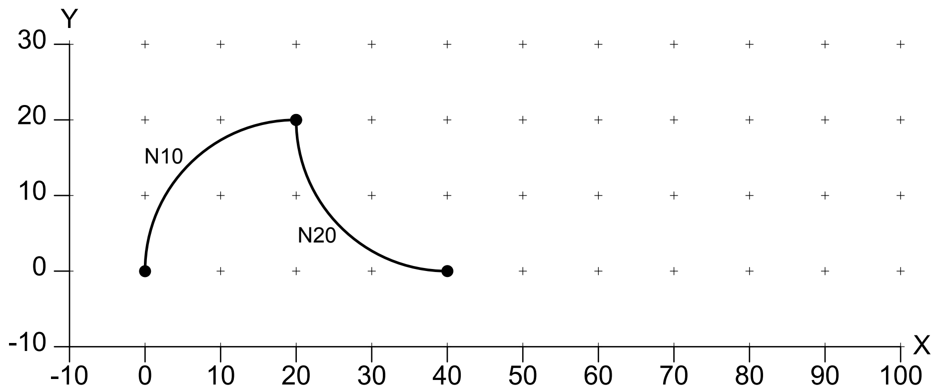


Figure “ExampleTransMirrorX”.

transScale

`transScale(factor:= LReal)`

Scales the coordinate system by the `factor` in the `X`-dimension, `Y`-dimension and `Z`-dimension. The transformation is pushed onto the stack of transformations.

Note	**NOTE:** The factor must be nonzero.

Note	**NOTE:** If the factor is negative, the coordinate system is effectively mirrored in the `X`-dimension, `Y`-dimension and `Z`-dimension. Thus, the orientation of the coordinate system is swapped.

Example:

The resulting path of the following example is shown in Figure “ExampleTransScale”. After scaling by a factor of `2`, the endpoint of segment `N20` is mapped to `[60,20,0]`.

```
N10 G01 X30 Y10
!transScale(2);
N20 G01 X30 Y10
!transPop();
```

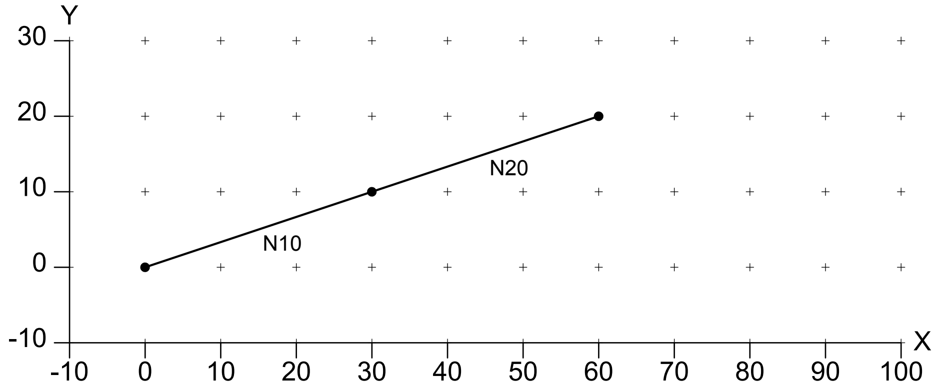


Figure “ExampleTransScale”.

transTranslate

`transTranslate(x:=LReal, y:=LReal, z:=LReal)`

Translate by vector `[x,y,z]`. The translation is pushed onto the stack of transformations.

Example:

The resulting path of the following example is shown in Figure “ExampleTransTranslate”. After translating by `[40,20,0]` the endpoint of segment `N20` is mapped to `[80,20,0]`.

```
N10 G01 X20 Y0
!transTranslate(40,20,0);
N20 G01 X40 Y0
!transPop();
```

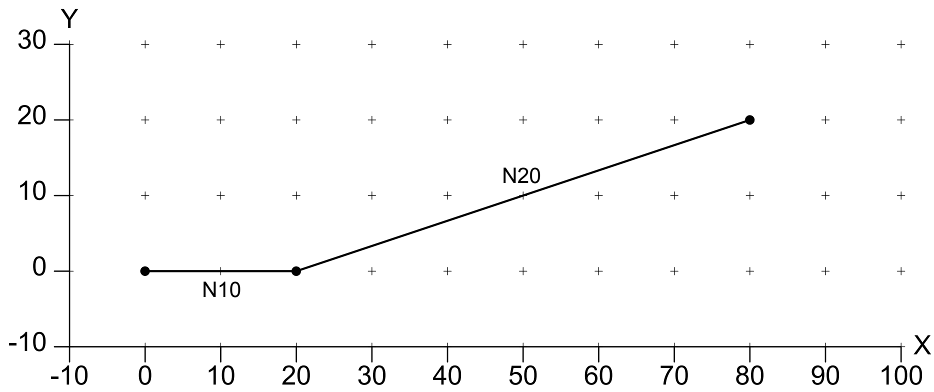


Figure “ExampleTransTranslate”.

transPop

`transPop()`

Pops a transformation from the stack of transformations.

Example:

The resulting path of the following example is shown in Figure “ExampleTransPop”. This example pushes the translation `[0,20,0]` onto the stack, followed by the translation `[0,10,0]`. Thereby, the effective translation for `N30` is `[0,30,0]`. The invocation of `transPop` removes the translation `[0,10,0]` from the stack. Thus, the endpoint of segment `N40` is translated by `[0,20,0]`. After removing the last translation from the stack the endpoint of segment `N50` is not translated at all.

```
N10 G01 X10 Y0
!transTranslate(0,20,0);
N20 G01 X30 Y0
!transTranslate(0,10,0);
N30 G01 X50 Y0
!transPop();
N40 G01 X70 Y0
!transPop();
N50 G01 X90 Y0
```

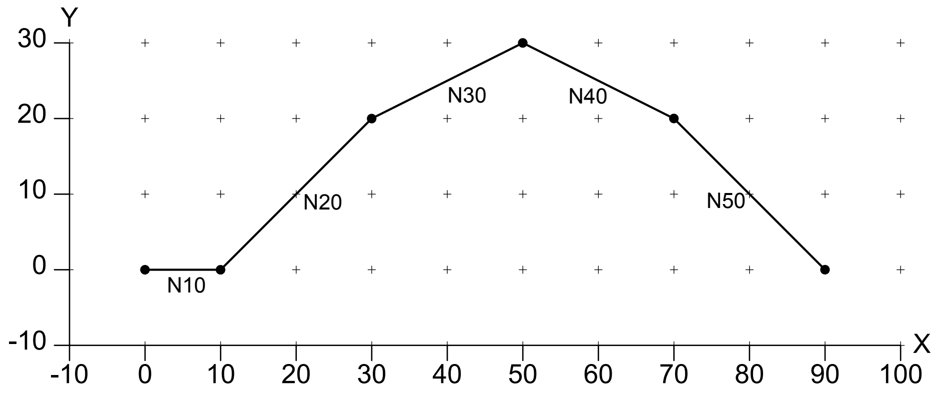


Figure “ExampleTransPop”.

transDepth

`transDepth(): UInt`

Yields the depth of the stack of transformations, i.e. the number of active transformations. See transRestore [▶ 67] for more details.

transRestore

`transRestore(depth:= UInt)`

Reduces the stack of transformations to the given depth. This command is typically used in conjunction with `transDepth()` to restore an earlier state of the stack.

Note	**NOTE:** The current depth of the stack must not be smaller than the given depth.

Example:

The resulting path of the following example is shown in Figure “ExampleTransDepthTransRestore”. A translation to `[40,10,0]` is initially pushed onto the transformation stack. The resulting depth is stored in variable `savedDepth`. The following code repeatedly performs a linear move to `X20 Y0` and a rotation by `45` degree. This resulting path is one half of an octagon, composed of segments `N10` to `N50`. When `N50` is processed, the transformation stack contains the initial translation and `4` rotations by `45` degree. The invocation of `transRestore(savedDepth)` restores the stack depth of `1` by removing all rotations. Hence, only the translation is applied to `N60`.

```
!VAR savedDepth : UINT; END_VAR
!transTranslate(40,10,0);
!savedDepth := transDepth();

N10 G01 X20 Y0
!transRotZ(45);
N20 G01 X20 Y0
!transRotZ(45);
N30 G01 X20 Y0
!transRotZ(45);
N40 G01 X20 Y0
!transRotZ(45);
N50 G01 X20 Y0
!transRestore(savedDepth);
N60 G01 X10 Y0
```

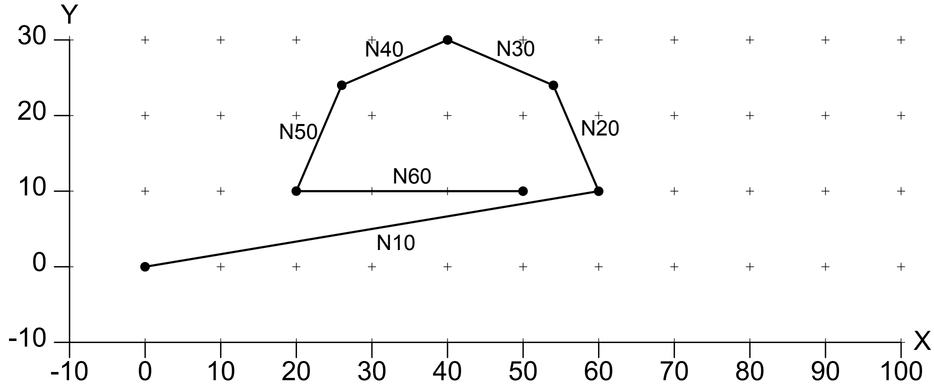


Figure “ExampleTransDepthTransRestore”.

3.6.3 Movement

moveCircle3d

`moveCircle3d(cx:=LReal, cy:=LReal, cz:=LReal, nx:=LReal, ny:=LReal, nz:=LReal, angle:=LReal, height:=LReal)`

Move circular by rotating around the center `cx`,`cy`,`cz` and the normal vector `nx`,`ny`,`nz` by the given `angle`. If `height` is nonzero, a helix is described. If `angle` is greater than a full circle, a multiturn circle or a multiturn helix is described. The rotation is performed according to the right hand rule. Using a negative angle or flipping the normal will reverse the direction of rotation. The angle value is interpreted using the current angle unit. The parameters `x`, `y`, `z`, `cx`, `cy`, `cz` are interpreted using the current length unit.

Note	**NOTE:** The radius must be nonzero.

Example:

The resulting path of the following example is shown in Figure “ExampleMoveCircle3D”. The invocation of `moveCircle3D` describes a helical movement. It starts at the current point that is `[40,10,0]`. The center axis of the helix is defined by the point `[30,10,0]` and direction `[gSin(22.5),0,gCos(22.5)]`. Compared to the workingplane normal `[0,0,1]` the axis is tilted by `22.5` degree in `X`-direction. The angle of `720+90` degree describes a multiturn helix. It exhibits a height of `30` with respect to the center axis. The endpoint of the helix is not explicitly programmed to avoid redundancy. If the user requires these coordinates, they can be retrieved by the `frameGet(…)` function, as demonstrated. The approximate coordinates are shown in Figure “ExampleMoveCircle3D”.

```
{
VAR
    x,y,z: LREAL;
```

```
END_VAR

!N10 G01 X40 Y10
moveCircle3D(cx:=30, cy:=10, cz:=0, nx:=gSin(22.5), ny:=0, nz:=gCos(22.5), an-
gle:=720+90, height:=30);
!frameGet(x=>x,y=>y,z=>z);
}
```

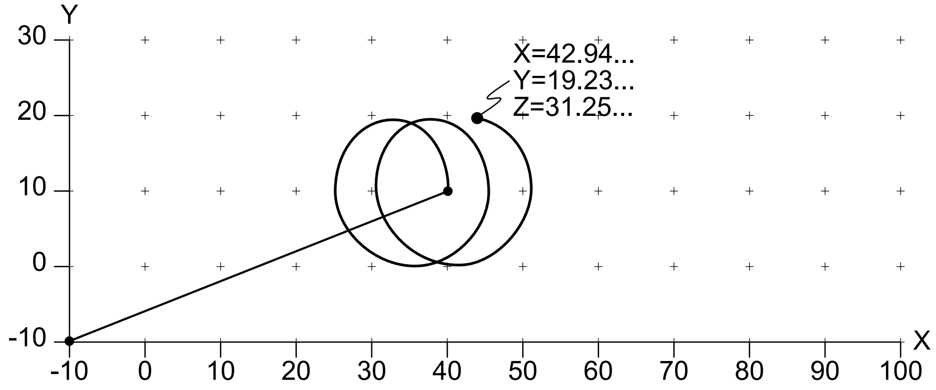


Figure “ExampleMoveCircle3D”.

3.6.4 Centerpoint Correction

centerpointCorrectionSet

`centerpointCorrectionSet(on:= bool)`

Activates the centerpoint correction for circles. The centerpoint correction will be used for circles that are defined using centerpoint programming. See G02 [▸ 35] and G03 [▸ 37]. Due to inaccuracies (e.g. rounding errors by the CAD program), the radius of the starting-point and endpoint with respect to the centerpoint may differ. If centerpoint correction is active, the center will be moved in such a way that the starting-radius and endradius are equal to their former average.

A limit for centerpoint correction can be configured with `centerpointCorrectionLimitSet(…)`. If this limit is exceeded, a runtime error will be reported.

centerpointCorrectionLimitSet

`centerpointCorrectionLimitSet(limit:= LReal)`

Sets the precision limit for the centerpoint of circles. If the given limit is exceeded, a runtime error is reported. The default limit value is 0.1 mm.

3.6.5 Tools

toolParamSet

`toolParamSet(tidx:= USInt, col:= USInt, val:= LReal)`

Set a parameter of the tool `tidx` (1..255) to `val`. The parameter is identified by `col` (0..15).

COL	DESCRIPTION

0	tool number For giving the tool a number. Written to the T-parameter in the cyclic channel interface.
1	tool type (10: drill, 20: miller) The drill is type 10. The miller is type 20.
2	length Describes the length of e.g. the drill.
3	
4	radius
5	length (added to the length value of column 2) Describes the wear on e.g. the drill. The wear has to be given as a negative value as it is added to the length.
6	
7	radius (added to the radius value of column 4)
8	x-shift Cartesian tool displacement in x-direction.
9	y-shift Cartesian tool displacement in y-direction.
10	z-shift Cartesian tool displacement in z-direction.

toolParam

```
toolParam(tidx:= USInt, col:= USInt): LReal
```

Yields the given tool parameter.

toolSet

```
toolSet(index:= USInt, nr:= Int, tooltype:= ToolType, length:= LReal, radius:= L
Real, lengthAdd:= LReal, radiusAdd:= LReal, offsetX:= LReal, offsetY:= LReal, of
fsetZ:= LReal)
```

Set all parameters of a tool. The index is used in D-words to refer to the tool. It must lie in the range 1 to 255. The parameter nr has only informational purpose. Typically, it is a company internal number to identify a certain tool. The parameter tooltype identifies the kind of tool, like a drill for instance. The remaining parameters are dimensions, which are visualized in Figure “ToolSetDimensions”. If the tool orientation is changed towards the negative (see P [▶ 43]-word), the value `length+lengthAdd` is implicitly negated. The parameters `length`, `radius`, `lengthAdd`, `radiusAdd`, `offsetX`, `offsetY` and `offsetZ` are interpreted using the current length unit.

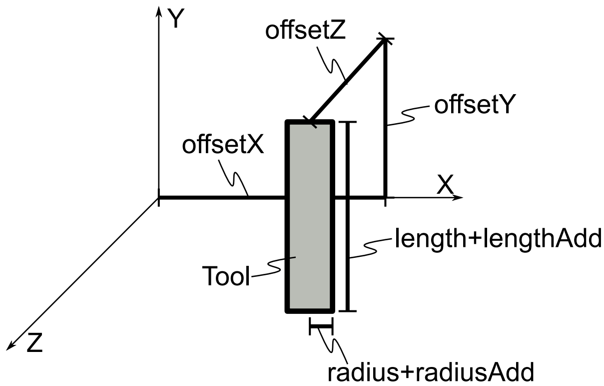


Figure “ToolSetDimensions”.

Example:

The example defines tool `1` as a drill of total length `48.5` and tool `2` as a mill with a length of `30` and a diameter of `5`.

```
!toolSet(index:=1, nr:=4711, tooltype:=tooltypeDrill, length:=50, lengthAdd:=-1.5);
!toolSet(index:=2, nr:=10783, tooltype:=tooltypeMill, length:=30, radius:=2.5);
```

End of example code.

ToolType

Enumeration of tool types.

```
    tooltypeDrill
    tooltypeMill
```

3.6.6 Synchronization

sync

`sync()`

Synchronizes the interpreter with the associated NC-channel. The `sync()`-command blocks until all pending NC-commands are completed, i.e. until the job-queue of the NC-channel is empty. This command replaces the former `@714`-command. Oftentimes, the `sync()`-command is combined with a preceeding `M`-function of type handshake. Then, the `sync()`-command will block until the `M`-function is acknowledged by the PLC.

wait

`wait()`

Waits for a `GoAhead`-signal from the PLC. The `wait()`-command blocks until this signal is received. This command replaces the former `@717`-command. Compared to a combination of an `M`-function and `sync()`, this kind of synchronization does not result in an empty job queue. Notably, an empty job queue forces the machine to halt.

Note

TIP:

The `GoAhead`-signal may be send from the PLC before the associated `wait()`-function is called. In this case the `wait()`-function does not block.

3.6.7 Query of Axes

queryAxes

`queryAxes()`

Set the MCS (machine coordinate system) coordinates of the interpreter to the actual coordinates of the physical axes. The MCS (machine coordinate system) coordinates are automatically translated to PCS (path coordinate system) coordinates, which are exposed to the programmer. They may also be retrieved by `frameGet(…)`. A combination of `sync()` and `queryAxes()` replaces the former `@716`-command.

- The `queryAxes()`-command considers the path axes (`X`, `Y`, `Z`), as well as the auxiliary axes (`Q1..Q5`).

Note

NOTE:

The `queryAxes()`-command should be preceeded by `sync()` to avoid unexpected behavior.

Example:

The resulting path of the following example is shown in Figure “ExampleQueryAxes”. The example assumes `M20` to be an `M`-function of type “handshake after”. The PLC is assumed to

- wait for `M20`,
- move the `Y`-axis to `20`,
- wait for completion of the movement,
- acknowledge `M20`.

The interpreter sends the line segment `N10` to the NC-channel followed by the `M`-function `M20`. Then the invocation of `sync()` blocks. The NC-channel signals the `M`-function to the PLC after the line segment `N10` has been processed. Then the PLC moves the tool from the end of segment `N10` to the beginning of segment `N20` and acknowledges `M20`. The interpreter resumes operation and invokes `queryAxes()`, which sets the internal “current point” to the endpoint of segment `N10'`. Therefore, the final block sends the line segment `N20` to the NC-channel.

```
N00
N10 G01 X40 M20
!sync();
!queryAxes();
N20 G01 X80
```

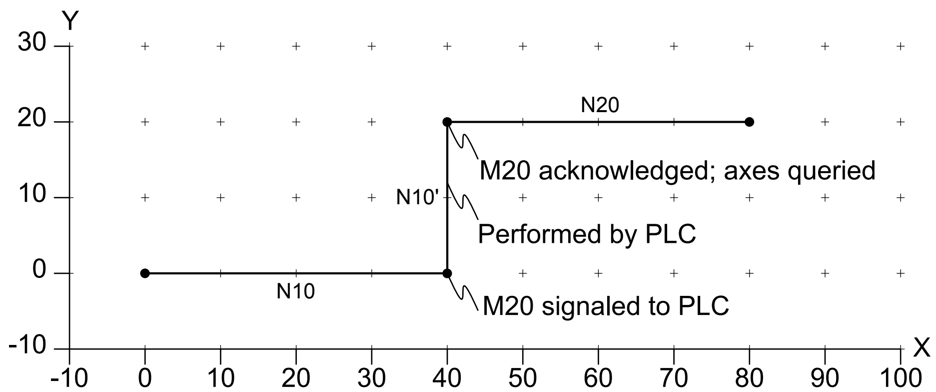


Figure “ExampleQueryAxes”.

3.6.8 Current Point

frameGet

`frameGet(x:=LReal, y:=LReal, z:=LReal, a:=LReal, b:=LReal, c:=LReal)`

Store the current frame of the PCS (program coordinate system) in `x`, `y`, `z` and `a`, `b`, `c`.

Example:

The output of the following example is shown below. The `G`-Code in the example performs a linear movement to the PCS (program coordinate system) point `[10,20,30]`. Then these coordinates are stored in `curX`, `curY`, `curZ` by `frameGet(…)`. The translation `[1,2,3]` that is pushed onto the transformation-stack leads to an adaption of the current PCS (program coordinate system) point such that the MCS (machine coordinate system) point `[10,20,30]` remains unchanged. Therefore, the subsequent call of `frameGet(…)` retrieves the PCS (program coordinate system) point `[9,18,27]`.

```
{
VAR
    curX, curY, curZ : LREAL;
END_VAR

!G01 X10 Y20 Z30 F65000

frameGet(x=>curX, y=>curY, z=>curZ);
MSG(toString('test',curX,' ',curY,' ',curZ,''));

transTranslate(1,2,3);
frameGet(x=>curX, y=>curY, z=>curZ);
MSG(toString(curX,' ',curY,' ',curZ,''));
}
```

Output:

```
10.000000 20.000000 30.000000
9.000000 18.000000 27.000000
```

qAxisGet

`qAxisGet(q1:=LReal, q2:=LReal, q3:=LReal, q4:=LReal, q5:=LReal)`

Store the current values of `Q`-axes in `q1` to `q5`. The `Q`-axes are the auxiliary axes.

3.6.9 Tool Radius Compensation

trcApproachDepartSet

`trcApproachDepartSet(approachRadius:= LReal, approachAngle:= LReal, departRadius := LReal, departAngle:= LReal)`

Configures the approach and depart behavior to use an arc of given radius and angle. If the product of radius and angle are zero, no approach or depart segment will be inserted.

The resulting configuration is used by `G41`/`G42`.

trcOffsetSet

`trcOffsetSet(offset:= LReal)`

Configures the amount of segment extension that is used to close gaps. If `offset` is zero, no extension will be performed.

The resulting configuration is used by `G41`/`G42`.

trcLimitSet

```
trcLimitSet(limit:= LReal)
```

Configures the lookahead that is used for collision elimination.

The resulting configuration is used by `G41`/`G42`.

trcParam

```
trcParam(): TrcParamType
```

Returns the current configuration as a structure value.

trcParamSet

```
trcParamSet(param:= TrcParamType)
```

Configures the tool radius compensation. This function is an alternative that summarizes `trcApproachDepartSet`, `trcOffsetSet` and `trcLimitSet`. It can be used in combination with `trcParam` to load, save and restore different TRC (tool radius compensation) configurations efficiently.

TrcParamType

```
TrcParamType
```

This structure contains all configuration parameters of the tool radius compensation. It consists of the following parameters.

```
approachradius: LREAL;
approachangle: LREAL;
departRadius: LREAL;
departAngle: LREAL;
offset: LREAL;
limit: LREAL;
```

See trcApproachDepartSet [▶ 73], trcOffsetSet [▶ 73], trcLimitSet [▶ 74] for a comprehensive description of the listed parameters.

collisionElimination

```
collisionElimination(nx:= LReal, ny:= LReal, nz:= LReal, limit:= ULInt)
```

Activates collision elimination with respect to the plane of the normal vector `nx`, `ny`, `nz`. Collisions within the projection of the path onto the plane are eliminated. Supplying a zero vector deactivates collision elimination. The `limit` parameter can be used to restrict elimination to the last `n` segments. By default, elimination is unlimited.

collisionEliminationFlush

```
collisionEliminationFlush()
```

This function can be called during active collision elimination to ignore any conflicts between the path preceeding the call and the path succeeding the call.

3.6.10 Suppression of G-Code Blocks

disableMask

```
disableMask():= LWord
```

Yields the current value of the disable mask. Note that the mask may also be set by the PLC. See disableMaskSet [▸ 75] for details.

disableMaskSet

```
disableMaskSet(mask:= LWord)
```

Sets the internal disable mask to the given value. The mask is used to suppress execution of G-Code blocks. The disable mask has 0 default value, i.e. no suppression is active by default. The mask consists of 64 bits.

In a binary notation like 2#1101 bits are numbered from right to left, starting with bit 0. For the value 2#1101 the bits 0, 2 and 3 are set by value one. The remaining bits are not set by exhibiting zero value.

Example:

The resulting path of the following example is shown in Figure “ExampleDisableMaskSet”. The disable mask is initially set to the binary value 2#1101, which is equal to the decimal value 13. The first G-Code, which is N10 in the given example, is always executed, independently of the current disable mask because there is no ‘/’-operator in the N10-line. N20 is only executed if bit 0 is not set. In the case bit 0 is set N20 is supressed, which is true in the given example. The same holds for N30, since ‘/’ is only a shorthand for ‘/0’. N40 is not supressed, since bit 1 is not set. The G-Codes N50 and N60 after disableMaskSet(0) are executed, since no bit in the disable mask is set. In contrast, the call disableMaskSet(-1) sets all bits of the mask. Consequently, the succeeding G-Codes that are prefixed with a ‘/’, N80 and N90, are disabled.

```
!disableMaskSet(2#1101);
N10 G01 X10 Y10
/0 N20 G01 X20 Y0
/ N30 G01 X30 Y0
/1 N40 G01 X40 Y10
!disableMaskSet(0);
/ N50 G01 X50 Y0
/1 N60 G01 X60 Y10
!disableMaskSet(-1);
N70 G01 X70 Y0
/1 N80 G01 X80 Y10
/2 N90 G01 X90 Y20
```

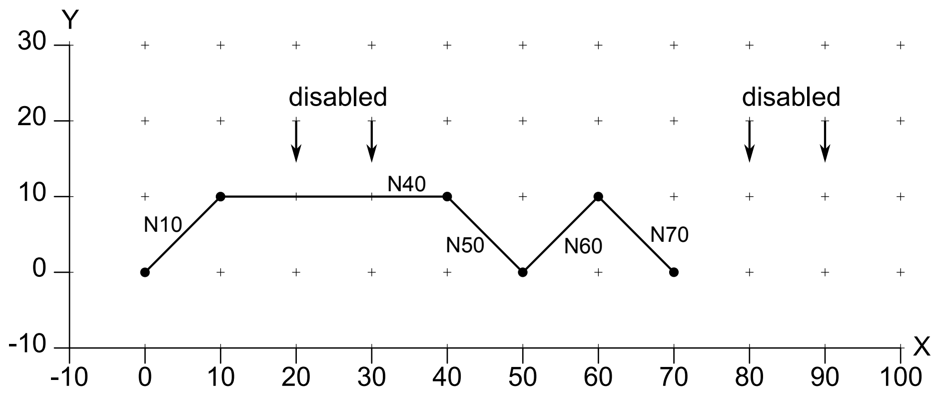


Figure “ExampleDisableMaskSet”.

3.6.11 Zero Offset Shift

zeroOffsetShiftSet

```
zeroOffsetShiftSet(g:= USInt, x:= LReal, y:= LReal, z:= LReal)
```

Sets the translation for G-Code g where g must be one of the numbers 54, 55, 56 or 57.

Example:

The resulting path of the following example is shown in Figure “ExampleZeroOffsetShiftSet”. The zero offset shift of `G54` is first set to the translation `[0,10,0]`. It gets active for `N20` and any later segment endpoints until a novel translation is applied. The second invocation of `zeroOffsetShiftSet` has an immediate effect. It applies to `N30` and any later segment endpoints until a novel translation is applied. The same holds for the last invocation. However, the block `N40` does not program the `Y`-coordinate. Therefore, the change does not become apparent for `N40`. (See section Transformations [▸ 80] for details.) Because the block `N50` programs the `Y`-coordinate, it applies the recent `[0,30,0]`-translation.

```
!zeroOffsetShiftSet(g:=54, x:=0, y:=10, z:=0);
N10 G01 X20 Y0
N20 G01 X40 Y0 G54
!zeroOffsetShiftSet(g:=54, x:=0, y:=20, z:=0);
N30 G01 X60 Y0
!zeroOffsetShiftSet(g:=54, x:=0, y:=30, z:=0);
N40 G01 X80
N50 G01 X90 Y0
```

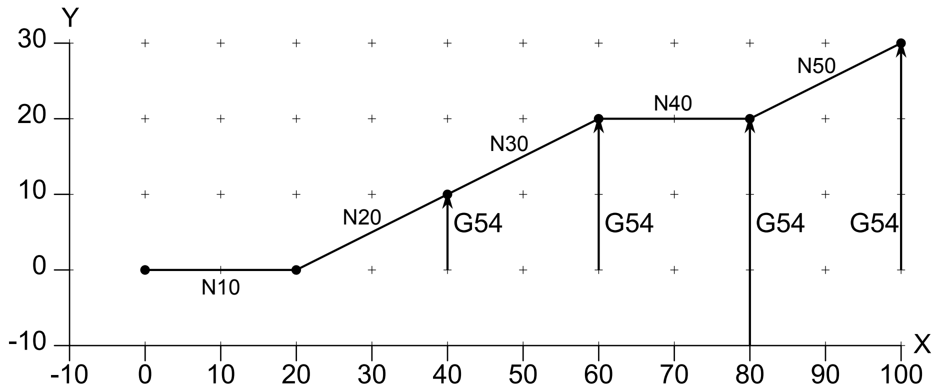


Figure “ExampleZeroOffsetShiftSet”.

3.6.12 Units

unitAngleSet

`unitAngleSet(unitAngle:= UnitAngle)`

Set the unit for angles to `unitAngle`. The default is `unitAngleDegree`. The unit for angles applies to all NC-related functions like `transRotX`. It does not apply to `ST`-standard functions like `sin`. For this reason the interpreter offers a set of NC-specific counterparts like `gSin` that consider the angle unit.

UnitAngle

Enumeration of the following values:

```
unitAngleRadian: 0…2pi
unitAngleDegree: 0…360
unitAngleGon: 0…400
unitAngleTurn: 0…1
```

.

unitLengthSet

`unitLengthSet(unitLength:= UnitLength)`

Set the unit for lengths to `unitLength`. The default is `unitLengthMillimeter`. The unit for length applies to all NC-related functions like `G01` or `zeroOffsetShiftSet(…)`.

UnitLength

Enumeration of the following values:

```
unitLengthMeter
unitLengthCentimeter
unitLengthMillimeter
unitLengthMicrometer
unitLengthNanometer
unitLengthInch
unitLengthFoot
```

.

unitTimeSet

`unitTimeSet(unitTime:= UnitTime)`

Set the unit for time to `unitTime`. The default is `unitTimeSecond`. The unit for time applies to all NC-related functions like `G04`. It does not apply to ST-standard functions like `currentLdt()`.

UnitTime

Enumeration of the following values:

```
unitTimeSecond
unitTimeMillisecond
unitTimeMicrosecond
unitTimeMinute
unitTimeHour
```

.

unitVelocitySet

`unitVelocitySet(unitLength:= UnitLength, unitTime:= UnitTime)`

Set the unit for velocity to `unitLength/unitTime`. The default is `unitLengthMillimeter/unitTimeMinute`. The unit for velocity applies to all NC-related functions. It is used by the `F`-parameter for instance.

3.6.13 Trigonometric (Unit Aware)

gSin

`gSin(angle:= LReal)`

Returns the sine of the given `angle` where the current angle unit is used to interpret the angle. (See section Units [▸ 76] for details.) The return type is `LReal`. This function is not overloaded.

gCos

`gCos(angle:= LReal)`

Returns the cosine of the given `angle` where the current angle unit is used to interpret the angle. (See section Units [▸ 76] for details.) The return type is `LReal`. This function is not overloaded.

gTan

`gTan(angle:= LReal)`

Returns the tangent of the given `angle` where the current angle unit is used to interpret the angle. (See section Units [▸ 76] for details.) The return type is `LReal`. This function is not overloaded.

gASin

`gASin(val:= LReal)`

- Returns the arcsine of `val` in the current angle unit. (See section Units [▶ 76] for details.)
- The return type is `LReal`. This function is not overloaded.
- The result lies within the interval `[-c/4,c/4]` where `c` is the angle of a full circle in the current angle unit.

Note	**CONSTRAINT:** The variable `val` must reside within the interval `[-1,1]`.

gACos

`gACos(val:= LReal)`

- Returns the arccosine of `val` in the current angle unit. (See section Units [▶ 76] for details.)
- The return type is `LReal`. This function is not overloaded.
- The result lies within the interval `[0,c/2]` where `c` is the angle of a full circle in the current angle unit.

Note	**CONSTRAINT:** The variable `val` must reside within the interval `[-1,1]`.

gATan

`gATan(val:= LReal)`

- Returns the arctangent of `val` in the current angle unit. (See section Units [▶ 76] for details.)
- The return type is `LReal`. This function is not overloaded.
- The result lies within the interval `[-c/4,c/4]` where `c` is the angle of a full circle in the current angle unit.

gATan2

`gATan2(y:= LReal, x:= LReal)`

- Returns the arctangent of `y/x` in the current angle unit. (See section Units [▶ 76] for details.)
- The return type is `LReal`. This function is not overloaded.
- The result lies within the interval `[-c/2,c/2]` where `c` is the angle of a full circle in the current angle unit.

3.6.14 Feed Mode

feedModeSet

`feedModeSet(feedMode:= FeedModeType)`

FeedModeType

Enumeration of the following values:

```
fmContour
fmInternalRadius
fmToolCenterPoint
```

.

3.6.15 Feed Interpolation

feedInterpolationSet

`feedInterpolationSet(feedInterpolation:= FeedInterpolationType)`

FeedInterpolationType

`FeedInterpolationType`

Enumeration of the following values:

```
fiConstant
fiLinear
```

.

3.6.16 Streaming of Large G-Code Files

runFile

`runFile(path:= string)`

The size of files that can be executed employing the `GST`-interpreter is limited. However, sometimes it is required to execute large files that may have been created e.g. by a `CAD`-program. Therefore, the user has the possibility to execute filestreams of native `G`-Code.

Executes the plain `G`-Code that is contained in the `G`-Code file given by `path`. The function call returns after all lines in the supplied file have been processed. The function is intended for streaming large `G`-Code files to the NC-kernel efficiently.

Note	**Native `G`-Code: No Structured Text Allowed** Note that the supplied `G`-Code file must not contain any `ST`-elements, but only plain `G`-Code.

A `G`-Code filestream from file `'myNativeGCodeFile.gcode'` can be called from a `GST`-program lining up the following syntax:

```
!runfile('myNativeGCodeFile.gcode');
```

.

3.6.17 Vertex Smoothing

smoothingSet

`smoothingSet(mainType:= SmoothingMainType, subType:= SmoothingSubType, value:= LReal)`

Sets the vertex smoothing behavior according to the given parameters.

SmoothingMainType

Enumeration of the following values:

```
smoothingNone
smoothingParabola
smoothingBiquadratic
smoothingBezier3
smoothingBezier5
```

.

SmoothingSubType

Enumeration of the following values:

```
smoothingRadius
smoothingDistance
smoothingAdaptive
```

.

3.6.18 Dynamic Override

dynOverrideSet

`dynOverrideSet(value:= LReal)`

Set the dynamic override of axes to the given `value`. The factor must lie in the range `0` to `1`.

3.6.19 Center Point Reference of Circles

circleCenterReferenceSet

`circleCenterReferenceSet(value:= ReferenceType)`

- Sets the center reference type for circles that are programmed by `G02`/`G03` using a center point, whose definition involves the `i`,`j`,`k`-parameters.
- For `referenceAbsolute` the center point of the circle is defined by the supplied `i`,`j`,`k`-vector.
- For `referenceRelative` the center point is defined by the sum of the circle starting-point and the supplied `i`,`j`,`k`-vector. This is the default and usual behavior of `G`-Code.

ReferenceType

Enumeration of the following values:

```
referenceAbsolute
referenceRelative
```

.

3.7 Transformations

The relation between the MCS (machine coordinate system) and the PCS (program coordinate system) is defined by the effective transformation T. T is the concatenation of the transformations T_Z, T_U and T_T ($T = T_Z * T_U * T_T$). Note that the order of concatenation is significant for the transformations do not commute in general. The transformation T_Z represents a (historical) zero offset shift, the transformation T_U represents a user defined transformation and the transformation T_T represents a tool transformation. They are described in detail later.

Figure “TransformationsTzTuTt” visualizes the relation between the MCS (machine coordinate system) and the PCS (program coordinate system):

- T_Z is defined to be a translation by `[20,20,0]`,

- T_U is a combination of the translation `[30,-10,0]`, followed by a rotation by `45` degree around the `Z`-axis,
- T_T is a translation by `[0,-10,0]`.

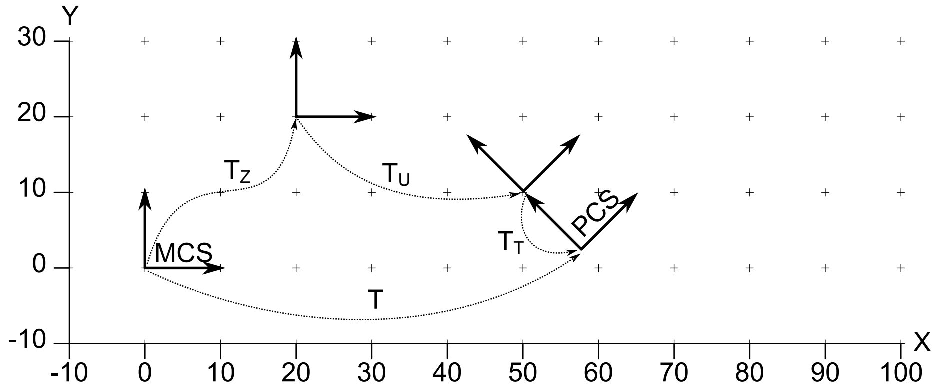


Figure “TransformationsTzTuTt”.

3.7.1 Modification of the Effective Transformation T and its Effect

Most `G`-Codes define only the destination point of a movement. Therefore, the interpreter maintains the current position of the tool. This point can be represented in MCS (machine coordinate system) coordinates and PCS (program coordinate system) coordinates while the equation `CurrentPointMCS = T * CurrentPointPCS` holds. In contrast to the previous implementation, this transformation equation also holds after a modification of T. This behavior is accomplished by adapting the `CurrentPointPCS`. The MCS (machine coordinate system) point is not adapted, as this would affect the machine. This behavior may be summarized roughly as: When the active transformation is changed, the current PCS (program coordinate system) point is adapted in a way that the modification shows no effect.

Example:

After `N10` the coordinates of the current PCS (program coordinate system) and MCS (machine coordinate system) point are `[20,10,80]`, since no transformation is active. The translation changes the current PCS (program coordinate system) point to `[28,7,84]`. Applying the translation on this point yields the unchanged MCS (machine coordinate system) point `[20,10,80]`. Hence, the translation shows no effect, although it is active. The block `N20` programs a movement to the PCS (program coordinate system) point `[25,7,10]`, which is mapped to the MCS (machine coordinate system) coordinate `[17,10,6]`. After the invocation of `transPop()` the current PCS (program coordinate system) point is set to the current MCS (machine coordinate system) point.

```
N10 G01 X20 Y10 Z80
!transTranslate(-8,3,-4);
N20 G01 X25 Z10
!transPop();
```

Example:

If the user wants the PCS (program coordinate system) point to remain unchanged, he has to retrieve and program it, as shown in the following code. However, the desire for an unchanged PCS (program coordinate system) point typically indicates a bad programming style. Actually, there should be no need for the following code.

```
{
VAR
pcsX, pcsY, pcsZ : LREAL;
END_VAR

// … G-Code …

frameGet(x=>pcsX,y=>pcsY,z=>pcsZ);
// … modify transformations …
```

```
 !G01 x=pcsX y=pcsY z=pcsZ
}
```

3.7.2 Components of the Effective Transformation T

Zero Offset Shift T_Z

The T_Z-transformation is affected by certain `G`-Codes. It has no effect if `G53` is active. Otherwise, T_Z is the combination of the three translations T_{Z58}, T_{Z59} and one of {T_{Z54}, …, T_{Z57}}. The former two translations are set via the `G`-Codes `G58` and `G59`. The latter translation is selected by the `G`-Codes `G54` to `G57`. One translation is associated with each of these 4 `G`-Codes. It can be set by the PLC or using the `ST`-function `zeroOffsetShiftSet`.

Tool Transformation T_T

T_T is defined by the currently selected tool. It has no effect if tool `0` (`D0`) is selected. Otherwise, it is a translation by `[offsetX, offsetY, offsetZ] + (length+lengthAdd) * D` where `D` is the normal of the current workingplane.

Userdefined Transformation T_U

T_U is defined by a stack of transformations. The stack of depth `N` contains elementary transformations T_{U1}, T_{U2}, …, $T_{U<N>}$ where $T_{U<N>}$ is the topmost transformation. Initially, the stack is empty. The userdefined transformation is the concatenation of these elementary transformations $T_U = T_{U1} * T_{U2} * \ldots * T_{U<N>}$. Note that the order is significant for the transformations do not commute in general. If the stack is empty, T_U is the identity transformation, which has no transformation effect.

3.7.3 Applying Transformations

A transformation is pushed onto the stack by the following `ST`-functions. The transformation pushed recently will be the topmost transformation on the transformation stack. When a transformation is pushed onto the transformation stack, the stack depth is increased by one and T_U is adapted accordingly.

```
transTranslate(x:= LREAL, y:= LREAL, z:= LREAL)
transRotX(angle:= LREAL)
transRotY(angle:= LREAL)
transRotZ(angle:= LREAL)
transMirrorX()
transMirrorY()
transMirrorZ()
transScale(factor:= LREAL)
```

3.7.4 Revoking Transformations

The function `transPop()` removes the topmost transformation from the transformation stack. When `transPop()` removes a transformation from the transformation stack, the stack depth is reduced by one and T_U is adapted accordingly. Commonly, the `transPop()`-function is used to revoke a temporary transformation.

Example:

In the following example the translation is applied to `N10`, `N20` and `N30`. The rotation is only applied to `N20` as it is revoked by `transPop()`. Figure “ExampleRevokingTransformations” shows the resulting MCS (machine coordinate system) path. Note that the rotation center is `[20,0,0]` in MCS (machine coordinate system), which is the origin in PCS (program coordinate system) after the preceeding translation.

```
!transTranslate(20,0,0);
N10 X10 Y0
!transRotZ(90);
N20 X20 Y0
```

```
 !transPop();
N30 X30 Y0
!transPop();
```

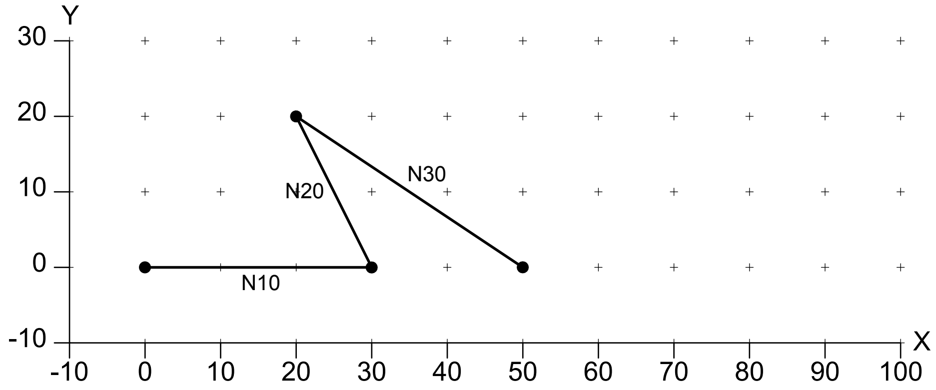


Figure “ExampleRevokingTransformations”.

3.7.5 Restoration of Stack

The function `transDepth()` yields the current depth of the stack. The function `transRestore(depth)` removes transformations from the stack until the given depth is reached. Typically, the two functions are combined to save and restore the state of the transformation stack.

It is good programming style to do this saving and restoring in the context of userdefined `ST`-functions.

Example:

Initially, within the following function the depth of the stack is stored in the variable `depth`. At the end of the function the initial state is restored by `transRestore`. Note that restoration only works properly if the stack depth does not fall below `depth` within the function. Instead of using `transDepth()` and `transRestore()` the stack depth could also be restored using `transPop()`. However, it may become cumbersome to keep pushing and popping of transformations synchronous, especially if transformations are pushed conditionally.

```
{
FUNCTION draw
VAR
    depth : UINT;
END_VAR
    depth := transDepth();
transTranslate(10,0,0);
    // … G-Code …
    transRotZ(45);
    // … G-Code …
    transMirrorX();
    // … G-Code …
    transRestore(depth);
END_FUNCTION
}
```

3.8 Error Reporting

Efficient development of `CNC`-programs requires decent support by development tools. This support includes proper reporting of programming errors for both, compile-time errors and runtime errors. An error message should point directly to the source code the error originates from and give a precise description of the circumstances under that the error occured (dynamic data). Such individual error messages help a developer substantially to fix errors in short time. The `GST`-interpreter yields such error messages, as described in the following texts.

3.8.1 Error Messages

In case of an error the interpreter produces a descriptive error message. An error message consists of a source code coordinate and a description. The source code coordinate links the error to its origin in the `GST`-program. It defines a range of source code stretching from the first character of the code range to the last character of the code range. Both, first and last character, are defined by their file, line and column. Note that the last character actually points to the first character behind the range, which is a common technical convention.

Example:

In the following example an integer variable `i` is declared and initialized. The initialization uses a floatingpoint literal. Since an implicit conversion from floatingpoint to integer is not allowed in `ST`, the interpreter produces the descriptive error message given below when the program is loaded. The error message does not only report that a type-error has occured, it also gives the precise position: File `aaa.gst`, line `3`, column `14` to `17`. This code range displays the literal ‘`1.5`’. In addition, the programmed type (`real`) and the expected type (`int`) are reported. With such a detailed error message bugs can be fixed by the developer easily.

```
{
VAR
    i : int := 1.5;
END_VAR
}
```

Error message:

```
aaa.gst: 3.14-3.17: Invalid implicit conversion from type
'<real literal>' to 'int'.
```

3.8.2 Compile-Time Errors and Runtime Errors

Errors may occur during program loading (so called compile-time errors) or during program execution (so called runtime errors). Fortunately, most errors are detected at compile-time. This detection includes missing files, syntax errors, type errors and unexpected identifiers. The developer gets feedback immediately when he tries to load the program. Thus, a part of unexpected failures during machining is avoided.

However, there are also errors that, by their nature, cannot be detected at compile-time. For instance, this circumstance includes a division by zero, since the divisor may be computed dynamically. If a runtime error occurs, the interpreter is stopped safely and an error message is produced. A runtime error message is similar to a compile-time error message. It even includes a reference to the pertinent source code.

Example:

In the following example the `FOR`-loop contains a division of `10` by the loop variable `i`. Since the variable `i` is iterated from `-3` to `3`, this program leads to an error during the 4th iteration, when `i` has the value `0`. This error is detected at runtime and stops the interpreter. The error message shown below is reported. It points precisely to the code ‘`10/i`’ in the example.

FILE aaa.gst:

```
{
VAR
    i, j : int;
END_VAR

FOR i := -3 TO 3 DO
    j := j + 10/i;
END_FOR;
}
```

Error message:

```
aaa.gst: 7.12-7.16: Division by zero
```

Example:

At runtime the interpreter also performs checking of array bounds. Consequently, invalid indices do not result in unpredictable and typically fatal crashes. The runtime error message precisely defines location and origin of the error at ‘`idx`’, reports the erroneously supplied index (`20`) and the valid index range (`10..19`).

FILE aaa.gst:

```
{
VAR
    idx : INT;
    a : ARRAY [10..19] OF INT;
END_VAR

FOR idx := 10 TO 20 DO
    a[idx] := i;
END_FOR;
}
```

Error message:

```
aaa.gst: 8.5-8.8: Out of bounds. 20 exceeds range 10..19.
```

3.8.3 Errors in G-Code

Error reporting is also performed for `G`-Code. This reporting includes compile-time errors and runtime errors. Runtime error reporting includes invalid use of `G`-Code, e.g. like a bad definition for a circle.

Example:

In the following example the value of a string variable `str` is assigned to the address letter `X` of the `G`-Code block. As before the position of the error is precisely identified at ‘`=str`’ in the code. In addition, the programmed type and the expected type are reported.

FILE aaa.gst:

```
{
VAR
    str : string := 'Hello World';
END_VAR
}

G01 X=str F6000
G01 Y100
```

Error message:

```
7.5-7.9: Invalid implicit conversion from type 'string[255]'
to 'lreal'
```

Example:

In the following example a sequence of circular arcs is processed by a `FOR`-loop. The radius of the arc is `4`. The distance between the starting-point and endpoint of the arc is successively increased within each iteration. During the 9th iteration the distance exceeds the circle diameter of `8`. The reported error message identifies the origin `G2 X=i*10+i U4` and gives information about the diameter and distance between starting-point and endpoint.

FILE aaa.gst:

```
{
VAR
    i : INT;
END_VAR

!G00 X0 Y0 Z0
FOR i := 1 TO 10 DO
    !G01 X=i*10
```

```
    !G02 X=i*10+i U4
END_FOR;
}
```

Error Message:

```
aaa.gst: 9.4-10.1: Invalid definition of circle. Distance
between start-point and end-point (=9.000000) is larger than
diameter (=8.000000).
```

3.8.4 Preprocessing

During preprocessing `#include`-directives are replaced by the contents of the referenced files. Care has been taken to maintain information about the origin of source code properly. Therefore, an error that is caused by code in an included file will refer to that included file and not to the result of preprocessing, as a simple implementation would do.

Example:

In the following example the file `aaa.gst` includes the file `bbb.gst`. In the latter file the variables `i` and `j` are used in `G`-Codes. Variable `i` is declared at the beginning of `aaa.gst`, but `j` is not. Therefore, the error message below is issued. As you can see it references the use of variable `j` in file `bbb.gst` properly.

FILE aaa.gst:

```
{
VAR
    i : INT;
END_VAR
}

G00 X0 Y0 Z0

#include "bbb.gst"

G00 X100
```

FILE bbb.gst:

```
G01 X=i
G01 Y=j
G01 Z100
```

Error message:

```
bbb.gst: 2.6-2.7: Undeclared variable or enumeration value
'j'
```

3.9 General Command Overview

Interpolations

Command	Description	Modal or Non-modal	Default
G00 [▶ 35]	Interpolation mode: Linear. Applying maximum velocity ignoring programmed velocity. Reset by `G01`, `G02`, `G03`.	Modal.	Default.
G01 [▶ 35]	Interpolation mode: Linear. Applying programmed velocity. Reset by `G00`, `G02`, `G03`.	Modal.	No.
G02 [▶ 35]	Clockwise interpolation mode: Circular or helical. Applying current velocity. Reset by `G00`, `G01`, `G03`.	Modal.	No.
G03 [▶ 37]	Counterclockwise interpolation mode: Circular or helical. Applying current velocity. Reset by `G00`, `G01`, `G02`.	Modal.	No.
G04 [▶ 37]	Defines a dwell time, i.e. suspends machining for a given duration.	Nonmodal.	No.
G09 [▶ 37]	Accurate stop.	Nonmodal.	No.

Workingplane Selection

Command	Description	Modal or Non-modal	Default
G17 [▶ 37]	Selects `XY`-plane as workingplane.	Modal.	Default.
G18 [▶ 37]	Selects `ZX`-plane as workingplane.	Modal.	No.
G19 [▶ 37]	Selects `YZ`-plane as workingplane.	Modal.	No.

Deactivation and Activation of Tool Radius Compensation

Command	Description	Modal or Non-modal	Default
G40 [▶ 37]	Deactivates Tool Radius Compensation (TRC).	Modal.	Default.
G41 [▶ 38]	Activates Tool Radius Compensation (TRC). Left.	Modal.	No.
G42 [▶ 39]	Activates Tool Radius Compensation (TRC). Right.	Modal.	No.

Set, Deactivate and Activate Zero-Offset-Shift Translations

Command	Description	Modal or Non-modal	Default
G53 [▸ 39]	Deactivates any zero-offset-shift translation.	Modal.	Default.
G54..G57 [▸ 39]	Activates the zero-offset-shift associated with the given G-Code. Activates the translation G58 and G59.	Modal.	No.
G58, G59 [▸ 39]	Sets the translation associated with the given G-Code.	Modal.	No.

Accurate Stop

Command	Description	Modal or Non-modal	Default
G60 [▸ 40]	Accurate stop.	Modal.	No.

Set Unit for Length

Command	Description	Modal or Non-modal	Default
G70 [▸ 40]	Sets the unit for lengths to inch. Does not affect the unit for velocity.	Modal.	No.
G71 [▸ 40]	Sets the unit for lengths to millimeter. Does not affect the unit for velocity.	Modal.	Default.

Switch to Absolute or Relative Coordinates

Command	Description	Modal or Non-modal	Default
G90 [▸ 41]	Switches to absolute programming.	Modal.	Default.
G91 [▸ 41]	Switches to incremental programming.	Modal.	No.

Set Unit for Length and Speed

Command	Description	Modal or Non-modal	Default
G700 [▸ 42]	Sets the unit for lengths to inch. Also applies to the interpretation of velocity.	Modal.	No.
G710 [▸ 42]	Sets the unit for lengths to millimeter. Also applies to the interpretation of velocity.	Modal.	Default.

IJK

Command	Description	Modal or Non-modal	Default
IJK [▶ 42]	Center point is `currentPoint + [vx,vy,vz]`. Current length unit is used for `vx,vy,vz`. `I` used by `G4` defines a duration.	Modal.	Default: Center point is `currentPoint + [0,0,0]`. The `IJK`-parameters are optional.

M-Functions

Command	Description
M [▶ 42]	Triggers the `M`-function `v`. The timing behavior depends on the definition of `v` in the System Manager of TwinCAT. There must not be more than one `M`-function of type handshake in a block.
M2	Predefined `M`-function. Signals program end.
M30	Predefined `M`-function. Signals program end.

Tool Orientation and Auxiliary Axes

Command	Description	Modal or Non-modal	Default
P [▶ 43]	Switches tool orientation.	Modal.	No.
Q [▶ 44]	Sets label for auxiliary axis.	Modal.	No.

Set Block Number

Command	Description
N [▶ 45]	Block number.

Set Radius

Command	Description	Modal or Non-modal	Default
U [▶ 45]	Sets the radius within the context of `G02` or `G03` to `|v|`.	Modal.	No.

Set Cartesian Coordinate

Command	Description
X [▶ 45]	Sets the `X`-coordinate of the next point to `v`. Uses current length unit for `v`.
Y [▶ 45]	Sets the `Y`-coordinate of the next point to `v`. Uses current length unit for `v`.
Z [▶ 46]	Sets the `Z`-coordinate of the next point to `v`. Uses current length unit for `v`.

Set Orientation Angle

Com-mand	Description	Modal or Non-modal	Default
A [▶ 46]	Sets the `A`-angle of the next orientation to `v`. Uses current length unit for `v`.	Nonmodal, but may influence succeeding blocks.	No.
B [▶ 46]	Sets the `B`-angle of the next orientation to `v`. Uses current length unit for `v`.	Nonmodal, but may influence succeeding blocks.	No.
C [▶ 46]	Sets the `C`-angle of the next orientation to `v`. Uses current length unit for `v`.	Nonmodal, but may influence succeeding blocks.	No.

Trigonometric

Command	Description
SIN [▶ 56]	Returns the sine of `x`; `x` in radians.
COS [▶ 56]	Returns the cosine of `x`; `x` in radians.
TAN [▶ 56]	Returns the tangent of `x`; `x` in radians.
ASIN [▶ 57]	Returns the arc sine of `x`; `x` in radians.
ACOS [▶ 57]	Returns the arc cosine of `x`; `x` in radians.
ATAN [▶ 57]	Returns the arc tangent of `x`; `x` in radians.
ATAN2 [▶ 57]	Returns the arc tangent of `y/x`; `y/x` in radians.

Arithmetic

Command	Description
ABS [▶ 55]	Returns the absolute value of `x`.
SQRT [▶ 55]	Returns the square root of `x`.
LN [▶ 56]	Returns the natural logarithm of `x`.
LOG [▶ 56]	Returns the decimal logarithm of `x`.
EXP [▶ 56]	Returns `e` raised to the power of `x`.
ADD [▶ 57]	Returns the sum of all parameters.
MUL [▶ 57]	Returns the product of all parameters.
SUB [▶ 58]	Returns the difference `x-y`.
DIV [▶ 58]	Returns the quotient `x/y`.
MOD [▶ 58]	Returns the remainder of the integer division `x/y`.
EXPT [▶ 58]	Returns `x` raised to the power of `y`.

Shift

Command	Description
SHL [▶ 59]	Returns the bitstring `x` shifted left by `y` bits.
SHR [▶ 59]	Returns the bitstring `x` shifted right by `y` bits.

Rotation

Command	Description
ROL [▸ 59]	Returns the bitstring `x` rotated left by `y` bits.
ROR [▸ 59]	Returns the bitstring `x` rotated right by `y` bits.

Logical Operations

Command	Description
AND [▸ 60]	Returns the bitwise Logical And of all parameters.
OR [▸ 60]	Returns the bitwise Logical Or of all parameters.
XOR [▸ 60]	Returns the bitwise Logical Exclusive Or of all parameters.
NOT [▸ 60]	Returns the bitwise complement of `x`.

Selection (Conditional Expressions)

Command	Description
SEL [▸ 60]	Returns `x1` if `cond` is `FALSE`, and `x2` otherwise.
MUX [▸ 60]	Returns `x<select>`.

Min, Max and Limit

Command	Description
MAX [▸ 61]	Returns the maximum of all parameters.
MIN [▸ 61]	Returns the minimum of all parameters.
LIMIT [▸ 61]	Returns `in` if it lies in the interval `[min,max]`. Otherwise, the violated bound (`min` or `max`) is returned.

Comparison

Command	Description
GT [▸ 61]	Returns `TRUE` if `x` is larger than `y`.
GE [▸ 61]	Returns `TRUE` if `x` is not smaller than `y`.
EQ [▸ 61]	Returns `TRUE` if `x` and `y` are equal.
LE [▸ 62]	Returns `TRUE` if `x` is not larger than `y`.
LT [▸ 62]	Returns `TRUE` if `x` is smaller than `y`.
NE [▸ 62]	Returns `TRUE` if `x` and `y` are not equal.

Comparison

Command	Description
The toString-Function [▸ 62]	Converts and concatenates the given arguments to one string.
Msg [▸ 62]	Send the given message to the message list of TwinCAT.

Transformations

Command	Description
transRotX/Y/Z [▶ 63] transRotX/Y/Z [▶ 63] transRotX/Y/Z [▶ 63]	Rotation around the respective axis by the given angle in the user-defined angle unit.
transRotA [▶ 64]	Rotate around vector `[x,y,z]` by the given `angle`.
transMirrorX/Y/Z [▶ 65] transMirrorX/Y/Z [▶ 65] transMirrorX/Y/Z [▶ 65]	Mirror with respect to the `X`-direction, `Y`-direction or `Z`-direction relative to the origin of the current PCS.
transScale [▶ 65]	Scales the coordinate system by the `factor` in the `X`-dimension, `Y`-dimension and `Z`-dimension.
transTranslate [▶ 66]	Translate by vector `[x,y,z]`.
transPop [▶ 66]	Pops a transformation from the stack of transformations.
transDepth [▶ 67]	Yields the depth of the stack of transformations, i.e. the number of active transformations.
transRestore [▶ 67]	Reduces the stack of transformations to the given depth.

Movement

Command	Description
moveCircle3d [▶ 68]	Move circular by rotating around the center `cx,cy,cz` and the normal vector `nx,ny,nz` by the given `angle`. If `height` is nonzero, a helix is described. The rotation is performed according to the right hand rule.

Centerpoint Correction

Command	Description
centerpointCorrectionSet [▶ 69]	Activates the centerpoint correction for circles. Used for circles that are defined by centerpoint programming.
centerpointCorrectionLimitSet [▶ 69]	Sets the precision limit for the centerpoint of circles.

Tools

Command	Description
toolParamSet [▶ 69]	Set a parameter of the tool `tidx` (1..255) to `val`. The parameter is identified by `col` (0..15).
toolParam [▶ 70]	Yields the given tool parameter.
toolSet [▶ 70]	Set all parameters of a tool.
ToolType [▶ 71]	Enumeration of tool types.

Synchronization

Command	Description
sync [▶ 71]	Synchronizes the interpreter with the associated NC-channel.
wait [▶ 71]	Waits for a `GoAhead`-signal from the PLC.

Query of Axes

Command	Description
queryAxes [▶ 72]	Set the MCS coordinates of the interpreter to the actual coordinates of the physical axes.

Current Point

Command	Description
frameGet [▶ 73]	Store the current frame of the PCS in `x`, `y`, `z` and `a`, `b`, `c`.
qAxisGet [▶ 73]	Store the current values of `Q`-axes `q1` to `q5`.

Tool Radius Compensation

Command	Description
trcApproachDepartSet [▶ 73]	Configures the approach and depart behavior to use an arc of given radius and angle.
trcOffsetSet [▶ 73]	Configures the amount of segment extension that is used to close gaps.
trcLimitSet [▶ 74]	Configures the lookahead that is used for collision elimination.
trcParam [▶ 74]	Returns the current configuration as a structure value.
trcParamSet [▶ 74]	Configures the tool radius compensation. Summarizes `trcApproachDepartSet`, `trcOffsetSet` and `trcLimitSet`.
TrcParamType [▶ 74]	Structure containing all configuration parameters of the tool radius compensation.
collisionElimination [▶ 74]	Activates collision elimination with respect to the plane of the normal vector `nx`, `ny`, `nz`.
collisionEliminationFlush [▶ 74]	To ignore conflicts between the path preceding the call and the path succeeding the call.

Command	Description	Modal or Non-modal	Default
G40 [▶ 37]	Deactivates Tool Radius Compensation (TRC).	Modal.	Default.
G41 [▶ 38]	Activates Tool Radius Compensation (TRC). Left.	Modal.	No.
G42 [▶ 39]	Activates Tool Radius Compensation (TRC). Right.	Modal.	No.

Suppression of G-Code Blocks

Command	Description
disableMask [▶ 74]	Yields the current value of the disable mask.
disableMaskSet [▶ 75]	Sets the internal disable mask to the given value.

Zero Offset Shift

Command	Description
zeroOffsetShiftSet [▶ 75]	Sets the translation for `G`-Code `g` where `g` must be one of the numbers `54`, `55`, `56` or `57`.

Units

Command	Description
unitAngleSet [▸ 76]	Set the unit for angles to `unitAngle`.
UnitAngle [▸ 76]	Enumeration of unit angles.
unitLengthSet [▸ 76]	Set the unit for lengths to `unitLength`.
UnitLength [▸ 77]	Enumeration of unit lengths.
unitTimeSet [▸ 77]	Set the unit for time to `unitTime`.
UnitTime [▸ 77]	Enumeration of unit times.
unitVelocitySet [▸ 77]	Set the unit for velocity to `unitLength/unitTime`.

Trigonometric (Unit Aware)

Command	Description
gSin [▸ 77]	Returns the sine of the given `angle` where the current angle unit is used to interpret the angle.
gCos [▸ 77]	Returns the cosine of the given `angle` where the current angle unit is used to interpret the angle.
gTan [▸ 77]	Returns the tangent of the given `angle` where the current angle unit is used to interpret the angle.
gASin [▸ 78]	Returns the arc sine of `val` in the current angle unit.
gACos [▸ 78]	Returns the arc cosine of `val` in the current angle unit.
gATan [▸ 78]	Returns the arc tangent of `val` in the current angle unit.
gATan2 [▸ 78]	Returns the arc tangent of `y/x` in the current angle unit.

Feed Mode

Command	Description
feedModeSet [▸ 78]	Set the feed mode.
FeedModeType [▸ 78]	Enumeration of feed mode types.

Feed Mode

Command	Description
feedInterpolationSet [▸ 79]	Set feed interpolation.
FeedInterpolationType [▸ 79]	Enumeration of feed interpolation types.

Streaming of Large G-Code Files

Command	Description
runFile [▸ 79]	Executes the plain `G`-Code that is contained in the `G`-Code file given by `path`.

Vertex Smoothing

Command	Description
smoothingSet [▸ 79]	Set the vertex smoothing behavior.
SmoothingMainType [▸ 79]	Enumeration of smoothing main types.
SmoothingSubType [▸ 80]	Enumeration of smoothing sub types.

Dynamic Override

Command	Description
dynOverrideSet [▸ 80]	Set the dynamic override of axes to the given `value`.

Center Point Reference of Circles

Command	Description
circleCenterReferenceSet [▸ 80]	Sets the center reference type for circles that are programmed by `G02/G03`.
ReferenceType [▸ 80]	Enumeration of reference types.

3.10 Comparative Command Overview

Command `NCI` Interpreter	Command `GST` Interpreter	Modal or Nonmodal	Description
ANG [▸ 128]		Nonmodal.	Contour line programming (angle).
AROT [▸ 129]	transRotA [▸ 64], transRotX/Y/Z [▸ 63], transRotX/Y/Z [▸ 63], transRotX/Y/Z [▸ 63]	Modal.	Rotation additive.

Command NCI Interpreter	Command GST Interpreter	Modal or Nonmodal	Description
CalcInvRot [▸ 129]		Nonmodal.	Calculates the inverse rotation of a vector.
CalcRot [▸ 129]		Nonmodal.	Calculates the rotation of a vector.
CDOF [▸ 173]	collisionElimination [▸ 74] Supplying a zero vector.	Modal.	Bottleneck detection off.
CDON [▸ 173]	collisionElimination [▸ 74] Supplying a nonzero vector.	Modal.	Bottleneck detection on.
	collisionEliminationFlush [▸ 74]		This function can be called during active collision elimination to ignore any conflicts between the path preceding the call and the path succeeding the call.
CFC [▸ 172]	feedInterpolationSet [▸ 79]	Modal.	Constant velocity at the contour.
CFIN [▸ 172]	feedModeSet [▸ 78]	Modal.	Constant velocity in the interior circle.
CFTCP [▸ 172]	feedModeSet [▸ 78]	Modal.	Constant velocity of the tool center point.
CIP [▸ 119]	moveCircle3d [▸ 68]	Nonmodal.	Circular interpolation. Move circular by rotating around the center `cx,cy,cz` and the normal vector `nx,ny,nz` by the given `angle`.
CPCOF [▸ 119]	centerpointCorrectionSet [▸ 69]	Modal.	Center point correction off.
CPCON [▸ 119]	centerpointCorrectionSet [▸ 69]	Modal.	Center point correction on.

Command NCI Interpreter	Command GST Interpreter	Modal or Nonmodal	Description
DelDTG [▸ 140]		Nonmodal.	Delete distance to go.
DYNOVR [▸ 156]	dynOverrideSet [▸ 80]	Modal.	Dynamic override.

Command NCI Interpreter	Command GST Interpreter	Modal or Nonmodal	Description
FCONST [▸ 123]	feedInterpolationSet [▸ 79]	Modal.	Constant feed programming.
FLIN [▸ 123]	feedInterpolationSet [▸ 79]	Modal.	Linear feed programming.

Command NCI Interpreter	Command GST Interpreter	Modal or Non-modal	Description
G00 [▸ 118]	G00 [▸ 35]	Modal.	Rapid traverse.
G01 [▸ 119]	G01 [▸ 35]	Modal.	Straight line interpolation.
G02 [▸ 119]	G02 [▸ 35]	Modal.	Clockwise circular interpolation.
G03 [▸ 119]	G03 [▸ 37]	Modal.	Counterclockwise circular interpolation.
G04 [▸ 122]	G04 [▸ 37]	Nonmodal.	Dwell time.
G09 [▸ 123]	G09 [▸ 37]	Nonmodal.	Accurate stop.

Command NCI Interpreter	Command GST Interpreter	Modal or Non-modal	Description
G17 [▸ 111]	G17 [▸ 37]	Modal.	Plane selection XY.
G18 [▸ 111]	G18 [▸ 37]	Modal.	Plane selection ZX.
G19 [▸ 111]	G19 [▸ 37]	Modal.	Plane selection YZ.

Command NCI Interpreter	Command GST Interpreter	Modal or Non-modal	Description
G40 [▸ 167]	G40 [▸ 37]	Modal.	No miller/ cutter radius compensation.
G41 [▸ 167]	G41 [▸ 38]	Modal.	Miller/ cutter radius compensation left.
G42 [▸ 167]	G42 [▸ 39]	Modal.	Miller/ cutter radius compensation right.

Command NCI Interpreter	Command GST Interpreter	Modal or Non-modal	Description
G53 [▸ 124]	G53 [▸ 39]	Modal.	Zero shift suppression.
G54 [▸ 124]	G54..G57 [▸ 39]	Modal.	1st adjustable zero shift.
G55 [▸ 124]	G54..G57 [▸ 39]	Modal.	2nd adjustable zero shift.
G56 [▸ 124]	G54..G57 [▸ 39]	Modal.	3rd adjustable zero shift.
G57 [▸ 124]	G54..G57 [▸ 39]	Modal.	4th adjustable zero shift.
G58 [▸ 124]	G58, G59 [▸ 39]	Modal.	1st programmable zero shift.
G59 [▸ 124]	G58, G59 [▸ 39]	Modal.	2nd programmable zero shift.

Command NCI Interpreter	Command GST Interpreter	Modal or Non-modal	Description
G60 [▸ 123]	G60 [▸ 40]	Modal.	Accurate stop.

Command NCI Interpreter	Command GST Interpreter	Modal or Non-modal	Description
G70 [▸ 113]	G70 [▸ 40]	Modal.	Dimensions in inch.
G71 [▸ 113]	G71 [▸ 40]	Modal.	Dimensions metric.
G74 [▸ 117]		Nonmodal.	Programmed traverse to reference point.

Command NCI Interpreter	Command GST Interpreter	Modal or Non-modal	Description
G90 [▶ 111]	G90 [▶ 41]	Modal.	Reference dimension notation.
G91 [▶ 111]	G91 [▶ 41]	Modal.	Incremental dimension notation.

Command NCI Interpreter	Command GST Interpreter	Modal or Non-modal	Description
G700 [▶ 113]	G700 [▶ 42]	Modal.	Dimensions in inches with calculation of the feed.
G710 [▶ 113]	G710 [▶ 42]	Modal.	Dimensions metric with calculation of the feed.

Command NCI Interpreter	Command GST Interpreter	Modal or Non-modal	Description
Mirror [▶ 132]	transMirrorX/Y/Z [▶ 65], transMirrorX/Y/Z [▶ 65], transMirrorX/Y/Z [▶ 65]	Modal.	Mirroring the coordinate system.
MOD [▶ 140]		Nonmodal.	Modulo movement.
MSG [▶ 161]	Msg [▶ 62]	Nonmodal.	Message from the NC program.

Command NCI Interpreter	Command GST Interpreter	Modal or Non-modal	Description
NORM [▶ 172]		Nonmodal.	Orthogonal approach off the contour and orthogonal departure from the contour.

Command NCI Interpreter	Command GST Interpreter	Modal or Non-modal	Description
P+ [▶ 111]	P [▶ 43]	Modal.	Feed direction positive.
P- [▶ 111]	P [▶ 43]	Modal.	Feed direction negative.
paramAutoAccurateStop [▶ 139]		Modal.	Automatic accurate stop.
paramAxisDynamics [▶ 156]		Modal.	Parameterization of the axis dynamics.
paramC1ReductionFactor [▶ 158]		Modal.	C1 reduction factor.
paramC2ReductionFactor [▶ 158]		Modal.	C2 reduction factor.
paramCircularSmoothing [▶ 138]		Modal.	Circular smoothing.
paramDevAngle [▶ 158]		Modal.	C0 reduction - deflection angle.
paramGroupVertex [▶ 138]		Modal.	Circular smoothing (old).
paramGroupDynamic [▶ 156]		Modal.	Pathway dynamics (old).
paramPathDynamics [▶ 156]		Modal.	Pathway dynamics.
Circular accuracy [▶ 120]		Modal.	Circular accuracy.
paramSplineSmoothing [▶ 136]	smoothingSet [▶ 79]	Modal.	Vertex smoothing. NC: Smoothing with Bezier Splines.
paramVertexSmoothing [▶ 133]	smoothingSet [▶ 79]	Modal.	Smoothing of segment transitions.
paramVeloJump [▶ 158]		Modal.	C0 reduction - maximum step change in velocity.
paramVeloMin [▶ 159]		Modal.	Minimum velocity.
paramZeroShift [▶ 124]	zeroOffsetShiftSet [▶ 75]	Modal.	Parameterization of the configurable zero shift.
PathAxesPos [▶ 160]	frameGet [▶ 73], qAxisGet [▶ 73]	Nonmodal.	Reads the actual position.

Command NCI Interpreter	Command GST Interpreter	Modal or Non-modal	Description
Additive Rotation [▶ 130]	transRotA [▶ 64], transRotX/Y/Z [▶ 63], transRotX/Y/Z [▶ 63], transRotX/Y/Z [▶ 63]	Modal.	Rotation additive.
ROT [▶ 129]		Modal.	Absolute rotation.
RotExOff [▶ 129]		Modal.	Extended rotation function off.
RotExOn [▶ 129]		Modal.	Extended rotation function on.
RotVec [▶ 129]		Nonmodal.	Calculation routine for rotating a vector.
RParam [▶ 115]	`!R1:=var1;` `!R2:=var2;` `!R3:=var3;` `…`	Nonmodal.	Initialization of an `R`-parameter.
RToDwordGetBit [▶ 115]		Modal.	Converts an `R`-parameter to `DWord` and checks whether a defined bit is set.

Command NCI Interpreter	Command GST Interpreter	Modal or Non-modal	Description
SEG [▶ 128]		Nonmodal.	Contour line programming (segment length).
skip VirtualMovements [▶ 161]		Modal.	Skip virtual movements.

Command NCI Interpreter	Command GST Interpreter	Modal or Non-modal	Description
ToolOffsetIncOff [▶ 164]		Modal.	Cartesian tool displacement and length compensation is not applied under `G91`.
ToolOffsetIncOn [▶ 164]		Modal.	Cartesian tool displacement and length compensation is applied under `G91`.
ToolParam [▶ 162]	toolParamSet [▶ 69]	Modal.	Set a tool parameter. `NC`: Writing and reading of tool parameters.
ToolParam [▶ 162]	toolParam [▶ 70]	Modal.	Yields the given tool parameter. `NC`: Writing and reading of tool parameters.
ToolParam [▶ 162]	toolSet [▶ 70]	Modal.	Set all parameters of a tool. `NC`: Writing and reading of tool parameters.
TPM [▶ 126]		Nonmodal.	Target position monitoring.

Command NCI Interpreter	Command GST Interpreter	Modal or Non-modal	Description
ZeroShiftIncOff [▶ 124]		Modal.	Zero shift is not applied under G91.
ZeroShiftIncOn [▶ 124]		Modal.	Zero shift is applied under G91.

Command NCI Interpreter	Command GST Interpreter	Description
	transScale [▶ 65]	Scales the coordinate system by the factor in the X-dimension, Y-dimension and Z-dimension.
	transDepth [▶ 67]	Yields the depth of the stack of transformations.
	transRestore [▶ 67]	Reduces the stack of transformations to the given depth.
	transPop [▶ 66]	Pops a transformation from the stack of transformations.

Command NCI Interpreter	Command GST Interpreter	Description
CIP [▶ 119]	moveCircle3d [▶ 68]	Move circular by rotating around the center cx,cy,cz and the normal vector nx,ny,nz by the given angle.

Command NCI Interpreter	Command GST Interpreter	Description
	queryAxes [▶ 72]	Set the MCS (machine coordinate system) coordinates of the interpreter to the actual coordinates of the physical axes.

Command NCI Interpreter	Command GST Interpreter	Description
	trcApproachDepartSet [▶ 73]	Configures the approach and depart behavior to use an arc of given radius and angle.
	trcOffsetSet [▶ 73]	Configures the amount of segment extension that is used to close gaps.
	trcLimitSet [▶ 74]	Configures the lookahead that is used for collision elimination.
	trcParam [▶ 74]	Returns the current configuration as a structure value.
	trcParamSet [▶ 74]	Configures the tool radius compensation.
	TrcParamType [▶ 74]	This structure contains all configuration parameters of the tool radius compensation.

Command NCI Interpreter	Command GST Interpreter	Description
	disableMask [▶ 74]	Yields the current value of the disable mask.
Block Skipping [▶ 108] /	disableMaskSet [▶ 75]	Sets the internal disable mask to the given value.

Command NCI Interpreter	Command GST Interpreter	Description
	runFile [▶ 79]	Streaming of large G-Code files.

Command NCI Interpreter	Versions	Command GST Interpreter	Description
@40 [▶ 115]	`@40 Kn Rn Rm …`		Save register on the stack.
@41 [▶ 115]	`@41 Rn Rm`		Save register on the stack.
@42 [▶ 115]	`@42 Kn … Rm Rn`		Restore register from stack.
@43 [▶ 115]	`@43 Rm Rn`		Restore register from stack.

Command NCI Interpreter	Versions	Command GST Interpreter	Description
@100 [▶ 151]	`@100 K±n @100 Rm`		Unconditional jump.

Command NCI Interpreter	Versions	Command GST Interpreter	Description
@111 [▶ 151]	`@111 Rn K/Rn Km …`	CASE OF [▶ 53]	Case block.

Command NCI Interpreter	Versions	Command GST Interpreter	Description
@121 [▶ 151]	`@121 Rn K/Rn Kn`		Jump if unequal.
@122 [▶ 151]	`@122 Rn K/Rn Kn`		Jump if equal.
@123 [▶ 151]	`@123 Rn K/Rn Kn`		Jump if less or equal.
@124 [▶ 151]	`@124 Rn K/Rn Kn`		Jump if less.
@125 [▶ 151]	`@125 Rn K/Rn Kn`		Jump if greater or equal.
@126 [▶ 151]	`@126 Rn K/Rn Kn`		Jump if greater.

Command NCI Interpreter	Versions	Command GST Interpreter	Description
@131 [▶ 153]	`@131 Rn K/Rn Kn`	WHILE [▶ 53]	Loop while equal.
@132 [▶ 153]	`@132 Rn K/Rn Kn`	WHILE [▶ 53]	Loop while unequal.
@133 [▶ 153]	`@133 Rn K/Rn Kn`	WHILE [▶ 53]	Loop while greater.
@134 [▶ 153]	`@134 Rn K/Rn Kn`	WHILE [▶ 53]	Loop while greater or equal.
@135 [▶ 153]	`@135 Rn K/Rn Kn`	WHILE [▶ 53]	Loop while less.
@136 [▶ 153]	`@136 Rn K/Rn Kn`	WHILE [▶ 53]	Loop while less or equal.

Command NCI Interpreter	Versions	Command GST Interpreter	Description
@141 [▶ 153]	`@141 Rn K/Rn Kn`	REPEAT [▶ 54]	Repeat until equal.
@142 [▶ 153]	`@142 Rn K/Rn Kn`	REPEAT [▶ 54]	Repeat until unequal.
@143 [▶ 153]	`@143 Rn K/Rn Kn`	REPEAT [▶ 54]	Repeat until greater.
@144 [▶ 153]	`@144 Rn K/Rn Kn`	REPEAT [▶ 54]	Repeat until greater or equal.
@145 [▶ 153]	`@145 Rn K/Rn Kn`	REPEAT [▶ 54]	Repeat until less.
@146 [▶ 153]	`@146 Rn K/Rn Kn`	REPEAT [▶ 54]	Repeat until less or equal.

Command NCI Interpreter	Versions	Command GST Interpreter	Description
@151 [▶ 153]	`@151 Rn K/Rn Kn`	FOR [▶ 53]	`FOR_TO` loop.

Command NCI Interpreter	Versions	Command GST Interpreter	Description
@161 [▶ 153]	`@161 Rn K/Rn Kn`	FOR [▶ 53]	`FOR_DOWNTO` loop.

Command NCI Interpreter	Versions	Command GST Interpreter	Description
@200	`@200 Rn`		Delete a variable.
@202	`@202 Rn Rm`		Swap two variables.

Command NCI Interpreter	Versions	Command GST Interpreter	Description
@302	`@302 K/R/Pn K/R/Pn R/Pn`		Read machine data bit.
@361 [▸ 160]	`@361 Rn Km`		Read machine-related actual axis value.
@372	`@372 Rn`		Set number of the NC channel in variable.

Command NCI Interpreter	Versions	Command GST Interpreter	Description
@402 [▸ 119]	`@402 K/R/Pn K/R/Pn K/R/Pn`	circleCenterReferenceSet [▸ 80]	Write machine data bit.

Command NCI Interpreter	Versions	Command GST Interpreter	Description
@610	`@610 Rn Rn`	ABS [▸ 55]	Find the absolute value of a variable.
@613	`@613 Rn Rn`	SQRT [▸ 55]	Find the square root of a variable.
@614	`@614 Rn Rm Rm`	`SQRT(a^2 + b^2)`	Find the square root of the sum of the squares of two variables ! `x := sqrt(a^2 + b^2);`.

Command NCI Interpreter	Versions	Command GST Interpreter	Description
@620 [▸ 153]	`@620 Rn`	`!var := var+1;`	Increment variable.
@621	`@621 Rn`	`!var := var-1;`	Decrement variable.
@622	`@622 Rn`		Find integer part of a variable.

Command NCI Interpreter	Versions	Command GST Interpreter	Description
@630 [▶ 115]	`@630 Rn Rm`	SIN [▶ 56]	Find the sine of a variable.
@630 [▶ 115]	`@630 Rn Rm`	gSin [▶ 77]	Find the sine of a variable.
@631 [▶ 115]	`@631 Rn Rm`	COS [▶ 56]	Find the cosine of a variable.
@631 [▶ 115]	`@631 Rn Rm`	gCos [▶ 77]	Find the cosine of a variable.
@632 [▶ 115]	`@632 Rn Rm`	TAN [▶ 56]	Find the tangent of a variable.
@632 [▶ 115]	`@632 Rn Rm`	gTan [▶ 77]	Find the tangent of a variable.
@633 [▶ 115]	`@633 Rn Rm`		Find the cotangent of a variable.
@634 [▶ 115]	`@634 Rn Rm`	ASIN [▶ 57]	Find the arc sine of a variable.
@634 [▶ 115]	`@634 Rn Rm`	gASin [▶ 78]	Find the arc sine of a variable.
@635 [▶ 115]	`@635 Rn Rm`	ACOS [▶ 57]	Find the arc cosine of a variable.
@635 [▶ 115]	`@635 Rn Rm`	gACos [▶ 78]	Find the arc cosine of a variable.
@636 [▶ 115]	`@636 Rn Rm`	gATan [▶ 78]	Find the arc tangent of a variable.
		gATan2 [▶ 78]	Returns the arc tangent of `y/x`.

Command NCI Interpreter	Versions	Command GST Interpreter	Description
@714 [▶ 150]	`@714`	sync [▶ 71]	Decoder stop.
@716 [▶ 150]	`@716`	A combination of sync [▶ 71] and queryAxes [▶ 72] replaces the former `@716`-command.	Decoder stop with rescan of the axis positions.
@717 [▶ 150]	`@717`	wait [▶ 71]	Decoder stop with external trigger event.

Command NCI Interpreter	Command GST Interpreter	Description
	LN [▶ 56]	Returns the natural logarithm of `x`.
	LOG [▶ 56]	Returns the decimal logarithm of `x`.
	EXP [▶ 56]	Returns `e` raised to the power of `x`.
	ADD [▶ 57]	Returns the sum of all parameters.
	MUL [▶ 57]	Returns the product of all parameters.
	SUB [▶ 58]	Returns the difference `x-y`.
	DIV [▶ 58]	Returns the quotient `x/y`.
	MOD [▶ 58]	Returns the remainder of the integer division `x/y`.
	EXPT [▶ 58]	Returns `x` raised to the power of `y`.

Command NCI Interpreter	Command GST Interpreter	Description
	GT [▶ 61]	Returns TRUE if x is larger than y.
	GE [▶ 61]	Returns TRUE if x is not smaller than y.
	EQ [▶ 61]	Returns TRUE if x and y are equal.
	LE [▶ 62]	Returns TRUE if x is not larger than y.
	LT [▶ 62]	Returns TRUE if x is smaller than y.
	NE [▶ 62]	Returns TRUE if x and y are not equal.

Command NCI Interpreter	Command GST Interpreter	Description
	Type Conversion (*_TO_*) [▶ 55], Type Conversion (*_TO_*) [▶ 55]	Explicit conversion between the given native types.

Logical Operations

Command NCI Interpreter	Command GST Interpreter	Description
	AND [▶ 60]	Returns the bitwise Logical And of all parameters.
	OR [▶ 60]	Returns the bitwise Logical Or of all parameters.
	XOR [▶ 60]	Returns the bitwise Logical Exclusive Or of all parameters.
	NOT [▶ 60]	Returns the bitwise complement of x.

Min, Max and Limit

Command NCI Interpreter	Command GST Interpreter	Description
	MAX [▶ 61]	Returns the maximum of all parameters.
	MIN [▶ 61]	Returns the minimum of all parameters.
	LIMIT [▶ 61]	Returns in if it lies in the interval [min,max]. Otherwise, the violated bound (min or max) is returned.

Rotation

Command NCI Interpreter	Command GST Interpreter	Description
	ROL [▶ 59]	Returns the bitstring x rotated left by y bits.
	ROR [▶ 59]	Returns the bitstring x rotated right by y bits.

Selection (Conditional Expressions)

Command NCI Interpreter	Command GST Interpreter	Description
	SEL [▶ 60]	Returns x1 if cond is FALSE, and x2 otherwise.
	MUX [▶ 60]	Returns x<select>.

Shift

Command NCI Interpreter	Command GST Interpreter	Description
	SHL [▸ 59]	Returns the bitstring `x` shifted left by `y` bits.
	SHR [▸ 59]	Returns the bitstring `x` shifted right by `y` bits.

Units

Command NCI Interpreter	Command GST Interpreter	Description
	unitAngleSet [▸ 76]	Set the unit for angles to `unitAngle`.
	UnitAngle [▸ 76]	Enumeration of unit angles.
	unitLengthSet [▸ 76]	Set the unit for lengths to `unitLength`.
	UnitLength [▸ 77]	Enumeration of unit lengths.
	unitTimeSet [▸ 77]	Set the unit for time to `unitTime`.
	UnitTime [▸ 77]	Enumeration of unit times.
	unitVelocitySet [▸ 77]	Set the unit for velocity to `unitLength/unitTime`.

4 Interpreter (DIN 66025/G-Code)

4.1 Basic Principles of NC Programming

4.1.1 Structure of an NC Program

An NC program is a text that is normally stored as a sequence of ASCII codes in a file on the hard disk. It consists of a sequence of NC blocks separated by line breaks (Return). Usually it is executed by being interpreted and worked through, character by character and line by line.

Program structure

The NC program is thus composed of three parts

- Program start (optional)
- Number of blocks
- Program end

Program start

At the beginning of an NC program the character "%" can represent the start of the program. The name of the program is then found following this character. The block for the program start does not necessarily have to be programmed.

Sample:

```
% Test1 (program start)
N10 G0 X100 Y100 Z0
M30 (program end)
```

NC block

Each NC block consists of one or several NC words, or even of none (an empty line), separated by spaces or tab characters. It is therefore not possible to use a space within a word.

Sample:

```
N10 G0 X100 Y100 Z0
```

NC word

The first character of an NC word specifies its meaning. It is either a letter or a special character.

Upper/lower case has, in general, no significance. Uniform use of upper case is, however, recommended for the sake of better readability. The optional following characters specify the meaning more precisely, or supply parameters for the execution.

In order to manage with such a limited supply of characters, an expression is not available for every variation of every function. It is rather the case that the significance and effect of many NC words is determined partly by the context. This can be a matter of the foregoing words in the block, but it can also depend on previous NC blocks. In a few cases the effect of NC words even depends on the machine data.

Program end

The end of the program is indicated by an M-function. Either M2 or M30 is used for this.

Effective Duration of Words

Commands such as G0 [▸ 118] or G17 [▸ 111], that have effects beyond the end of the block, are known, according to DIN 66025, as **modal**. These commands are effective as long as they are neither cancelled nor altered by another command.

Comments

If either parts of an NC block, or the whole of it, is not to be interpreted, the region concerned is to be placed within curved brackets.

Sample:

```
N10 G0 X100 (comment)
```

NOTE! A comment ends with the closing bracket, or, at the latest, at the end of the block. This means that a comment can't continue over a number of lines. Nested comments are also not possible.

Block numbers

Each block can be identified by a block number. The block number is accompanied by "N" for subordinate blocks and ":" for main blocks.

NOTE! The block number is not essential. A block not identified by a block number can not, however, be used as the target of a jump command. An error message, moreover, can only approximately report the location of the error (the last block number).

4.1.2 Block Skipping

It is often useful if not all blocks of a program are always executed. This makes it possible to implement similar processes with a single program.

In such cases, the blocks that belong to one variant are given a block skipping identifier. This must be written at the start of the block, and consists of a slash "/".

If several variants are required, the slash is extended with line information (0..15), for instance "/12". The line information (where "/" is equivalent to "/0") selects a bit from a word in the channel interface from the PLC to the NC. If this bit is **set**, the block is not interpreted.

In the NC the variable *'mSkipLine'* is evaluated for this purpose, which can be found among the inputs in the cyclic channel interface. The counterpart in the PLC can be found in the outputs under 'nSkipLine' [▸ 306] *(previously: nSkipBlock)* (see TwinCAT PLC library: NCI Interpreter).

If one of a number of variants is to be active, all the other suppressions must be set. Then only those blocks remain active that have either no identifier, or that have the desired identifier.

Note	**Active time of block skipping** The interpreter works an indeterminate number of blocks in advance of the execution. Block skipping can only operate correctly if it is set early enough (perhaps before the program starts), or if the interpreter is synchronized with the execution at a suitable location in the program (decoder stop [▸ 150]).

4.1.3 Look-Ahead

The actual velocity at the segment transition depends on a range of parameters. These include residual path length, dynamic parameters for the current segment, and (indirectly) the geometric angle at the segment transition.

Dynamic look-ahead (referred to as look-ahead below) ensures that the velocity can remain as high as possible at segment transitions. In the standard configuration 128 geometry entries are considered.

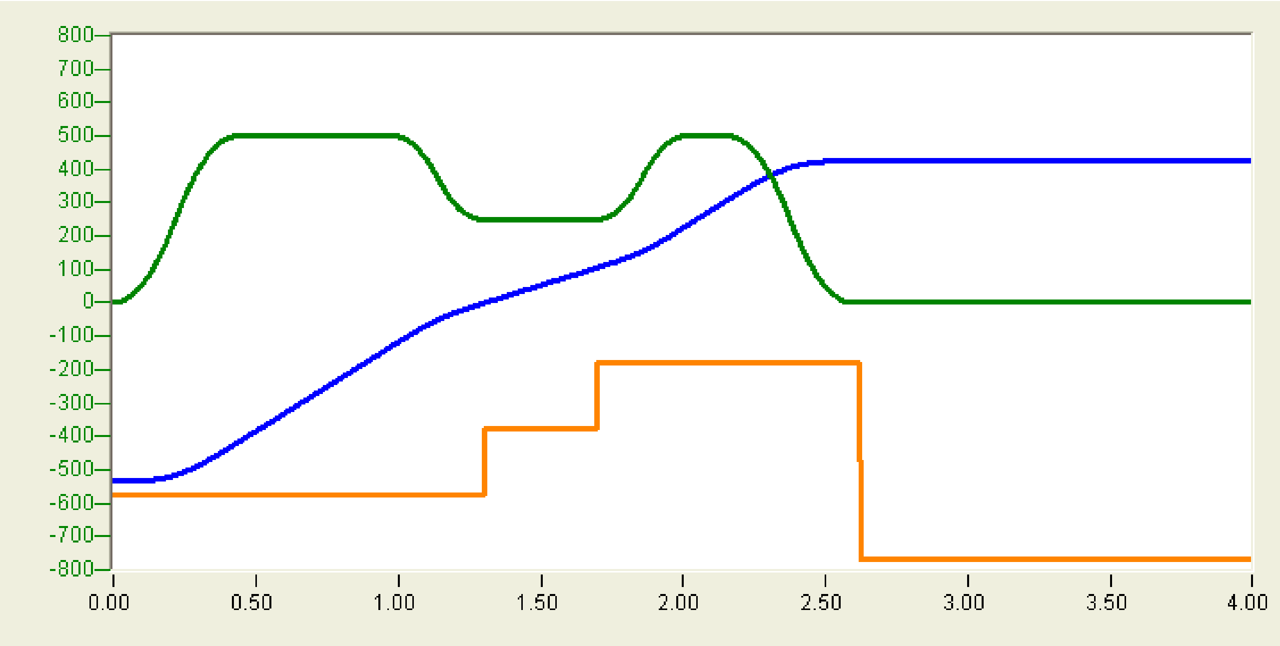
Without look-ahead the velocity is reduced to 0 at each segment transition (G60).

The number of geometry entries taken into account can be set in the DXD parameters [▸ 25].

Segments with different target velocity

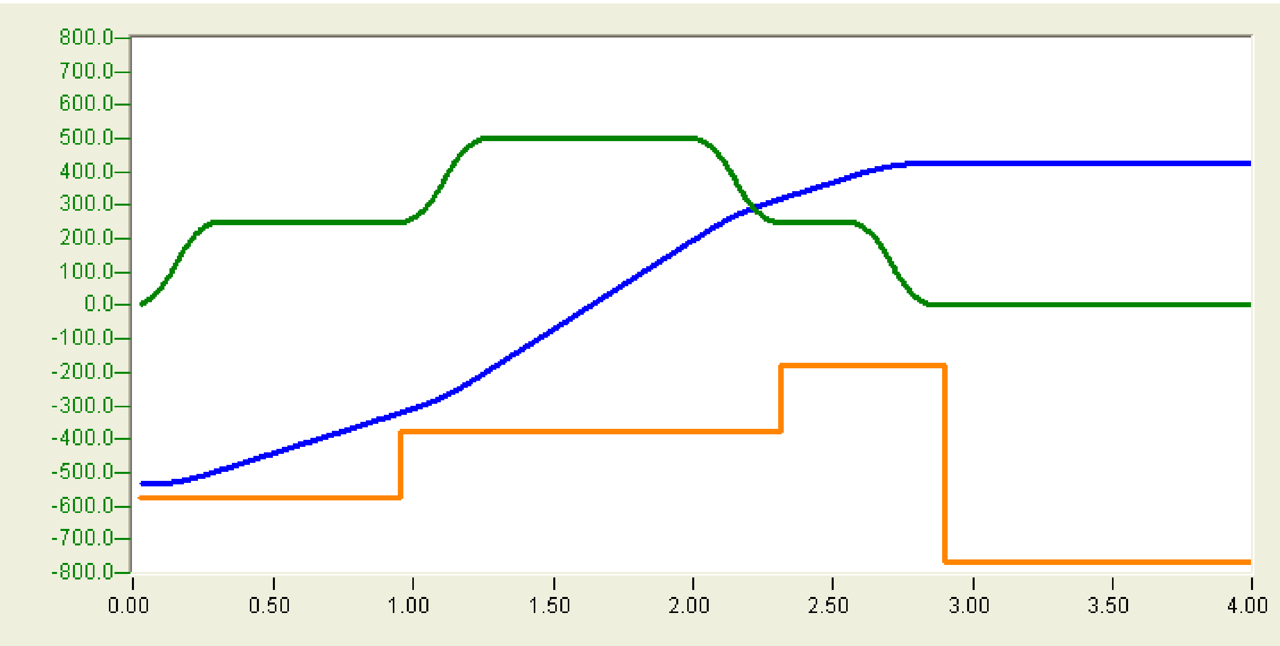
If the target velocity changes from a high velocity level to a lower level (N10 -> N20), the lower velocity will already have been reached at the start of the segment.

If the target velocity changes from a low velocity level to a higher level (N20 -> N30), the higher velocity is initiated with the segment transition. The system therefore always ensures that even at the segment boundary the current velocity does not exceed the programmed velocity.



green: Path velocity
blue: Position
orange: Block numbers

```
N10 G01 X600 F30000
N20 G01 X700 F15000
N30 G01 X900 F30000
M30
```



green: Path velocity
blue: Position
orange: Block numbers

```
N40 G01 X200 F15000
N50 G01 X800 F30000
N60 G01 X900 F15000
M30
```

4.1.4 Smoothing of Segment Transitions

Overview

Segment transitions with no continuous second differential cause instability in the dynamics unless the path velocity is reduced to *0* at those points. For dynamically stable segment transition at a finite speed it is possible to smooth segment transitions with Bezier splines which alter the local geometry and ensure that the complete path has a continuous second differential.

Tolerance spheres

A tolerance sphere is laid around every segment transition within which the path may deviate from its pre-set geometry for smoothing purposes. The radius of the tolerance sphere (parameterization [▸ 301]) is predetermined by the user and applied modally for all segment transitions that imply no exact positioning or stop in the segment transitions. The radii of the tolerance spheres are automatically reset adaptively, thus preventing tolerance spheres from overlapping in the case of small segments.

Dynamic parameters

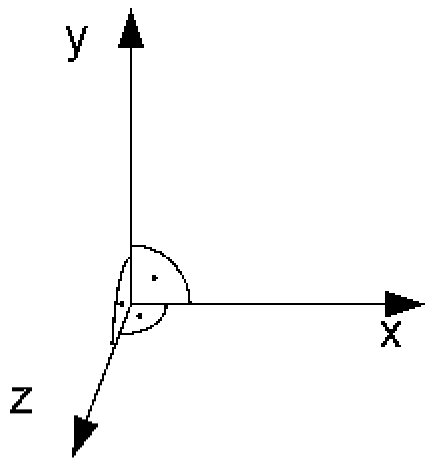
The smoothing enables faster dynamics. The system-determined maximum segment transition velocity *VeloLink* can be influenced by the user insofar as the system parameter C2 velocity reduction *C2* (parameterization [▸ 301]) sets the segment transition velocity to *C2x VeloLink*. The factor can be changed online.

General characteristics at segment transitions

When entering the tolerance sphere, the path acceleration is *0* and the path velocity equals the segment transition velocity. This is maintained within the tolerance sphere. The override is inactive within the tolerance sphere, i.e. the change of the velocity level caused by the override is interrupted within the tolerance sphere and continues after the exit from the tolerance sphere.

4.1.5 Co-ordinate System

The names of the axes of a machine tool are specified by DIN 66217. The letters X, Y and Z are allocated to the axes. These create a right-handed right-angle (Cartesian) coordinate system. In many machines, not all three axes are present at every location. In these cases individual letters are allocated in some meaningful way, and the axes that are not present are ignored.



4.1.6 Dimensional Notation

Dimensional data can optionally be referred to an absolute origin or to the current set value.

Absolute dimensions

Command	G90
Cancellation	G91

All positional data in absolute dimensions are always given with reference to the currently valid origin.

In terms of tool movement, this means that, under absolute dimensioning, it is the position to which the tool should move that is described.

Incremental Dimensions

Command	G91
Cancellation	G90

When dimensions are incremental, positional data is related to the immediately preceding point. In addition to the path axes, the auxiliary axes (Q1..Q5) are also taken into account.

For the tool movement this means that the incremental dimension describes by how much the tool is moved.

Units

The units for length, angle etc. are described in the following table:

	Unit
Positions and lengths	mm
Angle	degree
Times	sec
Feed	mm/min

4.1.7 Working Plane and Feed Direction

In order to describe circles (except CIP [▶ 119]), and for the compensation of cutter radius [▶ 167] and tool length [▶ 164], it is necessary to specify the working plane.

Working Plane XY

Command	G17
Cancellation	G18 or G19

The function G17 specifies the XY plane as the working plane and the feed direction as the Z direction.

The function acts as:

- Plane for tool radius compensation [▶ 167]
- Feed direction for tool length compensation [▶ 164] (offset)
- Plane for circle interpolation

Note	**Changing the working plane** The working plane cannot be changed while tool compensation is active.

Working Plane ZX

Command	G18
Cancellation	G17 or G19

The function G18 specifies the ZX plane as the working plane and the feed direction as the Y direction.

Working Plane YZ

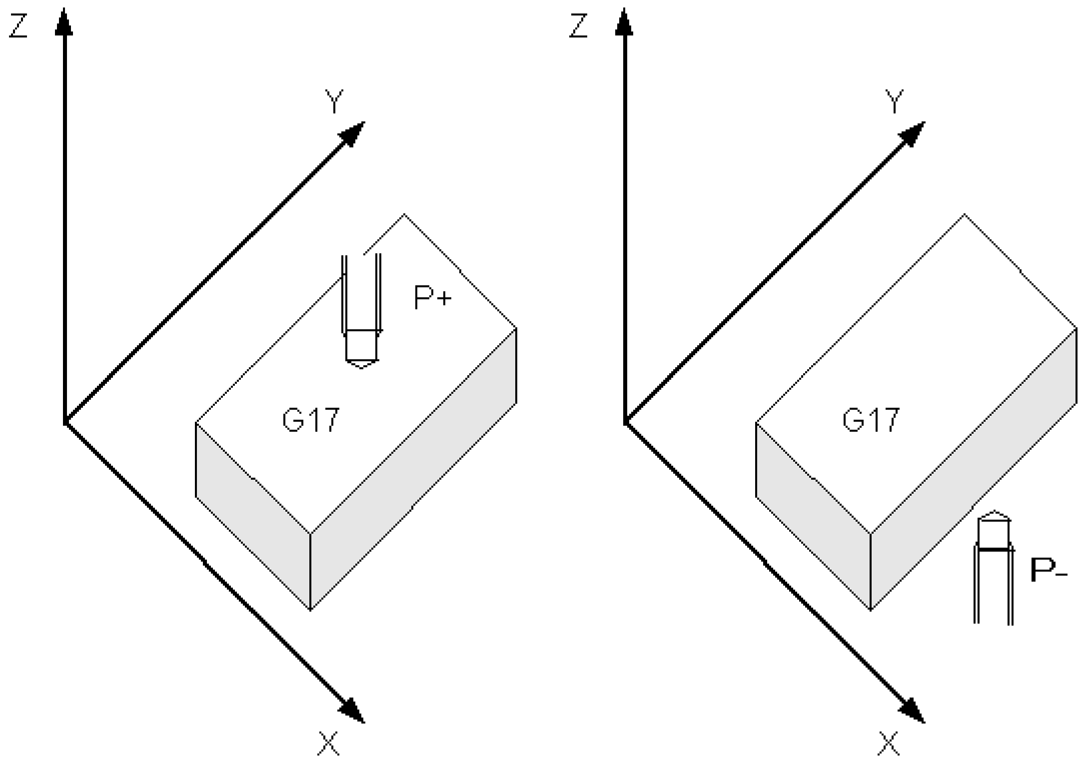
Command	G19
Cancellation	G17 or G18

The function G19 specifies the YZ plane as the working plane and the feed direction as the X direction.

Specification of the feed direction

Command	P
Parameter	+ feed direction positive (default) - feed direction negative

Parameterization of the feed direction is required for tool length compensation. It is used to specify whether the tool operates above or below the workpiece.



Sample:

```
N10 G0 X0 Y0 Z0 F6000
N20 D2 P- Z
N30 G01 X100
N40 D0 Z
N50 M30
```

In this sample the length compensation operates below the workpiece.

4.1.8 Inch/metric dimensions

G70	Dimensions in inches
G71	Dimensions in millimeters (default)
G700	Dimensions in inches with calculation of the feed
G710	Dimensions in millimeters with calculation of the feed

Dimensions in millimeters (G71) is active by default. Information on whether the coordinates have to be converted is stored in the machine parameters [▶ 11] (Interpreter tab). The basic dimension system in millimeters is also set there by default.

The effects of the changeover

If the basic dimension system is not the same as the current dimension system (set with G70 or G71), then certain parameters and co-ordinates must be converted. The conversion factor required here is stored in the machine parameters, like the basic dimension system. The changeover has effects on the following parameters:

- Path information for the path axes (X, Y & Z)
- Path information for the auxiliary axes (Q1..Q5)
- Intermediate point co-ordinates (I, J, K)
- Circle radii (B or U)
- Programmable zero shift
- Rounding radius (circle and spline smoothing)

There are also parameters that always remain in the **basic dimension system**, and are not converted. These include the

- adjustable zero shift
- Tool data
- feeds (except under G700 or G710)

Sample 1:

Basic dimension system: inch

```
...
N10 G71      (metric dimensions)
N20 G01 X100 (conversion is carried out)
N30 G70      (dimensions in inches)
N40 G01 Y100 (conversion is not necessary, because)
....         (the basic dimensions are also inches)
```

Sample 2:

Basic dimension system: millimeters

```
...
N10 G71 (metric dimensions)
N20 G01 X100 (conversion is not necessary, because)
    (the basic dimensions are also metric)
N30 G70 (dimensions in inches)
N40 G01 Y100 (conversion is carried out)
```

Zero shifts (NPV)

Adjustable zero shifts (G54-G57) always remain in the basic dimension system, and are not converted. In the case of the programmable zero shifts (G58 & G59) the effect depends on the current dimension system when the shift is selected.

Sample 3:

Basic dimension system: millimeters

```
...
N10 G71    (mm - default)
N20 G54    (activates adjustable zero offset shift)
```

```
N30 G58 X100 (programmable zero offset shift)
N40 G01 X0 F6000 (the axis travels to 100 in the machine co-ordinate system)
N50 G70    (inch)
N60 G01 X0   (zero offset shift is programmed under G71 => zero offset shift remains unchanged)
       (i.e. the axis does not move)
N70 G58 X100 (new programmable zero offset shift - now in inches)
N80 G01 X0   (axis moves out by zero offset shift - to 2540 in the machine co-ordinate system)
```

4.1.9 Single Block Operation

To test a new NC program, the NCI can be switched to single block mode with the function block ItpSingleBlock [▶ 226]. When single block mode is active, the NC program is stopped after each line. The user has to acknowledge execution of the next line. This can be done by pressing '**NC start (F5)**' in the XAE under the Editor tab or by setting the input 'bTriggerNext' in the PLC function block ItpSingleBlock [▶ 226].

A distinction is made between two modes:

- Interpreter single block mode
- NC kernel single block mode

Interpreter single block mode

If interpreter single block mode is active, the NC program is stopped after each line in the **interpreter**. This remains true even if the line only contains calculations, and no physical movement is programmed.
This enables re-writing of R-parameters, for example.

Interpreter single block mode should be activated before the NC program is started. If this is not possible, an M-function can be reserved for the activation and combined with a decoder stop.
If interpreter single block mode is activated during processing of the NC program without M-function and decoder stop, it is impossible to predict when it will be active. Theoretically it is possible that the memories in the NC kernel (SVB & SAF) are filled and contain more than 100 geometry entries. The single block can only take effect once these memories have been fully processed.

NC kernel single block mode

Like in interpreter single block mode, in NC kernel single block mode the NC blocks are executed individually. The difference is that in NC kernel single block mode all entries (e.g. geometry entries) have already passed through the interpreter. It is therefore not possible to overwrite R-parameters retrospectively, for example.

This operating mode has the advantage that single block mode can be enabled during processing of the NC program. If a geometry entry is executed (i.e. the axes are moved) during the activation, the system stops at the next possible end of segment. This is usually the current segment. For activation after program startup no M-function with decoder stop is required.

If NC kernel single block mode is used in conjunction with blending, block relaying takes place in the blending sphere. The programmed blending continues to be executed (from TwinCAT V2.10 Build 1301).

Alternatives to activation

We recommend activating single block mode with ItpSingleBlock [▶ 226].

For reasons of compatibility with previous TwinCAT versions, single block mode can be activated via the cyclic channel interface.

Single block mode can be selected or deselected in the cyclic channel interface of the PLC. To this end the variable 'nItpMode' has to be masked correctly in the PLC/NC channel interface.

Set bit 14 (0x4000) to switch on interpreter single block mode. Resetting the bit turns single block mode off again.

It is also possible to trigger the single block from the PLC by means of this interface. Bit 15 must be set for this. The effect is the same as activating NC start in the XAE.

4.1.10 Arithmetic Parameters

The arithmetic parameters (known as R-parameters for short) are interpreter variables that are named by an expression of the form "R<n>". Since 'n' is an integer in the range 0..999, a total of 1000 R-parameters are available. Of these, the first 900 values (R0..R899) are local variables for the NC channel. They can only be accessed by the channel's interpreter. The R-parameters R900..R999 are declared globally. They exist only once for each NC, and all channels access the same storage. This makes it possible to exchange data (e.g. for part tracing, collision avoidance etc.) over the channel boundaries.

Mathematical Calculations

The R-parameters (like the axis co-ordinates, feedrates etc.) are declared as variables of type 'double'. This makes full use of the computer's arithmetic capacity. The number of places before and after the decimal point is not restricted by a format specification. The arithmetical resolution does, nevertheless, have a limit. In practice this is only visible in particularly critical cases. Examples of this include the differences of very large numbers that are almost equal, or trigonometrical functions in particular ranges of angles.

Assignment of R-Parameters

```
N100 R5=17.5
N110 R6=-4
N120 R7=2.5 R8=1
```

As can be seen in the third line, it is quite possible to make more than one assignment in one block. This speeds interpretation slightly, but it can be more difficult to localize an error in the line.

Calculation formula

A calculation formula is an extension of assignment. It consists of a target parameter, an assignment sign and a series of values (R-parameters and constants) separated by arithmetical instructions.

```
N100 R1=R2+R3-17.5*R9/2.5
```

Such a formula, contrary to normal mathematical practice, is processed strictly from left to right.

The illustrated formula is calculated as follows:

1. The contents of R2 is loaded into the arithmetic unit
2. The contents of R3 is loaded into the arithmetic unit
3. The arithmetic unit carries out the + instruction
4. The value 17.5 is loaded into the arithmetic unit
5. The arithmetic unit carries out the - instruction
6. The contents of R9 is loaded into the arithmetic unit
7. The arithmetic unit carries out the * instruction
8. The value 2.5 is loaded into the arithmetic unit
9. The arithmetic unit carries out the / instruction
10. The content of the arithmetic unit is stored in R-parameter R1

Mathematical functions

The interpreter provides standard computing functions. DIN 66025 does not specify any syntax here. The computing functions are called via @6xx (see appendix - @-command overview [▸ 178]).

The trigonometrical functions are always calculated in degrees.

Sample:

```
N10 R2=0 R3=45
N20 @630 R2 R3
```

In this sample the sine of R3 is calculated in degrees. The result is then written into R2.

R-parameter access from the PLC

You can read the R-parameters into the PLC, or write the R-parameters from the PLC. Special PLC function blocks are provided for this purpose

- ItpReadRParams [▸ 258]
- ItpWriteRParams [▸ 270]

During writing of the R-parameters, ensure that the interpreter is ahead of the block execution. In other words, writing of the R-parameters from the PLC should take place before the NC program starts or be linked to a decoder stop [▸ 150].

For debugging purposes, all R-parameters can be written to a file at any time. This process can be triggered via ADS (see ADS interface - channel functions IndexOffset 0x24 & 0x25 [▸ 382]).

Other functions

RToDwordGetBit

This function converts an R-parameter to a DWord and then checks whether a particular bit is set. The result is again stored in an R-parameter.

Command	RToDwordGetBit[<dest>; <src>; <bit>]
Parameter <dest>	R-parameter in which the result is entered
Parameter <src>	R-parameter containing the number that is to be converted and checked
Parameter <bit>	Bit to be checked (0..31)

Sample:

```
N10 R1=7
N20 RToDwordGetBit[R2;R1;0]
R10=31
N30 RToDwordGetBit[R3;R1;R10]
```

Enter 1 in R2 and 0 in R3.

Initialization of R-parameters

'set RParam' is used to assign a value to a contiguous block of R-parameters.

Command	#set RParam(<start index>; <count>; <value>)#
Parameter <start index>	Describes the first R-parameter to be written
Parameter <count>	Number of R-parameters to be written
Parameter <value>	Assigned value

Sample:

```
N10 G01 X100 Y200 F6000
N15 R2=3000
N20 #set RParam( 1; 2; 0.0 )# (R2 is overwritten again here)
N30 G01 X500
```

Saving R-Parameters

If the content of R-parameters [▸ 115] is required for subsequent use, while in the meantime the R-parameters are used for a different purpose, it can temporarily be stored in the values stack of the arithmetic unit.

Two possibilities exist for this:

- enumeration of the R-parameters
- giving the range of R-parameters

Saving the values:

Command	@40 <number> R<n> R<m>... @41 <1st R-parameter> <last R-parameter>

Restoring the values:

Command	@42 <number> R<n> R<m> @43 <last R-parameter> <1st R-parameter>

When restoring the values, call the parameters in reverse order.

Sample 1:

```
(saving the data)
N100 @40 K4 R800 R810 R823 R4

N110 R800=4711
N120 ...

(restoring the data)
N200 @42 K4 R4 R823 R810 R800
```

Sample 2:

```
(saving the data)
N100 @41 R800 R805

N110 R800=4711
N120 ...

(restoring the data)
N200 @43 R805 R800
```

Note	**Stack size** The value stack of the arithmetic unit has limited capacity. If it overflows, the NC program is interrupted with an error message. That can occur as the value is saved, but can also occur in the course of subsequent formula evaluation.

4.2 Programming Movement Statements

4.2.1 Referencing

By default, axis referencing (homing) should take place before the 3D-group is formed from the PTP channel. Or it can be done from the NC program.

If axes are referenced in PTP mode, it can be done for several axes simultaneously. If axes are referenced from the NC program, it can only be done for one axis at a time.

Command	G74
Cancellation	End of block

Sample:

```
N10 G74 X
N20 G74 Y
```

Note	**Referencing with own block** Referencing must be carried out within its own block. G74 may only refer to one axis. This command is only applicable for the main axes (X,Y,Z).

4.2.2 Rapid Traverse

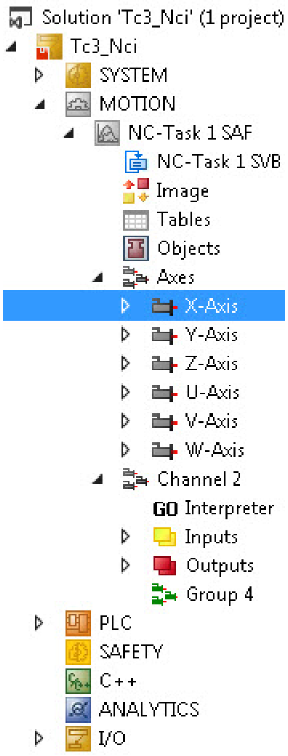
Command	G0
Cancellation	G1 [▶ 119], G2 [▶ 119] or G3 [▶ 119]

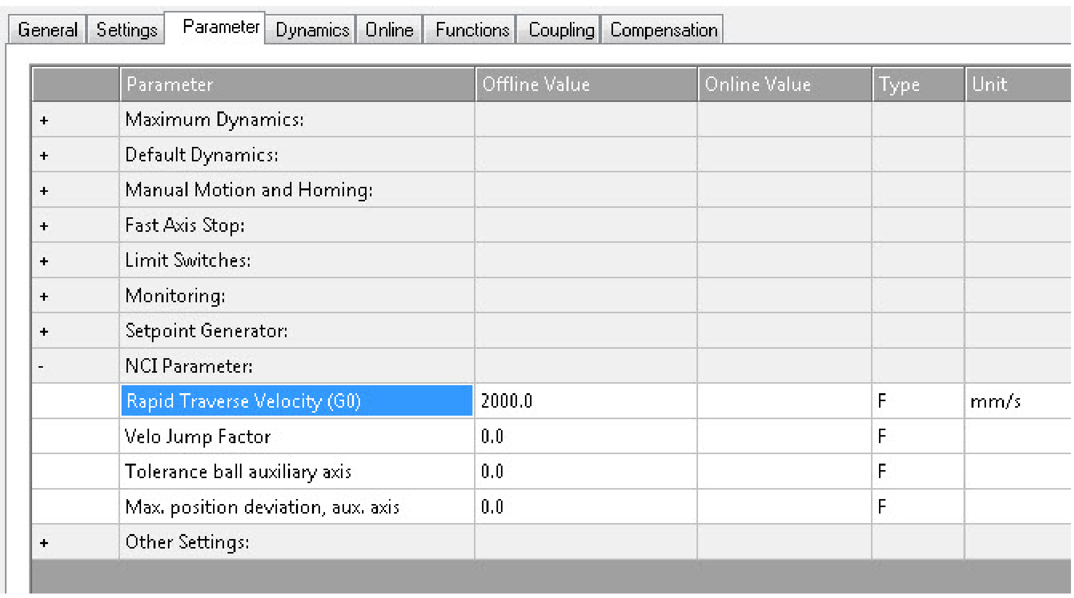
Rapid traverse is used to position the tool quickly, and is not to be used for machining the workpiece. The axes are moved with maximum velocity.

If a number of axes are to be driven in rapid traverse, the velocity is determined by that axis that requires the most time for its movement.

An accurate stop (G60 [▶ 123]) is cancelled with G0.

The rapid traverse velocity is set individually for each axis. This can be edited in the axis parameters in the XAE under NCI parameters.





	Parameter	Offline Value	Online Value	Type	Unit
+	Maximum Dynamics:				
+	Default Dynamics:				
+	Manual Motion and Homing:				
+	Fast Axis Stop:				
+	Limit Switches:				
+	Monitoring:				
+	Setpoint Generator:				
-	NCI Parameter:				
	Rapid Traverse Velocity (G0)	2000.0		F	mm/s
	Velo Jump Factor	0.0		F	
	Tolerance ball auxiliary axis	0.0		F	
	Max. position deviation, aux. axis	0.0		F	
+	Other Settings:				

4.2.3 Linear Interpolation

Command	G1 or G01 (default)
Cancellation	G0 [▸ 118], G2 [▸ 119] or G3 [▸ 119]

Under linear interpolation the tool moves, with feedrate F, along a straight line that can be freely located in space. The movement of the axes involved is completed at the same moment.

The feedrate (short: feed), F, describes the rate of displacement in millimeters per minute. This value is effective globally, so that it is not necessary to program it again if the same feed is to be used later for other geometrical movements.

Sample:

```
N10 G90
N20 G01 X100.1 Y200 F6000
```

In this example the axes are moved linearly to the position described. The Z axis is not mentioned in this program, and therefore retains its old position.

4.2.4 Circular Interpolation

Circular arcs can be programmed in a number of ways. Two types must be distinguished. One of these is an arc in the working plane [▸ 111] (e.g. the XY plane), and the other is an arc that can be freely located in space (a CIP circle).

Clockwise circular interpolation

Command	G2 or G02
Cancellation	G0 [▸ 118], G1 [▸ 119] or G3 [▸ 120]

Function G2 describes the path of a circular arc clockwise. This function requires the working plane [▸ 111] to have already been defined (G17 [▸ 111] by default).

In order to describe the circle unambiguously, further parameters are required in addition to the end point. A choice is available between center point programming and radius programming.

Radius programming

In radius programming the radius of the circle is programmed as well as the end point. Either of the letters *'B'* or *'U'* may be used for the radius.

Since the direction is prescribed with G2, the circle is also unambiguously described. The starting point is determined by the foregoing geometrical movements.

Sample 1:

```
N10 G01 G17 X100 Y100 F6000
N20 G02 X200 B200
```

Note	**Angle programming for angles >180°** If an angle of more than 180° is to be traversed, the radius must be stated negatively.

Note	**Full circle programming** The start and the end points must be different, so that the center can be calculated. Radius programming can therefore not be used for programming a full circle. Centre point programming can be used for this purpose.

Centre point programming

Centre point programming represents an alternative to the method that has just been described. The advantage of center point programming is that full circles can also be described in this way.

Under the standard settings, the center point is always given relatively to the starting point of the arc. The parameters I, J and K are used for this purpose. With

- I for the X-component
- J for the Y-component and
- K for the Z-component.

At least one of these parameters is 0 and therefore does not have to be included in the program.

Sample 2:

```
N10 G01 G17 X100 Y100 F6000
N20 G02 I50 J0 (J is optional) X200
N30 M30 (program end)
```

Sample 3:

```
N10 G01 G18 X100 Y100 Z100 F6000
N20 G02 I0 K50 X150 Z150 (quarter circle in ZX plane)
N30 M30
```

By programming an item of machine data it is however also possible to enter the center point absolutely. The command @402 is required for write access to a machine data bit.

In the following example, the circle from the first example is programmed using the absolute circle center.

Sample 4:

```
N10 G01 G17 X100 Y100 F6000
N20 @402 K5003 K5 K1 (centre point programming on)
N30 G02 I150 J100 X200
N40 @402 K5003 K5 K0 (centre point programming off)
N50 M30
```

Anticlockwise Circular Interpolation

Command	G3 or G03
Cancellation	G0 [▸ 118], G1 [▸ 119] or G2 [▸ 119]

The function G3 describes a circular arc anticlockwise. The parameters and entry possibilities are the same as under G2.

Circular accuracy

Command	#set paramRadiusPrec(<param>)#
Parameter	param: Maximum permitted radius tolerance 0.001 < param < 1.0 (default 0.1)

The 'set paramRadiusPrec' function is used to parameterize the required circular accuracy. This parameter affects circles programmed with G02 or G03.

With center point programming, an error is generated if the difference in radius length is greater than <param>.

Centre point correction

Command	CPCON (standard setting)
Cancellation	CPCOF

In center point programming the circle is overdetermined. For data consistency, the center point is usually corrected. Normally only a marginal modification of the center point is required. After the center point correction, the magnitude of the input radius equals the output radius.

It the start and end point are very close together, the center point offset may be large. This may lead to problems with automatically generated G-Code (postprocessor). For manually written G-Code, the CPCON setting (center point correction on) is recommended.

CIP circle

Command	CIP
Cancellation	End of block

The circles discussed so far can only be used in the principal planes. The CIP circle can also be used to program a circle anywhere in space. For this purpose it is necessary to program not only an end point but also some other point on the path.

So that the circle can be described unambiguously, it is necessary that the three points (the starting point is given implicitly) must not be collinear. It is thus not possible to program a full circle in this way.

I, J and K are available as path point parameters. By default their values are relative to the starting point of a circular path.

Sample 5:

```
N10 G01 X100 Y100 F6000
N20 CIP X200 Y200 I50 J50 K50
```

NOTE! For the CIP circle motion, cutter radius compensation [▶ 167] **must not be active.**

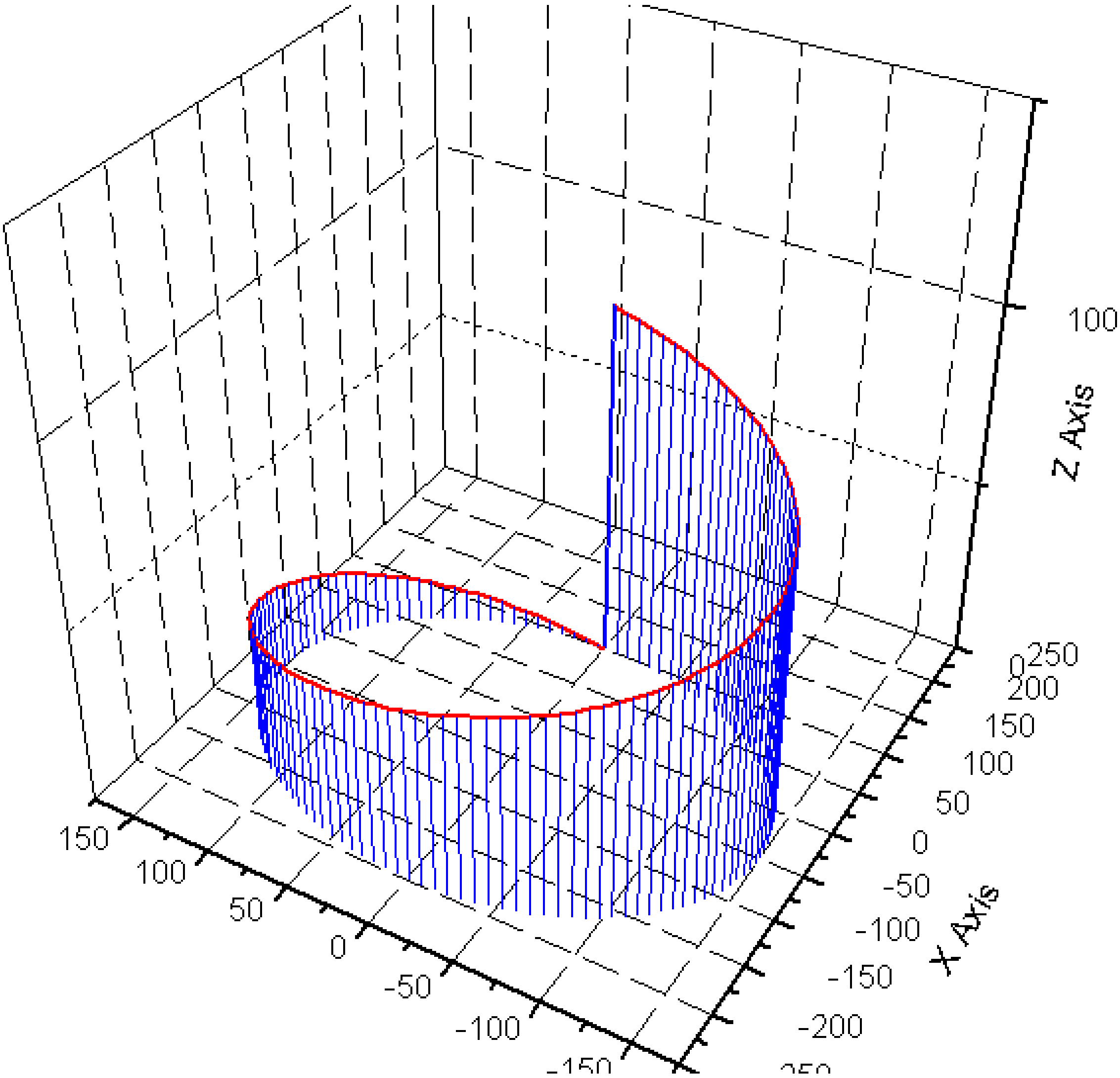
4.2.5 Helix

If a circular motion is superimposed onto a perpendicular linear movement, a helix is obtained. A helix can only be programmed in the principal planes. The same parameters as are used for a circle in the principal plane are used. At the same time the axis that is perpendicular to the plane is driven.

The helix can be used together with the cutter radius compensation [▶ 167].

Sample:

```
N10 G01 G17 X100 Y0 Z0 F6000
N20 G03 I-50 Z100
M30
```



4.2.6 Dwell Time

Command	G4 or G04
Cancellation	End of block
Parameter	F or X

G4 is used to switch on a dwell time. It is used to interrupt workpiece machining between two NC blocks for a programmed time (in seconds).

Sample:

```
N10 G01 X100 F6000
N20 G04 X0.5 (pause in sec)
N30 G02 X300
...
```

NOTE! The dwell time must be programmed in a dedicated set, and the parameters (X or F) must be programmed after G04.

4.2.7 Accurate Stop

block-by-block

Command	G9 or G09
Cancellation	End of block

The accurate stop instruction is used, for example, when sharp contour corners must be manufactured. At the contour transition the set path velocity is reduced to zero and then increased again. This ensures that the programmed position is approached precisely.

NOTE! G09 acts only on the set value side. The actual values can be checked with TPM (target position monitoring), for example.

modal

Command	G60
Cancellation	G0 [▶ 118]

Description:

see above

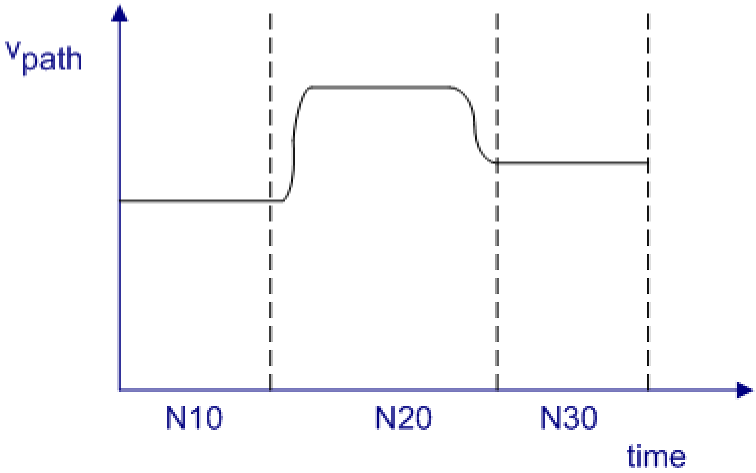
see also target position monitoring [▶ 126] (TPM)

4.2.8 Feed interpolation

Constant feed interpolation

Command	FCONST (standard setting)
Cancellation	FLIN

The programmed velocity is applied as fast as possible with the constant feed interpolation (default).



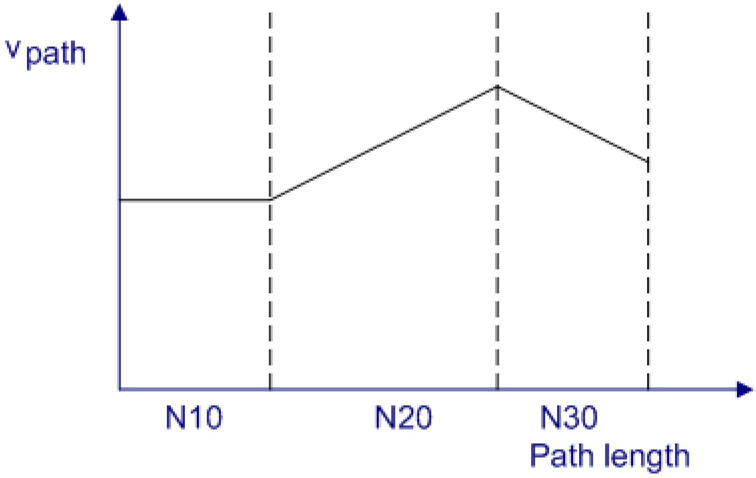
Sample 1:

```
N05 FCONST
N10 G01 X1000 F50000
N20 G01 X2500 F80000
N30 G01 X3500 F60000
...
```

Linear feed interpolation

Command	FLIN
Cancellation	FCONST

The linear feed interpolation transfers the velocity linearly over the path from v_start to v_end.



Sample 1:

```
N05 FCONST
N10 G01 X1000 F50000
N15 FLIN
N20 G01 X2500 F80000
N30 G01 X3500 F60000
...
```

NOTE! If the velocity on the segment transition has to be reduced more drastically than the programmed segment velocity, due to the geometry or M function for example, then the linear velocity is maintained as long as possible. The reduced segment velocity will delayed, only if required dynamically.

4.2.9 Zero Offset Shifts

A range of zero offset shifts are available in TwinCAT NC I. They describe the distance between the origins of the workpiece and of the machine.

Zero shift suppression

Command	G53
Cancellation	G54 [▸ 124] to G59 [▸ 125]

The zero shift is suppressed modally with G53. The suppression affects both the adjustable and the programmable zero shift.

Adjustable zero shift

Command	G54 G55 G56 G57
Cancellation	G53 [▸ 124] or selection of another configurable zero shift

The commands G54 to G57 can be used within the NC program to switch back and forth between the zero shifts.

Parameterization

The configurable zero shift can be parameterized in different ways

1. PLC function block ItpWriteZeroShiftEx [▶ 231] (recommended standard)
2. XAE Interpreter element [▶ 11]
3. from the DIN-program

The parameters are saved for each interpolation channel. This means that the adjustable zero shifts are channel dependent.

NOTE! The selection of a zero shift must be made in its own block. In order for the movement corresponding to the shift to be actually made it is necessary that at least the axes involved are named in a following geometrical block.

Sample 1:

```
N10 G01 X100 Y0 Z0 F6000
N20 G54 (activates adjustable zero offset shift (NPV))
N30 G01 X Y Z
N40 M30
```

In sample 1 all involved axes are named in line 30. The effect of this is that the zero shifts are applied to all the axes.

Sample 2:

```
N10 G01 X100 Y0 Z0 F6000
N20 G54 (activates adjustable zero offset shift (NPV))
N30 G01 X200 Y
```

In line 30 of sample 2 the X axis is taken to position 200 + shift in the X direction. The Y axis only moves to accommodate the shift, and the Z axis is not moved.

Parameterization from the DIN program

Command	#set paramZeroShift(G<n>; <value x>; <value y>; <value z>)#
Parameter G<n>	Zero shift to be parameterized (G54..G59)
Parameter <value>	Coordinates of the zero shift

'*#set paramZeroShift(..)#*' parameterizes the zero shift but does not activate it. This requires explicit programming of the G-Code.

Sample 3:

```
N10 G01 X100 Y0 Z0 F6000
N20 R12=200
N30 #set paramZeroShift( G54; 100.0; R12; -20)#
N40 G54 (activates adjustable zero offset shift (NPV))
N50 G01 X200 Y Z
```

Programmable zero shift

Command	G58 or G59
Cancellation	G53 [▶ 124]

Programmable zero shifts exist in addition to the adjustable ones. This type of zero shift is directly described from the NC program.

Note	**Addition of zero shifts** The programmable zero shift is only effective when the adjustable zero shift is active. This means that the total shift is the sum of • set zero shift (G54, G55, G56 or G57) • first programmable zero shift (G58) • second programmable zero shift (G59)

Sample 4:

```
N10 G01 X100 Y0 Z0 F6000
N20 G54 (activates adjustable zero offset shift (NPV))
N30 G58 X0.5 Y0.5 Z0.5 (1st prg. zero offset shift)
N50 X Y Z (movements for the zero offset shift)
...
M30
```

Behavior with incremental dimension notation

Default behavior

Changing the origin also affects the incremental dimension.

Sample 5:

```
N10 G01 X100 Y0 Z0 F6000
N20 G54 (activates adjustable zero offset shift (NPV))
N25 G58 X10 Y10 Z0
N30 G91 (Incr. dimensions)
N40 G01 X200 Y0
N50 ...
```

In N40 Y moves to 10 in the basic coordinate system. A shift in origin also shifts the point of reference for incremental dimension programming, resulting in a travel path for Y.

In this way a contour, which is fully programmed based on the incremental dimension, can be positioned at any point through a zero shift.

The behavior of G91 is parameterizable.

Command	Description
ZeroShiftIncOn	The zero shifts are also applied under G91 once the axis is named
ZeroShiftIncOff	The zero shift is not applied under G91

Sample 6:

```
N10 G01 X100 Y0 Z0 F6000
N15 ZeroShiftIncOff
N20 G54 (activates adjustable zero offset shift (NPV))
N25 G58 X10 Y10 Z0
N30 G91 (Incr. dimensions)
N40 G01 X200 Y
N50 ...
```

Since 'ZeroShiftInc**Off**' is set in sample 6 , the X-axis in N40 is moved by 200 mm independently of the new zero shift. The Y-axis does not move as no target coordinate is programmed for it.

See also ToolOffsetIncOn/Off [▸ 164]

4.2.10 Target Position Monitoring

Command	TPM
Cancellation	End of block

The command 'TPM' is used to trigger target position monitoring from the NC program. At the end of the geometry this always leads to an accurate stop on the set value side and subsequent checking of the target position window. Block relaying takes place when the monitoring conditions are met for all axes in the group.

Like for PTP, this function is enabled and parameterized individually for each axis. This means that different limits can be selected for auxiliary axes than for the path axes, for example.

Sample 1:

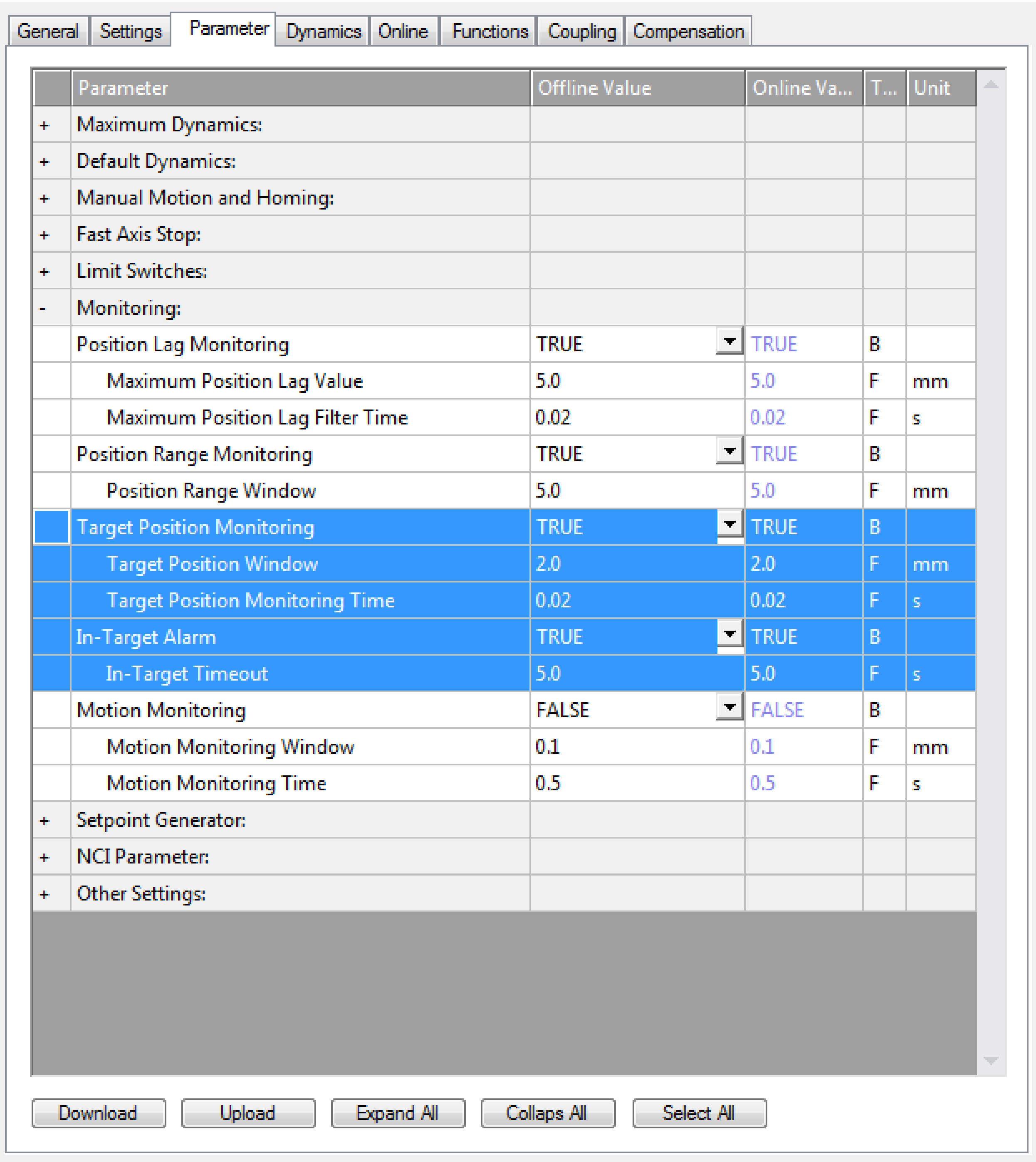
```
N10 G01 X100 Y100 F6000
N20 G01 X300 Y100 TPM
...
```

At the end of the motion of N20, target position monitoring is performed both for the X axis and for Y axis (provided target position monitoring is enabled for both axes).

Sample 2:

```
N10 G01 X100 Y100 F6000
N20 G01 X300 Y100
N30 M61 (Type Handshake)
N40 TPM
...
```

TPM can also be programmed in a dedicated block. In this case the last positioning is checked (of N20 in this case).



NOTE! If target position monitoring is enabled for an axis, the target position alarm (PEH) should also be active. Time monitoring results in a channel error after the timeout (or before), if the axis is not yet in the target position window. In order to avoid unnecessary channel errors, a sufficiently large timeout value should be selected (e.g. 5 - 10 s). If no PEH time monitoring is active and the axis

is permanently outside the position window, no block relaying takes place and the NC remains stationary when viewed from outside. The SAF is in Waiting state (not to be confused with Interpreter state).

See also accurate stop [▸ 123] (G09).

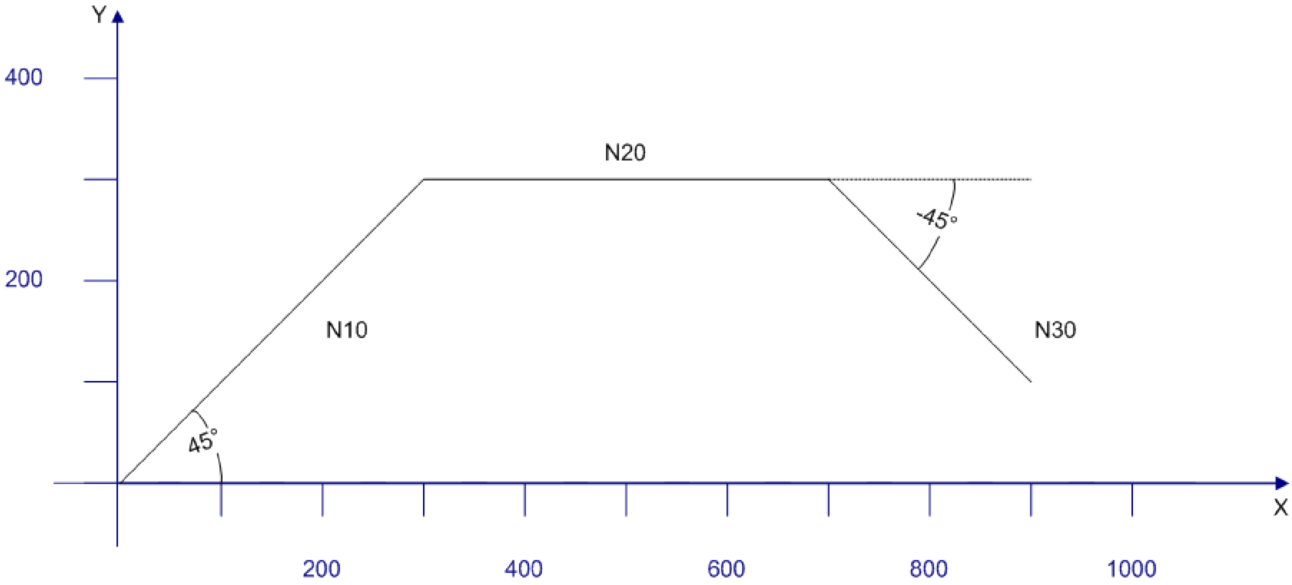
4.2.11 Contour definitions

Angle and segment length

In this type of programming the angle and the magnitude (segment length) are always quoted, similarly to polar co-ordinates.

Parameter	Description
ANG	Angle in degrees with reference to the abscissa (-360 ≤ ang ≤ 360)
SEG	Magnitude of the segment length

Sample 1:




```
N10 G01 ANG=45  SEG=424.264 F60000
N20 G01 ANG=0   SEG=400
N30 G01 ANG=-45 SEG=282.843
```

or

```
N10 G01 ANG=45  SEG=424.264 F60000
N20 G01 X700 Y300
N30 G01 ANG=-45 SEG=282.843
```

Restrictions:

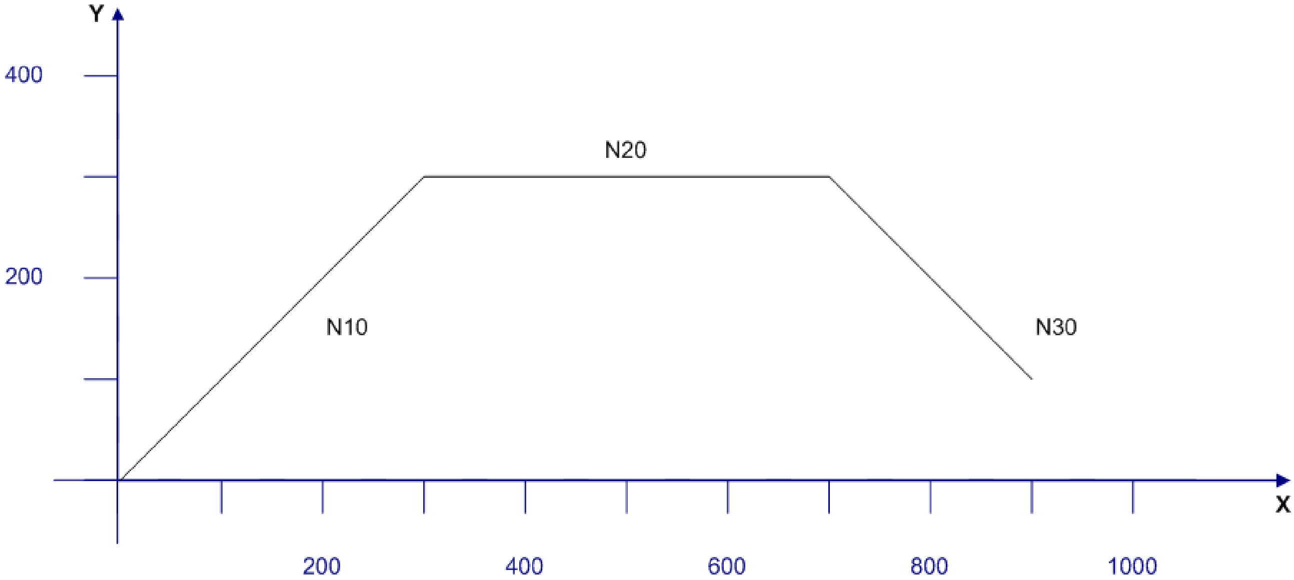
- The programming may only be done in the chosen principal plane.
- The length of the segment must be greater than zero, and refers to the projection in the principal plane.

NOTE! It is additionally possible to program rounding or chamfering. The ANG and SEG parameters must be programmed in every block. The assignment may use R-parameters, but formulas cannot be programmed.

Angle and one component in the plane

As above, an angle is programmed, but the length of the segment is no longer specified directly. It is calculated from a component in the selected principal plane.

Sample 2:




```
N10 G01 ANG=45 X300

N20 G01 ANG=0 Z700
R10=100
N30 G01 ANG=315
X=R10
```

Note	**Runtime error** If either two components in the plane are quoted or none at all, the result is a runtime error. A runtime error is also generated if the movement is parallel to the abscissa or to the ordinate, and there is therefore no intersection.

4.2.12 Rotation

It is also possible to program a rotation as well as the zero shift [▶ 124]. A distinction is drawn between absolute and additive rotation.

The rotation can turn the co-ordinate axes (X, Y and Z) in the workpiece coordinate system.

This makes it possible to machine inclined surfaces (in the plane or in space).

Absolute Rotation

Command	ROT X<value(x)> Y<value(y)> Z<value(z)>
Cancellation	ROT (without parameters)

The rotation instructions must be programmed in their own block. Angles must always be stated in degrees.

Direction of Rotation

A positive angle describes rotation in the direction of the positive co-ordinate axis, the rotation being anti-clockwise.

Carrying Out the Rotation

The sequence of rotations is of critical importance when a coordinate system is being rotated. In TwinCAT NC I rotations are always carried out in the following sequence around the global coordinate system:

1. Rotation around the Z-axis,
2. Rotation around the Y-axis,
3. Rotation around the X-axis.

This sequence is maintained even if the parameters are programmed in a different order.

The origin of the tool coordinate system is always used as the center point of the rotation. This means that the total zero offset shift currently active describes the rotation center.

Additive Rotation

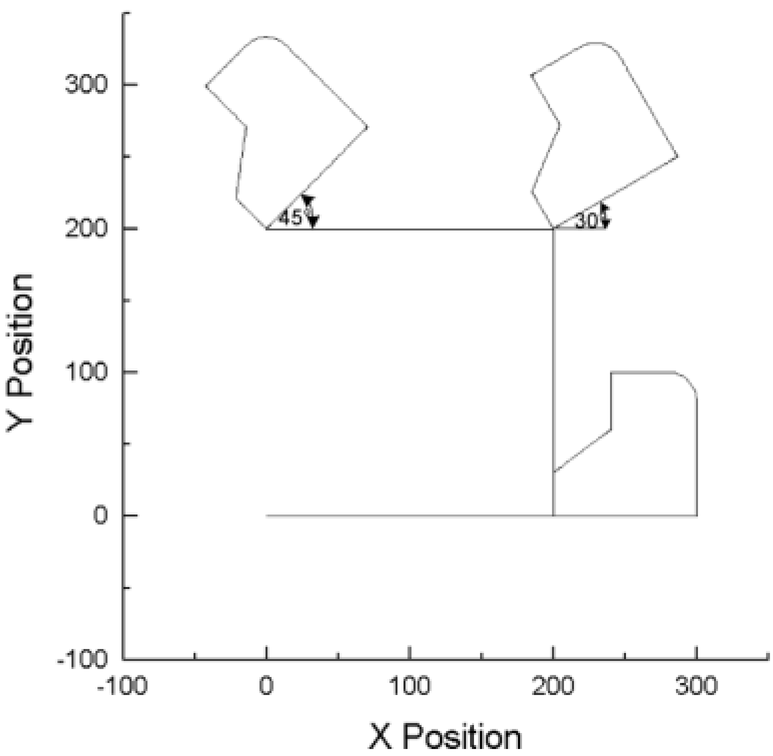
In addition to absolute programming of rotation it is also possible to carry this out additively. The same conditions apply to this as do to absolute rotation.

Command	AROT X <Wert(x)> Y<Wert(y)> Z<Wert(z)>
Cancellation	ROT (without parameters)

Sample:

```
N10 G01 G17 X0 Y0 Z0 F60000
N20 G55
N30 G58 X200 Y0
N50 L47
N60 G58 X200 Y200
N65 ROT Z30
N70 L47
N80 G58 X0 Y200
N90 AROT Z15
N100 L47
N110 M30
```

```
L47
N47000 G01 X0 Y0 Z0 (movements for zero shift & rotation)
N47010 G91 (incremental dimensions)
N47020 G01 X100
N47030 G01 Y80
N47040 G03 X-20 Y20 I-20 J0
N47050 G01 X-40
N47060 G01 Y-40
N47070 G01 X-40 Y-30
N47080 G01 Y-30
N47090 G90
N47100 M17
```



In this example, the same contour is traversed under different rotations. Since the contour (L47) is programmed in incremental dimensions, and the starting point is described by means of the programmed zero shift, the rotation is clear to see.

Note:

Once the ROT or AROT command has been programmed, the complete path vector (X, Y & Z) must be assigned.

Rotation extensions

In the default configuration the whole path vector must be programmed after each ROT command. Since this is difficult to realize in some applications, this calculation can optionally be performed automatically in the interpreter. To use this option, 'RotExOn' should be included at the start of the NC program.

Command	RotExOn
Cancellation	RotExOff

Sample:

```
N10 RotExOn
...
N100 G54 (activate zero point & point of rotation)
N110 ROT X90
N120 G0 Z3 (preposition the tool)
N130 G01 Z-10 F6000 (lower to cutting depth)
N140 G01 X100
N150 G01 Z3 (raise to preposition)
...
N1000 RotExOff
N1010 M30
```

Calculate rotation

Command	CalcRot[R<s>; R<t>; R<u>]
	CalcInvRot[R<s>; R<t>; R<u>]
Parameter	The 3 R-parameters describe the vector to be calculated. The calculation will write the result into this R-parameter, and the original value will therefore be overwritten.

The function **CalcRot** rotates a three-dimensional vector through the current rotation angle. The rotation angles had been determined by ROT or AROT. The sequence of the calculation is the same as is used for the rotation itself, that is Z, Y and X.

The **CalcInvRot** function behaves in precisely the opposite way. The signs of the currently valid rotation angles are inverted, and the order of calculation is X, Y and Z. In other words, the vector is turned back, so to speak.

Neither CalcRot nor CalcInvRot generate any geometry, but merely carry out the calculation of the vector.

Sample:

```
N10 G01 X40 Y10 Z0 F6000 (the axes are moved
without rotation)
N20 R1=40 R2=10 R3=0

N30 ROT Z45

(What is the position to which X, Y, must be taken so that no
movement is executed?)
N40 CalcInvRot[R1; R2; R3]
N50 G01 X=R1 Y=R2 Z=R3 (R1=35.35 R2=-21.21 R3=0)
N60 ...
```

Command	RotVec[R<x>; R<y>; R<z>; R<α>; R<β>; R<γ>]
Parameter	The 3 R-parameters (x..z) describe the vector to be rotated through. The calculation will write the result into this R-parameter, and the original value will therefore be overwritten. The last 3 R-parameters describe the angle.

The function **RotVec** rotates a three-dimensional vector through the specified angle. The order of the rotation is Z, Y and X, like for ROT. RotVec is a calculation routine solely for rotating a vector. It has no effect on ROT or AROT.

4.2.13 Mirror

The mirror functionality changes the sign of named axes. This enables subroutines to be reused.

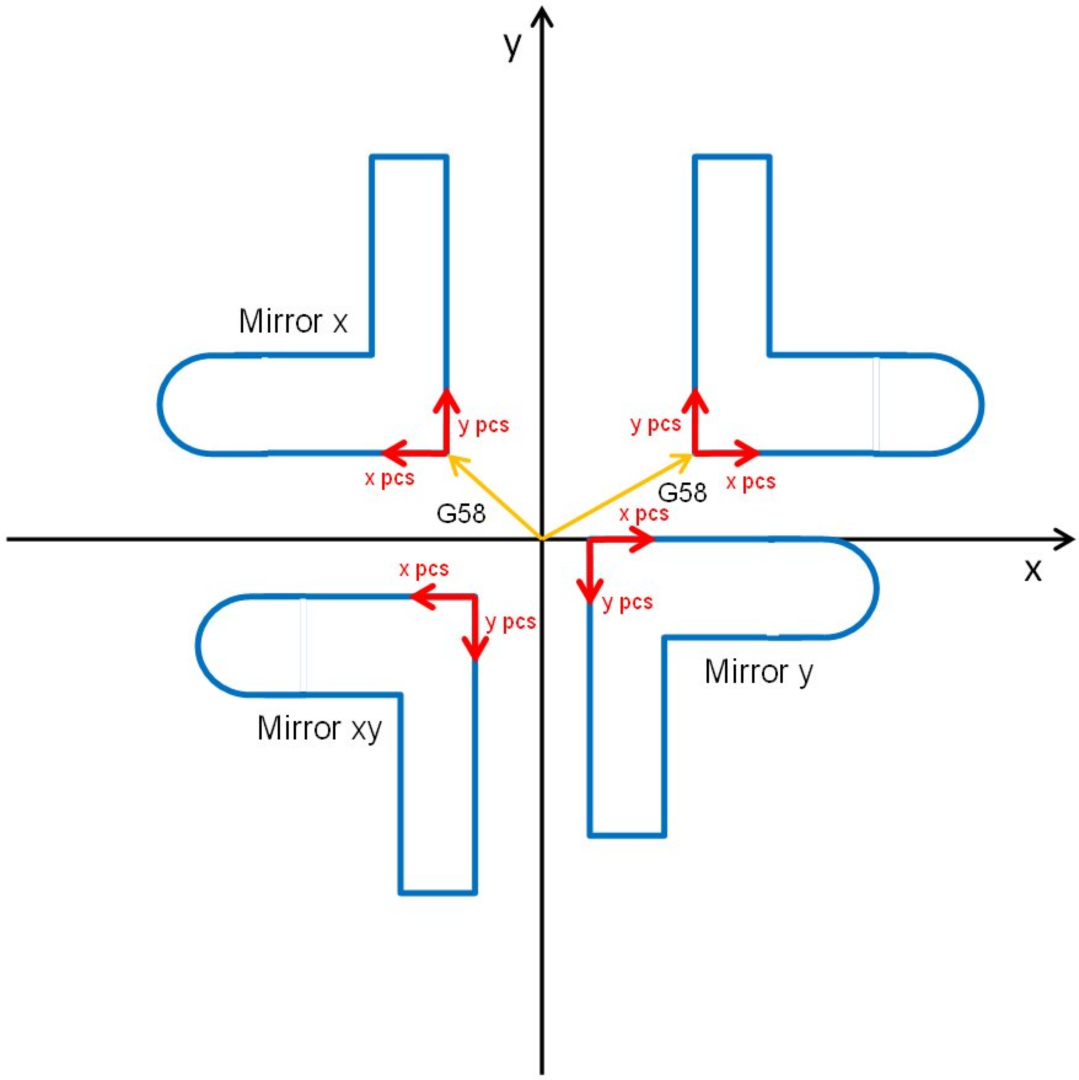
Mirroring

Command	Mirror <opt. X> <opt. Y> <opt. Z>
Cancellation	Mirror (without parameters)

The mirror instructions must be programmed in a dedicated block. Mirrored axes must be named without further parameters.

Sample:

```
N20 G54
N30 G58 X100 Y100
N40 L100

N50 G58 X-100 Y100
N60 Mirror X
N70 L100

N80 G58 X-50 Y-50
N90 Mirror X Y
N100 L100

N110 G58 X10 Y-10
N120 Mirror Y
N130 L100

N140 Mirror (turn off mirror)
N150 G0 X0 y0
M02

L100
N1000 G0 X200 Y0 Z10 F60000 (move to start pos)
N1020 G01 Z0
N1030 G03 X200 Y100 J50
N1040 G01 X50
N1050 G01 Y400
N1060 G01 X0
N1070 G01 Y0
N1080 G01 X200
N1090 G01 Z10
M17
```



If a zero shift is present (G54...G59), the mirror functionality depends on the currently programmed coordinate system.

4.2.14 Smoothing of segment transitions

4.2.14.1 Overview

Overview

In general, at segment transitions polygon functions (`G01` blocks) contain kinks within their contour. At these transitions polygon functions are not steadily differentiable with respect to their spatial coordinate, thus leading to dynamic-unsteadinesses, if at these transitions the path velocity is not reduced to zero value. To actually avoid to have to reduce path velocity to zero value segment transitions of polygon functions can be smoothed out by blending at those transitions.

	Execution	Supported Segment Transitions	Acceleration of Axis Components	Max. Tolerance	Adaptive Tolerance Radius	Command
Circular Smoothing [▶ 138]	Interpreter	Straight line/ straight line	Step change in acceleration (value parameterizable via the `C1` factor)	1/2 of the input or output segment	No	`paramCircularSmoothing(…)`
Parabolic Smoothing [▶ 135] <type>: **2**	NC kernel	Straight line/ straight line	Step change in acceleration to a constant level (value parameterizable via the `C1` factor)	1/3 of the input or output segment	Can be selected	`paramVertexSmoothing(…)`
Biquadratic Smoothing [▶ 135] <type>: **3**	NC kernel	Straight line/ straight line	Constant acceleration - the acceleration is 0 at the entry and exit - no intermediate point required	1/3 of the input or output segment	Can be selected	`paramVertexSmoothing(…)`
Bézier Curve of the 3rd Order [▶ 135] <type>: **4**	NC kernel	All	Step change in acceleration to a linear level (can be parameterized with the `C1` factor)	1/3 of the input or output segment	Can be selected, has an effect for straight-line transitions	`paramVertexSmoothing(…)`
Bézier Curve of the 5th Order [▶ 136] <type>: **5**	NC kernel	All	Constant acceleration - the acceleration is 0 at the entry and exit - no intermediate point required	1/3 of the input or output segment	Can be selected, has an effect for straight-line transitions	`paramVertexSmoothing(…)`
'Old' Bézier Blending [▶ 136] <type>: **1**	NC kernel	All	Constant acceleration - the acceleration is 0 at the entry, the exit and at the symmetric intermediate point	1/4 of the input or output segment	No	`paramSplineSmoothing(…)` `paramVertexSmoothing(…)`

Blending takes effect from the transition between the subsequent two segments.

Note	**Principle of Blending** The radius of the tolerance sphere can be altered at any time within the NC program and can be switched off again by setting the radius to 0. Blending remains active until the next reset of the interpreter or a TwinCAT runtime restart.

4.2.14.2 Parabolic smoothing

Parabola smoothing

Command	#set paramVertexSmoothing(<type>; <subtype>; <radius>)#
Parameter <type>	For parabola smoothing: **2**
Parameter <subtype>	1: Constant tolerance radius [▶ 138] 2: Distance between intersection and vertex [▶ 138] 3: Adaptive tolerance radius [▶ 138]
Parameter <radius>	Max. radius of the tolerance sphere

For parabola smoothing a parabola is inserted geometrically into the segment transition. This ensures a steady velocity transition within the tolerance radius.

The parabola is only inserted for straight line/straight line transitions.

4.2.14.3 Biquadratic smoothing

Bi-quad smoothing

Command	#set paramVertexSmoothing(<type>; <subtype>; <radius>)#
Parameter <type>	For parabola smoothing: **3**
Parameter <subtype>	1: Constant tolerance radius [▶ 138] 2: Distance between intersection and vertex [▶ 138] 3: Adaptive tolerance radius [▶ 138]
Parameter <radius>	Max. radius of the tolerance sphere

With biquadratic smoothing there is no step change in acceleration in the axis components. With the same radius, a smaller input velocity may therefore be required than for parabolic smoothing.

The operating principle of the subtypes is identical to that of the parabolic subtypes.

4.2.14.4 Bezier curve of the 3rd order

Bezier curve of the 3th order

Command	#set paramVertexSmoothing(<type>; <subtype>; <radius>)#
Parameter <type>	for the Bezier curve of the 3th order: **4**
Parameter <subtype>	1: Constant tolerance radius [▶ 138] 2: Distance between intersection and vertex [▶ 138] 3: Adaptive tolerance radius [▶ 138]
Parameter <radius>	Max. radius of the tolerance sphere

In case of the 3rd order Bezier curve a step change in acceleration appears in the axis components when the tolerance sphere is entered. The max. size is limited by the acceleration of the axis components and the C1 factor.

This blending can be used for all segment transitions. The subtypes 2 and 3 only work for straight line / straight line transitions.

Note	**Acute angles at the segment transition** The Bezier splines are generated by default, even at very acute angles. In order to avoid the dynamic values being exceeded, a considerable reduction velocity is required in this case. However, since the dynamics are held constant in the spline, the movement across the spline can be quite slow. In this case it is often practical to start the segment transition with an accurate stop. The command AutoAccurateStop [▶ 139] can be used to avoid having to calculate the angles manually.

4.2.14.5 Bezier curve of the 5th order

Bezier curve of the 5th order

Command	#set paramVertexSmoothing(<type>; <subtype>; <radius>)#
Parameter <type>	for the Bezier curve of the 5th order: **5**
Parameter <subtype>	1: Constant tolerance radius [▶ 138] 2: Distance between intersection and vertex [▶ 138] 3: Adaptive tolerance radius [▶ 138]
Parameter <radius>	Max. radius of the tolerance sphere

With 5th order Bezier blending, **no** step change in acceleration occurs in the axis components on entry into the tolerance sphere. In other words, the path axis acceleration is always constant if blending is selected.

This blending can be used for all segment transitions. The subtypes 2 and 3 only work for straight line / straight line transitions.

Note	**Acute angles at the segment transition** The Bezier splines are generated by default, even at very acute angles. In order to avoid the dynamic values being exceeded, a considerable reduction velocity is required in this case. However, since the dynamics are held constant in the spline, the movement across the spline can be quite slow. In this case it is often practical to start the segment transition with an accurate stop. The command AutoAccurateStop [▶ 139] can be used to avoid having to calculate the angles manually.

4.2.14.6 Old Bezier blending type

Note	**Functions for compatibility with existing projects** These functions are provided for compatibility reasons. For new projects Bezier curve of the 3rd order [▶ 135] or Bezier curve of the 5th order [▶ 136] should be used.

Old Bezier blending with paramVertexSmoothing

Command	#set paramVertexSmoothing(<type>; <subtype>; <radius>)#
Parameter <type>	For Bezier Spline smoothing: **1**
Parameter <subtype>	For Bezier Spline smoothing: **1**
Parameter <radius>	radius of the tolerance sphere

Sample 1:

```
N10 R57=100
#set paramVertexSmoothing(1; 1;R57)#
```

Old Bezier blending with paramSplineSmoothing

With the aid of smoothing, it is possible to insert a Bezier spline automatically between two geometrical entries. It is only necessary to program the radius of the tolerance sphere. This describes the maximum permissible deviation from the programmed contour in the segment transition. The advantage of this type of smoothing as opposed to rounding with an arc is that there are no step changes in acceleration at the segment transitions.

The radius of the tolerance sphere can be altered at any time within the NC program, and can be switched off again by setting the radius to 0. If the radius is not reset to 0, it remains active until the next interpreter reset or TwinCAT restart.

Command	#set paramSplineSmoothing(<radius>)#
Parameter <radius>	Radius of the tolerance sphere

or alternatively

#set paramVertexSmoothing(...)

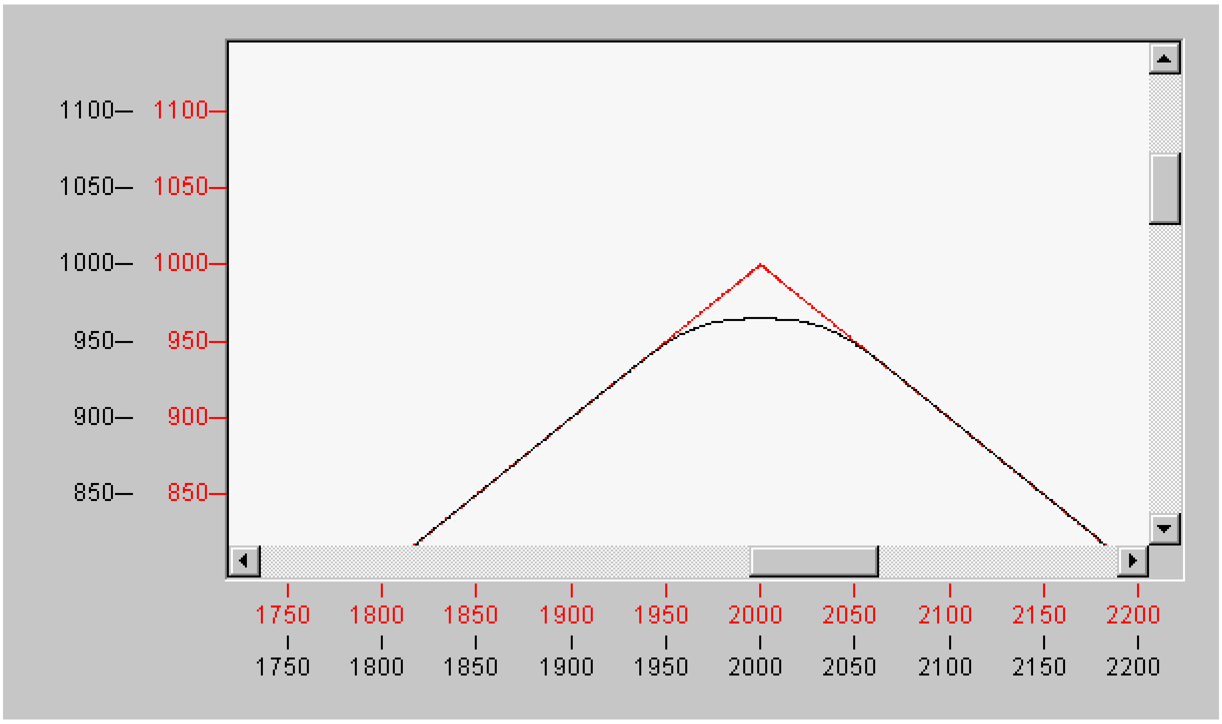
Sample 1:

```
N10 R57=100
#set paramSplineSmoothing(R57)#
```

Sample 2:

```
N10 G01 X0 Y0 F6000
N20 X1000
#set paramSplineSmoothing(100)#
N30 X2000 Y1000
N40 X3000 Y0
M30
```

The new parameter is valid from the transition between the subsequent two segments. In example 2, the new value for the tolerance sphere is applicable at the segment transition from N30 to N40. The diagram below shows a contour with and without spline at the segment transition.

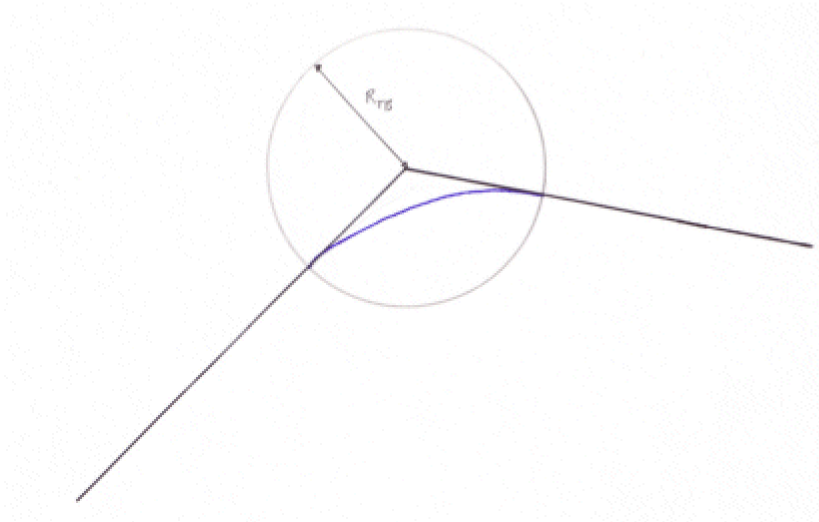


The splines are generated even at very sharp angles by default. In order to avoid the dynamic values being exceeded, a considerable reduction velocity is required in this case. However, as the dynamics are held constant, the movement across the spline can be quite slow. In this case it is often practical to start the segment transition with an accurate stop. In order to avoid manual calculation of the angles, an 'AutoAccurateStop [▶ 139]' command is available which can also be initiated via the NC program.

4.2.14.7 Subtypes

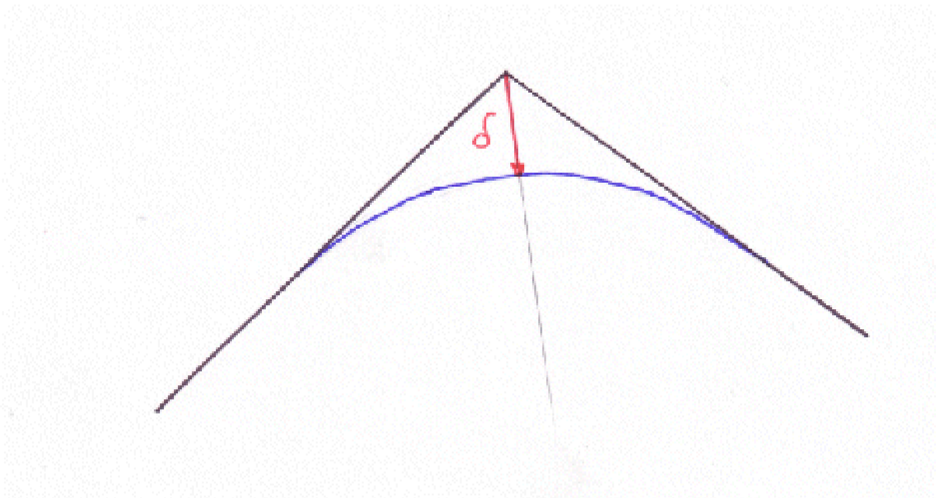
Constant tolerance radius (subtype 1)

If subtype 1 is selected, the maximum tolerance radius (R_{TB}) is used for blending. R_{TB} is reduced if and only if the input or output segment is less than 3*R_{TB}.



Distance between intersection and vertex (subtype 2)

The distance between the programmed segment transition and the vertex of the parabola is specified with the subtype 2. The tolerance radius (R_{TB}) results from this. If a segment is too short, then the distance is shortened so that the tolerance radius is a max. of 1/3.



Adaptive tolerance radius (subtype 3)

Within the tolerance radius (including constant tolerance radius) the system ensures that the maximum permissible acceleration is not exceeded. Depending on the deflection angle and the velocity, the maximum axis acceleration within the smoothing segment may be different. The aim of an adaptive tolerance radius is maximum acceleration during smoothing. In order to achieve this, the smoothing radius is reduced based on the programmed velocity and dynamics. In other words, if the programmed velocity is changed, the tolerance radius can also change. The override has no influence on the radius.

4.2.15 Circular Smoothing

It is possible with the aid of circular smoothing to insert an arc automatically between two straight lines. It is only necessary to program the radius of the arc.

The radius of the circular smoothing can be altered at any time within the NC program, and can be switched off again by setting the radius to 0. Rounding must be switched off before the end of the program or a decoder stop [▶ 150].

Command	#set paramCircularSmoothing(<radius>)#
Parameter <radius>	Radius of the circular smoothing arc

Sample:

```
N10 R57=4.5
#set paramCircularSmoothing(R57)#
…
#set paramCircularSmoothing(0)#
N1000 M02
```

NOTE! When combined with cutter radius compensation, please note that first the radius compensation is calculated, then the circular smoothing is added. The smoothing radius thus refers to the TCP.

NOTE! The old command paramGroupVertex continues to be supported. However, it cannot be used to transfer R parameters.

Syntax:

```
#set paramGroupVertex(<grp>,<radius>)#
```

The first parameter describes the group to which the circular smoothing refers. This value is currently always 1. The second parameter is used to specify the circular smoothing radius.

4.2.16 Automatic Accurate Stop

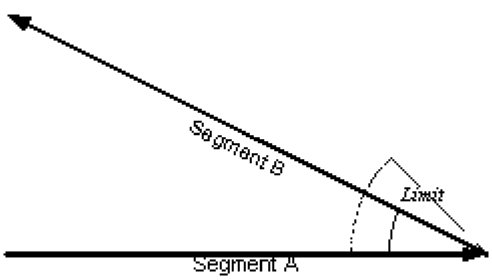
Command	#set paramAutoAccurateStop(<angle>)#
Parameter <angle>	Limit angle (in degrees) after which an accurate stop is inserted
Deselect	#set paramAutoAccurateStop(**0**)#

An accurate stop after a defined limit angle is inserted between two segments with the aid of the 'AutoAccurateStop' command.

For circle segments, the angle is calculated from the tangents at the points of entry and leaving.

Sample:

```
#set paramAutoAccurateStop(45)# (angle in
degrees)
N10 G01 X1000 Y0 Z0 F60000 (start position: X0 Y0 Z0)
N20 X0 Y500
...
```



An accurate stop is inserted between segments A and B in this example.

Application field:

This command should be used in conjunction with Bezier blending, if acute angles are programmed in the NC program.

See also:

- Bezier curve of the 3th order [▶ 135]
- Bezier curve of the 5th order [▶ 136]
- 'Old' Bezier curve [▶ 136]

NOTE! This function has not yet been implemented for segment transitions with a helix.

4.2.17 Delete Distance to Go

Command	DelDTG
Cancellation	End of block

The DelDTG (**del**ete **d**istance **t**o **g**o) command is activated block by block via the NC program. This command enables deleting of the residual distance of the current geometry from the PLC with the function block ItpDelDtgEx [▶ 191]. In other words, if the command is issued while the block is processed, the motion is stopped with the usual deceleration ramps. The NC program then processes the next block. An error message is generated if the PLC command is not issued during the execution of a block with "delete distance to go" selected.

The "delete distance to go" command always effects an implicit decoding stop, i.e. an exact positioning always occurs at the end of the block.

Sample:

```
N10 G01 X0 Y0 F6000
N20 DelDTG G01 X2000
N30 G01 X0
```

NOTE! DelDTG must not be active when cutter radius compensation is active.

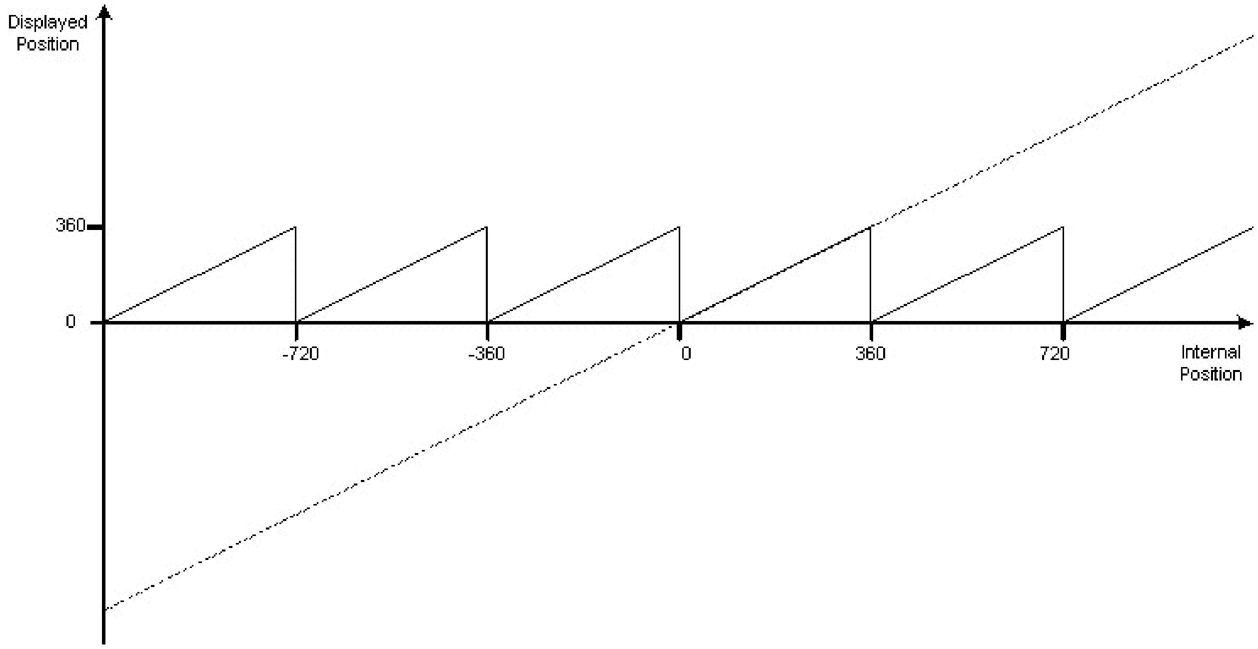
4.2.18 Modulo Movements

Command	MOD[<axis and target modulo position>]
Cancellation	End of block
Parameter 1	Axis for modulo operation
Parameter 2	Arithmetic sign for the direction of rotation (optional)
Parameter 3	Modulo position

The modulo position is programmed in the same way as normal positioning.

The MOD command is effective for specific blocks, and must be explicitly programmed for every axis that is selected for modulo operation. The modulo position's arithmetic sign specifies the direction of rotation.

- Positive sign: The axes moves in the 'larger' direction
- Negative sign: The axes moves in the 'smaller' direction
- Exception: The axis cannot move to modulo -0, since zero has no sign



Sample 1:

```
N10 G90
N20 G01 MOD[X200] Y30 F600
N30 G01 X200
```

N20 specifies a move in a positive direction for X to modulo position 200. Y moves to absolute position 30. In block N30, X is moved absolutely to position 200, i.e. **not** modulo.

Modulo movements of more than 360 degrees

The MOD command also allows movements of more than a 360 degrees to be made.

Modulo position = number of necessary rotations * 360 + modulo position

Sample 2:

```
N10 G90
N20 G01 X3610 F6000
N30 R1=360
N40 G01 MOD[X=R1+20]
```

In this example, the X axis moves 370 degrees to modulo position 20.

Restrictions and notes of for modulo movements:

- No radius compensation may be active for the modulo axis.
- No zero shift may be active for the modulo axis.
- During relative programming (G91 [▶ 111]) the modulo command is not evaluated, so that the axis referred to in square brackets is treated as if the MOD command had not been given.

Modulo factor

The modulo factor is constant, and is 360.

4.2.19 Auxiliary axes

Auxiliary axes (also known as Q axes) can be added to an interpolation group in addition to the actual path axes (X, Y & Z). The auxiliary axis can be seen as a type of slave for the path, i.e. it has no direct influence on the path velocity. In addition to the 3 path axes, 5 auxiliary axes can also be interpolated for each channel.

The function block 'CfgBuildExt3DGroup [▶ 182] ' from the library Tc2_NCI, for example, may be used for adding to the interpolation group from the PLC.

Syntax

The auxiliary axes are addressed as Q1..Q5 from the part program. The numerical value may be assigned directly, or an R-parameter.

Sample 1:

```
(start position X=Y=Z=Q1=0)
N10 G01 X100 Q1=47.11 F6000
...
```

If an NC block is programmed with one or more path axes and an auxiliary axis, both axes start **simultaneously** and also reach the destination **together**.

Swiveling of the auxiliary axes

The term "swiveling of the auxiliary axes" is used if the path length within a motion set is zero. This is often the case during 'swiveling' of a tool, with the feed angle relative to the contour being changed.

Since the path length is zero, there is no link to the path, and the movements of the auxiliary axes are calculated via a virtual path. However, this has no influence on the real path of X, Y and Z, but here too all auxiliary axes are started simultaneously and also arrive at the destination simultaneously.

Here too, the velocity is specified via the F-parameter and now refers to the auxiliary axis with the greatest travel distance.

Sample 1:

```
(start position X=Y=Z=Q1=Q1=0)
N10 G01 X100 F6000
N20 Q1=100 Q2=200 F3000
...
```

In N20, the velocity of Q2 is now 3000 and that of Q1 is 1500, since the travel distance is Q1=Q2/2.

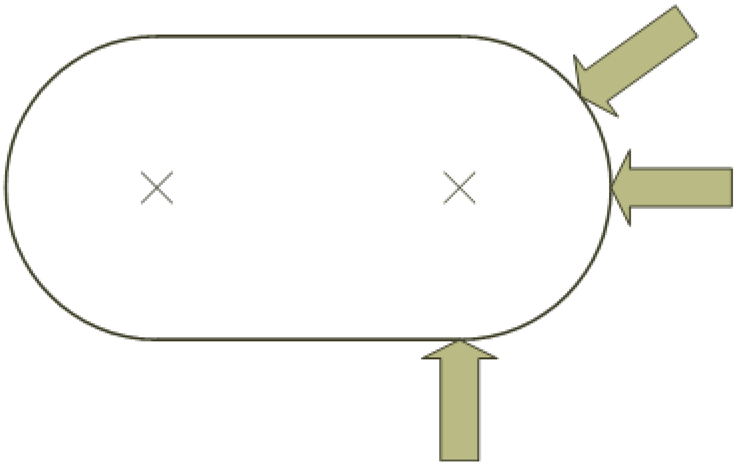
4.2.19.1 Calculation of the velocity

Initially, only the path axes (X, Y and Z) are considered for the calculation of the path velocity.
The path and the travel distance of the individual auxiliary axes result in a fixed coupling ratio for each auxiliary axis within a segment. The target velocity of the auxiliary axis is thus also known. If this velocity is greater than the permitted maximum velocity for this auxiliary axis, the path velocity is reduced until the upper speed limit is adhered to. In other words, exceeding of the velocity limits of the auxiliary axes also has an indirect effect on the path velocity.

4.2.19.2 Path velocity at segment transitions

The reduction of the path velocity is explained below by means of an example. The contour of a stadium is particularly suitable for this purpose. The aim is for the feed angle of a tool relative to the path tangent to remain constant.

On the stadium straight, the orientation of the tool remains constant, i.e. the tool is not turned. In contrast, the orientation relative to the base coordinate system must be changed continuously within the circle. Assuming the path velocity in the transition between straight and circle is not reduced to zero, a step change in velocity is inevitably generated for the swiveling axis (but not for the path axes!).



This step change in velocity of the auxiliary axis is freely parameterizable and depends on the machine. Extreme cases would be for the path velocity at such segment transitions to be reduced to zero, or for the velocity not to be reduced at all.

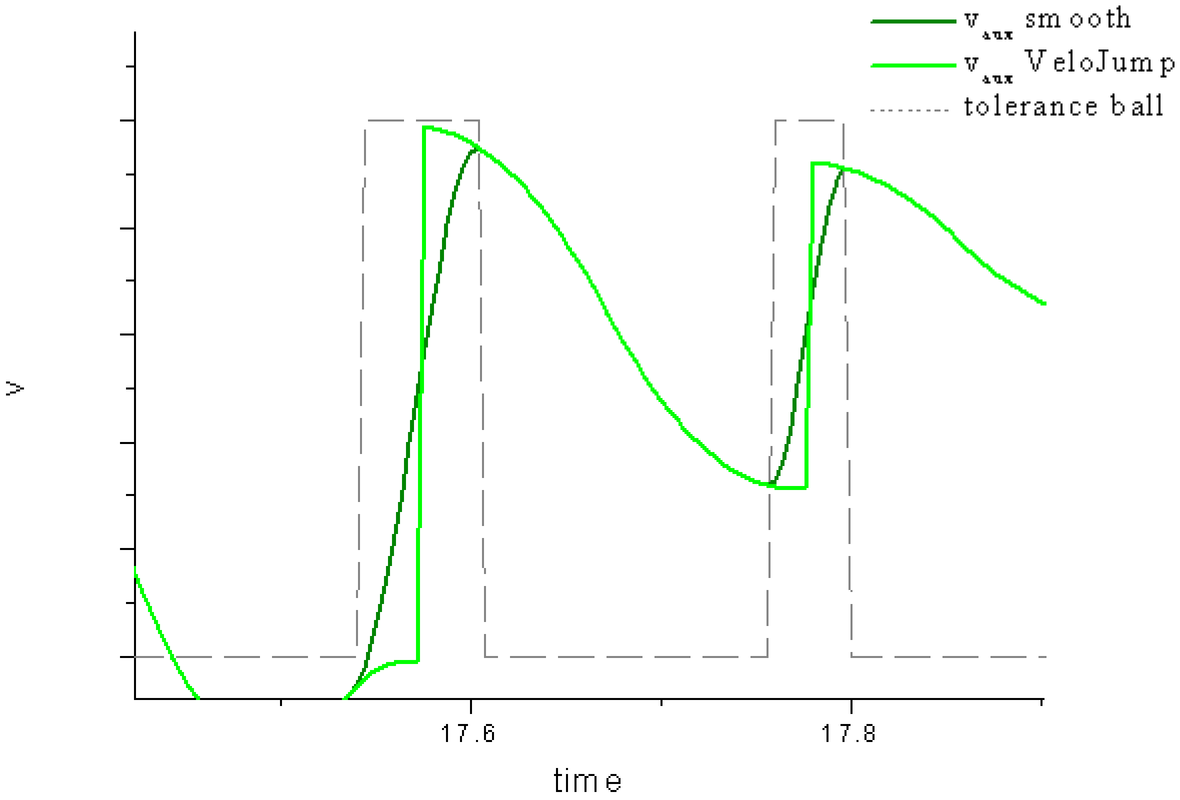
The global axis parameter 'VeloJumpFactor', which can be set individually for each axis, is used for the parameterization. The resulting velocity and the calculation is described in more detail in the TwinCAT NCI appendix on page Parameterization [▶ 301].

Smoothing of the velocity at segment transitions

As has been described above, step changes in velocity can occur at the segment transitions. The size of these steps can be affected by the VeloJump parameter.

It is further possible for a tolerance sphere to be specified for every auxiliary axis. This sphere is symmetrical with the path at a segment transition. On entering this sphere, the velocity of the auxiliary axis is continuously modified to reach the set velocity at the exit of the sphere. The step changes in velocity are, in other words, eliminated. This does imply that the auxiliary axis is subject to a positional error when it is within the sphere. On entering the sphere the change to the new target velocity of the axis starts immediately. This avoids an overshoot in position, and the position is again precise at the borders of the sphere.

If it happens that the specified sphere is larger than 1/3 of the path, its radius is automatically restricted to that value.



Selection and Deselection

The tolerance sphere of the auxiliary axis is an axis parameter (IO: 0x108). It can be set in the axis interface in the XAE or via ADS [▸ 407].

Note	**Parameterization of the axis parameters** The parameters described here only take effect for axes that are in the interpolation group as auxiliary axes (Q1..Q5). For path axes (x,y,z), the parameters 'Veloc. discontinuity factor', 'Tolerance sphere auxiliary axis' and 'Max. positional deviation, auxiliary axis' have no influence.

Diagnosis

It is possible to record the tolerance sphere of each auxiliary axis and the position error that results from this for diagnostic purposes. It is also possible to access the variables via ADS. They are to be found in the group status [▸ 396] (IO: 0x54n and 0x56n).

Effect on VeloJump, if the size of the tolerance sphere is reduced

If the size of the tolerance sphere has to be reduced due to the given geometry, the VeloJump parameter is automatically adjusted for this segment transition. I.e. the path velocity in the transition is reduced more strongly. So the dynamics of the auxiliary axis is not exceeded for smaller tolerance spheres.

Positional deviation of the auxiliary axis if the tolerance sphere has to be reduced

The parameter 'maximum permitted positional deviation of the auxiliary axis' **only** takes effect if the tolerance sphere would have to be reduced due to the geometry.

The aim is to keep the path velocity high despite the smaller tolerance sphere, as long as the resulting position error does not exceed a threshold value. To this end the velocity of the auxiliary axis is kept constant and the position error is calculated. If the error is smaller than the maximum positional deviation the velocity is maintained for this segment transition, and the resulting position error is compensated in the next segment (the tolerance sphere then becomes unnecessary for this segment transition).

In the event that the position error would exceed the maximum deviation, the reduced tolerance sphere takes effect, including the VeloJump factor. And the path velocity is reduced if necessary.

Sample 1:

Initial conditions:

- Set tolerance sphere: 5
- Max. positional deviation: 1
- The given geometry results in an effective tolerance sphere of 0.2, for example
- The potential positional deviation is 0.3

Resultant behavior:

- The path velocity remains at a constant high level
- The velocity of the auxiliary axis is kept constant
- For this transition no tolerance sphere is required
- The resulting positional deviation is compensated in the subsequent segment

Sample2:

Initial conditions:

- Set tolerance sphere: 5
- Max. positional deviation: 1
- The given geometry results in an effective tolerance sphere of 1.2, for example
- The potential positional deviation is 1.1

Resultant behavior:

- The tolerance sphere is adjusted
- The VeloJump parameter is adjusted
- The path velocity is reduced at the segment transition
- There is **no** positional deviation that has to be compensated

Parameterization

The parameterization of the maximum permitted positional deviation is an axis parameter [▸ 407]. By default this feature is switched off (deviation = 0.0)

4.3 Supplementary Functions

4.3.1 M-Functions

Task: Signal exchange between NC and PLC

A range of equipment, such as collet chucks, drill drives, transport equipment etc. is best not driven directly by the NC, but indirectly, using the PLC as an adapting and linking controller. This makes it easy to consider feedback or safety conditions, without having to adapt the NC program, or even the NC system. The NC’s M-functions involve digital signal exchange: functions are switched on or off, activated or deactivated. The transfer of numerical working parameters is not provided for here, but can be implemented in other ways (H-functions [▸ 149], T-numbers [▸ 149] etc.).

4.3.1.1 Available M-functions

Number of M-functions

A total of 160 M-functions are available per channel

M function	Meaning
0..159	Freely definable M-functions (except 2, 17, 30)
2	Program end
17	End of subprogram
30	Program end with deletion of all fast M functions

All M-functions (apart from the 3 pre-defined M-functions - M2, M17, M30) are freely definable. This means that, depending on the machine type, M8 can be used to switch on a cooling medium or indeed for any other functionality, for example. The machine manufacturer can select the function as required.

Like any other rules, the rules for reserved M-functions are read when TwinCAT is started. Additionally, an internal code is generated for these functions in the interpreter, which is responsible for the behavior described. These 3 M-functions therefore do not have to be described in the table. It makes sense to parameterize M2 and M30, even if M-functions are used.

Types of M-functions

Basically, two signal exchange versions are available: fast signal bits, or transfer secured by handshake.

Secured Handshakes

M-functions that require feedback must be processed using bi-directional signal exchange between the NC and the PLC. If an M-function of type handshake is programmed, the velocity is reduced to 0 at this point. The PLC uses the ItpIsHskMFunc [▸ 210] function to check whether an M-function with handshake is present, in which case the number of the M-function is determined via ItpGetHskFunc [▸ 210]. The NC is in a waiting state and will not process further NC commands until the PLC has acknowledged the M-function. Processing of the NC program continues once acknowledgement has been received from the PLC (ItpConfirmHsk [▸ 190]).

This procedure permits the operation of the equipment controlled by the NC to be securely coordinated with the equipment controlled by the PLC. It is therefore advisable to acknowledge the M-function for starting the spindle (e.g. M3) once a minimum speed has been reached.

Since this kind of M-function involves synchronous functions, it is only ever possible for one M-function with handshake to be active in the NC program.

Fast signal bits

If no feedback is required from the PLC, fast signal bits can be used for activating M-functions. Since the NC does not have to wait for the PLC with these M-functions, look-ahead [▸ 108] can combine the segments. In this way it is possible to apply an M-function without velocity reduction.

This type of M-function is useful for in-motion activation of a nozzle for applying adhesive, for example.

A combination of fast signal bits and handshake is also possible. Since a handshake always requires acknowledgement from the PLC, the velocity has to be reduced to 0 in this case.

4.3.1.2 Resetting of M-functions

Resetting fast signal bits

The signal bits are active until they are reset explicitly, or until an M30 (end of program) or channel reset is executed.

Resetting with reset list

Each M-function can reset up to 10 fast M-functions. If cooling medium is switched on with M8, for example, the cooling medium can be switched off again with M9. To this end simply enter M8 in the reset list for M9.

Automatic reset

During parameterization of the M-function an 'auto-reset flag' can be set. This means that the M-function is automatically reset at the end of the block.

In order for the PLC to be able to see the signal, the duration of the motion block must be long enough, or this M-function is combined with a handshake. The handshake may come from the same or a different M-function.

Reset from the PLC

The fast M-functions can be reset from the PLC via the ItpResetFastMFunc [▶ 263] function block. For reasons of transparency, mixed resets using via PLC and NC should be avoided.

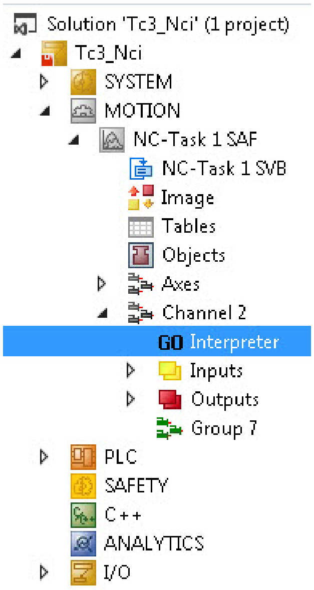
Delete all pending M functions

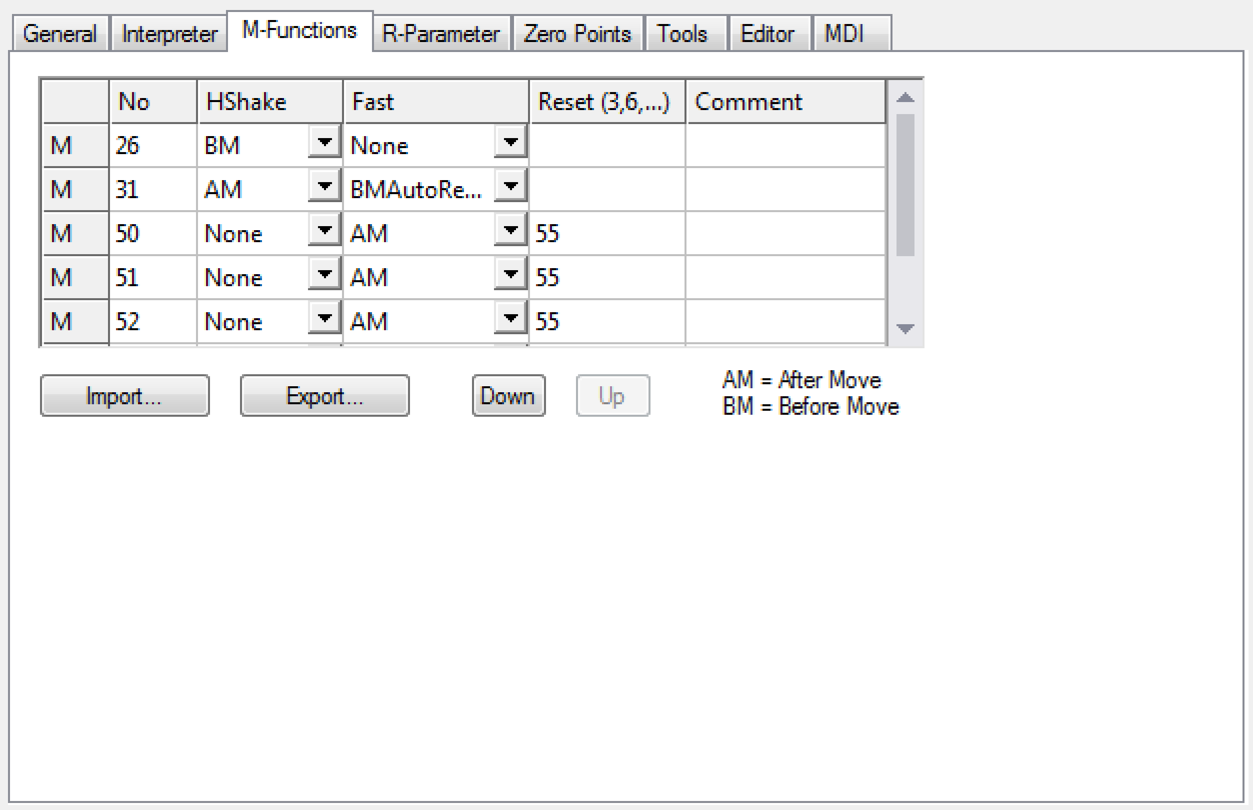
A channel stop and a channel reset are used to reset all pending M functions. This is true for the 'handshake' type M functions, and also for the fast signal bits. If the NC program is terminated properly with M30, all M-functions are also cleared.

4.3.1.3 Parameterization of M-functions

The M-functions are parameterized in TwinCAT XAE. A dedicated M-function table is used for each interpolation channel.

A restart of the TwinCAT configuration is required to activate a configuration of M-functions.





No

Number of M-function to be parameterized. The value must be between 0 and 159

HShake

If a value other than 'None' is entered, the M-function is of type 'Handshake'

- *None*: No handshake
- *BM* (Before Move)**:** If a movement is programmed in the same block, the handshake is completed **before** the movement.
- *AM* (After Move): If a movement is programmed in the same block, the handshake is completed **after** the movement.

Fast

If a value other than 'None' is entered, a 'fast signal bit' type M-function is executed

- *None*: No fast M-function is executed
- *BM* (Before Move): If a movement is programmed in the same block, the output is completed **before** the movement.
- *AM* (After Move): If a movement is programmed in the same block, the output is completed **after** the movement.
- *BMAutoReset* (Before Move & Automatic Reset): If a movement is programmed in the same block, the output is completed **before** the movement. In addition, the M-function is automatically cancelled at the end of the block, i.e. the M-function is active on a per-block basis. In order to ensure that the PLC recognizes the M-function, the duration of the associated motion block must be long enough (at least 2 PLC cycles), or an additional M-function with handshake should be programmed.
- *AMAutoReset* (After Move & Automatic Reset): This parameterization is only meaningful if either an M-function of type handshake is programmed at the same time (or parameterized), or if the M-function is only used for resetting other M-functions. Without an additional handshake the PLC will usually not be able to detect this M-function.
- All other combinations can be selected for compatibility reasons.

Reset

Up to 10 M-functions can be entered for cancellation when a reset is called.

NOTE! In the event that no reset-signal-bit is in fact set, the bits to be cleared are reset immediately before setting the new signal bits.

Import/Export

The M-functions are parameterized individually for each channel. The parameterization can be transferred to other channels via the import/export function.

4.3.1.4 Combination of M functions

- Within each line, only **one** 'handshake' type M function must be programmed!
- Within a single line, up to 10 'signal bit' M functions may be programmed
- A combination of the two options above is allowed

Sample:

```
N10 G01 X1000 F60000
N20 M10 M11 M12 X2000 (M10 & M11 are signal bits)
(M12 is of type handshake)
M30
```

Examples of meaningful and practically applicable rule combinations:

- An M-function is to be active for the duration of a movement and then be automatically cleared. Select 'None' in the HShake column and 'BMAutoReset' in the Fast column. The signal bit generated could, for instance, control a glue application valve.
- An M-function starts a drill motor, and the subsequent movements may only be started after an appropriate run-up time, and then only when the drill is ready for operation. Select 'BM' in the HShake column. The PLC acknowledges the request after a certain delay time and only if the frequency converter is ready for operation.
- A drill motor is started with an M-function. In order not to have to wait for the drive to run up, the M-function is programmed in the block before the one for the drill movement. In the following movement (the drill movement itself) it is however still essential to ensure that the drive has reached its full rotation speed. For this variant either two different M-functions have to be used (lead signal as signal bit, safety query as handshake) or a Fast 'BMAutoReset' and HShake 'AM' M-function is used.

4.3.1.5 Behavior in case of an error

If a runtime error occurs during the execution of an NC program (e.g. following error monitoring is activated), the NC program is interrupted. In this case the M functions, provided they are set, remain pending. This means that the PLC program may have to ensure that M functions are not executed.

4.3.2 H, T and S Parameters

H-, T- and S-parameters are used to transfer parameters from the NC interpreter to the PLC.

In this context the H-parameter stands for auxiliary parameter and is of type DINT (32 bit signed).

The T and S parameters are of type WORD, and stand for Tool and Spindle.

Sample:

```
H=4711
R1=23
S=R1
T4711
```

NOTE! No R-parameter can be assigned for the T-parameter. Furthermore, the assignment is made without assignment operator ('=').

T- and S-parameters take effect at the start of a block, H-parameters take effect at the end of the programmed block.

4.3.3 Decoder stop

Code	Function
@714 [▶ 150]	Decoder stop
@716 [▶ 150]	Decoder stop with axis position rescan
@717 [▶ 151]	Decoder stop with trigger event, conditional decoder stop

4.3.3.1 Decoder stop (@714)

The interpreter offers the option to execute a decoder stop in the NC program. In this case the interpreter waits until a certain external event occurs. Execution of the NC program does not continue until this event has taken place.

A decoder stop can be used, for instance, to switch block skipping [▶ 108] on or off from the PLC, or to re-assign R parameters [▶ 115].

Two events are available for continuing processing:

- Acknowledgement of an M-function [▶ 145]
- SAF task is empty

Acknowledgement of an M-function

Decoding of the NC program is interrupted until the M-function [▶ 145], which is programmed immediately prior to the decoder stop, is acknowledged. In other words, the M-function must be of type "handshake".

Sample 1:

```
N10...
N20 M43 (M-function with handshake)
N30 @714 (decoder stop)
N40 ...
```

SAF task is empty

The decoder stop does not necessarily have to be programmed in conjunction with an M-function. If the SAF task runs out of travel commands, an event is sent to the interpreter. This event causes the interpreter to start up again.

NOTE! The decoder stop must not be programmed when the tool compensation or circle smoothing are active, because they wouldn`t work anymore.

4.3.3.2 Decoder Stop with Axis Position Rescan (@716)

In addition to the common decoder stop (see Decoder stop (@714) [▶ 150]), there is a decoder stop at which the axis positions of the interpolation channel are read again. This stop is required, if, for example, axes are moved during a tool change via PTP and are subsequently not returned to the old position. Another possible application is a change in axis configuration via an M function (with handshake).

If a decoder stop with rescan is programmed, it is essential to program an M-function with handshake immediately before it.

Sample 2:

```
N10...
N20 M43 (M function with handshake carries out a tool change, for
example)
N30 @716 (Decoder stop with rescan)
N40 ...
```

NOTE! The decoder stop must not be programmed when the tool compensation or circle smoothing are active, because they wouldn`t work anymore.

4.3.3.3 Decoder Stop with external trigger event (@717)

Sometimes the question of whether the NC part of the program must wait or can continue may depend, for instance, on events in the PLC. With the two types of M-functions [▸ 145] this can give rise to the following problems:

- Handshake: Because of the M-function's handshake the path velocity must be brought to zero at the location where the M-function is programmed, after which confirmation is awaited from the PLC.
- On The Fly (also known as a fast M-function): Because no confirmation from the PLC is waited for, there is also no way for the partial program to wait for the PLC.
- Even a combination of the two types of M-function does not help here.

Sample:

During positioning with a flying M-function, a process A is initiated by the NC partial program. It is assumed here that the set of processes in the NC program is typically long enough for process A to be completed in the PLC. If A is ready, then the NC partial program should execute the next segment with look-ahead. In case A is not ready, however, then the NC should stop at the end of the segment and wait until process A has finished. It is exactly this scenario that can be implemented with the command @717. The PLC here sends the so-called 'GoAhead [▸ 207]' command when process A has finished.

```
N10 ...
N20 G0 X0 Y0 Z0
N30 G01 X500 F6000
N40 M70 (flying M-function that triggers process A)
N50 G01 X700
N60 @717 (decoder stop with external trigger event)
N70 G01 X1000
N80 ...
```

If the GoAhead signal reaches the PLC early enough, then blocks N50 and N70 are linked by look-ahead, and the path velocity is not then reduced. If the signal arrives during the deceleration phase of N50, then the velocity is once more increased. Otherwise, the machine waits for the signal from the PLC.

NOTE! The decoder stop must not be programmed when the tool compensation or circle smoothing are active, because they wouldn`t work anymore.

The function block 'ItpGoAheadEx' returns the error code 0x410A, if no @717 is present in the interpreter at the time of the call.

4.3.4 Jumps

Code	Function
@100 [▸ 152]	Unconditional jump
@121 [▸ 152]	Jump if unequal
@122 [▸ 152]	Jump if equal
@123 [▸ 152]	Jump if less or equal
@124 [▸ 152]	Jump if less
@125 [▸ 152]	Jump if greater or equal
@126 [▸ 152]	Jump if greater
@111 [▸ 152]	Case block

Unconditional jump

Command	@100
Parameter	K or R

The parameter describes the jump destination. This must have an indication of direction ('+' or '-').

Sample 1:

```
N10 ..
...
N120 @100 K-10
```

In this example, execution continues from line 10 after line 110 has been interpreted. The sign indicates the direction in which the line to be searched can be found.

Jump if unequal

Command	@121	
Parameter 1	R<n>	Comparison value
Parameter 2	K or R<m>	Comparison value
Parameter 3	K	Jump destination with direction indication

Sample 2:

```
N10 ..
...
R1=14
N120 @121 R1 K9 K-10
N130 ...
```

Jump if equal

cf. Jump if not equal [▸ 152]

Jump if less or equal

cf. Jump if not equal [▸ 152]

Jump if less

cf. Jump if not equal [▸ 152]

Jump if greater or equal

cf. Jump if not equal [▸ 152]

Jump if greater

cf. Jump if not equal [▸ 152]

Case block

Command	@111	
Parameter 1	R<n>	Comparison value
Parameter 2	K or R<m>	First comparison value
Parameter 3	K	First jump destination
Parameter 4	K or R<m>	Second comparison value
...		

Sample 3:

```
N100 R2=12 (R2=13) (R2=14)
N200 @111 R2 K12 K300
K13 K400
K14 K500

N300 R0=300
N310 @100 K5000

N400 R0=400
N410 @100 K5000

N500 R0=500
N510 @100 K5000

N5000 M30
```

A case block is made in line 200. If R2 = 12 a jump is made to line 300.

If R2 = 13, the jump destination is line 400. If R2 = 14, the jump destination is line 500.

In the event that none of the conditions is satisfied, execution simply continues with the next line (in this case, line 300).

4.3.5 Loops

The various types of loop are described below.

Code	Loop type	Aborting condition
@131	While Loop [▶ 153]	while equal
@132	While Loop [▶ 153]	while not equal
@133	While Loop [▶ 153]	while greater
@134	While Loop [▶ 153]	while greater or equal
@135	While Loop [▶ 153]	while less
@136	While Loop [▶ 153]	while less or equal
@141	Repeat Loop [▶ 154]	repeat until equal
@142	Repeat Loop [▶ 154]	repeat until not equal
@143	Repeat Loop [▶ 154]	repeat until greater
@144	Repeat Loop [▶ 154]	repeat until greater or equal
@145	Repeat Loop [▶ 154]	repeat until less
@146	Repeat Loop [▶ 154]	repeat until less or equal
@151	For-To Loop [▶ 154]	
@161	For-DownTo Loop [▶ 154]	

Loops can be nested.

While loops

Command	@13<n>	where 1<= n <= 6
Parameter 1	R<m>	Comparison value
Parameter 2	K or R<k>	Comparison value
Parameter 3	K	Jump destination for the case that the condition is not met

A while loop is executed for as long as the condition is satisfied. The test is made at the beginning of the loop. If the condition is not or no longer met, a jump to the specified line takes place (parameter 3).

At the end of the While loop an unconditional jump (@100 [▶ 151]) must be programmed. The target of this jump is the line number of the while loop.

The loop's exit condition is specified with <n>.

Sample 1:

```
N100 R6=4
N200 @131 R6 K4 K600 (K600 is the target of the jump, when the condition is no longer satisfied)
N210 ...
N220 @100 K-200

N600 ...

N5000 M30
```

The loop (lines 200 to 220) is repeated for as long as R6 = 4. Once the condition is no longer satisfied, execution jumps to line 600.

Repeat loops

Command	@14<n>	where 1<= n <= 6
Parameter 1	R<m>	Comparison value
Parameter 2	K or R<k>	Comparison value
Parameter 3	K	Jump destination at the start of the loop

In a repeat loop, the interrogation takes place at the end of the loop. This means that the loop is executed at least once. The loop is only ended, to continue with the rest of the program, when the condition is satisfied.

Sample 2:

```
N200 ...
N210 ...

N300 @141 R6 K25 K200
```

The loop is repeated until R6 = 25. The second constant in line 300 gives the jump target (the start of the loop).

For-To loops

Command	@151 <variable> <value> <constant>

A for-to loop is a counting loop that is executed until the *variable* equals the *value*. The test is made at the beginning of the loop. If that condition is satisfied, execution jumps to the line specified by the *constant*.

The variable must be incremented (@620) at the end of the loop, and there must be an unconditional jump to the start of the loop.

Sample 3:

```
N190 R6=0
N200 @151 R6 K20 K400
N210 ...
N290 @620 R6 (increment R6)
N300 @100 K-200
```

For-Downto Loops

Command	@161 <variable> <value> <constant>

A for-downto loop is a counting loop. The behaviour is similar to that of a for-to loop. The difference is merely that the variable is decremented (@621) by 1 at the end of the loop.

4.3.6 Subroutine techniques

As in other fields, it is also valuable in NC programming to organize frequently used command sequences as subroutines. This makes it possible to employ pre-prepared and tested functions in various workpiece programs.

Subroutines are identified within a program by a number. This number must be unique: there must be only one subroutine with a particular number (1..>2.000.000.000).

As interpretation proceeds, the calling program is suspended. The text of the subroutine is worked through, as often as necessary. Processing then returns to the calling program after the call location.

It is of course possible for one subroutine to call another subroutine. This call is executed in a similar way. This causes a stack of return information to be created. For technical reasons this nesting of subroutines is presently limited to 20 levels.

Definition of a Subroutine

The code for a subroutine can be written to the same file as the calling program. In this case the subroutine is linked directly: it is automatically also loaded as the file is read. If it is to be made generally available then it must be written in its own file that must be located in the CNC directory.

The name of the file begins with the letter 'L', and continues with a string of digits. This digit string must repeat the subroutine number, without any leading '0's.

The code should contain a label to indicate the starting point of the subroutine. Like the file name, it consists of the letter 'L' and the digit sequence described above.

The interpreter starts immediately after this label.

Subroutine syntax:

```
(Datei L2000.NC)
L2000
N100...
N110...
...
N5000 M17 (return command)
```

Calling a Subroutine

The following syntax must be used to call a subroutine from some block within the NC program. It is important that the expression "L2000" does not stand at the start of the line, in order to avoid confusion with a subroutine label.

```
(syntax of the subroutine call)
N100 L2000
```

In the following sample the expression "P5" causes the subroutine to be repeated 5 times.

```
(n-fold subroutine call (here: 5- fold))
N100 L2000 P5
```

Dynamic subroutine call

In some cases the subroutine to be called is not known until runtime. In this case the subroutine can be called with an R-parameter, thereby avoiding the need for a CASE instruction. The value for R must be allocated or calculated in a dedicated line.

```
(Dynamic call of a subroutine)
N099 R47=R45+1
N100 L=R47
```

Parameter passing

Parameters are passed to subroutines with the aid of R-parameters [▶ 115]. Note that R-parameters are not saved automatically (see Rescuing R-parameters [▶ 115]).

Use of Parameters

R-parameters can, in general, be freely used within subroutines. This has a number of consequences that can lead to errors if they are not borne in mind. On the other hand their careful use offers the NC-programmer a range of useful working techniques.

Results of Subroutines

If an R-parameter is changed without its contents being saved and restored, the change is effective after a return from the subroutine. If this was not intended, the result can be machine behavior that was not planned.

This feature can however be deliberately used in order to make the continuation of the processing dependent on the results of a subroutine. No restriction need be considered here other than those on the R-parameters.

Sample:

```
N100 L2000
N110 R2=R3+R4
...
N999 M30
L2000
N10 R3=17.5
N20 R4=1
N99 M17
```

Values are specified here in a subroutine. The values are then used in the calling program.

Ending a Subroutine

A subroutine is ended with M17.

4.3.7 Dynamic Override

Command	DynOvr=<value> or DynOvr = R<n>
Cancellation	DynOvr=1

Sample:

```
N10 G01 X100 Y200 F6000
N20 DynOvr=0.4
N30 G01 X500
```

'DynOvr' can be used to make percentage changes to the dynamic parameters of the axes in the group while the NC program is running. This also results in new values for the motion dynamics. The new dynamic values become valid, without any stop, when the line is executed. This means, for the example illustrated above, that in block 10 the old values will still be used for the deceleration, while the new values will be used for acceleration in block 20.

Scope of Definition

$0 < DynOvr \leq 1$

See also change in path dynamics [▸ 156].

4.3.8 Altering the Motion Dynamics

Command	#set paramPathDynamics
Parameter <acc>	Value of the maximum permitted path acceleration in mm/s^2.
Parameter <dec>	Value of the maximum permitted deceleration in mm/s^2.
Parameter <jerk>	Value of the maximum permitted jerk in mm/s^3.

Sample:

```
N10 G01 X100 Y200 F6000
N15 R4=3000
N20 #set paramPathDynamics( 700; 700; R4 )#
N30 G01 X500
```

The 'paramPathDynamics' command can be used to alter the motion dynamics as the NC program is running. The new dynamic values become effective as from the line in which they are programmed. For the example illustrated, this means that the whole of block 10 is still treated with the default values. The new parameters are used for block 30 from the start of the segment.

This command limits all path axes to the parameterized dynamic values, although the path itself can have higher dynamics, depending on its orientation. The dynamics of auxiliary axes remains unchanged.

See also dynamic override [▸ 156].

NOTE! The dynamic values changed via the NC program remain active until the interpreter is next reset and/or TwinCAT has been restarted.

NOTE! The old command 'paramGroupDynamics' continues to be valid. However, it cannot be used to transfer R parameters.

Command	#set paramGroupDynamics(<grp>,<acc>,<dec>,<jerk>)#
Parameter <grp>	Group for which the alteration of the motion dynamics is to be effective. Presently always 1.
Parameter <acc>	Value of the maximum permitted path acceleration in mm/s^2.
Parameter <dec>	Value of the maximum permitted deceleration in mm/s^2.
Parameter <jerk>	Value of the maximum permitted jerk in mm/s^3.

Sample:

```
N10 G01 X100 Y200 F6000
N20 #set paramGroupDynamics( 1, 700, 700, 3000 )#
N30 G01 X500
```

Change in axis dynamics

Command	#set paramAxisDynamics
Parameter <axis>	Axis in the interpolation group: X: 0 Y: 1 Z: 2 Q1: 3 ... Q5: 7
Parameter <acc>	Value of the maximum permitted acceleration in mm/s^2
Parameter <dec>	Value of the maximum permitted deceleration in mm/s^2.
Parameter <jerk>	Value of the maximum permitted jerk in mm/s^3.

Sample:

```
N10 G01 X100 Y200 F6000
N15 R4=30000
N20 #set paramAxisDynamics( 0; 1500; 1400; R4 )#
N30 G01 X500
```

'paramAxisDynamics' can be used to change the axis dynamics at runtime. Generally the behavior is the same as for 'paramPathDynamics', except that here the dynamics can be specified individually for each axis.

4.3.9 Change of the Reduction Parameters

C0 reduction [▸ 158]
C1 reduction [▸ 159]
C2 reduction [▸ 159]

C0 reduction

In some types of machine it is not absolutely necessary to reduce the path velocity to 0 at knee-points. 2 reduction methods are available

- VeloJump
- DeviationAngle

VeloJump

Command	#set paramVeloJump(<C0X>; <C0Y>; <C0Z>)#
Parameter <C0X>	Reduction factor for *C0* transitions: *X* axis: *C0X* ≥ *0.0*
Parameter <C0Y>	Reduction factor for *C0* transitions: *Y* axis: *C0Y* ≥ *0.0*
Parameter <C0Z>	Reduction factor for *C0* transitions: *Z* axis: *C0Z* ≥ *0.0*

The 'paramVeloJump' command can be used to alter the velocity step change factors as the NC program is running. The new values come into effect via the block execution in the programmed line. You can find further details of the means of operation in the appendix under Parameterization [▸ 301].

Sample:

```
N10 G01 X100 Y200 F6000
N20 R2=4.5
N30 #set paramVeloJump( 1.45; R2; R2 )#
N40 G01 X500
```

Note	**Resetting parameters** The VeloJump parameters changed via the NC program remain active until the interpreter is next reset and/or TwinCAT has been restarted.

DeviationAngle (not yet released)

Command	#set paramDevAngle(<C0Factor>; <AngleLow>; <AngleHeigh>)#
Parameter <C0Factor>	Path reduction factor for *C0* transitions: 1.0 ≥ *C0* ≥ *0.0*
Parameter <AngleLow>	Angle in degrees from which reduction takes effect: $0 \leq \varphi\_l < \varphi\_h \leq \pi$
Parameter <AngleHeigh>	Angle in degrees from which reduction to $v\_link = 0.0$ takes effect: $0 \leq \varphi\_l < \varphi\_h \leq \pi$

The 'paramDevAngle' command is used to describe the parameters for the C0 reduction. In contrast to the VeloJump reduction method, in which the velocity step change is influenced directly, in the DeviationAngle method the velocity step change depends upon the angle. You can find further details of the means of operation in the appendix under Parameterization [▸ 301].

Sample:

```
N10 G01 X100 Y200 F6000
N20 #set paramDevAngle(0.15; 5; 160 )#
N30 G01 X500
```

Note	**Resetting parameters** The DeviationAngle parameters changed via the NC program remain active until the interpreter is next reset and/or TwinCAT has been restarted.

C1 reduction factor

Command	#set paramC1ReductionFactor(<C1Factor>)#
Parameter <C1Factor>	C1 reduction factor

The 'paramC1ReductionFactor' command is used to change the C1 reduction factor while the NC program is running.

The new parameter comes into effect at the segment transition at which the reduction factor is programmed. In the example shown, the new value for the C1 reduction is therefore already effective in the segment transition from N10 to N30.

A floating point value or an 'R parameter' can be provided as parameter.

You can find further details of the means of operation in the appendix under Parameterization [▶ 301].

Sample:

```
N10 G01 X100 Y200 F6000
N20 #set paramC1ReductionFactor( 0.45 )#
N30 G01 X500
```

Note	**Resetting parameters** The C1 reduction factor changed via the NC program remains active until the interpreter is next reset and/or TwinCAT has been restarted.

C2 reduction factor

Command	#set paramC2ReductionFactor(<C2Factor>)#
Parameter <C2Factor>	C2 reduction factor

The 'paramC2ReductionFactor' command is used to change the C2 reduction factor while the NC program is running.

The command takes effect in the segment transition for which the reduction factor is programmed. In the example shown, the new value for the C2 reduction is therefore already effective in the segment transition from N10 to N30.

A floating point value or an 'R parameter' can be provided as parameter.

Sample:

```
N10 G01 X100 Y200 F6000
N20 #set paramC2ReductionFactor( 1.45 )#
N30 G01 X500
```

Note	**Resetting parameters** The C2 reduction factor changed via the NC program remains active until the interpreter is next reset and/or TwinCAT has been restarted.

4.3.10 Change of the Minimum Velocity

Command	#set paramVeloMin(<VeloMin>)#
Parameter <VeloMin>	Minimum path velocity

The 'paramVeloMin' command can be used to alter the minimum path velocity while the NC program is running. The new velocity comes into effect via the block execution in the programmed line.

A floating point value or an 'R parameter' can be provided as parameter.

Sample:

```
N10 G01 X100 Y200 F6000
N20 #set paramVeloMin( 2.45 )#
N30 G01 X500
```

Note	**Resetting parameters** The minimum velocity changed via the NC program remains active until the interpreter is next reset and/or TwinCAT has been restarted.

Note	**Programming the velocity** The unit of velocity is mm/sec and is therefore equivalent to the usual XAE units.

4.3.11 Read Actual Axis Value

Command	@361	
Parameter 1	R<n>	R parameter to which the actual axis value is assigned
Parameter 2	K<m>	Constant for the axis coordinate that is to be read 0: X axis 1: Y axis 2: Z axis 3: Q1 axis 4: Q2 axis . . . 7: Q5 axis

Sample 1:

```
N10 G0 X0 Y0 Z0 F24000
N30 G01 X1000
N40 @361 R1 K0 (read position of x axis)
N50 R0=X
N60 G01 X=R0+R1
N70 M30
```

A decoder stop is implicitly executed by @361 command. This ensures that, in this example, the position is read when block N30 has been processed.

A possible application would be in combination with the deletion of any remaining travel.

Read actual axes value without decoder stop

Command	#get PathAxesPos(R<a>; R<b>; R<c>)#	
Parameter 1	R<a>	R parameter to which the actual axes value of the X axis is assigned
Parameter 2	R<b>	R parameter to which the actual axes value of the Y axis is assigned
Parameter 3	R<c>	R parameter to which the actual axes value of the Z axis is assigned

The command #get PathAxesPos( )# reads the current actual positions of the path axes (X, Y & Z). It behaves similarly to @361, with the difference that this command does not trigger an implicit decoder stop. This means that the programmer must himself ensure that at the time when the command is being processed in the interpreter the axes have not yet moved, or else a decoder stop (@714) must be programmed in the block before this command.

#get PathAxesPos( )# is an alternative to @361, but it is linked to certain specific conditions.

Sample 2:

```
@714(optional)
N27 #get PathAxesPos( R0; R1; R20 )#
```

NOTE! If a path axis is not assigned (e.g. no axis is assigned to Z) the value 0 is passed to the associated R parameter.

4.3.12 Skip virtual movements

Command	#skip VirtualMovements(<parameter>)#
Parameter	0 (default): virtual movements are “completed”. 1: Virtual movements are skipped

Movements of unavailable but programmed main axes (X, Y & Z) can be skipped with the command 'skip VirtualMovements'.

Sample:

The interpolation group (CfgBuildGroup) contains only assignments for the X and Y axis. The Z axis is **not** assigned, but programmed in the parts program.

```
(Startposition X0 Y0 Z0)
N10 #skip VirtualMovements(1)#
N20 G01 X100 Y200 F6000
N30 G01 Z1000 (virtual movement, because z is not assigned)
N40 G01 X500
```

Segment N30 is skipped during execution of this program.

4.3.13 Messages from NC program

Command	#MSG (<message level>; <mask>; "<text>")#
<message level>	• ITP The message is issued from the interpreter. This means that the message generally appears well before the execution in the NC kernel. • NCK The message is issued from the NC kernel when the NC block is executed. This means it appears at the same time as the block execution (SAF)
<mask>	STRING
<text>	the text to be displayed

```
N10 G0 X0 Y0
N20 G01 X100 Y0 F6000

N30 #MSG( NCK; STRING; "this is a text")#

N40 G01 X200 Y-100
```

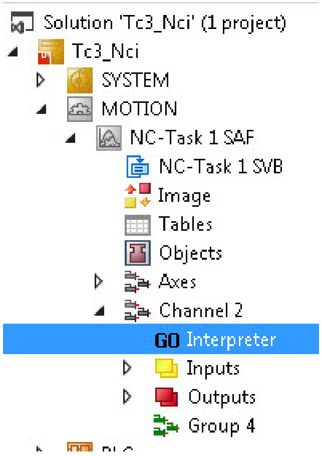
The text can **not** be used to transfer further parameters (e.g. R-parameters).

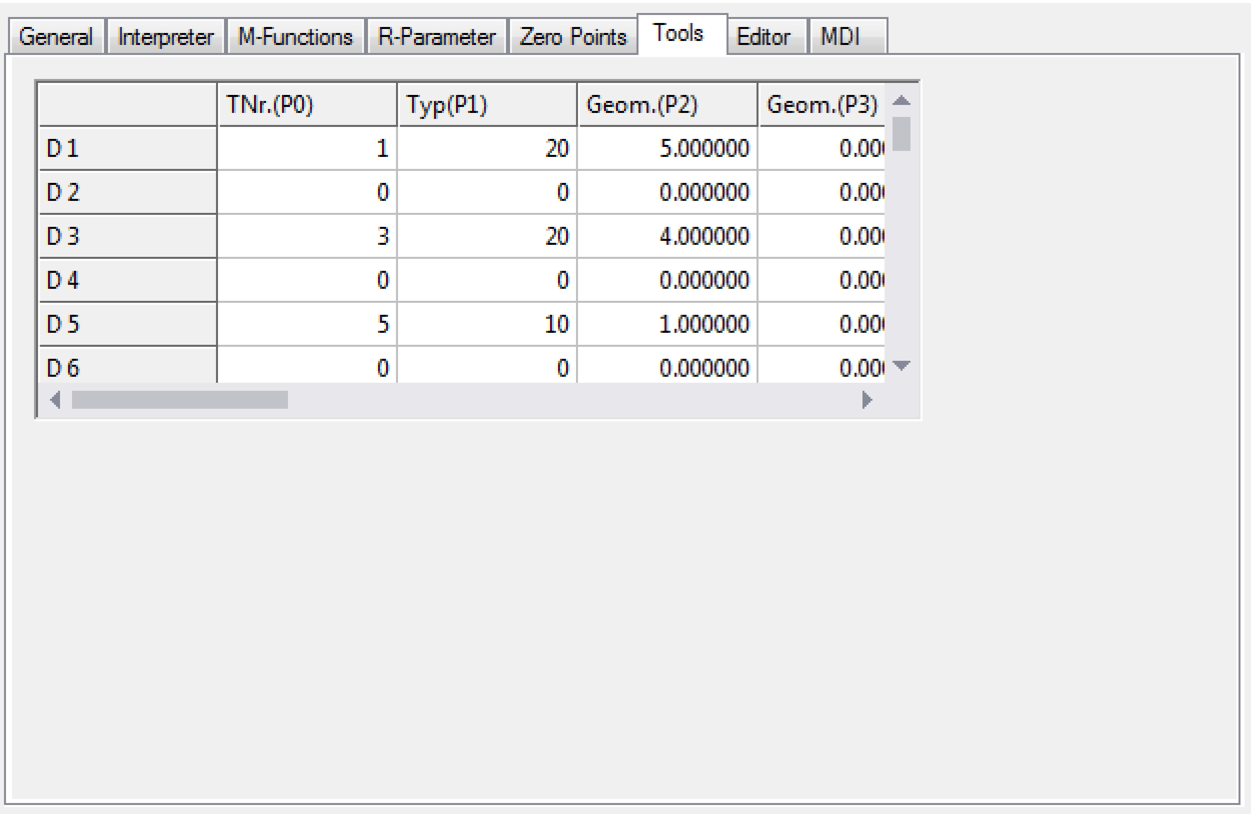
Internally the message is handled like a note.

4.4 Tool Compensation

4.4.1 Tool Data

The NC has 255 memory locations (D1..D255) available for each channel for tool data. The parameters for the tool data can be written directly in the XAE. The data is saved as an ASCII file (<channel ID>.wz) which is kept in the TwinCAT\CNC directory. These files are automatically loaded when TwinCAT is started.





	TNr.(P0)	Typ(P1)	Geom.(P2)	Geom.(P3)
D 1	1	20	5.000000	0.00
D 2	0	0	0.000000	0.00
D 3	3	20	4.000000	0.00
D 4	0	0	0.000000	0.00
D 5	5	10	1.000000	0.00
D 6	0	0	0.000000	0.00

Currently two tool types are supported:

- Drills
- Shaft Cutters

The relevant columns (parameters) for this type of tool are described below.

Drills

Parameter	Meaning
0	**Tool number** When this D-word is called, a tool number that is specified here can be given at the same time.
1	**Tool type** The drill is **type 10**.
2	**Geometry: Length** Describes the length of the drill.
5	**Wear: Length** Describes the wear on the drill. The wear has to be given as a negative value, since it is added to the length.
8	Cartesian tool displacement [▶ 164] in X direction
9	Cartesian tool displacement in Y direction
10	Cartesian tool displacement in Z direction

Shaft Cutters

Parameter	Meaning
0	**Tool number** When this D-word is called, a tool number that is specified here can be given at the same time.
1	**Tool type** The shaft cutter is **type 20**
2	**Geometry: Length** Length of the shaft cutter.
4	**Geometry: Radius**
5	**Wear: Length**
7	**Wear: Radius**
8	Cartesian tool displacement [▶ 164] in X direction
9	Cartesian tool displacement in Y direction
10	Cartesian tool displacement in Z direction

Writing of tool data

Editing tool data with the XAE

As already mentioned, the tool data can be written directly from the XAE. To do this, edit the window shown above.

Parameterization of tool data via the PLC

In addition, tool data can be read and written from the PLC with the function block ItpWriteToolDescEx [▶ 230].

Writing tool data from the parts program

In some applications, it is more convenient to write the tool data directly from the part program.

The tool set to be overwritten must not be active during the write process. This means, for example, if tool radius compensation with parameter set D10 is active, this cannot be overwritten, as long as D10 is still selected.

Command	#set ToolParam(<Zeile>; <Spalte>;<Wert>)#
Parameter <line>	Describes the tool parameter line (1..255) Corresponds to the D number
Parameter <column>	Column to be written (0..15)
Parameter <value>	Parameter value to be transmitted

Sample:

```
N10 G0 X0 Y0 Z0
N20 G01 X100 F60000
N30 R1=10 R2=4 R3=20.3
N40 #set ToolParam(10; 0; 5)# #set ToolParam(10;1;20)#
N50 #set ToolParam(R1; R2; R3)#
N60 G41 X200 Y D10
...
```

NOTE! No formulas may be transmitted as parameters. Writing of the tool data does not require a decoder stop.

Reading tool data from the parts program

This command can be used to assign tool data to an R-parameter.

Command	#get ToolParam(<line>; <column>;<R-Param>)#
Parameter <line>	Writes to the tool parameter line (1..255); this corresponds to the D number
Parameter <column>	Column to be written (0..15)
Parameter <R-Param>	R-parameter in which the date is entered

Sample:

```
N10 G0 X0 Y0 Z0
N20 G01 X100 F60000
N30 R1=10 R2=4
N40 #get ToolParam(10; 0; R5)# #getToolParam(10;1;R20)#
N50 #get ToolParam(R1; R2; R3)#
N60 G41 X200 Y D10
...
```

Notes:

NOTE! No formulas may be transmitted as parameters. Reading of the tool data does not require a decoder stop.

4.4.2 Selecting and Deselecting the Length Compensation

Length compensation can only be selected when G0 [▶ 118] or G1 [▶ 119] are in effect. The working plane [▶ 111] must be selected to which the length compensation is perpendicular.

The feed direction is specified with P (see working plane and feed direction [▶ 111]).

To effect the movement corresponding to the length compensation, the axis concerned must at least be mentioned.

Sample:

```
N10 G17 G01 X0 Y0 Z0 F6000
N20 D1 X10 Y10 Z
N30 ...
N90 M30
```

NOTE! Length correction is automatically selected when cutter radius compensation [▶ 167] is selected.To deselect length correction, D0 has to be programmed. It is again here necessary to at least mention the axis concerned in order to move to the new position.

4.4.3 Cartesian Tool Translation

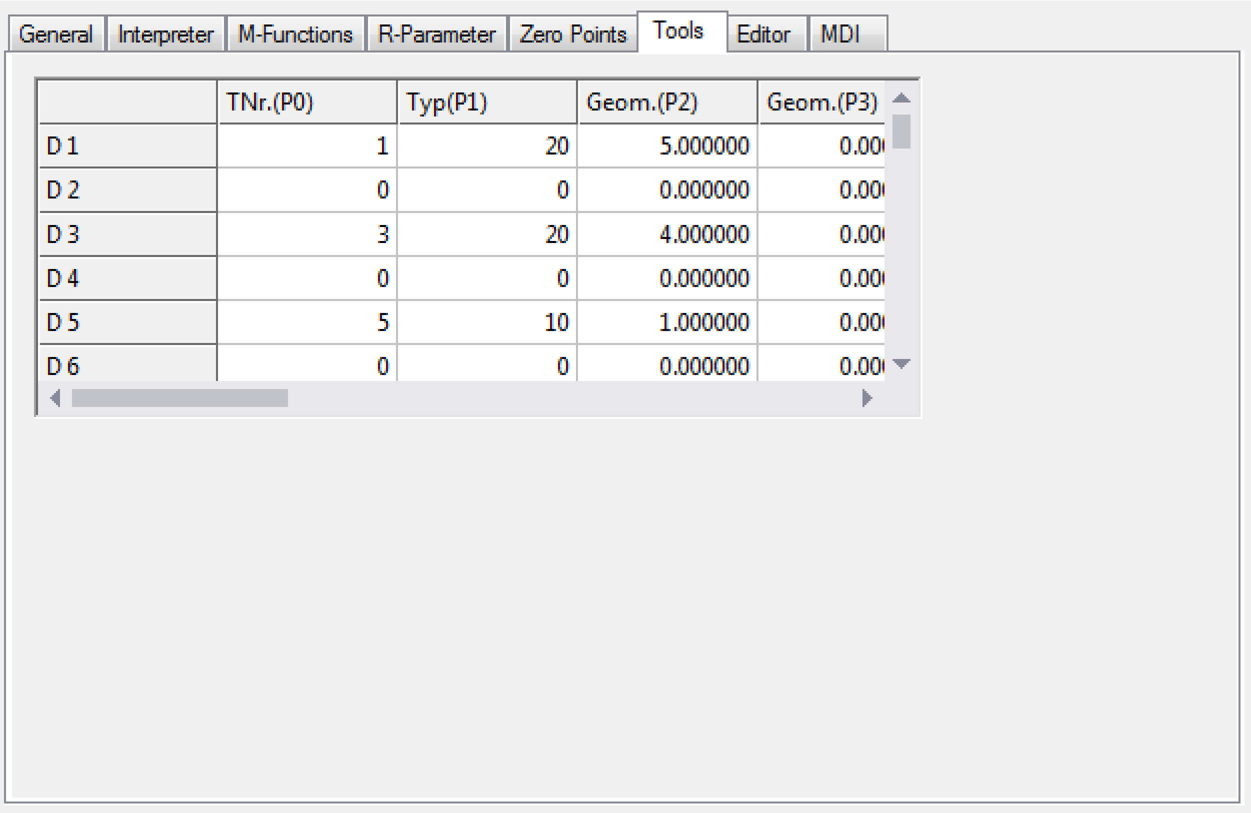
Cartesian tool displacement refers to an offset between the reference point of the tool carrier and the reference point of the tool itself. In many cases, these reference points have the same location, so that a 0 can be entered for the tool displacement.

Parameter

The parameters for a translation are entered into the tool data [▶ 162] in the same way as the tool length etc. Parameters 8 to 10 are available for this purpose. Here

- P8 always describes the X-component
- P9 always describes the Y-component
- P10 always describes the Z-component

independently of the choice of level.



Selecting and deselecting Cartesian tool displacement

As in the case of length compensation, tool displacement is switched on with D<n> (n>0). In order to travel to the translated location, the axes must at least be named. This means that the displacement affects the positioning when the axis is called for the first time. It is also possible for a new final position to be entered for the axis.

The function is switched off with D0. Here again, it is necessary for the axes at least to be named, if the axes are to travel to their new co-ordinates.

Sample 1:

```
N10 G17 G01 X0 Y0 Z0 F6000
N20 D1 X10 Y10 Z (Z-Axis is repositioned)
N30 ...
N90 M30
```

Sample 2:

```
N10 G17 G01 X0 Y0 Z0 F6000
N20 D1 X10 Y10 (Z-Axis is not moved)
N30 ...
N90 M30
```

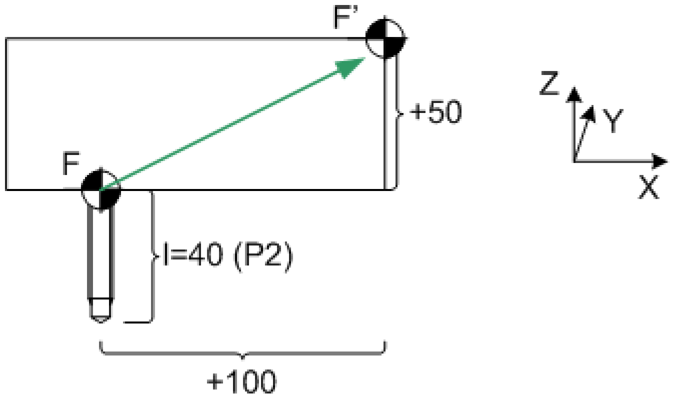
Note

Using tool displacement and rotation

If the Cartesian tool displacement is used in combination with rotation [▶ 129], then the compensation will only be correct if the aggregate (the tool carrier) is also rotated through the same angle.

Application sample

It often happens that a processing machine's tool carrier contains a number of tools. The appropriate tool is pneumatically activated according to the kind of machining required. Since, however, the tools are located at different positions, Cartesian tool displacement is required.



Tool parameters

Parameter	Value
0	0..65535
1	10
2	40
5	0
8	100.0
9	0.0
10	50

Behavior with incremental dimension notation

Default behavior

If a new tool offset (and also length compensation) is selected in incremental dimensions (G91), then the compensation is applied once the axis is named.

Sample 3:

```
(Tooloffset D1: X10 Y20 Z30)
N10 G01 D1 X100 Y0 Z0 F6000
N20 G91 (incremental dimension)
N30 D2 (Tooloffset D2: X100
Y200 Z300)
N30 Z10
N40 ...
```

Command	Description
ToolOffsetIncOn	The tool displacements and length compensations are also applied under G91 once the axis is named
ToolOffsetIncOff	The tool displacement and length compensation are **not** applied under G91

Sample 4:

```
(Tooloffset D1: X10 Y20 Z30)
N05 ToolOffsetIncOff
N10 G01 D1 X100 Y0 Z0 F6000
N20 G91 (incremental dimension)
N30 D2 (Tooloffset D2: X100
Y200 Z300)
N30 Z10
N40 ...
```

In N10 the Tooloffset is applied to all 3 axes. I.e. the axes move in the machine coordinate system (MCS) to X110 Y10 Z30.

In N30 the new Tooloffset of the Z-axis is **not** applied. This results in MCS X110 Y10 **Z40.**

See also ZeroShiftIncOn/Off [▶ 124]

4.4.4 Cutter Radius Compensation

4.4.4.1 Miller/Cutter Radius Compensation Off

Miller/Cutter Radius Compensation Off

Command	G40
Cancellation	G41 [▶ 167] or G42 [▶ 168]

The G40 function switches the miller/cutter radius compensation off. The length radius compensation [▶ 164] will still remain active until it is switched off with D0. Between G40 and end of program it is madatory to program at least one geometry element.

4.4.4.2 Miller/cutter radius compensation left

Miller/cutter radius compensation left

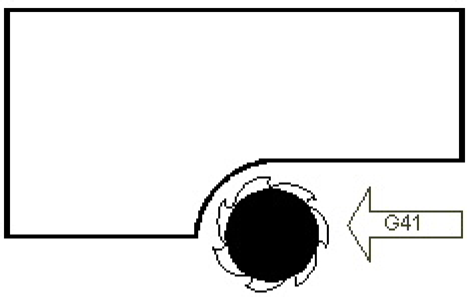
Command	G41
Cancellation	G40 [▶ 167]

The function G41 switches on the miller/cutter radius compensation. The tool is located to the **left** of the workpiece in the direction of movement.

As has already been seen for the length compensation [▶ 164], the cutter radius compensation can only be activated when G0 [▶ 118] or G1 [▶ 119] is in effect. The axes of the plane must be driven when the cutter radius compensation is selected.

Sample:

```
N10 G17 G01 X0 Y0 Z0 F6000
N20 G41 X10 Y20 Z D1
N30 X30
N40 G40 X10 Y10 Z
N50 M30
```



Note

Cutter radius compensation does not apply to full circles

The cutter radius compensation does not support full circles. Full circles have to be split into semicircles, for sample.

Notes:

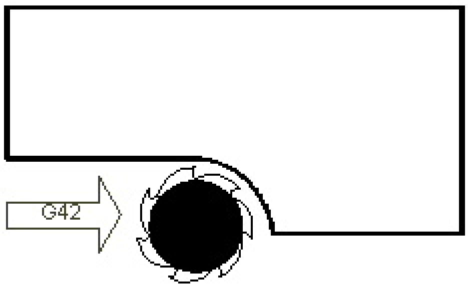
- The cutter radius compensation should be deactivated before the end of the NC program, in order to close it properly. Between G40 and end of program it is mandatory to program at least one geometry element.
- If a decoder stop [▶ 150] is programmed, cutter radius compensation has to be disabled first.
- For arcs, radius compensation can lead to a change in the path velocity at the contour. See also 'Path velocity in arcs [▶ 172]'.
- See Orthogonal contour approach/departure [▶ 172].

4.4.4.3 Miller/cutter radius compensation right

Miller/cutter radius compensation right

Command	G42
Cancellation	G40 [▶ 167]

The function G42 switches on the miller/cutter radius compensation. The tool is located to the **right** of the workpiece in the direction of movement.



Note	**Cutter radius compensation does not apply to full circles** The cutter radius compensation does not support full circles. Full circles have to be split into semicircles, for sample.

NOTE! If a change is to be made from G41 to G42, then a G40 should be programmed between the two movements.

4.4.4.4 Departure and approach behavior of the miller/cutter radius compensation

This chapter describes the approach and departure behavior when the miller/cutter radius compensation is switched on or off. This behavior depends on the start position and cannot be influenced in any other way.

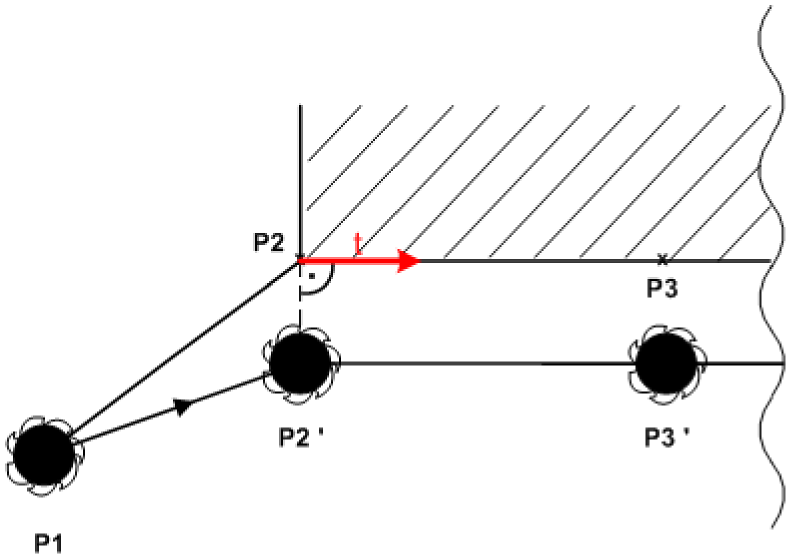
After the radius compensation is switched on, it must be applied. This means the cutter is at one point P1 (without radius compensation) and travels to P2', with the cutter radius being compensated at point P2'.

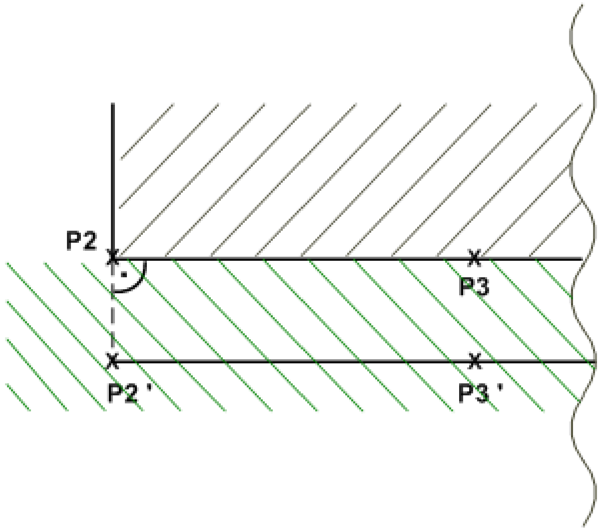
Point P2' depends on the start position P1 within the plain. A distinction is made between 3 basic cases. These cases are exemplified below during application of the radius compensation with a programmed G42 (right compensation).

Similar rules apply for the deactivation of the compensation, except that the tangent t is determined at the end of the path segment, with similar conditions being derived.

Case 1: P1 to the right of the path tangent t

If the starting point P1 is to the right of the path tangent t, P2' is orthogonal to the tangent. This start-up behavior applies to the range hatched in green.

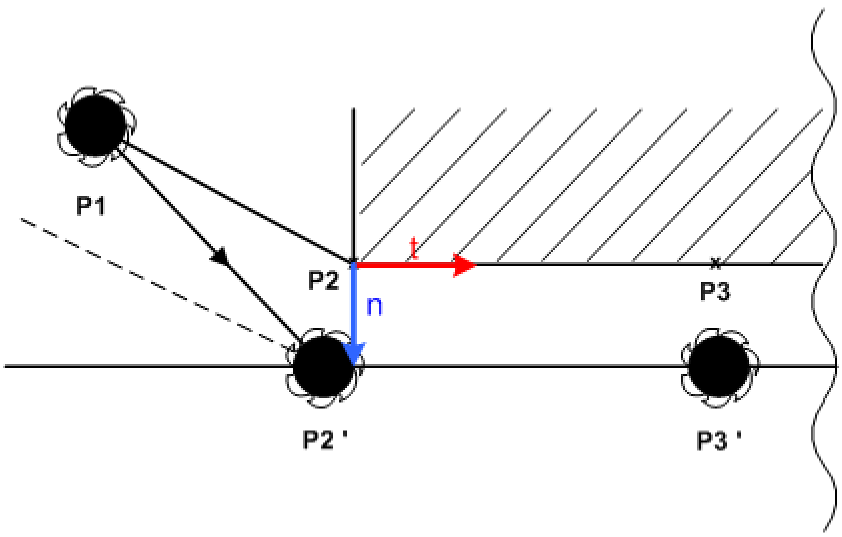


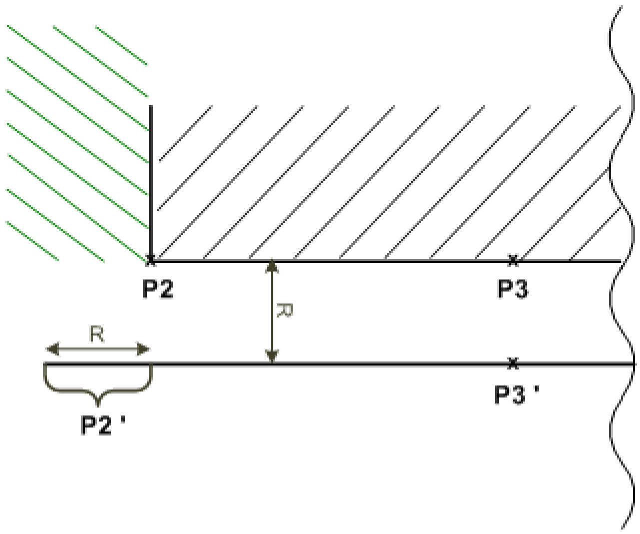


Case 2: P1 the right of the normal n and to the left of the path tangent t

If the start position P1 is to the right of the normal n and to the left of the path tangent t, P2' is moved. P2' results from the intersection of the parallel of P1P2 and the offset distance P2P3. Both straight lines are offset by radius R.

This behavior applies to the range hatched in green.

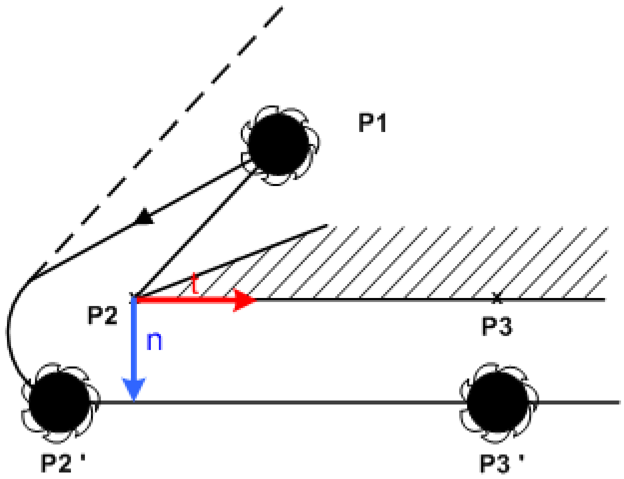


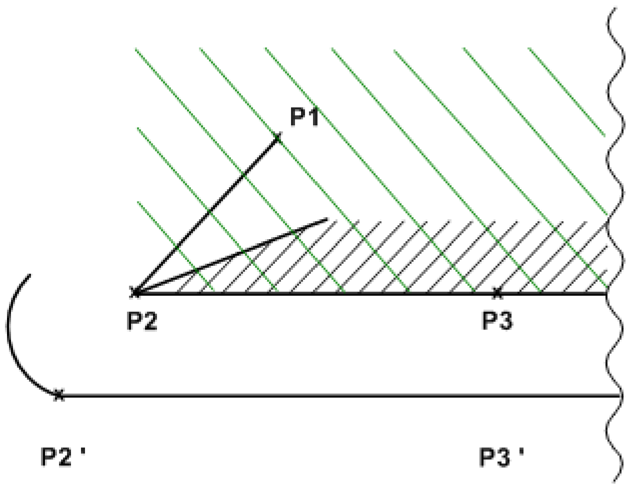


Case 3: P1 to the left of the normal n and to the left of the path tangent t

If the start position P1 is to the left of the normal n and also to the left of the path tangent t, an additional circle segment is inserted during approaching of P2'. In order to avoid free cutting at P2, P2' is not orthogonal to the start tangent of the section P2P3.

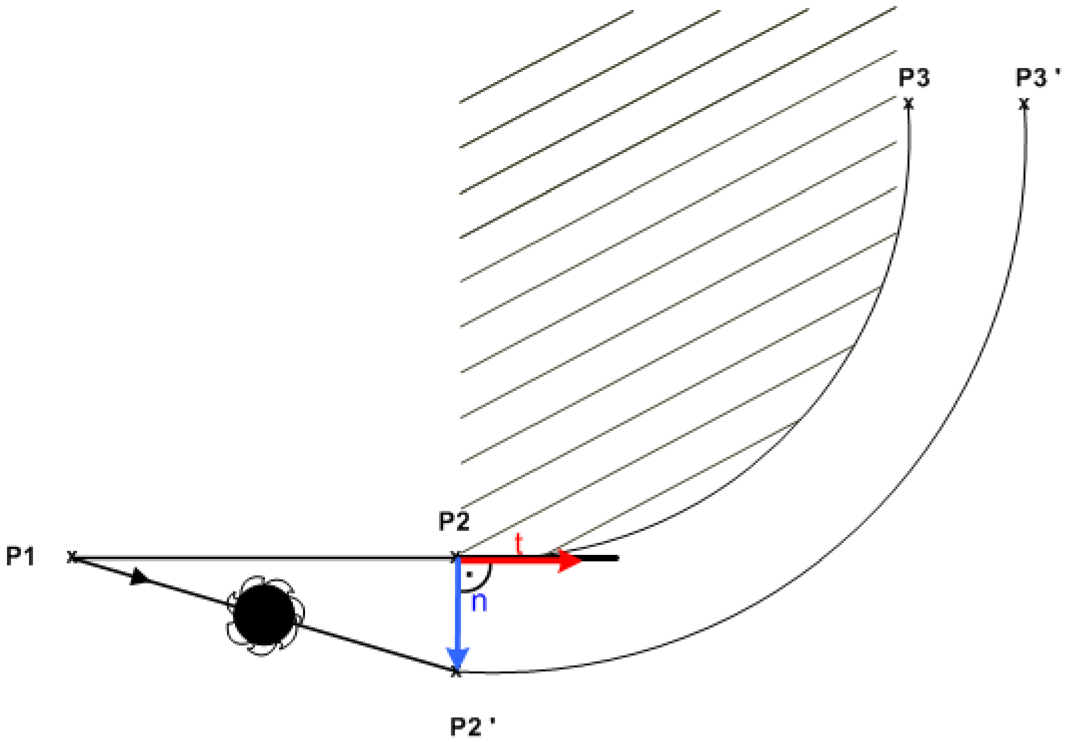
The additional circle segment is inserted for all start positions within the hatched green region.

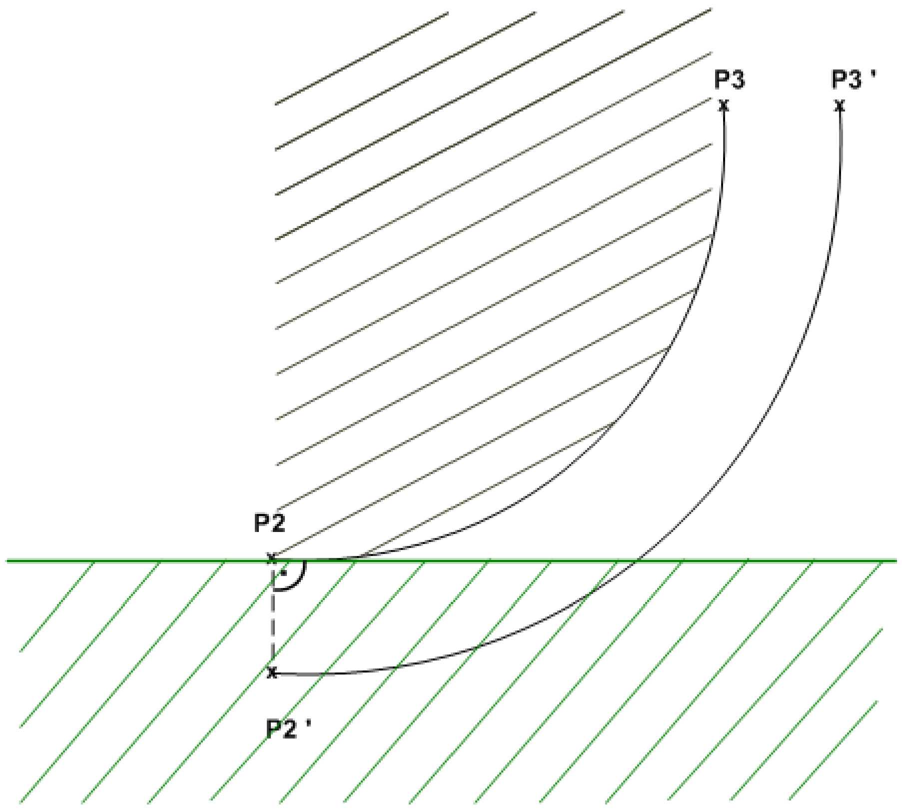




A circle segment follows after the offset

The radius compensation is invariably applied via a straight line. (This must be set in the part program, since otherwise a runtime error will be generated). The contour can then start with a circle. The rules for starting and stopping are the same as before, i.e. here too the path tangent of the contour for P2 is determined, and a distinction is made between the three cases described.





If P2' is always to be approached orthogonal to the path tangent of P2, independent of the starting point, this can be realized with an additional command (see Orthogonal contour approach/departure [▸ 172]).

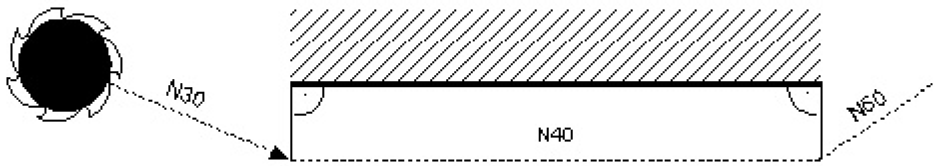
4.4.5 Orthogonal Contour Approach/Departure

Command	NORM
Cancellation	End of block
Programmable with	G40 [▸ 167] G41 [▸ 167] G42 [▸ 167]

The 'NORM' command has the effect that the contour is approached orthogonally when cutter radius compensation is switched on. The actual position of the cutter is irrelevant. When de-selecting, the last segment with active compensation is also left orthogonally.

Sample:

```
N10 G17
N20 G01 X0 Y0 Z0 F6000
N30 G42 NORM X100 Y0 D5
N40 X200
N50 G40 NORM X220 Y0
N60 M30
```



🛈 **NOTE! The Norm command has hitherto only been implemented for straight line/straight line transitions.**

4.4.6 Path Velocity in Arcs

When the cutter radius compensation [▸ 167] is active, the programmed radius changes for arcs. This in turn alters the velocity. The following commands are used to specify whether the feed value refers to the contour or the tool center point.

Constant Feed at the Contour

Command	CFC
Cancellation	CFIN or CFTCP

With CFC (constant feed contour) the feedrate at the contour is held constant.

Constant Feed at the Internal Radius

Command	CIN
Cancellation	CFC or CFTCP

With CFIN (constant feed internal radius) the feedrate at internal radii is reduced. This results in a constant velocity at the contour. The velocity at external radii is not increased.

Constant Feed of the Tool Centre Point

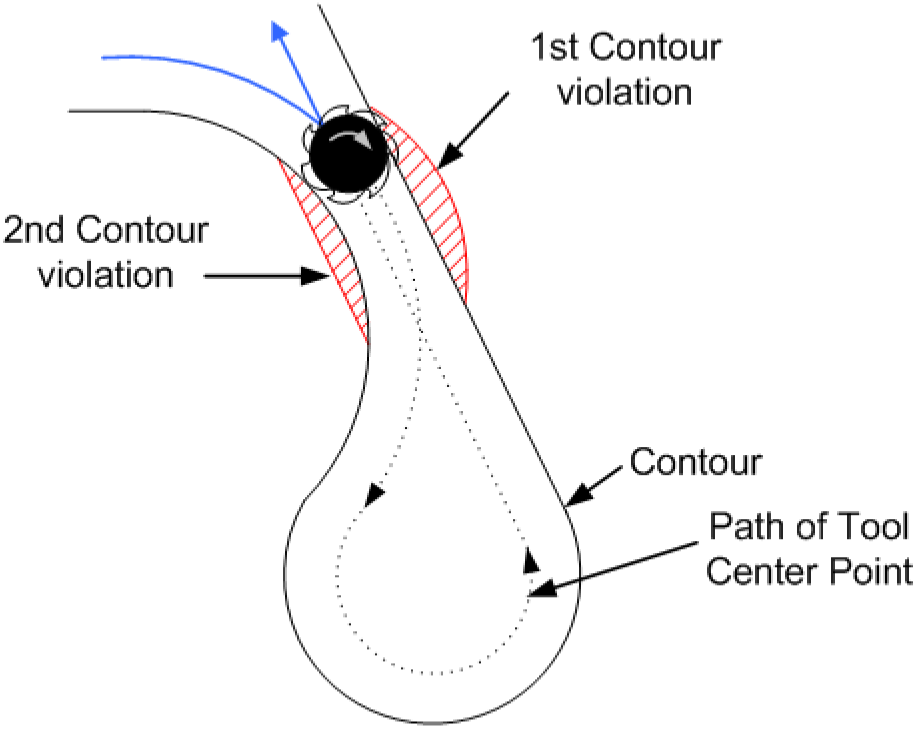
Command	CFTCP
Cancellation	CFC or CFIN

With CFTCP (constant feed tool center point) the feedrate of the tool's center point is kept constant. This means that at internal radii the velocity at the contour is increased, and that it is correspondingly reduced at external radii.

4.4.7 Bottle Neck Detection

If the cutter radius is not considered when a part program is created, the cutter may inadvertently process the opposite side of the workpiece. This leads to a contour collision with the workpiece, or, in other words, a bottleneck was programmed.

Command	CDON
Cancellation	CDOF



In this form, this behavior can only occur in combination with cutter radius compensation (G41/G42). In order to prevent such contour collisions, monitoring can be switched on from the part program via **CDON**. For it to be active, cutter radius compensation must also be selected.

The response of the NCI when a bottleneck is detected can be parameterized via the PLC. 3 cases are distinguished:

- Error and abort
  If a bottleneck is detected, TwinCAT generates a runtime error and aborts the program execution.
- Notification and modification of the contour
  If a bottleneck is detected, the contour is modified such that a contour collision is avoided (see Figure 1: blue line). However, this also means that segments may be left out, depending on the program. Furthermore, a note is entered in the application viewer to say that a bottleneck was detected.
- Notification and contour collision
  If a bottleneck is detected, the contour is not changed and no error is generated. Only a message is entered in the application viewer.

Significant computing resources are required for contour collision monitoring. It should therefore only be selected if it is actually required. Furthermore, the amount of look-ahead for the bottleneck detection should be specified. This requires the number of future segments to be determined that are monitored relative to the n-th segment, in order to check for bottlenecks. The selected number of segments should not be too large, since this would put unnecessary strain on the system. The value for the look-ahead is also parameterized from the PLC.

Function blocks for parameterizing the bottleneck detection:

- ItpSetBottleNeckModeEx [▶ 219]
- ItpGetBottleNeckModeEx [▶ 195]
- ItpSetBottleNeckLookAheadEx [▶ 218]

- ItpGetBottleNeckLookAheadEx [▶ 195]

Sample:

```
N10 G0 X0 Y0 Z0
N20 CDON
N30 G01 G41 D3 X100 F6000 (cutter radius 30mm)
...
N40 G01 X200
N50 G02 X220 Y-74.641 I0 J-40
N60 G01 X300 Y-104
N70 G01 X230 Y120
N80 G40 D0 Y200
N90 CDOF
...
M30
```

4.5 Command overview

4.5.1 General command overview

Command	Description	block-by-block / modal	Default
ANG [▸ 128]	Contour line programming (angle)	s	
AROT [▸ 129]	Rotation additive	m	
CalcInvRot [▸ 129]	Calculates the inverse rotation of a vector	s	
CalcRot [▸ 129]	Calculates the rotation of a vector	s	
CDOF [▸ 173]	Bottleneck detection off	m	Default
CDON [▸ 173]	Bottleneck detection on	m	
CFC [▸ 172]	Constant velocity at the contour	m	Default
CFIN [▸ 172]	Constant velocity in the interior circle	m	
CFTCP [▸ 172]	Constant velocity of tool center point	m	
CIP [▸ 119]	Circular interpolation	s	
CPCOF [▸ 119]	Centre point correction off	m	
CPCON [▸ 119]	Centre point correction on	m	Default
DelDTG [▸ 140]	Delete Distance to Go	s	
DYNOVR [▸ 156]	Dynamic Override	m	
FCONST [▸ 123]	Constant feed programming	m	Default
FLIN [▸ 123]	Linear feed programming	m	
G00 [▸ 118]	Rapid traverse	m	
G01 [▸ 119]	Straight line interpolation	m	Default
G02 [▸ 119]	Clockwise circular interpolation	m	
G03 [▸ 119]	Anticlockwise circular interpolation	m	
G04 [▸ 122]	Dwell time	s	
G09 [▸ 123]	Accurate stop	s	
G17 [▸ 111]	Plane selection XY	m	Default
G18 [▸ 111]	Plane selection ZX	m	
G19 [▸ 111]	Plane selection YZ	m	
G40 [▸ 167]	No miller/cutter radius compensation	m	Default
G41 [▸ 167]	Miller/cutter radius compensation left	m	
G42 [▸ 167]	Miller/cutter radius compensation right	m	
G53 [▸ 124]	Zero shift suppression	m	Default
G54 [▸ 124]	1st adjustable zero shift	m	
G55 [▸ 124]	2nd adjustable zero shift	m	
G56 [▸ 124]	3rd adjustable zero shift	m	
G57 [▸ 124]	4th adjustable zero shift	m	
G58 [▸ 124]	1st programmable zero shift	m	

Command	Description	block-by-block / modal	Default
G59 [▸ 124]	2nd programmable zero shift	m	
G60 [▸ 123]	Accurate stop	m	
G70 [▸ 113]	Dimensions inch	m	
G71 [▸ 113]	Dimensions metric	m	Default
G74 [▸ 117]	Programmed traverse to reference point	s	
G90 [▸ 111]	Reference dimension notation	m	Default
G91 [▸ 111]	Incremental dimension notation	m	
G700 [▸ 113]	Dimensions in inches with calculation of the feed	m	
G710 [▸ 113]	Dimensions metric with calculation of the feed	m	
Mirror [▸ 132]	Mirroring coordinate system	m	
MOD [▸ 140]	Modulo movement	s	
MSG [▸ 161]	Messages from the NC program	s	
NORM [▸ 172]	orthogonal approach of and departure from the contour	s	
P+ [▸ 111]	Feed direction positive	m	Default
P- [▸ 111]	Feed direction negative	m	
paramAutoAccurateStop [▸ 139]	Automatic Accurate Stop	m	
paramAxisDynamics [▸ 156]	Parameterization of the axis dynamics	m	
paramC1ReductionFactor [▸ 158]	C1 reduction factor	m	
paramC2ReductionFactor [▸ 158]	C2 reduction factor	m	
paramCircularSmoothing [▸ 138]	Circular smoothing	m	
paramDevAngle [▸ 158]	C0 reduction - deflection angle	m	
paramGroupVertex [▸ 138]	Circular smoothing (old)	m	
paramGroupDynamic [▸ 156]	Pathway dynamics (old)	m	
paramPathDynamics [▸ 156]	Pathway dynamics	m	
paramRadiusPrec [▸ 120]	Circular accuracy	m	
paramSplineSmoothing [▸ 136]	Smoothing with Bezier Splines	m	
paramVertexSmoothing [▸ 133]	Smoothing of Segment Transitions	m	
paramVeloJump [▸ 158]	C0 reduction - max. step change in velocity	m	

Command	Description	block-by-block / modal	Default
paramVeloMin [▶ 159]	Minimum velocity	m	
paramZeroShift [▶ 124]	Parameterization of the configurable zero shift	m	
PathAxesPos [▶ 160]	Reads the actual position	s	
ROT [▶ 129]	Absolute rotation	m	
RotExOff [▶ 129]	Extended rotation function off	m	Default
RotExOn [▶ 129]	Extended rotation function on	m	
RotVec [▶ 129]	Calculation routine for rotating a vector	s	
RParam [▶ 115]	Initialization of R-parameters	s	
RToDwordGetBit [▶ 115]	Converts an R-parameter to DWord and checks whether a defined bit is set	m	
SEG [▶ 128]	Contour line programming (segment length)	s	
skip VirtualMovements [▶ 161]	Skip virtual movements	m	
ToolOffsetIncOff [▶ 164]	Cartesian tool displacement and length compensation is not applied under G91	m	
ToolOffsetIncOn [▶ 164]	Cartesian tool displacement and length compensation is applied under G91	m	Default
ToolParam [▶ 162]	Writing and reading of tool parameters	m	
TPM [▶ 126]	Target position monitoring	s	
ZeroShiftIncOff [▶ 124]	Zero shift is not applied under G91	m	
ZeroShiftIncOn [▶ 124]	Zero shift is applied under G91	m	Default

Address	Description
Q<n> [▶ 141]	Axis label for auxiliary axis (1 <= n <= 5)

4.5.2 @-Command Overview

Several variations of these commands are often possible, since K for a constant, R for an R-parameter or P for an R-parameter used as a pointer can be used for parameters. For example, the notation K/R/Pn should be understood to mean "either a number or an R-parameter or a pointer".

The following @-commands are available:

Command	Versions	Function
@40 [▶ 115]	@40 Kn Rn Rm	Save register on the stack
@41 [▶ 115]	@41 Rn Rm	Save register on the stack
@42 [▶ 115]	@42 Kn .... Rm Rn	Restore register from stack
@43 [▶ 115]	@43 Rm Rn	Restore register from stack
@100 [▶ 151]	@100 K±n @100 Rm	Unconditional jump
@111 [▶ 151]	@111 Rn K/Rn Km ...	Case block
@121 [▶ 151]	@121 Rn K/Rn Kn	Jump if unequal
@122 [▶ 151]	@122 Rn K/Rn Kn	Jump if equal
@123 [▶ 151]	@123 Rn K/Rn Kn	Jump if less or equal
@124 [▶ 151]	@124 Rn K/Rn Kn	Jump if less
@125 [▶ 151]	@125 Rn K/Rn Kn	Jump if greater or equal
@126 [▶ 151]	@126 Rn K/Rn Kn	Jump if greater
@131 [▶ 153]	@131 Rn K/Rn Kn	Loop while equal
@132 [▶ 153]	@132 Rn K/Rn Kn	Loop while unequal
@133 [▶ 153]	@133 Rn K/Rn Kn	Loop while greater
@134 [▶ 153]	@134 Rn K/Rn Kn	Loop while greater or equal
@135 [▶ 153]	@135 Rn K/Rn Kn	Loop while less
@136 [▶ 153]	@136 Rn K/Rn Kn	Loop while less or equal
@141 [▶ 153]	@141 Rn K/Rn Kn	Repeat until equal
@142 [▶ 153]	@142 Rn K/Rn Kn	Repeat until unequal
@143 [▶ 153]	@143 Rn K/Rn Kn	Repeat until greater
@144 [▶ 153]	@144 Rn K/Rn Kn	Repeat until greater or equal
@145 [▶ 153]	@145 Rn K/Rn Kn	Repeat until less
@146 [▶ 153]	@146 Rn K/Rn Kn	Repeat until less or equal
@151 [▶ 153]	@151 Rn K/Rn Kn	FOR_TO loop
@161 [▶ 153]	@161 Rn K/Rn Kn	FOR_DOWNTO loop
@200	@200 Rn	Delete a variable
@202	@202 Rn Rm	Swap two variables
@302	@302 K/R/Pn K/R/Pn R/Pn	Read machine data bit
@361 [▶ 160]	@361 Rn Km	Read machine-related actual axis value
@372	@372 Rn	Set number of the NC channel in variable
@402 [▶ 119]	@402 K/R/Pn K/R/Pn K/R/Pn	Write machine data bit
@610	@610 Rn Rn	Find absolute value of a variable
@613	@613 Rn Rn	Find square root of a variable
@614	@614 Rn Rm Rm	Find square root of the sum of the squares of two variables x = sqrt(a^2 + b^2)
@620 [▶ 153]	@620 Rn	Increment variable
@621	@621 Rn	Decrement variable
@622	@622 Rn	Find integer part of a variable
@630 [▶ 115]	@630 Rn Rm	Find sine of a variable
@631 [▶ 115]	@631 Rn Rm	Find cosine of a variable
@632 [▶ 115]	@632 Rn Rm	Find tangent of a variable

Command	Versions	Function
@633 [▸ 115]	@633 Rn Rm	Find cotangent of a variable
@634 [▸ 115]	@634 Rn Rm	Find arcsine of a variable
@635 [▸ 115]	@635 Rn Rm	Find arccosine of a variable
@636 [▸ 115]	@636 Rn Rm	Find arctangent of a variable
@714 [▸ 150]	@714	Decoder stop
@716 [▸ 150]	@716	Decoder stop with rescan of the axis positions
@717 [▸ 150]	@717	Decoder stop with external trigger event

Machine data

Access to the following machine data is supported:

Byte	Bit	Action
5003 [▸ 119]	5	0: IJK words specify the distance between the center of the circle and the starting point. IJK are absolute data giving the center of the circle.

5 PLC NCI Libraries

Requirements

Overview of PLC NCI libraries	Description
PLC Library: Tc2_NCI [▶ 181]	Function blocks for the configuration of the interpolation group (e.g. formation of the 3D group) and for operating the interpreter (G-Code (DIN 66025)) such as loading and starting the NC program.
PLC Library: Tc2_PlcInterpolation [▶ 276]	Programming of multi-dimensional movements from the PLC (alternative to using G-Code (DIN 66025))

5.1 PLC Library: Tc2_NCI

5.1.1 Configuration

The library Tc2_NCI provides function blocks for general NC axis configuration. This makes it possible to configure or to reconfigure axes in a simple way directly from the PLC.

Function Block	Description
CfgBuild3DGroup [▶ 181]	Groups up to 3 PTP axes into a 3D group
CfgBuildExt3DGroup [▶ 182]	Groups up to 3 PTP axes and 5 auxiliary axes into a 3D group
CfgAddAxisToGroup [▶ 183]	Configures a single axis at a particular location within a group (PTP, 3D, FIFO)
CfgReconfigGroup [▶ 184]	Removes 3D (or FIFO) axis allocations and returns of the axes to their personal PTP group
CfgReconfigAxis [▶ 185]	Returns a single axis from, for example, a 3D group, to its personal PTP group
CfgRead3DAxisIds [▶ 186]	Reads the axis IDs (axis allocation) of a 3D group
CfgReadExt3DAxisIds [▶ 187]	Reads the axis IDs (axis allocation) of a 3D group with auxiliary axes

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.1.1 CfgBuild3DGroup

This function block configures a 3D group with up to 3 PTP axes (X, Y and Z).

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nGroupId        : UDINT;
nXAxisId        : UDINT;
```

```
nYAxisId        : UDINT;
nZAxisId        : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nGroupId: ID of the 3D group

nXAxisId: ID of the PTP axes

nYAxisId: ID of the PTP axes

nZAxisId: ID of the PTP axes

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

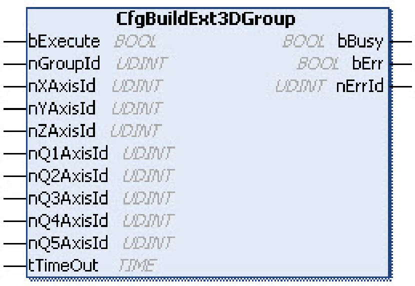
bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.1.2 CfgBuildExt3DGroup



This function block configures a 3D group with up to 3 path axes (X, Y and Z). Additionally, up to 5 auxiliary axes (Q1..Q5) can be configured.

The axis IDs of the PTP axes that are to be included in the interpolation group are applied at the inputs **nXAxisId** to **nQ5AxisId**.

NOTE! The assignment of the auxiliary axes must start with nQ1AxisId. No gaps between auxiliary axes are permitted. For example, if nQ3AxisId is to be assigned, nQ2AxisId must also be assigned a valid Axis ID.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nGroupId        : UDINT;
nXAxisId        : UDINT;
nYAxisId        : UDINT;
nZAxisId        : UDINT;
nQ1AxisId       : UDINT;
nQ2AxisId       : UDINT;
nQ3AxisId       : UDINT;
nQ4AxisId       : UDINT;
nQ5AxisId       : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is executed when a positive edge is encountered.

nGroupId: ID of the 3D group

nXAxisId: Axis IDs of the PTP axes to be included in the interpolation group

nYAxisId: Axis IDs of the PTP axes to be included in the interpolation group

nZAxisId: Axis IDs of the PTP axes to be included in the interpolation group

nQ1AxisId: Axis IDs of the PTP axes to be included in the interpolation group

nQ2AxisId: Axis IDs of the PTP axes to be included in the interpolation group

nQ3AxisId: Axis IDs of the PTP axes to be included in the interpolation group

nQ4AxisId: Axis IDs of the PTP axes to be included in the interpolation group

nQ5AxisId: Axis IDs of the PTP axes to be included in the interpolation group

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.1.3 CfgAddAxisToGroup

The CfgAddAxisToGroup function block configures a single axis at a particular location within an existing group (PTP, 3D, FIFO).

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nGroupId        : UDINT;
nAxisId         : UDINT;
nIndex          : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nGroupId: ID of the target group

nAxisId: ID of the axis to be configured

nIndex: Position of the axis within the group; can have values between 0 and n-1. Depending on the type of group, n has the following significance: PTP: n = 1, 3D: n = 3, FIFO: n = 8

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.1.4 CfgReconfigGroup

The CfgReconfigGroup function block removes the axis allocation of an existing group (NCI or FIFO), returning the axes to their personal PTP groups.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nGroupId        : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nGroupId: ID of the group to be resolved

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.1.5 CfgReconfigAxis

The CfgReconfigAxis function block returns a single axis from, for example, a 3D group, to its personal PTP group.

Interface

```
VAR_INPUT
bExecute        : BOOL;
nAxisId         : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nAxisId: ID of the axis to be returned

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.1.6 CfgRead3DAxisIds

The function block CfgRead3DAxisIds reads the axis configuration of a 3D group.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nGroupId        : UDINT;
pAddr           : PVOID;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nGroupId: ID of the 3D group

pAddr: Address of the variable into which the function block writes the axis IDs of the group assignment (array with three elements of type UDINT)

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Sample:

```
VAR
(* instance *)
ReadAxIds : CfgRead3DAxisIds;
AxIds : ARRAY[1..3] OF UDINT;
END_VAR
```

```
ReadAxIds( bExecute := TRUE,
nGroupId := 4,
pAddr := ADR( AxIds ),
tTimeOut := T#1s );
```

AxIds now contains the three axis IDs for the 3D group with the group ID 4.

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.1.7 CfgReadExt3DAxisIds

The function block CfgReadExt3DAxisIds reads the axis configuration of the extended 3D group.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nGroupId        : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nGroupId: ID of the 3D group

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
stExt3dGroup    : NCI_EXT3DGROUP;
END_VAR
```

stExt3dGroup: Instance of the structure NCI_EXT3DGROUP (enter axis IDs of the current interpolation group here)

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

```
TYPE NCI_EXT3DGROUP :
STRUCT
nXAxisId        : UDINT;
nYAxisId        : UDINT;
nZAxisId        : UDINT;
nQ1AxisId       : UDINT;
nQ2AxisId       : UDINT;
nQ3AxisId       : UDINT;
nQ4AxisId       : UDINT;
```

```
nQ5AxisId       : UDINT;
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2 NCI POUs

The TwinCAT library Tc2_NCI contains function blocks for operating the NC interpreter from the PLC.

The following function blocks are included in the library Tc2_NCI.

Function Block	Description
ItpConfirmHsk [▶ 190]	Acknowledges an M-function of type handshake
ItpDelDtgEx [▶ 191]	Triggers “Delete Distance to go” in the NC
ItpEnableDefaultGCode [▶ 192]	Executes a user-defined standard G-Code before the start of each NC program
ItpEStopEx [▶ 193]	Triggers the NCI EStop
ItpGetBlockNumber [▶ 194]	Provides the block number of the NC program for the cyclic interface
ItpGetBottleNeckLookAheadEx [▶ 195]	Provides the value of the look-ahead for bottleneck detection
ItpGetBottleNeckModeEx [▶ 195]	Provides the response mode for bottleneck detection
ItpGetChannelId [▶ 196]	Provides the channel ID
ItpGetChannelType [▶ 197]	Provides the channel type of the cyclic interface
ItpGetCyclicLRealOffsets [▶ 198]	Provides the index offset of the LREAL variables used in the cyclic channel interface
ItpGetCyclicUdintOffsets [▶ 199]	Provides the index offset of the UDINT variables used in the cyclic channel interface
ItpGetError [▶ 200]	Provides the error number
ItpGetGeoInfoAndHParamEx [▶ 200]	Reads information of the currently active segment and past and future segments.
ItpGetGroupAxisIds [▶ 202]	Provides the axis IDs that were configured for the group
ItpGetGroupId [▶ 202]	Provides the group ID
ItpGetHParam [▶ 203]	Provides the current H-parameter from the NC
ItpGetHskMFunc [▶ 203]	Provides the current M-function number of type handshake
ItpGetItfVersion [▶ 204]	Provides the current version of the cyclic interface
ItpGetOverridePercent [▶ 204]	Provides the channel override in percent
ItpGetSetPathVelocity [▶ 205]	Returns the current set path velocity
ItpGetSParam [▶ 205]	Provides the current S-parameter from the NC
ItpGetStateInterpreter [▶ 206]	Provides the current interpreter status
ItpGetTParam [▶ 206]	Provides the current T-parameter from the NC
ItpGoAheadEx [▶ 207]	Triggers the GoAhead function (decoder stop with external trigger event)
ItpHasError [▶ 208]	Determines whether an error is present
ItpIsFastMFunc [▶ 208]	Determines whether the M-function number provided is present in the form of a fast M-function
ItpIsEStopEx [▶ 209]	Determines whether an EStop is executed or pending
ItpIsHskMFunc [▶ 210]	Determines whether an M-function of type handshake is present
ItpLoadProgEx [▶ 210]	Loads an NC program using program names
ItpReadCyclicLRealParam1 [▶ 212]	Reads the first LReal parameter from the cyclic channel interface
ItpReadCyclicUdintParam1 [▶ 212]	Reads the first UDINT parameter from the cyclic channel interface
ItpReadRParamsEx [▶ 213]	Reads calculation parameters
ItpReadToolDescEx [▶ 214]	Reads the tool description from the NC
ItpReadZeroShiftEx [▶ 215]	Reads the zero shift from the NC
ItpResetEx2 [▶ 216]	Carries out a reset of the interpreter or of the NC channel

Function Block	Description
ItpResetFastMFuncEx [▶ 217]	Resets a fast signal bit
ItpSetBottleNeckLookAheadEx [▶ 218]	Sets the value of the look-ahead for bottleneck detection
ItpSetBottleNeckModeEx [▶ 219]	Sets the response mode when bottleneck detection is switched on
ItpSetCyclicLRealOffsets [▶ 220]	Sets the index offsets of the LREAL variables used in the cyclic channel interface
ItpSetCyclicUdintOffsets [▶ 221]	Sets the index offsets of the UDINT variables used in the cyclic channel interface
ItpSetOverridePercent [▶ 222]	Sets the channel override in percent
ItpSetSubroutinePathEx [▶ 223]	Optionally sets the search path for subroutines
ItpSetToolDescNullEx [▶ 224]	Sets all tool parameters (including number and type) to zero
ItpSetZeroShiftNullEx [▶ 225]	Set all zero shifts to null
ItpSingleBlock [▶ 226]	Enables or disables the single block mode in the NCI.
ItpStartStopEx [▶ 227]	Starts or stops the interpreter (NC channel)
ItpStepOnAfterEStopEx [▶ 228]	Enables further processing of the parts program after an NCI EStop
ItpWriteRParamsEx [▶ 229]	Writes calculation parameters
ItpWriteToolDescEx [▶ 230]	Writes the tool description into the NC
ItpWriteZeroShiftEx [▶ 231]	Writes the zero shift into the NC
Block search (for description of the functionality see Blocksearch [▶ 232])	
ItpBlocksearch [▶ 233]	Sets the interpreter to a user-defined location, so that NC program execution continues from this point.
ItpGetBlocksearchData [▶ 236]	Reads the current state after an NC program interruption.
ItpStepOnAfterBlocksearch [▶ 237]	Starts the motion after a block search.
Retrace	
ItpEnableFeederBackup [▶ 237]	Enables the backup list for retracing
ItpIsFeederBackupEnabled [▶ 238]	Reads whether the backup list for retracing is active
ItpIsFirstSegmentReached [▶ 240]	Reads whether the start position is reached during retracing
ItpIsFeedFromBackupList [▶ 239]	Reads whether feeder entries were sent from the backup list
ItpIsMovingBackwards [▶ 240]	Reads whether backward movement occurs on the current path
ItpRetraceMoveBackward [▶ 241]	Performs a backward movement on the path
ItpRetraceMoveForward [▶ 242]	Performs a forward movement on the path. This is called to cancel retracing.

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.1 ItpConfirmHsk

The ItpConfirmHsk function block confirms the currently present M-function.

If the channel override is set to 0 or an E-stop is active, no M-functions are acknowledged during this time. The busy signal of ItpConfirmHsk therefore remains active and must continue to be called.

VAR_INPUT

```
VAR_INPUT
bExecute  : BOOL;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc : NCTOPLC_NCICHANNEL_REF;
sPlcToNci : PLCTONC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

sPlcToNci: The structure of the cyclic channel interface from the PLC to the NCI. (Type: PLCTONC_NCICHANNEL_REF [▶ 308])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.2 ItpDelDtgEx

The function block ItpDelDtgEx triggers residual distance deletion. There is a more detailed description in the Interpreter [▶ 140]documentation.

VAR_INPUT

```
VAR_INPUT
bExecute  : BOOL;
tTimeOut  : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.3 ItpEnableDefaultGCode

The function block ItpEnableDefaultGCode enables execution of a user-defined G-Code before the start of each NC program from the PLC. The default program is executed before the loaded program when the actual NC program starts.

This function block enables rotation of the coordinate system for all NC programs to be executed, for example.

The standard G-Code must be stored as “DefaultGCode<Channel-Number>.def” in the TwinCAT\Mc\Nci directory.

VAR_IN

```
VAR_IN
bExecute             : BOOL;
bUseDefaultGCode     : BOOL;
tTimeOut             : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

bUseDefaultGCode: If this variable is TRUE, the default G-Code is activated through a rising edge at bExecute. If the variable is FALSE, the default G-Code is deactivated.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.4 ItpEStopEx

The function block ItpEStopEx triggers the NCI EStop and enables a controlled stop on the path. The limit values for the deceleration and the jerk are transferred as parameters. If these should be smaller than the currently active dynamic parameters, the transferring parameters are rejected.

VAR_INPUT

```
VAR_INPUT
bExecute       : BOOL;
fDec           : LREAL;
fJerk          : LREAL;
tTimeOut       : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

fDec: Max. deceleration during stopping. If fDec is smaller than the currently active deceleration, fDec is rejected. This ensures that the deceleration occurs with the standard ramp as a minimum.

fJerk: Max. jerk during stopping. If fJerk is smaller than the currently active jerk, fJerk is rejected.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

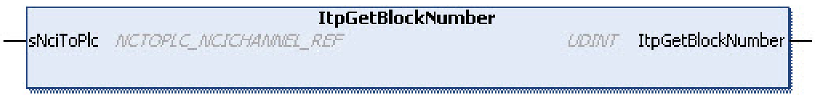
nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

See also ItpStepOnAfterEStopEx [▶ 228].

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.5 ItpGetBlockNumber



ItpGetBlockNumber is a function that returns the block number of the NC program for the cyclic interface.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

ItpGetBlockNumber: Block number of the active geometry segment

Sample

```
VAR
nBlockNumber        : UDINT;
sNciToPlc AT%I*     : NCTOPLC_NCICHANNEL_REF;
END_VAR
nBlockNumber := ItpGetBlockNumber(sNciToPlc);
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.6 ItpGetBottleNeckLookAheadEx

The function block ItpGetBottleNeckLookAheadEx determines the maximum size of the look-ahead for the bottleneck detection (contour collision monitoring).

There is a more detailed description in the Interpreter [▶ 173] documentation.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy           : BOOL;
bErr            : BOOL;
nErrId          : UDINT;
nLookAhead      : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. If the function block has a timeout error, ‘Error’ is TRUE and ‘nErrId’ is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

nLookAhead: Value of the look-ahead for bottleneck detection

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.7 ItpGetBottleNeckModeEx

The function block ItpGetBottleNeckModeEx reads the behavior in the event of a contour collision (bottleneck).

There is a more detailed description in the Interpreter [▶ 173] documentation.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc               : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy           : BOOL;
bErr            : BOOL;
nErrId          : UDINT;
eBottleNeckMode : E_ItpBnMode
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation or in the NC error documentation (error codes above 0x4000).

eBottleNeckMode: Enum for the behavior in the event of a contour collision

```
TYPE E_ItpBnMode:
(
ItpBnm_Abort := 0,
ItpBnm_Adjust := 1,
ItpBnm_Leave := 2
);
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.8 ItpGetChannelId

ItpGetChannelId is a function that determines the channel ID from the cyclic interface.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

ItpGetChannelId: Channel ID (type: UDINT)

Sample

```
VAR
nChnId          : UINT;
sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
nChnId := ItpGetChannelId( sNciToPlc );
```

see also: ItpGetGroupId [▶ 202]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.9 ItpGetChannelType

ItpGetChannelType is a function that returns the channel type of the cyclic interface.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

ItpGetChannelType: Channel type (type: E_ItpChannelType)

```
TYPE E_ItpChannelType :
(
ItpChannelTypeNone,
ItpChannelTypeInterpreter,
ItpChannelTypeKinematic,
ItpChannelType_InvalidItfVer := 16#4B14 (*ErrTcNciItp_ItfVersion the cyclic channel inter-
face does not match to the requested function/fb *)
);
END_TYPE
```

Sample

```
VAR
nChannelType    : E_ItpChannelType;
sNciToPlc AT%I* : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
nChannelType := ItpGetChannelType( sNciToPlc );
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.10 ItpGetCyclicLrealOffsets

The function block ItpGetCyclicLRealOffsets is used to read the current configuration of the cyclic channel interface for LREAL variables.

VAR_INPUT

```
VAR_INPUT
bExecute            : BOOL;
tTimeOut            : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy               : BOOL;
bErr                : BOOL;
nErrId              : UDINT;
nIndexOffsetParam1 : UDINT;
nIndexOffsetParam2 : UDINT;
nIndexOffsetParam3 : UDINT;
nIndexOffsetParam4 : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

nIndexOffsetParam1: Group state (index offset) for parameter 1

nIndexOffsetParam2: Group state (index offset) for parameter 2

nIndexOffsetParam3: Group state (index offset) for parameter 3

nIndexOffsetParam4: Group state (index offset) for parameter 4

see also:

- ItpReadCyclicLRealParam1 [▶ 212]
- ItpSetCyclicLRealOffsets [▶ 220]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.11 ItpGetCyclicUDintOffsets

The function block ItpGetCyclicUDintOffsets is used to read the current configuration of the cyclic channel interface for UDINT variables.

VAR_INPUT

```
VAR_INPUT
bExecute                    : BOOL;
tTimeOut                    : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc               : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy                : BOOL;
bErr                 : BOOL;
nErrId               : UDINT;
nIndexOffsetParam1 : UDINT;
nIndexOffsetParam2 : UDINT;
nIndexOffsetParam3 : UDINT;
nIndexOffsetParam4 : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

nIndexOffsetParam1: Group state (index offset) for parameter 1

nIndexOffsetParam2: Group state (index offset) for parameter 2

nIndexOffsetParam3: Group state (index offset) for parameter 3

nIndexOffsetParam4: Group state (index offset) for parameter 4

see also:

- ItpReadCyclicUDintParam1 [▶ 212]
- ItpSetCyclicUdintOffsets [▶ 221]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.12 ItpGetError

ItpGetError is a function that returns the error number. A description of the NC error codes can be found here [▶ 313].

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc              : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

ItpGetError: Error number

NOTE! ItpGetError evaluates the variable 'nItpErrCode' from the cyclic interface.

Sample

```
VAR
bItpError       : BOOL;
nErrId          : UINT;
sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
bItpError := ItpHasError( sNciToPlc );
IF bItpError THEN
nErrId := ItpGetError( sNciToPlc );
…
END_IF
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.13 ItpGetGeoInfoAndHParamEx

The function block `ItpGetGeoInfoAndHParamEx` reads informations of the currently active segment and past and future segments. These include block number, H-parameter and residual path length on the segment.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

VAR_OUTPUT

```
VAR_OUTPUT
stTab                : ST_ItpPreViewTabEx;
nErrId               : UDINT;
END_VAR
```

stTab: Structure containing the segment data. See ST_ItpPreViewTabEx [▶ 201].

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in `nErrId` can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

```
TYPE ST_ItpPreViewTabEx :
STRUCT
nDcTime                : UDINT := 0
nReserved              : UDINT := 0;
arrLines               : ARRAY[1..NCI_MAX_PREVIEWTABLINES] OF ST_ItpPreViewTabLine;
END_STRUCT
END_TYPE
```

nDcTime: Current time stamp in ns. This time stamp can be used e.g. in interplay with the `Tc2_NciXFC` library.

arrLines: Array of segment-related information (size 20). The entry at position 11 of the array corresponds to the currently active segment. Segments that have already been processed are displayed at positions 1-10 of the array, future segments at positions 12-20. See ST_ItpPreViewTabLine [▶ 201].

```
TYPE ST_ItpPreViewTabLine :
STRUCT
fLength                : LREAL := 0.0;
nBlockNo               : UDINT := 0;
nHParam                : UDINT := 0;
nEntryID               : UDINT := 0;
nReserved              : UDINT := 0;
END_STRUCT
END_TYPE
```

fLength: Remaining segment length. For segments that are not yet active this corresponds to the total segment length. For past segments the distance moved since the end of the segment is specified.

nBlockNo: Block number programmed by the user

nHParam: Value of the H-parameter [▶ 149] that is active from the start of the next segment

nEntryID: Command ID generated by the system

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.14 ItpGetGroupAxisIds

ItpGetGroupAxisIds is a function that returns an array of axes IDs that were configured for the group.

VAR_IN_OUT

```
FUNCTION ItpGetGroupAxisIds

VAR_IN_OUT
sNciToPlc AT%I*    : NCTOPLC_NCICHANNEL_REF;
nNciAxisIds        : ARRAY[1..8] OF DWORD;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

sNciAxisIds: Array of axis IDs

Return value

ItpGetGroupAxisIds: Error number

NOTE! ItpGetGroupAxisIds evaluates the information of the variable 'nAcsAxisIDs[8]' from the cyclic interface.

Sample

```
VAR
nNciAxisIds      : ARRAY[1..8] OF DWORD;
sNciToPlc AT%I* : NCTOPLC_NCICHANNEL_REF;
nVersionErr      : DWORD;
END_VAR
```

```
nVersionErr := ItpGetGroupAxisIds(nNciAxisIds, sNciToPlc );
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.15 ItpGetGroupId

ItpGetGroupId is a function that determines the group ID from the cyclic interface.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

ItpGetGroupId: Group ID

Sample

```
VAR
nGrpId          : UINT;
sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
nGrpId := ItpGetGroupId( sNciToPlc );
```

See also: ItpGetChannelId [▸ 196]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.16 ItpGetHParam

ItpGetHParam is a function that returns the current H-parameter.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc             : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

Return value

ItpGetHParam: H parameter

NOTE! ItpGetHParam evaluates the variable 'nHFuncValue' from the cyclic interface.

Sample

```
VAR
nHParam         : DINT;
sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
nHParam := ItpGetHParam( sNciToPlc );
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.17 ItpGetHskMFunc

ItpGetHskMFunc supplies the number of the M-function of type handshake.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc             : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

ItpGetHskMFunc: Number of the M-function

NOTE! ItpGetHskMFunc evaluates the variable 'nHskMFuncNo' from the cyclic interface.

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.18 ItpGetItfVersion

ItpGetItfVersion is a function that determines the version number of the cyclic interface.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc              : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

ItpGetItfVersion: Version number of the cyclic interface

Sample

```
VAR
nItfVer         : UINT;
sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
nItfVer := ItpGetItfVersion( sNciToPlc );
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.19 ItpGetOverridePercent

The ItpGetOverridePercent function returns the axis channel override as a percentage. It is essential to remember that this is not a value from the NC. The value, which is transferred as set value to the NC, is evaluated.

VAR_IN_OUT

```
VAR_IN_OUT
sPlcToNci           : PLCTONC_NCICHANNEL_REF;
END_VAR
```

sPlcToNci: Structure of cyclic channel interface between PLC and NCI (type: PLCTONC_NCICHANNEL_REF [▸ 308])

Return value

ItpGetOverridePercent: Override in percent

Sample

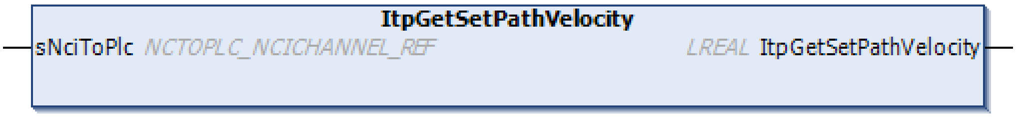
```
VAR
sPlcToNci AT%Q*: PLCTONC_NCICHANNEL_REF;
fOverride       : LREAL;
END_VAR
```

```
fOverride := ItpGetOverridePercent( sPlcToNci );
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.20 ItpGetSetPathVelocity



ItpGetSetPathVelocity is a function that reads the current set bath velocity from the cyclic interface.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

Return value

ItpGetSetPathVelocity: Current set bath velocity

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.21 ItpGetSParam

ItpGetSParam is a function that returns the current S-parameter.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

ItpGetSParam: S parameter

NOTE! ItpGetSParam evaluates the variable 'nSpindleRpm' from the cyclic interface.

Sample

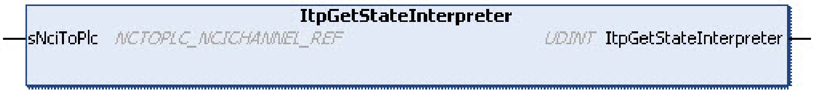
```
VAR
nSParam         : UINT;
sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
nSParam := ItpGetSParam( sNciToPlc );
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.22 ItpGetStateInterpreter



ItpGetStateInterpreter is a function that returns the interpreter status.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

ItpGetStateInterpreter: Current interpreter status

NOTE! ItpGetStateInterrpreter evaluates the variable 'nItpState' from the cyclic interface.

Sample

```
VAR
nItpState       : UDINT;
sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
nItpState := ItpGetStateInterpreter( sNciToPlc );
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.23 ItpGetTParam

ItpGetTParam is a function that returns the current T-parameter.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

ItpGetTParam: T parameter

NOTE! ItpGetTParam evaluates the variable 'nTool' from the cyclic interface.

Sample

```
VAR
nTParam         : UINT;
sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
nTParam := ItpGetTParam( sNciToPlc );
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.24 ItpGoAheadEx

The ItpGoAheadEx function block may only be used in association with the decoder stop '@717' [▶ 150]. There is a more detailed description of this decoder stop in the interpreter documentation [▶ 107].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.25 ItpHasError

ItpHasError is a function that determines whether the interpreter is in an error state.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc             : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

If there is an error, the function returns TRUE.

NOTE! ItpHasError evaluates the variable 'nItpErrCode' from the cyclic interface. If this value does not equal 0, TRUE is returned.

Sample

```
VAR
bItpError      : BOOL;
sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
bItpError := ItpHasError( sNciToPlc );
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.26 ItpIsFastMFunc

ItpIsFastMFunc is a function that determines whether the fast M-function is set for the supplied M-function number.

VAR_IN

```
FUNCTION ItpIsFastMFunc

VAR_IN
nFastMFuncNo    : INT;
END_VAR
```

nFastMFuncNo: Number of the M-function that is to be checked.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

The function returns TRUE if the fast bit of the M-function is set.

NOTE! ItpIsFastMFunc evaluates the variable 'nFastMFuncMask' from the cyclic interface.

Sample

```
(*this enum is defined by the user *)

TYPE FastMFuncs:
(
M10_CoolingFluidOn := 10, (*fast M-Funktion M10*)
M11_CoolingFluidOff := 11,
M12_FanOn := 12,
M13_FanOff := 13
);
END_TYPE
```

```
VAR
sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF
enFastMFuncs    : FastMFuncs;
bTurnFanOn     : BOOL;
END_VAR
```

```
bTurnFanOn := ItpIsFastMFunc( M12_FanOn,sNciToPlc );
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.27 ItpIsEStopEx

The function ItpIsEStopEx indicates whether an EStop command was triggered.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

If the return value is TRUE, the function was preceded by an EStop (e.g. ItpEStopEx). The flag does **not** provide information as to whether the axes have already stopped or are still on the braking ramp.

After execution of ItpStepOnAfterEStopEx, ItpIsEStopEx returns FALSE again.

NOTE! ItpIsEStopEx evaluates the cyclic interface.

see also:

ItpEStopEx [▶ 193]

ItpStepOnAfterEStopEx [▶ 228]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.28 ItpIsHskMFunc

ItpIsHskMFunc determines whether an M-function of type handshake is present.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc              : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

The function returns TRUE if an M-function of type handshake is present.

NOTE! ItpIsHskFunc evaluates the variable 'nHskMFuncReq' from the cyclic interface.

Sample

```
VAR
bMFuncRequest    : BOOL;
sNciToPlc AT%I* : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
bMFuncRequest := ItpIsHskMFunc( sNciToPlc );
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.29 ItpLoadProgEx

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
sPrg            : STRING(255);
nLength         : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The function block reads the NC program when a rising edge is encountered

sPrg: Name of the NC program that is loaded

nLength: String length of the program name

tTimeOut: ADS Timeout-Delay

The NC program is looked up in directory "TwinCAT\Mc\Nci", if no further information is available. It is however also possible to give an absolute path.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc               : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Sample

```
VAR
in_stItpToPlc AT %I*            : NCTOPLC_NCICHANNEL_REF;
fbLoadProg                      : ItpLoadProgEx;
sProgramPath                    : STRING (255):= 'TestIt.nc';
END_VAR
```

```
fbLoadProg(
bExecute := TRUE,
sPrg := sProgramPath,
nLength := LEN(sProgramPath),
tTimeOut := t#200ms,
sNciToPlc := in_stItpToPlc
);
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.30 ItpReadCyclicLRealParam1

This function reads the first LREAL parameter from the cyclic channel interface. This parameter is configured previously with ItpSetCyclicLRealOffsets [▸ 220].

Parameter 2 to 4 are read via the same mechanism (e.g. ItpReadCyclicLRealParam2).

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

Return value

Parameter 1 of type LREAL.

See also:

- ItpReadCyclicUdintParam1 [▸ 212]
- ItpSetCyclicLRealOffsets [▸ 220]
- ItpGetCyclicLRealOffsets [▸ 198]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.31 ItpReadCyclicUdintParam1

This function reads the first UDINT parameter from the cyclic channel interface. This parameter is configured previously with ItpSetCyclicUdintOffsets [▸ 221].

Parameter 2 to 4 are read via the same mechanism (e.g. ItpReadCyclicUdintParam2).

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

Return value

Parameter 1 of type UDINT.

See also:

- ItpReadCyclicLRealParam1 [▸ 212]
- ItpSetCyclicUdintOffsets [▸ 221]

- ItpGetCyclicUdintOffsets [▶ 199]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.32 ItpReadRParamsEx

The ItpReadRParamsEx function block reads the NC's calculation parameters, also known as R-parameters. A more detailed description of the calculation parameters can be found here [▶ 115]. A total of 1000 R-parameters are available, of which the first 900 (0..899) are local, so that they are only visible in the current NC channel. The other 100 (900..999) R-parameters are global, and are thus visible from anywhere in the NC.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
pAddr           : PVOID;
nIndex          : DINT;
nCount          : DINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: A rising edge starts the read operation.

pAddr: Address of the target variables of the data to be read. Data are written directly from the specified address, i.e. nIndex is not to be interpreted as offset from pAddr. The data are usually stored in an array of type LREAL, which has to be defined by the user.

nIndex: Describes the index of the R-parameter to be read from an NC perspective.

nCount: Number of R-parameters to be read

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc                : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

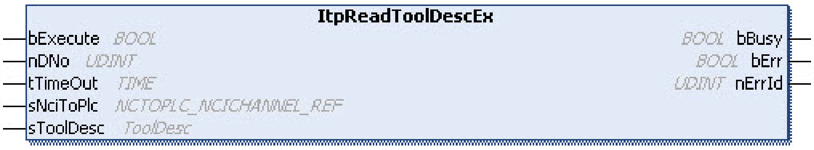
bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.33 ItpReadToolDescEx



The ItpReadToolDescEx function block reads the tool parameters for the supplied D-word.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nDNo            : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nDNo: D-word for which the tool parameters are to be read. nDNo can have values between 1 and 255.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc       : NCTOPLC_NCICHANNEL_REF;
sToolDesc       : ToolDesc;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading.

sToolDesc: A structure into which the tool parameters of nDNo are written. The meaning of the parameters depends on the tool type, and can be found in the tool data [▸ 162]. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

```
TYPE ToolDesc:
STRUCT
nToolNumber     : UDINT;  (*valid range from 0 .. 65535*)
nToolType       : UDINT;
fParam          : ARRAY [2..15] OF LREAL;
END_STRUCT
END_TYPE
```

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

See also:

ItpWriteToolDescEx [▸ 230]

ItpSetToolDescNullEx [▸ 224]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.34 ItpReadZeroShiftEx

The ItpReadZeroShiftEx function block reads the zero shift components X, Y and Z for the given zero shift.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nZsNo           : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nZsNo: Number of the zero shift. G54 to G59 are zero shifts at the NC. The valid range of values for 'nZsNo' is therefore from 54 to 59.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc       : NCITOPLC_NCICHANNEL_REF;
sZeroShiftDesc  : ZeroShiftDesc;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

sZeroShiftDesc: The structure containing the components of the zero shift.

```
TYPE ZeroShiftDesc:
STRUCT
fShiftX         : LREAL;
fShiftY         : LREAL;
fShiftZ         : LREAL;
END_STRUCT
END_TYPE
```

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

NOTE! For reasons of compatibility, there are two entries (coarse and fine) for each axis in each zero shift (e.g. G54). These two entries must be added together. This function block evaluates both the entries and adds them together automatically.

See also:

ItpWriteZeroShiftEx [▸ 231]

ItpSetZeroShiftNullEx [▸ 225]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.35 ItpResetEx2

The function block 'ItpResetEx2' executes a channel reset, which deletes all existing tables of the NC channel. In contrast to the outdated function block ItpReset, an active channel is stopped first, before the reset is executed. This simplifies programming in the PLC, since no explicit check is necessary to ascertain whether the axes are still in motion.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

tTimeOut: ADS timeout delay (the bBusy signal can be active for longer than tTimeOut)

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.36 ItpResetFastMFuncEx

The fast M-function nMFuncNo [▸ 145] is reset with a rising edge at input bExecute. In the event of the M-function not being available, **no** error is returned.

This function block represents an alternative to Auto-reset or reset with another M-function (reset list during parameterization of the M-function). For reasons of transparency, mixed resets using an M-function and this function block should be avoided.

VAR_INPUT

```
VAR_INPUT
bExecute         : BOOL;
nMFuncNo         : UINT;
tTimeOut         : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nMFuncNo: Flying M-function that is to be reset

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc               : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.37 ItpSetBottleNeckLookAheadEx

The function block ItpSetBottleNeckLookAheadEx determines the maximum number of segments the system may look ahead for bottleneck detection (contour collision monitoring). Note that segments, which were added as a result of radius compensation (e.g. additional segments at acute angles) are taken into account.

There is a more detailed description in the Interpreter [▶ 173] documentation.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nLookAhead      : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nLookAhead: Specifies the look-ahead value

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.38 ItpSetBottleNeckModeEx

The function block ItpSetBottleNeckModeEx specifies the behavior in the event of a contour collision (bottleneck).

There is a more detailed description in the Interpreter [▶ 173] documentation.

VAR_INPUT

```
VAR_INPUT
bExecute       : BOOL;
eBottleNeckMode: E_ItpBnMode
tTimeOut       : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

eBottleNeckMode: Enum for the behavior in the event of a contour collision

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc             : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

```
TYPE E_ItpBnMode:
(
ItpBnm_Abort    := 0,
ItpBnm_Adjust   := 1,
ItpBnm_Leave    := 2
);
END_TYPE
```

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.39 ItpSetCyclicLrealOffsets

The function block ItpSetCyclicLrealOffsets is used to describe the cyclic channel interface for the 4 freely configurable LREAL variables. Variables (index offsets) can be selected from the group state [▶ 396].

The functionality is only active if nIndexOffsetParam1 is not equal 0.

VAR_INPUT

```
VAR_INPUT
bExecute           : BOOL;
tTimeOut           : TIME;
nIndexOffsetParam1 : UDINT;
nIndexOffsetParam2 : UDINT;
nIndexOffsetParam3 : UDINT;
nIndexOffsetParam4 : UDINT;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

tTimeOut: ADS Timeout-Delay

nIndexOffsetParam1: Group state (index offset) for parameter 1

nIndexOffsetParam2: Group state (index offset) for parameter 2

nIndexOffsetParam3: Group state (index offset) for parameter 3

nIndexOffsetParam4: Group state (index offset) for parameter 4

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

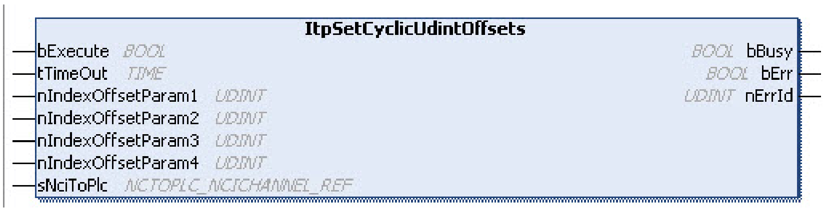
See also:

- ItpReadCyclicLRealParam1 [▶ 212]
- ItpGetCyclicLRealOffsets [▶ 198]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.40 ItpSetCyclicUDintOffsets



The function block ItpSetCyclicUDintOffsets is used to describe the cyclic channel interface for the 4 freely configurable UDINT variables. Variables (index offsets) can be selected from the group state [▶ 396].

The functionality is only active if nIndexOffsetParam1 is not equal 0.

VAR_INPUT

```
VAR_INPUT
bExecute           : BOOL;
tTimeOut           : TIME;
nIndexOffsetParam1 : UDINT;
nIndexOffsetParam2 : UDINT;
nIndexOffsetParam3 : UDINT;
nIndexOffsetParam4 : UDINT;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

tTimeOut: ADS Timeout-Delay

nIndexOffsetParam1: Group state (index offset) for parameter 1

nIndexOffsetParam2: Group state (index offset) for parameter 2

nIndexOffsetParam3: Group state (index offset) for parameter 3

nIndexOffsetParam4: Group state (index offset) for parameter 4

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

See also:

- ItpReadCyclicUDintParam1 [▸ 212]
- ItpGetCyclicUdintOffsets [▸ 199]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.41 ItpSetOverridePercent

The function ItpSetOverridePercent writes the axes channel override into the cyclic interface of the NCI. The override is passed as a percentage.

VAR_INPUT

```
FUNCTION ItpSetOverridePercent

VAR_INPUT
fOverridePercent : LREAL;
END_VAR
```

fOverridePercent: Axis channel override as a percentage

VAR_IN_OUT

```
VAR_IN_OUT
sPlcToNci         : PLCTONC_NCICHANNEL_REF;
END_VAR
```

sPlcToNci: Structure of cyclic channel interface between PLC and NCI (type: PLCTONC_NCICHANNEL_REF [▸ 308])

Return value

ItpSetOverridePercent: always TRUE

Sample

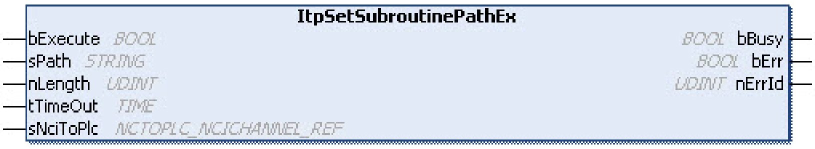
```
VAR
sPlcToNci AT%Q*: PLCTONC_NCICHANNEL_REF;
fOverride       : LREAL;
END_VAR
```

```
fOverride := 47.11;
ItpSetOverridePercent( fOverride, sPlcToNci );
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.42 ItpSetSubroutinePathEx



With ItpSetSubroutinePathEx function block, the search path for subroutines can optionally be set.

If a subroutine still has to be integrated, the file is searched in the following order:

1. optional search path (ItpSetSubroutinePath)
2. path from which the main program was loaded
3. TwinCAT\Mc\Nci directory

Only one optional path can take effect, which remains active until it is overwritten with another path or an empty string.

After a TwinCAT restart, the path has to be re-assigned.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
sPath           : STRING;
nLength         : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

sPath: Optional path for subroutines. Is deactivated with an empty string

nLength: String length

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

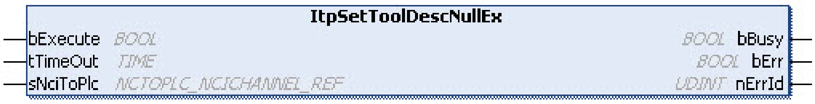
bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.43 ItpSetToolDescNullEx



FB ItpSetToolDescNullEx overwrites all tool parameters (incl. number & type) of the channel with zero.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
tTimeOut        : TIME;
END_VAR
```

bExecute: A rising edge results in overwriting of all tool parameters of the NC channel with zero.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc               : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

See also:

ItpWriteToolDescEx [▸ 230]

ItpReadToolDescEx [▸ 214]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.44 ItpSetZeroShiftNullEx

The function block ItpSetZeroShiftNullEx overwrites all zero shifts of the channel with zero.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
tTimeOut        : TIME;
END_VAR
```

bExecute: A rising edge results in overwriting of all zero shifts of the NC channel with zero.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc                : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

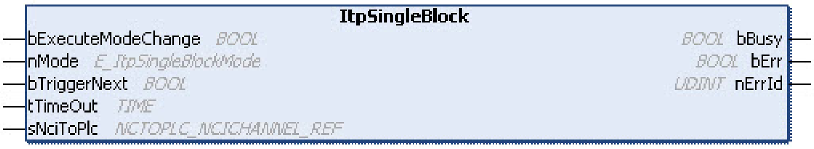
See also:

- ItpWriteZeroShiftEx [▸ 231]
- ItpReadZeroShiftEx [▸ 215]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.45 ItpSingleBlock



The ItpSingleBlock function block activates or deactivates single block mode in the NCI. Block relaying can be triggered directly from the PLC with the input 'bTriggerNext'. Alternatively the Start button of the interpreter (F5) can be used in the XAE .

A more detailed description can be found in the interpreter documentation [▸ 114].

VAR_INPUT

```
VAR_INPUT
bExecuteModeChange   : BOOL;
nMode                : E_ItpSingelBlockMode;
bTriggerNext         : BOOL
tTimeOut             : TIME;
END_VAR
```

bExecuteModeChange: Single block mode (nMode) is activated through a rising edge at this input.

nMode: Operation mode for single block (cf. single block mode):

- ItpSingleBlockOff: single block off
- ItpSingleBlockNck: single block in NC kernel
- ItpSingleBlockIntp: single block in interpreter

bTriggerNext: Block relaying is triggered by a rising edge at this input.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

```
TYPE E_ItpSingleBlockMode:
(
ItpSingleBlockOff    := 0,
ItpSingleBlockNck    := 1,
ItpSingleBlockIntp   := 16#4000
);
END_TYPE
```

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.46 ItpStartStopEx

The function block ItpStartStopEx starts or stops the NC channel.

VAR_INPUT

```
VAR_INPUT
bStart          : BOOL;
bStop           : BOOL;
tTimeOut        : TIME;
END_VAR
```

bStart: A positive edge starts the NC channel

bStop: A positive edge stops the NC channel. A stop command deletes all the tables in the NC and brings the axes to a controlled halt.

NOTE! The bStop input has a higher priority than the bStart input, so that if both inputs receive a positive edge, a channel stop will be executed.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.47 ItpStepOnAfterEStopEx

The function block ItpStepOnAfterEStopEx enables further processing of the parts program after a programmed EStopEx.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc                : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

See also:

ItpEStopEx [▸ 193]

ItpIsEStopEx [▸ 209]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.48 ItpWriteRParamsEx

The function block ItpWriteRParamsEx writes R-parameters into the NC.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
pAddr           : DWORD;
nIndex          : DINT;
nCount          : DINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: A rising edge starts the write operation.

pAddr: Address of the variables containing the data to be written. Data are used directly from the specified address, i.e. nIndex is not to be interpreted as offset from pAddr. The data are usually read from an array of type LREAL, which has to be defined by the user.

nIndex: Describes the index of the R-parameter to be written from an NC perspective.

nCount: Number of R-parameters to be written

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc                    : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Sample

```
VAR
arrfRParam90to99 : ARRAY[0..9] OF LREAL;
fbWriteRParamEx  : ItpWriteRParamsEx;
n                : INT := 0;
bWriteParam      : BOOL := FALSE;
sNciToPlc AT%I*  : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
FOR n:=0 TO 9 DO
arrfRParam90to99[n] := 90 + n;
END_FOR

fbWriteRParam(
 bExecute := bWriteParam,
 pAddr := ADR( arrfRParam90to99[0] ),
 nIndex := 90,
 nCount := 10,
 tTimeOut := T#200ms,
 sNciToPlc := sNciToPlc );
```

In this example the parameters R90 to R99 are written from an NC perspective.

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.49 ItpWriteToolDescEx

The function block **ItpWriteToolDescEx** writes a block of tool parameters.

VAR_INPUT

```
VAR_INPUT
bExecute       : BOOL;
nDNo           : UDINT;
tTimeOut       : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nDNo: D-word for which the tool parameters are to be read. nDNo can have values between 1 and 255.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc      : NCTOPLC_NCICHANNEL_REF;
sToolDesc      : ToolDesc;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

sToolDesc: The structure that contains the new tool parameters. This structure is only accessed for reading. The meaning of the parameters depends on the tool type, and can be found in the tool data [▶ 162].

```
TYPE ToolDesc:
STRUCT
nToolNumber    : UDINT; (*valid range from 0 .. 65535*)
nToolType      : UDINT;
fParam         : ARRAY [2..15] OF LREAL;
END_STRUCT
END_TYPE
```

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

See also:

ItpReadToolDescEx [▶ 214]

ItpSetToolDescNullEx [▶ 224]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.50 ItpWriteZeroShiftEx

The function block **ItpWriteZeroShiftEx** writes the shift components X, Y and Z for the specified zero shift.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nZsNo           : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nZsNo: Number of the zero shift.

G54 to G59 are zero shifts at the NC. G58 and G59 can only be edited from the NC program. The valid range of values for 'nZsNo' is therefore from 54 to 57.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
sZeroShiftDesc : ZeroShiftDesc;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

sZeroShiftDesc: The structure containing the components of the zero shift. This structure is only accessed for reading.

```
TYPE ZeroShiftDesc:
STRUCT
fShiftX         : LREAL;
fShiftY         : LREAL;
```

```
fShiftZ        : LREAL;
END_STRUCT
END_TYPE
```

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

NOTE! For reasons of compatibility every zero shift that can be set has two parameters (coarse and fine) for each axis. When using this function block to write a new zero shift, the new value is written into the 'fine parameter'. A value of 0.0 is entered into the 'coarse parameter'.

This makes it possible to use a function block such as ItpReadZeroShiftEx [▶ 215] to read and modify a zero shift and to send it back to the NC.

See also:

- ItpReadZeroShiftEx [▶ 215];
- ItpSetZeroShiftNullEx [▶ 225]

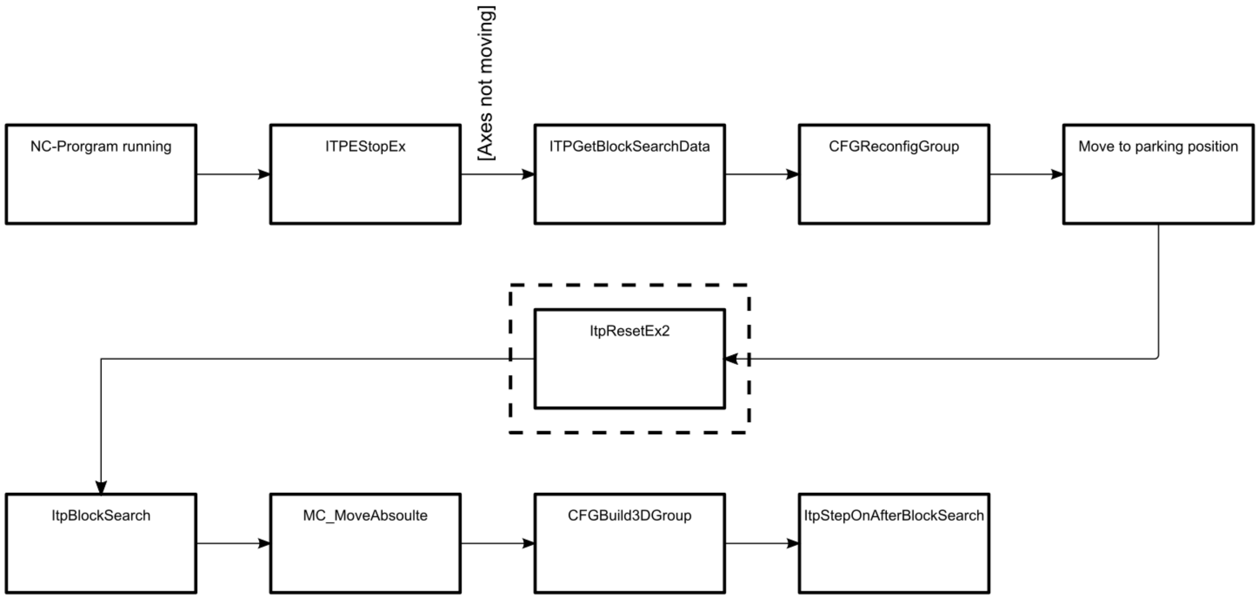
Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.51 Blocksearch

Block search can be used to interrupt a program for a tool change or at the end of a shift. After the interruption the program can continue at the previous position.

The diagram illustrates how the block search is used.



5.1.2.51.1 ItpBlocksearch

The function block ItpBlocksearch sets the interpreter to the point defined at the inputs. The input values can be taken from function block ItpGetBlocksearchData [▶ 236] or set manually. Once the interpreter has been set to the defined location with ItpBlocksearch, the motion can continue with ItpStepOnAfterBlocksearch [▶ 237] at the position indicated at output sStartPosition.

VAR_INPUT

```
VAR_INPUT
bExecute            : BOOL;
nBlockId            : UDINT;
eBlockSearchMode    : E_ItpBlockSearchMode;
eDryRunMode         : E_ItpDryRunMode;
fLength             : LREAL;
sPrgName            : STRING(255);
nPrgLength          : UDINT;
tTimeOut            : TIME;
sAxesList           : ST_ItpAxes;
sOptions            : ST_ItpBlockSearchOptions;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nBlockId: Block number or EntryCounter of the segment in the NC program used as starting point.

eBlockSearchMode: Defines whether the specified nBlockId is a block number (e.g. N4711) or continuous EntryCounter. A prerequisite for using the block number is that it is unique. See E_ItpBlockSearchMode [▶ 234]

eDryRunMode: Defines which program lines are executed and which are skipped. See E_ItpDryRunMode [▶ 234]

fLength: Remaining length within the segment selected with nBlockId in percent.

sPrgName: Name or path of the program to be executed.

nPrgLength: Indicates the length of string *sPrgName*.

tTimeOut: ADS timeout delay

sAxesList: Definition of the axes in the NCI group. See ST_ItpAxes [▸ 235]

sOptions: Provides information on retrace.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy               : BOOL;
bErr                : BOOL;
nErrId              : UDINT;
bDone               : BOOL;
sStartPosition      : ST_ItpBlockSearchStartPosition;
END_VAR
```

bBusy: Remains TRUE until the function block has executed a command request, but no longer than the time specified at the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs.

bErr: Becomes TRUE if an error occurs during command execution. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

bDone: The output becomes TRUE when the command was executed successfully.

sStartPosition: Indicates the start position from which the NC program continues. The individual axes should be moved to this position before ItpStepOnAfterBlocksearch [▸ 237] is executed. See ST_ItpBlockSearchStartPosition [▸ 235]

E_ItpBlockSearchMode

E_ItpBlockSearchMode is used to define in which way the block search is executed.

```
TYPE E_ItpBlockSearchMode :
(
ItpBlockSearchMode_Disable        := 0,
ItpBlockSearchMode_BlockNo        := 1,
ItpBlockSearchMode_EntryCounter   := 2
);
END_TYPE
```

ItpBlockSearchMode_Disable: Block search disabled (initial value).

ItpBlockSearchMode_BlockNo: The block search is executed via the block number (e.g. N4711) programmed by the user in the NC program. A prerequisite is that the user-defined block number is unique.

ItpBlockSearchMode_EntryCounter: The block search is executed via a unique EntryCounter. This EntryCounter is implicitly unique, but it is not visible to the user in the NC program.

E_ItpDryRunMode

E_ItpDryRunMode defines how blocks are handled, which are located before the block in which the block search is started, when the program is executed.

```
TYPE E_ItpDryRunMode :
(
ItpDryRunMode_Disable              := 0,
ItpDryRunMode_SkipAll              := 1,
ItpDryRunMode_SkipMotionOnly       := 2,
ItpDryRunMode_SkipDwellAndMotion   := 3
);
END_TYPE
```

ItpDryRunMode_Disable: DryRun disabled (initial value).

ItpDryRunMode_SkipAll: All previous blocks are skipped. R-parameters are written.

ItpDryRunMode_SkipMotionOnly: Only movement blocks are skipped. R-parameters are written, and dwell times and M-functions are executed.

ItpDryRunMode_SkipDwellAndMotion: Movement blocks and dwell times are skipped. R-parameters are written and M-functions are executed.

ST_ItpAxes

The structure ST_ItpAxes contains the axes that were in the NCI group during program execution.

```
TYPE ST_ItpAxes :
STRUCT
nAxisIds          : ARRAY[1..8] OF UDINT;
END_STRUCT
END_TYPE
```

nAxisIds: Array of axes that were in the NCI group. The order is nAxisIds[1]=X, nAxisIds[2]=Y, nAxisIds[3]=Z, nAxisIds[4]=Q1, nAxisIds[5]=Q2… The axis ID can be read from the cyclic axis interface.

St_ItpBlockSearchOptions

The structure contains information on the retrace functionality.

```
TYPE ST_ItpBlockSearchOptions :
STRUCT
bIsRetrace        : BOOL:= FALSE;
bRetraceBackward  : BOOL:= FALSE;
END_STRUCT
END_TYPE
```

bIsRetrace: Indicates whether the retrace functionality is active.

bRetraceBackward: Indicates whether backward movement took place on the path.

ST_ItpBlockSearchStartPosition

The structure indicates the position at which the NC program continues after a block search. The user is responsible for moving the axes to the corresponding positions.

```
TYPE ST_ItpBlockSearchStartPosition :
STRUCT
fStartPosition    : ARRAY[1..8] OF LREAL;
END_STRUCT
END_TYPE
```

fStartPosition: Array of axis positions at which the NC program continues.

The order is fStartPosition[1]=X, fStartPosition [2]=Y, fStartPosition [3]=Z, fStartPosition [4]=Q1, fStartPosition [5]=Q2…

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.51.2 ItpGetBlocksearchData

The function block ItpGetBlocksearchData reads the current position on the path. Usually this command is called at standstill. Subsequently ItpBlockSearch [▸ 233] can be used to set the interpreter to the position stored in sBlockSearchData.

VAR_INPUT

```
VAR_INPUT
bExecute           : BOOL;
tTimeOut           : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

bTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy              : BOOL;
bErr               : BOOL;
nErrId             : UDINT;
sBlockSearchData   : ST_ItpBlockSearchData;
END_VAR
```

bBusy: Remains TRUE until the function block has executed a command request, but no longer than the time specified at the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs.

bErr: Becomes TRUE if an error occurs during command execution. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

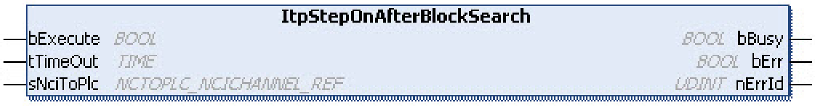
sBlockSearchData: Contains information on the current position on the path.

```
TYPE ST_ItpBlockSearchData :
STRUCT
fLength            : LREAL;(* remaining distance of actual movement block in percent*)
nBlockNo           : UDINT;(* number of the actual block *)
nBlockCounter      : UDINT;(* counter value of the actual block *)
bIsRetrace         : BOOL;(* e.g. by activ Retrace *)
bRetraceBackward   : BOOL;(* e.g. by activ Retrace Backward *)
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.51.3 ItpStepOnAfterBlocksearch



Starts the motion after a block search.

The axes first have to be moved to the positions output by ItpBlocksearch [▶ 233].

VAR_INPUT

```
VAR_INPUT
bExecute            : BOOL;
tTimeOut            : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

bTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc               : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

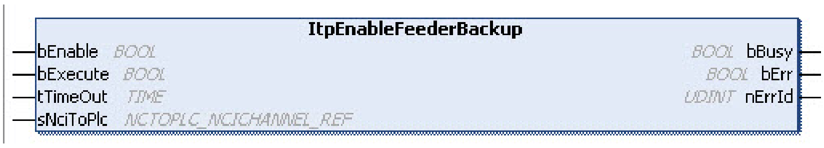
nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.52 Retrace

5.1.2.52.1 ItpEnableFeederBackup



The function block ItpEnableFeederBackup enables storing of the path for retracing. It has to be activated once before the NC program (G-Code) is started. If the Blocksearch [▶ 232] functionality is used, ItpEnableFeederBackup has to be activated before ItpBlocksearch [▶ 233] is called. Feeder backup is executed as long as a TwinCAT restart or bEnable = FALSE is triggered with a rising edge at bExecute.

If feeder backup is not enabled, retracing does not work. This can be verified via ItpIsFeederBackupEnabled [▶ 238].

VAR_INPUT

```
VAR_INPUT
bEnable         : BOOL;
bExecute        : BOOL;
tTimeOut        : TIME;
END_VAR
```

bEnable: TRUE: enables feeder backup, FALSE: disables feeder backup

bExecute: The command is triggered by a rising edge at this input.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc              : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

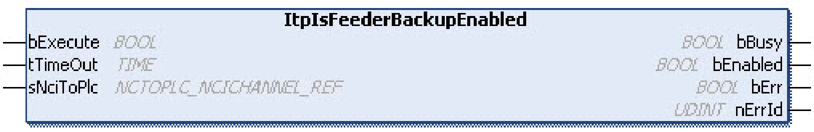
bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.52.2 ItpIsFeederBackupEnabled



The function block ItpIsFeederBackupEnabled indicates whether feeder backup is enabled. Feeder backup must be enabled before reversing can take place. This activates storing of the path.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc              : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy           : BOOL;
bEnabled        : BOOL;
bErr            : BOOL;
nErrId          : UDINT;
END_VAR
```

bBusy: The bBusy output remains TRUE until the function block has executed a command, with the maximum duration specified by the time associated with the 'Timeout' input. While bBusy = TRUE, no new instruction will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bEnabled: TRUE: Backup list for tracing is enabled, FALSE: Backup list for tracing is disabled

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. If the function block has a timeout error, 'Error' is TRUE and 'nErrId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.52.3 ItpIsFeedFromBackupList

The function ItpIsFeedFromBackupList becomes TRUE when the feed entries (SAF & SVB) were sent from the backup list. During backward movement all entries are sent from the backup list. If the program is executed in forward mode, the first entries usually also originate from the backup list. This is dependent of the number of retraced entries and the number of entries in the SVB and SAF tables at the time at which tracing was called. All further commands originate from the ‚original' code.

While the NCI is processing the backup list, not all functions are available or meaningful. Here are a few examples:

- Decoder stops such as @714 are not evaluated
- Modifications of R-parameters do not take effect as long as the motion takes place on the backup path (forward or backward). R-parameters modifications take effect again as soon as the path data no longer come from the backup list.

VAR_IN_OUT

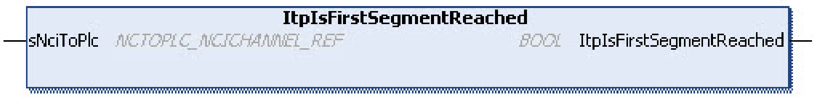
```
VAR_IN_OUT
sNciToPlc              : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.52.4 ItpIsFirstSegmentReached



ItpIsFirstSegmentReached is a function that determines whether the program start position is reached during retracing, based on the cyclic channel interface.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc              : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

The function returns TRUE when the start position of the G-Code program is reached. If the version number of the cyclic channel interface is less than 6, the return value is always FALSE.

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.52.5 ItpIsMovingBackwards

ItpIsMovingBackwards is a function that determines whether backward movement takes place on the path of the current G-Code program, based on the cyclic channel interface.

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc              : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

Return value

The function returns TRUE when backward movement takes place on the path. If the version number of the cyclic channel interface is less than 6, the return value is always FALSE.

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.52.6 ItpRetraceMoveBackward

The function block ItpRetraceMoveBackward deals with the geometric entries at the actual position at the start of the part program (G-Code).

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc              : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Procedure

1. Activate feeder backup list (see ItpEnableFeederBackup [▶ 237])
   ⇨ The NC program is stopped with ItpEStopEx [▶ 193]
2. Wait and ensure that all axes in the group are at standstill
3. Call ItpRetraceMoveBackward
4. Stop backward movement with ItpEStop, otherwise the program returns to the start
5. Call ItpRetraceMoveForward [▶ 242] to move forward again
6. Call ItpEStopEx and ItpRetraceMoveBackward etc., if required.

NOTE! Do not use in conjunction with vertex blending. M-functions are suppressed during backward movement.

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.2.52.7 ItpRetraceMoveForward

The function block ItpRetraceMoveForward transfers all entries from the current block (e.g. position) in forward travel direction to the NC kernel. It is called to reverse the direction after ItpRetraceMoveBackward [▸ 241] was called.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc               : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▸ 306])

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

see also: ItpRetraceMoveBackward [▸ 241]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

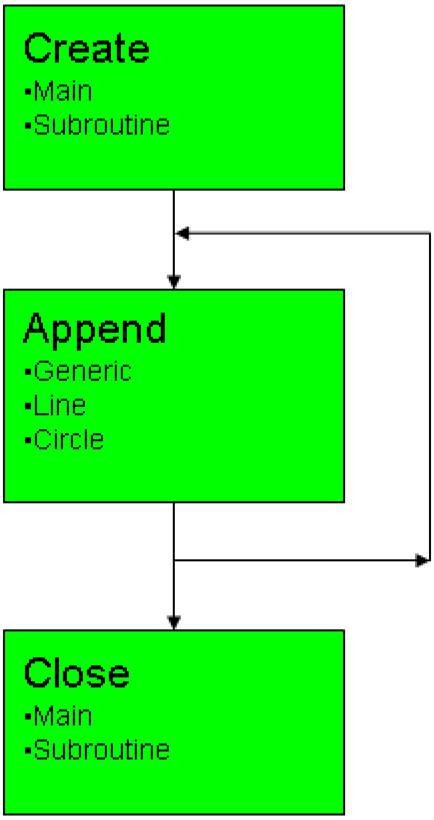
5.1.3 Parts program generator

The function blocks ItpPpg* provide an option for creating a parts program (G-Code file) from the PLC. During program generation a distinction is made between a main program (ItpPpgCreateMain [▸ 248]) and a subroutine (ItpPpgCreateSubroutine [▸ 249]).

Subsequently ItpPpgAppend* can be used to add various NC lines. The following function blocks are available:

- ItpPpgAppendGeoLine [▸ 246] adds a linear motion.
- ItpPpgAppendGeoCircleByRadius [▸ 245] adds a circle with radius specification.
- ItpPpgAppendGenericBlock [▸ 244] inserts a self-defined line, such as activation of rounding or M-functions.

Once the parts program is complete, it is closed with the routines ItpPpgCloseMain [▸ 247] or ItpPpgCloseSubroutine [▸ 247].



The following function blocks can be used:

Function Block	Description
ItpPpgAppendGenericBlock [▸ 244]	Appends a generic NC line to a specified parts program
ItpPpgAppendGeoCircleByRadius [▸ 245]	Adds a circle to a specified parts program
ItpPpgAppendGeoLine [▸ 246]	Adds a linear motion to a specified parts program
ItpPpgCloseMain [▸ 247]	Closes a previously opened parts program
ItpPpgCloseSubroutine [▸ 247]	Closes a previously opened subroutine
ItpPpgCreateMain [▸ 248]	Opens or generates a parts program
ItpPpgCreateSubroutine [▸ 249]	Opens or generates a subroutine

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.3.1 ItpPpgAppendGenericBlock

The function block ItpPpgAppendGenericBlock adds a generic line to the parts program. It can be used to activate an M-function or rounding, for example.

Before the actual call, call ItpPpgCreateMain [▸ 248] or ItpPpgCreateSubroutine [▸ 249].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    sPathName       : STRING;
    sBlock          : STRING;
    tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

sPathName: Name of the parts program including path name

sBlock: Generic line to be added to the parts program

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

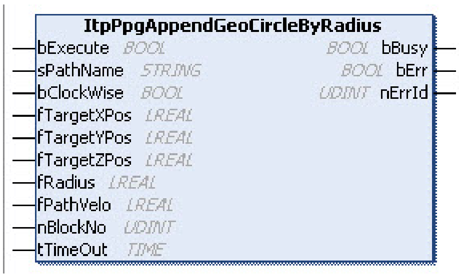
bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.3.2 ItpPpgAppendGeoCircleByRadius



The function block ItpPpgAppendGeoCircleByRadius adds a circular motion to the parts program. The circle is parameterized by the radius.

Before the actual call, call ItpPpgCreateMain [▸ 248] or ItpPpgCreateSubroutine [▸ 249].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    sPathName       : STRING;
    bClockWise      : BOOL;
    fTargetXPos     : LREAL;
    fTargetYPos     : LREAL;
    fTargetZPos     : LREAL;
    fRadius         : LREAL;
    fPathVelo       : LREAL;
    nBlockNo        : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

sPathName: Name of the parts program including path name

bClockwise: If TRUE, the movement along the circle is clockwise, otherwise counter-clockwise

fTargetXPos: Target position of the X axis

fTargetYPos: Target position of the Y axis

fTargetZPos: Target position of the Z axis

fRadius: Circle radius

fPathVelo: Path velocity

nBlockNo: Line number in the parts program

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

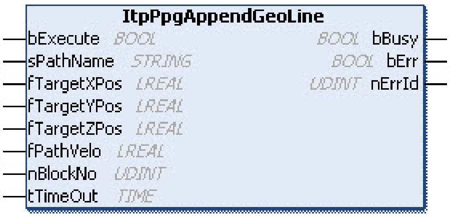
bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.3.3 ItpPpgAppendGeoLine



The function block ItpPpgAppendGeoLine adds a linear motion to the parts program. In addition to the actual target position, the path velocity and the line number are transferred.

Before the actual call, call ItpPpgCreateMain [▸ 248] or ItpPpgCreateSubroutine [▸ 249].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
sPathName       : STRING;
fTargetXPos     : LREAL;
fTargetYPos     : LREAL;
fTargetZPos     : LREAL;
fPathVelo       : LREAL;
nBlockNo        : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

sPathName: Name of the parts program including path name

fTargetXPos: Target position of the X axis

fTargetYPos: Target position of the Y axis

fTargetZPos: Target position of the Z axis

fPathVelo: Path velocity

nBlockNo: Line number in the parts program

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy       : BOOL;
bErr        : BOOL;
nErrId      : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.3.4 ItpPpgCloseMain

The function block ItpPpgCloseMain completes the main program with the corresponding code for the interpreter (M02).

Before the actual call, call ItpPpgCreateMain [▶ 248].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
sPathName       : STRING;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

sPathName: Name of the parts program including path name

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

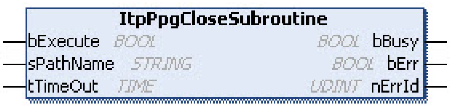
bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.3.5 ItpPpgCloseSubroutine



The function block ItpPpgCloseSubroutine completes the subroutine with the corresponding code for the interpreter (M17).

Before the actual call, call ItpPpgCreateSubroutine [▸ 249].

VAR_INPUT

```
VAR_INPUT
bExecute       : BOOL;
sPathName      : STRING;
tTimeOut       : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

sPathName: Name of the parts program including path name

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.3.6 ItpPpgCreateMain

The function block ItpPpgCreateMain generates a new file, which can later be processed as main program. If the file does not yet exist, it is created, otherwise it is overwritten.

VAR_INPUT

```
VAR_INPUT
bExecute       : BOOL;
sPathName      : STRING;
tTimeOut       : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

sPathName: Name of the parts program including path name

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.3.7 ItpPpgCreateSubroutine

The function block ItpPpgCreateSubroutine generates a new file, which can later be processed as subroutine. If the file does not yet exist, it is created, otherwise it is overwritten.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
sPathName       : STRING;
nSubroutineId   : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

sPathName: Name of the subroutine including path name

nSubroutineId: Number of the subroutine

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4 Blocks for compatibility with existing programs

Note

Function blocks for compatibility

The purpose of the function blocks listed is to ensure compatibility with existing projects. It is **not** advisable to use these function blocks for new projects. Instead, the equivalent function blocks shown in the table above should be used.

Function Block	Description
ItpDelDtg [▶ 251]	Triggers “Delete Distance to go” in the NC
ItpEStop [▶ 251]	Triggers the NCI EStop
ItpGetBottleNeckLookAhead [▶ 252]	Provides the value of the look-ahead for bottleneck detection
ItpGetBottleNeckMode [▶ 253]	Provides the response mode for bottleneck detection
ItpGetGeoInfoAndHParam [▶ 254]	Reads information of the currently active segment and past and future segments.
ItpGoAhead [▶ 255]	Triggers the GoAhead function
ItpIsEStop [▶ 256]	Determines whether an EStop is executed or pending
ItpLoadProg [▶ 257]	Loads an NC program using program names
ItpReadRParams [▶ 258]	Reads calculation parameters
ItpReadToolDesc [▶ 259]	Reads the tool description from the NC
ItpReadZeroShift [▶ 260]	Reads the zero shift from the NC
ItpReset [▶ 261]	Carries out a reset of the interpreter or of the NC channel
ItpResetEx [▶ 262]	Carries out a reset of the interpreter or of the NC channel.
ItpResetFastMFunc [▶ 263]	Resets a fast signal bit
ItpSetBottleNeckLookAhead [▶ 264]	Sets the value of the look-ahead for bottleneck detection
ItpSetBottleNeckMode [▶ 265]	Sets the response mode when bottleneck detection is switched on
ItpSetSubroutinePath [▶ 266]	Optionally sets the search path for subroutines
ItpSetToolDescNull [▶ 267]	Sets all tool parameters (including number and type) to zero
ItpSetZeroShiftNull [▶ 268]	Sets all origins to zero
ItpStartStop [▶ 269]	Starts or stops the interpreter (NC channel)
ItpStepOnAfterEStop [▶ 269]	Enables further processing of the parts program after an NCI EStop
ItpWriteRParams [▶ 270]	Writes calculation parameters
ItpWriteToolDesc [▶ 272]	Writes the tool description into the NC
ItpWriteZeroShift [▶ 273]	Writes the zero shift into the NC

5.1.4.1 ItpDelDtg

The ItpDelDtg function block triggers deletion of the remaining travel. There is a more detailed description in the Interpreter [▶ 140]documentation.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpDelDtgEx [▶ 191].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nChnId          : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nChnId: Channel ID

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.2 ItpEStop

The function block ItpEStop triggers the NCI EStop and enables a controlled stop on the path. The limit values for the deceleration and the jerk are transferred as parameters. If these are smaller than the currently active dynamic parameters, the transferred parameters are rejected.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpEStopEx [▸ 193].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nGrpId          : UDINT;
fDec            : LREAL;
fJerk           : LREAL;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nGrpId: group ID

fDec: Max. deceleration during stopping. If fDec is smaller than the currently active deceleration, fDec is rejected. This ensures that the deceleration occurs with the standard ramp as a minimum.

fJerk: Max. jerk during stopping. If fJerk is smaller than the currently active jerk, fJerk is rejected.

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

See also:

ItpStepOnAfterEStop [▸ 269]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.3 ItpGetBottleNeckLookAhead

The function block ItpGetBottleNeckLookAhead determines the maximum size of the look-ahead for the bottleneck detection (contour collision monitoring).

There is a more detailed description in the Interpreter [▸ 173] documentation.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpGetBottleNeckLookAheadEx [▶ 195].

VAR_INPUT

```
VAR_INPUT
bExecute       : BOOL;
nChnId         : UDINT;
tTimeOut       : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nChnId: Channel ID

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy          : BOOL;
bErr           : BOOL;
nErrId         : UDINT;
nLookAhead     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. If the function block has a timeout error, ‘Error’ is TRUE and ‘nErrId’ is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

nLookAhead: Value of the look-ahead for bottleneck detection

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.4 ItpGetBottleNeckMode

The function block ItpGetBottleNeckMode reads the behavior in the event of a contour collision (bottleneck).

There is a more detailed description in the Interpreter [▶ 173] documentation.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpGetBottleNeckModeEx [▶ 195].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nChnId          : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nChnId: Channel ID

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy           : BOOL;
bErr            : BOOL;
nErrId          : UDINT;
eBottleNeckMode: E_ItpBnMode
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. If the function block has a timeout error, ‘Error’ is TRUE and ‘nErrId’ is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

eBottleNeckMode: Enum for the behavior in the event of a contour collision

```
TYPE E_ItpBnMode:
(
ItpBnm_Abort   := 0,
ItpBnm_Adjust  := 1,
ItpBnm_Leave   := 2
);
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.5 ItpGetGeoInfoAndHParam

The function block ItpGetGeoInfoAndHParam reads information of the currently active segment and past and future segments. These include block number, H-parameter and residual path length on the segment.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpGetGeoInfoAndHParamEx [▶ 200].

VAR_IN_OUT

```
VAR_IN_OUT
sNciToPlc            : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: The structure of the cyclic channel interface from the NCI to the PLC. This structure is only accessed for reading. (type: NCTOPLC_NCICHANNEL_REF [▶ 306])

VAR_OUTPUT

```
VAR_OUTPUT
stTab                   : ST_ItpPreViewTabEx;
nErrId                  : UDINT;
END_VAR
```

stTab: Structure containing the segment data.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.6 ItpGoAhead

The function block ItpGoAhead may only be used in association with the decoder stop '@717' [▶ 150]. There is a more detailed description of this decoder stop in the interpreter documentation [▶ 107].

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpGoAheadEx [▶ 207].

VAR_INPUT

```
VAR_INPUT
bExecute         : BOOL;
nChnId           : UDINT;
tTimeOut         : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nChnId: Channel ID

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.7 ItpIsEStop

Via bEStop, the function block ItpIsEStop provides information as to whether an EStop command was triggered. If bEStop is TRUE, then an EStop was initiated (e.g. ItpEStop). The flag does **not** provide information as to whether the axes have already stopped or are still on the braking ramp.

After the execution of ItpStepOnAfterEStop, ItpIsEStop will once again return FALSE.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpIsEStopEx [▶ 209].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nGrpId          : UDINT
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nGrpId: group ID

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy           : BOOL;
bEStop          : BOOL;
bErr            : BOOL;
nErrId          : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bEStop: TRUE: EStop command was executed, FALSE: No EStop present

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

See also:

ItpEStop [▸ 251]

ItpStepOnAfterEStop [▸ 269]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.8 ItpLoadProg

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpLoadProgEx [▸ 210].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nChnId          : UDINT;
sPrg            : STRING;
nLength         : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: A rising edge at this input triggers execution of the NC program

nChnId: Channel ID

sPrg: Name of the NC program that is executed

nLength: String length of the program name

tTimeOut: ADS Timeout-Delay

NOTE! The NC program is looked up in directory "TwinCAT\Mc\Nci", if no further information is available. It is however also possible to give an absolute path.

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

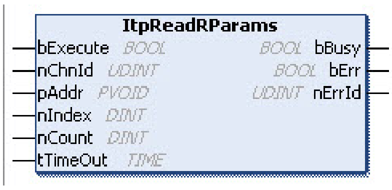
bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.9 ItpReadRParams



Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpReadRParamsEx [▸ 213].

The ItpReadRParams function block reads the NC's calculation parameters, also known as R-parameters. A more detailed description of the calculation parameters can be found here [▸ 115]. A total of 1000 R-parameters are available, of which the first 900 (0..899) are local, so that they are only visible in the current NC channel. The other 100 (900..999) R-parameters are global, and are thus visible from anywhere in the NC.

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nChnId          : UDINT;
pAddr           : PVOID;
nIndex          : DINT;
nCount          : DINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: A rising edge starts the read operation

nChnId: ID of the NC channel whose R-parameters are to be read

pAddr: Address of the target variables of the data to be read. The data are written by the NC directly from the specified address. i.e. nIndex is not to be interpreted as offset from pAddr. The data are usually in an array of type LREAL, which has to be defined by the user.

nIndex: Describes the index of the R-parameter to be read from an NC perspective.

nCount: Number of R-parameters to be read

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

See also:

ItpWriteRParams [▶ 270]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.10 ItpReadToolDesc

The ItpReadToolDesc function block reads the tool parameters for the supplied D-word.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpReadToolDescEx [▶ 214].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nChnId          : UDINT;
nDNo            : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nChnId: Channel ID

nDNo: D-word for which the tool parameters are to be read. nDoNo can have values between 1 and 255.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sToolDesc       : ToolDesc;
END_VAR
```

sToolDesc: A structure into which the tool parameters of nDNo are written. The meaning of the parameters depends on the tool type, and can be found in the tool data [▶ 162].

```
TYPE ToolDesc:
STRUCT
nToolNumber     : UDINT; (*valid range from 0 .. 65535*)
nToolType       : UDINT;
fParam          : ARRAY [2..15] OF LREAL;
END_STRUCT
END_TYPE
```

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

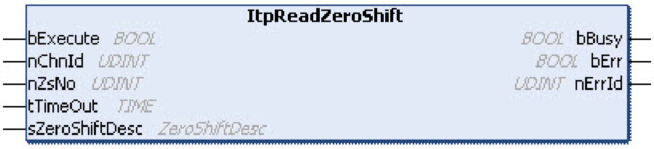
see also:

ItpWriteToolDesc [▸ 272]; ItpSetToolDescNull [▸ 267]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.11 ItpReadZeroShift



The **ItpReadZeroShift** function block reads the offset shift components X, Y and Z for the given zero shift.

NOTE! For reasons of compatibility, there are two entries (coarse and fine) for each axis in each zero shift (e.g. G54). These two entries must be added together. This function block evaluates both the entries and adds them together automatically.

Note

Outdated version

The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpReadZeroShiftEx [▸ 215].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nChnId          : UDINT;
nZsNo           : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nChnId: Channel ID

nZsNo: Number of the zero shift; on the NC side G54 to G59 are zero shifts. The valid range of values for 'nZsNo' is therefore from 54 to 59.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sZeroShiftDesc : ZeroShiftDesc;
END_VAR
```

sZeroShiftDesc: The structure containing the components of the zero shift.

```
TYPE ZeroShiftDesc:
STRUCT
fShiftX         : LREAL;
fShiftY         : LREAL;
fShiftZ         : LREAL;
END_STRUCT
END_TYPE
```

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

see also:

ItpWriteZeroShift [▸ 273]; ItpSetZeroShiftNull [▸ 268]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.12 ItpReset

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpResetEx2 [▸ 216].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nChnId          : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: A rising edge at this input triggers a reset of the NC channel

nChnId: Channel ID

tTimeOut: ADS Timeout-Delay

NOTE! A reset deletes all tables in the NC. The axes are halted immediately. For this reason a reset should only be carried out either in the event of an error or when the axes are stationary.

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.13 ItpResetEx

The function block 'ItpResetEx' executes a channel reset, which deletes all existing tables of the NC channel. In contrast to the conventional ItpReset [▶ 261], an active channel is stopped first, before the reset is executed. This simplifies programming in the PLC, since no explicit check is necessary to ascertain whether the axes are still in motion.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpResetEx2 [▶ 216].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nGrpId          : UDINT;
nChnId          : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nGrpId: group ID

nChnId: Channel ID

tTimeOut: ADS timeout delay (the bBusy signal can be active for longer than tTimeOut)

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

see also: ItpStartStop [▸ 269]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.14 ItpResetFastMFunc

This function block represents an alternative to Auto-reset or reset with another M-function (reset list during parameterization of the M-function). For the sake of clarity, mixed operation involving resetting with an M-function and this function block should be avoided.

The fast M-function [▸ 145] **nMFuncNo** is reset with a rising edge at input **bExecute**. In the event of the M-function not being available, **no** error is returned.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpResetFastMFuncEx [▸ 217].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nChnId          : UDINT;
nMFuncNo        : UINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nChnId: Channel ID

nMFuncNo: Flying M-function that is to be reset

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.15 ItpSetBottleNeckLookAhead

The function block ItpSetBottleNeckLookAhead determines the maximum number of segments the system may look ahead for bottleneck detection (contour collision monitoring). Note that segments, which were added as a result of radius compensation (e.g. additional segments at acute angles) are taken into account.

There is a more detailed description in the Interpreter [▶ 173] documentation.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpSetBottleNeckLookAheadEx [▶ 218].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nChnId          : UDINT;
nLookAhead      : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nChnId: Channel ID

nLookAhead: Specifies the look-ahead value

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.16 ItpSetBottleNeckMode

The function block ItpSetBottleNeckMode specifies the behavior in the event of a contour collision (bottleneck).

There is a more detailed description in the Interpreter [▶ 173] documentation.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpSetBottleNeckModeEx [▶ 219].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nChnId          : UDINT;
eBottleNeckMode: E_ItpBnMode
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nChnId: Channel ID

eBottleNeckMode: Enum for the behavior in the event of a contour collision

tTimeOut: ADS Timeout-Delay

```
TYPE E_ItpBnMode:
(
ItpBnm_Abort    := 0,
ItpBnm_Adjust   := 1,
ItpBnm_Leave    := 2
);
END_TYPE
```

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

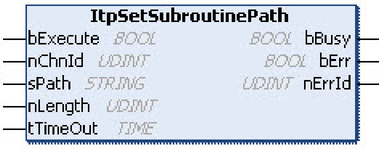
bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.17 ItpSetSubroutinePath



With ItpSetSubroutinePath function block, the search path for subroutines can optionally be set.

If a subroutine still has to be integrated, the file is searched in the following order:

- optional search path (ItpSetSubroutinePath)
- path from which the main program was loaded
- TwinCAT\Mc\Nci directory

Only one optional path can be active at any one time. It remains active until it is

- overwritten with another path or
- with an empty string

.

After a TwinCAT restart, the path has to be re-assigned.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpSetSubroutinePathEx [▸ 223].

Interface

```
VAR_INPUT
bExecute        : BOOL;
nChnId          : UDINT;
sPath           : STRING;
nLength         : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nChnId: Channel ID

sPath: Optional path for subroutines; is disabled with an empty string.

nLength: String length

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.18 ItpSetToolDescNull

FB ItpSetToolDescNull overwrites all tool parameters (incl. number & type) of the channel with zero.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpSetToolDescNullEx [▸ 224].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nChnId          : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: A rising edge results in overwriting of all tool parameters of the NC channel with zero.

nChnId: ID of the NC channel

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

See also:

- ItpWriteToolDesc [▶ 272],
- ItpReadToolDesc [▶ 259]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.19 ItpSetZeroShiftNull

FB ItpSetZeroShiftNull overwrites all zero shifts of the channel with zero.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpSetZeroShiftNullEx [▶ 225].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nChnId          : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: A rising edge results in overwriting of all zero shifts of the NC channel with zero.

nChnId: ID of the NC channel

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

See also:

- ItpWriteZeroShift [▶ 273]
- ItpReadZeroShift [▶ 260]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.20 ItpStartStop

Note	**Outdated version**
	The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpStartStopEx [▸ 227].

Interface

```
VAR_INPUT
bStart          : BOOL;
bStop           : BOOL;
nChnId          : UDINT;
tTimeOut        : TIME;
END_VAR
```

bStart: A positive edge starts the NC channel

bStop: A positive edge stops the NC channel. A stop command deletes all the tables in the NC and brings the axes to a controlled halt.

nChnId: Channel ID

tTimeOut: ADS Timeout-Delay

NOTE! The bStop input has a higher priority than the bStart input, so that if both inputs receive a positive edge, a channel stop will be executed.

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.21 ItpStepOnAfterEStop

The function block ItpStepOnAfterEStop enables further processing of the parts program after a programmed EStop.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpStepOnAfterEStopEx [▸ 228].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nGrpId          : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nGrpId: group ID

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▸ 308] or in the NC error documentation [▸ 313] (error codes above 0x4000).

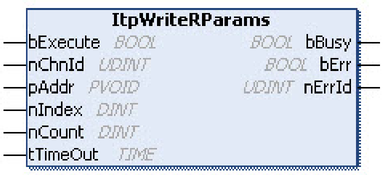
See also:

- ItpEStop [▸ 251]
- ItpIsEStop [▸ 256]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.22 ItpWriteRParams



The ItpWriteRParams function block writes R-parameters into the NC.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpWriteRParamsEx [▸ 229].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nChnId          : UDINT;
pAddr           : PVOID;
nIndex          : DINT;
nCount          : DINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: A rising edge starts the write operation.

nChnId: ID of the NC channel whose R-parameters are to be written.

pAddr: Address of the variables containing the data to be written. Data are used directly from the specified address, i.e. nIndex is not to be interpreted as offset from pAddr. The data are usually in an array of type LREAL, which has to be defined by the user.

nIndex: Describes the index of the R-parameter to be written from an NC perspective.

nCount: Number of R-parameters to be written

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrId'. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

Sample

In this example the parameters R90 to R99 are written from an NC perspective.

```
VAR
arrfRParam90to99 : ARRAY[0..9] OF LREAL;
fbWriteRParam    : ItpWriteRParams;
n                : INT := 0;
bWriteParam      : BOOL := FALSE;
END_VAR
```

```
FOR n:=0 TO 9 DO
arrfRParam90to99[n] := 90 + n;
END_FOR

fbWriteRParam(
 bExecute := bWriteParam,
 nChnId := 2,
 pAddr := ADR( arrfRParam90to99[0] ),
 nIndex := 90,
 nCount := 10,
 tTimeOut := T#200ms );
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.23 ItpWriteToolDesc

The **ItpWriteToolDesc** function block writes a block of tool parameters.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpWriteToolDescEx [▶ 230].

VAR_INPUT

```
VAR_INPUT
bExecute        : BOOL;
nChnId          : UDINT;
nDNo            : UDINT;
tTimeOut        : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nChnId: Channel ID

nDNo: D-word for which the tool parameters are to be read. nDoNo can have values between 1 and 255.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sToolDesc       : ToolDesc;
END_VAR
```

sToolDesc: The structure that contains the new tool parameters. This structure is only accessed for reading. The meaning of the parameters depends on the tool type, and can be found in the tool data [▶ 162].

```
TYPE ToolDesc:
STRUCT
nToolNumber     : UDINT; (*valid range from 0 .. 65535*)
nToolType       : UDINT;
fParam          : ARRAY [2..15] OF LREAL;
END_STRUCT
END_TYPE
```

VAR_OUTPUT

```
VAR_OUTPUT
bBusy     : BOOL;
bErr      : BOOL;
nErrId    : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

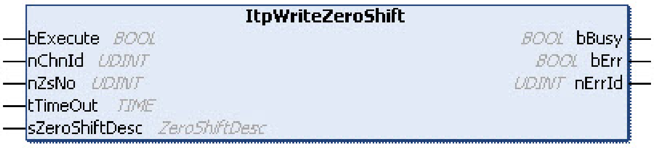
See also:

- ItpReadToolDesc [▸ 259]
- ItpSetToolDescNull [▸ 267]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.4.24 ItpWriteZeroShift



The function block **ItpWriteZeroShift** writes the shift components X, Y and Z for the specified zero shift.

For reasons of compatibility every zero shift that can be set has two parameters (coarse and fine) for each axis. When using this function block to write a new zero shift, the new value is written into the 'fine parameter'. A value of 0.0 is entered into the 'coarse parameter'. This makes it possible to use a function block such as ItpReadZeroShift [▸ 260] to read and modify a zero shift and to send it back to the NC.

Note	**Outdated version** The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use the function block ItpWriteZeroShiftEx [▸ 231].

VAR_INPUT

```
VAR_INPUT
bExecute       : BOOL;
nChnId         : UDINT;
nZsNo          : UDINT;
tTimeOut       : TIME;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

nChnId: Channel ID

nZsNo: Number of the zero shift.

On the NC side G54 to G59 are zero shifts; G58 and G59 can only be edited from the NC program. The valid range of values for 'nZsNo' is therefore from 54 to 57.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
sZeroShiftDesc : ZeroShiftDesc;
END_VAR
```

sZeroShiftDesc: The structure containing the components of the zero shift. This structure is only accessed for reading.

```
TYPE ZeroShiftDesc:
STRUCT
fShiftX : LREAL;
fShiftY : LREAL;
fShiftZ : LREAL;
END_STRUCT
END_TYPE
```

VAR_OUTPUT

```
VAR_OUTPUT
bBusy      : BOOL;
bErr       : BOOL;
nErrId     : UDINT;
END_VAR
```

bBusy: This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'Timeout' input. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrId’. Is reset to FALSE by the execution of a command at the inputs.

nErrId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

See also:

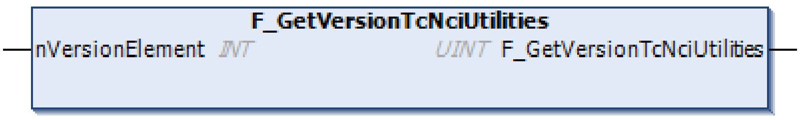
- ItpReadZeroShift [▶ 260]
- ItpSetZeroShiftNull [▶ 273]

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.5 Obsolete

5.1.5.1 F_GetVersionTcNciUtilities



This function returns part of the three-part version number of the TwinCAT 2 PLC library TcNciUtilities.lib as UINT.

Note	**Outdated version** The sole purpose of this function is to ensure compatibility with existing projects. For new projects please use the global structure stLibVersion_Tc2_NCI.

VAR_INPUT

```
FUNCTION F_GetVersionNciUtilities

VAR_INPUT
nVersionElement       : UINT;
END_VAR
```

nVersionElement: Part of the version number to be read (range: [1..3])

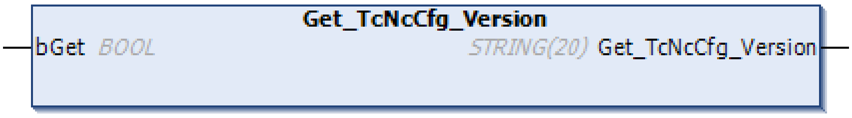
Return value

F_GetVersionNciUtilities: Version number

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.5.2 Get_TcNcCfg_Version



This function returns the version number of the TwinCAT 2 PLC library TcNcCfg.lib as string.

Note

Outdated version

The sole purpose of this function is to ensure compatibility with existing projects. For new projects please use the global structure stLibVersion_Tc2_NCI.

Return value

Get_TcNcCfg_Version Version number

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.1.5.3 ItpGetVersion

ItpGetVersion is a function that returns the version number of the TwinCAT PLC library TcNC.lib as string.

Note

Outdated version

The sole purpose of this function is to ensure compatibility with existing projects. For new projects please use the global structure stLibVersion_Tc2_NCI.

VAR_INPUT

```
FUNCTION ItpGetVersion

VAR_INPUT
END_VAR
```

Return value

ItpGetVersion: Version number

Sample

```
VAR
strVersion: STRING(20);
END_VAR
```

```
strVersion := ItpGetVersion();
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_NCI

5.2 PLC Library: Tc2_PlcInterpolation

The TcPlcInterpolation library offers an alternative to the application of G-Code (DIN 66025). This library can be used to execute interpolated movement commands directly from the PLC, without using G-Code.

In a first step a table of different movement commands and additional functions is written. To this end structures such as ST_NciGeoLine are transferred to the FB NciFeedTablePreparation. This appends the movement command to the table. Once the table is full or all required entries have been added, NciFeedTable is called in order to transfer the table content to the NC kernel. The data transfer directly starts the execution.

Function blocks that are required for grouping of axes (or for channel control (channel override) can be found in the PLC Library: Tc2_NCI [▶ 181].

Function Block	Description
FB_NciFeedTablePreparation [▶ 278]	Fills a table with NCI movements in the PLC
FB_NciFeedTable [▶ 279]	Transfers a previously written table to the NC kernel and starts the motion

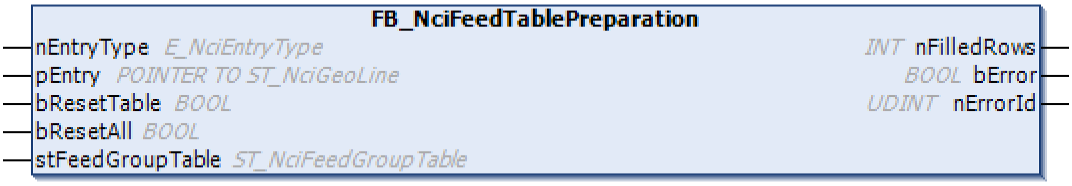
The following structures can be used as input parameters for the function block NciFeedTablePreparation:

Structures	Enum	Description
	Organization	
	E_NciEntryTypeNone	No function
ST_NciGeoStart [▸ 281]	E_NciEntryTypeGeoStart	Sets the start position for the first geometry entry
ST_NciEndOfTables [▸ 292]	E_NciEntryTypeEndOfTables	Indicates the end of the geometry table
	Movement commands	
ST_NciGeoLine [▸ 282]	E_NciEntryTypeGeoLine	Describes a straight line
ST_NciGeoCirclePlane [▸ 282]	E_NciEntryTypeGeoCirclePlane	Describes a circle in the main plane (center point programming)
ST_NciGeoCircleCIP [▸ 284]	E_NciEntryTypeGeoCircleCIP	Describes a circle anywhere in the space
ST_NciGeoBezier3 [▸ 284]	E_NciEntryTypeGeoBezier3	Describes a 3rd order Bezier with control points
ST_NciGeoBezier5 [▸ 285]	E_NciEntryTypeGeoBezier5	Describes a 5th order Bezier with control points
ST_NciDwellTime [▸ 290]	E_NciEntryTypeDwellTime	Describes a dwell time
	Path parameters	
ST_NciBaseFrame [▸ 289]	E_NciEntryTypeBaseFrame	Describes a zero shift and rotation
ST_NciVertexSmoothing [▸ 288]	E_NciEntryTypeVertexSmoothing	Activates blending at segment transitions
ST_NciTangentialFollowingDesc [▸ 291]	E_NciEntryTypeTfDesc	Activates tangential following of the tool
	Dynamics	
ST_NciDynOvr [▸ 288]	E_NciEntryTypeDynOvr	Modifies the dynamic override
ST_NciAxisDynamics [▸ 290]	E_NciEntryTypeAxisDynamics	Limits the axis dynamics
ST_NciPathDynamics [▸ 289]	E_NciEntryTypePathDynamics	Limits the path dynamics
ST_NciFeedrateIpol [▸ 290]	E_NciEntryTypeFeedrateIpol	Sets the feed interpolation type
	Parameter commands	
ST_NciHParam [▸ 287]	E_NciEntryTypeHParam	Sets an H-parameter (DINT)
ST_NciSParam [▸ 288]	E_NciEntryTypeSParam	Sets an S-parameter (WORD)
ST_NciTParam [▸ 288]	E_NciEntryTypeTParam	Sets a T-parameter (WORD)
ST_NciMFuncFast [▸ 286]	E_NciEntryTypeMFuncFast	Parameterizes a fast M-function (no handshake)
ST_NciMFuncHsk [▸ 286]	E_NciEntryTypeMFuncHsk	Parameterizes an M-function with handshake
ST_NciMFuncResetAllFast [▸ 287]	E_NciEntryTypeResetAllFast	Resets all fast M-functions

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_PlcInterpolation

5.2.1 FB_NciFeedTablePreparation



The function block **FB_NciFeedTablePreparation** appends an entry of a specific type to the feed table (stFeedGropupTable). An appended entry can generate more than one row in the table. If the table has not enough free rows, an error is returned and no entry is added to the table. In this case the entry either has to be added to another table or to the same table, after FB_NciFeedTable was executed. This function block deals with modal functions, such as tangential following. It is therefore important to always use the same instance of this function block. The function block can be called repeatedly in a PLC cycle.

VAR_INPUT

```
VAR_INPUT
nEntryType        : E_NciEntryType;
pEntry            : POINTER TO ST_NciGeoLine;
bResetTable       : BOOL;
bResetAll         : BOOL;
END_VAR
```

nEntryType: Specifies the entry type, e.g. line, circle, tangential following

pEntry: Pointer to entry structure – must match nEntryType

bResetTable: If bResetTable is TRUE, the table ‚stFeedGroupTable' is set to zero and nFilledRows is also set to zero. If nErrorId = ErrNciFeedTableFull, this error is reset. All modal flags (such as tangential following) remain constant.

bResetAll: Like bResetTable. In addition, all modal flags are set to their default values, and all error IDs are reset.

VAR_IN_OUT

```
VAR_IN_OUT
stFeedGroupTable : ST_NciFeedGroupTable
END_VAR
```

stFeedGroupTable: Table containing the rows for the NC kernel.

VAR_OUTPUT

```
VAR_OUTPUT
nFilledRows       : INT;
bError            : BOOL;
nErrorId          : UDINT;
END_VAR
```

nFilledRows: Number of filled rows.

bError: Becomes TRUE as soon as an error has occurred.

nErrorId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313] (error codes above 0x4000).

NOTE! If bResetTable, bResetAll, or bError is true, no further entries are accepted.

NOTE! The error code 0x4B72 indicates that the table is full and the last entry was not accepted.

Sample:

```
stGeoLine.nDisplayIndex := 1;
stGeoLine.fEndPosX := 0;
stGeoLine.fEndPosY := 400;
stGeoLine.fEndPosZ := 100;
```

```
stGeoLine.fEndPosQ1 :=-90;
stGeoLine.fVelo := 1000; (*mm per sec*)

fbFeedTablePrep(
nEntryType := E_NciEntryTypeGeoLine,
pEntry := ADR(stGeoLine),
bResetTable:= FALSE,
stFeedGroupTable:= stNciFeedGroupTable,
nFilledRows=> nFilledRows,
bError => bError,
nErrorId => nErrorId);
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_PlcInterpolation

5.2.2 FB_NciFeedTable

The function block **FB_NciFeedTable** transfers a given table to the NC kernel. If the override is set and the approvals are enabled, execution is started immediately. bFeedingDone becomes TRUE when the transfer is complete. This signal can be used for overwriting the table with NciFeedTablePreparation [▶ 278]. In NciFeedTablePreparation the table first has to be reset.

bChannelDone indicates complete execution of the tables in the NC kernel. The identifier ST_NciEndOfTables [▶ 280] must therefore be placed at the end of the last table.

VAR_INPUT

```
VAR_INPUT
bExecute           : BOOL;
bReset             : BOOL;
bLogFeederEntries  : BOOL;
END_VAR
```

bExecute: The command is triggered by a rising edge at this input.

bReset: Triggers a channel reset and also resets the function block

bLogFeederEntries: If TRUE, a log file 'PlcItpFeed.log' is written in the TwinCAT\Mc\Nci folder. It contains all entries that are sent to the NC kernel via ADS. If bLogFeederEntries = TRUE, more time is required until bFeedingDone becomes TRUE.

VAR_IN_OUT

```
VAR_IN_OUT
stFeedGroupTable   : ST_NciFeedGroupTable;
stNciToPlc         : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

stFeedGroupTable: Table containing the rows for the NC kernel.

stNciToPlc: The structure of the cyclic channel interface between NCI and PLC.

VAR_OUTPUT

```
VAR_OUTPUT
bFeedingDone       : BOOL;
bChannelDone       : BOOL;
bFeedBusy:         : BOOL;
bResetBusy:        : BOOL;
```

```
bError              : BOOL;
nErrorId            : UDINT;
END_VAR
```

bFeedingDone: Becomes TRUE once all table rows have been sent to the NC kernel.

bChannelDone: Becomes TRUE once all entries of the table in the NC kernel were executed and ST_NciEndOfTables was detected.

bFeedBusy: Becomes TRUE when the function block sends entries to the NC kernel.

bResetBusy: Becomes TRUE when a reset is executed.

bError: Becomes TRUE as soon as an error has occurred.

nErrorId: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs. The error numbers in ErrId can be looked up in the ADS error documentation [▶ 308] or in the NC error documentation [▶ 313].

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v3.1.0	PC or CX (x86 or x64)	Tc2_PlcInterpolation

5.2.3 Types and Enums

E_NciEntryType

```
TYPE E_NciEntryType :
(
    E_NciEntryTypeNone := 0,
    E_NciEntryTypeGeoStart := 1,
    E_NciEntryTypeGeoLine := 2,
    E_NciEntryTypeGeoCirclePlane := 3,
    E_NciEntryTypeGeoCircleCIP := 4,
    E_NciEntryTypeGeoBezier3 := 10,
    E_NciEntryTypeGeoBezier5 := 11,
    E_NciEntryTypeMFuncHsk := 20,
    E_NciEntryTypeMFuncFast := 21,
    E_NciEntryTypeMFuncResetAllFast := 23,
    E_NciEntryTypeHParam := 24,
    E_NciEntryTypeSParam := 25,
    E_NciEntryTypeTParam := 26,
    E_NciEntryTypeDynOvr := 50,
    E_NciEntryTypeVertexSmoothing := 51,
    E_NciEntryTypeBaseFrame := 52,
    E_NciEntryTypePathDynamics := 53,
    E_NciEntryTypeAxisDynamics := 55,
    E_NciEntryTypeDwellTime := 56,
    E_NciEntryTypeFeedrateIpol := 57,
    E_NciEntryTypeTfDesc := 100,
    E_NciEntryTypeEndOfTables := 1000
);
END_TYPE
```

Structures	Enum	Description
	Organization	
	E_NciEntryTypeNone	No function
ST_NciGeoStart [▸ 281]	E_NciEntryTypeGeoStart	Sets the start position for the first geometry entry
ST_NciEndOfTables [▸ 292]	E_NciEntryTypeEndOfTables	Indicates the end of the geometry table
	Movement commands	
ST_NciGeoLine [▸ 282]	E_NciEntryTypeGeoLine	Describes a straight line
ST_NciGeoCirclePlane [▸ 282]	E_NciEntryTypeGeoCirclePlane	Describes a circle in the main plane (center point programming)
ST_NciGeoCircleCIP [▸ 284]	E_NciEntryTypeGeoCircleCIP	Describes a circle anywhere in the space
ST_NciGeoBezier3 [▸ 284]	E_NciEntryTypeGeoBezier3	Describes a 3rd order Bezier with control points
ST_NciGeoBezier5 [▸ 285]	E_NciEntryTypeGeoBezier5	Describes a 5th order Bezier with control points
ST_NciDwellTime [▸ 290]	E_NciEntryTypeDwellTime	Describes a dwell time
	Path parameters	
ST_NciBaseFrame [▸ 289]	E_NciEntryTypeBaseFrame	Describes a zero shift and rotation
ST_NciVertexSmoothing [▸ 288]	E_NciEntryTypeVertexSmoothing	Activates blending at segment transitions
ST_NciTangentialFollowingDesc [▸ 291]	E_NciEntryTypeTfDesc	Activates tangential following of the tool
	Dynamics	
ST_NciDynOvr [▸ 288]	E_NciEntryTypeDynOvr	Modifies the dynamic override
ST_NciAxisDynamics [▸ 290]	E_NciEntryTypeAxisDynamics	Limits the axis dynamics
ST_NciPathDynamics [▸ 289]	E_NciEntryTypePathDynamics	Limits the path dynamics
ST_NciFeedrateIpol [▸ 290]	E_NciEntryTypeFeedrateIpol	Sets the feed interpolation type
	Parameter commands	
ST_NciHParam [▸ 287]	E_NciEntryTypeHParam	Sets an H-parameter (DINT)
ST_NciSParam [▸ 288]	E_NciEntryTypeSParam	Sets an S-parameter (WORD)
ST_NciTParam [▸ 288]	E_NciEntryTypeTParam	Sets a T-parameter (WORD)
ST_NciMFuncFast [▸ 286]	E_NciEntryTypeMFuncFast	Parameterizes a fast M-function (no handshake)
ST_NciMFuncHsk [▸ 286]	E_NciEntryTypeMFuncHsk	Parameterizes an M-function with handshake
ST_NciMFuncResetAllFast [▸ 287]	E_NciEntryTypeResetAllFast	Resets all fast M-functions

ST_NciGeoStart

Sets the start position for the first geometry entry. This is necessary, if the first geometry entry is a circle or if tangential following in the first segment is ON. This structure can optionally be written at each start of the first table.

```
TYPE ST_NciGeoStart :
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeGeoStart; (*do not override this parameter *)
fPosX: LREAL;
fPosY: LREAL;
fPosZ: LREAL;
fPosQ1: LREAL;
fPosQ2: LREAL;
fPosQ3: LREAL;
fPosQ4: LREAL;
```

```
fPosQ5: LREAL;
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▸ 280])

fPosX: Start position X

fPosY: Start position Y

fPosZ: Start position Z

fPosQ1: Start position Q1

fPosQ2: Start position Q2

fPosQ3: Start position Q3

fPosQ4: Start position Q4

fPosQ5: Start position Q5

ST_NciGeoLine

Describes a straight line with specified velocity.

```
TYPE ST_NciGeoLine :
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeGeoLine; (*do not override this parameter *)
nDisplayIndex: UDINT;
fEndPosX: LREAL;
fEndPosY: LREAL;
fEndPosZ: LREAL;
fEndPosQ1: LREAL;
fEndPosQ2: LREAL;
fEndPosQ3: LREAL;
fEndPosQ4: LREAL;
fEndPosQ5: LREAL;
fVelo: LREAL;
bRapidTraverse: BOOL;
bAccurateStop: BOOL; (* VeloEnd := 0 *)
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▸ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

fEndPosX: Target position X

fEndPosY: Target position Y

fEndPosZ: Target position Z

fEndPosQ1: Target position Q1

fEndPosQ2: Target position Q2

fEndPosQ3: Target position Q3

fEndPosQ4: Target position Q4

fEndPosQ5: Target position Q5

fVelo: Target path velocity, like F in G-Code, but in basic units per second (e.g. mm/s)

bRapidTraverse: TRUE has the same effect as G0, FALSE treats this entry like G01

bAccurateStop: Accurate stop (TRUE has the same effect as G09)

ST_NciGeoCirclePlane

Describes a circle in the main plane. The center point is specified in absolute coordinates.

The orthogonal component at the center is assigned internally. If a circle is programmed in the XY plane, for example, ‚fCenterZ‘ is assigned internally. If the user has assigned the value explicitly, the value is nevertheless overwritten by the function block. A helix can be described by programming the height. If helix is programmed in the XY plane, for example, the lifting height of the helix is specified absolutely with ‚fEndPosZ‘.

```
TYPE ST_NciGeoCirclePlane :
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeGeoCirclePlane; (*do not override this parameter *)
nDisplayIndex: UDINT;
fEndPosX: LREAL;
fEndPosY: LREAL;
fEndPosZ: LREAL;
fCenterX: LREAL;
fCenterY: LREAL;
fCenterZ: LREAL;
fEndPosQ1: LREAL;
fEndPosQ2: LREAL;
fEndPosQ3: LREAL;
fEndPosQ4: LREAL;
fEndPosQ5: LREAL;
fVelo: LREAL;
bClockwise: BOOL;
bAccurateStop: BOOL; (* VeloEnd := 0 *)
nPlane: E_NciGeoPlane := E_NciGeoPlaneXY;
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▶ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

fEndPosX: Target position X

fEndPosY: Target position Y

fEndPosZ: Target position Z

fCenterX: Centre position X in absolute coordinates

fCenterY: Centre position Y in absolute coordinates

fCenterZ: Centre position Z in absolute coordinates

fEndPosQ1: Target position Q1

fEndPosQ2: Target position Q2

fEndPosQ3: Target position Q3

fEndPosQ4: Target position Q4

fEndPosQ5: Target position Q5

fVelo: Target path velocity in basic units per second (e.g. mm/s), like F in G-Code

bClockwise: If TRUE, the circle is drawn clockwise, otherwise counter-clockwise (similar to G02, G03)

bAccurateStop: accurate stop [▶ 123] (TRUE has the same effect as G09)

nPlane: Specifies the plane: XY, YZ, or ZX (similar to G17..G19) (type: E_NciGeoPlane [▶ 283])

Note	**Circle segment as start segment** If the first geometry segment is a circle, the start position must set with ST_NciGeoStart [▶ 281].

E_NciGeoPlane

```
TYPE E_NciGeoPlane :
(
E_NciGeoPlaneXY := 17,
E_NciGeoPlaneZX := 18,
```

```
E_NciGeoPlaneYZ := 19
);
END_TYPE
```

ST_NciGeoCircleCIP

The CIP circle can be used to describe a circle anywhere in space. It does not have to be in the main plane. In order for the circle to be described unambiguously, not all 3 points (the starting point is specified implicitly) may lie on straight line. It is thus not possible to program a full circle in this way.

```
TYPE ST_NciGeoCircleCIP :
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeGeoCircleCIP; (* do not overwrite this parameter *)
nDisplayIndex: UDINT;
fEndPosX:      LREAL;
fEndPosY:      LREAL;
fEndPosZ:      LREAL;
fCIPPosX:      LREAL;
fCIPPosY:      LREAL;
fCIPPosZ:      LREAL;
fEndPosQ1:     LREAL;
fEndPosQ2:     LREAL;
fEndPosQ3:     LREAL;
fEndPosQ4:     LREAL;
fEndPosQ5:     LREAL;
fVelo:         LREAL;
bAccurateStop: BOOL; (* VeloEnd := 0 *)
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▸ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

fCIPPosX: X position in absolute coordinates (point on circular path)

fCIPPosY: Y position in absolute coordinates (point on circular path)

fCIPPosZ: Z position in absolute coordinates (point on circular path)

fEndPosX: Target position X

fEndPosY: Target position Y

fEndPosZ: Target position Z

fEndPosQ1: Target position Q1

fEndPosQ2: Target position Q2

fEndPosQ3: Target position Q3

fEndPosQ4: Target position Q4

fEndPosQ5: Target position Q5

fVelo: Target path velocity in basic units per second (e.g. mm/s), like F in G-Code

bAccurateStop: accurate stop [▸ 123] (TRUE has the same effect as G09)

Note	**Circle segment as start segment** If the first geometry segment is a circle, the start position must set with ST_NciGeoStart [▸ 281].

ST_NciGeoBezier3

Describes a 3rd-order Bézier curve with the aid of control points. The start position results from the previous segment. The third control point is determined by the target position.

```
TYPE ST_NciGeoBezier3:
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeGeoBezier3; (*do not override this parameter *)
```

```
nDisplayIndex: UDINT;
fControlPoint1X: LREAL;
fControlPoint1Y: LREAL;
fControlPoint1Z: LREAL;
fControlPoint2X: LREAL;
fControlPoint2Y: LREAL;
fControlPoint2Z: LREAL;
fEndPosX: LREAL;
fEndPosY: LREAL;
fEndPosZ: LREAL;
fEndPosQ1: LREAL;
fEndPosQ2: LREAL;
fEndPosQ3: LREAL;
fEndPosQ4: LREAL;
fEndPosQ5: LREAL;
fVelo: LREAL;
bAccurateStop: BOOL; (* VeloEnd := 0 *)
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▸ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

fControlPoint1X: X component control point 1

fControlPoint1Y: Y component control point 1

...

fControlPoint2Z: Z component control point 2

fEndPosX: Target position X

fEndPosY: Target position Y

fEndPosZ: Target position Z

fEndPosQ1: Target position Q1

fEndPosQ2: Target position Q2

fEndPosQ3: Target position Q3

fEndPosQ4: Target position Q4

fEndPosQ5: Target position Q5

fVelo: Target path velocity in basic units per second (e.g. mm/s), like F in G-Code

bAccurateStop: accurate stop [▸ 123] (TRUE has the same effect as G09)

ST_NciGeoBezier5

Describes a 5th-order Bézier curve with the aid of control points. The start position results from the previous segment. The fifth control point is determined by the target position.

```
TYPE ST_NciGeoBezier5:
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeGeoBezier5; (*do not override this parameter *)
nDisplayIndex: UDINT;
fControlPoint1X: LREAL;
fControlPoint1Y: LREAL;
fControlPoint1Z: LREAL;
fControlPoint2X: LREAL;
fControlPoint2Y: LREAL;
fControlPoint2Z: LREAL;
fControlPoint3X: LREAL;
fControlPoint3Y: LREAL;
fControlPoint3Z: LREAL;
fControlPoint4X: LREAL;
fControlPoint4Y: LREAL;
fControlPoint4Z: LREAL;
fEndPosX: LREAL;
```

```
fEndPosY: LREAL;
fEndPosZ: LREAL;
fEndPosQ1: LREAL;
fEndPosQ2: LREAL;
fEndPosQ3: LREAL;
fEndPosQ4: LREAL;
fEndPosQ5: LREAL;
fVelo: LREAL;
bAccurateStop: BOOL; (* VeloEnd := 0 *)
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▶ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

fControlPoint1X: X component control point 1

fControlPoint1Y: Y component control point 1

...

fControlPoint4Z: Z component control point 4

fEndPosX: Target position X

fEndPosY: Target position Y

fEndPosZ: Target position Z

fEndPosQ1: Target position Q1

fEndPosQ2: Target position Q2

fEndPosQ3: Target position Q3

fEndPosQ4: Target position Q4

fEndPosQ5: Target position Q5

fVelo: Target path velocity in basic units per second (e.g. mm/s), like F in G-Code

bAccurateStop: accurate stop [▶ 123] (TRUE has the same effect as G09)

ST_NciMFuncHsk

Describes an M-function [▶ 145] of type handshake. The M-function number is between 0 and 159.

```
TYPE ST_NciMFuncHsk :
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeMFuncHsk; (*do not override this parameter *)
nDisplayIndex: UDINT;
nMFunc: INT;
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▶ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

nMFunc: M-function number (0..159)

> **Note**
>
> **M-functions in the PlcInterpolation library**
>
> If M-functions are used in the PlcInterpolation library, they do not have to be entered in the user interface of the XAE. An M-function always takes effect at the programmed location.

ST_NciMFuncFast

Parameterizes up to 8 fast M-functions [▶ 145]. The first M-function must be assigned nMFuncIn0, the second nMFuncIn1 etc. -1 indicates the end of the assignments.

```
TYPE ST_NciMFuncFast :
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeMFuncFast; (*do not override this parameter *)
nDisplayIndex: UDINT;
nMFuncIn0: INT;
nMFuncIn1: INT;
nMFuncIn2: INT;
nMFuncIn3: INT;
nMFuncIn4: INT;
nMFuncIn5: INT;
nMFuncIn6: INT;
nMFuncIn7: INT;
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▸ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

nMFuncIn0: fast M-function number (0..159)

nMFuncIn1: Fast M-function number (0..159); -1 indicates the end of the list.

nMFuncIn2: Fast M-function number (0..159); -1 indicates the end of the list.

nMFuncIn3: Fast M-function number (0..159); -1 indicates the end of the list.

nMFuncIn4: Fast M-function number (0..159); -1 indicates the end of the list.

nMFuncIn5: Fast M-function number (0..159); -1 indicates the end of the list.

nMFuncIn6: Fast M-function number (0..159); -1 indicates the end of the list.

nMFuncIn7: Fast M-function number (0..159); -1 indicates the end of the list.

Note	**M-functions in the PlcInterpolation library** If M-functions are used in the PlcInterpolation library, they do not have to be entered in the user interface of the XAE. An M-function always takes effect at the programmed location.

ST_NciMFuncResetAllFast

Resets all fast M-functions [▸ 145].

```
TYPE ST_NciMFuncResetAllFast :
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeMFuncResetAllFast; (*do not override this parameter *)
nDisplayIndex: UDINT;
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▸ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

ST_NciHParam

Sets an H-parameter [▸ 149] in the cyclic channel interface.

```
TYPE ST_NciHParam :
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeHParam; (*do not override this parameter *)
nDisplayIndex: UDINT;
nHParam: UDINT;
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▸ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

nHParam: H-parameter from NC to PLC

ST_NciSParam

Sets an S-parameter [▶ 149] in the cyclic channel interface.

```
TYPE ST_NciSParam :
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeSParam; (*do not override this parameter *)
nDisplayIndex: UDINT;
nSParam: UINT;
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▶ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

nSParam: S-parameter from NC to PLC

ST_NciTParam

Sets an T-parameter [▶ 149] in the cyclic channel interface.

```
TYPE ST_NciTParam :
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeTParam; (*do not override this parameter *)
nDisplayIndex: UDINT;
nTParam: UINT;
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▶ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

nTParam: T-parameter from NC to PLC

ST_NciDynOvr

Modal functions for changing the path dynamics.

See DynOvr [▶ 156] in the interpreter documentation [▶ 107].

```
TYPE ST_NciDynOvr :
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeDynOvr; (*do not override this parameter*)
nDisplayIndex: UDINT;
fDynOvr: LREAL;
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▶ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

fDynOvr: Value for dynamic override (1 < fDynOvr <= 1)

ST_NciVertexSmoothing

Modal function for activating blending at the segment transition. Blending is active until it is cancelled by setting the radius to 0.

A more detailed description of the parameter can be found in the interpreter documentation [▶ 107]. (paramVertexSmoothing [▶ 133]).

```
TYPE ST_NciVertexSmoothing :
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeVertexSmoothing; (*do not override this parameter *)
nDisplayIndex: UDINT;
nType: UDINT; (*type of smoothing, e.g. parabola, bi-quad *)
nSubtype: UDINT; (*e.g. adaptive, constant radius *)
fRadius: LREAL; (*max. radius for tolerance ball *)
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▸ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

nType: Blending type: 2: parabola, 3: Bi-quadratic, 4: Bezier 3rd order, 5: 5th order Bezier

nSubtype: 1: constant tolerance radius, 2: distance between intersection and vertex, 3: Adaptive tolerance radius

fRadius: Radius of the blending sphere in basic units (e.g. mm)

ST_NciBaseFrame

The structure ST_NciBaseFrame describes a modal zero shift and rotation. The operating principle is the same as for zero shift and rotation in the interpreter, i.e. the point of rotation is the current origin (see rotation [▸ 129] in the interpreter documentation [▸ 107]).

```
TYPE ST_NciBaseFrame:
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeBaseFrame; (*Do not override this parameter *)
nDisplayIndex: UDINT;
fShiftX: LREAL;
fShiftY: LREAL;
fShiftZ: LREAL;
fRotX: LREAL;
fRotY: LREAL;
fRotZ: LREAL;
fShiftQ1: LREAL;
fShiftQ2: LREAL;
fShiftQ3: LREAL;
fShiftQ4: LREAL;
fShiftQ5: LREAL;
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▸ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

fShiftX: Zero shift in X direction

fShiftY: Zero shift in Y direction

fShiftZ: Zero shift in Z direction

fRotX: Rotation of the X axis

fRotY: Rotation of the Y axis

fRotZ: Rotation of the Z axis

fShiftQ1: Offset of the Q1 axis

fShfitQ2: Offset of the Q2-axis

fShiftQ3: Offset of the Q3-axis

fShiftQ4: Offset of the Q4-axis

fShiftQ5: Offset of the Q5-axis

ST_NciPathDynamics

The structure *ST_NciPathDynamics* sets the path dynamics (acceleration, deceleration, jerk). The operating principle is the same as for *paramPathDynamics* in the interpreter (see paramPathDynamics [▸ 156] in the interpreter documentation [▸ 107]).

```
TYPE ST_NciPathDynamics:
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypePathDynamics; (*do not override this parameter *)
nDisplayIndex: UDINT;
fAcc: LREAL;
```

```
fDec: LREAL;
fJerk: LREAL;
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▸ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

fAcc: Maximum permitted path acceleration

fDec: Maximum permitted path deceleration

fJerk: Maximum permitted path jerk

ST_NciAxisDynamics

The structure *ST_NciAxisDynamics* sets the path axis dynamics (acceleration, deceleration, jerk). The operating principle is the same as for *paramAxisDynamics* in the interpreter (see paramAxisDynamics [▸ 156] in the interpreter documentation [▸ 107])

```
TYPE ST_NciAxisDynamics:
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeAxisDynamics; (*Do not override this parameter*)
nDisplayIndex: UDINT;
nAxis: UDINT;
fAcc: LREAL;
fDec: LREAL;
fJerk: LREAL;
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▸ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

nAxis: Axis in interpolation group X:0 Y:1 Z:2 Q1:3 ... Q5:7

fAcc: Maximum permitted axis acceleration

fDec: Maximum permitted axis deceleration

fJerk: Maximum permitted axis jerk

ST_NciDwellTime

The structure *ST_NciDwellTime* is used to activate a dwell time in seconds (see dwell time [▸ 122] in the interpreter documentation [▸ 107])

```
TYPE ST_NciDwellTime:
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeDwellTime; (*Do not override this parameter *)
nDisplayIndex: UDINT;
fDwellTime: LREAL;
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▸ 280])

nDisplayIndex: For display purposes, such as block number in G-Code

fDwellTime: Dwell time in seconds

ST_NciFeedrateIpol

The structure *ST_NciFeedrateIpol* can be used to set the feed interpolation (see Feed interpolation [▸ 123]).

```
TYPE ST_NciFeedrateIpol :
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeFeedrateIpol;(*Do not overwrite this parameter*)
nDisplayIndex: UDINT;
eFeedrateIpol: E_NciFeedrateIpol;(*E_NciFeedrateIpolConstant = FCONST, E_NciFeedrateIpolLinear=FLIN
```

```
*)
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▸ 280])

eFeedrateIpol: specifies the feed interpolation.

```
TYPE E_NciFeedRateIpol :(
E_NciFeedrateIpolConstant,
E_NciFeedrateIpolLinear
)
END_TYPE
```

ST_NciTangentialFollowingDesc

This is a modal command for switching tangential following on or off.

```
TYPE ST_NciTangentialFollowingDesc :
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeTfDesc; (*do not override this parameter *)
bTangOn: BOOL;
nTangAxis: E_NciAxesInGroup; (*axis used for tangential following *)
nPathAxis1: E_NciAxesInGroup; (*describing the plane e.g. x*)
nPathAxis2: E_NciAxesInGroup; (*e.g. y ==> g17, xy plane*)
fOffset: LREAL; (*geo tangent is 0 degree, counting is mathmatical positive *)
fCriticalAngle1: LREAL;
nTfBehavior: E_TangentialFollowingBehavior; (*what to do if angle becomes bigger than critical angle
1 *)
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▸ 280])

bTangOn: If TRUE, tangential following is switched on.

nTangAxis: Axis (Q1..Q5) that is used as tangential axis (type: E_NciAxesInGroup [▸ 291]).

nPathAxis1: First path axis describing the plane and orientation for calculating the tangent.

nPathAxis2: Second path axis describing the plane and orientation for calculating the tangent.

fOffset: Offset of the tangential axis

fCriticalAngle1: Critical angle 1. The response in cases where the angle between two segments is greater than fCriticalAngle1 is specified with nTfBehavior.

nTfBehavior: see fCriticalAngle1 (type: E_TangentialFollowingBehavior [▸ 291])

E_NciAxesInGroup

```
TYPE E_NciAxesInGroup :
(
NoneAxis := 0,
XAxis,
YAxis,
ZAxis,
Q1Axis,
Q2Axis,
Q3Axis,
Q4Axis,
Q5Axis
);
END_TYPE
```

E_TangentialFollowingBehavior

```
TYPE E_TangentialFollowingBehavior :
(
E_TfIngoreAll, (*ignore critical angle *)
E_TfErrorOnCritical1 (*if angle becomes bigger than critical angle 1 ==> error *)
);
END_TYPE
```

E_TfIngoreAll: The critical angle is ignored.

E_TfErrorOnCritical1: An error is returned if the critical angle is exceeded.

ST_NciEndOfTables

Indicates the last entry of the last table. Is used for signaling the bChannelDone flag in FB_NciFeedTable [▸ 279].

```
TYPE ST_NciEndOfTables :
STRUCT
nEntryType: E_NciEntryType := E_NciEntryTypeEndOfTables; (*do not override this parameter *)
END_STRUCT
END_TYPE
```

nEntryType: Do not override this parameter (type: E_NciEntryType [▸ 280])

6 Samples

NCI: NCISimpleSample

Download:

https://infosys.beckhoff.com/content/1033/TF5100_TC3_NC_I/Resources/zip/9007202693487883.zip

The example NCISimpleSample shows how an G-Code program is loaded from the PLC and processing is started.

You need to copy the enclosed parts program TestIt.nc into the TwinCAT\Mc\Nci directory. Otherwise the parts program will not be found during loading. Alternatively you can adjust the path in the PLC program.

PLC interpolation: PlcInterpolationSimpleSample

Download:

https://infosys.beckhoff.com/content/1033/TF5100_TC3_NC_I/Resources/zip/18014401453622155.zip

The sample shows how a movement can be affected with the library Tc2_PlcInterpolation directly from the PLC.

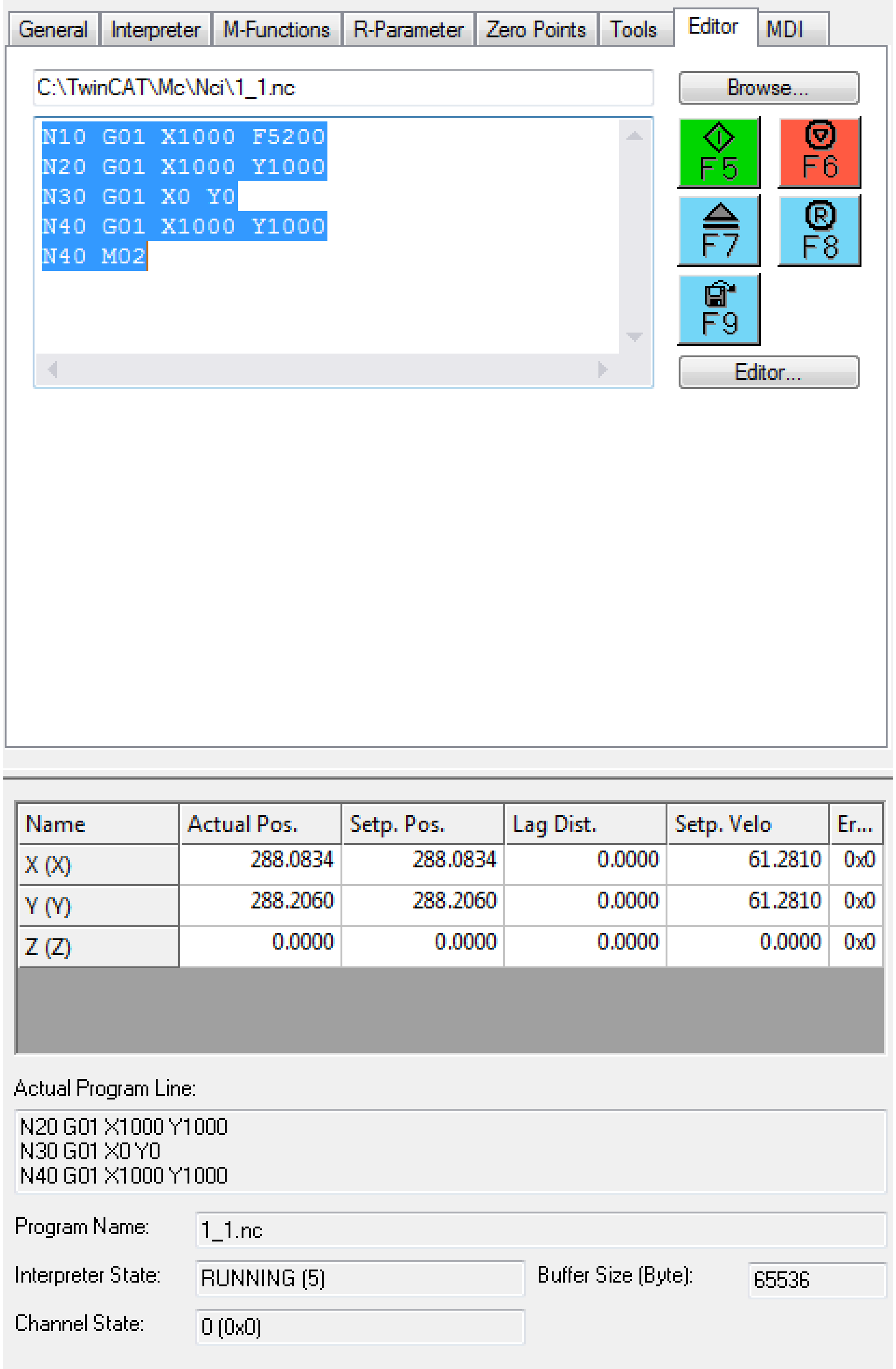
7 Appendix

7.1 Display of the parts program

Reading of the current NC line via ADS

This ADS Read command returns a maximum of three lines of the current parts program, i.e. the current line of code and perhaps two previously processed lines.

Function	ADS-Read
Port	500 (dec)
Index Group	0x2300 + channel ID
Index Offset	0x2000 0001
Data	string (30 bytes min.)



Reading of the current program name

This ADS Read command returns the program name of the current main NC program (in this case 1_1.nc).

Function	ADS-Read
Port	500 (dec)
Index Group	0x2100 + channel ID
Index Offset	0x7
Data	string, 100 characters max.

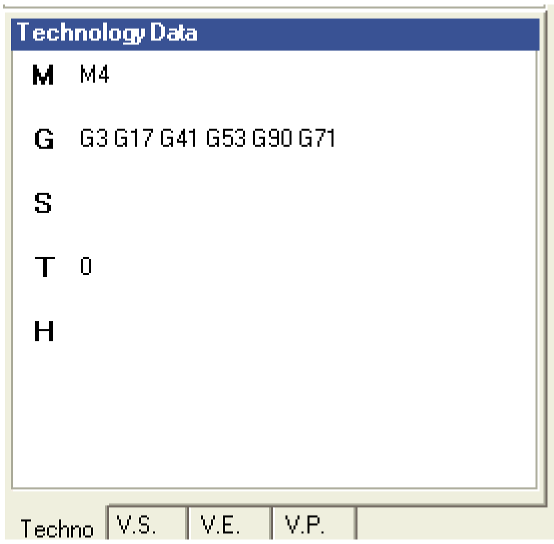
Reading of the current file information

In contrast to the 'Reading the current NC line' function, in this case not the line itself is read, but associated line information. The return value is the current program name (e.g. file name of the subroutine) and a file offset. Based on this information, the user interface can open the associated file and highlight the respective line. The display is no longer limited to 3 rows, i.e. any number of lines can be displayed.

In the event of an NCI load or runtime error, information about the associated line of code can be obtained via this route.

Function	ADS-Read	
Port	500 (dec)	
Index Group	0x2100 + channel ID	
Index Offset	0x12	
Data	UINT32	Current display of 1: SAF- 2: Interpreter 3: Error offset
	UINT32	File offset
	char[260]	path + program name

7.2 Display of technology data



The currently active technology data such as G functions, zero shifts and rotation can be read via ADS.

Activation for reading the technology data

In order to read the above-mentioned parameters, activation via ADS is required first.

The function must be activated before the start of the NC program, or earlier. It remains active until either a TwinCAT restart is performed or the function is reset explicitly.

Function	ADS-Write
Port	500 (dec)
Index Group	0x2000 + channel ID
Index Offset	0x0053
Data	DWORD 0: disable (default) 1: enable

Reading the currently active zero shift

This command reads the active zero shift of the segment currently in block execution (SAF). If no zero shift is active (G53), the structure for the individual components contains a zero vector. These data can be used for switching the display between machine coordinates and programming coordinates, for example.

The data, which are read with the function block 'ItpReadZeroShift', for example, may differ from these values, since the interpreter data are read with the function block, which may already take into account new offsets.

Function	ADS-Read	
Port	500 (dec)	
Index Group	0x2100 + channel ID	
Index Offset	0x0014	
Data	{	
	UINT32	block counter
	UINT32	dummy
	LREAL[3]	zero shift G54..G57
	LREAL[3]	zero shift G58
	LREAL[3]	zero shift G59
	}	

Reading the currently active rotation

This command reads the active rotation of the segment currently in block execution (SAF).

Function	ADS-Read	
Port	500 (dec)	
Index Group	0x2100 + channel ID	
Index Offset	0x0015	
Data	{	
	UINT32	block counter
	UINT32	dummy
	LREAL[3]	rotation of X, Y & Z in degrees
	}	

Reading the currently active G-Code

The G-Code is subdivided into groups. For example, the geometries types with modal effect (G01, G02...) and the plane selection (G17..G19) form separate groups. When the G-Code information is read, the enumerator for the groups is also read. These can then be displayed in an application-specific manner.

Since the read command comes with a parameter to be read, not all groups have to be read. The memory provided is always filled by group 1. If, for example, the transferred memory size is 3x8 bytes, the data for the block counter, group 1 and 2 are returned.

Function	ADS-Read	
Port	500 (dec)	
Index Group	0x2100 + channel ID	
Index Offset	0x0013	
Data	{	
	UINT32	block counter
	UINT32	Group 1: ModalGeoTypes
	UINT32	Group 2: BlockwiseGeoTypes
	UINT32	Group 3: ModalPlaneSelection
	UINT32	Group 4: ModalToolCompensation
	UINT32	Group 5: ModalToolFeedDirection
	UINT32	Group 6: ModalZeroShift
	UINT32	Group 7: ModalAccurateStop
	UINT32	Group 8: BlockwiseAccurateStop
	UINT32	Group 9: ModalDesignationAbsInc
	UINT32	Group 10: ModalDesignationInchMetric
	UINT32	Group 11: ModalFeedRateInCurve
	UINT32	Group 12: ModalCenterpointCorr
	UINT32	Group 13: ModalCircleCpAbsInc
	UINT32	Group 14: ModalCollisionDetection
	UINT32	Group 15: ModalRotation
	UINT32	Group 16: ModalCalcExRot
	UINT32	Group 17: ModalDiam
	UINT32	Group 18: ModalFeedrateIpol
	UINT32	Group 19: ModalMirror
	}	

```
#define GCodeOffset 0x1000
#define CommonIdentOffset 0x2000 // used for non-g-code commands, like rot, cfc...
```

Group 1: ModalGeoTypes

```
enum GCodeGroup_ModalGeoTypes
{
ModalGeoTypeUndefined = 0,
ModalGeoTypeG0 = 0 + GCodeOffset, // line - rapid traverse
ModalGeoTypeG01 = 1 + GCodeOffset, // straight line
ModalGeoTypeG02 = 2 + GCodeOffset, // circle clockwise
ModalGeoTypeG03 = 3 + GCodeOffset // circle anticlockwise
};
```

Group 2: BlockwiseGeoTypes

```
enum GCodeGroup_BlockwiseGeoTypes
{
BlockwiseGeoTypeNone = 0,
BlockwiseGeoTypeG04 = 4 + GCodeOffset, // dwell time
BlockwiseGeoTypeG74 = 74 + GCodeOffset, // homing
BlockwiseGeoTypeCip = 1 + CommonIdentOffset // circle parametrized with 3 points
};
```

Group 3: ModalPlaneSelection

```
enum GCodeGroup_ModalPlaneSelection
{
ModalPlaneSelectUndefined = 0,
ModalPlaneSelectG17 = 17 + GCodeOffset, // xy-plane
ModalPlaneSelectG18 = 18 + GCodeOffset, // zx-plane
ModalPlaneSelectG19 = 19 + GCodeOffset // yz-plane
};
```

Group 4: ModalToolCompensation

```
enum GCodeGroup_ModalToolCompensation
{
ModalToolCompUndefined = 0,
ModalToolCompG40 = 40 + GCodeOffset, // tool compensation off
ModalToolCompG41 = 41 + GCodeOffset, // tool compensation left
ModalToolCompG42 = 42 + GCodeOffset // tool compensation right
};
```

Group 5: ModalToolFeedDirection

```
enum GCodeGroup_ModalToolFeedDirection
{
ModalToolFeedDirUndefined = 0,
ModalToolFeedDirPos = 2 + CommonIdentOffset, // tool feed direction positive
ModalToolFeedDirNeg = 3 + CommonIdentOffset // tool feed direction negative
};
```

Group 6: ModalZeroShift

```
enum GCodeGroup_ModalZeroShift
{
ModalZeroShiftUndefined = 0,
ModalZeroShiftG53 = 53 + GCodeOffset, // zero shift off
ModalZeroShiftG54G58G59 = 54 + GCodeOffset, // zero shift G54 + G58+ G59
ModalZeroShiftG55G58G59 = 55 + GCodeOffset, // zero shift G55 + G58+ G59
ModalZeroShiftG56G58G59 = 56 + GCodeOffset, // zero shift G56 + G58+ G59
ModalZeroShiftG57G58G59 = 57 + GCodeOffset // zero shift G57 + G58+ G59
};
```

Group 7: ModalAccurateStop

```
enum GCodeGroup_ModalAccurateStop
{
ModalAccurateStopNone = 0,
ModalAccurateStopG60 = 60 + GCodeOffset // modal accurate stop
};
```

Group 8: BlockwiseAccurateStop

```
enum GCodeGroup_BlockwiseAccurateStop
{
BlockwiseAccurateStopNone = 0,
BlockwiseAccurateStopG09 = 9 + GCodeOffset, // common accurate stop
BlockwiseAccurateStopTpm = 4 + CommonIdentOffset // target position monitoring
};
```

Group 9: ModalDesignationAbsInc

```
enum GCodeGroup_ModalDesignationAbsInc
{
ModalDesignAbsIncUndefined = 0,
ModalDesignAbsIncG90 = 90 + GCodeOffset, // absolute designation
ModalDesignAbsIncG91 = 91 + GCodeOffset // incremental designation
};
```

Group 10: ModalDesignationInchMetric

```
enum
GCodeGroup_ModalDesignationInchMetric
{
ModalDesignInchMetricUndefined = 0,
ModalDesignInchMetricG70 = 70 + GCodeOffset, // designation inch
ModalDesignInchMetricG71 = 71 + GCodeOffset, // designation metric
ModalDesignInchMetricG700 = 700 + GCodeOffset, // designation inch & feedrate recalculated
ModalDesignInchMetricG710 = 710 + GCodeOffset // designation metric & feedrate recalculated
};
```

Group 11: ModalFeedRateInCurve

```
enum GCodeGroup_ModalFeedRateInCurve
{
ModalFeedRateInCurveUndefined = 0,
ModalFeedRateInCurveCfc = 5 + CommonIdentOffset, // constant feed contour
ModalFeedRateInCurveCfin = 6 + CommonIdentOffset, // constant feed inner contour
ModalFeedRateInCurveCftcp = 7 + CommonIdentOffset // constant feed tool center point
};
```

Group 12: ModalCenterpointCorr

```
enum GCodeGroup_ModalCenterpointCorr
{
ModalCenterpointCorrUndefined = 0,
ModalCenterpointCorrOn = 8 + CommonIdentOffset, // circle centerpoint correction on
ModalCenterpointCorrOff = 9 + CommonIdentOffset // circle centerpoint correction off
};
```

Group 13: ModalCircleCpAbsInc

```
enum GCodeGroup_ModalCircleCpAbsInc
{
ModalCircleCpUndefined = 0,
ModalCircleCpIncremental = 10 + CommonIdentOffset, // circle centerpoint incremental to start point
ModalCircleCpAbsolute = 11 + CommonIdentOffset // circle centerpoint absolute
};
```

Group 14: ModalCollisionDetection

```
enum GCodeGroup_ModalCollisionDetection
{
ModalCollisionDetectionUndefined = 0,
ModalCollisionDetectionOn = 12 + CommonIdentOffset, //collision detection on
ModalCollisionDetectionOff = 13 + CommonIdentOffset //collision detection off
};
```

Group 15: ModalRotation

```
enum GCodeGroup_ModalRotation
{
ModalRotationUndefined = 0,
ModalRotationOn = 14 + CommonIdentOffset, // rotation is turned on
ModalRotationOff = 15 + CommonIdentOffset // rotation is turned off
};
```

Group 16: ModalCalcExRot

```
enum GCodeGroup_ModalCalcExRot
{
ModalCalcExRotUndefined = 0,
ModalCalcExRotOn = 16 + CommonIdentOffset, // extended calculation for rotation turned on
ModalCalcExRotOff = 17 + CommonIdentOffset // extended calculation for rotation turned off
};
```

Group 17: ModalDiam

```
enum GCodeGroup_ModalDiam
{
ModalDiamUndefined = 0,
ModalDiamOn = 18 + CommonIdentOffset, // diameter programming on
ModalDiamOff = 19 + CommonIdentOffset // diameter programming off
};
```

Group 18: ModalFeedrateIpol

```
enum GCodeGroup_ModalFeedrateIpol
{
ModalFeedrateIpolUndefined = 0,
ModalFeedrateIpolConst = 20 + CommonIdentOffset, // federate interpolation constant (default)
ModalFeedrateIpolLinear = 21 + CommonIdentOffset // federate interpoaltion linear to remaining path
};
```

Group 19: ModalMirror

```
enum GCodeGroup_ModalMirror
{
// value - (32+CommonIdentOffset) shows the bitmask for mirrored axes
// that's why the sequence seems to be strange...
//
ModalMirrorUndefined = 0,
ModalMirrorOff = 32 + CommonIdentOffset,
ModalMirrorX = 33 + CommonIdentOffset,
ModalMirrorY = 34 + CommonIdentOffset,
ModalMirrorXY = 35 + CommonIdentOffset,
ModalMirrorZ = 36 + CommonIdentOffset,
ModalMirrorZX = 37 + CommonIdentOffset,
ModalMirrorYZ = 38 + CommonIdentOffset,
ModalMirrorXYZ = 39 + CommonIdentOffset
};
```

7.3 Displaying the remaining path length

If calculation of the remaining path length is switched active, it is calculated up to as far as the next accurate stop, or as far as the last geometric segment in memory (block preparation). An accurate stop is, for instance, generated by G09 or by G60. However, M-functions of type handshake, decoder stops and G04 implicitly generate an accurate stop.

Activation:

Index Group: 0x3000 + Group ID
Index Offset: 0x0508

see index offset specification for group parameters [▶ 391]

Reading the remaining path length:
Reading is again implemented through ADS, and can also be recorded with TwinCAT Scope.

Index Group: 0x3100 + Group ID
Index Offset: 0x0522

The remaining path length can be transferred with the cyclic channel interface to the PLC via ItpSetCyclicLrealOffsets [▶ 220].
see index offset specification for group state [▶ 396]

7.4 Parameterisation

The parameterization of the NCI comprises the standard dynamic parameters (acceleration, deceleration, jerk) and their online changes, along with the minimum velocity and the parameters for the reduction of the path velocity including online change.

General characteristics at segment transitions

- Velocity: The segment set velocity VS changes at the segment transition from VS_in to VS_out. At the segment transition the velocity is always reduced to the lower of the two values.
- Acceleration: The current path acceleration is always returned to $a = 0$ at segment transition.
- Jerk: The jerk unit J changes according to the geometry at the segment transition. This can cause a significant step change in dynamics.
- It is possible to smooth segment transitions [▶ 110].

Table 1: NCI group parameters

Parameter	Meaning and boundary conditions
Curve velocity reduction mode [▸ 303]	Coulomb, cosine or VELOJUMP
Minimum velocity [▸ 302]	Path velocity which may not be less than this value (except peaks with movement reversal): $V\_min \geq 0.0$
Reduction method for C1 transitions [▸ 303]	Reduction factor for *C1* transitions: $C1 \geq 0.0$
VELOJUMP: C0 reduction factors *C0X*, C0Y, C0Z	Reduction factors for *C0* transitions for *X*, Y, Z axis: *C0X ≥ 0.0, C0Y ≥ 0.0, C0Z ≥ 0.0 (axis parameters, online modification in* interpreter [▸ 158] possible).
DEVIATIONANGLE: Reduction factor *C0 C0*	Path reduction factor for *C0* transitions: $1.0 \geq C0 \geq 0.0$
DEVIATIONANGLE: Critical angle (low) φ_l	Angle from which a velocity reduction is applied at the segment transition: $0 \leq \varphi\_l < \varphi\_h \leq \pi$
DEVIATIONANGLE: Critical angle (high) φ_h	Angle from which the velocity at the segment transition (*v_link) is reduced to 0.0*: $0 \leq \varphi\_l < \varphi\_h \leq \pi$
Tolerance sphere radius [▸ 136] *TBR*	Radius of the tolerance spheres: 1000.0 mm $\geq TBR \geq 0.1$ *mm*
C2 reduction factor [▸ 158] *C2*	Reduction factor for smoothed transitions: $C2 \geq 0.0$
Global software limit positions for the path [▸ 304]	Switches monitoring of the global software end positions for the path axes

Minimum velocity

Each NCI group has a minimum path velocity $V\_min \geq 0.0$. The actual velocity should always exceed this value. User-specified exceptions are: programmed stop at segment transition, path end and override requests which lead to a velocity below the minimum value. A systemic exception is a motion reversal. With the reduction method DEVIATIONANGLE the deflection angle is $\varphi \geq \varphi\_h$, in which case the minimum velocity is ignored. *V_min* must be less than the set value for the path velocity (F word) of each segment.

The minimum velocity can be set to a new value $V\_min \geq 0.0$ in the NC program at any time. The unit is *mm/sec*.

Classification of the segment transitions

In general, the transition from one segment to the next is not indefinitely smooth. Therefore, it is necessary to reduce the velocity at the transition point in order to avoid dynamic instability. For this purpose, the transitions are geometrically classified and the effective transition velocity - V_link - is determined in three steps.

Segments - as geographical objects - are defined here as curves in terms of differential geometry and are parameterized by the arc length.
A segment transition from a segment *S_in* to a segment *S_out* is classified in geometrical terms as type *Ck*, where *k* is a natural number (including *0*), if each segment has *k* continuous arc length differentials and the k^{th} derivatives at the transition point correspond.

C0 transitions have a knee-point at the transition point.

C1 transitions appear smooth, but are not smooth in dynamic terms. One example is the straight line-semi circle transition in the stadium: at the transition point there is a step change in acceleration.

C2 transitions (and of course *Ck* transitions with $k > 2$) are dynamically smooth (jerk restricted).

Reduction method for C2 transitions

As at all transitions, at *C2* transitions *V_link* is set to equal the minimum of both set segment velocities: $V\_link = min(V\_in, V\_out)$. There is no further reduction.

Reduction method for C1 transitions

First, *V_link* is set to the lower of the two segment target velocities: *V_link* = min(*V_in,V_out*). The geometrically induced absolute step change in acceleration *AccJump* in the segment transition is calculated depending on the geometry types *G_in* and *G_out,* and the plane selection *G_in* and *G_out* of the segments to be connected, at velocity *V_link*. If this is greater than *C1* times the path acceleration/(absolute) deceleration *AccPathReduced* permissible for the geometries and planes, the velocity *V_link* is reduced until the resulting step change in acceleration is equal to *AccPathReduced.* If this value is less than *V_min*, then *V_min* takes priority.

NOTE! When changing the dynamic parameters, the permissible path acceleration for the geometries and planes and thereby the reaction of the reduction changes automatically.

Interface: XAE [▶ 20] and interpreter [▶ 158]

Reduction modes for C0 transitions

Several reduction methods are available for *C0* transitions. The reduction method VELOJUMP reduces the velocity after permitted step changes in velocity for each axis. The reduction method DEVIATIONANGLE reduces the velocity depending on the deflection angle φ (angle between the normalized end tangent *T_in* of the incoming segment *S_in* and the normalized start tangent *T_out* of the outgoing segment *S_out*). The cosine reduction method is a purely geometrical method (see curve velocity reduction method [▶ 22]).

The VELOJUMP method is recommended for mechanically independent axes, while for mechanically coupled axes (the *Y* axis is attached to the *X* axis, for example) the DEVIATIONANGLE method is usually recommended.

Reduction method for C0 transitions: VELOJUMP

If *V_link = min(V_in,V_out)*, and for each axis *V_jump[i] = C0[i] * min(A+[i],-A-[i]) * T* is the permitted absolute step change in velocity for the axis *[i]*, wherein *C0[i]* is the reduction factor and *A+[i], A-[i]* are the acceleration/deceleration limits for the axis *[i]*, and *T* is the cycle time. The VELOJUMP reduction method ensures that the path velocity is reduced at the segment transition *V_link* until the absolute step change in the set axis velocity of axis *[i]* is at most *V_jump[i]*. *V_min* nevertheless has priority: if *V_link* is less than *V_min*, *V_link* is set to *V_min*. In the case of movement reversal with no programmed stop, there will be a step change in axis velocity.

NOTE! When changing the dynamic parameters, the maximum permissible step changes in axis velocity automatically change at the same time.

Reduction method for C0 transitions: DEVIATIONANGLE

NOTE! When changing the dynamic parameters, the reduction factors do not automatically change at the same time.

Changing the parameters for C0 transitions: DEVIATIONANGLE

Table 2: Parameter

Parameter	Meaning and boundary conditions
DEVIATIONANGLE: Reduction factor *C0C0*	Path reduction factor for *C0* transitions: 1.0 ≥ *C0* ≥ *0.0*
DEVIATIONANGLE: Critical angle (low) *φ_l*	Angle from which reduction takes effect: *0 ≤ φ_l < φ_h ≤ π*
DEVIATIONANGLE: Critical angle (high) *φ_h*	Angle from which reduction to *v_link = 0.0* takes effect: *0 ≤ φ_l < φ_h ≤ π*

Interface: Interpreter [▶ 158]

Cosine reduction method

See here [▶ 22].

Tolerance sphere radius and C2 reduction factor

These parameters are described under the heading Smoothing of segment transitions [▶ 110].

Global software limit positions for the path

The *'Global software limit position monitoring for the path'* offers two different ways of monitoring the end position.

Limit position monitoring by the SAF task

This type of end position monitoring is always active if the limit position for the axis has been switched to active (axis parameter). The monitoring is carried out component for component by the SAF task. This means that if the end position is exceeded, the path velocity is instantly set to 0, and the entire interpolation group has an error.

This type of monitoring is activated through the axes parameters, and **not** by means of the group parameters described here.

Software limit positions on the path

To prevent the path velocity being set to 0 immediately when a violation of the software end positions is encountered, the function *'Global software end position monitoring of the path'* must be enabled. If this is active, the movement stops at the NC block in which the end positions were violated. The velocity is reduced via a ramp.

- So that the monitoring is only executed for the desired path axes, the software limit positions for the axis components must be selected (axis parameters).
- The monitoring is carried out for the standard geometry segments. These include
  Straight line
  Circle
  Helix
- Curves with splines are not monitored. The set values associated with the splines are always within the tolerance sphere. Otherwise the limit position monitoring will make use of the SAF task.
- Because meaningful and generally applicable monitoring of the end positions can only be carried out at the NC program's run-time (before lookahead) it is possible that the path axes will move as far as (but not including) the NC block in which the limit positions are exceeded.
- If for some reason the axes are located outside the software limit positions it is possible to move back into the correct region in a straight line.

Parameterization:

XAE: Group parameters [▶ 20]

7.4.1 Path override (interpreter override types)

The path override is a velocity override. This means that changing the override creates a new velocity, but does not affect the ramps (acceleration or jerk). The used override types only differ in terms of reference velocity.

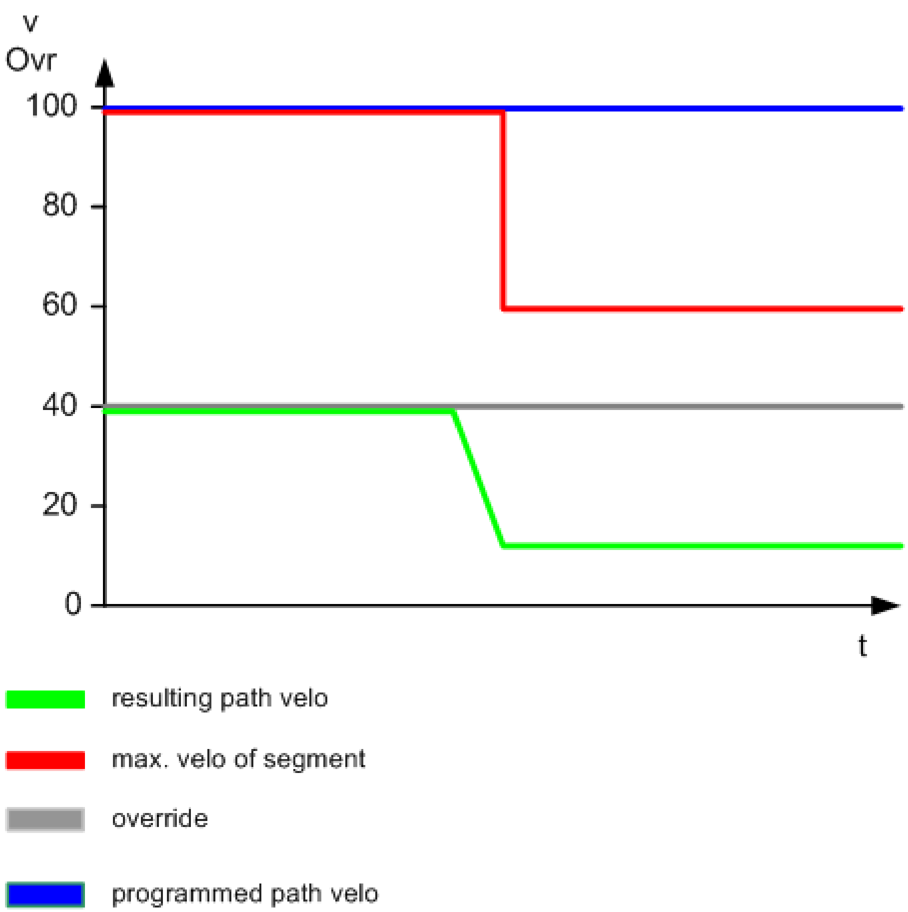
The parameterization takes place in the interpolation channel under the group parameters [▶ 22].

Option 'Reduced' - based on the reduced velocity (default)

Because of the relevant dynamic parameters (braking distance, acceleration etc.) it is not possible for the programmed velocity (the blue line) be achieved in every segment. For this reason a velocity, possibly reduced, (the red line) is calculated for each geometric segment. In the standard case, the override is made with reference to this segment velocity.

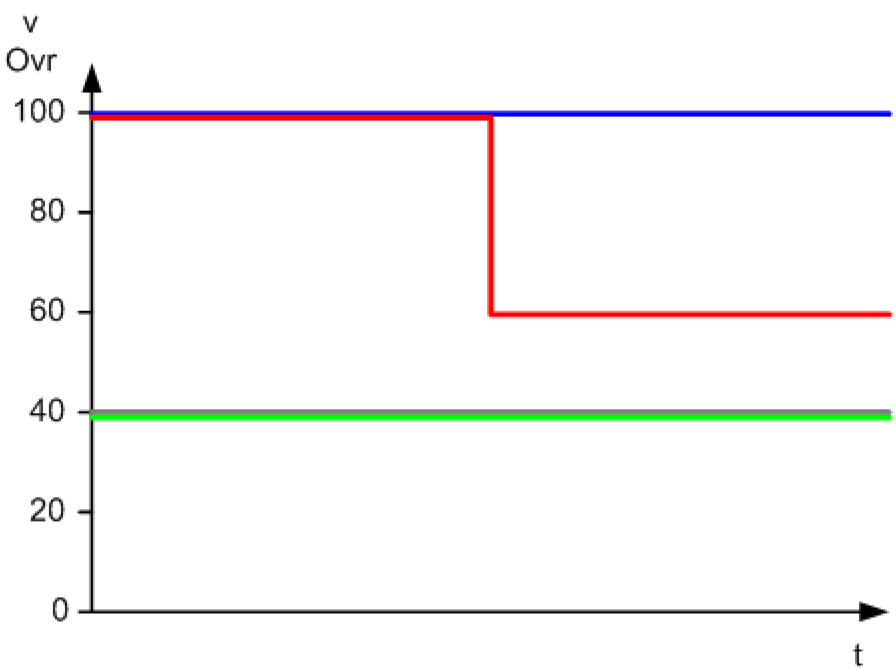
The advantage of this override type is that if override values are small the machine operates with an approximately linear reduction in velocity, and this is therefore the correct setting for most applications.

v_res = v_max * Override



Option 'Original' - based on the programmed path velocity

The override value is based on the velocity programmed by the user. The maximum segment velocity only has a limiting effect.



Selection 'Reduced [0 ... >100%]' - based on internally reduced velocity with the option to specify a value greater than 100%

The override type behaves like 'Reduced' [▶ 304]. With this override type it is possible to travel along the path more quickly than programmed in the G-Code. There is no limitation to 120%, for example. The maximum possible path velocity is limited by the maximum velocities of the axis components (G0 velocity) and their dynamics.

If limitation to a particular value, e.g. 120%, is required, this can be set in the PLC project.

7.5 Cyclical Channel Interface

The channel interface is responsible for the cyclic data exchange between the PLC and the NCI.

From the NCI to the PLC (160 bytes)

```
TYPE NCTOPLC_NCICHANNEL_REF :
STRUCT
BlockNo             : UDINT;
FastMFuncMask       : ARRAY [1..5] OF DWORD;
HskMFuncNo          : UINT;
HskMFuncReq         : WORD;
HFuncValue          : UDINT;
SpindleRpm          : UINT;
Tool                : UINT;
ChnState            : NCTOPLC_NCICHANNEL_REF_CHN_STATE;
IntParams           : ARRAY [0..3] OF UDINT;
DoubleParams        : ARRAY [0..3] OF LREAL;
PathVelo            : LREAL;
LoadedProg          : UDINT;
ItpMode             : WORD;
ItpState            : UINT;
ErrorCode           : UDINT;
ChnId               : UINT;
GrpId               : UINT;
ItfVersion          : UINT;
_reserved1          : UINT;
ChnOperationState   : UDINT;
McsAxisIDs          : ARRAY [0..7] OF USINT;
AcsAxisIDs          : ARRAY [0..7] OF USINT;
_reserved2          : ARRAY [1..24] OF USINT;
END_STRUCT
END_TYPE
```

Variable name	Data type	Description
BlockNo	UDINT	block number
FastMFuncMask	ARRAY OF DWORD	Bit mask for evaluation of the fast M-functions [▶ 145]
HskMFuncNo	UINT	Number of synchronous M-function present (M-function with handshake)
HskMFuncReq	WORD	Flag indicating that a synchronous M-function is present 0: no synchronous M-function is present 1: a synchronous M-function is present
HFuncValue	DINT	Value of the auxiliary function
SpindleRpm	WORD	Spindle rotation speed
Tool	WORD	Tool number
ChnState	NCTOPLC_NCICHANNEL_REF_CHN_STATE	DWORD with status information for the channel (see status information for the channel (ChnState) [▶ 307]
IntParams	ARRAY [0..3] OF UDINT	Data of the freely configurable channel interface (see ItpSetCyclicUDintOffsets [▶ 221])
DoubleParams	ARRAY [0..3] OF LREAL	Data of the freely configurable channel interface (see ItpSetCyclicLrealOffsets [▶ 220])
PathVelo	LREAL	Current path set velocity
LoadedProg	UDINT	Name of the currently executed NC program. If the name is not a UDINT, this value is 0.
ItpMode	WORD	Bit mask that indicates execution in interpreter mode.
ItpState	UINT	Status [▶ 11] of the interpreter
ErrorCode	UDINT	Error code [▶ 313] of the interpreter channel
ChnId	UINT	Channel ID
GrpId	UINT	group ID
ItfVersion	UINT	Version of this cyclic channel interface
ChnOperationState	UDINT	Channel state for a channel of the kinematic transformation; has no purpose for an interpolation channel.
McsAxisIDs	ARRAY [0..7] OF USINT	IDs of the MCS axes for a kinematic transformation channel; has no purpose for an interpolation channel.
AcsAxisIDs	ARRAY [0..7] OF USINT	IDs of the ACS axes for a kinematic transformation channel; has no purpose for an interpolation channel.

Channel status information (ChnState)

In the XAE **the channel status information can only be read with a plain text name, from the PLC only via the bit number.**

Name	Bit number (zero based)	Description
bIsInterpolationChannel	0	Indicates that the linked channel is an interpolation channel.
bIsKinematicChannel	1	Indicates that the structure is linked to a channel for the kinematic transformation.
bIsEStopRequested	8	Indicates that an ItpEStop was called, without checking whether the axes are already at standstill.
bIsFeedFromBackupList	10	For retracing the current entries from the interpreter backup list are sent.
bIsMovingBackward	11	Indicates that the current motion is a reversing motion.
bRetraceStartPosReached	12	Indicates that the program start was reached during reversing.

From PLC to NCI (128 bytes)

```
TYPE PLCTONC_NCICHANNEL_REF :
STRUCT
SkipLine            : WORD; (* Mask to skip lines *)
ItpMode             : WORD;
MFuncGranted        : WORD; (* granted signal of the M-function *)
_reserved1          : UINT;
ChnAxesOvr          : UDINT; (* Channel override in percent * 100 *)
ChnSpindleOvr       : UDINT;
_reserved2          : ARRAY [1..112] OF USINT;
END_STRUCT
END_TYPE
```

Variable name	Data type	Description
SkipLine	WORD	Bit mask with which block skipping [▸ 108] of the NCI is parameterized from the PLC
ItpMode	WORD	Bit mask with which the interpreter execution mode can be altered. This is, for instance, required if the interpreter is to operate in single block [▸ 114] mode.
MFuncGranted	WORD	Flag with which an M-function of type 'Handshake' is confirmed. 0: Not acknowledged 1: Acknowledgement
ChnAxesOvr	UDINT	Channel override for the axes from 0...1000000 (corresponds to 0 - 100%)
ChnSpindleOvr	UDINT	Channel override for the spindle between 0 and 1000000 (corresponds to 0 - 100%); currently not supported.

7.6 ADS Return Codes

Error codes: 0x000 [▸ 309]..., 0x500 [▸ 309]..., 0x700 [▸ 310]..., 0x1000 [▸ 312]...

Global Error Codes

Hex	Dec	Description
0x0	0	no error
0x1	1	Internal error
0x2	2	No Rtime
0x3	3	Allocation locked memory error
0x4	4	Insert mailbox error
0x5	5	Wrong receive HMSG
0x6	6	target port not found
0x7	7	target machine not found
0x8	8	Unknown command ID
0x9	9	Bad task ID
0xA	10	No IO
0xB	11	Unknown ADS command
0xC	12	Win 32 error
0xD	13	Port not connected
0xE	14	Invalid ADS length
0xF	15	Invalid ADS Net ID
0x10	16	Low Installation level
0x11	17	No debug available
0x12	18	Port disabled
0x13	19	Port already connected
0x14	20	ADS Sync Win32 error
0x15	21	ADS Sync Timeout
0x16	22	ADS Sync AMS error
0x17	23	ADS Sync no index map
0x18	24	Invalid ADS port
0x19	25	No memory
0x1A	26	TCP send error
0x1B	27	Host unreachable
0x1C	28	Invalid AMS fragment

Router Error Codes

Hex	Dec	Name	Description
0x500	1280	ROUTERERR_NOLOCKEDMEMORY	No locked memory can be allocated
0x501	1281	ROUTERERR_RESIZEMEMORY	The size of the router memory could not be changed
0x502	1282	ROUTERERR_MAILBOXFULL	The mailbox has reached the maximum number of possible messages. The current sent message was rejected
0x503	1283	ROUTERERR_DEBUGBOXFULL	The mailbox has reached the maximum number of possible messages. The sent message will not be displayed in the debug monitor
0x504	1284	ROUTERERR_UNKNOWNPORTTYPE	Unknown port type
0x505	1285	ROUTERERR_NOTINITIALIZED	Router is not initialized
0x506	1286	ROUTERERR_PORTALREADYINUSE	The desired port number is already assigned
0x507	1287	ROUTERERR_NOTREGISTERED	Port not registered
0x508	1288	ROUTERERR_NOMOREQUEUES	The maximum number of Ports reached
0x509	1289	ROUTERERR_INVALIDPORT	Invalid port
0x50A	1290	ROUTERERR_NOTACTIVATED	TwinCAT Router not active

General ADS Error Codes

Hex	Dec	Name	Description
0x700	1792	ADSERR_DEVICE_ERROR	error class <device error>
0x701	1793	ADSERR_DEVICE_SRVNOTSUPP	Service is not supported by server
0x702	1794	ADSERR_DEVICE_INVALIDGRP	invalid index group
0x703	1795	ADSERR_DEVICE_INVALIDOFFSET	invalid index offset
0x704	1796	ADSERR_DEVICE_INVALIDACCESS	reading/writing not permitted
0x705	1797	ADSERR_DEVICE_INVALIDSIZE	parameter size not correct
0x706	1798	ADSERR_DEVICE_INVALIDDATA	invalid parameter value(s)
0x707	1799	ADSERR_DEVICE_NOTREADY	device is not in a ready state
0x708	1800	ADSERR_DEVICE_BUSY	device is busy
0x709	1801	ADSERR_DEVICE_INVALIDCONTEXT	invalid context (must be in Windows)
0x70A	1802	ADSERR_DEVICE_NOMEMORY	out of memory
0x70B	1803	ADSERR_DEVICE_INVALIDPARM	invalid parameter value(s)
0x70C	1804	ADSERR_DEVICE_NOTFOUND	not found (files, ...)
0x70D	1805	ADSERR_DEVICE_SYNTAX	syntax error in command or file
0x70E	1806	ADSERR_DEVICE_INCOMPATIBLE	objects do not match
0x70F	1807	ADSERR_DEVICE_EXISTS	object already exists
0x710	1808	ADSERR_DEVICE_SYMBOLNOTFOUND	symbol not found
0x711	1809	ADSERR_DEVICE_SYMBOLVERSIONINVAL	symbol version invalid
0x712	1810	ADSERR_DEVICE_INVALIDSTATE	server is in invalid state
0x713	1811	ADSERR_DEVICE_TRANSMODENOTSUPP	AdsTransMode not supported
0x714	1812	ADSERR_DEVICE_NOTIFYHNDINVALID	Notification handle is invalid
0x715	1813	ADSERR_DEVICE_CLIENTUNKNOWN	Notification client not registered
0x716	1814	ADSERR_DEVICE_NOMOREHDLS	no more notification handles
0x717	1815	ADSERR_DEVICE_INVALIDWATCHSIZE	size for watch too big
0x718	1816	ADSERR_DEVICE_NOTINIT	device not initialized
0x719	1817	ADSERR_DEVICE_TIMEOUT	device has a timeout
0x71A	1818	ADSERR_DEVICE_NOINTERFACE	query interface failed
0x71B	1819	ADSERR_DEVICE_INVALIDINTERFACE	wrong interface required
0x71C	1820	ADSERR_DEVICE_INVALIDCLSID	class ID is invalid
0x71D	1821	ADSERR_DEVICE_INVALIDOBJID	object ID is invalid
0x71E	1822	ADSERR_DEVICE_PENDING	request is pending
0x71F	1823	ADSERR_DEVICE_ABORTED	request is aborted
0x720	1824	ADSERR_DEVICE_WARNING	signal warning
0x721	1825	ADSERR_DEVICE_INVALIDARRAYIDX	invalid array index
0x722	1826	ADSERR_DEVICE_SYMBOLNOTACTIVE	symbol not active
0x723	1827	ADSERR_DEVICE_ACCESSDENIED	access denied
0x724	1828	ADSERR_DEVICE_LICENSENOTFOUND	missing license
0x725	1829	ADSERR_DEVICE_LICENSEEXPIRED	license expired
0x726	1830	ADSERR_DEVICE_LICENSEEXCEEDED	license exceeded
0x727	1831	ADSERR_DEVICE_LICENSEINVALID	license invalid
0x728	1832	ADSERR_DEVICE_LICENSESYSTEMID	license invalid system id
0x729	1833	ADSERR_DEVICE_LICENSENOTIMELIMIT	license not time limited
0x72A	1834	ADSERR_DEVICE_LICENSEFUTUREISSUE	license issue time in the future
0x72B	1835	ADSERR_DEVICE_LICENSETIMETOLONG	license time period to long
0x72c	1836	ADSERR_DEVICE_EXCEPTION	exception occured during system start
0x72D	1837	ADSERR_DEVICE_LICENSEDUPLICATED	License file read twice
0x72E	1838	ADSERR_DEVICE_SIGNATUREINVALID	invalid signature
0x72F	1839	ADSERR_DEVICE_CERTIFICATEINVALID	public key certificate
0x740	1856	ADSERR_CLIENT_ERROR	Error class <client error>
0x741	1857	ADSERR_CLIENT_INVALIDPARM	invalid parameter at service
0x742	1858	ADSERR_CLIENT_LISTEMPTY	polling list is empty
0x743	1859	ADSERR_CLIENT_VARUSED	var connection already in use
0x744	1860	ADSERR_CLIENT_DUPLINVOKEID	invoke ID in use
0x745	1861	ADSERR_CLIENT_SYNCTIMEOUT	timeout elapsed
0x746	1862	ADSERR_CLIENT_W32ERROR	error in win32 subsystem
0x747	1863	ADSERR_CLIENT_TIMEOUTINVALID	Invalid client timeout value

Hex	Dec	Name	Description
0x748	1864	ADSERR_CLIENT_PORTNOTOPEN	ads-port not opened
0x750	1872	ADSERR_CLIENT_NOAMSADDR	internal error in ads sync
0x751	1873	ADSERR_CLIENT_SYNCINTERNAL	hash table overflow
0x752	1874	ADSERR_CLIENT_ADDHASH	key not found in hash
0x753	1875	ADSERR_CLIENT_REMOVEHASH	no more symbols in cache
0x754	1876	ADSERR_CLIENT_NOMORESYM	invalid response received
0x755	1877	ADSERR_CLIENT_SYNCRESINVALID	sync port is locked

RTime Error Codes

Hex	Dec	Name	Description
0x1000	4096	RTERR_INTERNAL	Internal fatal error in the TwinCAT real-time system
0x1001	4097	RTERR_BADTIMERPERIODS	Timer value not vaild
0x1002	4098	RTERR_INVALIDTASKPTR	Task pointer has the invalid value ZERO
0x1003	4099	RTERR_INVALIDSTACKPTR	Task stack pointer has the invalid value ZERO
0x1004	4100	RTERR_PRIOEXISTS	The demand task priority is already assigned
0x1005	4101	RTERR_NOMORETCB	No more free TCB (Task Control Block) available. Maximum number of TCBs is 64
0x1006	4102	RTERR_NOMORESEMAS	No more free semaphores available. Maximum number of semaphores is 64
0x1007	4103	RTERR_NOMOREQUEUES	No more free queue available. Maximum number of queue is 64
0x100D	4109	RTERR_EXTIRQALREADYDEF	An external synchronization interrupt is already applied
0x100E	4110	RTERR_EXTIRQNOTDEF	No external synchronization interrupt applied
0x100F	4111	RTERR_EXTIRQINSTALLFAILED	The apply of the external synchronization interrupt failed
0x1010	4112	RTERR_IRQLNOTLESSOREQUAL	Call of a service function in the wrong context
0x1017	4119	RTERR_VMXNOTSUPPORTED	Intel VT-x extension is not supported
0x1018	4120	RTERR_VMXDISABLED	Intel VT-x extension is not enabled in system BIOS
0x1019	4121	RTERR_VMXCONTROLSMISSING	Missing function in Intel VT-x extension
0x101A	4122	RTERR_VMXENABLEFAILS	Enabling Intel VT-x fails

TCP Winsock Error Codes

Hex	Dec	Description
0x274d	10061	A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond.
0x2751	10065	No connection could be made because the target machine actively refused it. This error normally occurs when you try to connect to a service which is inactive on a different host - a service without a server application.
0x274c	10060	No route to a host. A socket operation was attempted to an unreachable host
		Further Winsock error codes: Win32 Error Codes

7.7 Overview of NC errors

Error code (hex)	Description
0x4000 - 0x4FFF: NC error code range	
0x40nn	General errors [▶ 314]
0x41nn	Channel Errors [▶ 317]
0x42nn	Group Errors [▶ 321]
0x43nn	Axis Errors [▶ 340]
0x44nn	Encoder Errors [▶ 347]
0x45nn	Controller Errors [▶ 353]
0x46nn	Drive Errors [▶ 357]
0x4Ann	Table Errors [▶ 362]
0x4Bnn	NC PLC errors [▶ 366]
0x4Cnn	Kinematic Transformation [▶ 372]
0x8000 … 0x8FFF: New extended NC error code range	
0x81nn - 0x811F	Bode plot (diagnosis) [▶ 372]
0x8120 - 0x8FFF	further errors [▶ 375]

7.7.1 General NC Errors

Error(Hex)	Error(Dec)	Error type	Description
4000	**16384**	internal	**"Internal error"** Internal system error in the NC on ring 0, no further details.
4001	**16385**	memory	**"Memory error"** The ring-0 memory management is not providing the required memory. This is usually a result of another error, as a result of which the controller will halt normal operation (now if not before).
4002	**16386**	internal	**"Nc retain data error (persistent data)"** Error while loading the Nc retain data. The axes concerned are no longer referenced (status flag "Homed" is set to FALSE). Possible reasons are: - Nc retain data not found - Nc retain data expired (old backup data) - Nc retain data corrupt or inconsistent
4010	**16400**	parameter	**"Channel identifier not allowed"** Either an unacceptable value (not 1...255) has been used, or a channel that does not exist in the system has been named.
4011	**16401**	parameter	**"Group identifier not allowed"** Either an unacceptable value (not 1...255) has been used, or a group that does not exist in the system has been named.
4012	**16402**	parameter	**"Axis identifier not allowed"** Either an unacceptable value (not 1...255) has been used, or an axis that does not exist in the system has been named.
4013	**16403**	parameter	**"Encoder identifier not allowed"** Either an unacceptable value (not 1...255) has been used, or a encoder that does not exist in the system has been named.
4014	**16404**	parameter	**"Controller identifier not allowed"** Either an unacceptable value (not 1...255) has been used, or a controller that does not exist in the system has been named.
4015	**16405**	parameter	**"Drive identifier not allowed"** Either an unacceptable value (not 1...255) has been used, or a drive that does not exist in the system has been named.
4016	**16406**	parameter	**"Table identifier not allowed"** Either an unacceptable value (not 1...255) has been used, or a table that does not exist in the system has been named.
4020	**16416**	internal	**"No process image"** No PLC-axis interface during creation of an axis.
4021	**16417**	internal	**"No process image"** No axis-PLC interface during creation of an axis.
4022	**16418**	internal	**"No process image"** No encoder-I/O interface during creation of an axis.
4023	**16419**	internal	**"No process image"** No I/O-encoder interface during creation of an axis.
4024	**16420**	internal	**"No process image"** No drive-I/O interface during creation of an axis.
4025	**16421**	internal	**"No process image"** No I/O-drive interface during creation of an axis.
4030	**16432**	internal	**"Coupling type not allowed"** Unacceptable master/slave coupling type.
4031	**16433**	internal	**"Axis type not allowed"** Unacceptable type specification during creation of an axis.

Error(Hex)	Error(Dec)	Error type	Description
4040	**16448**	internal	**"Axis is incompatible"** Axis is not suitable for the intended purpose. A high speed/low speed axis, for example, cannot function as a slave in an axis coupling.
4050	**16464**	internal	**"Channel not ready for operation"** The channel is not complete, and is therefore not ready for operation. This is usually a consequence of problems at system start-up.
4051	**16465**	internal	**"Group not ready for operation"** The group is not complete, and is therefore not ready for operation. This is usually a consequence of problems at system start-up.
4052	**16466**	internal	**"Axis not ready for operation"** The axis is not complete, and is therefore not ready for operation. This is usually a consequence of problems at system start-up.
4060	**16480**	internal	**"Channel exists"** The channel that is to be created already exists.
4061	**16481**	internal	**"Group exists"** The group that is to be created already exists.
4062	**16482**	internal	**"Axis exists"** The axis that is to be created already exists.
4063	**16483**	internal	**"Table exists"** The table that is to be created already exists, resp. it is tried internally to use an already existing table id (e.g. for the universal flying saw).
4070	**16496**	internal	**"Axis index not allowed"** The location within the channel specified for an axis is not allowed.
4071	**16497**	internal	**"Axis index not allowed"** The location within the group specified for an axis is not allowed.

7.7.2 Channel Errors

Error(Hex)	Error(Dec)	Error type	Description
4101	**16641**	Parameter	**"Group index not allowed"** The location within the channel specified for a group is not allowed.
4102	**16642**	Address	**"Null pointer"** The pointer to the group is invalid. This is usually a consequence of an error at system start-up.
4103	**16643**	Internal	**"No process image"** It is not possible to exchange data with the PLC. Possible causes: n the channel does not have an interface (no interpreter present) n The connection to the PLC is faulty
4104	**16644**	Parameter	**"M-function index not allowed"** Unacceptable M-function (not 0...159) detected at the execution level.
4105	**16645**	Memory	**"No memory"** No more system memory is available. This is usually the result of another error.
4106	**16646**	Function	**"Not ready"** The function is not presently available, because a similar function is already being processed. This is usually the result of access conflicts: more than one instance wants to issue commands to the channel. This can, for example, be the consequence of an incorrect PLC program.
4107	**16647**	Function	**"Function/command not supported"** A requested function or command is not supported by the channel.
4108	**16648**	Parameter	**"Invalid parameter while starting"** Parameters to start the channel (TwinCAT-Start) are invalid. Typically there is an invalid memory size or channel type requested.
4109	**16649**	Function	**"Channel function/command not executable"** A channel function e.g. interpreter start is not executable because the channel is already busy, no program is loaded or in an error state.
410A	**16650**	Function	**"ItpGoAhead not executable"** The requested command is not executable, because the interpreter is not executing a decoder stop.
4110	**16656**	Parameter	**"Error opening a file"** The specified file does not exist. Sample: NC program unknown.
4111	**16657**	NC programming	**"Syntax error during loading"** The NC has found a syntax error when loading an NC program.
4112	**16658**	NC programming	**"Syntax error during interpretation"** The NC has found a syntax error when executing an NC program.
4113	**16659**	NC programming	**"Missing subroutine"** The NC has found a missing subroutine while loading.
4114	**16660**	Memory	**"Loading buffer of interpreter is too small"** The capacity of the interpreter loading buffer has been exceeded.
4115	**16661**	Internal	“**Symbolic**” - reserved
4116	**16662**	Internal	“**Symbolic**” - reserved
4117	**16663**	NC programming	**"Subroutine incomplete"** Header of subroutine is missing
4118	**16664**	NC programming	**"Error while loading the NC program"** The maximum number of loadable NC programs has been reached. Possible cause: Too many sub-programs were loaded from a main program.

Error(Hex)	Error(Dec)	Error type	Description
4119	**16665**	NC programming	**"Error while loading the NC program"** The program name is too long.
4120	**16672**	NC programming	**"Divide by zero"** The NC encountered a computation error during execution: division by 0.
4121	**16673**	NC programming	**"Invalid circle parameterization"** The NC encountered a computation error during execution: The specified circle cannot be calculated.
4122	**16674**	NC programming	**"Invalid FPU-Operation"** The NC encountered an invalid FPU-Operation during execution. This error occurs e.g. by calculating the square root of a negative number.
4130	**16688**	NC programming	**"Stack overflow: subroutines"** The NC encountered a stack overflow during execution: too many subroutine levels.
4131	**16689**	NC programming	**"Stack underflow: subroutines"** The NC encountered a stack underflow during execution: too many subroutine return commands. Note: A main program must not end with a return command.
4132	**16690**	NC programming	**"Stack overflow: arithmetic unit"** The NC encountered a stack overflow during execution: The calculation is too complex, or has not been correctly written.
4133	**16691**	NC programming	**"Stack underflow: arithmetic unit"** The NC encountered a stack underflow during execution: The calculation is too complex, or has not been correctly written.
4140	**16704**	Parameter	**"Register index not allowed"** The NC encountered an unacceptable register index during execution: Either the program contains an unacceptable value (not R0...R999) or a pointer register contains an unacceptable value.
4141	**16705**	NC programming	**"Unacceptable G-function index"** The NC has encountered an unacceptable G-function (not 0...159) during execution.
4142	**16706**	NC programming	**"Unacceptable M-function index"** The NC has encountered an unacceptable M-function (not 0...159) during execution.
4143	**16707**	NC programming	**"Unacceptable extended address"** The NC has encountered an unacceptable extended address (not 1...9) during execution.
4144	**16708**	NC programming	**"Unacceptable index to the internal H-function"** The NC has encountered an unacceptable internal H-function in the course of processing. This is usually a consequence of an error during loading.
4145	**16709**	Parameter	**"Machine data value unacceptable"** While processing instructions the NC has detected an impermissible value for the machine data (MDB) (not 0…7).
4150	**16720**	Parameter	**"Cannot change tool params here"** The NC has encountered an unacceptable change of parameters for the tool compensation during execution. This error occurred for instance by changing the tool radius and programming a circle in the same block.
4151	**16721**	Parameter	**"Cannot calculate tool compensation"** The NC has encountered an error by the calculation of the tool compensation.

Error(Hex)	Error(Dec)	Error type	Description
4152	**16722**	NC programming	**Tool compensation:** The plane for the tool compensation cannot be changed here. This error occurred for instance by changing the tool plane when the compensation is turned on or active.
4153	**16723**	NC programming	**Tool compensation:** The D-Word is missing or invalid by turning on the tool compensation.
4154	**16724**	NC programming	**Tool compensation:** The specified tool radius is invalid because the value is less or equal zero.
4155	**16725**	NC programming	**Tool compensation:** The tool radius cannot be changed here
4156	**16726**	Internal	**Tool compensation:** Collision Detection Table is full.
4157	**16727**	Internal	**Tool compensation:** Internal error while turning on the contour collision detection.
4158	**16728**	Internal	**Tool compensation:** Internal error within the contour collision detection: update reversed geo failed.
4159	**16729**	NC programming	**Tool compensation:** Unexpected combination of geometry types by active contour collision detection.
415A	**16730**	NC programming	**Tool compensation:** Programmed inner circle is smaller than the cutter radius
415B	**16731**	NC programming	**Tool compensation:** Bottle neck detection recognized contour violation
415C	**16732**	Memory	Table for corrected entries is full
415D	**16733**	Memory	Input table for tangential following is full
415E	**16734**	Memory	Executing table for tangential following is full
415F	**16735**	Internal	Geometric entry for tangential following cannot be calculated
4160	**16736**	Internal	reserved
4161	**16737**	Internal	reserved
4162	**16738**	Parameter	The actual active interpolation rules (g-code), zero-shifts, or rotation cannot be detected
4170	**16752**	NC programming	**"Error while loading: Invalid parameter"** The NC has found an invalid parameter while loading an NC program.
4171	**16753**	Internal	**"Invalid contour start position"** The NC encountered a computation error during execution: The specified contour cannot be calculated because the initial position is not on the contour.
4172	**16754**	Internal	**"Retrace: Invalid internal entry index"** The NC encountered an invalid internal entry index during execution of the retrace function.

7.7.3 Group Errors

Error(Hex)	Error(Dec)	Error type	Description	
4200	**16896**	Parameter	**"Group ID not allowed"** The value for the group ID is not allowed, e.g. because it has already been assigned, is less than or equal to zero, or is greater than 255.	
			Value range: [1 ... 255]	Unit: 1
4201	**16897**	Parameter	**"Group type not allowed"** The value for the group type is unacceptable because it is not defined. Type 1: PTP group with slaves (servo) Type 4: DXD group with slaves (3D group) Type 5: High/low speed group Type 6: Stepper motor group Type 9: Encoder group with slaves (servo) …	
			Value range: [1 ... 12]	Unit: 1
4202	**16898**	Initialization	**"Master axis index not allowed"** The value for the master axis index in an interpolating 3D group is not allowed, because, for instance, it has gone outside the value range. Index 0: X axis (first master axis) Index 1: Y axis (second master axis) Index 2 : Z axis (third master axis)	
			Value range: [0, 1, 2]	Unit: 1
4203	**16899**	Initialization	**"Slave axis index not allowed" (INTERNAL ERROR)** The value for the slave axis index in a group is not allowed, because, for instance, it has passed outside the value range, the slave location to be used when inserting a new slave connection is already occupied, or because no slave is present when such a connection is being removed. Index 0: First slave axis Index 1: Second slave axis Index 2: etc.	
			Value range: [0 ... 7]	Unit: 1
4204	**16900**	Initialization	**"INTERNAL ERROR"** (GROUPERR_INTERNAL)	
4205	**16901**	Parameter	**"Invalid cycle time for statement execution task (SAF)"** The value of the cycle time for the NC block execution task (SAF 1/2) is not allowed, because it has passed outside the value range.	
			Value range: [0.001 ... 0.1]	Unit: s
4206	**16902**	Initialization	"GROUPERR_RANGE_MAXELEMENTSINAXIS "	
4207	**16903**	Parameter	**"Invalid cycle time for the statement preparation task (SVB)"** The value of the cycle time for the NC statement preparation task (SVB 1/2) is not allowed, because it has passed outside the value range.	
			Value range: [0.001 ... 1.0]	Unit: s
4208	**16904**	Parameter	**"Single step mode not allowed"** The flag for the activation or deactivation of single step mode is not allowed. Value 0: Passive (buffered operation) Value 1: Active (single-block operation)	
			Value range: [0, 1]	Unit: 1
4209	**16905**	Parameter	**"Group deactivation not allowed" (INTERNAL ERROR)** The flag for the deactivation or activation of the complete group is not allowed. Value 0: Group active Value 1: Group passive	
			Value range: [0, 1]	Unit: 1

Error(Hex)	Error(Dec)	Error type	Description
420A	**16906**	Initialization	**"Statement execution state (SAF state) not allowed" (INTERNAL ERROR)** The value for the state of the block execution state machine (SAF state) is not allowed. This error occurs on passing outside the range of values, or if the state machine enters an error state. Value range: [0 ... 5] Unit: 1
420B	**16907**	Address	**"Channel address"** The group does not have a channel, or the channel address has not been initialized.
420C	**16908**	Address	**"Axis address (master axis)"** The group does not have a master axis (or axes) or the axis address(es) has (have) not been initialized.
420D	**16909**	Address	**"Master axis address"** A new master/slave coupling is to be inserted into the group, but there is no valid address for the leading master axis.
420E	**16910**	Address	**"Slave axis address"** A master/slave coupling is to be inserted into the group, but there is no valid address for the slave axis.
420F	**16911**	Address	**"Slave set value generator address"** A master/slave coupling is to be inserted into the group, but there is no valid address for the slave set value generator.
4210	**16912**	Address	**"Encoder address"** An axis in the group does not have an encoder, or the encoder address has not been initialized.
4211	**16913**	Address	**"Controller address"** An axis in the group does not have a controller, or the controller address has not been initialized.
4212	**16914**	Address	**"Drive address"** An axis in the group does not have a drive, or the drive address has not been initialized.
4213	**16915**	Address	**"GROUPERR_ADDR_MASTERGENERATOR"**
4214	**16916**	Address	**"Axis interface NC to PLC address"** Group/axis does not have an axis interface from the NC to the PLC, or the axis interface address has not been initialized.
4215	**16917**	Address	**"Slave axis address"** An existing master/slave coupling is to be removed from the group, but there is no valid address for the slave axis.
4216	**16918**	Address	**"Table address unknown"** The table, respectively the table ID, is unknown. This table is used for the master/slave coupling or for the characteristic curve.
4217	**16919**	Address	**"NcControl address"** The NcControl address has not been initialized.
4218	**16920**	Initialization	**"Axis is blocked for commands while persistent NC data are queued"** Axis is blocked for commands while waiting for valid IO data to accept the queued persistent NC data.
4219	**16921**	Function	**"The scaling mode MASTER-AUTOOFFSET is invalid because no reference table was found".** The used scaling mode MASTER-AUTOOFFSET is invalid in this context because an existing reference table is missing. This error can occur for example when adding cam tables without a unique reference to an existing cam table.
421A	**16922**	Parameter	**"The master axis start position does not permit synchronization"** When a slave axis is being coupled on, the position of the master axis does not permit synchronization at the given synchronization positions.

Error(Hex)	Error(Dec)	Error type	Description
421B	**16923**	Parameter	**"Slave coupling factor (gearing factor) of 0.0 is not allowed"** A master/slave coupling with a gearing factor of 0.0 is being created. This value is not allowed, since it does not correspond to any possible coupling, and division will generate an FPU exception.
421C	**16924**	Function	**"Insertion of master axis into group not allowed"** A master axis is to be inserted into a group at a location that is already occupied by another master axis. Maybe the reconfiguration cannot be done, because this axis has got an existing slave coupling. This master/slave coupling must be revoked before.
421D	**16925**	Function	**"Deletion of master axis from group not allowed" (INTERNAL ERROR)** A master axis is to be removed from a location in a group that is not in fact occupied by master axis.
421E	**16926**	Function	**"Function/feature is not supported from the setpoint generator** A function or feature is not supported from the setpoint generator (e.g. PTP master setpoint generator). This can be in general or only in a special situation.
421F	**16927**	Initialization	**"Group initialization"** Group has not been initialized. Although the group has been created, the rest of the initialization has not been performed (1. Initialization of group I/O, 2. Initialization of group, 3. Reset group).
4220	**16928**	Monitoring	**"Group not ready / group not ready for new task"** The group is being given a new task while it is still in the process of executing an existing task. This request is not allowed because it would interrupt the execution of the previous task. The new task could, for instance, be a positioning command, or the "set actual position" function. Precisely the converse relationships apply for the "set new end position" function. In that case, the group/axis must still be actively moving in order to be able to cause a change in the end position.
4221	**16929**	Monitoring	**"Requested set velocity is not allowed"** The value requested for the set velocity of a positioning task is less than or equal to zero, larger than the "maximum velocity" (see axis parameters), or, in the case of servo-drives, is larger than the "reference velocity" of the axis (see drive parameters).
4222	**16930**	Monitoring	**"Requested target position is not allowed (master axis)"** The requested value for the target position of a positioning task is not within the software end locations. In other words, it is either less than the minimum software end location or larger than the maximum software end location. This check is only carried out if the relevant end position monitoring is active.
4223	**16931**	Monitoring	**"No enable for controller and/or feed (Master axis)"** The axis enables for the master axis needed for positioning are not present. This can involve the controller enable and/or the relevant, direction-dependent feed enable (see axis interface PlcToNc).

Error(Hex)	Error(Dec)	Error type	Description
4224	**16932**	Monitoring	**"Movement smaller than one encoder increment" (INTERNAL ERROR)** The distance that a group/axis is supposed to move is smaller than the physical significance of one encoder increment. In other words the movement is smaller than the scaling factor of the axis. The reaction to this is that the axis is reported as having logically finished without having actively moved. This means that an external error is not generated for the user. This error is also issued for high/low speed axes if a loop movement with nonzero parameters is smaller than the sum of the creeping and braking distances. In such a case it is not meaningful to either exceed or to fail to reach the target position.
4225	**16933**	Monitoring	**"Drive not ready during axis start"** During an axis start it is ascertained that the drive is not ready. The following are possible causes: - the drive is in the error state (hardware error) - the drive is in the start-up phase (e.g. after an axis reset that was preceded by a hardware error) - the drive is missing the controller enable (ENABLE) Note: The time required for "booting" a drive after a hardware fault can amount to several seconds.
4226	**16934**	Monitoring	**"Invalid parameters of the emergency stop."** Either, both, the deceleration and the jerk are less than zero or one of the parameters is weaker than the corresponding parameter of the start data.
4227	**16935**	Function	**"The setpoint generator is inactive such that no instructions are accepted."**
4228	**16936**	Monitoring	**"Requested traverse distance is not allowed"** The requested traverse distance or looping distance is smaller than the braking distance of the two/speed axis.
4229	**16937**	Monitoring	**"Requested target position is not allowed (slave axis)"** The value for the target position of a positioning task when calculated for the slave axis is not within the software end locations. In other words, it is either less than the minimum software end location or larger than the maximum software end location. This check is only carried out if the relevant end position monitoring is active.
422A	**16938**	Monitoring	**"No enable for controller and/or feed (slave axis)"** The axis enables for one or more coupled slave axes needed for positioning are not present. This can involve the controller enable and/or the relevant, direction-dependent feed enable (see axis interface PlcToNc).
422B	**16939**	Parameter	**"The activation position (position threshold) is out of range of the actual positioning"** The activation position (position threshold) of a new axis command (e.g. "new velocity activated at a position") is out of range. E.g. the activation position is before the actual position or behind the target position.
422C	**16940**	Parameter	**"The start or activation data of the external setpoint generation are not valid"** This may be caused through: 1. The external setpoint generation is active and a new activation with a start type (1: absolute, 2: relative) unequal to the current one is send. 2. The internal setpoint generation is active (e.g. PTP) and the external one is activated with the type absolute (two setpoint generators of the type absolute are not possible).
422D	**16941**	Parameter	**"Velocity is not constant"** For changing the dynamic parameter 'acceleration' und 'deceleration' the axis has to be in dynamic state without acceleration and deceleration (that means constant velocity).

Error(Hex)	Error(Dec)	Error type	Description
422E	**16942**	Parameter	**"Jerk less than or equal to 0.0 is not allowed"** A value less than or equal to 0.0 for the jerk (PTP and CNC) is not allowed, since the jerk is by definition positive, and with a jerk of 0.0, division will generate an FPU exception.
422F	**16943**	Parameter	**"Acceleration less than or equal to 0.0 is not allowed"** A value less than or equal to 0.0 for the acceleration (PTP and CNC) is not allowed, since the acceleration is positive by definition, and an acceleration of 0.0 will not allow a motion to be generated.
4230	**16944**	Parameter	**"Absolute deceleration value less than or equal to 0.0 is not allowed"** A value less than or equal to 0.0 for the absolute value of the deceleration (PTP and CNC) is not allowed, since the absolute value of the deceleration is positive by definition, and an absolute value of the deceleration of 0.0 will not allow a motion to be generated.
4231	**16945**	Parameter	**"Set velocity less than or equal to 0.0 is not allowed"** A value less than or equal to 0.0 or outside the range from 10^{-3} up to 10^{+10} for the set velocity (PTP and CNC) is not allowed, since the set velocity is by definition strictly positive, and with a set velocity of 0.0, division will generate an FPU exception.
4232	**16946**	Monitoring	**"Loss of precision when trying a positioning"** The positioning is so long in space or time that decimal parts loose there relevance LOSS_OF_PRECISION).
4233	**16947**	Parameter	**"Cycle time less than or equal to 0.0 is not allowed"** A value less than or equal to 0.0 for the cycle time (PTP and CNC) is not allowed, since the cycle time is by definition strictly positive, and with a cycle time of 0.0, division will generate an FPU exception.
4234	**16948**	Internal	**"PTP data type <intasdouble> range exceeded"** Such extreme parameters have been supplied for the start task, the override or the new target position that the internal data type loses its precision.
4235	**16949**	Function	**"PTP LHL velocity profile cannot be generated" (INTERNAL ERROR)** Such extreme parameters have been supplied for the start task, the override or the new target position that it is not possible to generate a velocity profile of the type LHL (Low-High-Low).
4236	**16950**	Function	**"PTP HML velocity profile cannot be generated" (INTERNAL ERROR)** Such extreme parameters have been supplied for the override or the new target position that it is not possible to generate a velocity profile of the type HML (High-Middle-Low).
4237	**16951**	Address	**"Start data address is invalid"** The address of the start data is invalid.
4238	**16952**	Parameter	**"Velocity override (start override) is not allowed"** The value for the velocity override is not allowed, because it is less than 0.0% or more than 100.0% (see axis interface PlcToNc). Here, 100.0 % corresponds to the integral value 1000000 in the axis interface. Value range: [0 ... 1000000]
4239	**16953**	Parameter	**"Start type not allowed"** The start type supplied does not exist.
423A	**16954**	Monitoring	**"Velocity overflow (overshoot in the velocity)"** The new dynamic with the parameterized jerk is so weak that a velocity overflow will occur (overshoot in the velocity). The command is therefore not supported.

Error(Hex)	Error(Dec)	Error type	Description
423B	**16955**	Parameter	**"Start parameter for the axis structure is invalid"** External or internal parameters for the start structure for a positioning task are invalid. Thus, for instance, the scaling factor, the SAF cycle time or the requested velocity may be less than or equal to zero, which is not allowed.
423C	**16956**	Parameter	**"Override generator initialization parameter invalid"** One of the override generator (re)initialization parameters is invalid.
423D	**16957**	Monitoring	**"Slave axis has not set value generator" (INTERNAL ERROR)** It is found that a slave axis within a group does not have a valid slave generator (set value generator). A slave axis and a slave set value generator must always be present as a pair. This is an internal error.
423E	**16958**	Function	**"Table is empty"** Either the SVB table or the SAF table does not contain any entries.
423F	**16959**	Function	**"Table is full"** The SVB table or the SAF table has no more free lines.
4240	**16960**	Memory	**"No memory available"** SVB memory allocation for dynamic entry in SAF table failed.
4241	**16961**	Function	**"Table already contains an entry" (INTERNAL ERROR)** SAF table entry abandoned, because, incorrectly, an entry already exists.
4242	**16962**	Function	**"Stop is already active"** The stop instruction is not forwarded, because it has already been activated.
4243	**16963**	Function	**"Compensation has not been carried out over the full compensation section"** The compensations start parameters do not permit compensation over the full section to be compensated. For this reason the compensation will be carried out over a smaller section.
4244	**16964**	Parameter	**"Internal parameters for the compensation are invalid" (INTERNAL ERROR)** Invalid internal parameters or start parameters of the lower-level generator.
4245	**16965**	Function	**"Compensation active"** Start of compensation refused, because compensation is already active. It's also possible that the M/S axes are not active moved. Therefore an execution of the compensation is impossible.
4246	**16966**	Function	**"Compensation not active"** Stop of compensation refused, because compensation is not active.
4247	**16967**	Function	**"Compensation type invalid"** The type supplied for the section compensation is invalid. At the present time only compensation type 1 (trapezoidal velocity profile) is allowed.
4248	**16968**	Function	**"Axis address for compensation invalid" (INTERNAL ERROR)** The address of the master of slave axis on which the section compensation is to act is invalid. This is an internal error.
4249	**16969**	Address	**"Invalid slave address" (INTERNAL ERROR)** The slave address given for on-line coupling/decoupling is invalid.
424A	**16970**	Function	**"Coupling velocity invalid"** The velocity of what is to become the master axis is 0, which means that on-line coupling is not possible.
424B	**16971**	Function	**"Coupling velocities not constant"** The velocity of what is to become the master axis and the velocity of what is to become the slave axis are not constant, so that on-line coupling is not possible.

Error(Hex)	Error(Dec)	Error type	Description
424C	**16972**	Parameter	**"Cycle time less than or equal to 0.0 is not allowed"** A value less than or equal to 0.0 for the cycle time (Slave) is not allowed, since the cycle time is by definition strictly positive, and with a cycle time of 0.0, division will generate an FPU exception.
424D	**16973**	Function	**"Decoupling task not allowed"** The slave axis is of such a type (e.g. a table slave) or is in such a state (master velocity 0) that on-line decoupling is not possible.
424E	**16974**	Function	**"Function not allowed"** The function cannot logically be executed, e.g. some commands are not possible and not allowed for slave axes.
424F	**16975**	Parameter	**"No valid table weighting has been set"** The weighting factor of each table is 0, so that no table can be read.
4250	**16976**	Function	**"Axis type, actual position type or end position type is not allowed"** The start type for a positioning task in invalid. Valid start types are ABSOLUTE (1), RELATIVE (2), CONTINUOUS POSITIVE (3), CONTINUOUS NEGATIVE (4), MODULO (5), etc. It is also possible that the types for setting a new actual position or for travel to a new end position are invalid.
4251	**16977**	Function	**"Function not presently supported"** An NC function has been activated that is currently not released for use, or which is not even implemented. This can be a command which is not possible or not allowed for master axes.
4252	**16978**	Monitoring	**"State of state machine invalid" (INTERNAL ERROR)** The state of an internal state machine is invalid. This is an internal error.
4253	**16979**	Monitoring	**"Reference cam became free too soon"** During the referencing process for an axis it is moved in the direction of the referencing cam, and is only stopped again when the cam signal is reached. After the axis has then also physically stopped, the referencing cam must remain occupied until the axis subsequently starts back down from the cam in the normal way.
4254	**16980**	Monitoring	**"Clearance monitoring between activation of the hardware latch and appearance of the sync pulse"** When the clearance monitoring is active, a check is kept on whether the number of increments between activation of the hardware latch and occurrence of the sync pulse (zero pulse) has become smaller than a pre-set value. This error is generated when that happens. (See parameters for the incremental encoder)
4255	**16981**	Memory	**"No memory available"** The dynamic memory allocation for the set value generator, the SVB table or the SAF table has failed.
4256	**16982**	Monitoring	**"The table slave axis has no active table"** Although the table slave axis has tables, none of the tables is designated as active. If this occurs during the run time the whole master/slave group is stopped by a run time error.
4257	**16983**	Function	**"Function not allowed"** The requested function or the requested task is not logically allowed. An example for such an error message would be "set an actual position" for an absolute encoder (M3000, KL5001, etc.).
4258	**16984**	Function	**"Stopping compensation not allowed"** It is not possible to stop the compensation, since compensation is already in the stopping phase.
4259	**16985**	Function	**"Slave table is being used"** The slave table cannot be activated, because it is currently being used.

Error(Hex)	Error(Dec)	Error type	Description
425A	**16986**	Function	**"Master or slave axis is processing a job (e.g. positioning command) while coupling is requested"** A master/slave coupling of a certain slave type (e.g. linear coupling) cannot be executed. he master or intended slave axis is not in stand still state and is executing a job (e.g. positioning) at the same time as the coupling request received. For this couple type this is not allowed.
425B	**16987**	Parameter	**"Slave (start) parameter is incorrect"** One of the slave start/coupling parameters is not allowed (Coupling factor is zero, the master position scaling of an cam is zero, etc.).
425C	**16988**	Parameter	**"Slave type is incorrect"** The slave type does not match up to the (SVB) start type.
425D	**16989**	Function	**"Axis stop is already active"** The axis stop/Estop is not initiated, because the stop/estop is already active.
425E	**16990**	Function	**"Maximum number of tables per slavegenerator reached"** The maximum number of tables per slave generator is reached (e.g. "MC_MultiCamIn" is limited to 4 tables).
425F	**16991**	Function	**"The scaling mode is invalid".** The used scaling is invalid in this context. Either the mode is not defined or yet not implemented or however it cannot in this constellation be put into action. For example MASTER-AUTOOFFSET cannot be used when a cam table is coupled in relative mode because this is a contradiction. Further MASTER-AUTOOFFSET cannot be used when a cam table is coupled for the first time because a relationship to an existing reference table is missing.
4260	**16992**	Monitoring	**"Controller enable"** Controller enable for the axis or for a coupled slave axis is not present (see axis interface PlcToNc). This error occurs if the controller enable is withdrawn while an axis or a group of axes (also a master/ slave group) is being actively positioned. The error also occurs if a PTP axis or a coupled slave axis is started without controller enable.
4261	**16993**	Function	**"Table not found"** No table exists with the ID prescribed or the table ID is not unique.
4262	**16994**	Function	**"Incorrect table type"** The table referred to in the function is of the incorrect type.
4263	**16995**	Function	**"Single step mode"** This error occurs if single step mode is selected for a group or axis and a new task is requested while one of the individual tasks is still being processed.
4264	**16996**	Function	**"Group task unknown (asynchronous table entry)"** The group has received a task whose type or sub-type is unknown. Valid tasks can be single or multi-dimensional positioning tasks (Geo 1D, Geo 3D), referencing tasks, etc.
4265	**16997**	Function	**"Group function unknown (synchronous function)"** The group has received a function whose type is unknown. Valid functions are "Reset", "Stop", "New end position", "Start/stop section compensation", "Set actual position", "Set/reset referencing status" etc.
4266	**16998**	Function	**"Group task for slave not allowed"** Group tasks are usually only possible for master axes, not for slave axes. A slave axis only moves as an indirect result of a positioning task given to its associated master axis. A slave can thus never directly be given a task. Exception: see axis parameter "Allow motion commands to slave axis".

Error(Hex)	Error(Dec)	Error type	Description
4267	**16999**	Function	**"Group function for slave not allowed"** Group functions are in principle only possible for master axes, not for slave axes. The only exception is represented by the "Start/stop section compensation" function, which is possible both for masters and for slaves. A slave cannot directly execute any other functions beyond this.
4268	**17000**	Function	**"GROUPERR_GROUPFUNC_NOMOTION"**
4269	**17001**	Parameter	**"Startposition=Setpoint Position"** Invalid position parameters.
426A	**17002**	Parameter	**"Parameters of the delay-generator are invalid"** Invalid external/internal parameters of the delay generator (delay time, cycle time, tics).
426B	**17003**	Parameter	**"External parameters of the superimposed instruction are invalid"** Invalid external parameters of the superimposed functionality (acceleration, deceleration, velocity, process velocity, length).
426C	**17004**	Parameter	**"Invalid override type."**
426D	**17005**	Function	**"Activation position under/overrun"** The requested activation position is located in the past of the master (e.g. when exchanging a cam table).
426E	**17006**	Function	**"Activation impossible: Master is standing"** The required activation of the correction is impossible since the master axis is not moving. A synchronization is not possible, because the master axis standing and the slave axis is still not synchronous.
426F	**17007**	Function	**"Activation mode not possible"** The requested activation mode is not possible when the slave axis is moving. Otherwise the slave velocity would jump to zero.
4270	**17008**	Parameter	**"Start parameter for the compensation is invalid"** One of the dynamic parameters for the compensation is invalid (necessary condition): Acceleration (>0) Deceleration (>0) Process velocity (>0)
4271	**17009**	Parameter	**"Start parameter for the compensation is invalid"** Velocity camber is negative.
4272	**17010**	Parameter	**"Start parameter for the compensation is invalid"** The section on which the compensation is to occur is not positive.
4273	**17011**	Monitoring	**"Target position under/overrun" (INTERNAL ERROR)** The position (calculated from the modulo-target-position) where the axis should stand at end of oriented stop has been run over.
4274	**17012**	Monitoring	**"Target position will be under/overrun" (INTERNAL ERROR)** The position (calculated from the modulo-target-position) where the axis should stand at end of oriented stop is too near and will be run over.
4276	**17014**	Monitoring	**"GROUPERR_GUIDERSTARTDATA"**
4277	**17015**	Monitoring	**"Dynamic parameters not permitted" (INTERNAL ERROR)** The dynamic parameters resulting from internal calculation like acceleration, deceleration and jerk are not permitted.
4279	**17017**	Monitoring	**"GROUPERR_GUIDEROVERRUN"**
427A	**17018**	Monitoring	**"GROUPERR_GUIDERLOOKAHEAD"**
427B	**17019**	Monitoring	**"GROUPERR_GUIDERLOOKAHEADEND"**
427C	**17020**	Monitoring	**"GROUPERR_GUIDERLOOKAHEADREQU"**
427D	**17021**	Monitoring	**"GROUPERR_GUIDERMODE"**

Error(Hex)	Error(Dec)	Error type	Description
427E	**17022**	Monitoring	**"A requested motion command could not be realized (BISECTION)"** A requested motion command could not be realized using the requested parameters. The command has been executed best possible and this message is therefore to be understood just as a warning. Samples: An axis motion command is requested while the axis is in a unfavorable dynamic situation (acceleration phase), in which the covered distance is too short or the velocity is clearly too high. Another possibility is a slave axis, which is decoupled in motion in an unfavorable dynamic situation and is afterwards given a motion as in the previous case.
427F	**17023**	Monitoring	**"The new target position either has been overrun or will be overrun"** The new target position either has been overrun or will be overrun, since until there it is impossible to stop. An internal stop command is commended.
4280	**17024**	Monitoring	**"Group not ready / group not ready for new task" (INTERNAL ERROR / INFORMATION)** The group is being given a new task while it is still in the process of executing an existing task. This request is not allowed because it would interrupt the execution of the previous task. The new task could, for instance, be a positioning command, or the "set actual position" function. Precisely the converse relationships apply for the "set new end position" function. In that case, the group/axis must still be actively moving in order to be able to cause a change in the end position.
4281	**17025**	Parameter	**"The parameters of the oriented stop (O-Stop) are not admitted."** The modulo-target position should not be smaller than zero and not larger or equal than the encoder mod-period (e.g. in the interval [0.0,360.0]). **Note:** In the case of error the axis is safely stopped, but is afterwards not at the requested oriented position.
4282	**17026**	Monitoring	**"The modulo target position of the modulo-start is invalid"** The modulo target position is outside of the valid parameter range. So the position value should not be smaller than zero and not greater or equal than the encoder modulo-period (e. g. in the interval [0.0,360.0] for the modulo start type "SHORTEST_WAY (261)").
4283	**17027**	Parameter	**"The online change activation mode is invalid".** The activation can be used with online scaling or with online modification of motion function. The used activation is invalid in this context. Either the mode is not defined or yet not implemented or however it cannot in this constellation be put into action (e.g. when linear tables are used with an unexpected cyclic activation mode NEXTCYCLE or NEXTCYCLEONCE). In some case, the activation mode may be valid but the command cannot be executed due to a pending previous command.
4284	**17028**	Parameter	**"The parameterized jerk rate is not permitted".** The jerk rate is smaller than the minimum jerk rate. The minimum value for jerk rate is 1.0 (e.g. mm/s^3).
4285	**17029**	Parameter	**"The parameterized acceleration or deceleration is not permitted".** The parameterized acceleration or deceleration is lower than the permitted minimum acceleration. The value for minimum acceleration is calculated from minimum jerk rate and NC cycle time (minimum jerk rate multiplied with NC cycle time). The unit for example is mm/s^2.

Error(Hex)	Error(Dec)	Error type	Description
4286	**17030**	Parameter	**"The parameterized velocity is not permitted".** The parameterized target velocity is lower than the minimum velocity (but the value zero is permitted). The value for minimum velocity is calculated from the minimum jerk rate and the NC cycle time (minimum jerk rate multiplied with the square of the NC cycle time). The unit for example is mm/s.
4287	**17031**	Monitoring	**"A activation cannot be executed due to a pending activation"** A activation e.g. "CamIn", "CamScaling" or "WriteMotionFunction" cannot be executed due to a pending activation (e.g. "CamIn", "CamScaling", "WriteMotionFunction"). Only activation can be enabled.
4288	**17032**	Monitoring	**"Illegal combination of different cycle times within an axis group"** A logical axis group includes elements (axes) with different cycle times for a common setpoint generator and I/O-execution, resp. This situation can occur with Master/Slave-coupling or configuring 3D- and FIFO-groups (including path, auxiliary, and slave axes).
4289	**17033**	Monitoring	**"Illegal motion reversal"** Due to the actual dynamical state a motion reversal will happen. To avoid this motion reversal the axis command is not performed and the previous system state restored.
428A	**17034**	Monitoring	**"Illegal moment for an axis command because there is an old axis command with activation position still active"** The moment for the command is illegal because there is still an old command with activation position active (e.g. "go to new velocity at threshold position" or "reach new velocity at threshold position").
428B	**17035**	Monitoring	**"Error in the stop-calculation routine" (INTERNAL ERROR)** Due to an internal error in the stop-calculation routine the current commando cannot be performed. The previous system state is restored.
428C	**17036**	Monitoring	**"A command with activation position cannot fully be performed because the remaining path is too short"**A command with activation position (threshold) like "reach a new velocity at a position" can be just partially executed because the path from the actual position to the activation position is too short.
428D	**17037**	Monitoring	**"Illegal decouple type when decoupling a slave axis"** The decouple and restart command contains an invalid decouple type.
428E	**17038**	Monitoring	**"Illegal target velocity when decoupling a slave axis"** The decouple and restart command contains an illegal target velocity [1 < V <Vmax].
428F	**17039**	Monitoring	**"The command new dynamic parameter cannot be performed since this would require a new target velocity"**Das Kommando zum Aktivieren neuer Dynamikparameter wie Beschleunigung, Verzögerung und Ruck kann nicht durchgeführt werden, da dies eine neue beauftragte Fahrgeschwindigkeit erfordern würde. This situation can occur, for example, if the axis is near the target position in an accelerated state and the dynamics parameter are chosen softer.
4290	**17040**	Monitoring	**"A command with activation position cannot be performed because the axis is already in the brake phase"** A command with activation position (threshold) e.g. "reach new velocity at position" cannot be performed because the axis is already in the brake phase and the remaining path from the actual position to the activation position is too short.

Error(Hex)	Error(Dec)	Error type	Description
4291	**17041**	Monitoring	**"Decouple routine of slave axis doesn't return a valid solution"** Internal jerk scaling of decouple routine cannot evaluate a valid solution (decoupling slave axis and transform to master axis). The command is rejected because velocity can become too high, a reversal of movement can occur, or the target position can be passed.
4292	**17042**	Monitoring	**"Command not be executed because the command buffer is full filled"** The command is rejected because the command buffer is full filled.
4293	**17043**	Internal	**"Command is rejected due to an internal error in the Look Ahead" (INTERNAL ERROR)** The command is rejected due to an internal error in the "look ahead".
4294	**17044**	Monitoring	**"Command is rejected because the segment target velocity is not realized"** The command is rejected, because the new target segment velocity *Vrequ* is not realizable and an internal optimizing is impossible.
4295	**17045**	Monitoring	**"Successive commands have the same final position"** Successive commands have the same final position. So the moving distance is zero.
4296	**17046**	Monitoring	**"Logical positioning direction is inconsistent with the direction of the buffer command"** In the extended buffer mode, where the actual end position is replaced by the new buffer start position, the logical positioning direction is inconsistent with the direction of the buffer command (=> contradiction). A buffered command (*BufferMode, BlendingLow, BlendingPrevious, BlendingNext, BlendingHigh*) is rejected with error 0x4296 if the command is using the Beckhoff specific *optional BlendingPosition* but the blending position is located beyond the target position of the previous motion command.
4297	**17047**	Monitoring	**"Command is rejected because the remaining positioning length is to small"** The command is rejected because the remaining path length is too small. E.g. when the buffer mode is used and the remaining positioning length in the actual segment is too small for getting the axis in a force free state or to reach the new target velocity at the change of segment.
429B	**17051**	Monitoring	„collect error for invalid start parameters“ This error refers to a wrong parameterization of the user (collect error). E. g. dynamic parameters like Velo, Acc or Dec could be equal or less than zero. Or following errors: - BaseFrequence < 0.0 - StartFrequence < 1.0 - StepCount < 1, StepCount > 200 - BaseAmplitude <= 0.0 - StepDuration <= 0.0 - StopFrequence >= 1/(2*CycleTime)
429C	**17052**	Monitoring	**"Reference cam is not found"** During the referencing process for an axis it is moved in the direction of the referencing cam. This reference cam, however, was not found as expected (=> leads to the abortion of the referencing procedure).

Error(Hex)	Error(Dec)	Error type	Description	
429D	**17053**	Monitoring	**"Reference cam became not free"** During the referencing process for an axis it is moved in the direction of the referencing cam, and is only stopped again when the cam signal is reached. After the axis has also come to a physical standstill, the axis is subsequently started regularly from the cam again. In this case, the reference cam did not become free again as expected when driving down (=> leads to the abortion of the referencing procedure).	
429E	**17054**	Monitoring	**"IO sync pulse was not found (only when using hardware latch)"** If the hardware latch is activated, a sync pulse (zero pulse) is expected to be found and a sync event triggered following the expiry of a certain time or a certain distance. If this is not the case, the reaction is an error and the abortion of the referencing procedure.	
42A0	**17056**	Internal	**"Group/axis consequential error"** Consequential error resulting from another causative error related to another axis within the group. Group/axis consequential errors can occur in relation to master/slave couplings or with multiple axis interpolating DXD groups. If, for instance, it is detected that the following error limit of a master axis has been exceeded, then this consequential error is assigned to all the other master axes and slave axes in this group.	
42A1	**17057**	Parameter	**"Velocity reduction factor for C0/C1 transition is not allowed"** A C0 transition describes two geometries which, while they are themselves continuous, no not have either continuous first or second differentials. The velocity reduction factor C0 acts on such transitions. Note: A C1 transition is characterized by the two geometries being continuous, but having only a first differential that is continuous. The velocity reduction factor C1 acts on such transitions.	
			Value range: [0.0 ... 1.0]	Unit: 1
42A2	**17058**	Parameter	**"Critical angle at segment transition not allowed"**	
			Value range: (0.0 ... 180.0]	Unit: degree
42A3	**17059**	Parameter	**"Radius of the tolerance sphere"** is in an invalid rang	
			Value range: [0.0 ... 100.0]	Unit: e.g. mm
42A4	**17060**	Parameter	Not implemented.	
42A5	**17061**	Parameter	**"Start type"**	
			Value range: [0,1]	Unit: 1
42A6	**17062**	Parameter	Not implemented.	
42A7	**17063**	Parameter	**"Blending"** with given parameters not possible	
42A8	**17064**	Parameter	Not implemented.	
42A9	**17065**	Parameter	**"Curve velocity reduction method not allowed" (INTERNAL ERROR)** The curve velocity reduction method does not exist.	
42AA	**17066**	Parameter	**"Minimum velocity not allowed"** The minimum velocity that has been entered is less than 0.0.	
42AB	**17067**	Parameter	**"Power function input not allowed" (INTERNAL ERROR)** The input parameters in the power_() function lead to an FPU exception.	
42AC	**17068**	Parameter	**"Dynamic change parameter not allowed"** A parameter that controls alterations to the dynamics is invalid. Parameter: 1. Absolute motion dynamics change: All parameters must be strictly positive. 2. Relative reduction c_f: 0.0 < c_f <= 1.0	
42AD	**17069**	Memory	**"Memory allocation error" (INTERNAL ERROR)**	

Error(Hex)	Error(Dec)	Error type	Description
42AE	**17070**	Function	**"The calculated end position differs from the end position in the nc instruction (internal error)."**
42AF	**17071**	Parameter	**"Calculate remaining chord length"** invalid value Value range: [0,1]
42B0	**17072**	Function	**"Set value generator SVB active"** Starting the set value generator (SVB, SAF) has been refused, since the SVB task is already active.
42B1	**17073**	Parameter	**"SVB parameter not allowed" (INTERNAL ERROR)** A parameter related to the internal structure of the set value generator (SVB) results in logical errors and/or to an FPU exception. Affects these parameters: Minimum velocity (>0.0), TimeMode, ModeDyn, ModeGeo, StartType, DistanceToEnd, TBallRadius.
42B2	**17074**	Parameter	**"Velocity reduction factor not allowed"** A parameter that controls reduction of the velocity at segment transitions is invalid. Parameter: 1. Transitions with continuous first differential: VeloVertexFactorC1 2. Not once continuously differentiable transitions: VeloVertexFactorC0, CriticalVertexAngleLow, CriticalVertexAngleHigh.
42B3	**17075**	Parameter	**"Helix is a circle"** The helix has degenerated to a circle, and should be entered as such.
42B4	**17076**	Parameter	**"Helix is a straight line"** The helix has degenerated to a straight line, and should be entered as such.
42B5	**17077**	Parameter	**"Guider parameter not allowed"** One of the guider's parameters leads to logical errors and/or to an FPU exception.
42B6	**17078**	Address	**"Invalid segment address" (INTERNAL ERROR)** The geometry segment does not have a valid geometry structure address or does not have a valid dynamic structure address.
42B7	**17079**	Parameter	**"Unparameterized generator" (INTERNAL ERROR)** The SVB generator is not yet parameterized and is therefore unable to operate.
42B8	**17080**	Address	**"Unparameterized table" (INTERNAL ERROR)** The table has no information concerning the address of the corresponding dynamic generator.
42BA	**17082**	Internal	**"The calculation of the arc length of the smoothed path failed (internal error)."**
42BB	**17083**	Parameter	**"The radius of the tolerance ball is too small (smaller than 0.1 mm)."**
42BC	**17084**	Internal	Error while calculating DXD-Software-Limit switches (internal error)
42BD	**17085**	Function	**"NC-Block violates software limit switches of the group"** At least one path axis with active software limit monitoring has violated the limit switches. Therefore the geometric entry is denied with an error.
42BE	**17086**	Parameter	**"GROUPERR_DXD_SOFTENDCHECK"**
42BF	**17087**	Parameter	**"GROUPERR_DXD_RTTG_VELOREFERENCE"**
42C0	**17088**	Internal	**"Interpolating group contains axes of an incorrect axis type"** An interpolating 3D group may only contain continuously guided axes of axis type 1 (SERVO).
42C1	**17089**	Internal	**"Scalar product cannot be calculated"** The length of one of the given vectors is 0.0.
42C2	**17090**	Internal	**"Inverse cosine cannot be calculated"** The length of one of the given vectors is 0.0.

Error(Hex)	Error(Dec)	Error type	Description
42C3	**17091**	Parameter	**"Invalid table entry type"** The given table entry type is unknown.
42C4	**17092**	Parameter	**"Invalid DIN66025 information type" (INTERNAL ERROR)** The given DIN66025 information type is unknown. Known types: G0, G1, G2, G3, G17, G18, G19.
42C5	**17093**	Parameter	**"Invalid dimension" (INTERNAL ERROR)** The CNC dimension is unknown. Known dimensions: 1, 2, 3. Or: The CNC dimension is invalid for the given geometrical object. For a circle the dimension must be 2 or 3, while for a helix it must be 3.
42C6	**17094**	Parameter	**"Geometrical object is not a straight line"** The given object, interpreted as a straight line, has a length of 0.0.
42C7	**17095**	Parameter	**"Geometrical object is not a circle"** Interpreted as a circular arc, the given object has a length of 0.0, or an angle of 0.0 or a radius of 0.0.
42C8	**17096**	Parameter	**"Geometrical object is not a helix"** Interpreted as a circular arc, the given object has a length of 0.0, or an angle of 0.0, or a radius of 0.0. or a height of 0.0.
42C9	**17097**	Parameter	**"Set velocity less than or equal to 0.0 is invalid"** A value less than or equal to 0.0 for the set velocity (CNC) is not allowed, since the set velocity is positive by definition, and a set velocity of 0.0 cannot generate any motion.
42CA	**17098**	Address	**"Address for look-ahead invalid" (INTERNAL ERROR)** The address supplied for the look-ahead is invalid.
42CB	**17099**	Function	**"Set value generator SAF active"** Starting the set value generator (SAF) has been refused, since the SAF task is already active.
42CC	**17100**	Function	**"CNC set value generation not active"** Stop or change of override refused, because the set value generation is not active.
42CD	**17101**	Function	**"CNC set value generation in the stop phase"** Stop or change of override refused, because the set value generation is in the stop phase.
42CE	**17102**	Parameter	**"Override not allowed"** An override of less than 0.0 % or more than 100.0 % is invalid.
42CF	**17103**	Address	**"Invalid table address" (INTERNAL ERROR)** The table address given for the initialization of the set value generator is invalid, or no valid logger connection (report file) is present.
42D0	**17104**	Parameter	**"Invalid table entry type"** The given table entry type is unknown.
42D1	**17105**	Memory	**"Memory allocation failed"** Memory allocation for the table has failed.
42D2	**17106**	Memory	**"Memory allocation failed"** Memory allocation for the filter has failed.
42D3	**17107**	Parameter	**"Invalid parameter"** Filter parameter is not allowed.
42D4	**17108**	Function	**"Delete Distance To Go failed"** Delete Distance to go (only interpolation) failed. This error occurred, if e.g. the command 'DelDTG' was not programmed in the actual movement of the nc program.
42D5	**17109**	Internal	**"The setpoint generator of the flying saw generates incompatible values (internal error)"**

Error(Hex)	Error(Dec)	Error type	Description
42D6	**17110**	Function	**"Axis will be stopped since otherwise it will overrun its target position (old PTP setpoint generator)"** If, for example, in case of a slave to master transformation for the new master a target position is commanded that will be overrun because of the actual dynamics the axis will be stopped internally to guarantee that the target position will not be overrun.
42D7	**17111**	Function	**"Internal error in the transformation from slave to master."**
42D8	**17112**	Function	**"Wrong direction in the transformation of slave to master."**
42DA	**17114**	Parameter	**"Parameter of Motion Function (MF) table incorrect"** The parameter of the Motion Function (MF) are invalid. This may refer to the first time created data set or to online changed data.
42DB	**17115**	Parameter	**"Parameter of Motion Function (MF) table incorrect"** The parameter of the Motion Function (MF) are invalid. This may refer to the first time created data set or to online changed data. The error cause can be, that an active MF point (no IGNORE point) points at a passive MF point (IGNORE point).
42DC	**17116**	Monitoring	**"Internal error by using Motion Function (MF)"** An internal error occurs by using the Function (MF). This error cannot be solved by the user. Please ask the TwinCAT Support.
42DD	**17117**	Function	**"Axis coupling with synchronization generator declined because of incorrect axis dynamic values"** The axis coupling with the synchronization generator has been declined, because one of the slave dynamic parameter (machine data) is incorrect. Either the maximum velocity, the acceleration, the deceleration or the jerk is smaller or equal to zero, or the expected synchronous velocity of the slave axis is higher as the maximum allowed slave velocity.
42DE	**17118**	Function	**"Coupling conditions of synchronization generator incorrect"** During positive motion of the master axis it has to be considered, that the master synchronous position is larger than the master coupling position ("to be in the future"). During negative motion of the master axis it has to be considered that the master synchronous position is smaller than the master coupling position.
42DF	**17119**	Monitoring	**"Moving profile of synchronization generator declines dynamic limit of slave axis or required characteristic of profile"** One of the parameterized checks has recognized an overstepping of the dynamic limits (max. velocity, max. acceleration, max. deceleration or max. jerk) of the slave axis, or an profile characteristic (e.g. overshoot or undershoot in the position or velocity) is incorrect. See also further messages in the windows event log and in the message window of the System Manager.
42E0	**17120**	Parameter	**"Invalid parameter"** The encoder generator parameter is not allowed.
42E1	**17121**	Parameter	**"Invalid parameter"** The external (Fifo) generator parameter is not allowed.
42E2	**17122**	Function	**"External generator is active"** The external generator cannot be started, as it is already active.
42E3	**17123**	Function	**"External generator is not active"** The external generator cannot be stopped, as it is not active.

Error(Hex)	Error(Dec)	Error type	Description
42E4	**17124**	Function	**"NC-Block with auxiliary axis violates software limit switches of the group"** At least one auxiliary axis with active software limit monitoring has violated the limit switches. Therefore the geometric entry is denied with an error.
42E5	**17125**	Function	**"NC-Block type Bezier spline curve contains a cusp (singularity)"** The Bezier spline curve contain a cusp, i.e. at a certain interior point both the curvature and the modulus of the velocity tend to 0 such that the radius of curvature is infinite. Note: Split the Bezier curve at that point into two Bezier spline curves according to the de "Casteljau algorithm". This preserves the geometry and eliminates the interior singularity.
42E7	**17127**	Parameter	**"Value for dead time compensation not allowed"** The value for the dead time compensation in seconds for a slave coupling to an encoder axis (virtual axis) is not allowed.
			Value range: [0.0 ... 60.0] — Unit: s
42E8	**17128**	Parameter	**"GROUPERR_RANGE_NOMOTIONWINDOW"**
			Value range: [0.0 ... 1000.0] — Unit: e.g. mm/s
42E9	**17129**	Parameter	**"GROUPERR_RANGE_NOMOTIONFILTERTIME"**
			Value range: [0.0 ... 60.0] — Unit: s
42EA	**17130**	Parameter	**"GROUPERR_RANGE_TIMEUNITFIFO"**
			Value range: (0.0 ... 1000.0] — Unit: s
42EB	**17131**	Parameter	**"GROUPERR_RANGE_OVERRIDETYPE"**
			Value range: [1, 2] — Unit: 1
42EC	**17132**	Parameter	**"GROUPERR_RANGE_OVERRIDECHANGETIME"**
			Value range: (0.0 ... 1000.0] — Unit: s
42ED	**17133**	Parameter	**"GROUPERR_FIFO_INVALIDDIMENSION"** **Note:** Since TC 2.11 Build 1547 the FIFO-dimension (number of axes) has been increased from 8 to 16.
			Value range: [1 ... 8] resp. [1 ... 16] — Unit: 1 (number of axes)
42EE	**17134**	Address	**"GROUPERR_ADDR_FIFOTABLE"**
42EF	**17135**	Monitoring	**"Axis is locked for motion commands because a stop command is still active"** The axis/group is locked for motion commands because a stop command is still active. The axis can be released by calling *MC_Stop* with Execute=FALSE or by using *MC_Reset* (*TcMC2.Lib*).
42F0	**17136**	Parameter	**"Invalid number of auxiliary axes"** The local number of auxiliary axes does not tally with the global number of auxiliary axes.
42F1	**17137**	Parameter	**"Invalid reduction parameter for auxiliary axes"** The velocity reduction parameters for the auxiliary axes are inconsistent.
42F2	**17138**	Parameter	**"Invalid dynamic parameter for auxiliary axes"** The dynamic parameters for the auxiliary axes are inconsistent.
42F3	**17139**	Parameter	**"Invalid coupling parameter for auxiliary axes"** The coupling parameters for the auxiliary axes are inconsistent.
42F4	**17140**	Parameter	**"Invalid auxiliary axis entry"** The auxiliary axis entry is empty (no axis motion).
42F6	**17142**	Parameter	**"Invalid parameter"** The limit for velocity reduction of the auxiliary axes is invalid. It has to be in the interval 0..1.0

Error(Hex)	Error(Dec)	Error type	Description
42F8	**17144**	Parameter	**"Block search - segment not found"** The segment specified as a parameter could not be found by the end of the NC program. Possible cause: • nBlockId is not specified in the mode described by eBlockSearchMode
42F9	**17145**	Parameter	**"Blocksearch – invalid remaining segment length" The remaining travel in the parameter fLength is incorrectly parameterized**
42FB	**17147**	Monitoring	**"INTERNAL ERROR"** (GROUPERR_SLAVE_INTERNAL)
42FF	**17151**	Monitoring	**„GROUPERR_WATCHDOG“** (Customer specific error code)

7.7.4 Axis Errors

Error(Hex)	Error(Dec)	Error type	Description	
4300	**17152**	Parameter	**"Axis ID not allowed"** The value for the axis ID is not allowed, e.g. because it has already been assigned, is less than or equal to zero, is greater than 255, or does not exist in the current configuration.	
			Value range: [1 ... 255]	Unit: 1
4301	**17153**	Parameter	**"Axis type not allowed"** The value for the axis type is unacceptable because it is not defined. Type 1: Servo Type 2: Fast/creep Type 3: Stepper motor	
			Value range: [1 ... 3]	Unit: 1
4306	**17158**	Parameter	**"Slow manual velocity not allowed"** The value for the slow manual velocity is not allowed.	
			Value range: [0.0, 10000.0]	Unit: e.g. m/min
4307	**17159**	Parameter	**"Fast manual velocity not allowed"** The value for the fast manual velocity is not allowed.	
			Value range: [0.0, 10000.0]	Unit: e.g. m/min
4308	**17160**	Parameter	**"High speed not allowed"** The value for the high speed is not allowed.	
			Value range: [0.0, 10000.0]	Unit: e.g. m/min
4309	**17161**	Parameter	**"Acceleration not allowed"** The value for the axis acceleration is not allowed.	
			Value range: [0.0, 1000000.0]	Unit: e.g. m/s/s
430A	**17162**	Parameter	**"Deceleration not allowed"** The value for the axis deceleration is not allowed.	
			Value range: [0.0, 1000000.0]	Unit: e.g. m/s/s
430B	**17163**	Parameter	**"Jerk not allowed"** The value for the axis jerk is not allowed.	
			Value range: [0.0, 1000000.0]	Unit: e.g. m/s/s/s
430C	**17164**	Parameter	**"Delay time between position and velocity is not allowed"** The value for the delay time between position and velocity ("idle time compensation") is not allowed.	
			Value range: [0, 0.1]	Unit: s
430D	**17165**	Parameter	**"Override-Type not allowed"** The value for the velocity override type is not allowed. Type 1: With respect to the internal reduced velocity (default value) Type 2: With respect to the original external start velocity	
			Value range: [1 ... 4]	Unit: 1
430E	**17166**	Parameter	**"NCI: Velo-Jump-Factor not allowed"** The value for the velo-jump-factor ("VeloJumpFactor") is not allowed. This parameter only works for TwinCAT NCI.	
			Value range: [0, 1000000]	Unit: 1
430F	**17167**	Parameter	**"NCI: Radius of tolerance sphere for the auxiliary axes is invalid"** It was tried to enter an invalid value for the size of the tolerance sphere. This sphere affects only auxiliary axes!	
			Value range: [0, 1000]	Unit: e.g. mm
4310	**17168**	Parameter	**"NCI: Value for maximum deviation for the auxiliary axes is invalid"** It was tried to enter an invalid value for the maximum allowed deviation. This parameter affects only auxiliary axes!	
			Value range: [0, 10000]	Unit: e.g. mm

Error(Hex)	Error(Dec)	Error type	Description	
4312	**17170**	Parameter	**"Referencing velocity in direction of cam not allowed"** The value for the referencing velocity in the direction of the referencing cam is not allowed.	
			Value range: [0.0, 10000.0]	Unit: e.g. m/min
4313	**17171**	Parameter	**"Referencing velocity in sync direction not allowed"** The value for the referencing velocity in the direction of the sync pulse (zero track) is not allowed.	
			Value range: [0.0, 10000.0]	Unit: e.g. m/min
4314	**17172**	Parameter	**"Pulse width in positive direction not allowed"** The value for the pulse width in the positive direction is not allowed (pulsed operation). The use of the pulse width for positioning is chosen implicitly through the axis start type. Pulsed operation corresponds to positioning with a relative displacement that corresponds precisely to the pulse width.	
			Value range: [0.0, 1000000.0]	Unit: e.g. mm
4315	**17173**	Parameter	**"Pulse width in negative direction not allowed"** The value for the pulse width in the negative direction is not allowed (pulsed operation). The use of the pulse width for positioning is chosen implicitly through the axis start type. Pulsed operation corresponds to positioning with a relative displacement that corresponds precisely to the pulse width.	
			Value range: [0.0, 1000000.0	Unit: e.g. mm
4316	**17174**	Parameter	**"Pulse time in positive direction not allowed"** The value for the pulse width in the positive direction is not allowed (pulsed operation).	
			Value range: [0.0, 600.0]	Unit: s
4317	**17175**	Parameter	**"Pulse time in negative direction not allowed"** The value for the pulse width in the negative direction is not allowed (pulsed operation).	
			Value range: [0.0, 600.0]	Unit: s
4318	**17176**	Parameter	**"Creep distance in positive direction not allowed"** The value for the creep distance in the positive direction is not allowed.	
			Value range: [0.0, 100000.0]	Unit: e.g. mm
4319	**17177**	Parameter	**"Creep distance in negative direction not allowed"** The value for the creep distance in the negative direction is not allowed.	
			Value range: [0.0, 100000.0]	Unit: e.g. mm
431A	**17178**	Parameter	**"Braking distance in positive direction not allowed"** The value for the braking distance in the positive direction is not allowed.	
			Value range: [0.0, 100000.0]	Unit: e.g. mm
431B	**17179**	Parameter	**"Braking distance in negative direction not allowed"** The value for the braking distance in the negative direction is not allowed.	
			Value range: [0.0, 100000.0]	Unit: e.g. mm
431C	**17180**	Parameter	**"Braking time in positive direction not allowed"** The value for the braking time in the positive direction is not allowed.	
			Value range: [0.0, 60.0]	Unit: s

Error(Hex)	Error(Dec)	Error type	Description	
431D	**17181**	Parameter	**"Braking time in negative direction not allowed"** The value for the braking time in the negative direction is not allowed.	
			Value range: [0.0, 60.0]	Unit: s
431E	**17182**	Parameter	**"Switching time from high to low speed not allowed"** The value for the time to switch from high to low speed is not allowed.	
			Value range: [0.0, 60.0]	Unit: s
431F	**17183**	Parameter	**"Creep distance for stop not allowed"** The value for the creep distance for an explicit stop is not allowed.	
			Value range: [0.0, 100000.0]	Unit: e.g. mm
4320	**17184**	Parameter	**"Motion monitoring not allowed"** The value for the activation of the motion monitoring is not allowed.	
			Value range: [0, 1]	Unit: 1
4321	**17185**	Parameter	**"Position window monitoring not allowed"** The value for the activation of the position window monitoring is not allowed.	
			Value range: [0, 1]	Unit: 1
4322	**17186**	Parameter	**"Target window monitoring not allowed"** The value for the activation of target window monitoring is not allowed.	
			Value range: [0, 1]	Unit: 1
4323	**17187**	Parameter	**"Loop not allowed"** The value for the activation of loop movement is not allowed.	
			Value range: [0, 1]	Unit: 1
4324	**17188**	Parameter	**"Motion monitoring time not allowed"** The value for the motion monitoring time is not allowed.	
			Value range: [0.0, 600.0]	Unit: s
4325	**17189**	Parameter	**"Target window range not allowed"** The value for the target window is not allowed.	
			Value range: [0.0, 10000.0]	Unit: e.g. mm
4326	**17190**	Parameter	**"Position window range not allowed"** The value for the position window is not allowed.	
			Value range: [0.0, 10000.0]	Unit: e.g. mm
4327	**17191**	Parameter	**"Position window monitoring time not allowed"** The value for the position window monitoring time is not allowed.	
			Value range: [0.0, 600.0]	Unit: s
4328	**17192**	Parameter	**"Loop movement not allowed"** The value for the loop movement is not allowed.	
			Value range: [0.0, 10000.0]	Unit: e.g. mm
4329	**17193**	Parameter	**"Axis cycle time not allowed"** The value for the axis cycle time is not allowed.	
			Value range: [0.001, 0.1]	Unit: s
432A	**17194**	Parameter	**"Stepper motor operating mode not allowed"** The value for the stepper motor operating mode is not allowed.	
			Value range: [1, 2]	Unit: 1
432B	**17195**	Parameter	**"Displacement per stepper motor step not allowed"** The value for the displacement associated with one step of the stepper motor is not allowed (step scaling).	
			Value range: [0.000001, 1000.0]	Unit: e.g. mm/ STEP

Error(Hex)	Error(Dec)	Error type	Description
432C	**17196**	Parameter	**"Minimum speed for stepper motor set value profile not allowed"** The value for the minimum speed of the stepper motor speed profile is not allowed.
			Value range: [0.0, 1000.0] — Unit: z. B. m/min
432D	**17197**	Parameter	**"Stepper motor stages for one speed stage not allowed"** The value for the number of steps for each speed stage in the set value generation is not allowed.
			Value range: [0, 100] — Unit: 1
432E	**17198**	Parameter	**"DWORD for the interpretation of the axis units not allowed"** The value that contains the flags for the interpretation of the position and velocity units is not allowed.
			Value range: [0, 0xFFFFFFFF] — Unit: 1
432F	**17199**	Parameter	**"Maximum velocity not allowed"** The value for the maximum permitted velocity is not allowed.
			Value range: [0.0, 10000.0] — Unit: e.g. m/min
4330	**17200**	Parameter	**"Motion monitoring window not allowed"** The value for the motion monitoring window is not allowed.
			Value range: [0.0, 10000.0] — Unit: e.g. mm
4331	**17201**	Parameter	**"PEH time monitoring not allowed"** The value for the activation of the PEH time monitoring is not allowed (PEH: positioning end and halt).
			Value range: [0, 1] — Unit: 1
4332	**17202**	Parameter	**"PEH monitoring time not allowed"** The value for the PEH monitoring time (timeout) is not allowed (PEH: positioning end and halt). default value: 5s
			Value range: [0.0, 600.0] — Unit: s
4333	**17203**	Parameter	**"AXISERR_RANGE_DELAYBREAKRELEASE"**
4334	**17204**	Parameter	**"AXISERR_RANGE_DATAPERSISTENCE"**
433A	**17210**	Parameter	**"AXISERR_RANGE_POSDIFF_FADING_ACCELERATION"**
433B	**17211**	Parameter	**"Fast Axis Stop Signal Type not allowed"** The value for the Signal Type of the 'Fast Axis Stop' is not allowed [0...5].
4340	**17216**	Initialization	**"Axis initialization"** Axis has not been initialized. Although the axis has been created, the rest of the initialization has not been performed (1. Initialization of axis I/O, 2. Initialization of axis, 3. Reset axis).
4341	**17217**	Address	**"Group address"** Axis does not have a group, or the group address has not been initialized (group contains the set value generation).
4342	**17218**	Address	**"Encoder address"** The axis does not have an encoder, or the encoder address has not been initialized.
4343	**17219**	Address	**"Controller address"** The axis does not have a controller, or the controller address has not been initialized.
4344	**17220**	Address	**"Drive address"** The axis does not have a drive, or the drive address has not been initialized.
4345	**17221**	Address	**"Axis interface PLC to NC address"** Axis does not have an axis interface from the PLC to the NC, or the axis interface address has not been initialized.
4346	**17222**	Address	**"Axis interface NC to PLC address"** Axis does not have an axis interface from the NC to the PLC, or the axis interface address has not been initialized.

Error(Hex)	Error(Dec)	Error type	Description
4347	**17223**	Address	**"Size of axis interface NC to PLC is not allowed" (INTERNAL ERROR)** The size of the axis interface from NC to PLC is not allowed.
4348	**17224**	Address	**"Size of axis interface PLC to NC is not allowed" (INTERNAL ERROR)** The size of the axis interface from PLC to NC is not allowed.
4356	**17238**	Monitoring	**"Controller enable"** Controller enable for the axis is not present (see axis interface SPS®NC). This enable is required, for instance, for an axis positioning task.
4357	**17239**	Monitoring	**"Feed enable minus"** Feed enable for movement in the negative direction is not present (see axis interface SPS®NC). This enable is required, for instance, for an axis positioning task in the negative direction.
4358	**17240**	Monitoring	**"Feed enable plus"** Feed enable for movement in the positive direction is not present (see axis interface SPS®NC). This enable is required, for instance, for an axis positioning task in the positive direction.
4359	**17241**	Monitoring	**"Set velocity not allowed"** The set velocity requested for a positioning task is not allowed. This can happen if the velocity is less than or equal to zero, larger than the maximum permitted axis velocity, or, in the case of servo-drives, is larger than the reference velocity of the axis (see axis and drive parameters).
435A	**17242**	Monitoring	**"Movement smaller than one encoder increment" (INTERNAL ERROR)** The movement required of an axis is, in relation to a positioning task, smaller than one encoder increment (see scaling factor). This information is, however, handled internally in such a way that the positioning is considered to have been completed without an error message being returned.
435B	**17243**	Monitoring	**"Set acceleration monitoring" (INTERNAL ERROR)** The set acceleration has exceeded the maximum permitted acceleration or deceleration parameters of the axis.
435C	**17244**	Monitoring	**"PEH time monitoring"** The PEH time monitoring has detected that, after the PEH monitoring time that follows a positioning has elapsed, the target position window has not been reached. The following points must be checked: Is the PEH monitoring time, in the sense of timeout monitoring, set to a sufficiently large value (e.g. 1-5 s)? The PEH monitoring time must be chosen to be significantly larger than the target position monitoring time. Have the criteria for the target position monitoring (range window and time) been set too strictly? **Note:** The PEH time monitoring only functions when target position monitoring is active!
435D	**17245**	Monitoring	**"Encoder existence monitoring / movement monitoring"** During the active positioning the actual encoder value has changed continuously for a default check time from NC cycle to NC cycle less than the default minimum movement limit. => Check, whether axis is mechanically blocked, or the encoder system failed, etc... **Note:** The check is not performed while the axis is logically standing (position control), but only at active positioning (it would make no sense if there is a mechanical holding brake at the standstill)!

Error(Hex)	Error(Dec)	Error type	Description
435E	**17246**	Monitoring	**"Looping distance less than breaking distance"** The absolute value of the looping distance is less or equal than the positive or negative breaking distance. This is not allowed.
4361	**17249**	Monitoring	**"Time range exceeded (future)"** The calculated position lies too far in the future (e.g. when converting a position value in a DC time stamp).
4362	**17250**	Monitoring	**"Time range exceeded (past)"** The calculated position lies too far in the past (e.g. when converting a position value in a DC time stamp).
4363	**17251**	Monitoring	**"Position cannot be determined"** The requested position cannot be determined. Case 1: It was not passed through in the past. Case 2: It cannot be reached in future. A reason can be a zero velocity value or an acceleration that causes a turn back.
4364	**17252**	Monitoring	**"Position indeterminable (conflicting direction of travel)"** The direction of travel expected by the caller of the function deviates from the actual direction of travel (conflict between PLC and NC view, for example when converting a position to a DC time).
43A0	**17312**	Internal	**"Axis consequential error"** Consequential error resulting from another causative error related to another axis. Axis consequential errors can occur in relation to master/slave couplings or with multiple axis interpolating DXD groups.

7.7.5 Encoder Errors

Error(Hex)	Error(Dez)	Error type	Description	
4400	**17408**	parameter	**"Encoder ID not allowed"** The value for the encoder ID is not allowed, e.g. because it has already been assigned, is less than or equal to zero, or is bigger than 255.	
			Value range: [1 ... 255]	Unit: 1
4401	**17409**	parameter	**"Encoder type not allowed"** The value for the encoder type is unacceptable because it is not defined. Type 1: Simulation (incremental) Type 2: M3000 (24 bit absolute) Type 3: M31x0 (24 bit incremental) Type 4: KL5101 (16 bit incremental) Type 5: KL5001 (24 bit absolute SSI) Type 6: KL5051 (16 bit BISSI)	
			Value range: [1 ... 6]	Unit: 1
4402	**17410**	parameter	**"Encoder mode"** The value for the encoder (operating) mode is not allowed. Mode 1: Determination of the actual position Mode 2: Determination of the actual position and the actual velocity (filter)	
			Value range: [1, 2]	Unit: 1
4403	**17411**	parameter	**"Encoder counting direction inverted?"** The flag for the encoder counting direction is not allowed. Flag 0: Positive encoder counting direction Flag 1: Negative encoder counting direction	
			Value range: [0, 1]	Unit: 1
4404	**17412**	initialization	**"Referencing status"** The flag for the referencing status is not allowed. Flag 0: Axis has not been referenced Flag 1: Axis has been referenced	
			Value range: [0, 1]	Unit: 1
4405	**17413**	parameter	**"Encoder increments for each physical encoder rotation"** The value for the number of encoder increments for each physical rotation of the encoder is not allowed. This value is used by the software for the calculation of encoder overruns and underruns.	
			Value range: [255, 0xFFFFFFFF]	Unit: INC
4406	**17414**	parameter	**"Scaling factor"** The value for the scaling factor is not allowed. This scaling factor provides the weighting for the conversion of an encoder increment (INC) to a physical unit such as millimeters or degrees.	
			Value range: [0.000001, 100.0]	Unit: e.g. mm/INC
4407	**17415**	parameter	**"Position offset (zero point offset)"** The value for the position offset of the encoder is not allowed. This value is added to the calculated encoder position, and is interpreted in the physical units of the encoder.	
			Value range: [-1000000.0, 1000000.0]	Unit: e.g. mm
4408	**17416**	parameter	**"Modulo factor"** The value for the encoder's modulo factor is not allowed.	
			Value range: [1.0, 1000000.0]	Unit: e.g. mm
4409	**17417**	parameter	**"Position filter time"** The value for the actual position filter time is not allowed (P-T1 filter).	
			Value range: [0.0, 60.0]	Unit: s
440A	**17418**	parameter	**"Velocity filter time"** The value for the actual velocity filter time is not allowed (P-T1 filter).	
			Value range: [0.0, 60.0]	Unit: s
440B	**17419**	parameter	**"Acceleration filter time"** The value for the actual acceleration filter time is not allowed (P-T1 filter).	
			Value range: [0.0, 60.0]	Unit: s
440C	**17420**	initialization	**"Cycle time not allowed" (INTERNAL ERROR)** The value of the SAF cycle time for the calculation of actual values is not allowed (e.g. is less than or equal to zero).	

Error(Hex)	Error(Dez)	Error type	Description	
440D	**17421**	initialization	**""** ENCERR_RANGE_UNITFLAGS	
440E	**17422**	parameter	**"Actual position correction / measurement system error correction"** The value for the activation of the actual position correction ("measuring system error correction") is not allowed.	
			Value range: [0, 1]	Unit: 1
440F	**17423**	parameter	**"Filter time actual position correction"** The value for the actual position correction filter time is not allowed (P-T1 filter).	
			Value range: [0.0, 60.0]	Unit: s
4410	**17424**	parameter	**"Search direction for referencing cam inverted"** The value of the search direction of the referencing cam in a referencing procedure is not allowed. Value 0: Positive direction Value 1: Negative direction	
			Value range: [0, 1]	Unit: 1
4411	**17425**	parameter	**"Search direction for sync pulse (zero pulse) inverted"** The value of the search direction of the sync pulse (zero pulse) in a referencing procedure is not allowed. Value 0: Positive direction Value 1: Negative direction	
			Value range: [0, 1]	Unit: 1
4412	**17426**	parameter	**"Reference position"** The value of the reference position in a referencing procedure is not allowed.	
			Value range: [-1000000.0, 1000000.0]	Unit: e.g. mm
4413	**17427**	parameter	**"Clearance monitoring between activation of the hardware latch and appearance of the sync pulse" (NOT IMPLEMENTED)** The flag for the clearance monitoring between activation of the hardware latch and occurrence of the sync/zero pulse ("latch valid") is not allowed. Value 0: Passive Value 1: Active	
			Value range: [0, 1]	Unit: 1
4414	**17428**	parameter	**"Minimum clearance between activation of the hardware latch and appearance of the sync pulse" (NOT IMPLEMENTED)** The value for the minimum clearance in increments between activation of the hardware latch and occurrence of the sync/zero pulse ("latch valid") during a referencing procedure is not allowed.	
			Value range: [0, 65536]	Unit: INC
4415	**17429**	parameter	**"External sync pulse" (NOT IMPLEMENTED)** The value of the activation or deactivation of the external sync pulse in a referencing procedure is not allowed. Value 0: Passive Value 1: Active	
			Value range: [0, 1]	Unit: 1
4416	**17430**	parameter	**"Scaling of the noise rate is not allowed"** The value of the scaling (weighting) of the synthetic noise rate is not allowed. This parameter exists only in the simulation encoder and serves to produce a realistic simulation.	
			Value range: [0, 1000000]	Unit: 1
4417	**17431**	parameter	**„Tolerance window for modulo-start“** The value for the tolerance window for the modulo-axis-start is invalid. The value must be greater or equal than zero and smaller than the half encoder modulo-period (e. g. in the interval [0.0,180.0)).	
			Value range: [0.0, 180], Max: 0.5*modulo-periode	Unit: e. g. mm or degree

Error(Hex)	Error(Dez)	Error type	Description	
4418	**17432**	parameter	**„Encoder reference mode“** The value for the encoder reference mode is not allowed, resp. is not supported for this encoder type.	
			Value range: [0, 5]	Unit: 1
4419	**17433**	parameter	**„Encoder evaluation direction“** The value for the encoder evaluation direction (log. counter direction) is not allowed.	
			Value range: [0, 3]	Unit: 1
441A	**17434**	parameter	**„Encoder reference system“** The value for the encoder reference system is invalid (0: incremental, 1: absolute, 2: absolute+modulo).	
			Value range: [0, 2]	Unit: 1
441B	**17435**	parameter	**„Encoder position initialization mode“** When starting the TC system the value for the encoder position initialization mode is invalid.	
			Value range: [0, 1]	Unit: 1
441C	**17436**	parameter	**„Encoder sign interpretation (UNSIGNED- / SIGNED- data type)“** The value for the encoder sign interpretation (data type) for the encoder the actual increment calculation (0: Default/not defined, 1: UNSIGNED, 2:/ SIGNED) is invalid.	
			Value range: [0, 2]	Unit: 1
4420	**17440**	parameter	**"Software end location monitoring minimum not allowed"** The value for the activation of the software location monitoring minimum is not allowed.	
			Value range: [0, 1]	Unit: 1
4421	**17441**	parameter	**"Software end location monitoring maximum not allowed"** The value for the activation of the software location monitoring maximum is not allowed.	
			Value range: [0, 1]	Unit: 1
4422	**17442**	function	**"Actual value setting is outside the value range"** The "set actual value" function cannot be carried out, because the new actual position is outside the expected range of values.	
			Value range: [-1000000.0, 1000000.0]	Unit: e.g. mm
4423	**17443**	parameter	**"Software end location minimum not allowed"** The value for the software end location minimum is not allowed.	
			Value range: [-1000000000.0, 1000000000.0]	Unit: e.g. mm
4424	**17444**	parameter	**"Software end location maximum not allowed"** The value for the software end location maximum is not allowed.	
			Value range: [-1000000000.0, 1000000000.0]	Unit: e.g. mm
4425	**17445**	parameter	**„Filter mask for the raw data of the encoder is invalid“** The value for the filter mask of the encoder raw data in increments is invalid.	
			Value range: [0, 0xFFFFFFFF]	Unit: 1
4426	**17446**	parameter	**„Reference mask for the raw data of the encoder is invalid“** The value for the reference mask (increments per encoder turn, absolute resolution) for the raw data of the encoder is invalid. E.g. this value is used for axis reference sequence (calibration) with the reference mode "Software Sync".	
			Value range: [0x0000000F, 0xFFFFFFFF]	Unit: 1

Error(Hex)	Error(Dez)	Error type	Description
4430	**17456**	function	**"Hardware latch activation (encoder)"** Activation of the encoder hardware latch was implicitly initiated by the referencing procedure. If this function has already been activated but a latch value has not yet become valid ("latch valid"), another call to the function is refused with this error.
4431	**17457**	function	**"External hardware latch activation (encoder)"** The activation of the external hardware latch (only available on the KL5101) is initiated explicitly by an ADS command (called from the PLC program of the Visual Basic interface). If this function has already been activated, but the latch value has not yet been made valid by an external signal ("external latch valid"), another call to the function is refused with this error.
4432	**17458**	function	**"External hardware latch activation (encoder)"** If a referencing procedure has previously been initiated and the hardware still signals a valid latch value ("latch valid"), this function must not be called. In practice, however, this error can almost never occur.
4433	**17459**	function	**"External hardware latch activation (encoder)"** If this function has already been initiated and the hardware is still signaling that the external latch value is still valid ("extern latch valid"), a further activation should not be carried out and the commando will be declined with an error (the internal handshake communication between NC and IO device is still active). In that case the validity of the external hardware latch would immediately be signaled, although the old latch value would still be present.
4434	**17460**	monitoring	**"Encoder function not supported"** An encoder function has been activated that is currently not released for use, or which is not even implemented.
4435	**17461**	monitoring	**„Encoder function is already active“** An encoder function can not been activated because this functionality is already active.
4440	**17472**	initialization	**"Encoder initialization"** Encoder has not been initialized. Although the axis has been created, the rest of the initialization has not been performed (1. Initialization of axis I/O, 2. Initialization of axis, 3. Reset axis).
4441	**17473**	address	**"Axis address"** The encoder does not have an axis, or the axis address has not been initialized.
4442	**17474**	address	**"I/O input structure address"** The drive does not have a valid I/O input address in the process image.
4443	**17475**	address	**"I/O output structure address"** The encoder does not have a valid I/O output address in the process image.
4450	**17488**	monitoring	**"Encoder counter underflow monitoring"** The encoder's incremental counter has underflowed.
4451	**17489**	monitoring	**"Encoder counter overflow monitoring"** The encoder's incremental counter has overflowed.
4460	**17504**	monitoring	**"Software end location minimum (axis start)"** With active monitoring of the software end location for a minimum, a start has been made from a position that lies below the software end location minimum.
4461	**17505**	monitoring	**"Software end location maximum (axis start)"** With active monitoring of the software end location for a maximum, a start has been made from a position that lies above the software end location maximum.

Error(Hex)	Error(Dez)	Error type	Description
4462	**17506**	monitoring	**"Software end location minimum (positioning process)"** With active monitoring of the software end location for a minimum, the actual position has fallen below the software end location minimum. In the case of servo axes (continuously driven axes) this limit is expanded by the magnitude of the parameterized following error window (position).
4463	**17507**	monitoring	**"Software end location maximum (positioning process)"** With active monitoring of the software end location for a maximum, the actual position has exceeded the software end location maximum. In the case of servo axes (continuously driven axes) this limit is expanded by the magnitude of the parameterized following error window (position).
4464	**17508**	monitoring	**„Encoder hardware error“** The drive resp. the encoder system reports a hardware error of the encoder. An optimal error code is displayed in the message of the event log.
4465	**17509**	monitoring	**„Position initialization error at system start“** At the first initialization of the set position was this for all initialization trials (without over-/under-flow, with underflow and overflow) out of the final position minimum and maximum.
4470	**17520**	monitoring	**"SSI transformation fault or not finished"** The SSI transformation of the FOX 50 module was faulty for some NC-cycles or did not finished respectively.
44A2	**17570**	monitoring	**"ENCERR_ADDR_CONTROLLER"**
44A3	**17571**	monitoring	**"ENCERR_INVALID_CONTROLLERTYPE"**

7.7.6 Controller Errors

Error(Hex)	Error(Dec)	Error type	Description	
4500	**17664**	parameter	**"Controller ID not allowed"** The value for the controller ID is not allowed, e.g. because it has already been assigned, is less than or equal to zero, or is greater than 255.	
			Value range: [1 ... 255]	Unit: 1
4501	**17665**	parameter	**"Controller type not allowed"** The value for the controller type is unacceptable because it is not defined. Type 1: P-controller (position) . . . Type 7: High/low speed controller Type 8: Stepper motor controller Type 9: Sercos controller	
			Value range: [1 ... 8]	Unit: 1
4502	**17666**	parameter	**"Controller operating mode not allowed"** The value for the controller operating mode is not allowed.	
			Value range: [1]	Unit: 1
4503	**17667**	parameter	**"Weighting of the velocity pre-control not allowed"** The value for the percentage weighting of the velocity pre-control is not allowed. The parameter is pre-set to 1.0 (100%) as standard.	
			Value range: [0.0 ... 1.0]	Unit: %
4504	**17668**	parameter	**"Following error monitoring (position) not allowed"** The value for the activation of the following error monitoring is not allowed.	
			Value range: [0, 1]	Unit: 1
4505	**17669**	parameter	**"Following error (velocity) not allowed"** The value for the activation of the following error monitoring (velocity) is not allowed.	
			Value range: [0, 1]	Unit: 1
4506	**17670**	parameter	**"Following error window (position) not allowed"** The value for the following error window (maximum allowable following error) is not allowed.	
			Value range: [0.0, 10000.0]	Unit: e.g. mm
4507	**17671**	parameter	**"Following error filter time (position) not allowed"** The value for the following error filter time (position) is not allowed.	
			Value range: [0.0, 600.0]	Unit: s
4508	**17672**	parameter	**"Following error window (velocity) not allowed"** The value for the following error window (velocity) is not allowed.	
			Value range: [0.0, 10000.0]	Unit: e.g. m/min
4509	**17673**	parameter	**"Following error filter time (velocity) not allowed"** The value for the following error filter time (velocity) is not allowed.	
			Value range: [0.0, 600.0]	Unit: s
4510	**17680**	parameter	**"Proportional gain Kv or Kp (controller) not allowed"** *position* The value for the proportional gain (Kv factor or Kp factor) is not allowed.	
			Value range: [0.0, 10000.0]	Unit: e.g. mm/s/mm
4511	**17681**	parameter	**"Integral-action time Tn (controller) not allowed"** *position* The value for the integral-action time is not allowed (I proportion of the PID T1 controller).	
			Value range: [0.0, 60.0]	Unit: s
4512	**17682**	parameter	**"Derivative action time Tv (controller) not allowed"** *position* The value for the derivative action time is not allowed (D proportion of the PID T1 controller).	
			Value range: [0.0, 60.0]	Unit: s
4513	**17683**	parameter	**"Damping time Td (controller) not allowed"** *position* The value for the damping time is not allowed (D proportion of the PID T1 controller). Suggested value: 0.1 * Tv	
			Value range: [0.0, 60.0]	Unit: s

Error(Hex)	Error(Dec)	Error type	Description	
4514	**17684**	function	**"Activation of the automatic offset compensation not allowed"** Activation of the automatic offset compensation is only possible for certain types of controller (with no I component).	
4515	**17685**	parameter	**"Additional proportional gain Kv or Kp (controller) not allowed"** *position* The value for the second term of the proportional gain (Kv factor or Kp factor) is not allowed.	
			Value range: [0.0, 10000.0]	Unit: e.g. mm/s/mm
4516	**17686**	parameter	**"Reference velocity for additional proportional gain Kv or Kp (controller) not allowed"** *position* The value for the reference velocity percentage data entry, to which the additional proportional gain is applied, is not allowed. The standard setting for the parameter is 0.5 (50%).	
			Value range: [0.0 ... 1.0]	Unit: %
4517	**17687**	parameter	**"Proportional gain Pa (proportion) not allowed"** *acceleration* The value for the proportional gain (Pa factor) is not allowed.	
			Value range: [0.0, 1000000.0]	Unit: s
4518	**17688**	parameter	**"Proportional gain Kv (velocity controller) not allowed"** The value for the proportional gain (Kv factor) is not allowed.	
			Value range: [0.0, 10000.0]	Unit: 1
4519	**17689**	parameter	**“Reset time Tn (velocity controller) not allowed”** The value for the integral-action time is not allowed (I proportion of the PID T1 controller).	
			Value range: [0.0, 60.0]	Unit: s
451A	**17690**	address	**"CONTROLERR_RANGE_ACCJUMPLIMITINGMODE"**	
451B	**17691**	address	**"CONTROLERR_RANGE_ACCJUMPVALUE"**	
451C	**17692**	address	**"CONTROLERR_RANGE_FILTERTIME"**	
451D	**17693**	parameter	**„Dead zone not allowed“** The value for the dead zone from the position error or the velocity error (system deviation) is not allowed (only for complex controller with velocity or torque interface).	
			Value range: [0.0, 10000.0]	Unit: mm resp. mm/s
4520	**17696**	parameter	**”Rate time Tv (velocity controller) not allowed”** The value for the derivative action time is not allowed (D proportion of the PID T1 controller).	
			Value range: [0.0, 60.0]	Unit: s
4521	**17697**	parameter	**"Damping time Td (velocity controller) not allowed"** The value for the damping time is not allowed (D proportion of the PID T1 controller). Suggested value: 0.1 * Tv	
			Value range: [0.0, 60.0]	Unit: s
4522	**17698**	parameter	**"CONTROLERR_RANGE_IOUTPUTLIMIT"**	
4523	**17699**	parameter	**"CONTROLERR_RANGE_DOUTPUTLIMIT"**	
4524	**17700**	parameter	**"CONTROLERR_RANGE_POSIDISABLEWHENMOVING"**	
4540	**17728**	initialization	**"Controller initialization"** Controller has not been initialized. Although the controller has been created, the rest of the initialization has not been performed (1. Initialization of controller, 2. Reset controller).	
4541	**17729**	address	**"Axis address"** Controller does not know its axis, or the axis address has not been initialized.	
4542	**17730**	address	**"Drive address"** Controller does not know its drive, or the drive address has not been initialized.	

Error(Hex)	Error(Dec)	Error type	Description
4550	**17744**	monitoring	**"Following error monitoring (position)"** With active following error monitoring (position) a following error exceedance has occurred, whose magnitude is greater than the following error window, and whose duration is longer than the parameterized following error filter time.
4551	**17745**	monitoring	**"Following error monitoring (velocity)"** With active following error monitoring (velocity) a velocity following error exceedance has occurred, whose magnitude is greater than the following error window, and whose duration is longer than the parameterized following error filter time.
45A0	**17824**	monitoring	**"CONTROLERR_RANGE_AREA_ASIDE"**
45A1	**17825**	monitoring	**"CONTROLERR_RANGE_AREA_BSIDE"**
45A2	**17826**	monitoring	**"CONTROLERR_RANGE_QNENN"**
45A3	**17827**	monitoring	**"CONTROLERR_RANGE_PNENN"**
45A4	**17828**	monitoring	**"CONTROLERR_RANGE_AXISIDPRESP0"**

7.7.7 Drive Errors

Error(hex)	Error(dec)	Error type	Description	
4600	**17920**	Parameter	**‘"Drive ID not allowed"** The value for the drive ID is not allowed, e.g. because it has already been assigned, is less than or equal to zero, or is greater than 255.	
			Value range: [1 ... 255]	Unit: 1
4601	**17921**	Parameter	**‘Drive type impermissible’** The value for the drive type is impermissible, since it is not defined.	
			Value range: [1, 20]	Unit: 1
4602	**17922**	Parameter	**‘Drive operating mode impermissible’** The value for the drive operating mode is impermissible (mode 1: standard).	
			Value range: [1]	Unit: 1
4603	**17923**	Parameter	**"Motor polarity inverted?"** The flag for the motor polarity is not allowed. Flag 0: Positive motor polarity flag 1: Negative motor polarity	
			Value range: [0, 1]	Unit: 1
4604	**17924**	Parameter	**‘Drift compensation/speed offset (DAC offset)’** The value for the drift compensation (DAC offset) is impermissible.	
			Value range: [-100.0, 100.0]	Unit: e.g. m/min
4605	**17925**	Parameter	**‘Reference speed (velocity pre-control)’** The value for the reference speed (also called velocity pilot control) is impermissible.	
			Value range: [0.0, 10000.0]	Unit: e.g. m/min
4606	**17926**	Parameter	**‘Reference output in percent’** The value for the reference output in percent is impermissible. The value 1.0 (100 %) usually corresponds to a voltage of 10.0 V.	
			Value range: [0.0, 5.0]	Unit: %
4607	**17927**	Parameter	**‘Quadrant compensation factor’** The value for the quadrant compensation factor is impermissible.	
			Value range: [0.0, 100.0]	Unit: 1
4608	**17928**	Parameter	**‘Velocity reference point’** The value for the velocity reference point in percent is impermissible. The value 1.0 corresponds to 100 percent.	
			Value range: [0.01, 1.0]	Unit: %
4609	**17929**	Parameter	**‘Output reference point’** The value for the output reference point in percent is impermissible. The value 1.0 corresponds to 100 percent.	
			Value range: [0.01, 1.0]	Unit: %
460A	**17930**	Parameter	**‘Minimum or maximum output limits (output limitation)’** The value for the minimum and/or maximum output limit is impermissible. This will happen if the range of values is exceeded, the maximum limit is smaller than the minimum limit, or the distance between the minimum and maximum limits is zero. The minimum limit is initially set to –1.0 (-100 percent) and the maximum limit to 1.0 (100 percent).	
			Value range: [-1.0, 1.0]	Unit: %
460B	**17931**	Parameter	**“DRIVEERR_RANGE_MAXINCREMENT”**	
460C	**17932**	Parameter	**“DRIVEERR_RANGE_ DRIVECONTROLDWORD”**	
460D	**17933**	Parameter	**“DRIVEERR_RANGE_ RESETCYCLECOUNTER”**	
460F	**17935**	Parameter	**‘Drive torque output scaling impermissible’** The value is impermissible as drive torque output scaling (rotary motor) or as force output scaling (linear motor).	
			Value range: [0, 1000000]	Unit: 1
4610	**17936**	Parameter	**„Drive velocity output scaling is not allowed“** The value for the drive velocity output scaling is not allowed.	
			Value range: [0, 1000000]	Unit: 1

Error(hex)	Error(dec)	Error type	Description	
4611	**17937**	Parameter	**'Profi Drive DSC proportional gain Kpc (controller) impermissible'** *Positions* The value for the Profi Drive DSC position control gain (Kpc factor) is impermissible.	
			Value range: [0, 0xFFFFFF]	Unit: 0.001 * 1/s
4612	**17938**	Parameter	**'Table ID is impermissible'** The value for the table ID is impermissible.	
			Value range: [0, 255]	Unit: 1
4613	**17939**	Parameter	**'Table interpolation type is impermissible'** The value is impermissible as the table interpolation type.	
			Value range: 0 (LINEAR), 2 (SPLINE)	Unit: 1
4614	**17940**	Parameter	**'Output offset in percent is impermissible'** The value is impermissible as an output offset in percent (+/- 1.0).	
			Value range: [-1.0, 1.0]	Unit: %
4615	**17941**	Parameter	**'Profi Drive DSC scaling for calculation of "Xerr" (controller) impermissible'** *Positions:* the value is impermissible as Profi Drive DSC scaling for the calculation of 'Xerr'.	
			Value range: [0, 1000000]	Unit: 1
4616	**17942**	Parameter	**'Drive acceleration output scaling impermissible'** The value is impermissible as drive acceleration/deceleration output scaling.	
			Value range: [0, 1000000]	Unit: 1
4617	**17943**	Parameter	**'Drive position output scaling impermissible'** The value is impermissible as drive position output scaling.	
			Value range: [0, 1000000]	Unit: 1
461C	**17948**	Parameter	**'Drive filter type impermissible for command variable filter for the output position'** The value is impermissible as a drive filter type for the smoothing of the output position (command variable filter for the setpoint position).	
			Value range: [0, 2]	Unit: 1
461D	**17949**	Parameter	**'Drive filter time impermissible for command variable filter for the output position'** The value is impermissible as a drive filter time for the smoothing of the output position (command variable filter for the setpoint position).	
			Value range: [0.0, 1.0]	Unit: s
461E	**17950**	Parameter	**'Drive filter order impermissible for command variable filter for the output position'** The value is impermissible as a drive filter order (P-Tn) for the smoothing of the output position (command variable filter for the setpoint position).	
			Value range: [0, 10]	Unit: 1
4620	**17952**	Parameter	**'Bit mask for stepper motor cycle impermissible'** A value of the different stepper motor masks is impermissible for the respective cycle.	
			Value range: [0, 255]	Unit: 1
4621	**17953**	Parameter	**'Bit mask for stepper motor holding current impermissible'** The value for the stepper motor holding mask is impermissible.	
			Value range: [0, 255]	Unit: 1
4622	**17954**	Parameter	**'Scaling factor for actual torque (actual current) impermissible'** The value is impermissible as a scaling factor for the actual torque (or actual current).	
			Value range: [0, 1E+30]	Unit:

Error(hex)	Error(dec)	Error type	Description	
4623	**17955**	Parameter	**'Filter time for actual torque is impermissible'** The value is impermissible as a filter time for the actual torque (or the actual current) (P-T1 filter).	
			Value range: [0.0, 60.0]	Unit: s
4624	**17956**	Parameter	**'Filter time for the temporal derivation of the actual torque is impermissible'** The value is impermissible as a filter time for the temporal derivation of the actual torque (or actual current (P-T1 filter).	
			Value range: [0.0, 60.0]	Unit: s
4627	**17959**	Function	**DRIVEOPERATIONMODEBUSY.** The activation of the drive operation mode failed, because another object with OID ... is already using this interface.	
0x4630 ... 0x463F: Error codes are reserved for external drive errors (e.g. stepper motor terminal or 'MC_PowerStepper' function block)				
4630	**17968**	Monitoring	**'Overtemperature'** Overtemperature was detected or reported in the drive or terminal.	
4631	**17969**	Monitoring	**'Undervoltage'** Undervoltage was detected or reported in the drive or terminal.	
4632	**17970**	Monitoring	**'Wire break in phase A'** A wire break in phase A was detected or reported in the drive or terminal.	
4633	**17971**	Monitoring	**'Wire break in phase B'** A wire break in phase B was detected or reported in the drive or terminal.	
4634	**17972**	Monitoring	**'Overcurrent in phase A'** Overcurrent was detected or reported in phase A in the drive or terminal.	
4635	**17973**	Monitoring	**'Overcurrent in phase B'** Overcurrent was detected or reported in phase B in the drive or terminal.	
4636	**17974**	Monitoring	**'Torque overload (stall)'** A torque overload (stall) was detected or reported in the drive or terminal.	
4640	**17984**	Initialization	**'Drive initialization'** Drive has not been initialized. Although the drive has been created, the rest of the initialization has not been performed (1. Initialization of drive I/O, 2. Initialization of drive, 3. Reset drive).	
4641	**17985**	Address	**'Axis address'** Drive does not know its axis, or the axis address has not been initialized.	
4642	**17986**	Address	**'Address IO input structure'** Drive has no valid IO input address in the process image.	
4643	**17987**	Address	**'Address IO output structure'** Drive has no valid IO output address in the process image.	
4650	**18000**	Monitoring	**'Drive hardware not ready to operate'** The drive hardware is not ready for operation. The following are possible causes: - the drive is in the error state (hardware error) - the drive is in the start-up phase (e.g. after an axis reset that was preceded by a hardware error) - the drive is missing the controller enable (ENABLE) Note: The time required for "booting" a drive after a hardware fault can amount to several seconds.	
4651	**18001**	Monitoring	**Error in the cyclic communication of the drive (Life Counter).** Reasons for this could be an interrupted fieldbus or a drive that is in the error state.	
4652	**18002**	Monitoring	**'Changing the table ID when active controller enable is impermissible'.** Changing (deselecting, selecting) the characteristic curve table ID is not permissible when the controller enable for the axis is active.	

Error(hex)	Error(dec)	Error type	Description
4655	**18005**	Monitoring	**‘Invalid IO data for more than ‘n’ continuous NC cycles’** The axis (encoder or drive) has detected invalid IO data (e.g. n=3) for more than ‘n’ continuous NC cycles (NC SAF task). EtherCAT fieldbus: ‘working counter error ('WCState')’As a result it is possible that the encoder referencing flag will be reset to FALSE (i.e. the encoder is given the status ‘unreferenced’). Lightbus fieldbus: ‘CDL state error ('CdlState')’ As a result it is possible that the encoder calibration flag will set to FALSE (that means uncalibrated).

7.7.8 Table Errors

Error(Hex)	Error(Dec)	Error type	Description	
4A00	**18944**	parameter	**"Table ID not allowed"** The value for the table ID is not allowed, e.g. because it has already been assigned, is less than or equal to zero, or is greater than 255.	
			Value range: [1 ... 255]	Unit: 1
4A01	**18945**	parameter	**"Table type not allowed"** The value for the table type is unacceptable because it is not defined.	
			Value range: [1]	Unit: 1
4A02	**18946**	parameter	**"Number of lines in the table not allowed"** The value of the number of lines in the table is not allowed, because, for example, it is smaller than two at linear interpolation and smaller than four at spline interpolation.	
			Value range: [2, 0xFFFF]	Unit: 1
4A03	**18947**	parameter	**"Number of columns in the table is not allowed"** The value of the number of columns in the table is not allowed, because, for example, it is less than or equal to zero (depends upon the type of table or slave).	
			Value range: [1, 0xFFFF]	Unit: 1
4A04	**18948**	parameter	**"Step size (position delta) not allowed"** The value for the step size between two lines (position delta) is not allowed, because, for example, it is less than or equal to zero.	
			Value range: [0.001, 1.0E+6]	Unit: e.g. mm
4A05	**18949**	parameter	**"Period not allowed"** The value for the period is not allowed, because, for example, it is less than or equal to zero.	
			Value range: [0.001, 1.0E+9]	Unit: e.g. mm
4A06	**18950**	parameter	**"Table is not monotonic"** The value for the step size is not allowed, because, for example, it is less than or equal to zero.	
4A07	**18951**	initialization	**„Table sub type is not allowed“** The value for the table sub type is not allowed or otherwise the table class (slave type) do not match up to the table main type. Table sub type: (1) equidistant linear position table, (2) equidistant cyclic position table, (3) none equidistant linear position table, (4) none equidistant cyclic position table	
			Value range: [1, 4]	Unit: 1
4A08	**18952**	initialization	**„Table interpolation type is not allowed“** The value for the table interpolation type is allowed. Table interpolation type: (0) linear-interpolation, (1) 4-point-interpolation, (2) spline-interpolation	
			Value range: [0, 2]	Unit: 1
4A09	**18953**	initialization	**"Incorrect table main type"** The table main type is unknown or otherwise the table class (slave type) do not match up to the table main type. Table main type: (1) camming table, (2) characteristic table, (3) 'motion function' table (MF)	
4A10	**18960**	initialization	**"Table initialization"** Table has not been initialized. Although the table has been created, the rest of the initialization has not been performed. For instance, the number of lines or columns may be less than or equal to zero.	
4A11	**18961**	initialization	**"Not enough memory"** Table could not be created, since there is not enough memory.	
4A12	**18962**	function	**"Function not executed, function not available"** The function has not been implemented, or cannot be executed, for the present type of table.	

Error(Hex)	Error(Dec)	Error type	Description
4A13	**18963**	function	**"Line index not allowed"** The start line index or the stop line index to be used for read or write access to the table is not allowed. For instance, the line index may be greater than the total number of lines in the table.
4A14	**18964**	function	**"Column index not allowed"** The start column index or the stop column index to be used for read or write access to the table in not allowed. For instance, the column index may be greater than the total number of columns in the table.
4A15	**18965**	function	**"Number of lines not allowed"** The number of lines to be read from or written to the table is not allowed. The number of lines must be an integer multiple of the number of elements in a line (n * number of columns).
4A16	**18966**	function	**"Number of columns not allowed"** The number of columns to be read from or written to the table is not allowed. The number of columns must be an integer multiple of the number of elements in a column (n * number of lines).
4A17	**18967**	function	**"Error in scaling or in range entry"** The entries in the table header are inconsistent, e.g. the validity range is empty. If the error is generated during the run time it is a run time error and stops the master/slave group.
4A18	**18968**	function	**"Multi table slave out of range"** The slave master position is outside the table values for the master. The error is a run-time error, and stops the master/slave group.
4A19	**18969**	function	**"Solo table underflow"** The slave master position is outside the table values for the master. The master value of the equidistant table, to be processed linearly, lies under the first table value. The error is a run-time error, and stops the master/slave group.
4A1A	**18970**	function	**"Solo table overflow"** The slave master position is outside the table values for the master. The master value of the equidistant table, to be processed linearly, lies above the first table value. The error is a run-time error, and stops the master/slave group.
4A1B	**18971**	parameter	**"Incorrect execution mode"** The cyclic execution mode can only be "true" or "false".
4A1C	**18972**	parameter	**"Impermissible parameter"** The Fifo parameter is not allowed.
4A1D	**18973**	parameter	**"Fifo is empty"** The Fifo of the external generator is empty. This can signify end of track or a run time error.
4A1E	**18974**	parameter	**"Fifo is full"** The Fifo of the external generator is full. It is the user‘s task to continue to attempt to fill the Fifo with the rejected values.
4A1F	**18975**	parameter	**„Point-Index of Motion Function invalid“** The point index of a Motion Function Point of a Function Table is invalid. First the point index has to be larger than zero and second it has to be numerical continuously for one column in the Motion Function Table (e.g. 1,2,3,... or 10,11,12,...). Remark: The point index is not online-changeable but must be constant.
4A20	**18976**	initialization	**„No diagonalization of matrix“** The spline can not be calculated. The master positions are not correct.
4A21	**18977**	initialization	**„Number of spline points to less“** The number of points of a cubic spline has to be greater than two.

Error(Hex)	Error(Dec)	Error type	Description
4A22	**18978**	initialization	**„Fifo must not be overwritten“** Fifo must not be overwritten since then the active line would be overwritten. It is the task of the user to secure that the active line is not modified.
4A23	**18979**	function	**„Insufficient number of Motion Function points“** The number of valid Motion Function points is less than two. Either the entire number of points is to low or the point type of many points is set to *Ignore Point*.

7.7.9 NC-PLC Errors

Error(Hex)	Error(dec)	Error type	Description
4B00	**19200**	Parameter	**"Axis was stopped"** The axis was stopped during travel to the target position. The axis may have been stopped with a PLC command via ADS, a call via AXFNC, or by the System Manager.
4B01	**19201**	Parameter	**"Axis cannot be started"** The axis cannot be started because: • the axis is in error status, • the axis is executing another command, • the axis is in protected mode, • the axis is not ready for operation.
4B02	**19202**	Parameter	**"Control mode not permitted"** No target position control, and no position range control.
4B03	**19203**	Parameter	**"Axis is not moving"** The position and velocity can only be restarted while the axis is physically in motion.
4B04	**19204**	Parameter	**"Wrong mode for RestartPosAndVelo"** Wrong mode.
4B05	**19205**	Parameter	**"Command not permitted"** • Continuous motion in an unspecified direction • Read/Write parameters: type mismatch
4B06	**19206**	Parameter	**"Parameter incorrect"** • Incorrect override: > 100% or < 0% • Incorrect gear ratio: RatioDenominator = 0
4B07	**19207**	Parameter	**"Timeout axis function block"** After positioning, all "MC_Move..." blocks check whether positioning was completed successfully. In the simplest case, the "AxisHasJob" flag of the NC axis is checked, which initially signifies that positioning was logically completed. Depending on the parameterization of the NC axis, further checks (quality criteria) are used: • "Position range monitoring" If position range monitoring is active, the system waits for feedback from the NC. After positioning, the axis must be within the specified positioning range window. If necessary, the position controller ensures that the axis is moved to the target position. If the position controller is switched off (Kv=0) or weak, the target may not be reached. • "Target position monitoring" If target position monitoring is active, the system waits for feedback from the NC. After positioning, the axis must be within the specified target position window for at least the specified time. If necessary, the position controller ensures that the axis is moved to the target position. If the position controller is switched off (Kv=0) or weak, the target may not be reached. Floating position control may lead to the axis oscillating around the window but not remaining inside the window. If the axis is logically at the target position (logical standstill) but the parameterized position window has not been reached, monitoring of the above-mentioned NC feedback is aborted with error 19207 (0x4B07) after a constant timeout of 6 seconds.
4B08	**19208**	Parameter	**"Axis is in protected mode"** The axis is in protected mode (e.g. coupled) and cannot be moved.

Error(Hex)	Error(dec)	Error type	Description
4B09	**19209**	Parameter	**"Axis is not ready"** The axis is not ready and cannot be moved.
4B0A	**19210**	Parameter	**"Error during referencing"** Referencing (homing) of the axis could not be started or was not successful.
4B0B	**19211**	Parameter	**"Incorrect definition of the trigger input"** The definition of the trigger signal for block MC_TouchProbe is incorrect. The defined encoder-ID, the trigger signal or the trigger edge are invalid.
4B0C	**19212**	Function	**"Position latch was disabled"** The function block MC_TouchProbe has detected that a measuring probe cycle it had started was disabled. The reason may be an axis reset, for example.
4B0D	**19213**	Function	**'NC status feedback timeout'** A function was successfully sent from the PLC to the NC. An expected feedback in the axis status word has not arrived.
4B0E	**19214**	Function	**"Additional product not installed"** The function is available as an additional product but is not installed on the system.
4B0F	**19215**	Function	**"No NC Cycle Counter Update"** – The NcToPlc Interface or the NC Cycle Counter in the NcToPlc Interface was not updated.
Error numbers 0x4B10 to 0x4B2F are used in the *TwinCAT NCI* **context:**			
4B10	**19216**	Function	**"M-function query missing"** This error occurs if the M-function was confirmed, but the request bit was not set.
4B11	**19217**	Parameter	**"Zero offset index is outside the range"** The index of the zero offset is invalid.
4B12	**19218**	Parameter	**"R-parameter index or size is invalid"** This error occurs if the R-parameters are written or read but the index or size are outside the range.
4B13	**19219**	Parameter	**"Index for tool description is invalid"**
4B14	**19220**	Function	**"Version of the cyclic channel interface does not match the requested function or the function block"** This error occurs if an older TwinCAT version is used to call new functions of a later TcNci.lib version.
4B15	**19221**	Function	**"Channel is not ready for the requested function"** The requested function cannot be executed, because the channel is in the wrong state. This error occurs during reverse travel, for example, if the axis was not stopped with ItpEStop first.
4B16	**19222**	Function	**"Requested function is not activated"** The requested function requires explicit activation.
4B17	**19223**	Function	**"Axis is already in another group"** The axis has already been added to another group.
4B18	**19224**	Function	**"Block search could not be executed successfully"** The block search has failed. Possible causes: • Invalid block number
4B19	**19225**	Parameter	**"Invalid block search parameter"** This error occurs if the FB ItpBlocksearch is called with invalid parameters (e.g. E_ItpDryRunMode, E_ItpBlockSearchMode)
4B20	**19232**	Function	**"Cannot add all axes"** This error occurs if an auxiliary axis is to be added to an interpolation group, but the function fails. It is likely that a preceding instruction of an auxiliary axis was skipped.
Error numbers 0x4B30 to 0x4B3F are used in the *TcMcCam library* (MC_NC_TableErrorCodes):			

Error(Hex)	Error(dec)	Error type	Description
4B30	**19248**	Parameter	**"Pointer is invalid"** A pointer to a data structure is invalid, e.g. Null • Data structure MC_CAM_REF was not initialized
4B31	**19249**	Parameter	**"Memory size invalid"** The specification of the memory size (SIZE) for a data structure is invalid. • The value of the size parameter is 0 or less than the size of one element of the addressed data structure. • The value of the size parameter is less than the requested amount of data. • The value of the size parameter does not match other parameters as number of points, number of rows or number of columns.
4B32	**19250**	Parameter	**"Cam table ID is invalid"** The ID of a cam table is not between 1 and 255.
4B33	**19251**	Parameter	**"Point ID is invalid"** The ID of a point (sampling point) of a motion function is less than 1.
4B34	**19252**	Parameter	**"Number of points is invalid"** The number of points (sampling points) of a cam plate to be read or written is less than 1.
4B35	**19253**	Parameter	**"MC table type is invalid"** The type of a cam plate does not match the definition *MC_TableType*.
4B36	**19254**	Parameter	**"Number of rows invalid"** The number of rows (sampling points) of a cam table is less than 1.
4B37	**19255**	Parameter	**"Number of columns invalid"** The number of columns of a cam table is invalid. • The number of columns of a motion function is not equal 1 • The number of columns of a standard cam table is not equal 2 • The number of columns does not match another parameter (ValueSelectMask)
4B38	**19256**	Parameter	**"Step size invalid"**. The increment for the interpolation is invalid, e.g. less than or equal to zero.
Error numbers 0x4B0F, 0x4B40 to 0x4B4F are used in the *TcNc-Lib*:			
4B40	**19264**	Monitoring	**"Terminal type not supported"** The terminal used is not supported by this function block.
4B41	**19265**	Monitoring	**"Register read/write error"** This error implies a validity error.
4B42	**19266**	Monitoring	**"Axis is enabled"** The axis is enabled but should not be enabled for this process.
4B43	**19267**	Parameter	**"Incorrect size of the compensation table"** The specified table size (in bytes) does not match the actual size
4B44	**19268**	Parameter	The minimum/maximum position in the compensation table does not match the position in the table description (ST_CompensationDesc)
4B45	**19269**	Parameter	**"Not implemented"** The requested function is not implemented in this combination
Error numbers 0x4B50 to 0x4B5F are used in the *TcRemoteSyn-Lib*:			
Error numbers 0x4B60 to 0x4B6F are used in the *TcMc2-Lib* **in the** *buffered commands* **context:**			

Error(Hex)	Error(dec)	Error type	Description
4B60	**19296**	Monitoring	**"Motion command did not become active"** A motion command has been started and has been buffered and confirmed by the NC. Nevertheless, the motion command did not become active (possibly due to a terminating condition or an internal NC error).
4B61	**19297**	Monitoring	**"Motion command could not be monitored by the PLC"** A motion command has been started and has been buffered and confirmed by the NC. The PLC has not been able to monitor the execution of this command and the execution status is unclear since the NC is already executing a more recent command. The execution state is unclear. This error may come up with very short buffered motion commands which are executed during one PLC cycle.
4B62	**19298**	Monitoring	**"Buffered command was terminated with an error"** A buffered command was terminated with an error. The error number is not available, because a new command is already being executed.
4B63	**19299**	Monitoring	**"Buffered command was completed without feedback"** A buffered command was completed but there was no feedback to indicate success or failure.
4B64	**19300**	Monitoring	**" 'BufferMode' is not supported by the command"** The 'BufferMode' is not supported by this command.
4B65	**19301**	Monitoring	**"Command number is zero"** The command number for queued commands managed by the system unexpectedly has the value 0.
4B66	**19302**	Monitoring	**"Function block was not called cyclically"** The function block was not called cyclically. The command execution could not be monitored by the PLC, because the NC was already executing a subsequent command. The execution state is unclear.
Error numbers 0x4B70 to 0x4B8F are used in the *TcPlcInterpolation-Lib*:			
4B71	**19313**	Parameter	**"Invalid NCI entry type".** The FB FB_NciFeedTablePreparation was called with an unknown nEntryType.
4B72	**19314**	Function	**"NCI feed table full"** The table is full, and the entry is therefore not accepted. Remedy: Transfer the context of the table with FB_NciFeedTable to the NC-Kernel. If bFeedingDone = TRUE, the table can be reset in FB_NciFeedTablePreparation with bResetTable and then filled with new entries.
4B73	**19315**	Function	internal error
4B74	**19316**	Parameter	**"ST_NciTangentialFollowingDesc: tangential axis is not an auxiliary axis"** The entry for tangential following contains a tangential axis that is not an auxiliary axis.
4B75	**19317**	Parameter	**ST_NciTangentialFollowingDesc:** nPathAxis1 or nPathAxis2 is not a path axis. It is therefore not possible to determine the plane.
4B76	**19318**	Parameter	**ST_NciTangentialFollwoingDesc:** nPathAxis1 and nPathAxis2 are the same. It is therefore not possible to determine the plane.
4B77	**19319**	Parameter	**ST_NciGeoCirclePlane**: Circle incorrectly parameterized
4B78	**19320**	Function	Internal error during calculation of tangential following

Error(Hex)	Error(dec)	Error type	Description
4B79	**19321**	Monitoring	**Tangential following:** Monitoring of the deviation angle was activated during activation of tangential following (E_TfErrorOnCritical1), and an excessively large deviation angle was detected in the current segment.
4B7A	**19322**	Function	not implemented
4B7B	**19323**	Parameter	**Tangential following**: the radius of the current arc is too small
4B7C	**19324**	Parameter	**FB_NciFeedTablePreparation**: pEntry is NULL
4B7D	**19325**	Parameter	**FB_NciFeedTablePreparation**: the specified nEntryType does not match the structure type
4B7E	**19326**	Parameter	**ST_NciMFuncFast and ST_NciMFuncHsk**: the requested M-function is not between 0 and 159
4B7F	**19327**	Parameter	**ST_NciDynOvr**: the requested value for the dynamic override is not between 0.01 and 1
4B80	**19328**	Parameter	**ST_NciVertexSmoothing**: invalid parameter. This error is generated if a negative smoothing radius or an unknown smoothing type is encountered.
4B81	**19329**	Parameter	**FB_NciFeedTablePrepartion:** The requested velocity is not in the valid range
4B82	**19330**	Parameter	**ST_Nci***: invalid parameter
Error numbers 0x4BA0 - 0x4BAF are used in the *TcNcKinematicTransformation-Lib***:**			
4BA0	**19360**	Function	**KinGroup error:** the kinematic group is in an error state. This error may occur if the kinematic group is in an error state or an unexpected state when it is called (e.g. simultaneous call via several FB instances).
4BA1	**19361**	Function	**KinGroup timeout:** timeout during call of a kinematic block
Error numbers 0x4B90 - 0x4B9F are used in the *Tc3_MC2_AdvancedHoming-Lib* **(PLCopen Part 5: Homing Procedures):**			
4B90	**19344**	Parameter	Determined **drive type** is not supported
4B91	**19345**	Parameter	**Direction** is impermissible
4B92	**19346**		**SwitchMode** is impermissible
4B93	**19347**		**Mode** for the parameter handling is impermissible
4B94	**19348**		Parameterization of the torque limits is inconsistent
4B95	**19349**		Parameterization of the position lag limit is impermissible (<=0).
4B96	**19350**		Parameterization of the distance limit is impermissible (<0)
4B97	**19351**		An attempt was made to back up parameters again, although they have already been backed up.
4B98	**19352**		An attempt was made to restore parameters, although none have been backed up.
Error numbers 0x4BB0 to 0x4BBF are used in the *Tc2_MC2_Drive-Lib***:**			
4BB0	**19376**	Function	The current axis position or the axis position resulting from the new position offset exceeds the valid range of values.
4BB1	**19377**	Function	The new position offset exceeds the valid range of values [AX5000: 2^31]
4BB2	**19378**	Function	The current axis position or the axis position resulting from the new position offset falls below the valid range of values.
4BB3	**19379**	Function	The new position offset falls below the valid range of values [AX5000: -2^31]

Error(Hex)	Error(dec)	Error type	Description
4BB4	**19380**	Function	The activated feedback and/or storage location (AX5000: P-0-0275) differ from the parameterization on the function block.
4BB5	**19381**	Function	Reinitialisation of the Nc actual position has failed. e.g. reference system = "ABSOLUTE (with single overflow)" & software end position monitoring is disabled.

7.7.10 Kinematic transformation

Error(Hex)	Error(Dec)	Error type	Description
4C00	**19456**		**Transformation failed.**
4C01	**19457**		**Ambiguous answer.** The answer of the transformation is not explicit.
4C02	**19458**		**Invalid axis position:** The transformation can not be calculated with the current position data. Possible causes: • The position is outside the working area of the kinematics
4C03	**19459**	Configuration	**Invalid dimension:** The dimension of the paramerterized input parameter does not match the dimension expected by the kinematic object. Possible causes: • Too many position values are supplied for this configuration. Check the number of parameterized axes.
4C04	**19460**		**NCERR_KINTRAFO_REGISTRATION**
4C05	**19461**	Internal	**Newton iteration failed:** The Newton iteration does not converge.
4C06	**19462**	Internal	**Jacobi matrix cannot be inverted**
4C07	**19463**	Configuration	**Invalid cascade:** This kinematic configuration is not permitted.
4C08	**19464**	Programming	**Singularity:** The machine configuration results in singular axis velocities.
4C0B	**19467**	Internal	**No metainfo:** Metainfo pointer is null.
4C20	**19488**	Internal	**Transformation failed:** Call of extended kinematic model failed.
4C30	**19504**	Programming	**Invalid input frame:** Programmed Cartesian position cannot be reached in the ACS configuration.
4C50	**19536**	Internal	**Invalid offset:** Access violation detected in the observer.

7.7.11 Bode Return Codes

The following bode plot specific error codes are used in the bode plot server:

Code Hex	Code Dec	Symbol	Description
0x8100	33024	INTERNAL	Internal error
0x8101	33025	NOTINITIALIZED	Not initialized (e.g. no nc axis)
0x8102	33026	INVALIDPARAM	Invalid parameter
0x8103	33027	INVALIDOFFSET	Invalid index offset
0x8104	33028	INVALIDSIZE	Invalid parameter size
0x8105	33029	INVALIDSTARTPARAM	Invalid start parameter (set point generator)
0x8106	33030	NOTSUPPORTED	Not supported
0x8107	33031	AXISNOTENABLED	Nc axis not enabled
0x8108	33032	AXISINERRORSTATE	Nc axis in error state
0x8109	33033	DRIVEINERRORSTATE	IO drive in error state
0x810A	33034	AXISANDDRIVEINERROR-STATE	Nc axis AND IO drive in error state
0x810B	33035	INVALIDDRIVEOPMODE	Invalid drive operation mode active or requested (no bode plot mode)
0x810C	33036	INVALIDCONTEXT	Invalid context for this command (mandatory task or windows context needed)
0x810D	33037	NOAXISINTERFACE	**Missing TCom axis interface (axis null pointer).** There is no connection to the NC axis. Either no axis (or axis ID) has been parameterized, or the parameterized axis does not exist.
0x810E	33038	INPUTCYCLECOUNTER	**Invalid input cycle counter from IO drive (e.g. frozen).** The cyclic drive data are backed up by an 'InputCycleCounter' during the bode plot recording. This allows firstly the detection of an unexpected communication loss (keyword: LifeCounter) and secondly a check for temporal data consistency to be performed. **Sample 1:** This error can occur if the cycle time of the calling task is larger than the assumed drive cycle time (in this case, however, the error occurs right at the start of the recording). **Sample 2:** This error can occur if the calling task has real-time errors (e.g. the "Exceed Counter" of the task increments or the task has a lower priority, as is often the case, for example, with the PLC). In this case the error can also occur at any time during the recording. **Sample 3:** This error can occur more frequently if the real-time load on the computer is quite high (>50 %). **Note:** Refer also to the corresponding AX5000 drive error code F440.

0x810F	33039	POSITIONMONITORING (=> NC Runtime Error)	**Position monitoring: Axis position is outside of the maximum allowed moving range.** The axis has left the parameterized position range window, whereupon the recording was aborted and the NC axis was placed in the error state 0x810F (with standard NC error handling). The position range window acts symmetrically around the initial position of the axis (see also parameter description *Position Monitoring Window*). Typical error message in the logger: *"BodePlot: 'Position Monitoring' error 0x%x because the actual position %f is above the maximum limit %f of the allowed position range (StartPos=%f, Window=%f)"*
0x8110	33040	DRIVELIMITATIONDETECTE D	**Driver limitations detected (current or velocity limitations) which causes a nonlinear behavior and invalid results of the bode plot.** A bode plot recording requires an approximately linear transmission link. If the speed or current is limited in the drive unit, however, this non-linear behavior is detected and the bode plot recording is aborted. Reasons for these limitations can be: choosing too large an amplitude for the position, speed or torque interface, or an unsuitable choice of amplitude scaling mode (see also parameter description *Amplitude Scaling Mode, Base Amplitude, Signal Amplitude*). Typical error message in the logger: *"BodePlot: Sequence aborted with error 0x%x because the current limit of the drive has been exceeded (%d times) which causes a nonlinear behavior and invalid results of the bode plot"*
0x8111	33041	LIFECOUNTERMONITORING (=> NC Runtime Error)	**Life counter monitoring (heartbeat): Lost of communication to GUI detected after watchdog timeout is elapsed.** The graphical user interface from which the bode plot recording was started is no longer communicating with the bode plot driver in the expected rhythm (keyword: ‘Life Counter’). Therefore the recording is terminated immediately and the NC axes are placed in the error state 0x8111 (with standard NC error handling). Possible reasons for this can be an operating interface crash or a major malfunction of the Windows context. Typical error message in the logger: *"BodePlot: Sequence aborted with GUI Life Counter error 0x%x because the WatchDog timeout of %f s elapsed ('%s')"*
0x8112	33042	NCERR_BODEPLOT_WCSTA TE	**WC state error (IO data working counter)** IO working counter error (WC state), for example due to real-time errors, EtherCAT CRC errors or telegram failures, EtherCAT device not communicating (OP state), etc.
0x8113-0x811F	33043-33055	RESERVED	**Reserved area**

7.7.12 Further Error Codes

Table 3:

Error(Hex)	Error(Dec)	ErrorType	Description
0x8120	**33056**	Environment	**Invalid configuration for Object (e.g. in System Manager)**
0x8121	**33057**	Environment	**Invalid environment for Object (e.g. TcCom-Object's Hierarchy or missing/faulty Objects)**
0x8122	**33058**	Environment	**Incompatible Driver or Object**
0x8135	**33077**	Internal	**Function Block Inputs are inconsitent.**Some Inputs of the Function Block are inconsistent during. Probably Communicator and its IID, which both have to be set or unset.
0x813b	**33083**	Parameter	**Transition Mode is invalid**
0x813c	**33084**	Parameter	**BufferMode is invalid**
0x813d	**33085**	FunctionBlock	**Only one active Instance of Function Block per Group is allowed.**
0x813e	**33086**	State	**Command is not allowed in current group state.**
0x813f	**33087**	FunctionBlock	**Slave cannot synchronize.**The slave cannot reach the SlaveSyncPosition by the time the master has reached the MasterSyncPos.
0x8140	**33088**	Parameter	**Invalid value for one or more of the dynamic parameters (A, D, J).**
0x8141	**33089**	Parameter	**IdentInGroup is invalid.**
0x8142	**33090**	Parameter	**The number of axes in the group is incompatible with the axes convention.**
0x8143	**33091**	Communication	**Function Block or respective Command is not supported by Target.**
0x8145	**33093**	FunctionBlock	**Mapping of Cyclic Interface between Nc and Plc missing (e.g. AXIS_REF, AXES_GROUP_REF, ...).**
0x8146	**33094**	FunctionBlock	**Invalid Velocity Value**The velocity was not set or the entered value is invalid
0x8148	**33096**	FunctionBlock	**Invalid Input Value**
0x8149	**33097**	Parameter	**Unsupported Dynamics for selected Group Kernel**
0x814a	**33098**	Parameter	**The programmed position dimension incompatible with the axes convention.**
0x814b	**33099**	FunctionBlock	**Path buffer is invalid. E.g. because provided buffer has invalid address or is not big enough**
0x814c	**33100**	FunctionBlock	**Path does not contain any element**
0x814d	**33101**	FunctionBlock	**Provided Path buffer is too small to store more Path Elements**
0x814e	**33102**	Parameter	**Dimension or at least one Value of Transition Parameters is invalid**
0x814f	**33103**	FunctionBlock	**Invalid or Incomplete Input Array**
0x8150	**33104**	FunctionBlock	**Path length is zero**
0x8151	**33105**	State	**Command is not allowed in current axis state.**
0x8152	**33106**	State	**TwinCAT System is shutting down and cannot complete request.**
0x8160	**33120**	NC Programming	**Circle Specification in Path is invalid.**The specification of a circle segment in the programmed interpolated path (e.g. via MC_MovePath) has an invalid or ambiguous descripition. Probably its center cannot be determined reliably.
0x8161	**33121**	NC Programming	**Maximum stream lines reached.**The maximum number of stream lines is limited. Please refer to function block documentation for details.
0x8163	**33123**	FunctionBlock	**Invalid First Segment.**The corresponding element can only be analyzed with a well-defined start point.

Error(Hex)	Error(Dec)	ErrorType	Description
0x8164	**33124**	FunctionBlock	**Invalid auxiliary point.**The auxiliary point is not well-defined.
0x8166	**33126**	FunctionBlock	**Invalid parameter for GapControlMode**Invalid parameter for GapControlMode, most likely in combination with the group parameter GapControlDirection
0x8167	**33127**	External	**Group got unsupported Axis Event (e.g. State Change)**Group got unsupported Axis Event (e.g. State Change e.g. triggered by a Single Axis Reset)
0x8f76 - 0x8ff9	**36726 - 36857**	Internal	**Unexpected Internal Error**
0x8ffb - 0x8ffe	**36859 - 36862**	Internal	**Unexpected Internal Error**

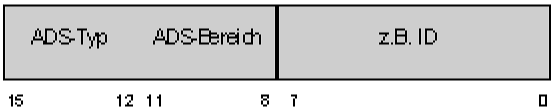
7.8 Specification "Index group" for NC (ID [0x01...0xFF])

Note: This documentation contains all TC3 specific modifications and new features.

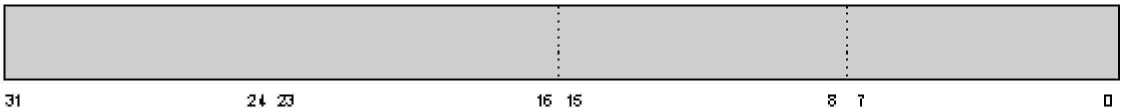
Index-Group (Hex)	Description	Remarks
0x1000	Ring-0-Manager: Parameter [▶ 380]	Optional !
0x1100	Ring-0-Manager: State [▶ 381]	Optional !
0x1200	Ring-0-Manager: Functions [▶ 381]	Optional !
0x1300	Ring-0-Manager: Cyclic process data	Not implemented !
0x2000 + ID	Channel with corresponding ID: parameters [▶ 382]	
0x2100 + ID	Channel with corresponding ID: state [▶ 385]	
0x2200 + ID	Channel with corresponding ID: functions [▶ 387]	
0x2300 + ID	Channel with corresponding ID: cyclic process data [▶ 390]	
0x3000 + ID	Group with corresponding ID: parameters [▶ 391]	Optional !
0x3100 + ID	Group with corresponding ID: state [▶ 396]	Optional !
0x3200 + ID	Group with corresponding ID: functions [▶ 401]	Optional !
0x3300 + ID	Group with corresponding ID: cyclic process data	Not implemented !
0x4000 + ID	Axis with corresponding ID: parameters [▶ 407]	
0x4100 + ID	Axis with corresponding ID: state [▶ 419]	
0x4200 + ID	Axis with corresponding ID: functions [▶ 428]	
0x4300 + ID	Axis with corresponding ID: cyclic process data [▶ 447]	
0x5000 + ID	Encoder with corresponding ID: parameters [▶ 452]	Optional !
0x5100 + ID	Encoder with corresponding ID: state [▶ 456]	Optional !
0x5200 + ID	Encoder with corresponding ID: functions [▶ 460]	Optional !
0x5300 + ID	Encoder with corresponding ID: cyclic process data [▶ 463]	Optional !
0x6000 + ID	Controller with corresponding ID: Parameter [▶ 467]	Optional !
0x6100 + ID	Controller with corresponding ID: State [▶ 471]	Optional !
0x6200 + ID	Controller with corresponding ID: Functions [▶ 474]	Optional !
0x6300 + ID	Controller with corresponding ID: cyclic process data	Not implemented !
0x7000 + ID	Drive with corr. ID: parameters [▶ 475]	Optional !
0x7100 + ID	Drive with corr. ID: state [▶ 478]	Optional !
0x7200 + ID	Drive with corr. ID: functions [▶ 478]	Optional !
0x7300 + ID	Drive with corr. ID: cyclic process data [▶ 479]	Optional !
0x**0**A0**00** + **ID**	Tables (n x m) with corresponding ID: parameters [▶ 482] 0x0A000+ID for table ID [1..255] 0x1A000+ID for table ID [256..4095] . . . 0xFA000+ID for table ID [3840..4095]	Maximum number of tables extended to 4095 (from TC3.1 B4021)
0x**0**A1**00** + **ID**	Tables (n x m) with corresponding ID: state [▶ 486] 0x0000A100+IDLowByte for table ID [1..255] 0x0001A100+IdLowByte for table ID [256..4095] … 0x000FA100+IdLowByte for table ID [3840..4095] 0x000**n**A1**00**+**IdLowByte** for table ID [1..4095] (TabID = n * 256 + IdLowByte)	

Index-Group (Hex)	Description		Remarks
0x*0*A*200* + *ID*	Tables (n x m) with corresponding ID: functions [▶ 487] 0x000*0*A1*00*+*IDLowByte* for table ID [1..255] 0x000*1*A1*00*+*IdLowByte* for table ID [256..4095] … 0x000*F*A1*00*+*IdLowByte* for table ID [3840..4095] 0x000*n*A1*00*+*IdLowByte* for table ID [1..4095] (TabID = n * 256 + IdLowByte)		
0x0A300 + ID	Tables (n x m) with corresponding ID: cyclic process data 0x0000A100+IDLowByte for table ID [1..255] 0x0001A100+IdLowByte for table ID [256..4095] … 0x000FA100+IdLowByte for table ID [3840..4095] 0x000nA100+IdLowByte for table ID [1..4095] (TabID = n * 256 + IdLowByte)		Not implemented !
0xF000 ... 0xFFFF	reserved area (TwinCAT system area)		
IndexGroup:	IndexOffset:		
0xF081	0x00000000 ... 0xFFFFFFFF (n elements)	ADSIGRP_SUMUP_WRITE The *Read-Write-command* contains a list in the Write-data of multiple separate *ADS-Write-commands* (like a group request). Structure of the Write-Data: *[IdxGrp(1), IdxOff(1), WriteLen(1), ..., IdxGrp(n), IdxOff(n), WriteLen(n), WriteData(1), ..., WriteData(n)]* Structure of the Read-Data: *[Error(1), ..., Error(n)]*	
0xF082	0x00000000 ... 0xFFFFFFFF (n elements)	ADSIGRP_SUMUP_READWRITE The *Read-Write-command* contains a list in the Write-data of multiple separate *ADS-Read-Write-commands* (like a group request). Structure of the Write-Data: *[IdxGrp(1), IdxOff(1),ReadLen(1), WriteLen(1), ..., IdxGrp(n), IdxGrp(n), ReadLen(n), WriteLen(n), WriteData(1), ..., WriteData(n)]* Structure of the Read-Data: *[Error(1), ReadLen(1), ..., Error(n), ReadLen(n), ReadData(1), ..., ReadData(n)]*	
0xF084	0x00000000 ... 0xFFFFFFFF (n elements)	ADSIGRP_SUMUP_READ (READEX2) The *Read-Write-command* contains a list in the Write-data of multiple separate *ADS-Read-commands* (like a group request). Structure of the Write-Data: *[IdxGrp(1), IdxOff(1), ReadLen(1), ..., IdxGrp(n), IdxGrp(n), ReadLen(n)]* Structure of the Read-Data: *[Error(1), ReadLen(1), ..., Error(n), ReadLen(n), ReadData(1), ..., ReadData(n)]*	

Index-Group:



Index-Offset:



7.8.1 Specification Ring-0-Manager

7.8.1.1 "Index offset" specification for Ring-0 parameter (Index group 0x1000)

Index offset (Hex)	Access	Ring 0 Manager	Data type	Phys. unit	Definition range	Description	Remarks
0x00000010	Read	every	UINT32	100 ns		cycle time SAF task	
0x00000012	Read	every	UINT32	100 ns		cycle time SVB task	
0x00000014	Read	every	INT32	ns		Global Time Compensation Shift (for SAF Task)	
0x00000020	Read/Write	every	UINT16	1	0/1	cyclic data consistence check and correction of the NC setpoint values	

7.8.1.2 "Index offset" specification for Ring-0 state (Index group 0x1100)

Index offset (Hex)	Access	Ring 0 Manager	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	UINT32	1	0, 1...255	quantitiy of Channel	
0x00000002	Read	every	UINT32	1	0, 1...255	quantitiy of group	
0x00000003	Read	every	UINT32	1	0, 1...255	quantitiy of Axis	
0x00000004	Read	every	UINT32	1	0, 1...255	quantitiy of Encoder	
0x00000005	Read	every	UINT32	1	0, 1...255	quantitiy of controller	
0x00000006	Read	every	UINT32	1	0, 1...255	quantitiy of Drives	
0x0000000A	Read	every	UINT32	1	0, 1...255	quantitiy of table (n x m)	
0x00000010	Read	every	UINT32	1		cycle time error counter SAF task (not scopeable)	reserved!
0x00000014	Read	every	UINT32	1		IO-cycle time error counter SAF task (not scopeable)	reserved!
0x00000020	Read	every	UINT32	s		computing time SAF task (not scopeable)	reserved!
0x00000031	Read	every	UINT32[n]	1	0, 1...255	supplies the Channel - IDs for all Channel in the system	
0x00000032	Read	every	UINT32[n]	1	0, 1...255	supplies the group-IDs for all group in the system	
0x00000033	Read	every	UINT32[n]	1	0, 1...255	supplies the axis IDs for all Axis in the system	
0x00000034	Read	every	UINT32[n]	1	0, 1...255	supplies the Encoder-IDs for all Encoder in the system	
0x00000035	Read	every	UINT32[n]	1	0, 1...255	supplies the controller - IDs for all controller in the system	
0x00000036	Read	every	UINT32[n]	1	0, 1...255	supplies the Drive-IDs for all Drives in the system	
0x0000003A	Read	every	UINT32[n]	1	0, 1...255	supplies the table -IDs for all table in the system	
0x000001nn	Read	every	UINT32	1	0, 1...255	supplies for the Encoder-ID the appropriate axis IDnn = Encoder-ID	reserved!
0x000002nn	Read	every	UINT32	1	0, 1...255	supplies for the Controller-ID the appropriate axis IDnn = Controller-ID	reserved!
0x000003nn	Read	every	UINT32	1	0, 1...255	supplies for the Drive-ID the appropriate axis IDnn = Drive-ID	reserved!

7.8.1.3 "Index offset" specification for Ring-0 functions (Index group 0x1200)

Index offset (Hex)	Access	Ring 0 Manager	Data type	Phys. unit	Definition range	Description	Remarks
0x00000020	Write	every	VOID	1		Clear cycle time error counter SAF & SVB	reserved!

7.8.2 Specification Channels

7.8.2.1 "Index offset" specification for Channel parameter (Index group 0x2000 + ID)

Index-Offset (Hex)	Access	Channel type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	UINT32	1		Channel ID	
0x00000002	Read	every	UINT8[30+1]	1		Channel name	
0x00000003	Read	every	UINT32	1	ENUM	Channel type [▸ 487]	
0x00000004	Read	every	UINT32	1	ENUM	Interpreter type [▸ 487]	
0x00000005	Read	every	UINT32	1		program load buffer size in bytes	
0x00000006	Read	every	UINT32	1		program no. according to job list	
0x00000007	Read / Write	every	UINT32	1	ENUM	set load log mode [▸ 488]	
0x00000008	Read / Write	every	UINT32	1	ENUM	set trace mode [▸ 488]	
0x00000009	Read / Write	every	UINT32	1		RESERVED	
0x0000000A	Read / Write	every	UINT32	1	0/1	Records all feeder entries in a log file named "TcNci.log"	
0x0000000B	Read / Write	every	UINT32	1	0/1	channel specific level for NC logger messages 0: errors only 1: all NC messages	
0x00000010	ReadWrite	every	**Write**				
			{				
			UINT32	1	0..159	Start index of M-function	
			UINT32	1	1..160	number of M-functions to be read	
			}				
			Read [n]				
			{				
			UINT8	1	0..159	Rule bit mask of the M-function	
			INT32[10]	1	-1..159	Number of M-functions to be cleared	
			}				
0x00000011	Write	Interpolation				Write M-function description	Only used internally!
0x00000012	Read / Write	Interpolation	LREAL64	1		factor for G70	
0x00000013	Read / Write	Interpolation	LREAL64	1		Factor for G71	
0x00000014	Write	Interpolation	{			axes user symbols	not yet released
			char[32]			user symbol (null-terminated)	
			char[10]			system symbol (null-terminated)	
			}				
0x00000015	Read / Write	Interpolation	UINT16 resp. UINT32	1	0/1 default: FALSE	Activation of default G-code	NEW from TC3.1 B4014
0x00000021	Read	every	UINT32	1		group ID (only explicit for 3D and FIFO channel)	
0x00000031	Read / Write	Interpolation	UINT16	1		Standard Output Port of the Interpreter	reserved function, no standard!

Index-Offset (Hex)	Access	Channel type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000032	Read / Write	Interpolation	UINT16	1	0/1	cartesian tool offset entry	reserved function, no standard!
0x00000040	Read / Write	Interpolation	{			**Target address of interpreter hooks**	reserved function, no standard!
			char[6]			Ams Net ID	
			UINT16			Port	
			UINT32			Index Group	
			UINT32			Index Offset	
			}				
0x00000050	Read / Write	Interpolation	UINT32	1	ENUM	Reaction if at the radius compensation a bottle neck is recognized 0: Error and abort 1: Note & trouble shooting 2: Only note, without outline modulation	
0x00000051	Read / Write	Interpolation	UINT32	1	1..24	Look ahead for bottleneck detection	
0x00000052	Read / Write	Interpolation	UINT32	1	0/1	Chamfer on/off	reserved function, no standard!
0x00000053	Read / Write	Interpolation	UINT32	1		Activation for reading the currently effective interpolation rules, zero shifts and rotation 0: off 1: on	
0x00000054	Read / Write	Interpolation	UINT32	1	0/1	Retrace on/off	reserved function, no standard!
0x00000055	Read / Write	Interpolation	UINT32[4]	1		Configuration of the cyclic channel interface for UINT32; up to 4 index offsets can be configured.	
0x00000056	Read / Write	Interpolation	UINT32[4]	1		Configuration of the cyclic channel interface for LREAL; up to 4 index offsets can be configured.	
0x00010K0L	Read / Write	every	REAL64	e.g. mm	±MAX REAL64	value for zero shift (NPV)	
					[1..3]	Axis index K=1 → X K=2 → Y K=3 → Z	
					[1..0xA]	L=1 → G54F L=2 → G54G L=3 → G55F ...	
0x0002ww00	Read / Write	every	UINT16			Tool number: values for tool compensation	
0x0003ww00	Read / Write	every	UINT16		[1...50]	Tool type: ww = tool 1...50	
0x0004wwnn	Read / Write	every	REAL64		[1...14]	Parameter: nn = Index 1...14	
0x000500gg	Read / Write	every	REAL64	e.g. mm	≥ 0 (value) [1...9] (g)	Radius of the tolerance sphere gg = channel group (default: 1)	

7.8.2.2 "Index offset" specification for Channel state (Index group 0x2100 + ID)

Index-Offset (Hex)	Access	Channel type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	INT32	1	ENUM	error code Channel [▶ 313]	
0x00000002	Read	every	UINT32	1		number of groups in the Channel	
0x00000003	Read	every	UINT32	1	ENUM	Interpreter status [▶ 488]	Cannot be traced by oscil-loscope!
0x00000004	Read	every	UINT32	1	ENUM	interpreter/channel operation mode [▶ 488]	
0x00000005	Read	every	UINT32	1		currently loaded pro-gram	
0x00000007	Read	every	UINT8[...]	1		program name of cur-rently loaded program (100 characters, null-terminated)	max. 100 char-acters, null-ter-minated
0x00000008	Read	Inter-preter	UINT32	1	[0,1]	Interpreter simulation mode 0: off (default) 1: on	Cannot be traced by oscil-loscope!
0x00000010	Read	Inter-preter	UINT32	1		text index If the interpreter is in the aborted state, the current text index can be read out here	Cannot be traced by oscil-loscope!
0x00000011	ReadWrite	Inter-preter	**Write**				Cannot be traced by oscil-loscope!
			UINT32	1		text index	
			Read				
			UINT8[..]	1		Line of the NC part program from the text index	
0x00000012	Read	Inter-preter	{				
			UINT32	1		Current display for 1: SAF 2: Interpreter 3: Error offset	
			UINT32	1		Fileoffset	
			UINT8[260]	1		path + program name	
			}				
0x00000013	Read	Inter-preter	UINT32[18]			Display for currently effective G-code	
0x00000014	Read	Inter-preter	{			Determines the cur-rently effective zero shift	
			UINT32	1		block counter	
			UINT32			dummy	
			LREAL[3]	1		zero shift G54..G57	
			LREAL[3]	1		zero shift G58	
			LREAL[3]	1		zero shift G59	
			}				
0x00000015	Read	Inter-preter	{			Determines the cur-rently effective rotation	
			UINT32	1		block counter	
			UINT32	1		dummy	
			LREAL[3]	1		rotation of X, Y & Z in degrees	
			}				
0x00000016	Read	Inter-preter	UINT32	1	[0,1]	Feeder Info	Only used inter-nally! Not stan-dard
0x00000100	Read	every	UINT32 [n]	1	[0, 1...255]	Returns the respective axis IDs in the channel number: [1...255] axis ID's: [0, 1...255]	Cannot be traced by oscil-loscope!

7.8.2.3 "Index offset" specification for Channel functions (Index group 0x2200 + ID)

Index offset (Hex)	Access	Channel type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Write	every	UINT32	1		load NC-program with program number	
0x00000002	Write	every	VOID			start Interpreter	
0x00000003	Write	every	VOID			RESERVED	
0x00000004	Write	every	UINT8[...]			load NC-program by name. The standard NC path does not also have to be given although it may. Other paths are also permitted.	
0x00000005	Write	every	UINT16	ENUM	cf. appendix interpreter operation mode [▸ 488]	set the interpreter/channel operation mode	
0x00000006	Write	Interpreter	UINT8[...]			set path for subroutines	
0x00000008	Write	Interpreter	UINT32	1		Interpreter simulation mode: 0: off (default) 1: on	not yet released
0x0000000F	Write	every	VOID			RESERVED	
0x00000010	Write	every	VOID			"Reset" Channel	
0x00000011	Write	every	VOID			"Stop" Channel	
0x00000012	Write	every	VOID			"Retry" Channel (restart Channel)	
0x00000013	Write	every	VOID			"Skip" Channel (skip task/block)	
0x00000014/0x00000015	Write	every	{			"Enable Retrace" /"Disable Retrace"	reserved function, no standard!
			UINT32	1	>0	Feeder directiion:1: forward, 2: backward	
			UINT32	1	≥ 0	Entry index	
			REAL64[3]	mm	±∞	Pos. of the main axes X, Y, Z	
			REAL64[5]	mm	±∞	Pos. of the auxiliary axes Q1, ..., Q5	
			}				
0x00000020	Write	every	VOID			"Save" zero offset shift (NPV)	
0x00000021	Write	every	VOID			"Load" zero offset shift (NPV)	
0x00000022	Write	every	VOID			"Save" tool compensations	
0x00000023	Write	every	VOID			"Load" tool compensations	
0x00000024	Write	Interpolation	{			Saves snapshot of the interpreter in a given file	
			char[32]			filename in TwinCAT \CNC-folder	
			UINT32	1	0..1	Mask: 0x1: R-Parameters 0x2: Zeroshifts 0x4: Tool Desc	
			}				
0x00000025	Write	Interpolation	{			Reads snapshot of a given file to the interpreter	
			char[32]			filename in TwinCAT \CNC-folder	
			UINT32	1	0..1	Mask: 0x1: R-Parameters 0x2: Zeroshifts 0x4: Tool Desc	
			}				

Index offset (Hex)	Access	Channel type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000026	Write	Inter-polation	VOID			set all tool parameters (incl. type & number) to null	
0x00000027	Write	Inter-polation	VOID			set all zero offset shifts to null	
0x00000030	Write	every	VOID			restart (Go Ahead) of the Interpreter after pro-grammed Interpreter stop	
0x00000040	Write	every	VOID			Triggerevent for deletion of any remaining travel in the NCI	
0x00000041	Write	every				RESERVED for fair events	
0x00000050	Write	Inter-polation	VOID	1		set *ExecIdleInfo*in the in-terpreter	reserved func-tion, no standard!
0x00000051	Write	Inter-polation	UINT32	1		set block skip mask in the interpreter parameter: *SkippingMask*	reserved func-tion, no standard!
0x00000052	Write	Inter-polation	UINT32	1		set *ItpOperationMode*in the interpreter parameter: OperationMode mask	reserved func-tion, no standard!
0x00000053	Write	Inter-polation	VOID			set *ScanningFlag*in the NC device	reserved func-tion, no standard!
0x00000054	Write	Inter-polation				scan position	reserved func-tion, no standard!
double[8]			position				
0x00000055	Write	Inter-polation				reserved	
0x00000056	Write	Inter-polation	VOID			set Interpreter in the *Aborted*state	reserved func-tion, no standard!
0x00000060	Write	Interp-polation	UINT16	1	0..159	manual reset of a fast M Function	

7.8.2.4 "Index offset" specification for cyclic Channel -process data (Index group 0x2300 + ID)

Index offset (Hex)	Access	Channel type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000000	Read	every (PLC→NC)	{128 Byte}		STRUCT s. Channel -interface	CHANNEL STRUCTURE (PLC→NC) Remark: Size and alignment changed.	Current PLC structure: NciChannelFromPlc*PLCTONC_NCICHANNEL_REF*
0x00000001	Read	every	UINT8[...] min. 30 Byte	1		Interpreter program display	Cannot be traced by oscilloscope!
0x00000002	Read / Write	every (PLC→NC)	UINT32	%	[0...1000000]	speed override Channel (Axis in the Channel)	1000000 = 100%
0x00000003	Read / Write	every (PLC→NC)	UINT32	%	[0...1000000]	speed override spindle	1000000 = 100%
0x00000080	Read	every (NC→PLC)	{160 Byte}		STRUCT s. Channel -interface	CHANNEL STRUCTURE (NC→PLC) Remark: Size and alignment changed.	Current PLC structure: NciChannelToPlc*NCTOPLC_NCICHANNEL_REF*
0x10000000 +RegIndex	Read / Write	every	REAL64	1	[0...999]	R parameter of the Interpreter	Cannot be traced by oscilloscope!
0x20000001	Read	every	UINT8[...] min. 30 Byte	1	[1...9]	program display of group attention handling (SAF)	Cannot be traced by oscilloscope!

7.8.3 Specification Groups

7.8.3.1 "Index offset" specification for group parameter (Index group 0x3000 + ID)

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	UINT32	1		group ID	
0x00000002	Read	every	UINT8[30+1]	1		group name	
0x00000003	Read	every	UINT32	1	ENUM	group type [▸ 488]	
0x00000004	Read	every	UINT32	µs		SAF-cycle time group	
0x00000005	Read	every	UINT32	µs		SVB-cycle time group	
0x00000006	Read / Write	every	UINT16	1	0/1	Single block operation mode?	
0x0000000B	Read	every	UINT32	1		Size of the SVB table (max. number of SVB entries	
0x0000000C	Read	every	UINT32	1		Size of the SAF table (max. number of SAF entries	
0x00000010	Read / Write	every	UINT32	1	[1,2...32] Default: 1	Internal SAF cycle time divisor (divides the internal SAF cycle time by this factor)	e.g. for DXD group
0x00000021	Read	Channel: every	UINT32	1		Channel ID	
0x00000022	Read	Channel: every	UINT8[30+1]	1		Channel name	
0x00000023	Read	Channel: every	UINT32	1	ENUM	Channel type [▸ 487]	
0x00000024	Read	Channel: every	UINT32	1	>0	Number in the Channel	
0x00000500	Read / Write	DXD group	INT32	ENUM	[0, 1]	Cornering velocity reduction method [▸ 488] 0: Coulomb-Scattering 1: Cosinus law 2: VeloJump	
0x00000501	Read / Write	DXD group	REAL64	1	[0.0...1.0]	Velocity reduction factor C0 transition (continuous, but neither once nor twice continuously differentiable)	
0x00000502	Read / Write	DXD group	REAL64	1	[0.0...1.0]	Velocity reduction factor C1 transition (continuous and continuously differentiable once)	
0x00000503	Read / Write	DXD group	REAL64	degree	[0.0...180.0]	Critical angle at segment transition "Low" (must be strictly less than or equal to the velocity reduction angle C0)	
0x00000504	Read / Write	DXD group	REAL64	degree	[0.0...180.0]	Critical angle at segment transition "High" (must be strictly less than or equal to the velocity reduction angle C0)	
0x00000505	Read / Write	DXD group	REAL64	mm/s	≥ 0	Minimum velocity, which must not be undershot at segment transitions, despite possible velocity reduction.	Attention: Parameter is not saved in the solution and is not transferred as NC boot parameter!
0x00000506	Read / Write	DXD group	REAL64	e.g. mm	[0.0...1000.0]	Radius of the tolerance sphere for blending	not implemented!

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000507	Read / Write	DXD group	REAL64	1		Velocity reduction factor C2 transition	
0x00000508	Read / Write	DXD group	UINT16	1	0/1	Enables calculation of the total remaining path length	NEW from TC3.1 B4020.40
0x00000509	Read / Write	DXD group	UINT16	1	0/1 Default: 1	General activation of the software limit position monitoring for the main axes (X, Y, Z) (see encoder parameters)	
0x0000050A	Read / Write	DXD group	UINT32	1	0/1	NCI Overridetype 0: related to internal reduced velocity (without iteration) 1: related to original external (programmed) velocity 2: Relative to the internally reduced velocity (0 ... >100%)	
0x0000050C	Read	DXD group	UINT32	1	[128 ... 1024] Default: 128	User-defined maximum number of the NCI SAF tables entries	NEW from TC3.1 B4014 boot parameters
0x00000510	Read / Write	DXD group	REAL64	1	≥ 0	For reduction method VeloJumpReductionfactor for C0 transitions: X axis	not implemented!
0x00000511	Read / Write	DXD group	REAL64	1	≥ 0	For reduction method VeloJumpReductionfactor for C0 transitions: Y axis	not implemented!
0x00000512	Read / Write	DXD group	REAL64	1	≥ 0	For reduction method VeloJumpReductionfactor for C0 transitions: Z axis	not implemented!
0x00000513	Read / Write	DXD group	LREAL64	1	]0.0..1.0[	Blending for auxiliary axes: If the effective path velo is smaller than the programmed one multiplied with this factor, then an accurate stop is inserted and the tolerance ball is deleted	not yet released
0x00000514	Read / Write	DXD group	UINT32	1	[1 … 20] Default: 1	Maximum number of transferred jobs per nc cycle (from SVB to SAF)	NEW from TC3.1 B4020.40
0x00000604	Read / Write	Encoder group	REAL64	e.g. mm/s	[0.0...1000.0]	Velocity window resp. standstill window	Base Unit / s
0x00000605	Read / Write	Encoder group	REAL64	s	[0.0...60.0]	Filter time for standstill window in seconds	
0x00000606	Read / Write	Encoder group	REAL64	s	[0.0...60.0]	Dead time compensation master/slave Coupling ("angle precontrol")	
0x00000701	Read	FIFO group	UINT32	1	[1...16]	FIFO dimension (m = number of axes) Note: The FIFO dimension was increased to 16.	(n x m) FIFO boot data
0x00000702	Read	FIFO group	UINT32	1	[1...10000]	FIFO size (length) (n = number of FIFO entries)	(n x m) FIFO boot data

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000703	Read	FIFO group	UINT32	1	[0, 1, 4]	Interpolation type for FIFO setpoint generator 0: INTERPOLATION-TYPE_LINEAR (default) 1: INTERPOLATION-TYPE_4POINT 4: INTERPOLATION-TYPE_CUBICSPLINE (with 6 points)	NEW from TC3.1 B4020
0x00000704	Read / Write	FIFO group	UINT32	1	[1, 2]	Override type for FIFO setpoint generator Type 1: OVERRIDE-TYPE_INSTANTANEOUS (default) Type 2: OVERRIDE-TYPE_PT2	
0x00000705	Read / Write	FIFO group	REAL64	s	> 0.0	P-T2-time for override change (T1=T2=T0)	
0x00000706	Read / Write	FIFO group	REAL64	s	≥ 0.0	Time delta for two sequenced FIFO entries (FIFO entry timebase)	
0x00000801	ReadWrite	Kinematic group	**Write**			Calculation of the kinematic forward transformation for the positions (ACS -> MCS)	
			{				
			REAL64[8]	e.g. degree	±∞	ACS (Axis Coordinate System) axis positions, max. dimension: 8	
			UINT32	1	≥ 0	Reserve	
			UINT32	1	≥ 0	Reserve	
			}				
			Read				
			{				
			REAL64[8]	e.g. mm	±∞	MCS (Machine Coordinate System) axis positions, max. dimension: 8	
			UINT32	1	≥ 0	Reserve	
			UINT32	1	≥ 0	Reserve	
			}				

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000802	ReadWrite	Kinematic group	**Write**			Calculation of the kinematic inverse transformation for the positions (MCS -> ACS)	
			{				
			REAL64[8]	e.g. mm	±∞	MCS (Machine Coordinate System) axis positions, max. dimension: 8	
			UINT32	1	≥ 0	Reserve	
			UINT32	1	≥ 0	Reserve	
			}				
			Read				
			{				
			REAL64[8]	e.g. degree	±∞	ACS (Axis Coordinate System) axis positions, max. dimension: 8	
			UINT32	1	≥ 0	Reserve	
			UINT32	1	≥ 0	Reserve	
			}				

7.8.3.2 "Index offset" specification for Group state (Index group 0x3100 + ID)

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	INT32	1	ENUM	error code group [▸ 321]	
0x00000002	Read	every	UINT32	1		number of master axes	
0x00000003	Read	every	UINT32	1		number of slave axes	
0x00000004	Read	every	UINT32	1	s. ENUM	SVB group state (state)	
0x00000005	Read	every	UINT32	1	s. ENUM	SAF group state (main state)	
0x00000006	Read	every	UINT32	1	s. ENUM	moving state (state)	
0x00000007	Read	every	UINT32	1	s. ENUM	SAF sub-group state (sub state)	
0x00000008	Read	every	UINT32	1	s. ENUM	Referencing state (state)	
0x00000009	Read	every	UINT32	1	s. ENUM	Coupling state (state)	Cannot be traced by oscilloscope!
0x0000000A	Read	every	UINT32	1	≥0	Coupling table index	Cannot be traced by oscilloscope!
0x0000000B	Read	every	UINT32	1	≥0	current number of SVB entries/tasks	*Symbolic access: 'SvbEntries' (DXD)*
0x0000000C	Read	every	UINT32	1	≥0	current number of SAF entries/tasks	*Symbolic access: 'SafEntries' (DXD)*
0x0000000D	Read	every	UINT32	1		Current block number (only active for interpolation group)	*Symbolic access:'block number' (DXD)*
0x0000000E	Read	every	UINT32	1	≥0	current number of free SVB entries/tasks	Cannot be traced by oscilloscope!
0x0000000F	Read	every	UINT32	1	≥0	current number of free SAF entries/tasks	Cannot be traced by oscilloscope!
0x00000011	Read	every	UINT16	1	0/1	Emergency Stop (E-Stop) active?	Cannot be traced by oscilloscope!

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000110	Read	PTP group	{			Internal NC information (resolutions)	reserved!
			REAL64	e.g. mm	± ∞	ExternalEndPosition	
			REAL64	e.g. mm/s	>0	ExternalTargetVelocity	
			REAL64	e.g. mm/s^2	>0	ExternalAcceleration	
			REAL64	e.g. mm/s^2	>0	ExternalDeceleration	
			REAL64	e.g. mm/s^3	>0	ExternalJerk	
			UINT32	1	>0	ExternalOverrideType	
			REAL64	e.g. mm	± ∞	InternalEndPosition	
			REAL64	e.g. mm/s	>0	InternalTargetVelocity (refers to 100 %)	
			REAL64	%	[0 ... 100]	InternalActualOverride	
			REAL64	e.g. mm/s^2	>0	InternalAcceleration	
			REAL64	e.g. mm/s^2	>0	InternalDeceleration	
			REAL64	e.g. mm/s^3	>0	InternalJerk	
			REAL64	e.g. mm	>0	PositionResolution	
			REAL64	e.g. mm/s	≥0	VelocityResolution	
			REAL64	e.g. mm/s^2	≥0	AccelerationResolution	
			REAL64	e.g. mm/s	≥0	VelocityResolutionAtAccelerationZero	
			}				
0x00000500	Read	DXD group	REAL64	e.g. mm	≥ 0	Path rest way (remaining arc length) on the current path segment	*Symbolic access: 'SetPathRemLength'*
0x00000501	Read	DXD group	REAL64	e.g. mm	≥ 0	racked out arc length on the current path segment	*Symbolic access: 'SetPathLength'*
0x00000502	Read	DXD group	REAL64	e.g. mm/s	≥ 0	current path set velocity	*Symbolic access: 'SetPathVelo'*
0x00000503	Read	DXD group	REAL64	e.g. mm/s^2	± ∞	current path set acceleration	*Symbolic access: 'SetPathAcc'*
0x00000504	Read	DXD group	REAL64	e.g. mm/s^2	≥ 0	amount of the current vectorial set acceleration	*Symbolic access: 'SetPathAbsAcc'*
0x00000505	Read	DXD group	REAL64	e.g. mm/s	≥ 0	maximum segment end path set velocity	*Symbolic access: 'SetPathVeloEnd'*
0x00000506	Read	DXD group	REAL64	e.g. mm/s	≥ 0	Segment maximum path set velocity	*Symbolic access: 'SetPathVeloMax'*
0x00000507	Read	DXD group	REAL64	e.g. mm	≥ 0	current relative braking distance based on the current arc length	*Symbolic access: 'SetPathStopDist'*
0x00000508	Read	DXD group	REAL64	e.g. mm	± ∞	Safety distance = segment arc length - current arc length - relative braking distance	*Symbolic access:'SetPathSecurityDist'*
0x00000509	Read	DXD group	REAL64	1	0/1	Segment transition	*Symbolic access:'SetPathSegmentChange'*

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x0000050A	Read	DXD group	REAL64	%	[0 ... 100]	path velocity override	*Symbolic access:'SetPathOverride'*
0x00000511	Read	DXD group	REAL64	e.g. mm/s	≥ 0	amount of the path actual velocity	*Symbolic access: 'ActPathAbsVelo'*
0x00000512	Read	DXD group	REAL64	e.g. mm/s^2	± ∞	path actual acceleration on the current segment	*Symbolic access: 'ActPathAcc'*
0x00000513	Read	DXD group	REAL64	e.g. mm/s^2	≥ 0	amount of the path actual acceleration on the current segment	*Symbolic access:'ActPathAbsAcc'*
0x00000514	Read	DXD group	REAL64	e.g. mm	± ∞	Position error on the path in tangential direction (signed to indicate leading and lagging)	*Symbolic access:'PathDiffTangential'*
0x00000515	Read	DXD group	REAL64	e.g. mm	≥ 0	Position error on the path in orthogonal direction	Symbolic access:*'PathDiffOrthogonal'*
0x00000520	Read	DXD-group	REAL64	1	≥ 0	Covered arc length of the current segment, normalized to 1.0	
0x00000521	Read	DXD-group	REAL64	1	0/1	Change of partial segment (radius of tolerance ball)	
0x00000522	Read	DXD group	REAL64	1	≥ 0	Total remaining path length to the last geometry entry or the next accurate stop. Refers to group parameter 0x508.	
0x00000523	Read	DXD group	REAL64	1	≥ 0	programmed velocity of the current segment	
0x00000530	Read	DXD group	{			Current or last target position of the main axes X, Y and Z	
			REAL64	e.g. mm	± ∞	Target position X-axis	
			REAL64	e.g. mm	± ∞	Target position Y-axis	
			REAL64	e.g. mm	± ∞	Target position Z-axis	
			}				
0x00000531	Read	DXD group	{			Current or last target position of the auxiliary axes Q1 to Q5	
			REAL64[5]	e.g. mm	± ∞	Target position of axis Q1 to Q5	
			}				
0x00000532	Read	DXD group	{			Reads path length, H parameter and Entry ID of the next 11 segments in relation to the current DC time	not generally released
			UINT32			DC Time	
			UINT32			reserved	
			PreViewTab[11]			11*24 Bytes	
			}				
			PreViewTab				
			{				
			REAL64	e.g. mm		Segment length	
			UINT32	1		block number	
			UINT32	1		H-Parmeter	
			UINT32	1		Entry ID	
			UINT32	1		reserved	
			}				

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x0000054n	Read	DXD group	REAL64	1	0/1	within the tolerance ball of the auxiliary axis n = 1..5 number of the auxiliary axis (not axis ID)	
0x00000550	Read	DXD group	{			Reading axis IDs inside the 3D-group:	
			UINT32	1	[0, 1...255]	X axis ID	
			UINT32	1	[0, 1...255]	Y axis ID	
			UINT32	1	[0, 1...255]	Z axis ID	
			}				
0x00000552	Read	DXD group FIFO group Kinematic group	{ UINT32[m] }	1	[0, 1...255]	axis allocation of the group: First axis ID. ..., m.-axis-IDm: dimension of the 3D group with main and auxiliary axes (X, Y, Z, Q1, Q2, Q3, Q4, Q5) or the FIFO group or the ACS axes of the kinematic group	
0x00000553	Read	Kinematic group	{			Reading the axis allocation (ID's) inside the kinematic group:	
			UINT32[8]	1	[0, 1...255]	MCS-axis ID's (Machine Coordinate System)	
			UINT32[8]	1	[0, 1...255]	ACS-axis ID's (Axis Coordinate System)	
			UINT32	1	≥ 0	Reserve	
			UINT32	1	≥ 0	Reserve (NEW)	
			}				
0x0000056n	Read	DXD group	REAL64	1	± ∞	current position error of the auxiliary axis within the tolerance ball (set value side only) Only for auxiliary axes n = 1..5 number of the auxiliary axis (not axis ID)	

7.8.3.3 "Index offset" specification for Group functions (Index group 0x3200 + ID)

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Write	every	VOID			Reset group	
0x00000002	Write	every	VOID			Stop group	
0x00000003	Write	every	VOID			Clear group (buffer/ task)	
0x00000004	Write	PTP group, 3D group	{			Emergency stop (E-stop) (emergency stop with controlled ramp)	
			REAL64	e.g. mm/ s^2	≥ 0.0	Deceleration (must be greater than or equal to the original deceler-ation)	
			REAL64	e.g. mm/ s^3	≥ 0.0	Jerk (must greater than or equal to the original jerk)	
			}				
0x00000005	Write	PTP group	{			Parameterizable stop (with controlled ramp)	Reserved func-tion, no standard!
			REAL64	e.g. mm/ s^2	≥ 0.0	deceleration	
			REAL64	e.g. mm/ s^3	≥ 0.0	Jerk	
			}				
0x00000006	Write	PTP group, 3D group	VOID			"Step on" after Emer-gency Stop (E-Stop)	
0x00000050	Write	PTP group, 3D group	{			axis allocation of the group:	
			UINT32	1	[0, 1...255]	X axis ID	
			UINT32	1	[0, 1...255]	Y axis ID	
			UINT32	1	[0, 1...255]	Z axis ID	
			}				
0x00000051	Write	PTP group, 3D group FIFO group	{			axis allocation of the group:	
			UINT32	1	[1...255]	Axis ID	
			UINT32	1	[0 ... (m-1)]	Place index of the axis in the group m: group dimension (PTP: 1;DXD: 3, FIFO: 16)	
			}				
0x00000052	Write	3D group FIFO group	{ UINT32[m] }	1	[0, 1...255]	axis allocation of the group: First axis ID. ..., m.-axis-IDm: dimen-sion of the 3D group (X, Y, Z, Q1, Q2, Q3, Q4, Q5) resp. FIFO group	
0x00000053	Write	3D group FIFO group Kinematic group	VOID			Delete the 3D axis al-location, FIFO axis al-location or Kinematic axis allocation and re-turn of the axes to their own PTP groups	
0x00000054	Write	Kinematic group	{			axis allocation of the kinematic group:	
			UINT32[8]	1	[0, 1...255]	MCS-axis ID's (Ma-chine Coordinate Sys-tem)	
			UINT32[8]	1	[0, 1...255]	ACS-axis ID's (Axis Coordinate System)	
			UINT32	1	≥ 0	Reserved	
			UINT32	1	≥ 0	Reserved (NEW)	
			}				
0x00000060	ReadWrite	3D group		1		internal "feed group" command ("Feeder")	Execute com-mand!

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000061	ReadWrite	3D group		1		internal "feed group" command ("Feeder")	Execute command!
0x00000110	Write	1D group	VOID			Reference 1D group ("calibration")	
0x00000111	Write	1D group	{			New end position 1D group	
			UINT32	ENUM	s. appendix	End position type [▸ 490] (s. appendix)	
			REAL64	e.g. mm	±∞	new end position (target position)	
			}				
0x0000011A	Write	1D group	{			set actual position 1D group	**caution by using!** Always to SAF Port 501!
			UINT32	ENUM	s. appendix	Actual position type [▸ 490] (s. appendix)	
			REAL64	e.g. mm	±∞	actual position for axis	
			}				
0x0000011B	Write	1D group	UINT32	1	0/1	Set reference flag ("calibrate flag")	**caution by using!**
0x00000120	Write	1D group	{			start 1D group (standard start):	
			UINT32	ENUM	s. appendix	Start type [▸ 489] (s. appendix)	
			REAL64	e.g. mm	±∞	End position (target position)	
			REAL64	mm/s	≥ 0.0	required velocity	
			}				
0x00000121	Write	1D group (SERVO)	{			start 1D group (extended start):	
			UINT32	ENUM	s. appendix	Start type [▸ 489] (s. appendix)	
			REAL64	e.g.mm	±∞	End position (target position)	
			REAL64	mm/s	≥0.0	required velocity	
			UINT32	1	0/1	Standard acceleration?	
			REAL64	mm/s^2	≥ 0.0	Acceleration	
			UINT32	1	0/1	Standard deceleration?	
			REAL64	mm/s^2	≥ 0.0	deceleration	
			UINT32	1	0/1	Standard jerk?	
			REAL64	mm/s^3	≥ 0.0	Jerk	
			}				

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000122	Write	1D group (MW servo)	{			Start 1D group (special start):	Reserved start function, no standard!
			UINT32	ENUM	s. appendix	Start type [▸ 489] (s. appendix)	
			REAL64	e.g. mm	±∞	End position (target position)	
			REAL64	mm/s	≥0.0	required start velocity	
			REAL64	e.g. mm	±∞	Position for a new velocity level	
			REAL64	mm/s	≥0.0	new end velocity level	
			UINT32	1	0/1	Standard acceleration?	
			REAL64	mm/s^2	≥0.0	Acceleration	
			UINT32	1	0/1	Standard deceleration?	
			REAL64	mm/s^2	≥0.0	deceleration	
			UINT32	1	0/1	Standard jerk?	
			REAL64	mm/s^3	≥0.0	Jerk	
			}				
0x00000126	Write	1D group	{			Start drive output:	
			UINT32	ENUM	s. appendix	Output type [▸ 496] (s. appendix)	
			REAL64	e.g. %	±∞	required output value (e.g. %)	
			}				
0x00000127	Write	1D group	VOID			Stop drive output	
0x00000128	Write	1D group	{			Change the drive output:	
			UINT32	ENUM	s. appendix	Output type [▸ 496] (s. appendix)	
			REAL64	e.g. %	±∞	required output value (e.g. %)	
			}				
0x00000130	Write	1D group (SERVO)	{			1D section compensation (SERVO):	
			UINT32	ENUM	s. appendix	Compensation type [▸ 490] (s. appendix)	
			REAL64	mm/s/s	≥ 0.0	Max. acceleration increase	
			REAL64	mm/s/s	≥ 0.0	Max. deceleration increase	
			REAL64	mm/s	≥ 0.0	Max. increase velocity	
			REAL64	mm/s	≥ 0.0	Base velocity for the process	
			REAL64	e.g. mm	±∞	Path difference to be compensated	
			REAL64	e.g. mm	≥ 0.0	Path distance for compensation	
			}				
0x00000131	Write	1D group SERVO	VOID			Stop section compensation (SERVO)	

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000140 (0x00n00140)	Write	Master/Slave coupling: 1D group(SERVO)	{			Master/slave coupling (SERVO):	Extension for "flying saw"!angle >0.0 and <= 90.0 degrees(parallel saw: 90.0 degrees)
			UINT32	ENUM	s. appendix	Slave type/coupling type [▶ 491] (s. appendix)	
			UINT32	1	[1...255]	Axis ID of the master axis/group	
			UINT32	1	[0...8]	Subindex n of the master axis (default: -value: 0)	
			UINT32	1	[0...8]	Subindex n of the slave axis (default: -value: 0)	
			REAL64	1	[±1000000.0]	Parameter 1: linear: Gearing factor FlySawVelo: Reserve FlySaw: Abs. synchronous position master [mm]	
			REAL64	1	[±1000000.0]	Parameter 2: linear: Reserve FlySawVelo: Reserve FlySawPos: Abs. synchronous position slave [mm]	
			REAL64	1	[±1000000.0]	Parameter 3: linear: Reserve FlySawVelo: Angle of inclination in [DEGREE] FlySawPos: angle of inclination in [DEGREE]	
			REAL64	1	[±1000000.0]	Parameter 4: linear: Reserve FlySawVelo: Gearing factor FlySawPos: Gearing factor	
			}				
0x00000141	Write	Master/Slave decoupling: 1D group(SERVO)	VOID			Master/slave decoupling (SERVO)	
0x00000142	Write	Master / slave parameter 1D group(servo)	{			Change of the coupling parameters (SERVO):	
			REAL64	1	[±1000000.0]	Parameter 1: linear: Gearing factor	
			REAL64	1	[±1000000.0]	Parameter 2: Linear: Reserve	
			REAL64	1	[±1000000.0]	Parameter 3: Linear: Reserve	
			REAL64	1	[±1000000.0]	Parameter 4: Linear: Reserve	
			}				
0x00000144	Write	Slave stop 1D group (SERVO)	VOID			Stop the "flying saw" (SERVO)	Only for "flying saw"
0x00000149	Write	Slave tables 1D group (SERVO)	REAL64	1	±∞	set the slave table scaling of a solo table coupling (SERVO)	Only for Solo table slave
0x00000150	Write	1D group	VOID			Deactivate complete 1D group/axis (disable)	
0x00000151	Write	1D group	VOID			Activate complete 1D group / axis (enable)	

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000160	Write	1D group	VOID			Deactivate drive output of the 1D group (disable)	
0x00000161	Write	1D group	VOID			Activate drive output of the 1D group (enable)	
0x00000362	Write	High/low speed group	UINT16	1	0/1	Release parking brake? 0: automatic activation (default) 1: mandatorily always released!	
0x00000701	Write	FIFO group	VOID			Start FIFO group (FIFO table must have been filled in advance)	(n*m)-FIFO
0x00000710	Write	FIFO group	{ REAL64[x*m]}	e.g. mm	±∞	Write x FIFO entries (lines): (x*m)-values (one or more lines)n: FIFO length (number of lines)m: FIFO dimension (number of columns) range of values x: [1 ... n]	Only possible on a line-by-line basis! (integer multiple)
0x00000711	Write	FIFO group	{ REAL64[x*m]}	e.g. mm	±∞	Overwrite the last x FIFO entries (lines): (x*m)-values (one or more lines)n: FIFO length (number of lines)m: FIFO dimension (number of columns) range of values x: [1 ... n]	Only possible on a line-by-line basis! (integer multiple)
0x00000801	Write	Kinematic group	VOID			Start Kinematic group	reserved function, no standard!

7.8.4 Specification Axes

7.8.4.1 "Index offset" specification for Axis parameter (Index group 0x4000 + ID)

Index-Offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Remarks
0x00n00000	Read	every (Structure for all axis parameters)	{			General AXIS PARAMETER STRUCTURE (NC/CNC), also includes subelements such as encoder, controller and drive (see MC_ReadParamSet in TcMc2.lib) Note: Size and alignment changed.	Modified from TC3
			UINT32	1		Axis ID	
			UINT8[30+1+1]	e.g. mm		Axis name	
			UINT32	1		Axis type [▸ 488]	
			. . .	. . .	. . .	. . .	
			}			1024 bytes (instead of 512 bytes)	
0x00000001	Read	every	UINT32	1		Axis ID	
0x00000002	Read	every	UINT8[30+1]	1		Axis name	
0x00000003	Read	every	UINT32	ENUM		Axis type [▸ 488]	
0x00000004	Read	every	UINT32	µs		cycle time axis (SAF)	
0x00000005	Read	every	UINT8[10+1]	1		physical unit	
0x00000006	Read / Write	every	REAL64	e.g. mm/s		ref. velocity in cam direction	
0x00000007	Read / Write	every	REAL64	e.g. mm/s		ref. velocity in sync direction	
0x00000008	Read / Write	every	REAL64	e.g. mm/s		velocity hand slow	
0x00000009	Read / Write	every	REAL64	e.g. mm/s		velocity hand fast	
0x0000000A	Read / Write	every	REAL64	e.g. mm/s	[0.0...1.0E20]	velocity rapid traverse	
0x0000000F	Read / Write	every	UINT16	1	0/1	position range monitoring?	
0x00000010	Read / Write	every	REAL64	e.g. mm	[0.0...1.0E6]	position range window	
0x00000011	Read / Write	every	UINT16	1	0/1	motion monitoring?	
0x00000012	Read / Write	every	REAL64	s	[0.0...600]	motion monitoring time	
0x00000013	Read / Write	every	UINT16	1	0/1	loop?	
0x00000014	Read / Write	every	REAL64	e.g. mm		loop movement (±)	
0x00000015	Read / Write	every	UINT16	1	0/1	target position monitoring?	
0x00000016	Read / Write	every	REAL64	e.g. mm	[0.0...1.0E6]	target position window	
0x00000017	Read / Write	every	REAL64	s	[0.0...600]	target position monitoring time	
0x00000018	Read / Write	every	REAL64	e.g. mm		pulse way in pos. direction	
0x00000019	Read / Write	every	REAL64	e.g. mm		pulse way in neg. direction	
0x0000001A	Read / Write	every	UINT32	ENUM (≥0)		Error reaction mode: 0: instantaneous (default) 1: delayed (e.g. for master/slave coupling)	
0x0000001B	Read / Write	every	REAL64	s	[0...1000]	Error reaction delay (if error reaction mode "delayed" is activated)	
0x0000001C	Read / Write	every	UINT16	1	0/1	Couple slaves via actual values if not operational?	

Index-Offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Remarks
0x0000001D	Read / Write	every	REAL64	e.g. mm/ s^2	[0, 0.01...1.0E10]	Acceleration for fading profile when switching from SET to ACTUAL values: Default: 0 (in this case the minimum from the axis acceleration is used, i.e. MIN(Acc, Dec))	
0x0000001E	Read / Write	every	UINT32	ENUM (≥0)		Fast Axis Stop Signal Type: Selection of the signal type that triggers a fast axis stop (see bit 7 in Drive->nStatus4) *"0 (SignalType_OFF)",* *"1 (Signal-Type_RisingEdge)","2 (Signal-Type_FallingEdge)","3 (SignalType_Both-Edges)","4 (Signal-Type_HighActive)","5 (SignalType_LowAc-tive)"*	
0x00000020	Read / Write	every	UINT16	1	0/1	Allow motion com-mands for slave axis? Default: FALSE	
0x00000021	Read / Write	every	UINT16	1	0/1	Allow motion com-mands for axes with active external set-point generator? Default: FALSE	
0x00000026	Read / Write	every	UINT32	1		Interpretation of the units (position, veloc-ity, time) bit 0: velocity in x/min instead of x/s Bit 1: position in thou-sands of the base unit Bit 2: modulo position display	see encoder! bit array
0x00000027	Read / Write	every	REAL64	e.g. mm/ s	[>0...1.0E20]	max. allowed velocity	
0x00000028	Read / Write	every	REAL64	e.g. mm	[0.0...1.0E6]	motion monitoring win-dow	
0x00000029	Read / Write	every	UINT16	1	0/1	PEH time monitoring?	Posi. end and accurate stop
0x0000002A	Read / Write	every	REAL64	s	[0.0...600]	PEH monitoring time	
0x0000002B	Read / Write	every	UINT16	1	0/1	Backlash compensa-tion?	
0x0000002C	Read / Write	every	REAL64	e.g. mm	[-1000.0 ...1000.0]	Backlash	
0x00000030	Read	every	UINT16	1	[0,1]	Persistent data e.g. for actual position and ref-erence state of the en-coder?	Boot parame-ters, cannot be changed online.
0x00000031	Read	every	{ UINT8[6] UINT16 UINT16 } 10 bytes	Am-sAddr: Am-sNetId, Am-sPortNo. Chan-nelNo	1	Reading of the hard-ware AMS address (AMS Net ID and AMS port no.) and Ether-CAT channel number (communication chan-nel 0,1,2,3…)	

Index-Offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000031	Read	every	{ UINT8[6] UINT16 UINT16 // UINT16 UINT32 UINT32 UINT32 UINT32 UINT32 UINT32 UINT32 UINT32 UINT32 UINT32[4] } 64 bytes	AmsAddr: AmsNetId, AmsPortNo. ChannelNo Reserved NcDriveID NcDriveIndex NcDriveType NcEncID NcEncIndex NcEncType NcAxisID NcAxisType SoftDriveObjectId Reserved	1	Reading of the hardware AMS address (AMS Net ID and device AMS port no.) and EtherCAT channel number (communication channel 0,1,2,3…) Supplemented by additional NC information such as NcDriveID, NcDriveType (see appendix) etc..	NEW from TC3 SoftDriveObjectId from NC build 4226
0x00000033	Read	every	{ UINT16 ApplRequestBit UINT16 ApplRequestType UINT32 ApplCmdNo UINT32 ApplCmdVersion . . . } 1024 bytes	1 1 Not implemented 1	0/1 ≥0 >0 ≥0	General APPLICATION REQUEST STRUCTURE (NC/NCI), e.g. for Application-Homing request (s. *MC_ReadApplicationRequest* in *TcMc2.lib*) Application request types: 0: NONE (IDLE) 1: HOMING	Changed in TC3
0x00000051	Read	Channel: every	UINT32			Channel ID	
0x00000052	Read	Channel: every	UINT8[30+1]			Channel name	
0x00000053	Read	Channel: every	UINT32			Channel type [▶ 487]	
0x00000054	Read	Group: every	UINT32			group ID	
0x00000055	Read	Group: every	UINT8[30+1]			group name	
0x00000056	Read	Group: every	UINT32			group type [▶ 488]	
0x00000057	Read	every	UINT32			Number of encoders	
0x00000058	Read	every	UINT32			Number of controllers	
0x00000059	Read	every	UINT32			Number of drives	
0x0000005A	Read	every	{			read all sub-elements of an axis:	
			UINT32[9]	1	[0, 1...255]	axis encoder IDs	
			UINT32[9]	1	[0, 1...255]	axis controller IDs	
			UINT32[9]	1	[0, 1...255]	axis drive IDs	
			} 108 bytes				
0x000000F1	Read / Write	every	REAL64	e.g. mm/s^2	Default: 1.0E5	Maximum permitted acceleration	NEW from TC 3.2
0x000000F2	Read / Write	every	REAL64	e.g. mm/s^2	Default: 1.0E6	Maximum permitted deceleration	NEW from TC 3.2

Index-Offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000101	Read / Write	Servo	REAL64	e.g. mm/ s^2	[0.01...1.0E20]	Acceleration (default data set)	
0x00000102	Read / Write	Servo	REAL64	e.g. mm/ s^2	[0.01...1.0E20]	Deceleration (default data set)	
0x00000103	Read / Write	Servo	REAL64	e.g. mm/ s^3	[0.1...1.0E30]	Jerk (default data set)	
0x00000104	Read / Write	Servo	REAL64	s	[0.0 ... 1.0] Default: 0.0 s	damping time between velocity and position values of the setpoint generator in seconds	
0x00000105	Read / Write	Servo	UINT32	ENUM	Default: type 1	Override type [▸ 489] for velocity: 1: related to internal reduced velocity (without iteration) 2: related to original external start velocity (without iteration) 3: related to internal reduced velocity (optimization by means of iteration) 4: related to original external start velocity (optimization by means of iteration)	
0x00000106	Read / Write	Servo	REAL64	1	[0.0 ... 1.0E6] Default: 0.0	Maximum permitted step change in velocity for dynamic reduction *DV = factor *min(A+, A-) * DT*	
0x00000107	Read / Write	Servo	UINT16	1	[0,1] Default: 1	activates acceleration and jerk limitation for the auxiliary axis (Q1 to Q5)	
	Read / Write	Servo	REAL64	e.g. mm	[0.0..1000.0]	Radius of the tolerance sphere for the auxiliary axes	
	Read / Write	Servo	REAL64	e.g. mm	[0.0..10000.0]	maximum allowed position deviation if the tolerance sphere is reduced Only for auxiliary axes	
0x0000010A	Read / Write	Servo	REAL64	e.g. mm/ s^2	[0.01 ... 1.0E20]	Fast Axis Stop: Acceleration (s.a. Fast Axis Stop Signal Type)	
0x0000010B	Read / Write	Servo	REAL64	e.g. mm/ s^2	[0.01 ... 1.0E20]	Fast Axis Stop: deceleration (s.a. Fast Axis Stop Signal Type)	
0x0000010C	Read / Write	Servo	REAL64	e.g. mm/ s^3	[0.1 ... 1.0E30]	Fast Axis Stop: Jerk (s.a. Fast Axis Stop Signal Type)	
0x00000201	Read / Write	Stepper motor	UINT32	ENUM		operation mode stepper motor	
0x00000202	Read / Write	Stepper motor	REAL64	e.g. mm/ STEP	[1.0E-6 ... 1000.0]	distance scaling of a motor step	
0x00000203	Read / Write	Stepper motor	REAL64	e.g. mm/ s	[0.0 ... 1000.0]	minimum velocity for velocity profile	
0x00000204	Read / Write	Stepper motor	UINT32	1	[0 ... 100]	number of steps per frequency/velocity step	
0x00000205	Read / Write	Stepper motor	UINT32	1		motor mask as sync pulse	Not implemented !
0x00000301	Read / Write	high/low	REAL64	e.g. mm	[0.0 ... 100000.0]	creep distance in pos. direction	

Index-Offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000302	Read / Write	high/low	REAL64	e.g. mm	[0.0 ... 100000.0]	Creep distance in neg. direction	
0x00000303	Read / Write	high/low	REAL64	e.g. mm	[0.0 ... 100000.0]	braking distance in pos. direction	
0x00000304	Read / Write	high/low	REAL64	e.g. mm	[0.0 ... 100000.0]	Braking distance in neg. direction	
0x00000305	Read / Write	high/low	REAL64	s	[0.0 ... 60.0]	braking deceleration in pos. direction	
0x00000306	Read / Write	high/low	REAL64	s	[0.0 ... 60.0]	Braking decel. in neg. direction	
0x00000307	Read / Write	high/low	REAL64	s	[0.0 ... 60.0]	Switching time from high to low speed	
0x00000308	Read / Write	high/low	REAL64	e.g. mm	[0.0 ... 100000.0]	creep distance stop	
0x00000309	Read / Write	high/low	REAL64	s	[0.0 ... 60.0]	damping time to re-lease brake	
0x0000030A	Read / Write	high/low	REAL64	s	[0.0 ... 60.0]	pulse time in pos. di-rection	
0x0000030B	Read / Write	high/low	REAL64	s	[0.0 ... 60.0]	Pulse time in neg. di-rection	
ENCODER:							
0x00n10001	Read	Encoder: every	UINT32	1	[1 ... 255]	Encoder ID n = 0: Standard encoder for the axes > 0: n-th en-coder for the axis (op-tional)	
0x00n10002	Read	Encoder: every	UINT8[30+1]	1	30 characters	encoder name	
0x00n10003	Read	Encoder: every	UINT32	1	s. ENUM (>0)	encoder type [▸ 493]	
0x00n10004	Read / Write	Encoder: every	UINT32	1	Byteoffset	Input address offset (IO-Input-Image)	change I/O ad-dress
0x00n10005	Read / Write	Encoder: every	UINT32	1	Byteoffset	Output address offset (IO-Output-Image)	change I/O ad-dress
0x00n10006	Read / Write	Encoder: every	REAL64	e.g. mm/ INC	[1.0E-12 ... 1.0E+30]	resulting scaling factor (numerator / denomi-nator) **Note:** from TC3 the scaling factor consists of two components – numerator and de-nominator (default: 1.0).	
0x00n10007	Read / Write	Encoder: every	REAL64	e.g. mm	[±1.0E+9]	Position offset	
0x00n10008	Read / Write	Encoder: every	UINT16	1	[0,1]	encoder count direc-tion	
0x00n10009	Read / Write	Encoder: every	REAL64	e.g. mm	[0.001 ... 1.0E +9]	modulo factor	
0x00n1000A	Read / Write	Encoder: every	UINT32	1	s. ENUM (>0)	encoder mode [▸ 494]	
0x00n1000B	Read / Write	Encoder: every	UINT16	1	0/1	soft end min. monitor-ing?	
0x00n1000C	Read / Write	Encoder: every	UINT16	1	0/1	soft end max. monitor-ing?	
0x00n1000D	Read / Write	Encoder: every	REAL64	mm		soft end position min.	
0x00n1000E	Read / Write	Encoder: every	REAL64	mm		soft end position max.	
0x00n1000F	Read / Write	Encoder: every	UINT32	1	s. ENUM (≥0) in the appendix	Encoder evaluation di-rection [▸ 494] (enable log. counting direction)	
0x00n10010	Read / Write	Encoder: every	REAL64	s	[0.0...60.0]	filter time for actual position value in sec-onds (P-T1)	
0x00n10011	Read / Write	Encoder: every	REAL64	s	[0.0...60.0]	filter time for actual ve-locity value in seconds (P-T1)	

Index-Offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Remarks
0x00n10012	Read / Write	Encoder: every	REAL64	s	[0.0...60.0]	filter time for actual acceleration value in seconds (P-T1)	
0x00n10013	Read / Write	Encoder: every	UINT8[10+1]	1		physical unit	Not implemented !
0x00n10014	Read / Write	Encoder: every	UINT32	1		Interpretation of the units (position, velocity, time) bit 0: velocity in x/min instead of x/sBit 1: position in thousands of the base unit	Not implemented ! bit array
0x00n10015	Read	Encoder: every	UINT32	INC	[0x0... 0xFFFFFFFF]	Encoder mask (maximum value of the encoder actual value in increments) **Note:** The encoder mask may be any numerical value (e.g. 3600000). Unlike in the past, it no longer has to correspond to a continuous series off binary one's (2^n-1).	Read-only parameter s. parameter "Encoder Sub Mask"
0x00n10016	Read / Write	Encoder: every	UINT16	1	0/1	Actual position correction (measurement system error correction)?	
0x00n10017	Read / Write	Encoder: every	REAL64	s	[0.0...60.0]	Filter time for actual position correction in seconds (P-T1)	
0x00n10019	Read / Write	Encoder: every	UINT32	1	s. ENUM (≥0) in the appendix	Encoder absolute dimensioning system [▶ 494]	
0x00n1001A	Read	Encoder: every	UINT32	1	s. ENUM (≥0)	Encoder position initialization	Not implemented !
0x00n1001B	Read / Write	Encoder: every	REAL64	e.g. mm	[≥0, modulo factor/2]	tolerance window for modulo-start	
0x00n1001C	Read	Encoder: every	UINT32	1	s. ENUM (≥0)	encoder sign interpretation [▶ 494] (data type)	
0x00n1001D	Read	Encoder: every	UINT16	1	0/1	Incremental or absolute encoder? 0: incremental encoder type 1: absolute encoder type	
0x00n10023	Read / Write	Encoder: every	REAL64	e.g. mm/ INC	[1.0E-12 ... 1.0E+30]	component of the scaling factor: numerator (=> scaling factor numerator / scaling factor denominator)	NEW from TC3
0x00n10024	Read / Write	Encoder: every	REAL64	1	[1.0E-12 ... 1.0E+30]	component of the scaling factor: denominator (=> scaling factor numerator / scaling factor denominator) Default: 1.0	NEW from TC3

0x00n10025	Read / Write	Encoder: every	{
			REAL64
			REAL64
			}
				
				e.g. mm/ INC
				1
					
					[1.0E-12 ... 1.0E+30]
					[1.0E-12 ... 1.0E+30]
						
						component of the scaling factor: numerator
						component of the scaling factor: denominator
						(=> scaling factor numerator / scaling factor denominator)
							NEW from TC3

Index-Offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Remarks
0x00n10030	Read / Write	Encoder: every	UINT32	1		Internal encoder control double word for specifying the operation modes and properties	NEW from TC3
0x00n10101	Read / Write	E: INC	UINT16	1	[0,1]	inverse search direction for ref.cam?	
0x00n10102	Read / Write	E: INC	UINT16	1	[0,1]	inverse search direction for sync pulse?	
0x00n10103	Read / Write	E: INC	REAL64	e.g. mm	[±1000000.0]	Reference position	
0x00n10104	Read / Write	E: INC	UINT16	1	[0,1]	distance monitoring between Ref. cams and sync pulse active?	Not implemented !
0x00n10105	Read / Write	E: INC	UINT32	INC	[0 ... 65536]	minimum distance between Ref. cams and sync pulse in increments	Not implemented !
0x00n10106	Read / Write	E: INC	UINT16	1	[0,1]	external sync pulse?	
0x00n10107	Read / Write	E: INC	UINT32	1	s. ENUM (>0) in the appendix	reference mode [▸ 495]	
0x00n10108	Read / Write	E: INC	UINT32	1	[0x0000000F... 0xFFFFFFFF] binary mask: (2^n - 1)	Encoder Sub Mask (maximum value of the absolute range of the encoder actual value in increments). Used, for example, as a reference mark for the referencing mode "Software Sync" and for the NC Retain Data "ABSOLUTE (MODULO)", "INCREMENTAL (SINGLETURN ABSOLUTE)". **Note 1:** The Encoder Sub Mask must be smaller than or equal to the Encoder Mask. **Note 2:** The Encoder Mask must be an integer multiple of the Encoder Sub Mask. **Note 3:** The Encoder Sub Mask must be a continuous sequence of binary ones (2^n-1), e.g. 0x000FFFFF.	s. parameter "Encoder Mask"
0x00n10110	Read / Write	E: INC (encoder simulation)	REAL64	1	[0.0 ... 1000000.0]	Scaling/weighting of the noise component for the simulation encoder	
CONTROLLER:							
0x00n20001	Read	Controller: every	UINT32	1	[1 ... 255]	Controller ID n = 0: Standard controller for the axes > 0: n-th controller of the axis (optional)	
0x00n20002	Read	Controller: every	UINT8[30+1]	1	30 characters	controller name	
0x00n20003	Read	Controller: every	UINT32	1	s. ENUM (>0)	controller type [▸ 492]	
0x00n2000A	Read / Write	Controller: every		1	s. ENUM (>0)	controller mode	

Index-Offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Remarks
0x00n2000B	Read / Write	Controller: every	REAL64	%	[0.0 ... 1.0]	Weighting of the velocity pre-control (default value: 1.0 = 100 %)	
0x00n20010	Read / Write	Controller: every	UINT16	1	0/1	following error monitoring position?	
0x00n20011	Read / Write	Controller: every	UINT16	1	0/1	following error monitoring velocity?	
0x00n20012	Read / Write	Controller: every	REAL64	e.g. mm		max. following error position	
0x00n20013	Read / Write	Controller: every	REAL64	s		max. following error time position	
0x00n20014	Read / Write	Controller: every	REAL64	e.g. mm/ s		max. following error velocity	
0x00n20015	Read / Write	Controller: every	REAL64	s		max. following error time velocity	
0x00n20100	Read / Write	P/PID (pos., (veloc.)	REAL64	1	[0.0...1.0]	Maximum output limitation (±) for controller total output	(default value: 0.5 == 50%)
0x00n20102	Read / Write	P/PID (pos.)	REAL64	e.g. mm/ s/ mm	[0.0...1000.0]	Proportional gain kp or kv respectively Unit: Base Unit / s / Base Unit	position control
0x00n20103	Read / Write	PID (pos.)	REAL64	s	[0.0 ... 60.0]	Integral action time Tn	position control
0x00n20104	Read / Write	PID (pos.)	REAL64	s	[0.0 ... 60.0]	Derivative action time Tv	position control
0x00n20105	Read / Write	PID (pos.)	REAL64	s	[0.0 ... 60.0]	Damping time Td	position control
0x00n20106	Read / Write	PP (Pos.)	REAL64	e.g. mm/ s/ mm	[0.0...1000.0]	Additional proportional gain, kp or kv respectively, that applies above a limit velocity in percent. Unit: Base Unit / s / Base Unit	position control
0x00n20107	Read / Write	PP (Pos.)	REAL64	%	[0.0...1.0]	Threshold velocity in percent above which the additional proportional gain, kp or kv respectively, applies	
0x00n20108	Read / Write	P/PID (Acc.)	REAL64	s	[0.0 ... 100.0]	Proportional gain ka	acceleration pre-control
0x00n2010D	Read / Write	P/PID	REAL64	mm	[0.0 ... 10000.0]	"dead band" for position error (control deviation) (for P/PID controllers with velocity or torque interface)	reserved function
0x00n2010F	Read / Write	P/PP/PID (Pos.) slave control	REAL64	(mm/s) / mm	[0.0...1000.0]	Slave coupling difference control: proportional gain k_{cp}	Slave coupling difference control:
0x00n20110	Read / Write	P (Pos.)	UINT16	1	0/1	Automatic offset calibration: active/passive	
0x00n20111	Read / Write	P (Pos.)	UINT16	1	0/1	Automatic offset calibration: hold mode	
0x00n20112	Read / Write	P (Pos.)	UINT16	1	0/1	Automatic offset calibration: fading mode	
0x00n20114	Read / Write	P (Pos.)	REAL64	%	[0.0 ... 1.0]	Automatic offset calibration: pre-control limit	
0x00n20115	Read / Write	P (Pos.)	REAL64	s	[0.1 ... 60.0]	Automatic offset calibration: time constant	
0x00n20116	Read / Write	PID (pos.)	REAL64	%	[0.0...1.0]	Maximum output limitation (±) for I-part in percent (default setting: 0.1 = 10%)	

Index-Offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Remarks
0x00n20117	Read / Write	PID (pos.)	REAL64	%	[0.0...1.0]	Maximum output limitation (±) for D-part in percent (default setting: 0.1 = 10%)	
0x00n20118	Read / Write	PID (pos.)	UINT16	1	0/1	Deactivation of the I-component during an active positioning process (assuming I-component active)? (default setting: 0 = FALSE)	
0x00n20120	Read / Write	P/PID (pos.)	REAL64	s	≥0	PT-1 filter value for position error (pos. control difference)	reserved function, no standard!
0x00n20202	Read / Write	P/PID (velocity)	REAL64	1	[0.0...1000.0]	Proportional gain kp or kv respectively	Velocity control
0x00n20203	Read / Write	PID (velocity)	REAL64	s	[0.0 ... 60.0]	Integral action time Tn	Velocity control
0x00n20204	Read / Write	PID (velocity)	REAL64	s	[0.0 ... 60.0]	Derivative action time Tv	Velocity control
0x00n20205	Read / Write	PID (velocity)	REAL64	s	[0.0 ... 60.0]	Damping time Td	Velocity control
0x00n20206	Read / Write	PID (velocity)	REAL64	%	[0.0...1.0]	Maximum output limitation (±) for I-part in percent (default setting: 0.1 = 10%)	Velocity control
0x00n20207	Read / Write	PID (velocity)	REAL64	%	[0.0...1.0]	Maximum output limitation (±) for D-part in percent (default setting: 0.1 = 10%)	Velocity control
0x00n2020D	Read / Write	P/PID (velocity)	REAL64	mm/s	[0.0 ... 10000.0]	"dead band" for velocity error (control deviation) (for P/PID controllers with velocity or torque interface)	reserved function
0x00n20220	Read / Write	P/PID (velocity)	REAL64	s	≥0	PT-2 filter value for velocity error (vel. control difference)	Velocity control, not standard!
0x00n20221	Read / Write	P/PID (velocity)	REAL64	s	≥0	PT-1 filter value for velocity error (vel. control difference)	reserved function, no standard!
0x00n20250	Read / Write	P/PI (observer)	UINT32	1	s. ENUM (≥0)	Observer mode [▶ 492] for controller with torque interface 0: OFF (default) 1: LUENBERGER	
0x00n20251	Read / Write	P/PI (observer)	REAL64	Nm / A	>0	Motor: torque constant K_T	
0x00n20252	Read / Write	P/PI (observer)	REAL64	kg m^2	>0	Motor: moment of inertia J_M	
0x00n20253	Read / Write	P/PI (observer)	REAL64	Hz	[100.0 ... 2000.0] default: 500	bandwidth f_0	
0x00n20254	Read / Write	P/PI (observer)	REAL64	1	[0.0 ... 2.0] default: 1.0	correction factor k_c	
0x00n20255	Read / Write	P/PI (observer)	REAL64	s	[0.0 ... 0.01] default: 0.001	velocity filter (1st order): time constant T	
0x00n20A03	Read / Write	P/PID (MW)	REAL64	cm^2	[0.0 ... 1000000]	cylinder area A_A of the A side in cm^2	reserved parameters !
0x00n20A04	Read / Write	P/PID (MW)	REAL64	cm^2	[0.0 ... 1000000]	cylinder area A_B of the B side in cm^2	reserved parameters !
0x00n20A05	Read / Write	P/PID (MW)	REAL64	cm^3/s	[0.0 ... 1000000]	nominal volume flow Q_{nom} in cm^3/s	reserved parameters !

Index-Offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Remarks
0x00n20A06	Read / Write	P/PID (MW)	REAL64	bar	[0.0 ... 1000000]	Rated pressure or valve pressure drop, P_{nom} in bar	reserved parameters !
0x00n20A07	Read / Write	P/PID (MW)	UINT32	1	[1 ... 255]	Axis ID for the system pressure Po	reserved parameters !
DRIVE:							
0x00n30001	Read	Drive: every	UINT32	1	[1 ... 255]	Drive ID	
0x00n30002	Read	Drive: every	UINT8[30+1]	1	30 characters	Drive name	
0x00n30003	Read	Drive: every	UINT32	1	s. ENUM (>0)	Drive type [▸ 496]	
0x00n30004	Read / Write	Drive: every	UINT32	1	Byteoffset	Input address offset (IO-Input-Image)	change I/O address
0x00n30005	Read / Write	Drive: every	UINT32	1	Byteoffset	Output address offset (IO-Output-Image)	change I/O address
0x00n30006	Read / Write	Drive: every	UINT16	1	[0,1]	motor polarity	
0x00n3000A	Read / Write	Drive: every	UINT32	1	s. ENUM (>0)	drive mode	
0x00n3000B	Read / Write	Drive: every	REAL64	%	[-1.0 ... 1.0]	Minimum output limit (default setting: -1.0 = -100%)	
0x00n3000C	Read / Write	Drive: every	REAL64	%	[-1.0 ... 1.0]	Maximum output limit (default setting: 1.0 = 100%)	
0x00n3000D	Read	Drive: every	UINT32	INC		Maximum number of output increments (output mask)	
0x00n30010	Read / Write	Drive: every	UINT32	1		Internal Drive Control double word to determine the drive operation modes	reserved!
0x00n30011	Read / Write	every	UINT32	1	≥ 5	Internal Drive Reset Counter (time in NC cycles for enable and reset)	reserved!
0x00n30101	Read / Write	D: Servo	REAL64	e.g. mm/s	>0	Reference velocity at reference output (velocity pre-control)	
0x00n30102	Read / Write	D: Servo	REAL64	%	[0.0 ... 5.0]	Reference output in percent (default setting: 1.0 = 100%)	
0x00n30103	Read	D: Servo	REAL64	e.g. mm/s	>0	resulting velocity at 100% output	
0x00n30104	Read / Write	D: Servo	REAL64	e.g. mm/s	±∞	velocity offset (DAC offset) for drift calibration (offset calibration) of the axis	
0x00n30105	Read / Write	D: Servo (Sercos, Profi Drive, AX200x, CANopen)	REAL64	1	[0.0 ... 100000000.0]	velocity scaling (scaling factor to react to the weight in the drive)	For Sercos, Profi Drive, AX200x, CANopen
0x00n30106	Read / Write	D: Profi Drive DSC	UINT32	0.001 * 1/s	≥ 0	Profibus/Profi Drive DSC: position control gain Kpc	Only for Profi Drive DSC
0x00n30107	Read / Write	D: Profi Drive DSC	REAL64	1	≥ 0.0	Profibus/Profi Drive DSC: scaling for calculating 'XERR' (Default: 1.0)	Only for Profi Drive DSC
0x00n30109	Read / Write	D: Servo (Sercos, CANopen)	REAL64	1	[0.0 ... 100000000.0]	position scaling (scaling factor to react to the weight in the drive)	For Sercos, CANopen
0x00n3010A	Read / Write	D: Servo (Sercos, Profi Drive, AX200x, CANopen)	REAL64	1	[0.0 ... 100000000.0]	acceleration scaling (scaling factor to react to the weight in the drive)	For Sercos, Profi Drive, AX200x, CANopen

Index-Offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Remarks
0x00n30120	Read / Write	D: Servo/hydraulics/	UINT32	1	≥ 0	Table ID (0: no table)	Only for KL4xxx, M2400, Universal
0x00n30121	Read / Write	D: Servo/hydraulics	UINT32	1	≥ 0	Interpolation type 0: Linear 2: Spline	Only for KL4xxx, M2400, Universal
0x00n30122	Read / Write	Servo/hydraulics	REAL64	%	[-1.0 ... 1.0]	Output offset in percent, note: Acts according to the characteristic evaluation !	Only for KL4xxx, M2400, Universal
0x00n30151	Read / Write	D: Servo / nonlinear	REAL64	1	[0.0 ... 100.0]	Quadrant compensation factor (relationship between quadrant I and III)	
0x00n30152	Read / Write	D: Servo / nonlinear	REAL64	1	[0.01 ... 1.0]	Velocity reference point in percent (1.0 = 100 %)	
0x00n30153	Read / Write	D: Servo / nonlinear	REAL64	1	[0.01 ... 1.0]	output reference point in percent (1.0 = 100 %)	
0x00030301	Read / Write	D: Stepper motor	UINT8	1		Bit mask: cycle 1	
0x00030302	Read / Write	D: Stepper motor	UINT8	1		Bit mask: cycle 2	
0x00030303	Read / Write	D: Stepper motor	UINT8	1		Bit mask: cycle 3	
0x00030304	Read / Write	D: Stepper motor	UINT8	1		Bit mask: cycle 4	
0x00030305	Read / Write	D: Stepper motor	UINT8	1		Bit mask: cycle 5	
0x00030306	Read / Write	D: Stepper motor	UINT8	1		Bit mask: cycle 6	
0x00030307	Read / Write	D: Stepper motor	UINT8	1		Bit mask: cycle 7	
0x00030308	Read / Write	D: Stepper motor	UINT8	1		Bit mask: cycle 8	
0x00030310	Read / Write	D: Stepper motor	UINT8	1		Bit mask: holding current	

7.8.4.2 "Index offset" specification for Axis state (Index group 0x4100 + ID)

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n00000	Read	every (online structure for axis data)	{			AXIS ONLINE STRUCTURE (NC/ CNC)	Changed from TC3, not oscilloscopeable! (NCAXIS-STATE_ ON-LINESTRUCT)
			INT32	1		Error state	
			INT32			Reserved	
			REAL64	e.g. mm		Actual position	
			REAL64	e.g. degree		Modulo actual position	
			REAL64	e.g. mm		Set position	
			REAL64	e.g. degree		Modulo set position	
			REAL64	e.g. mm/ s		Optional: actual velocity	
			REAL64	e.g. mm/ s		Set velocity	
			UINT32	%	0...1000000	speed override (1000000 == 100%)	
			UINT32			Reserved	
			REAL64	e.g. mm		Following error position	
			REAL64	e.g. mm		PeakHold value for max. neg. position lag (pos.)	
			REAL64	e.g. mm		Peak hold value for max. pos. following error. (pos.)	
			REAL64	%		Controller output in percent	
			REAL64	%		Total output in percent	
			UINT32	1	≥ 0	Axis status double word	
			UINT32	1	≥ 0	Axis control double word	
			UINT32	1	≥ 0	Slave coupling state (state)	
			UINT32	1	0; 1,2,3...	Axis control loop index	
			REAL64	e.g. mm/ s^2		Actual acceleration	
			REAL64	e.g. mm/ s^2		Set acceleration	
			REAL64	e.g. mm/ s^3		Set jerk (new from TC3.1 B4013)	
			REAL64	e.g. 100% = 1000		Set torque or set force (reserved, not implemented)	
			REAL64	e.g. 100% = 1000		Actual torque or actual force (new from TC3.1 B4013)	
			}			256 byte	
0x00000001	Read	every	UINT32	1		error code axis state [▶ 340]	*Symbolic access: 'ErrState"*
0x00n00009	Read	every	UINT32	1	≥ 0	Set cycle counter (SAF-Timestamp)	
0x00n0000A	Read	every	REAL64	e.g. mm		Set position	*Symbolic access: 'SetPos"*
0x00n0000B	Read	every	REAL64	e.g. degree		Modulo set position	*Symbolic access: 'SetPos-Modulo"*
0x00n0000C	Read	every	INT32	1		Modulo set rotation	
0x00n0000D	Read	every	REAL64	1	[-1.0, 0.0, 1.0]	Set direction	
0x00n0000E	Read	every	REAL64	e.g. mm/ s		Set velocity	*Symbolic access: 'SetVelo"*

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n0000F	Read	every	REAL64	e.g. mm/ s^2		Set acceleration	Symbolic access: *'SetAcc"*
0x00n00010	Read	every	REAL64	e.g. mm/ s^3		Set jerk (time derivation of the set acceleration)	
0x00n00011	Read	every	REAL64	e.g. Nm or N, e.g. 100%=1 000		Set torque (rot. motor) or set force (linear motor)	reserved, not implemented.
0x00n00012	Read	every	REAL64	1		Set coupling factor (set gear ratio)	
0x00n00013	Read	every	REAL64	e.g. mm		Expected target position	
0x00n00014	Read	Servo	{			Remaining travel time and distance (SERVO):	Always to SAF Port 501!
			REAL64	s	≥ 0	Remaining travel time	
			REAL64	e.g. mm	≥ 0	Remaining distance	
			}				
0x00n00015	Read	every	UINT32	1	≥ 0	Set command number	
0x00n00016	Read	Servo	REAL64	s	≥ 0	Positioning time of the last motion command (Start → target position window)	
0x00n00017	Read	Servo	REAL64	%	[0.0…1.0] 1.0=100%	Set override value for velocity Note: initially only implemented for FIFO group	NEW from TC3.1 B4020
0x00000018	ReadWrite	Servo	**Write**			Read the "Stop information" (stop distance, stop time)	Only port 500!
			REAL64	e.g. mm/ s^2	≥ 0	Deceleration for axis stop	
			REAL64	e.g. mm/ s^3	≥ 0	Jerk for axis stop	
			Read				
			REAL64	e.g. mm	≥ 0	Stop distance	
			REAL64	s	≥ 0	Stop time	
0x00n0001A	Read	every	REAL64	e.g. mm		Uncorrected set position	
0x00n0001D	Read	every	REAL64	1	[-1.0, 0.0, 1.0]	Uncorrected set position	
0x00n0001E	Read	every	REAL64	e.g. mm/ s		Uncorrected set velocity	
0x00n0001F	Read	every	REAL64	e.g. mm/ s^2		Uncorrected set acceleration	
0x00000020	Read	every	UINT32	1	s. ENUM	Coupling state (state)	
0x00000021	Read	every	UINT32	1	≥ 0	Coupling table index	
0x00000022	Read	servo master / slave coupling type: LINEAR, (&SPECIAL)	{			reading the coupling parameters (SERVO):	
			REAL64	1	[±1000000.0]	Parameter 1: Linear: Gearing factor	
			REAL64	1	[±1000000.0]	Parameter 2: Linear: Reserve	
			REAL64	1	[±1000000.0]	Parameter 3: Linear: Reserve	
			REAL64	1	[±1000000.0]	Parameter 4: Linear: Reserve	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000023	Read	Servo master / slave coupling type: LINEAR, (&SPECIAL)	REAL64	1	[±1000000.0]	Reading of the gear ratio (servo) type: LINEAR,	
0x00000024	Read	Servo	UINT32	1	≥ 0	Number / index of the active axis control circuit (triple of encoder, controller and axis interface)	
0x00000025	Read	Servo	UINT16	1	0/1	External set value specification via axis interface PCLtoNC active?	
0x00000026	Read	servo master / slave coupling type: SYNCHRONIZING	REAL64 [64]	1	±∞	Reading of the characteristic slave values synchronization profile type: SYNCHRONIZING	Modified from TC3
0x00000027	ReadWrite	Servo master / slave coupling type: TABULAR, MF	**Write**			Read the "Tabular coupling information"	Only port 500! Modified from TC3
			VOID or REAL64 or DWORD, DWORD, REAL64	e.g. mm	±∞	- no data for the "current information", - optional for a certain "master axis position" - for a certain table ID and optional “master axis position” (TC 3.1 B4017)	
			Read				
			REAL64 [32]		±∞	Read the tabular coupling information [▸ 499] structure	
0x00000028	ReadWrite	Servo master / slave coupling type: MULTI-CAM (CamAddition)	**Write**			Read the "multi tabular coupling information" (CamAddition)	Only port 500!
			UINT32	1	≥ 0	Table ID to which the query relates	
			Read				
			96 byte			Read the multi tabular coupling information [▸ 499] structure	
0x00000029	Read	Servo	UINT32	1		Delayed error code (error pre-warning) in case of a delayed error reaction (see bit *ErrorPropagationDelayed*)	
0x0000002A	Read	Servo	REAL64	e.g. mm	±∞	Position difference while fading from set position to actual position (fading part)	
0x0000002B	Read	Servo	REAL64	e.g. mm/s	±∞	Relative velocity while fading from set position to actual position (fading part)	
0x0000002C	Read	Servo	REAL64	e.g. mm/s ^2	±∞	Relative acceleration while fading from set position to actual position (fading part)	
0x0000002D	Read	Servo	UINT32	1	≥ 0	Counter for initialization command (InitializeCommandCounter)	NEW
0x0000002E	Read	Servo	UINT32	1	≥ 0	Counter for reset command (ResetCommandCounter)	NEW

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000040	Read	Servo	UINT32	1	≥ 0	Counter for correction of the NC set values in case of data inconsistency (Activation with Idx-Group 0x1000 and Idx-Offset 0x0020)	NEW from TC3.1 B4020
0x00000050	Read	every	UINT32	1		Set drive phase (SWGenerator)	Cannot be traced by oscilloscope!
0x00000051	Read	every	UINT16	1		Is the axis deactivated?	Cannot be traced by oscilloscope!
0x00n00060	Read / Write	every (online set value structure)	{			AXIS SET VALUE STRUCTURE (NC/CNC)	Cannot be traced by oscilloscope!
			REAL64	e.g. mm		Set position	
			REAL64	e.g. mm/s		Set velocity	
			REAL64	e.g. mm/s^2		Set acceleration / deceleration	
			REAL64	1	[-1.0, 0.0, 1.0]	Set travel direction	
			REAL64	e.g. mm/s^3		Set jerk	
			REAL64	Nm resp. N		Set torque or set force	
			}				
0x00n00061	Read / Write	every (online dynamics set value structure)	{			AXIS DYNAMIC SET VALUE STRUCTURE (NC/CNC)	
			REAL64	e.g. mm/s		Set velocity	
			REAL64	e.g. mm/s^2		Set acceleration / deceleration	
			REAL64	1	[-1.0, 0.0, 1.0]	Set travel direction	
			REAL64	e.g. mm/s^3		Set jerk	
			REAL64	Nm resp. N		Set torque or set force	
			}				
0x00000063	ReadWrite	for SERCOS/SoE only	**Write**			Read active "Drive Operation Mode"	NEW from TC 3.1 B4020 (NC 4249) Always to SAF Port 501!
			UINT32	1		Reserve	
			UINT32	1		Reserve	
			Read				
			UINT32	ENUM [▶ 497] (see appendix)	[0; 101, 102, …, 107]	currently active "Drive Operation Mode" (generic modes)	
			UINT32	1		Reserve	
0x00n10002	Read	every (Encoder)	REAL64	e.g. mm		Actual position (offset by actual position correction value) n = 0: standard encoder of the axes > 0: n^{th} encoder of the axis (optional)	*Symbolic access: 'ActPos"*
0x00n10003	Read	every (Encoder)	REAL64	e.g. degree		Modulo actual position	*Symbolic access: 'ActPos-Modulo'*
0x00n10004	Read	every (Encoder)	INT32	1		Modulo actual rotation	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n10005	Read	every (Encoder)	REAL64	e.g. mm/s		Optional: actual velocity	*Symbolic access: 'ActVelo"*
0x00n10006	Read	every (Encoder)	REAL64	e.g. mm/s^2		Optional: Actual acceleration	*Symbolic access: 'ActAcc"*
0x00n10007	Read	every (Encoder)	INT32	INC		Encoder actual increments	
0x00n10008	Read	every (Encoder)	INT64	INC		Software - actual increment counter	
0x00n10009	Read	every (Encoder)	UINT16	1	0/1	Reference flag ("calibrate flag")	
0x00n1000A	Read	every (Encoder)	REAL64	e.g. mm		Actual position correction value (measurement system error correction)	
0x00n1000B	Read	every (Encoder)	REAL64	e.g. mm		Actual position without actual position compensation value	Cannot be traced by oscilloscope!
0x00n10010	Read	every (Encoder)	REAL64	e.g. mm/s		Actual velocity without actual position compensation value	
0x00n10012	Read	every (Encoder)	REAL64	e.g. mm		Unfiltered actual position (charge with actual position compensation value)	
0x00n10014	Read	Encoder: SoE, CoE, MDP 742	REAL64	e.g. mm/s		Optional: Actual drive velocity (transferred directly from SoE, CoE or MDP 742 drive)	NEW from TC3.1 B4020.30
0x00n10015	Read	every (Encoder)	REAL64	e.g. mm/s		Optional: Unfiltered actual velocity	
0x00n10101	Read	INC (Encoder)	REAL64	e.g. mm		Read back of the position difference between activation of the internal hardware latch and the time when it becomes valid	Cannot be traced by oscilloscope!
0x00n20001	Read	R: every	INT32	1		Error state of the controller n = 0: standard controller of the axes > 0: n^{th} controller of the axis (optional)	
0x00n20002	Read	R: every	REAL64	e.g. mm/s		Controller output in absolute units	*Symbolic access: 'CtrlOutput"*
0x00n20003	Read	R: every	REAL64	%		Controller output in percent	Cannot be traced by oscilloscope!
0x00n20004	Read	R: every	REAL64	V		Controller output in volts	Cannot be traced by oscilloscope!
0x00n2000D	Read	R: every	REAL64	e.g. mm		Following error position (without dead time compensation)	Base Unit
0x00n2000F	Read	R: every	REAL64	e.g. mm		Following error position (with dead time compensation)	*Symbolic access: 'PosDiff"*
0x00n20010	Read	R: every	REAL64	e.g. mm		Peak hold value for maximum negative following error of the position	
0x00n20011	Read	R: every	REAL64	e.g. mm		Peak hold value for minimum positive following error of the position	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n20012	Read	R: every	REAL64	e.g. mm/ s		Following error velocity	Not implemented !
0x00n20021	Read	R: every	REAL64	e.g. mm		Difference (deviation) between the following error from master and slave axis (master error minus slave error)	*Symbolic access: 'PosDiff-Couple'*
0x00n20022	Read	R: every	REAL64	e.g. mm		PeakHold value for the maximum negative difference between master and slave axis following error of the position	Base Unit
0x00n20023	Read	R: every	REAL64	e.g. mm		PeakHold value for the maximum positive difference between master and slave axis following error of the position	Base Unit
0x00n20101	Read	R: P/PID (pos.)	REAL64	e.g. mm/ s		Proportional component of the controller in absolute units	
0x00n20102	Read	R: PID (pos.)	REAL64	e.g. mm/ s		I-part of the controller in absolute units	
0x00n20103	Read	R: PID (pos.)	REAL64	e.g. mm/ s		D-part of the controller in absolute units	
0x00n20104	Read	R: PID (pos.)	UINT16	1	0/1	Limitation of the I-part active?	
0x00n20105	Read	R: PID (pos.)	UINT16	1	0/1	Limitation of the D-part active?	
0x00n20106	Read	R: PID (pos.)	UINT16	1	0/1	ARW measures of the I component active? ARW: Anti Reset Windup	Not implemented !
0x00n20110	Read	R: PID (pos.)	REAL64	e.g. mm/ s		Acceleration pre-control Yacc of the controller in absolute units, note: Function depends on controller type!	Acceleration pre-control
0x00n20111	Read	R: PP (Pos.)	REAL64	mm/s/ mm	≥0	Internal interpolated proportional gain kp or kv	PP controller
0x00n20201	Read	R: P,PID (velocity)	REAL64	e.g. mm/ s		Velocity part of the controller	
0x00n20202	Read	R: P,PID (velocity)	REAL64	%		Velocity part of the controller in percent	Cannot be traced by oscilloscope!
0x00n20203	Read	R: P,PID (velocity)	REAL64	V		Velocity part of the controller in volts	Cannot be traced by oscilloscope!
0x00n20201	Read	R: P/PID (velocity)	REAL64	e.g. mm/ s		Proportional component of the controller in absolute units	
0x00n20202	Read	R: P/ PID (veloc.)	REAL64	e.g. mm/ s		I-part of the controller in absolute units	
0x00n20203	Read	R: P/ PID (veloc.)	REAL64	e.g. mm/ s		D-part of the controller in absolute units	
0x00n20204	Read	R: P/ PID (veloc.)	UINT16	1	0/1	Limitation of the I-part active?	
0x00n20205	Read	R: P/ PID (veloc.)	UINT16	1	0/1	Limitation of the D-part active?	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n20206	Read	R: P/ PID (veloc.)	UINT16	1	0/1	ARW measures for the I-part active? (ARW: Anti Reset Windup)	
0x00n2020A	Read	R: P/ PID (veloc.)	REAL64	e.g. mm/s		Total input size of the velocity controller	
0x00n20A00	Read	R: PID (MW)	REAL64	%	[-1.0...1.0]	Offsetting of the set velocity (pre-control)	reserved parameters !
0x00n20A01	Read	R: PID (MW)	REAL64	e.g. mm/s		Proportional component of the controller in absolute units or percent (according to output weight)	reserved parameters !
0x00n20A02	Read	R: PID (MW)	REAL64	e.g. mm/s		I-part of the controller in absolute units or percent (according to output weight)	reserved parameters !
0x00n20A03	Read	R: PID (MW)	REAL64	e.g. mm/s		D-part of the controller in absolute units or percent (according to output weight)	reserved parameters !
0x00n20A04	Read	R: PID (MW)	UINT16	1	0/1	Limitation of the I-part active?	reserved parameters !
0x00n20A05	Read	R: PID (MW)	UINT16	1	0/1	Limitation of the D-part active?	reserved parameters !
0x00n20A06	Read	R: PID (MW)	UINT16	1	0/1	ARW measures of the I component active? ARW: Anti Reset Windup	reserved parameters !
0x00n20A10	Read	R: PID (MW)	REAL64	e.g. mm/s		Acceleration pre-control Yacc of the controller in absolute units	reserved parameters !
0x00n30001	Read	D: every	INT32	1		Error state of the drive	
0x00n30002	Read	D: every	REAL64	e.g. mm/s		Total output in absolute units	*Symbolic access: 'DriveOutput'*
0x00n30003	Read	D: every	REAL64	%		Total output in percent	
0x00n30004	Read	D: every	REAL64	V		Total output in volts	Cannot be traced by oscilloscope!
0x00n30005	Read	D: every	REAL64	e.g. mm/s		PeakHold value for maximum negative total output	
0x00n30006	Read	D: every	REAL64	e.g. mm/s		PeakHold value for maximum positive total output	
0x00n30007	Read	D: every	REAL64	e.g. 100%=1000, e.g. Nm or N		Actual torque or actual force (typically 100%=1000)	NEW
0x00n30013	Read	D: every	REAL64	%		Total output in percent (based on non-linear characteristic curve!)	
0x00n30014	Read	D: every	REAL64	V		Total output in volt (based on non-linear characteristic curve!)	Cannot be traced by oscilloscope!
0x00n3011A	Read	D: Servo (Sercos, CANopen)	REAL64	e. g. mm		Optional Smoothingfilter: Filtered set position	NEW For Sercos, CANopen
0x00n3011E	Read	D: Servo (Sercos, CANopen)	REAL64	e. g. mm/s		Optional Smoothingfilter: Filtered set velocity	NEW For Sercos, CANopen

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n3011F	Read	D: Servo (Sercos, CANopen)	REAL64	e. g. mm/s^2		Optional Smoothingfilter: Filtered set acceleration / deceleration	NEW For Sercos, CANopen

7.8.4.3 "Index offset" specification for Axis functions (Index group 0x4200 + ID)

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000001	Write	every	VOID			Reset axis	For FIFO axes too!
0x00000002	Write	every	VOID			Stop axis	For FIFO axes too!
0x00000003	Write	every	VOID			Clear axis (task)	For FIFO axes too!
0x00000004	Write	every	{			Emergency stop (with controlled ramp)	Only for PTP axes!
			REAL64	e.g. mm/ s^2	> 0.0	Deceleration (must be greater than or equal to the original deceler-ation)	
			REAL64	e.g. mm/ s^3	> 0.0	Jerk (must greater than or equal to the original jerk)	
			}				
0x00000005	Write	PTP axis	{			Parameterizable stop (with controlled ramp)	For PTP axes only! Reserved function, no standard!
			REAL64	e.g. mm/ s^2	> 0.0	deceleration	
			REAL64	e.g. mm/ s^3	> 0.0	Jerk	
			}				
0x00000009	Write	PTP axis	{			Oriented stop (oriented end position)	Only for PTP axes!
			REAL64	e.g. de-gree	≥ 0.0	Modulo end position (modulo target posi-tion)	
			REAL64	e.g. mm/ s^2	> 0.0	Deceleration (currently not active)	
			REAL64	e.g. mm/ s^3	> 0.0	Jerk (currently not ac-tive)	
			}				
0x00000010	Write	every	VOID			Reference axis ("cali-bration")	
0x00000011	Write	every	{			New end position axis	Modified from TC3
			UINT32	ENUM	s. appendix	End position type [▶ 490] (see appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	new end position (tar-get position)	
			}				
0x00000012	Write	every	{			New end position and new velocity axis	
			UINT32	ENUM	s. appendix	command type [▶ 490] (s. appendix)	
			UINT32	ENUM	s. appendix	End position type [▶ 490] (s. appendix)	
			REAL64	e.g. mm	±∞	new end position (tar-get position)	
			REAL64	e.g. mm/ s	≥ 0.0	New end velocity (re-quested travelling speed)	
			REAL64	e.g. mm	±∞	Optional: Switching position from which the new travel profile is activated	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000015	Write	every	{			New dynamic parameters for active positioning	
			REAL64	e.g. mm/ s^2	> 0.0	Acceleration	
			REAL64	e.g. mm/ s^2	> 0.0	deceleration	
			REAL64	e.g. mm/ s^3	> 0.0	Optional: jerk (not yet implemented)	
			}				
0x00000016	**ReadWrite**	every SERVO	**Write** (80 bytes)			Universal axis start (UAS): merge of single commands, such as axis start, and online changes in combination with "Buffer Mode" (see TcMc2.lib)	Always to SAF Port 501! Modified from TC3
			{				
			UINT32	ENUM	s. appendix	Start type [▶ 489] (s. appendix)	
			UINT32	1	≥ 0	Bit mask for checks and operation modes (default value: 0)	
			REAL64	e.g. mm	±∞	End position (target position)	
			REAL64	e.g. mm/ s	≥ 0.0	Required velocity *Vrequ*	
			REAL64	e.g. mm/ s^2	≥ 0.0	Optional: Acceleration	
			REAL64	e.g. mm/ s^2	≥ 0.0	Optional: deceleration	
			REAL64	e.g. mm/ s^3	≥ 0.0	Optional: Jerk	
			UINT32	ENUM	s. appendix	Buffer Mode [▶ 489] (command buffer)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	Optional: blending position (command blending position)	
			REAL64	e.g. mm/ s	≥ 0.0	Optional: segment start velocity *Vi* *(0 ≤ Vi ≤ Vrequ)*	
			REAL64	e.g. mm/ s	≥ 0.0	Optional: segment end velocity *Vf* *(0 ≤ Vf ≤ Vrequ)*	
			}				
			Read				
			{				
			UINT16	1	≥ 0	Command number (job number)	
			UINT16	1	≥ 0	Command status	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000017	**ReadWrite**	SERVO	**Write** (80 bytes)			"Master/slave decoupling" and "Universal axis start (UAS)": Merge of decoupling command of a slave axis (IdxOffset: 0x00000041) and the subsequent universal axis start (UAS) (IdxOffset: 0x00000016)	not yet released!
			{				
			UINT32	ENUM	s. appendix	Start type [▸ 489] (s. appendix)	
			UINT32	1	≥ 0	Bit mask for checks and operation modes (default value: 0)	
			REAL64	e.g. mm	±∞	End position (target position)	
			REAL64	e.g. mm/s	≥ 0.0	Required velocity *Vrequ*	
			REAL64	e.g. mm/s^2	≥ 0.0	Acceleration	
			REAL64	e.g. mm/s^2	≥ 0.0	deceleration	
			REAL64	e.g. mm/s^3	≥ 0.0	Jerk	
			UINT32	ENUM	s. appendix	Buffer Mode [▸ 489] (command buffer)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	Optional: blending position (command blending position)	
			REAL64	e.g. mm/s	≥ 0.0	Optional: segment start velocity *Vi* *(0 ≤ Vi ≤ Vrequ)*	
			REAL64	e.g. mm/s	≥ 0.0	Optional: segment end velocity *Vf* *(0 ≤ Vf ≤ Vrequ)*	
			}				
			Read				
			{				
			UINT16	1	≥ 0	Command number (job number)	
			UINT16	1	≥ 0	Command status	
			}				
0x00000018	Write	every	VOID			Release axis lock for motion commands (TcMc2)	
0x00000019	Write	every	UINT32	1	> 0	Set external axis error (runtime error)	**Caution when using!**
0x00n0001A	Write	every	{			Set actual axis position	**Caution when using!** Also for FIFO axes! Always at SAF port 501!
			UINT32	ENUM	s. appendix	Actual position type [▸ 490] (s. appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	Actual position for axes = 0: Standard encoder of the axis n > 0: n-th encoder for the axis (optional)	Modified from TC3
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n0001B	Write	every	UINT32	1	0/1	Set reference flag ("calibrate flag")n = 0: standard encoder of the axis n > 0: n-th encoder for the axis (optional)	**Caution when using!** Also for FIFO axes!
0x00n0001C	Write	SERVO	{			Set only actual axis position without manipulating the set position (also for slave and with active process)	**Caution when using!**
			UINT32	ENUM	s. appendix	Actual position type [▸ 490] (s. appendix)	
			REAL64	e.g. mm	±∞	Actual position for axis n = 0: standard encoder of the axes > 0: n-th axis encoder (optional) **Caution when using!!!**	
			}				
0x00n0001D	Write	every	{			Set actual value of the axis on the drive side (Position interface and encoder offset of null required!) n = 0: Standard encoder of the axis n > 0: n-th encoder for the axis (optional)	**Caution when using !** Only for CANopen
			UINT32	ENUM	s. appendix	Actual position type [▸ 490] (s. appendix)	
			REAL64	e.g. mm	±∞	actual position for axis	
			}				
0x00n0001E	Write	every	{			Set a new encoder scaling factor on the fly (also when axis is in motion)	**Caution when using!** Always to SAF Port 501!
			UINT16	ENUM	1	Encoder scaling factor type 1: Absolute 2: Relative	Modified from TC3
			UINT16			ControlWord	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/ INC	[1.0E-8 ... 100.0]	New encoder scaling factor n = 0: Standard encoder of the axis n > 0: n-th encoder for the axis (optional)	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00n0001F	Write	every	{			Set actual axis position on the fly (also when axis is in motion)	**Caution when using!** Always on SAF port 501!
			UINT32	ENUM		Position type for setting actual value on the fly 1: Absolute 2: Relative	
			UINT32	1		Control double word, e.g. for "clearing the position lag"	
			REAL64			Reserve	
			REAL64	e.g. mm	±∞	New actual axis position	
			UINT32			Reserve	
			UINT32			Reserve	
			}				
0x00000020	Write	every 1D start	{			Standard axis start:	Modified from TC3
			UINT32	ENUM	s. appendix	Start type [▸ 489] (s. appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	End position (target position)	
			REAL64	e.g. mm/s	≥0.0	required velocity	
			}				
0x00000021	Write	every 1D start	{			Extended axis start (SERVO):	Modified from TC3
			UINT32	ENUM	s. appendix	Start type [▸ 489] (s. appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	End position (target position)	
			REAL64	e.g. mm/s	≥ 0.0	required velocity	
			UINT32	0/1	0/1	Standard acceleration?	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s^2	≥ 0.0	Acceleration	
			UINT32	0/1	0/1	Standard deceleration?	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s^2	≥ 0.0	deceleration	
			UINT32	0/1	0/1	Standard jerk?	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s^3	≥ 0.0	Jerk	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000022	Write	SERVO(MW)	{			Special axis start (SERVO):	Reserved start function, no standard! Modified from TC3
			UINT32	ENUM	s. appendix	Start type [▸ 489] (s. appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	End position (target position)	
			REAL64	mm/s	≥ 0.0	required start velocity	
			REAL64	e.g. mm	±∞	Position for a new velocity level	
			REAL64	e.g. mm/s	≥ 0.0	new end velocity level	
			UINT32	0/1	0/1	Standard acceleration?	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s^2	≥ 0.0	Acceleration	
			UINT32	0/1	0/1	Standard deceleration?	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s^2	≥ 0.0	deceleration	
			UINT32	0/1	0/1	Standard jerk?	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s^3	≥ 0.0	Jerk	
			}				
0x00000023	Write	SERVO	{			Start external set value specification (setting by cyclic axis interface PLCtoNC)	Modified from TC3
			UINT32	ENUM	1: Absolute 2: Relative	start type [▸ 489]	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	New end position (target position) optional !	
			REAL64			Reserve (TC3)	
			}				
0x00000024	Write	SERVO	VOID			Stop/disable external set value specification (cyclic axis interface PLCtoNC)	
0x00000025	Write	SERVO	{			Start reversing operation for positioning (SERVO):	Modified from TC3
			UINT32	ENUM	1	Start type [▸ 489] (default: 1)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	end position 1 (target position)	
			REAL64	e.g. mm	±∞	end position 2 (target position)	
			REAL64	0/1	0/1	required velocity	
			REAL64	s	≥ 0.0	idle time	
			}				
0x00000026	Write	every	{			Start drive output:	Modified from TC3
			UINT32	ENUM	s. appendix	Output type [▸ 496] (s. appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. %	±∞	required output value (e.g. %)	
			}				
0x00000027	Write	every	VOID			Stop drive output	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000028	Write	every	{			Change the drive output:	
			UINT32	ENUM	s. appendix	Output type [▸ 496] (s. appendix)	
			REAL64	e.g. %	±∞	required output value (e.g. %)	
			}				
0x00000029	Write	every	VOID			Instantaneously adopt current override value and freeze until next override change!	reserved function, no standard!

0x0000002A	Write	every	{
			32 bytes
			}
						calculate and set encoder offset	reserved function, no standard!

0x0000002B	ReadWrite	every	WriteData: see 'UAS' ReadData: s. 'UAS'			stop external setpoint generator and continuous endless motion ('UAS': Universal axis start)	reserved function, no standard!
0x0000002C	Write	every	UINT32		≥ 0	Set "homing state" (for internal use)	New from TC3
0x00000030	Write	SERVO	{			Start section compensation (SERVO)	Modified from TC3
			UINT32	ENUM	s. appendix	Compensation type [▸ 490] (s. appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s^2	≥ 0.0	Max. acceleration increase	
			REAL64	e.g. mm/s^2	≥ 0.0	Max. deceleration increase	
			REAL64	e.g. mm/s	> 0.0	Max. increase velocity	
			REAL64	e.g. mm/s	> 0.0	Base velocity for the process	
			REAL64	e.g. mm	±∞	Path difference to be compensated	
			REAL64	e.g. mm	> 0.0	Path distance for compensation	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000030	ReadWrite	SERVO returns the actually implemented parameters as return values	{ **READ+WRITE:**			Start section compensation (SERVO) Note: contained only in "TcMc2.lib"	Modified from TC3
			UINT32	ENUM	s. appendix	Compensation type [▸ 490] (s. appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s^2	≥ 0.0	=> Max. acceleration increase <= returns the implemented acceleration increase (new in "TcMc2.lib")	
			REAL64	e.g. mm/s^2	≥ 0.0	=> Max. deceleration increase <= returns the implemented deceleration increase (new in "TcMc2.lib")	
			REAL64	e.g. mm/s	> 0.0	=> requested max. increase velocity <= returns the implemented velocity of increase	
			REAL64	e.g. mm/s	> 0.0	Base velocity for the process	
			REAL64	e.g. mm	±∞	=> requested path difference to be compensated <= returns the implemented path difference	
			REAL64	e.g. mm	> 0.0	=> requested max. distance for compensation <= returns implemented distance	
			UINT32	1	≥ 0	<= returns Warning ID (e.g. 0x4243)	
			UINT32			Reserve (TC3)	
			}				
0x00000031	Write	SERVO	VOID			Stop section compensation (SERVO)	
0x00000032	Write	SERVO	{			Start reversing operation with velocity steps (SERVO): (can be used to determine the velocity step response)	Modified from TC3
			UINT32	ENUM	1	Start type [▸ 489] (default: 1)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s	±∞	Required velocity 1 (negative values also allowed)	
			REAL64	e.g. mm/s	±∞	Required velocity 2 (negative values also allowed)	
			REAL64	s	> 0.0	Travel time for velocity 1 and 2	
			REAL64	s	≥ 0.0	idle time	
			UINT32	1	0, 1,2,3...	Optional: number of repetitions, Default "0": unlimited in time	
			UINT32			Reserve (TC3)	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000033	Write	SERVO	{			Sinus Oscillation Sequence - used as single sinus oscillation (sinus generator) - used as sinus oscillation sequence (e.g. for bode plot)	Modified from TC3
			UINT32	ENUM	1	Start type [▸ 489] (fixed to start type 1 yet)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s	> 0.0	base amplitude (e.g. 2.5 mm/s)	
			REAL64	Hz	[0.0 .... 10.0]	base frequency (e.g. 1.953125 Hz)	
			REAL64	e.g. mm/s	≥ 0.0	start amplitude at begin (e.g. 0.0 mm/s)	
			REAL64	e.g. mm/REV	> 0.0	feed constant motor (per motor turn) (e.g. 10.0 mm/REV)	
			REAL64	Hz	≥ 1.0	frequency range: start frequency (e.g. 20.0 Hz)	
			REAL64	Hz	≤ 1/(2*dT)	frequency range: stop frequency (e.g. 500.0 Hz)	
			REAL64	s	> 0.0	step duration (e.g. 2.048s)	
			UINT32	1	[1 ... 200]	number of measurements (step cycles) (e.g. 20)	
			UINT32	1		number of parallel measurements (e.g. 1) not used yet!	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000040 (0x00n00040)	Write	Master/Slave coupling: (SERVO)	{			Master/slave coupling (SERVO):	Extension for "flying saw"!
			UINT32	ENUM	s. appendix	Slave type/ [▶ 491]coupling type (see appendix)	Angle >0.0 and £ 90.0 degrees (parallel saw: 90.0 degrees)
			UINT32	1	[1...255]	Axis ID of the master axis/group	
			UINT32	1	[0...8]	Subindex n of the master axis (default: value: 0)	
			UINT32	1	[0...8]	Subindex n of the slave axis (default: value: 0)	
			REAL64	1	[±1000000.0]	Parameter 1: linear: Gearing factor FlySawVelo: Reserve FlySaw: abs. synchron position master [mm]	
			REAL64	1	[±1000000.0]	Parameter 2: linear: Reserve FlySawVelo: Reserve FlySawPos: Abs. synchronous position slave [mm]	
			REAL64	1	[±1000000.0]	Parameter 3: linear: Reserve FlySawVelo: Angle of inclination in [DEGREE] FlySawPos: angle of inclination in [DEGREE]	
			REAL64	1	[±1000000.0]	Parameter 4: linear: Reserve FlySawVelo: Gearing factor FlySawPos: Gearing factor	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000040 (0x00n00040)	Write	Master/Slave coupling: (SERVO)	{			Master/slave coupling (SERVO):	Multi-master coupling (MC_GearIn-MultiMaster) Version V1 and V2 Modified from TC3
			UINT32	ENUM	s. appendix	Slave type/ [▶ 491]coupling type (see appendix)	
			UINT32	1	[1...255]	Axis ID of the master axis/group	
			UINT32	1	[1...8]	Subindex n of the master axis (default: value: 0)	
			UINT32	1	[1...8]	Subindex n of the slave axis (default: value: 0)	
			UINT32	1	[0...255]	Axis ID master 2	
			UINT32	1	[0...255]	Axis ID master 3	
			UINT32	1	[0...255]	Axis ID master 4	
			UINT32	1	[0...255]	Reserve (axis ID master 5)	
			UINT32	1	[0...255]	Reserve (axis ID master 6)	
			UINT32	1	[0...255]	Reserve (axis ID master 7)	
			UINT32	1	[0...255]	Reserve (axis ID master 8)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm/s^2		Maximum acceleration/deceleration of the slave axis	
			UINT32	1	≥ 0	Control mask, not yet in use (check and operation mode for profile)	
			UINT32			Reserve (TC3)	
			Extension V2 (Optional):				
			REAL64	e.g. mm/s^2	≥ 0.0	Maximum deceleration of the slave axis	
			REAL64	e.g. mm/s^3	≥ 0.0	Maximum jerk of the slave axis	
			REAL64	e.g. mm/s	≥ 0.0	Maximum velocity of the slave axis	
			REAL64			Reserve	
			REAL64			Reserve	
			} 64 or 104 bytes				
0x00000041	Write	Master/slave decoupling (SERVO)	VOID			Master/slave decoupling (SERVO)	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000041	Write	Master/slave decoupling with configurable follow-up function (SERVO)	{			Master/slave decoupling with configurable follow-up function (e.g. new end position, new velocity, stop, E-stop) (SERVO)	not yet released! Modified from TC3
			UINT32	ENUM	s. appendix	Decoupling type [▸ 491] (see appendix)	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	Optional: new end position	
			REAL64	e.g. mm/s	> 0.0	Optional: new requested velocity	
			REAL64	e.g. mm/s^2	≥ 0.0 (0: Default)	Optional: acceleration for new end position, new velocity and emergency stop (E-stop)	
			REAL64	e.g. mm/s^2	≥ 0.0 (0: Default)	Optional: deceleration for new end position, new velocity and emergency stop (E-stop)	
			REAL64	e.g. mm/s^3	≥ 0.0 (0: Default)	Optional: jerk for new end position, new velocity and emergency stop (E-stop)	
			}				
0x00000042	Write	Master / slave coupling type: LINEAR (&SPECIAL)	{			Change of the coupling parameters (SERVO):	
			REAL64	1	[±1000000.0]	Parameter 1: Linear: Gearing factor	
			REAL64	1	[±1000000.0]	Parameter 2: Linear: Reserve	
			REAL64	1	[±1000000.0]	Parameter 3: Linear: Reserve	
			REAL64	1	[±1000000.0]	Parameter 4: Linear: Reserve	
			}				
0x00000043	Write	Master / slave table- coupling type: TABULAR	{			Change of the table coupling parameters (SERVO):	
			REAL64	mm	±∞	Slave position offset	
			REAL64	mm	±∞	Master position offset	
			}				
0x00000043	Write	Master / slave table- coupling type: TABULAR and "Motion Function"	{			Change of the table coupling parameters (SERVO):	Also for "Motion Function"
			REAL64	mm	±∞	Slave position offset	
			REAL64	mm	±∞	Master position offset	
			REAL64	1	±∞ (<> 0.0)	Slave position scaling	
			REAL64	1	±∞ (<> 0.0)	Master position scaling	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000043	Write	Master / slave table- coupling type: TABULAR	{			Change of the table coupling parameters (SERVO):	
			REAL64	mm	±∞	Slave position offset	
			REAL64	mm	±∞	Master position offset	
			REAL64	1	±∞ (<> 0.0)	Slave position scaling	
			REAL64	1	±∞ (<> 0.0)	Master position scaling	
			REAL64	e.g. mm	±∞	Absolute master activation position	
			}				
0x00000044	Write	Slave-Stop (SERVO)	VOID			Stop the "flying saw" (SERVO)	Only for "flying saw"
0x00000045 (0x00n00045)	Write	Master/slave table coupling (SERVO)	{			Master/slave table coupling (SERVO):	
			UINT32	ENUM	s. appendix	Slave type/coupling type [▶ 491] (see appendix)	
			UINT32	1	[1...255]	Axis ID of the master axis	
			UINT32	1	[0...8]	Subindex n of the master axis (default: value: 0)	
			UINT32	1	[0...8]	Subindex n of the slave axis (default: value: 0)	
						SOLO TABLE SECTION:	
			REAL64	mm	±∞	Slave position offset (type: TABULAR)	
			REAL64	mm	±∞	Master position offset (type: TABULAR)	
			UINT32	1	[0,1]	Slave positions absolute (type: TABULAR)	
			UINT32	1	[0,1]	Master positions absolute (type: TABULAR)	
			UINT32	1	[1...255]	Table ID of the coupling table (type: TABULAR)	
						MULTI-TABLE SECTION:	
			UINT16	1	[0...8]	Number of tables (type: MULTITAB) Note: misused as interpolation type for solo tables	
			UNIT16	1	[0...8]	Number of profile tables (type: MULTITAB)	
			UNIT32[8]	1	[1...255]	Tables IDs of the coupling tables (type: MULTITAB)	
			}				
0x00000046	Write	Master / slave multi-tables	UINT32	1	[1...255]	Correction table activation, correction table ID	
0x00000046	Write	Master / slave multi-tables	{			Activation of correction table	Modified from TC3
			UINT32	1	[1...255]	correction table ID	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	Absolute master activation position	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000047	Write	Master / slave multi-tables	UINT32	1	[1..255]	Deactivation of profile table at the end of the cycle, table ID of the current monocyclic profile table	
0x00000048	ReadWrite	Master / slave multi-tables	Write: UINT32	1	[1..255]	Read the last correction offset: Table ID of the correction table	
			Read: REAL32	e.g. mm	±∞	Offset by departing the correction table with the according table ID	
0x00000049	Write	Master / slave table- coupling type: TABULAR	REAL64	1	±∞	Change the slave table scaling factor for the slave table column (default value: 1.0)	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x0000004A(0x 00n0004A)	Write	Master / slave universal table coupling (SERVO)	{			Master / slave solo table coupling (SERVO):	Modified from TC3
			UINT32	ENUM	s. appendix	Slave type/coupling type [▶ 491] (see appendix)	
			UINT32	1	[1...255]	Axis ID of the master axis	
			UINT32	1	[0...8]	Subindex n of the master axis (default: value: 0)	
			UINT32	1	[0...8]	Subindex n of the slave axis (default: value: 0)	
			UINT32	1	1...255]	Table ID of the coupling table (type: TABULAR)	
			UINT32	1		Tabular interpolation type	
			REAL64	mm	±∞	Slave position offset (type: TABULAR)	
			REAL64	mm	±∞	Master position offset (type: TABULAR)	
			REAL64	mm	±∞	Slave position scaling (type: TABULAR)	
			REAL64	mm	±∞	Master position scaling (type: TABULAR)	
			UINT32	1	[0,1]	Slave position absolute ? (Type: TABULAR)	
			UINT32	1	[0,1]	Master position absolute ? (Type: TABULAR)	
			UINT32	ENUM	s. appendix	Activation type of the change: 0: 'instantaneous' (default) 1: 'at master cam position' 2: 'at master axis position' 3: 'next cycle'	
			UINT32			Reserve (TC3)	
			REAL64	mm	±∞	Activation position	
			UINT32	ENUM	s. appendix	Master scaling type: 0: user defined (default) 1: scaling with auto offset 2: off	
			UINT32	ENUM	s. appendix	Slave scaling type: 0: user defined (default) 1: scaling with auto offset 2: off	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x0000004A(0x 00n0004A)	Write	Master / slave universal table coupling (SERVO)	{			Master / slave solo table coupling (SERVO):	Modified from TC3
			UINT32	ENUM	s. appendix	Slave type/coupling type [▸ 491] (see appendix)	
			UINT32	1	[1...255]	Axis ID of the master axis	
			UINT32	1	[0...8]	Subindex n of the master axis (default: value: 0)	
			UINT32	1	[0...8]	Subindex n of the slave axis (default: value: 0)	
			UINT32	1	1...255]	Table ID of the coupling table (type: TABULAR)	
			UINT32	1		Tabular interpolation type	
			REAL64	mm	±∞	Slave position offset (type: TABULAR)	
			REAL64	mm	±∞	Master position offset (type: TABULAR)	
			REAL64	mm	±∞	Slave position scaling (type: TABULAR)	
			REAL64	mm	±∞	Master position scaling (type: TABULAR)	
			UINT32	1	[0,1]	Slave position absolute ? (Type: TABULAR)	
			UINT32	1	[0,1]	Master position absolute ? (Type: TABULAR)	
			UINT32	ENUM	s. appendix	Activation type of the change: 0: 'instantaneous' (default) 1: 'at master cam position' 2: 'at master axis position' 3: 'next cycle'	
			UINT32			Reserve (TC3)	
			REAL64	mm	±∞	Activation position	
			UINT32	ENUM	s. appendix	Master scaling type: 0: user defined (default) 1: scaling with auto offset 2: off	
			UINT32	ENUM	s. appendix	Slave scaling type: 0: user defined (default) 1: scaling with auto offset 2: off	
			Extension for MultiCam:				
			UINT32	ENUM	s. appendix	Cam Operation Mode	
			UINT32	1	[1…255]	Reference table ID	
			BYTE[104]			Reserve (TC3)	
			}				

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x0000004B(0x 00n0004B)	Write	Master / slave universal flying saw (SERVO)	{			Master / slave synchronizing coupling (SERVO):	Modified from TC3
			UINT32	ENUM	s. appendix	Slave type/coupling type (see appendix)	
			UINT32	1	[1...255]	Axis ID of the master axis	
			UINT32	1	[0...8]	Subindex n of the master axis (default: value: 0)	
			UINT32	1	[0...8]	Subindex n of the slave axis (default: value: 0)	
			REAL64	1	±∞ (<> 0.0)	Gearing factor	
			REAL64	mm	±∞	Master synchron position	
			REAL64	mm	±∞	Slave synchron position	
			REAL64	mm/s	≥ 0.0	Slave velocity (optional)	
			REAL64	mm/s^2	≥ 0.0	Slave acceleration (optional)	
			REAL64	mm/s^2	≥ 0.0	Slave deceleration (optional)	
			REAL64	mm/s^3	≥ 0.0	Slave jerk (optional)	
			UINT32	1	≥ 0	Bit mask (default value: 0)	
			UINT32			Reserve (TC3)	
			}				
0x0000004D(0x 00n0004D)	Write	Master / slave table- coupling type: TABULAR and MF	{			Change of the table scaling (SERVO):	Modified from TC3
			UINT32	ENUM	s. appendix	Activation type of the change0: 'instantaneous' (default) 1: at master cam position' 2: 'at master axis position' 3: 'next cycle'	
			UINT32			Reserve (TC3)	
			REAL64	e.g. mm	±∞	Activation position	
			UINT32	ENUM	s. appendix	Master scaling type 0: user defined (default) 1: scaling with auto offset 2: off	
			UINT32	ENUM	s. appendix	Slave scaling type 0: user defined (default) 1: scaling with auto offset 2: off	
			REAL64	e.g. mm	±∞	Master position offset	
			REAL64	e.g. mm	±∞	Slave position offset	
			REAL64	1	±∞ (<> 0.0)	Master position scaling	
			REAL64	1	±∞	Slave position scaling	
			Optional extension for MultiCam:				
			UINT32	1	≥ 0	Cam Table ID	
			UINT32			Reserve (TC3)	
			}				
0x00000050	Write	every	VOID			Deactivate complete axis (disable)	

Index offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Note
0x00000051	Write	every	VOID			Activate complete axis (enable)	
0x00000052	Write	SERVO	{			Change of the active axis control loop (triple from encoder, controller and axis interfaces) with/without external set value specification:	Modified from TC3
			UINT32	1	≥ 0	Number/index of the axis control loop (default -value: 0)	
			UINT32	ENUM	see Annex (>0)	Switching type for synchronization behavior [▸ 500] 1: 'Standard'	
			REAL64	1	±∞	Synchronization value for switching (optional)	
			UINT32	0/ 1	0/1	External set value specification by means of axis interface ? Note: not used so far!	
			UINT32			Reserve (TC3)	
			}				
0x00000060	Write	every	VOID			Deactivate drive output (disable)	
0x00000061	Write	every	VOID			Activate drive output (enable)	
0x00000062	Write	high/low	UINT16	1	0/1	Release parking brake? 0: automatic activation (default) 1: must always be released! Note: reset to '0' when resetting the axis!	
0x00000063	Write	for SERCOS/ SoE only	{			Activate "Drive Operation Mode" (e.g. Position Velo, Torque, etc.)	NEW from TC 3.1 B4020 (NC 4249)
			UINT32	ENUM [▸ 497] (see appendix)	[0; 101, 102, …, 107]	new "Drive Operation Mode" (generic modes)	Always to SAF Port 501!
			UINT32	1	0	Reserve	
			UINT32	1	0	Reserve	
			UINT32	1	0	Reserve	
			}				
0x00000070	Write	every	VOID			Return of the axis from, e.g. a 3D group to its own PTP group	

7.8.4.4 "Index offset" specification for cyclic Axis process data (Index group 0x4300 + ID)

Index-Offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Remarks
0x00n00000	Read / Write	every (PLC→NC)	{ 128 bytes}		STRUCT see axis interface	AXIS STRUCTURE (PLC→NC) n = 0: standard axis interface n > 0: n-th axis interface (optional)	Write command only optional! Be aware of safety aspects! *PLCTONC_AXIS_REF*
0x00n00001	Read / Write	every (PLC→NC)	UINT32	1	>0	Control double word	Write command only optional!
0x00n00002	Read / Write	every (PLC→NC)	UINT16	1	0/1	Enable controller	Cannot be traced by oscilloscope!
0x00n00003	Read / Write	every (PLC→NC)	UINT16	1	0/1	Feed enable plus	Cannot be traced by oscilloscope!
0x00n00004	Read / Write	every (PLC→NC)	UINT16	1	0/1	Feed enable minus	Cannot be traced by oscilloscope!
0x00n00007	Read / Write	every (PLC→NC)	UINT16	1	0/1	Referencing cam	Cannot be traced by oscilloscope!
0x00n00021	Read / Write	every (PLC→NC)	UINT32	%	0...1000000	speed override (1000000 == 100%)	Write command only optional!
0x00n00022	Read / Write	every (PLC→NC)	UINT32	1	ENUM	operation mode axis	Write command only optional!
0x00n00025	Read / Write	every (PLC→NC)	REAL64	e.g. mm		Actual position correction value (measurement system error correction)	Write command only optional!
0x00n00026	Read / Write	every (PLC→NC)	REAL64	e.g. mm/s		External controller component (position controller component)	Write command only optional!
0x00n00027	Read / Write	every (PLC→NC)	{			External setpoint generation	Write command only optional!
			REAL64	e.g. mm	±∞	External set position	
			REAL64	e.g. mm/s	±∞	External set velocity	Modified from TC3
			REAL64	e.g. mm/s^2	±∞	External set acceleration	
			INT32	1	+1, 0, -1	External set travel direction	
			UINT32			Reserve (TC3)	
			REAL64			Reserve (TC3)	
			}				
0x00n00080	Read	every (PLC→NC)	{ 256 bytes}		STRUCT see axis interface	AXIS STRUCTURE (NC→PLC) Note: Size and alignment changed, n = 0: standard axis interface n > 0: n-th axis interface (optional)	Changed from TC3.*NCTOPLC_AXIS_REF*
0x00n00071	Read	every (PLC→NC)	UINT8	1	>0	Status double word: Byte 1	
0x00n00072	Read	every (PLC→NC)	UINT8	1	>0	Status double word: Byte 2	
0x00n00073	Read	every (PLC→NC)	UINT8	1	>0	Status double word: Byte 3	
0x00n00074	Read	every (PLC→NC)	UINT8	1	>0	Status double word: Byte 4	
0x00n00081	Read	every (PLC→NC)	UINT32	1	>0	Status double word (complete)	
0x00n00082	Read	every (PLC→NC)	UINT16	1	0/1	Axis is ready for operation	Cannot be traced by oscilloscope!

Index-Offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Remarks
0x00n00083	Read	every (PLC→NC)	UINT16	1	0/1	Axis has been referenced	Cannot be traced by oscilloscope!
0x00n00084	Read	every (PLC→NC)	UINT16	1	0/1	Axis in protected operation mode (e.g. slave axis)	Cannot be traced by oscilloscope!
0x00n00085	Read	every (PLC→NC)	UINT16	1	0/1	Axis is in rapid mode	Cannot be traced by oscilloscope!
0x00n00088	Read	every (PLC→NC)	UINT16	1	0/1	Axis has invalid IO data	Cannot be traced by oscilloscope!
0x00n00089	Read	every (PLC→NC)	UINT16	1	0/1	Axis is in an error state	Cannot be traced by oscilloscope!
0x00n0008A	Read	every (PLC→NC)	UINT16	1	0/1	Axis moving to larger values	Cannot be traced by oscilloscope!
0x00n0008B	Read	every (PLC→NC)	UINT16	1	0/1	Axis moving to smaller values	Cannot be traced by oscilloscope!
0x00n0008C	Read	every (PLC→NC)	UINT16	1	0/1	Axis is at logical standstill (only set values are considered) (position controller?)	Cannot be traced by oscilloscope!
0x00n0008D	Read	every (PLC→NC)	UINT16	1	0/1	Axis is being referenced	Cannot be traced by oscilloscope!
0x00n0008E	Read	every (PLC→NC)	UINT16	1	0/1	Axis is in position window	Cannot be traced by oscilloscope!
0x00n0008F	Read	every (PLC→NC)	UINT16	1	0/1	Axis is at target position (target position reached)	Cannot be traced by oscilloscope!
0x00n00090	Read	every (PLC→NC)	UINT16	1	0/1	Axis has constant velocity or rotary speed	Cannot be traced by oscilloscope!
0x00n0009A	Read	every (PLC→NC)	UINT16	1	0/1	Operation mode not executed (busy)	Cannot be traced by oscilloscope!
0x00n0009B	Read	every (PLC→NC)	UINT16	1	0/1	Axis has instructions, is carrying instructions out	Cannot be traced by oscilloscope!
0x00n000B1	Read	every (PLC→NC)	UINT32	1	≥0	Axis error code	
0x00n000B2	Read	every (PLC→NC)	UINT32	1	ENUM	Motion state of the axis (master state [▸ 497] / slave state [▸ 498])	
0x00n000B3	Read	every (PLC→NC)	UINT32	1	ENUM	Operation mode of the axis (rev. NC)	
0x00n000B4	Read	every (PLC→NC)	UINT32	1	ENUM	Axis referencing status	
0x00n000B5	Read	every (PLC→NC)	UINT32	1	ENUM	Axis coupling state	
0x00n000B6	Read	every (PLC→NC)	UINT32	1	≥0	SVB entries/tasks of the axis (PRE table)	
0x00n000B7	Read	every (PLC→NC)	UINT32	1	≥0	SAF entries/tasks of the axis (EXE table)	
0x00n000B8	Read	every (PLC→NC)	UINT32	1	≥0	Axis ID	

Index-Offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Remarks
0x00n000B9	Read	every (PLC→NC)	UINT32	1	≥0	Operation mode status double word: bit 0: position range monitoring active? Bit 1: target position window monitoring active? Bit 2: Loop path active? Bit 3: physical movement monitoring active? Bit 4: PEH time monitoring active? Bit 5: backlash compensation active? Bit 6: delayed error reaction mode active? Bit 7: modulo operation mode active (modulo axis)? Bit 16: trailing separation monitoring position active? Bit 17: following error monitoring speed active? Bit 18: end position monitoring min. active? Bit 19: end position monitoring max. active? Bit 20: actual position correction active?	
0x00n000BA	Read	every (PLC→NC)	REAL64	e.g. mm		Actual position (calculated absolute value)	
0x00n000BB	Read	every (PLC→NC)	REAL64	e.g. mm		Modulo actual position	
0x00n000BC	Read	every (PLC→NC)	INT32	1		Modulo rotations	
0x00n000BD	Read	every (PLC→NC)	REAL64	e.g. mm/s		Actual velocity (optional)	
0x00n000BE	Read	every (PLC→NC)	REAL64	e.g. mm		Following error position	
0x00n000BF	Read	every (PLC→NC)	REAL64	e.g. mm		Set position	
0x00n000C0	Read	every (PLC→NC)	REAL64	e.g. mm/s		Set velocity	
0x00n000C1	Read	every (PLC→NC)	REAL64	e.g. mm/s^2		Set acceleration	
0x00n10000	Read / Write	Encoder: every (NC→IO)	{ 40 bytes }		STRUCT see encoder IO interface	ENCODER-OUTPUT-STRUCTURE (NC→IO, 40 bytes) *NCENCODER-STRUCT_OUT2*	Write command only optional! Consider safety aspects!
0x00n10080	Read	Encoder: every (IO→NC)	{ 40 bytes }		STRUCT see encoder IO interface	ENCODER-INPUT-STRUCTURE (IO→NC, 40 bytes)*NCENCODER-STRUCT_IN2*	
0x00n30000	Read / Write	Drive: every (NC→IO)	{ 40 bytes }		STRUCT see drive IO interface	DRIVE-OUTPUT-STRUCTURE (NC→IO, 40 bytes) *NC-DRIVESTRUCT_OUT2*	Write command only optional! Consider safety aspects!

Index-Offset (Hex)	Access	Axis type	Data type	Phys. unit	Definition range	Description	Remarks
0x00n30080	Read	Drive: every (IO→NC)	{ 40 bytes }		STRUCT see drive IO interface	DRIVE-INPUT-STRUCTURE (NC→IO, 40 bytes) *NC-DRIVESTRUCT_IN2*	

7.8.5 Specification Encoder

7.8.5.1 "Index offset" specification for Encoder parameter (Index group 0x5000 + ID)

Index offset (Hex)	Access	group- type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	UINT32	1	[1 ... 255]	Encoder ID	
0x00000002	Read	every	UINT8[30+1]	1	30 symbol	Encoder name	
0x00000003	Read	every	UINT32	1	s. ENUM (>0)	Encoder type [▸ 493]	
0x00000004	Read / Write	every	UINT32	1	Byteoffset	input address offset (IO-Input-Image)	change I/O address
0x00000005	Read / Write	every	UINT32	1	Byteoffset	output address offset (IO-Output-Image)	change I/O address
0x00000006	Read / Write	every	REAL64	e.g. mm/ INC	[1.0E-12 ... 1.0E+30]	resulting scaling factor (numerator / denominator) **Note:** From TC3 there exists two components for the encoder scaling factor, numerator and denominator (default: 1.0).	
0x00000007	Read / Write	every	REAL64	e.g. mm	[1.0E+9]	position offset	
0x00000008	Read / Write	every	UINT16	1	[0,1]	encoder count direction	
0x00000009	Read / Write	every	REAL64	e.g. mm	[0.001 ... 1.0E +9]	modulo factor	
0x0000000A	Read / Write	every	UINT32	1	s. ENUM (>0)	Encoder mode [▸ 494]	
0x0000000B	Read / Write	every	UINT16	1	0/1	soft end min. monitoring ?	
0x0000000C	Read / Write	every	UINT16	1	0/1	soft end max. monitoring ?	
0x0000000D	Read / Write	every	REAL64	mm		soft end position min.	
0x0000000E	Read / Write	every	REAL64	mm		soft end position max.	
0x0000000F	Read / Write	every	UINT32	1	s. ENUM (≥0)s. appendix	encoder analysis direction [▸ 494]	
0x00000010	Read / Write	every	REAL64	s	[0.0...60.0]	filter time for actual position value in seconds (P-T1)	
0x00000011	Read / Write	every	REAL64	s	[0.0...60.0]	filter time for actual velocity value in seconds (P-T1)	
0x00000012	Read / Write	every	REAL64	s	[0.0...60.0]	filter time for actual acceleration value in seconds (P-T1)	
0x00000013	Read / Write	every	UINT8[10+1]	1		physical unit	Not implemented!
0x00000014	Read / Write	every	UINT32	1		interpretation of the units (position, velocity, time) Bit 0: velocity in x/min instead of x/s Bit 1: position in thousand part of the base unit	Not implemented! Bitarray
0x00000015	Read	every	UINT32	INC	[0x0... 0xFFFFFFFF]	Standard Encoder Mask (maximum incremental encoder feedback value) **Note:** The Encoder Mask can hold any value (e.g. 3600000), not only a continuous sequence of binary 1 bits (2^n-1).	NEW ReadOnly parameters. param. "Encoder Sub Mask"
0x00000015	Read	every (for 64 bit encoder feedback)	UINT64	INC	[0x0... 0x0000FFFFFFFFFFFF]	Optional 64 Bit Encoder Mask (maximum incremental encoder feedback value) **Note:** Only supported for the encoder type MDP 513 with 64 bit interface like EL5032 (DS402, EnDat2.2).	NEW from TC3

Index offset (Hex)	Access	group- type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000016	Read / Write	every	UINT16	1	0/1	actual position compensation (measuring system error compensation)?	
0x00000017	Read / Write	every	REAL64	s	[0.0...60.0]	filter time for actual position compensation in seconds (P-T1)	
0x00000018	Read / Write	every	UINT32	1	[0x0... 0xFFFFFFFF]	filter mask for row incremental value(0x0: full passage)	
0x00000019	Read / Write	every	UINT32	1	s. ENUM (≥0)s. appendix	encoder absolute dimensioning system [▸ 494]	
0x0000001B	Read / Write	every	REAL64	e.g. mm	[≥0, modulo-factor/2]	tolerance window for modulo-start	
0x0000001C	Read	every	UINT32	1	s. ENUM (≥0)	encoder sign interpretation [▸ 494] (data type)	
0x0000001D	Read	every	UINT16	1	0/1	Incremental- or absolut-encoder ? 0: incremental encoder type 1: absolute encoder type	
0x00000020	Read / Write	every	UINT32	1	s. ENUM (≥0)	Encoder dead time compensation mode 0: OFF (Default) 1: ON (with velocity) 2: ON (with velocity and acceleration)	
0x00000021	Read / Write	every	UINT32	1		Control double word (32 Bit) for the encoder dead time compensation:Bit 0 = 0: relative IO time (default) Bit 0 = 1: absolute IO time	
0x00000022	Read / Write	every	INT32	ns	[1.0E+9]	Sum of the parameterized time shift for the encoder dead time compensation (typically positive values)	
0x00000023	Read / Write	every	REAL64	e.g. mm/ INC	[1.0E-12 ... 1.0E+30]	component of the scaling factor: numerator (=> scaling factor numerator / scaling factor denominator)	NEW from TC3
0x00000024	Read / Write	every	REAL64	1	[1.0E-12 ... 1.0E+30]	component of the scaling factor: denominator (=> scaling factor numerator / scaling factor denominator) default: 1.0	NEW from TC3

0x00000025	Read / Write	every	{
			REAL64
			REAL64
			}
				
				e.g. mm/ INC
				1
					
					[1.0E-12 ... 1.0E+30]
					[1.0E-12 ... 1.0E+30]
						
						component of the scaling factor: numerator
						component of the scaling factor: denominator
						(=> scaling factor numerator / scaling factor denominator)
							NEW from TC3

0x00000030	Read / Write	every	UINT32	1		Internal Encoder Control DWord	NEW from TC3
0x00000101	Read / Write	INC	UINT16	1	[0,1]	search direction for Ref. cam inverse?	
0x00000102	Read / Write	INC		1	[0,1]	search direction for sync pulse inverse?	
0x00000103	Read / Write	INC	REAL64	e.g. mm	[1.0E+9]	reference positon	
0x00000104	Read / Write	INC	UINT16	1	[0,1]	distance moitoring between Ref. cams and sync pulse active?	Not implemented!

Index offset (Hex)	Access	group- type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000105	Read / Write	INC	UINT32	INC	[0 ...65536]	minimum distance between Ref. cams and sync pulse in increments	Not implemented!
0x00000106	Read / Write	INC	UINT16	1	[0,1]	external sync pulse?	
0x00000107	Read / Write	INC	UINT32	1	s. ENUM (>0) s. appendix	reference mode [▸ 495] s.appendix	
0x00000108	Read / Write	INC	UINT32	1	[0x0000000F... 0xFFFFFFFF]Binary mask: (2^n - 1)	Encoder Sub Mask (maximum incremental value of the enocder absolute range) For example used as reference signal for homing with "Software Sync" or used for the NC Retain Data ("ABSOLUTE (MODULO)", "INCREMENTAL (SINGLETURN ABSOLUTE)"). **Note 1:** The Encoder Sub Mask must be less or equal than the Encoder Mask. **Note 2:** The Encoder Mask must be an integer multiple of the Encoder Sub Mask. **Note 3:** The Encoder Sub Mask must be a continuous sequence of binary 1 bits (2^n-1); e.g. 0x000FFFFF.	NEW s. param. "Encoder Mask"
0x00000110	Read / Write	INC (encoder simulation)	REAL64	1	[0.0 ... 1000000.0]	scaling/weight of the noise part for the simulation encoder	

7.8.5.2 "Index offset" specification for Encoder state (Index group 0x5100 + ID)

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x00000001	Read	every	INT32			error state encoder	
0x00000002	Read	every	REAL64			Actual position (charge with actual position compensation value)	symbolic access possible! *'fPosIst'*
0x00000003	Read	every	REAL64			Modulo actual position	symbolic access possible! *'FModuloPosIst'*
0x00000004	Read	every	INT32			Modulo actual rotation	symbolic access possible! *'nModuloTurns'*
0x00000005	Read	every	REAL64			Optional: actual velocity	Base Unit / s *Symbolic access possible! 'fVeloIst'*
0x00000006	Read	every	REAL64			Optional: Actual acceleration	Base unit / s^2 *Symbolic access possible! 'fAccIst'*
0x00000007	Read	every	INT32			Encoder actual increments	symbolic access possible! *'nHardIncs'*
0x00000008	Read	every	INT64			Software - actual increment counter	symbolic access possible! *'nSoftIncs'*
0x00000009	Read / Write	every	UINT16			Reference flag ("calibrate flag")	
0x0000000A	Read	every	REAL64			Actual position correction value (measurement system error correction)	
0x0000000B	Read	every	REAL64			Actual position without actual position compensation value	
0x0000000C	Read	every	REAL64	e.g. mm		Actual position compensation value due to the dead time compensation	
0x0000000D	Read	every	REAL64	s		Sum of the time shifts for encoder dead time compensation (parameterized and variable dead time) Note: a dead time is specified in the system as a positive value.	
0x0000000E	Read	every	REAL64	e.g. mm		Internal position offset as a correction value for a value reduction to the base period (modulo range)	
0x00000010	Read	every	REAL64	e.g. mm/s		Actual velocity without actual position compensation value	
0x00000012	Read	every	REAL64	e.g. mm		Unfiltered actual position (charge with actual position compensation value)	
0x00000014	Read	Type: SoE, CoE, MDP 742	REAL64	e.g. mm/s		Optional: Actual drive velocity (transferred directly from SoE, CoE or MDP 742 drive)	Base Unit / s NEW from TC3.1 B4020.30
0x00000015	Read	every	REAL64	e.g. mm/s		Optional: Unfiltered actual velocity	Base Unit / s

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x00000016	Read	every	**READ**(16 bytes * N)			Read the actual position buffer	
			{				
			UINT32	ns	≥0	DcTimeStamp with 32 bits	
			UINT32			Reserve	
			REAL64	e.g. mm	±∞	Actual position for the associated time stamp	
			} [N]				
0x00000101	Read	INC	REAL64	e.g. mm		Read back the position difference between the hardware latch being activated and becoming valid	Cannot be traced by oscilloscope!
0x00000200	**ReadWrite**	**function group "Touch-ProbeV2":- SERCOS/SoE, - EtherCAT/ CoE (CANopen DS402) - SoftDrive (TCom), - MDP 511 (EL5101, EL5151, EL5021, EL7041, EL7342)**	**WRITE**(24 bytes)			read "Touch Probe" state (state of external latch)	Only for SAF-port 501
			{				
			UINT32	1	[1,2,3,4]	probe unit (probe 1, 2, 3, 4)	
			UINT32[5]			reserved	
			}				
			READ (64 bytes)				
			{				
			UINT32	1	[0/1]	touch probe rising edge active?	
			UINT32	1	[0/1]	touch probe rising edge became valid?	
			REAL64	e.g. mm		touch probe rising edge position value	
			UINT32	1	≥0	touch probe rising edge counter (continuous mode)	
			UINT32			reserved	
			UINT32	1	[0/1]	touch probe falling edge active?	
			UINT32	1	[0/1]	touch probe falling edge became valid?	
			REAL64	e.g. mm		touch probe falling edge position value	
			UINT32	1	≥0	touch probe falling edge counter (continuous mode)	
			UINT32[5]			reserved	
			}				
0x00000201	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT16	1	[0,1]	"External latch function" active ? or "Measuring probe function" active ? *(edge-independent)*	Cannot be traced by oscilloscope!
0x00000201	Read	**CANopen**	UINT32[4]	1	[0,1]	"External latch functions 1 to 4" active ? or "Measuring probe functions 1 to 4" active ?	Cannot be traced by oscilloscope!
0x00000202	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT16	1	[0,1]	External latch value became valid? or Measuring probe latched ? *(edge-independent)*	see also Axis interface NcToPlc (status double word)

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Note
0x00000202	Read	**CANopen**	UINT32[4]	1	[0,1]	Have external latch values 1 to 4 become valid? Or is measuring probe 1 to 4 latched?	see also Axis interface NcToPlc (status double word)
0x00000203	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT32	INC		External / measuring probe hardware incremental latch value	
0x00000204	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT64	INC		External / measuring probe software incremental latch value	
0x00000205	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	REAL64	e.g. mm		External / measuring probe position latch value	Base Unit
0x00000205	Read	**CANopen**	REAL64[4]	e.g. mm		External measuring sensor values / position latch values	Base Unit
0x00000206	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT32	INC		Difference hardware incremental latch values (NewLatch - LastLatch)	Cannot be traced by oscilloscope!
0x00000207	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	UINT64	INC		Difference software incremental latch values (NewLatch - LastLatch)	Cannot be traced by oscilloscope!
0x00000208	Read	**KL5101, SERCOS, AX2xxx, ProviDrive**	REAL64	e.g. mm		Difference software incremental latch values (NewLatch - LastLatch)	Cannot be traced by oscilloscope! Base Unit
0x00000210	Read	**KL5101, AX2xxx, ProviDrive**	UINT16	1	[0,1]	"External latch function" for *rising edge* active? or "Measuring probe function" for *rising edge* active?	Cannot be traced by oscilloscope!
0x00000210	Read	**CANopen**	UINT16[4]	1	[0,1]	"External latch function" for *rising edge* active? or "Measuring probe function" for *rising edge* active?	Cannot be traced by oscilloscope!
0x00000211	Read	**KL5101, AX2xxx, ProviDrive**	UINT16	1	[0,1]	"External latch function" for *falling edge* active? or "Measuring probe function" for *falling edge* active?	Cannot be traced by oscilloscope!
0x00000211	Read	**CANopen**	UINT16[4]	1	[0,1]	"External latch function" for *falling edge* active? or "Measuring probe function" for *falling edge* active?	Cannot be traced by oscilloscope!

7.8.5.3 "Index offset" specification for Encoder functions (Index group 0x5200 + ID)

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x0000001A	Write	every	{			Set actual position encoder/axis	Base Unit
			UINT32	ENUM	s. appendix	Actual position type [▸ 490] (s. appendix)	
			REAL64	mm	±∞	Actual position for encoder/axis **Caution when using !**	
			}				
0x0000001B	Write	every	VOID			Re-initialization of the actual encoder position Note: Only takes effect for reference system "ABSOLUTE (with single overflow)".	NEW from TC3
0x00000200	Write	**function group "Touch-ProbeV2": - SERCOS/ SoE, - EtherCAT/ CoE (CANopen DS402) - SoftDrive (TCom), - MDP 511 (EL5101, EL5151, EL5021, EL7041, EL7342)**	{			activate "Touch Probe" (external latch)	only for SAF-port 501
			UINT32	1	[1,2,3,4]	probe unit (probe 1, 2, 3, 4)	
			UINT32	1	[0,1]	signal edge (0=rising edge, 1=falling edge)	
			UINT32	1	[1,2]	probe mode (1=single, 2=continuous, ...)	
			UINT32	1	[1,2,3,4; 128,129]	signal source (1=input 1, 2=input 2, ...)	
			UINT32			reserved	
			UINT32			reserved	
			} 24 bytes				
0x00000201	Write	**KL5101,SERCOS,AX2xxx,PROFIDrive**	VOID			Activate "External Latch" or Activate "measuring probe function" *(typically rising edge)*	
0x00000201	Write	**CANopen**	UINT32[4]			Activate "External Latch" 1 to 4 or Activate "measuring probe function" 1 to 4 *(typically rising edge)*	
0x00000202	Write	**KL5101,SERCOSAX2xxx,PROFIDrive**	VOID			Activate "external latch" or activate "measuring probe function" *(falling edge)*	
0x00000202	Write	**CANopen**	UINT32[4]			Activate "external latch" 1 to 4 or activate "measuring probe function" 1 to 4 *(falling edge)*	
0x00000205	Write	**function group "Touch-ProbeV2": - SERCOS/ SoE, - EtherCAT/ CoE (CANopen DS402) - SoftDrive (TCom), - MDP 511 (EL5101, EL5151, EL5021, EL7041, EL7342)**	{			Deactivate "touch probe" (external latch)	only for SAF-port 501
			UINT32	1	[1,2,3,4]	probe unit (probe 1, 2, 3, 4)	
			UINT32	1	[0,1]	signal edge (0=rising edge, 1=falling edge)	
			UINT32			reserved	
			UINT32			reserved	
			UINT32			reserved	
			UINT32			reserved	
			} 24 bytes				

Index-Offset (Hex)	Access	Group type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000205	Write	**KL5101,SER-COS,AX2xxx,PROFIDrive**	VOID			Deactivate "external latch" or deactivate "measuring probe function"	
0x00000205	Write	**CANopen**	UINT32[4]			Deactivate "external latch" or deactivate "measuring probe function"	
0x00000210	Write	**KL5101,SER-COS,AX2xxx,PROFIDrive**	REAL64	e.g. mm	±∞	Set "External latch event" and "External latch position"	Only for Ether-CAT:

7.8.5.4 "Index offset" specification for cyclic Encoder process data (Index group 0x5300 + ID)

Index offset (Hex)	Access	group- type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000000	Read / Write	every (NC→IO)	{		STRUCT s. encoder-interface	ENCODER-OUTPUT-STRUCTURE (NC→IO, 40 Byte)NCENCODER-STRUCT_OUT2	Write command only optional! Consider safety aspects !
			INT32	INC	≥ 0	nDataOut1	
			INT32	INC	≥ 0	nDataOut2	
			UINT8	1	≥ 0	nCtrl1	
			UINT8	1	≥ 0	nCtrl2	
			UINT8	1	≥ 0	nCtrl3	
			UINT8	1	≥ 0	nCtrl4	
			INT32	INC	≥ 0	nDataOut3	
			INT32	INC	≥ 0	nDataOut4	
			INT32	INC	≥ 0	nDataOut5	
			INT32	INC	≥ 0	nDataOut6	
			UINT8	1	≥ 0	nCtrl5	
			UINT8	1	≥ 0	nCtrl6	
			UINT8	1	≥ 0	nCtrl7	
			UINT8	1	≥ 0	nCtrl8	
			INT32	1	≥ 0	reserved	
			INT32	1	≥ 0	reserved	
			} 40 bytes				
0x00000000	Read / Write	every (NC→IO), optional 64 bit encoder interface (e.g. MDP513 with 64Bit)	{		STRUCT s. encoder-interface	optional ENCODER-OUTPUT-STRUCTURE (NC→IO, 80 Byte)NCENCODERSTRUCT_OUT3	Write command only optional! Consider safety aspects !NEW from TC3
			UINT64	INC	≥ 0	nDataOut1	
			UINT64	INC	≥ 0	nDataOut2	
			UINT64	INC	≥ 0	nDataOut3	
			UINT64	INC	≥ 0	nDataOut4	
			UINT64	INC	≥ 0	nDataOut5	
			UINT64	INC	≥ 0	nDataOut6	
			UINT64	INC	≥ 0	nDataOut7	
			UINT64	INC	≥ 0	nDataOut8	
			UINT16	1	≥ 0	nCtrl1	
			UINT16	1	≥ 0	nCtrl2	
			UINT16	1	≥ 0	nCtrl3	
			UINT16	1	≥ 0	nCtrl4	
			UINT16	1	≥ 0	nCtrl5	
			UINT16	1	≥ 0	nComCtrl	
			INT32	1	≥ 0	reserved	
			} 80 bytes				

Index offset (Hex)	Access	group- type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Write	Every (NC→IO)	{		STRUCT s. Encoder-Interface	Bitwise access to ENCODER-OUTPUT-STRUCTURE (NC→IO, 40 Byte) NC-DRIVESTRUCT_OUT2	Write command only optional! Consider safety aspects !NEW from TC3
			UINT32	1	[0 … 39]	ByteOffset Relative address offset [0..39] in drive output structure. E.G.: To write “nControl1” the ByteOffset must be 8.	
			UINT32	1	[0x00000000… 0xFFFFFFFF]	BitSelectMask (BSM) The mask defines write enabled bits in a DWORD. Zero bits are protected and remain unaffected.	
			UINT32	1	[0x00000000… 0xFFFFFFFF]	Value Only those bits in value are overwritten where BSM equals 1.	
			}				
0x00000080	Read	every (IO→NC)	{		STRUCT s. encoder-interface	ENCODER-INPUT-STRUCTURE (IO→NC, 40 Byte)NCENCODER-STRUCT_IN2	
			INT32	INC	≥ 0	nDataIn1	
			INT32	INC	≥ 0	nDataIn2	
			UINT8	1	≥ 0	nState1	
			UINT8	1	≥ 0	nState2	
			UINT8	1	≥ 0	nState3	
			UINT8	1	≥ 0	nState4 (Bit0: *WcState*, Bit1: *InputToggle*)	
			INT32	INC	≥ 0	nDataIn3	
			INT32	INC	≥ 0	nDataIn4	
			INT32	INC	≥ 0	nDataIn5	
			INT32	INC	≥ 0	nDataIn6	
			UINT8	1	≥ 0	nState5	
			UINT8	1	≥ 0	nState6	
			UINT8	1	≥ 0	nState7	
			UINT8	1	≥ 0	nState8	
			INT32	[ns]	≥ 0	nDcInputTime (absolute/ relative *DcInputShift* for deadtime compensation)	
			INT32	1	≥ 0	reserved	
			} 40 bytes				

Index offset (Hex)	Access	group- type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000080	Read	every (NC→IO), optional 64 bit encoder interface (e.g. MDP513 with 64Bit)	{		STRUCT s. encoder-interface	optional ENCODER-INPUT-STRUCTURE (IO→NC, 80 Byte)NCENCODERSTRUCT_IN3	NEW from TC3
			UINT64	INC	≥ 0	nDataIn1	
			UINT64	INC	≥ 0	nDataIn2	
			UINT64	INC	≥ 0	nDataIn3	
			UINT64	INC	≥ 0	nDataIn4	
			UINT64	INC	≥ 0	nDataIn5	
			UINT64	INC	≥ 0	nDataIn6	
			UINT64	INC	≥ 0	nDataIn7	
			UINT64	INC	≥ 0	nDataIn8	
			UINT16	1	≥ 0	nState1	
			UINT16	1	≥ 0	nState2	
			UINT16	1	≥ 0	nState3	
			UINT16	1	≥ 0	nState4	
			UINT16	1	≥ 0	nState5	
			UINT16	1	≥ 0	nComState (Bit0: *WcState*, Bit1: *InputToggle*)	
			INT32	[ns]	≥ 0	nDcInputTime (absolute/ relative *DcInputShift* for deadtime compensation)	
			} 80 bytes				

7.8.6 Specification Controller

7.8.6.1 "Index offset" specification for Controller parameter (Index group 0x6000 + ID)

Index offset (Hex)	Access	controller type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	UINT32	1	[1 ... 255]	controller ID	
0x00000002	Read	every	UINT8[30+1]	1	30 symbol	controller name	
0x00000003	Read	every	UINT32	1	s. ENUM (>0)	controller type [▸ 492]	
0x0000000A	Read / Write	every	UINT32	1	s. ENUM (>0)	controller mode	DEFAULT: 1=STANDARD
0x0000000B	Read / Write	every	REAL64	%	[0.0 ... 1.0]	weight of the velocity pre control (standard value: 1.0 = 100 %)	
0x00000010	Read / Write	every	UINT16	1	0/1	following error monitoring positon?	
0x00000011	Read / Write	every	UINT16	1	0/1	following error monitoring velocity?	
0x00000012	Read / Write	every	REAL64	mm	[0.0…1.0E.6]	max. following error position	
0x00000013	Read / Write	every	REAL64	s	[0.0...600]	max. following error time position	
0x00000014	Read / Write	every	REAL64	mm/s	[0.0…1.0E.6]	max. following error velocity	
0x00000015	Read / Write	every	REAL64	s	[0.0…1.0E.6]	max. following error time velocity	
0x00000021	Read / Write	every	REAL64	1	[0.0...1000000.0]	scaling factor (multiplier) for position differences between master and slave axis (conversion in the same coordinate system)	reserved function, no standard!
0x00000100	Read / Write	P/PID (Pos., (velocity)	REAL64	1	[0.0...1.0]	Maximum output limitation ( ) for controller total output	(Standard value: 0.5 == 50%)
0x00000102	Read / Write	P/PID (Pos.)	REAL64	mm/s/ mm	[0.0...1000.0]	proportional amplification factor k_p resp. k_v	base unit / s / base unit position control
0x00000103	Read / Write	PID (Pos.)	REAL64	s	[0.0 ... 60.0]	Integral-action time Tn	position control
0x00000104	Read / Write	PID (Pos.)	REAL64	s	[0.0 ... 60.0]	Derivative action time Tv	position control
0x00000105	Read / Write	PID (Pos.)	REAL64	s	[0.0 ... 60.0]	Damping time Td	position control
0x00000106	Read / Write	PP (Pos.)	REAL64	mm/s/ mm	[0.0...1000.0]	add. proportional amplification factor kp resp. kv, that applies above a limit velocity in percent.	base unit / s / base unit position control
0x00000107	Read / Write	PP (Pos.)	REAL64	%	[0.0...1.0]	Threshold level velocity in percent, above which the additional proportional amplification factor kp resp. kv applies.	(Standard value: 0.01 == 1%)
0x00000108	Read / Write	P/PID (Acc.)	REAL64	s	[0.0 ... 100.0]	proportional amplification factor ka	acceleration pre control
0x0000010A	Read / Write	every	UINT32	1	ENUM	Filter for maximum slope of the nominal velocity (acceleration restricted): 0: Off, 1: Velo, 2: Pos +Velo	reserved function, no standard!
0x0000010B	Read / Write	every	REAL64	mm/s^2		Filter value for the maximum slope of the nominal velocity (max. acceleration)	reserved function, no standard!
0x0000010D	Read / Write	P/PID	REAL64	mm	[0.0 ... 10000.0]	'dead band' for position error (position deviation) (for P/PID-controller with velocity or torque interface)	reserved function
0x0000010F	Read / Write	P/PP/PID (Pos.) slave-control	REAL64	(mm/s) / mm	[0.0...1000.0]	slave coupling control: proportional gain k_{cp} for position deviation between master and slave	slave coupling control

Index offset (Hex)	Access	controller type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000110	Read / Write	P (Pos.)	UINT16	1	0/1	automatic offset calibration: active/passive	
0x00000111	Read / Write	P (Pos.)	UINT16	1	0/1	automatic offset calibration: hold mode	
0x00000112	Read / Write	P (Pos.)	UINT16	1	0/1	automatic offset calibration: fading mode	
0x00000114	Read / Write	P (Pos.)	REAL64	%	[0.0 ... 1.0]	automatic offset calibration: pre control limit	(Standard value: 0.05 == 5%)
0x00000115	Read / Write	P (Pos.)	REAL64	s	[0.1 ... 60.0]	automatic offset calibration: time constant	
0x00000116	Read / Write	PID (Pos.)	REAL64	%	[0.0...1.0]	Maximum output limitation ( ) for I- part in percent (default setting: 0.1 == 10 %)	
0x00000117	Read / Write	PID (Pos.)	REAL64	%	[0.0...1.0]	Maximum output limitation ( ) for D- part in percent (default setting: 0.1 == 10 %)	
0x00000118	Read / Write	PID (Pos.)	UINT16	1	0/1	Switch off the I-part during an active positioning process (as far as I-part active)? (default setting: 0 = FALSE)	
0x00000120	Read / Write	P/PID (Pos.)	REAL64	s	≥0	PT-1 Filtertime for position error (position-difference)	reserved function, no standard!
0x00000202	Read / Write	P/PID (velocity)	REAL64	1	[0.0...1000.0]	proportional amplification factor k_p resp. k_v	velocity control
0x00000203	Read / Write	PID (velocity)	REAL64	s	[0.0 ... 60.0]	Integral-action time T_n	velocity control
0x00000204	Read / Write	PID (velocity)	REAL64	s	[0.0 ... 60.0]	Derivative action time T_v	velocity control
0x00000205	Read / Write	PID (velocity)	REAL64	s	[0.0 ... 60.0]	Damping timeT_d	velocity control
0x00000206	Read / Write	PID (velocity)	REAL64	%	[0.0...1.0]	Maximum output limitation ( ) for I-part in percent (default setting: 0.1 == 10 %)	velocity control
0x00000207	Read / Write	PID (velocity)	REAL64	%	[0.0...1.0]	Maximum output limitation ( ) for D-part in percent (default setting: 0.1 = 10 %)	velocity control
0x0000020D	Read / Write	P/PID (velocity)	REAL64	mm/s	[0.0 ... 10000.0]	'dead band' for velocity error (velocity deviation) (for P/PID-controller with velocity or torque interface)	reserved function
0x00000220	Read / Write	P/PID (velocity)	REAL64	s	≥0	PT-2 Filtertime for velocity error (velocity-difference)	velocity control, no standard!
0x00000221	Read / Write	P/PID (velocity)	REAL64	s	≥0	PT-1 Filtertime for velocity error (velocity-difference)	reserved function, no standard!
0x00000250	Read / Write	P/PI (observer)	UINT32	1	s. ENUM (≥0)	OBSERVER-mode [▸ 492] for controller with torque interface 0: OFF (default) 1: LUENBERGER	
0x00000251	Read / Write	P/PI (observer)	REAL64	Nm / A	>0.0	motor: torque constant K_T	
0x00000252	Read / Write	P/PI (observer)	REAL64	kg m^2	>0.0	motor: moment of inertia J_M	

Index offset (Hex)	Access	controller type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000253	Read / Write	P/PI (observer)	REAL64	Hz	[100.0 ... 2000.0] Default: 500	bandwidth f_0	
0x00000254	Read / Write	P/PI (observer)	REAL64	1	[0.0 ... 2.0] Default: 1.0	correction factor k_c	
0x00000255	Read / Write	P/PI (observer)	REAL64	s	[0.0 ... 0.01] Default: 0.001	velocity filter (1. order): filter time constant T	
0x00000A03	Read / Write	PID (MW)	REAL64	cm^2	[0.0 ...1000000]	cylinder area A_A of side A in cm^2	
0x00000A04	Read / Write	PID (MW)	REAL64	cm^2	[0.0 ...1000000]	cylinder area A_B of side B in cm^2	
0x00000A05	Read / Write	PID (MW)	REAL64	cm^3/s	[0.0 ...1000000]	nominal volume flow Q_{nenn} in cm^3/s	
0x00000A06	Read / Write	PID (MW)	REAL64	bar	[0.0 ...1000000]	nominal pressure resp. valve pressure reduction P_{nenn} in bar	
0x00000A07	Read / Write	PID (MW)	UINT32	1	[1 ... 255]	axis ID for the system pressure P_o	

7.8.6.2 "Index offset" specification for Controller state (Index group 0x6100 + ID)

Index-Offset (Hex)	Access	Controller type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	INT32			error state controller [▶ 353]	Symbolic access possible! *'nErrState'*
0x00000002	Read	every	REAL64	e.g. mm/s		Controller output in absolute units	Base unit / s *Symbolic access possible! 'fOutput'*
0x00000003	Read	every	REAL64	%		Controller output in percent	Cannot be traced by oscilloscope!
0x00000004	Read	every	REAL64	V		Controller output in volts	Cannot be traced by oscilloscope!
0x0000000D	Read	every	REAL64	mm		Following error position (without dead time compensation)	Base Unit
0x0000000E	Read	every	REAL64	mm		Following error position (without set position correction)	Base Unit
0x0000000F	Read	every	REAL64	mm		Following error position (with set position correction and dead time compensation)	Base unit *Symbolic access possible! 'fPosDiff'*
0x00000010	Read	every	REAL64	mm		Peak hold value for maximum negative following error of the position	Base Unit
0x00000011	Read	every	REAL64	mm		Peak hold value for minimum positive following error of the position	Base Unit
0x00000012	Read	every	REAL64	mm/s		Following error velocity	Base Unit / s
0x00000021	Read	every	REAL64	mm		Difference (deviation) between the following error from master and slave axis (master error minus slave error)	Base unit *Symbolic access via axis possible! fPosDiffCouple*
0x00000022	Read	every	REAL64	mm		PeakHold value for the maximum negative difference between master and slave axis following error of the position	Base Unit
0x00000023	Read	every	REAL64	mm		PeakHold value for the maximum positive difference between master and slave axis following error of the position	Base Unit
0x00000101	Read	P/PID (pos.)	REAL64	e.g. mm/s		Proportional component of the controller in absolute units	
0x00000102	Read	PID (pos.)	REAL64	e.g. mm/s		I-part of the controller in absolute units	
0x00000103	Read	PID (pos.)	REAL64	e.g. mm/s		D-part of the controller in absolute units	
0x00000104	Read	PID (pos.)	UINT16	1	0/1	Limitation of the I-part active?	
0x00000105	Read	PID (pos.)	UINT16	1	0/1	Limitation of the D-part active?	
0x00000106	Read	PID (pos.)	UINT16	1	0/1	ARW measure for the I-part active?	ARW: Anti Reset Windup

Index-Offset (Hex)	Access	Controller type	Data type	Phys. unit	Definition range	Description	Remarks
0x0000010F	Read	P/PP/PID (ve-loc.)	REAL64	e.g. mm/s		Proportion of automatic offset compensation in absolute units	NEW
0x00000110	Read	PID (pos.)	REAL64	e.g. mm/s		Acceleration pre-control Y_{acc} of the controller in absolute units, note: Function depends on controller type!	acceleration pre-control
0x00000111	Read	PP (Pos.)	REAL64	mm/s/mm	≥0	Internal interpolated proportional gain kp or kv	PP controller
0x0000011A 0x0000011B 0x0000011C 0x0000011D 0x0000011E 0x0000011F 0x00000120 0x00000121 0x00000122 0x00000123 0x00000124	Read	P (Pos.)	UINT32 REAL64 REAL64 REAL64 REAL64 REAL64 REAL64 REAL64 REAL64 REAL64 REAL64	1 mm mm/s mm/s mm/s^2 mm mm mm/s mm/s^2 mm/s mm/s^2		Set velocity filter: InternalPhase InternalPosSollError ! TestVeloSet InternalLimitedVeloSoll InternalAccSollRel InternalPosSollRel PosSetCorrected ! VeloSetCorrected ! AccSollCorrected ! TestVeloSetCorrected TestAccSetCorrected	List! Reserved function, no standard!
0x00000201	Read	P,PID (velocity)	REAL64	e.g. mm/s		Velocity part of the controller	Base Unit / s
0x00000202	Read	P,PID (velocity)	REAL64	%		Velocity part of the controller in percent	Cannot be traced by oscilloscope!
0x00000203	Read	P,PID (velocity)	REAL64	V		Velocity part of the controller in volts	Cannot be traced by oscilloscope!
0x00000201	Read	P/PID (velocity)	REAL64	e.g. mm/s		Proportional component of the controller in absolute units	
0x00000202	Read	P/PID (velocity)	REAL64	e.g. mm/s		I-part of the controller in absolute units	
0x00000203	Read	P/PID (velocity)	REAL64	e.g. mm/s		D-part of the controller in absolute units	
0x00000204	Read	P/PID (velocity)	UINT16	1	0/1	Limitation of the I-part active?	
0x00000205	Read	P/PID (velocity)	UINT16	1	0/1	Limitation of the D-part active?	
0x00000206	Read	P/PID (velocity)	UINT16	1	0/1	ARW measure for the I-part active?	ARW: Anti Reset Windup
0x0000020A	Read	P/PID (velocity)	REAL64	e.g. mm/s		Total input size of the velocity controller	
0x00000250	Read	P/PI (observer)	REAL64	e.g. mm		Observer: Position difference (actual position - observer position	
0x00000251	Read	P/PI (observer)	REAL64	e.g. mm		Observer: Position	
0x00000252	Read	P/PI (observer)	REAL64	e.g. mm/s		Observer: Velocity 2 (for proportional component)	
0x00000253	Read	P/PI (observer)	REAL64	e.g. mm/s		Observer: Velocity 1 (for I-part)	
0x00000254	Read	P/PI (observer)	REAL64	e.g. mm/s^2		Observer: Acceleration	

Index-Offset (Hex)	Access	Controller type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000255	Read	P/PI (observer)	REAL64	A		Observer: actual motor current	
0x00000256	Read	P/PI (observer)	UINT16	1	0/1	Observer: Limitation of the I-part active?	
0x00000A00	Read	PID (MW)	REAL64	%	[-1.0...1.0]	Calculation of the set speed (pilot control) in percent	
0x00000A01	Read	PID (MW)	REAL64	e.g. mm/s		Proportional component of the controller in absolute units or percent (according to output weight)	
0x00000A02	Read	PID (MW)	REAL64	e.g. mm/s		I-part of the controller in absolute units or percent (according to output weight)	
0x00000A03	Read	PID (MW)	REAL64	e.g. mm/s		D-part of the controller in absolute units or percent (according to output weight)	
0x00000A04	Read	PID (MW)	UINT16	1	0/1	Limitation of the I-part active?	
0x00000A05	Read	PID (MW)	UINT16	1	0/1	Limitation of the D-part active?	
0x00000A10	Read	PID (pos.)	REAL64	e.g. mm/s		Acceleration pre-control Y_{acc} of the controller in absolute units	Acceleration pre-control

7.8.6.3 "Index offset" specification for Controller functions (Index group 0x6200 + ID)

Index offset (Hex)	Access	controller - type	Data type	Phys. unit	Definition range	Description	Remarks

7.8.7 Specification Drive

7.8.7.1 "Index offset" specification for Drive parameter (Index group 0x7000 + ID)

Index offset (Hex)	Access	Drive- type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	UINT32	1	[1 ... 255]	Drive ID	
0x00000002	Read	every	UINT8[30+1]	1	30 symbol	Drive name	
0x00000003	Read	every	UINT32	1	s. ENUM (>0)	Drive type [▸ 496]	
0x00000004	Read / Write	every	UINT32	1	Byteoffset	Input address offset (IO-Input-Image)	change I/O address
0x00000005	Read / Write	every	UINT32	1	Byteoffset	output address offset (IO-Output-Image)	change I/O address
0x00000006	Read / Write	every	UINT16	1	[0,1]	Motor polarity	
0x0000000A	Read / Write	every	UINT32	1	s. ENUM (>0)	Drive mode	Default: 1=STANDARD
0x0000000B	Read / Write	every	REAL64	%	[-1.0 ... 1.0]	Minimum output limit (output limitation) (default setting: -1.0 == -100%)	
0x0000000C	Read / Write	every	REAL64	%	[-1.0 ... 1.0]	Maximum output limit (output limitation) (default setting: 1.0 == 100%)	
0x0000000D	Read	every	UINT32	INC		Maximum number of output increments (output mask)	
0x00000010	Read / Write	every	UINT32	1		Internal Drive Control DWord	Reserved !
0x00000011	Read / Write	every	UINT32	1	≥ 5	Internal Drive Reset Counter (time in nc-cycles for 'enable' and 'reset')	Reserved !
0x00000020	Read / Write	every	UINT32	1	s. ENUM (≥0)	Drive dead time compensation mode 0: OFF (Default) 1: ON (with velocity) 2: ON (with velocity and acceleration)	s. appendix
0x00000021	Read / Write	every	UINT32	1		Control double word (32 Bit) for the drive dead time compensation: Bit 0 = 0: relative IO time (default) Bit 0 = 1: absolute IO time	
0x00000022	Read / Write	every	INT32	ns	[1.0E+9]	Sum of the parameterized time shift for the drive dead time compensation (typically positive values)	
0x00000031	Read / Write	every	REAL64	e.g. %/INC	[0.0 ... 1.0E+30]	Scaling factor for actual torque (resp. force or current) e.g. AX5xxx: 0.1 => ±100%	NEW from TC3.1
0x00000032	Read / Write	every	REAL64	s	[0.0 ... 60.0]	P-T1 filter time for actual torque (resp. force or current)	NEW from TC3.1
0x00000033	Read / Write	every	REAL64	s	[0.0 ... 60.0]	P-T1 filter time for actual torque derivative (resp. force or current)	NEW from TC3.1
0x00000101	Read / Write	Servo	REAL64	e.g. mm/s	>0.0	reference velocity at reference output	base unit / s
0x00000102	Read / Write	Servo	REAL64	%	[0.0 ... 5.0]	reference output in percent	
0x00000103	Read	Servo	REAL64	e.g. mm/s	>0.0	resultant velocity at 100% output	base unit / s
0x00000104	Read / Write	Servo	REAL64	e.g. mm/s	∞	velocity offset (DAC offset) for drift calibration (offset calibration) of the axis	base unit / s

Index offset (Hex)	Access	Drive- type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000105	Read / Write	Servo (Sercos, Profi Drive, AX200x, CANopen)	REAL64	1	[0.0 ... 100000000.0]	velocity scaling (scaling factor to react to the weight in the drive)	For Sercos, Profi Drive, AX200x, CANopen
0x00000106	Read / Write	Profi Drive DSC	UINT32	0.001 * 1/s	≥ 0	Profibus/Profi Drive DSC: proportional gain 'Kpc'	Only for Profi Drive DSC
0x00000107	Read / Write	Profi Drive DSC	REAL64	1	≥ 0.0	Profibus/Profi Drive DSC: scaling factor for calculating 'Xerr' (default setting: 1.0)	Only for Profi Drive DSC
0x00000109	Read / Write	Servo (Sercos, CANopen)	REAL64	1	[0.0 ... 100000000.0]	position scaling (scaling factor to react to the weight in the drive)	For Sercos, CANopen
0x0000010A	Read / Write	Servo (Sercos, Profi Drive, AX200x, CANopen)	REAL64	1	[0.0 ... 100000000.0]	acceleration scaling (scaling factor to react to the weight in the drive)	For Sercos, Profi Drive, AX200x, CANopen
0x0000010B	Read / Write	Servo (Sercos, Profi Drive, AX200x, CANopen)	REAL64	1	[0.0 ... 100000000.0]	torque scaling (rotary motor) resp. force scaling (linear motor) (scaling factor to react to the weight in the drive)	For Sercos, Profi Drive, AX200x, CANopen
0x0000010D	Read / Write	Servo (Sercos, CANopen)	REAL64	s	[0.0 ... 1.0]	delay for drive velocity output	For Sercos, CANopen
0x0000010E	Read / Write	Servo (Sercos, CANopen)	REAL64	s	[0.0 ... 1.0]	delay for drive acceleration output	For Sercos, CANopen
0x0000010F	Read / Write	Servo (Sercos, CANopen)	REAL64	s	[0.0 ... 1.0]	delay for drive torque resp. force output	For Sercos, CANopen
0x00000120	Read / Write	Servo/ Hydraulic	UINT32	1	≥ 0	table-ID (0: no table)	Only for KL4xxx, M2400, Universal
0x00000121	Read / Write	Servo/ Hydraulic	UINT32	1	≥ 0	Interpolation-type 0: Linear 2: Spline	Only for KL4xxx, M2400, Universal
0x00000122	Read / Write	Servo/ Hydraulic	REAL64	%	[-1.0 ... 1.0]	Output offset in percentRemark: effects after the characteristic table !	Only for KL4xxx, M2400, Universal
0x00000151	Read / Write	Servo / non linear	REAL64	1	[0.0 ... 100.0]	quadrant equalizing factor (relation between I and III quadr.)	
0x00000152	Read / Write	Servo / non linear	REAL64	1	[0.01 ... 1.0]	velocity calibration point in percent (1.0 == 100 %)	
0x00000153	Read / Write	Servo / non linear	REAL64	1	[0.01 ... 1.0]	output calibration point in percent (1.0 ==100 %)	
0x00000301	Read / Write	Stepper motor	UINT8			Bit mask: cycle 1	
0x00000302	Read / Write	Stepper motor	UINT8			Bit mask: cycle 2	
0x00000303	Read / Write	Stepper motor	UINT8			Bit mask: cycle 3	
0x00000304	Read / Write	Stepper motor	UINT8			Bit mask: cycle 4	
0x00000305	Read / Write	Stepper motor	UINT8			Bit mask: cycle 5	
0x00000306	Read / Write	Stepper motor	UINT8			Bit mask: cycle 6	
0x00000307	Read / Write	Stepper motor	UINT8			Bit mask: cycle 7	
0x00000308	Read / Write	Stepper motor	UINT8			Bit mask: cycle 8	
0x00000310	Read / Write	Stepper motor	UINT8			Bit mask: holding current	

7.8.7.2 "Index offset" specification for Drive state (Index group 0x7100 + ID)

Index offset (Hex)	Access	Drive- type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	INT32			error state drive	symbolic Access possible!*'nErrState'*
0x00000002	Read	every	REAL64	mm/s		total output in absolute units	base unit / s *symbolic Access possible!'fOutput'*
0x00000003	Read	every	REAL64	%		total output in per cent	
0x00000004	Read	every	REAL64	V		total output in Volt	Cannot be traced by oscilloscope!!
0x00000005	Read	every	REAL64	e.g. mm/s		PeakHold value for maximum negative total output	Base Unit / s
0x00000006	Read	every	REAL64	e.g. mm/s		PeakHold value for maximum positive total output	Base Unit / s
0x00000007	Read	every	REAL64	e.g. 100%=1000, e.g. Nm resp. N		actual torque resp. actual force value (typically 100%=1000)	NEW
0x0000000C	Read	every	REAL64	e.g. mm		set position compensation value for the drive output because of the dead time compensation	
0x0000000D	Read	every	REAL64	s		total time shift for the drive dead time compensation (parameterized and variable dead time)Remark: A dead time is defined in the system as a positive value.	
0x00000013	Read	every	REAL64	%		total output in percenty (after non linear valve diagramm !)	
0x00000014	Read	every	REAL64	V		total output in voltage (after non linear valve diagramm !)	Cannot be traced by oscilloscope!!
0x0000011A	Read	Servo (Sercos, CANopen)	REAL64	e. g. mm		Optional Smoothingfilter: Filtered set position	NEW For Sercos, CANopen
0x0000011E	Read	Servo (Sercos, CANopen)	REAL64	e. g. mm/s		Optional Smoothingfilter: Filtered set velocity	NEW For Sercos, CANopen
0x0000011F	Read	Servo (Sercos, CANopen)	REAL64	e. g. mm/s^2		Optional Smoothingfilter: Filtered set acceleration / deceleration	NEW For Sercos, CANopen

7.8.7.3 "Index offset" specification for Drive functions (Index group 0x7200 + ID)

Index offset (Hex)	Access	Drive- type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000102	Write	SERVO	{			Remove and delete the characteristic drive table	only for SAF-port 501!
			ULONG	1	>0	Table-ID s.a. axis function with index offset 0x00000012	
			}				

7.8.7.4 "Index offset" specification for cyclic Drive process data (Index group 0x7300 + ID)

Index offset (Hex)	Access	Drive- type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000000	Read / Write	every (NC→IO)	{		STRUCT s. drive-interface	DRIVE-OUTPUT-STRUCTURE (NC→IO, 40 Byte)*NC-DRIVESTRUCT_OUT2*	Write command only optional! Consider safety aspects !
			INT32	INC	≥ 0	nOutData1	
			INT32	INC	2^31	nOutData2	
			UINT8	1	≥ 0	nControl1	
			UINT8	1	≥ 0	nControl2	
			UINT8	1	≥ 0	nControl3	
			UINT8	1	≥ 0	nControl4	
			INT32	INC	≥ 0	nOutData3	
			INT32	INC	≥ 0	nOutData4	
			INT32	INC	≥ 0	nOutData5	
			INT32	INC	≥ 0	nOutData6	
			UINT8	1	≥ 0	nControl5	
			UINT8	1	≥ 0	nControl6	
			UINT8	1	≥ 0	nControl7	
			UINT8	1	≥ 0	nControl8	
			INT32	1	≥ 0	reserved	
			INT32	1	≥ 0	reserved	
			}				
0x00000001	Write	every (NC→IO)	{		STRUCT s. Drive-Interface	Bitwise access to DRIVE-OUTPUT-STRUCTURE(NC→IO, 40 Byte) *NC-DRIVESTRUCT_OUT2*	Write command only optional! Consider safety aspects
			UINT32	1	[0 … 39]	ByteOffset Relative address offset [0..39] in drive output structure. E.G.: To write “nControl1” the ByteOffset must be 8.	
			UINT32	1	[0x00000000… 0xFFFFFFFF]	BitSelectMask (BSM) The mask defines write enabled bits in a DWORD. Zero bits are protected and remain unaffected.	
			UINT32	1	[0x00000000… 0xFFFFFFFF]	Value Only those bits in value are overwritten where BSM equals 1.	
			}				

Index offset (Hex)	Access	Drive- type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000080	Read	every (IO→NC)	{		STRUCT s. drive-interface	DRIVE-INPUT-STRUCTURE (IO→NC, 40 Byte)*NC-DRIVESTRUCT_IN2*	
			INT32	INC	≥ 0	nInData1	
			INT32	INC	≥ 0	nInData2	
			UINT8	1	≥ 0	nStatus1	
			UINT8	1	≥ 0	nStatus2	
			UINT8	1	≥ 0	nStatus3	
			UINT8	1	≥ 0	nStatus4	
			INT32	INC	≥ 0	nInData3	
			INT32	INC	≥ 0	nInData4	
			INT32	INC	≥ 0	nInData5	
			INT32	INC	≥ 0	nInData6	
			UINT8	1	≥ 0	nStatus5	
			UINT8	1	≥ 0	nStatus6	
			UINT8	1	≥ 0	nStatus7	
			UINT8	1	≥ 0	nStatus8	
			INT32	1	≥ 0	reserved	
			INT32	1	≥ 0	reserved	
			}				

7.8.8 Specification Tables

7.8.8.1 "Index offset" specification for Table parameter (Index group 0xA000 + ID)

Index offset (Hex)	Access	table - type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000001	Read	every	UINT32	1	[1 ... 255]	table ID	
0x00000002	Read	every	UINT8[30+1]	1	30 symbol	table name	
0x00000003	Read	every	UINT32	1	s. ENUM (>0)	table sub type [▸ 498]	
0x00000004	Read	every	UINT32	1	s. ENUM (>0)	table main type [▸ 498]	
0x00000010	Read	every	UINT32	1	[0... 16777216]	number of lines (n)	
0x00000011	Read	every	UINT32	1	[0... 16777216]	number of columns (m)	
0x00000012	Read	every	UINT32	1	≥0	number of total elements (n*m)	
0x00000013	Read	equidistant Tab.	REAL64	e.g. mm	≥0.0	step width (position delta) (equidistant table)	base unit
0x00000014	Read	cyclic Tab.	REAL64	e.g. degree	≥0.0	master period (cyclic table)	base unit
0x00000015	Read	cyclic Tab.	REAL64	e.g. degree	≥0.0	slave difference per master period (cyclic table)	base unit
0x0000001A	Read /Write	"Motion Function"	{			activation mode for online change from table data (only MF)	
			UINT32	ENUM	s. appendix	activation mode: 0: 'instantaneous' (default) 1: 'master cam pos.' 2: 'master' axis pos.' 3: 'next cycle' 4: 'next cycle once' 5: 'as soon as possible' 6: 'off' 7: 'delete queued data'	
			REAL64	e.g. mm	±∞	activation position	
			UINT32	ENUM	s. appendix	master scaling type 0: user defined (default) 1: scaling with auto offset 2: off	
			UINT32	ENUM	s. appendix	slave scaling type 0: user defined (default) 1: scaling with auto offset 2: off	
			}				
0x00000020	Read /Write	every	{			write single value [n,m]:	
			UINT32	1	[0 ... 16777216]	n-th line	
			UINT32	1	[0 ... 16777216]	m-th column	
			REAL64	e.g. mm	±∞	single value	base unit
			}				
0x00000021	ReadWrite	every	*REAL64	e.g. mm	±∞	Read slave position to the given master position (relates only to the "row values" of the table)	

Index offset (Hex)	Access	table - type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000022	ReadWrite	"Motion Function"	**Write**			Read the "Motion Function" as fixed values ("scatter plot")	only line by line possible!(multiple integer)
			{				
			UINT16	1	0/1	Initialisation of data (copy of actual data) ?	
			UINT16	1	Bitmask (≥0)	Select bit mask (number of columns is one column for the masterposition plus number of bits):Bit 0: Pos (Slave) Bit 1: Velo (Slave) Bit 2: Acc (Slave) Bit 3: Jerk (Slave)	
			REAL64	z.B. mm	±∞	Startposition (Master)	
			REAL64	z.B. mm	> 0.0	Inkrement	
			}				
			Read				
			{				
			REAL64[x*m]	z.B. mm	±∞	Generating the values of x rows beginning with the master start position: (x*m)-values (one or more rows)	
			}				
0x00000023	ReadWrite	every	**Write**			Read slave values to given master position (relates only to the "row values" of the table)	
			REAL64	z.B. mm	±∞	master position	
			Read				
			{				
			REAL64	z.B. mm	±∞	slave position	
			REAL64	mm/s	±∞	slave velocity	
			REAL64	mm/s^2	±∞	slave acceleration	
			}				
0x00000050	Read /Write	every	REAL64 [64]	1	±∞	Characteristic table values [▶ 500]	
0x00000050	ReadWrite	every	**Write**			Read the characteristic table values in dependency to the nominal master velocity	
			REAL64 [64]	...	±∞	optional nominal master velocity "fMasterVeloNom" (normed => 1.0 mm/s), the other values are not used	
			Read				
			REAL64 [64]	...	±∞	Read the characteristic table values [▶ 500]	

Index offset (Hex)	Access	table - type	Data type	Phys. unit	Definition range	Description	Remarks
0x00000115	Write	monoton linear, monoton zykl.,	{			set /change the table scaling:	
			REAL64	1	[±1000000.0]	Original wightning of the table	
			REAL64	e.g. mm	[±1000000.0]	Position offset of the master column	
			REAL64	1	[±1000000.0]	Scaling of the master column	
			REAL64	e.g. mm	[±1000000.0]	Position offset of the slave column	
			REAL64	1	[±1000000.0]	Scaling of the slave column	
			REAL64	e.g. mm	[±1000000.0]	lower area boundary (start position)	
			REAL64	e.g. mm	[±1000000.0]	upper area boundary (end position)	
			}				
0x01000000 +n-te start line	Read / Write[<=16777216]	every	{ REAL64[x*m] }	e.g. mm	±∞	read/write x lines from the n-th line: (x*m)-values (one or more lines)value range n: [0 ... 16777216]	only line by line possible! (multiple integer)
0x02000000 +m-te Startspold	Read / Write[<=16777216]	every	{ REAL64[x*n] }	e.g. mm	∞	read/write x columns from m-th column: (x*n)-values (one or more columns)value range m: [0 ... 16777216]	only column by column possible! (multiple integer)
0x05000000 +n-te start line	Read / Write[<=16777216]	"Motion Function"(law of motion)Data:STRUCT[x*m]	{			read/write x lines from the n-th line: (x*m)-values (one or more lines)value range n: [0 ... 16777216]	only line by line possible! (multiple integer)
			UINT32	1		absolute point index (not checked)	
			UINT16	ENUM		function type 1: Polynom 1 15: Polynom 5	
			UINT16	ENUM		point type 0: default 1: ignore	
			INT32	1		relative adressindex to target point (default: 1)	
			REAL64	mm		master position	
			REAL64	mm		slave position	
			REAL64	mm/s		slave velocity	
			REAL64	mm/s^2		slave acceleration	
			REAL64	mm/s^3		slave jerk	
			}				

Index offset (Hex)	Access	table - type	Data type	Phys. unit	Definition range	Description	Remarks
0x06000000 +m-te Start-spold	Read / Write[<=16777216]	"Motion Function"(law of motion)Data:STRUCT[x*n]	{			read/write x columns from m-th column: (x*n)-values (one or more columns)value range m: [0 ... 16777216]	only column by column possible! (multiple integer)
			UINT32	1		absolute point index (not checked)	
			UINT16	ENUM		function type 1: Polynom 1 15: Polynom 5	
			UINT16	ENUM		point type 0: default 1: ignore	
			INT32	1		relative adressindex to target point (default: 1)	
			REAL64	mm		master position	
			REAL64	mm		slave position	
			REAL64	mm/s		slave velocity	
			REAL64	mm/s^2		slave acceleration	
			REAL64	mm/s^3		slave jerk	
			}				

7.8.8.2 "Index offset" specification for Table state (Index group 0xA100 + ID)

Index offset (Hex)	Access	table - type	Data type	Phys. unit	Definition range	Description	Remarks
0x0000000A	Read	every	INT32	1	≥ 0	'User Counter' (number of table user)	Cannot be traced by oscilloscope!

7.8.8.3 "Index offset" specification for Table functions (Index group 0xA200 + ID)

Index offset (Hex)	Access	table - type	Data type	Phys. unit	Definition range	Description	Remarks
0x00010000	Write	every	{			generates table with dimension (n*m):	table types: 1,2,3,4 Dimension: at least 2x1
			UINT32	1	s. ENUM (>0)	table type [▸ 498] (s. appendix)	
			UINT32	1	[2...16777216]	quantitiy of lines	
			UINT32	1	[1...16777216]	quantitiy of columns	
			}				
0x00010001	Write	valve diagram	{			generatesvalve diagram table with dimension (n*m):	table types: 1,3 Dimension: at least 2x1
			UINT32	1	s. ENUM (>0)	table type [▸ 498] (s. appendix)	
			UINT32	1	[2...16777216]	quantitiy of lines	
			UINT32	1	[1...16777216]	quantitiy of columns	
			}				
0x00010010	Write	"Motion Function" (law of motion)	{			generates"Motion Function" table with dimension (n*m):	table types: 3,4 Dimension: at least 2x1
			UINT32	1	s. ENUM (>0)	table type (s. appendix)	
			UINT32	1	[2...16777216]	quantitiy of lines	
			UINT32	1	[1...16777216]	quantitiy of columns	
			}				
0x00020000	Write	every	VOID			deletes table with dimension (n*m)	table types: 1,2,3,4
0x00030000	Write	every	VOID			initialised table Initialisation is no longer needed, because now it happens automatically in the following casesa) by coupling with table b) by selecting the slave position (s. table para.)	

7.8.9 Appendix

Enum Channel types

Define	Channel types
1	Standard
2	Interpreter
3	FIFO
4	Kinematic transformation

Enum Interpreter types

Define	Interpreter types
0	NOT DEFINED
1	reserved
2	DIN 66025 (Siemens dialect)

Enum Interpreter Operation modes

Define	interpreter/channel operation mode
0x0	Default (deactivates the other modes)
0x1	Single block mode in the NC core (Block execution task/SAF)
0x1000	reserved
0x2000	reserved
0x4000	Single block mode in the interpreter

Enum Interpolation load log mode

Define	Load log mode
0	Loader log off
1	Source only
2	Source & Compiled

Enum Interpolation Trace mode

Define	Trace mode
0	Trace off
1	Trace line numbers
2	Trace Source

Enum Interpreter state

moved to: System Manager interface for the interpreter - interpreter element

Enum Group types

Define	Group types
0	NOT DEFINED
1	PTP-Group + x Slave
2	1D-Group + x Slave
3	2D-Group + x Slave
4	3D-Group + x Slave
5	High/low speed + x Slave
6	Low cost stepper motor (dig. IO) + x Slave
7	Table Group + x Slave
9	Encoder Group + x Slave
11	FIFO Group + x Slave
12	Kinematic Transformation Group + x Slave

Enum Curve velocity reduction method

moved to: System Manager interface for the interpreter - group element

Enum Axis types

Define	Axis types
0	NOT DEFINED
1	Continuous axis (Servo)
2	Discrete axis (high/low speed)
3	Continuous axis (stepper motor)

Enum Stepper motor operation mode

Define	Stepper motor operation mode
0	NOT DEFINED
1	2-phase excitation (4 cycles)
2	1-2-phase excitation (6 cycles)
3	Power section

Enum Override types for PTP axes (velocity override)

Define	Override types
1	Reduced Old variant, replaced by "(3) Reduced (iterated)"
2	Original Old variant, replaced by "(4) Original (iterated)"
3	Reduced (iterated) Default value: the override value is related to the velocity which is internally reduced in a special case. This results in a directly proportional velocity (=> linear relationship) for the entire override range from 0 to 100%.
4	Original (iterated) The override value is always referred to the velocity programmed by the user. If this velocity cannot be driven, however, then a maximum override value results from which no higher velocity can be reached (=> limitation).

Enum Group/axis start types

Define	Group/axis start types
0	NOT DEFINED
1	Absolute start
2	Relative start
3	Continuous start positive
4	Continuous start negative
5	Modulo start (OLD)
261	Modulo start on the shortest distance
517	Modulo start in positive direction (with modulo tolerance window)
773	Modulo start in negative direction (with modulo tolerance window)
4096	Stop and lock (axis locked for motion commands)
8192	Halt (without motion lock)

Enum Command buffer types (buffer mode) for universal axis start (UAS)

Define	Buffer mode
0	ABORTING (default) (instantaneous, aborts current movement and deletes any buffered commands)
1	BUFFERED (stored in command buffer to be executed after an active movement)
18	BLENDING LOW (buffered, no stop, runs through intermediate target position at the lowest velocity of two commands)
19	BLENDING PREVIOUS (buffered, no stop, runs through intermediate target position at the velocity of the active command)
20	BLENDING NEXT (buffered, no stop, runs through intermediate target position at the velocity of the buffered command)
21	BLENDING HIGH (buffered, no stop, runs through intermediate target position at the highest velocity of two commands)

Enum End position types (new end position)

Define	End position types
0	NOT DEFINED
1	Absolute position
2	Relative position
3	Continuous position positive
4	Continuous position negative
5	Modulo position

Enum Command types for new end position with new velocity (new end position and/or new velocity)

Define	Command types for new end position with new velocity
0	NOT DEFINED
1	Position (instantaneous)
2	Velocity (instantaneous)
3	Position and velocity (instantaneous)
9	Position (switching position)
10	Velocity (switching position)
11	Position and velocity (switching position)

Enum Actual position types (set actual position)

Define	Actual position types
0	NOT DEFINED
1	Absolute position
2	Relative position
5	Modulo position

Enum Compensation types (section compensation or superimposed)

Define	Compensation types
0	NOT DEFINED
1	VELOREDUCTION_ADDITIVEMOTION The max. velocity VelocityDiff is reduced. The path over which the compensation trip is effective consists of length + distance.
2	VELOREDUCTION_LIMITEDMOTION The max. velocity VelocityDiff is reduced. The path over which the compensation trip is effective is defined by the Length parameter.
3	LENGTHREDUCTION_ADDITIVEMOTION The max. available path is reduced and consists of length + distance. The system tries to utilize the max. veloc. VelocityDiff.
4	LENGTHREDUCTION_LIMITEDMOTION The max. available path is reduced and is limited by the Length parameter. The system tries to utilize the max. veloc. VelocityDiff.

Enum Slave types

Define	Slave types
0	NOT DEFINED
1	Linear
2	Flying saw (velocity, jerk restricted profile)
3	Flying saw (position and velocity, jerk restricted profile)
5	Synchronization generator (velocity, jerk restricted profile)
6	Synchronization generator (position and velocity, jerk restricted profile)
10	Tabular
11	Multi-tabular
13	'Motion Function' (MF)
15	Linear with cyclic gearing factor change (ramp filter for acceleration limits)
100	Specific

Enum Slave decoupling types (for subsequent axis command)

Define	Slave decoupling types (for subsequent axis command)
0	Stop, E-stop or P-stop (default) (STOP)
1	Oriented stop (O-stop) (ORIENTEDSTOP)
2	Reduce any acceleration to 0 (force-free) and continue to endless target position (ENDLESS)
3	Continue to endless target position at new requested velocity (ENDLESS_NEWVELO)
4	New end position (NEWPOS)
5	New end position and new requested velocity (NEWPOSANDVELO)
6	Logical decoupling and stopping of axis immediately without velocity ramp (INSTANTANEOUSSTOP)

Enum Controller types

Define	Controller types
0	NOT DEFINED
1	P-controller (standard) (Position)
2	PP-controller (with ka) (Position)
3	PID-controller (with ka) (Position)
5	P-controller (Velocity)
6	PI controller (Velocity)
7	High/low speed controller (Position)
8	Stepper motor controller (Position)
9	SERCOS controller (Position in the drive)
10	RESERVED
11	RESERVED
12	RESERVED
13	RESERVED
14	TCom Controller (Soft Drive) (Position in the drive)

Enum Controller Observer mode

Define	Controller observer mode
0	No observer active (default)
1	"Luenberger" observer (classic observer design)

Enum Encoder types

Define	Encoder types
0	NOT DEFINED
1	Simulation Encoder (Incremental)
2	M3000 Encoder (Multi/Single-Turn) (Absolute)
3	M31x0 / M2000 Encoder (Incremental)
4	MDP 511 Encoder: EL7041, EL7342, EL5101, EL5151, EL2521, EL5021, IP5101 (Incremental)
5	MDP 500/501 Enc.: EL5001, IP5009, KL5001 (SSI) (Absolute)
6	MDP 510 Encoder: KL5051, KL2502-30K Encoder (BiSSI) (Incremental)
7	KL30xx Encoder (Analog) (Absolute)
8	SERCOS and EtherCAT SoE (Position) (Incremental)
9	SERCOS and EtherCAT SoE (Position and velocity) (Incremental)
10	Binary encoder (0/1) (Incremental)
11	M2510 Encoder (Absolut)
12	FOX50 Encoder (Absolute)
14	AX2000 (Lightbus) (Incremental)
15	Provi-Drive MC (Simodrive 611U) (Incremental)
16	Universal encoder (variable bit mask) (Incremental)
17	NC rear panel (Incremental)
18	Special CANopen type (e.g. Lenze Drive 9300) (Incremental)
19	MDP 513 (DS402): CANopen and EtherCAT CoE (AX2xx-B1x0/ B510, EL7201) (Incremental)
20	AX2xx-B900 (Ethernet) (Incremental)
21	KL5151 Encoder (Incremental)
24	IP5209 Encoder (Incremental)
25	KL2531/KL2541 Encoder (Stepper Motor) (Incremental)
26	KL2532/KL2542 Encoder (DC motor), KL2535/KL2545 (PWM current terminal) (Incremental)
27	Time base encoder (Time Base Generator) (Incremental)
28	TCom Encoder (Soft Drive) (Incremental)

Enum Encoder mode

Define	Encoder mode
0	NOT DEFINED
1	Determination of position
2	Determination of position and velocity
3	Determination of position, velocity and acceleration

Enum encoder evaluation direction (log. counting direction)

Define	Encoder evaluation direction (log. counting direction)
0	Evaluation in positive and negative counting direction (default configuration, i.e. compatible with the previous state)
1	Evaluation only in positive counting direction
2	Evaluation only in negative counting direction
3	Evaluation neither in positive nor in negative counting direction (evaluation blocked)

NOTE! Not for all encoder types; only for KL5101, KL5151, KL2531, KL2541, IP5209, Universal encoder, etc.

	Encoder types		
Encoder evaluation direction (log. counting direction)	KL5101, ...	Universal Encoder	other types
0: positive and negative	√	√	—
1: only positive	√	√	—
2: only negative	√	√	—
3: blocked	√	√	—

Enum Encoder sign interpretation (data type)

Define	Sign interpretation (data type) of the encoder actual increments
0	NOT DEFINED (default configuration, i.e. compatible with the previous state)
1	UNSIGNED: unsigned interpretation of the encoder actual increments
2	SIGNED: signed interpretation of the encoder actual increments

NOTE! For KL30xx/KL31xx only for the time being

Enum Encoder absolute dimensioning system

Define	Encoder absolute dimensioning system
0	INC: Incremental absolute dimension system with underflow and overflow offset (default, i.e. compatible with the previous state)
1	ABS: Absolute dimension system without underflow and overflow offset (no underflow or overflow of the encoder allowed)
2	ABS MODULO: Conditionally absolute dimension system, since it has underflow and overflow offset (absolute value that modulo (endless) continues)

NOTE! Not for all encoder types; only for Profi Drive MC, M3000, KL5001/EL5001, IP5009, SERCOS, UNIVERSAL, etc.

Enum Reference mode for incremental encoder

Define	Reference mode for incremental encoder
0	NOT DEFINED (default configuration, i.e. compatible with the previous state)
1	Latch event: shutdown of the PLC cam (falling edge)
2	Latch event: Hardware sync pulse (zero track)
3	Latch event: External hardware latch with rising edge (measuring probe or, respectively, measurement on the fly with rising edge)
4	Latch event: External hardware latch with negative edge (measuring probe or flying measurement with neg. edge)
5	Latch event: Synthetically emulated software sync pulse (software zero track); REQUIREMENT: absolute per motor revolution, e.g. resolver!
6	Latch event: Hardware latch event defined in the drive with rising edge (e.g. for SoftDrive) (NEW)
7	Latch event: Hardware latch event defined in the drive with falling edge (e.g. for SoftDrive) (NEW)
20	User-specific implementation of the referencing (PLC code): User request is signaled to the PLC with the ApplicationRequest-Bit (NEW)

	: Latch event					
Encoder types	**0: not defined**	**1: PLC cam (neg. edge)**	**2: Hardware sync pulse (zero/ C-track)**	**3: External hardware latch with pos. edge**	**4: External hardware latch with neg. edge**	**5: Software sync pulse (software zero track)**
AX2xxx-B200(Lightbus)	—	√	√	√	√	√ (resolver only)
AX2xxx-B510(CANopen)	—	√	—	—	—	√ (only resolver) (see "Reference mask" parameter)
AX2xxx-B1x0(EtherCAT)	—	√	√	√	√	√ (only resolver) (fixed 20-bit)
AX2xxx-B900(Ethernet)	—	√	√	√	√	√ (resolver only)
Sercos	—	√	√	√ (AX5xxx specific implemented)	√	√ (see "Reference mask" parameter)
Profi Drive	—	√	√	√	√	√
KL5101 IP5109	—	√	√	√	√	√
KL5111	—	√	√	—	—	√
KL5151	—	√	√	√	√	√ (not meaningful)
IP5209	—	√	√	—	—	√ (not meaningful)
CANopen (e.g. Lenze)	—	√	—	√ (input E1)	√ (input E2)	√ (resolver only) (fixed 16-bit)
other types	—	—	—	—	—	—

Enum Drive types

Define	Drive types
0	NOT DEFINED
1	Analog Servo Drive: M2400 DAC 1 (Analog)
2	Analog Servo Drive: M2400 DAC 2 (Analog)
3	Analog Servo Drive: M2400 DAC 3 (Analog)
4	Analog Servo Drive: M2400 DAC 4 (Analog)
5	MDP 252 Drive: Analog Servo Drive: KL4xxx, KL2502-30K (Analog)
6	MDP 252 Drive: Analog Servo Drive (non-linear): KL4xxx, KL2502-30K (Analog)
7	High/low speed drive (Digital)
8	Stepper motor drive (Digital)
9	SERCOS-Drive (Digital)
10	MDP 510 Drive: KL5051 (BiSSI-Interface) (Digital)
11	AX2000 (Lightbus) (Digital)
12	Provi-Drive MC (Simodrive 611U) (Digital)
13	Universal Drive (Analog)
14	NC rear panel (Analog)
15	Special CANopen type (e.g. Lenze Drive 9300) (Digital)
16	MDP 742 (DS402): CANopen and EtherCAT CoE (AX2xx-B1x0/B510) (Digital)
17	AX2xx-B900 Drive (Ethernet) (Digital)
20	KL2531/KL2541 Encoder (Stepper Motor) (Digital)
21	KL2532/KL2542 Encoder (DC motor), KL2535/KL2545 Encoder (PWM current terminal) (Digital)
22	TCom Drive (Soft Drive) (Digital)
23	MDP 733 Drive: Profile MDP 733 (EL7332, EL7342, EP7342) (Digital)
24	MDP 703 Drive: Profile MDP 703 (EL7031, EL7041, EP7041) (Digital)

Enum Drive-Output-Start types

Define	Enum Drive-Output-Start types
0	NOT DEFINED
1	Output value in percent
2	Output as velocity, e.g. m/min

Enum Drive Operation Mode

Define	Drive Operation Mode (generic operation modes independent from drive)
0	DEFAULT Mode (reactivates the NC default operation mode if mode is known)
1 (standard type)	torque control
2 (standard type)	velocity control with feedback 1
3 (standard type)	velocity control with feedback 2
4 (standard type)	position control with feedback 1 (lag less)
5 (standard type)	position control with feedback 2 (lag less)
17 (oversampling type)	torque control using dynamic container
18 (oversampling type)	velocity control with feedback 1 using dynamic container
19 (oversampling type)	velocity control with feedback 2 using dynamic container
20 (oversampling type)	position control with feedback 1 (lag less) using dynamic container
21 (oversampling type)	position control with feedback 2 (lag less) using dynamic container
100 (Sercos/SoE specific)	Sercos/SoE primary operation mode 0 (s. S-0-0032)
101 (Sercos/SoE specific)	Sercos/SoE secondary operation mode 1 (s. S-0-0033)
102 (Sercos/SoE specific)	Sercos/SoE secondary operation mode 2 (s. S-0-0034)
103 (Sercos/SoE specific)	Sercos/SoE secondary operation mode 3 (s. S-0-0035)
104 (Sercos/SoE specific)	Sercos/SoE secondary operation mode 4 (s. S-0-0284)
105 (Sercos/SoE specific)	Sercos/SoE secondary operation mode 5 (s. S-0-0285)
106 (Sercos/SoE specific)	Sercos/SoE secondary operation mode 6 (s. S-0-0286)
107 (Sercos/SoE specific)	Sercos/SoE secondary operation mode 7 (s. S-0-0287)

Enum Moving phases / Movement state for master axes

Define	Moving phases / Movement state (distinction between internal and external setpoint generation)
Internal setpoint generation	
0	Setpoint generator not active (INACTIVE)
1	Setpoint generator active (RUNNING)
2	Velocity override is zero (OVERRIDE_ZERO)
3	Constant velocity (PHASE_VELOCONST)
4	Acceleration phase (PHASE_ACCPOS)
5	Deceleration phase (PHASE_ACCNEG)
External setpoint generation:	
41	External setpoint generation active (EXTSETGEN_MODE1)
42	Internal and external setpoint generation active (EXTSET-GEN_MODE2)

Enum Moving phases / Movement state for slave axes

Define	Moving phases / Movement state
0	Slave generator not active (INACTIVE)
11	Slave is in a movement pre-phase (PRE-PHASE)
12	Slave is synchronizing (SYNCHRONIZING)
13	Slave is synchronized and moves synchronously (SYNCHRON)

NOTE! Only for slaves of the type synchronization generator for the time being

Enum Table main types

Define	Table main types
1	(n*m) Cam plate tables (Camming)
10	(n*m) Characteristic curves tables (Characteristics) (e.g. hydraulic valve characteristic curves) Only non-cyclic table sub-types (1, 3) are supported!
16	(n*m) "Motion Function" tables (MF) Only non-equidistant table sub-types (3, 4) are supported!

Enum Table sub-types

Define	Table sub types
1	(n*m) Table with equidistant master positions and no cyclic continuation of the master profile (equidistant linear)
2	(n*m) Table with equidistant master positions and cyclic continuation of the master profile (equidistant cyclic)
3	(n*m) Table with non-equidistant, but strictly monotonously increasing master positions and a non-cyclic continuation of the master profile (monotonously linear)
4	(n*m) Table with non-equidistant, but strictly monotonously increasing master positions and a cyclic continuation of the master profile (monotonously cyclic)

Enum Table interpolation types

Define	Table interpolation types between the reference points
0	Linear interpolation (NC_INTERPOLATIONTYPE_LINEAR) (Standard)
1	4-point interpolation (NC_INTERPOLATIONTYPE_4POINT) (for equidistant table types only)
2	Cubic spline interpolation of all reference points ("global spline") (NC_INTERPOLATIONTYPE_SPLINE
3	Sliding cubic spline interpolation via n interpolation points ("local spline") (NC_INTERPOLATIONTYPE_SLIDINGSPLINE)

Enum table operation mode

Define	Table operation mode for adding, exchange and removal of tables
0	(default)
1	Additive – addition of a further table
2	Exchange – replacement of an existing table with a new table
3	Remove – removal of an existing table

Structure of tabular (cam) coupling informationen

Tables		(CAM) Coupling information
nTableID;	1.	cam table ID
nTableMainType;	2.	e.g. CAMMING, CHARACTERISTIC, MO-TIONFUNCTION
nTableSubType;	3.	e.g. EQUIDIST_LINEAR, EQUIDIST_CY-CLE, NONEQUIDIST_LINEAR, NONEQUIDIST_CYCLE
nInterpolationType;	4.	e.g. LINEAR, 4POINT, SPLINE
nNumberOfRows;	5.	number of rows/elements
nNumberOfColumns;	6.	number of columns
fMasterCamStartPos	7.	master camming start position (first point in tabular)
fSlaveCamStartPos	8.	slave camming start position (first point in tabular)
fRawMasterPeriod;	9.	master period/cycle (raw value, not scaled)
fRawSlaveStroke;	10.	slave difference per master period/cycle (raw value, not scaled)
fMasterAxisCouplingPos	11.	total absolute master offset of cam origin when slave has been coupled
fSlaveAxisCouplingPos	12.	total absolute slave offset of cam origin when slave has been coupled
nMasterAbsolute	13.	master absolute position (0/1)
nSlaveAbsolute	14.	slave absolute position (0/1)
fMasterOffset;	15.	total master offset
fSlaveOffset;	16.	total slave offset
fMasterScaling;	17.	total master scaling
fSlaveScaling;	18.	total slave scaling
fSumOfSlaveStrokes	19.	sum of the slave srokes up to “fActualMas-terAxisPos”
fSumOfSuperpositionDistance	20.	sum of superposition distance (position compensation offset)
fActualMasterAxisPos;	21.	actual master axis setpos (absolute)
fActualSlaveAxisPos;	22.	actual slave axis setpos (absolute)
fActualMasterCamPos;	23.	actual master cam setpos
fActualSlaveCamPos;	24.	actual master cam setpos
nSlaveStateDWord	25.	slave state DWORD (s. AxisRef)
. . .	. . .	. . .

Structure of the characteristic values

Characteristic values		
fMasterVeloNom;	1.	master nominal velocity (standardized: => 1.0)
fMasterPosStart;	2.	master start position
fSlavePosStart;	3.	slave start position
fSlaveVeloStart;	4.	slave start velocity
fSlaveAccStart;	5.	slave start acceleration
fSlaveJerkStart;	6.	slave start jerk
fMasterPosEnd;	7.	master end position
fSlavePosEnd;	8.	slave end position
fSlaveVeloEnd;	9.	slave end velocity
fSlaveAccEnd;	10.	slave end acceleration
fSlaveJerkEnd;	11.	slave end jerk
fMPosAtSPosMin;	12.	master pos. at slave min. position
fSlavePosMin;	13.	slave minimum position
fMPosAtSVeloMin;	14.	master pos. at slave min. velocity
fSlaveVeloMin;	15.	slave minimum velocity
fMPosAtSAccMin;	16.	master pos. at slave min. acceleration
fSlaveAccMin;	17.	slave minimum acceleration
fSVeloAtSAccMin;	18.	slave velocity at slave min. acceleration
fSlaveJerkMin;	19.	slave minimum jerk
fSlaveDynMomMin;	20.	slave minimum dynamic momentum (NOT SUPPORTED YET !)
fMPosAtSPosMax;	21.	master pos. at slave max. position
fSlavePosMax;	22.	slave maximum position
fMPosAtSVeloMax;	23.	master pos. at slave max. velocity
fSlaveVeloMax;	24.	slave maximum velocity
fMPosAtSAccMax;	25.	master pos. at slave max. acceleration
fSlaveAccMax;	26.	slave maximum acceleration
fSVeloAtSAccMax;	27.	slave velocity at slave max. acceleration
fSlaveJerkMax;	28.	slave maximum jerk
fSlaveDynMomMax;	29.	slave minimum dynamic momentum (NOT SUPPORTED YET !)
fSlaveVeloMean;	30.	slave mean absolute velocity
fSlaveAccEff;	31.	slave effective acceleration
nCamTableID;	32.	Cam table ID
nNumberOfRows;	33.	Number of rows/entries e.g. number of points
nNumberOfColums;	34.	Number of columns (typically1 or 2)
nCamTableType;	35.	cam table type (10=EQUIDIST, 11=NONEQUIDIST, 22=MOTIONFUNC, 23=CHARACTERISTIC)
nPeriodic;	36.	linear or cyclic/periodic
nReserved	37.	reserved

Enum Axis control loop switch types

Define	Axis control loop switch types
0	NOT DEFINED
1	Simple switching (similar to an axis reset) (STANDARD)
2	Switching/synchronization by means of I/D-part of the controller to an internal initial value (jerk-free/smooth)
3	Switching/synchronization by means of I/D-part of the controller to a parameterizable initial value