Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BS®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT technology is patent protected, in particular by the following applications and patents:
with corresponding applications or registrations in various other countries.

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation Gmbh, Germany

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.
The reproduction, distribution and utilisation of this document as well as the communication of its contents to others without express authorisation are prohibited.
Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.
General and safety instructions

Icons used and their meanings

This documentation uses the following icons next to the safety instruction and the associated text. Please read the (safety) instructions carefully and comply with them at all times.

Icons in explanatory text

1. Indicates an action.
   ⇐ Indicates an action statement.

DANGER

Acute danger to life!
If you fail to comply with the safety instruction next to this icon, there is immediate danger to human life and health.

CAUTION

Personal injury and damage to machines!
If you fail to comply with the safety instruction next to this icon, it may result in personal injury or damage to machines.

NOTE

Restriction or error
This icon describes restrictions or warns of errors.

Tips and other notes
This icon indicates information to assist in general understanding or to provide additional information.

General example
Example that clarifies the text.

NC programming example
Programming example (complete NC program or program sequence) of the described function or NC command.

Specific version information
Optional or restricted function. The availability of this function depends on the configuration and the scope of the version.
# Table of contents

Notes on the documentation ........................................................................................................ 3

General and safety instructions .................................................................................................... 4

1 Kinematic transformation (TRAFO) .......................................................................................... 8

1.1 Introduction ............................................................................................................................. 8

1.2 Coordinate systems .................................................................................................................. 10

1.3 Position offsets ........................................................................................................................ 12

1.4 Modulo setting of axes ............................................................................................................ 14

2 Interfacing transformation via TcCOM .................................................................................... 15

2.1 TcCOM transformation interface .......................................................................................... 16

2.1.1 Transformation methods ..................................................................................................... 16

2.1.2 Working (instance data) of the transformation .................................................................. 17

2.1.3 Configuring and registering the transformation with the CNC ........................................ 22

3 Parametrisation ......................................................................................................................... 24

3.1 CNC parameters: Channel and tool ....................................................................................... 24

3.1.1 Transformation parameters of the tool ............................................................................. 25

3.1.2 Channel parameter ............................................................................................................ 25

3.2 TcCOM parameters ................................................................................................................ 27

4 Error handling and diagnosis ................................................................................................... 29

4.1 Error message ......................................................................................................................... 29

4.2 Diagnostic data ....................................................................................................................... 31

5 Working data of the transformation .......................................................................................... 32

6 Concatenating transformations, multistep transformations .................................................... 33

7 Generating a transformation ....................................................................................................... 35

7.1 System requirements ............................................................................................................... 35

7.2 Generation process .................................................................................................................. 35

7.2.1 Create project and transformation ..................................................................................... 36

7.2.2 Integrate transformation ..................................................................................................... 44

7.2.3 Debugging the transformation ............................................................................................ 46

7.2.4 Source code extension/encoding ......................................................................................... 49

7.3 Differences between extended transformation / standard transformation .......................... 50

8 Parameter ................................................................................................................................... 51

9 Additional options of extended transformation ....................................................................... 52

9.1 Version identifier of transformation interface ......................................................................... 52

9.2 Rotation sequence .................................................................................................................... 52

9.3 Modulo handling of axis positions ......................................................................................... 53

9.4 Use of extended parameters ................................................................................................... 54

9.5 Use of extended options .......................................................................................................... 55

10 Display the position of the additive transformation .................................................................. 58

Index ............................................................................................................................................. 59
# List of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Example of a kinematic transformation</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>Function of kinematic transformation</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>Coordinate systems in detail</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>Coordinate systems in detail</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>Access to kinematic parameters</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>Modulo handling of an axis</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>Interfacing kinematic transformation via TcCOM in TwinCAT3</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>Dimensioning the input and output coordinates</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>Kinematic transformation when intersection calculation is active</td>
<td>20</td>
</tr>
<tr>
<td>10</td>
<td>Kinematic transformation when intersection calculation is inactive</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>Identification of the transformation callers</td>
<td>21</td>
</tr>
<tr>
<td>12</td>
<td>Inserting a transformation module</td>
<td>23</td>
</tr>
<tr>
<td>13</td>
<td>Configuring kinematic transformation via TcCOM in TwinCAT3</td>
<td>23</td>
</tr>
<tr>
<td>14</td>
<td>Transformation parameters of the tool</td>
<td>25</td>
</tr>
<tr>
<td>15</td>
<td>Channel transformation parameter</td>
<td>26</td>
</tr>
<tr>
<td>16</td>
<td>Transformation parameters via TcCOM</td>
<td>27</td>
</tr>
<tr>
<td>17</td>
<td>TMC Editor</td>
<td>28</td>
</tr>
<tr>
<td>18</td>
<td>Concatenating kinematic transformations</td>
<td>33</td>
</tr>
<tr>
<td>19</td>
<td>Create a project</td>
<td>36</td>
</tr>
<tr>
<td>20</td>
<td>Generate CNC configuration</td>
<td>37</td>
</tr>
<tr>
<td>21</td>
<td>Create channel</td>
<td>38</td>
</tr>
<tr>
<td>22</td>
<td>Create axes</td>
<td>39</td>
</tr>
<tr>
<td>23</td>
<td>Create TwinCAT driver project</td>
<td>40</td>
</tr>
<tr>
<td>24</td>
<td>Create transformation class</td>
<td>41</td>
</tr>
<tr>
<td>25</td>
<td>Name transformation class</td>
<td>42</td>
</tr>
<tr>
<td>26</td>
<td>Create driver</td>
<td>43</td>
</tr>
<tr>
<td>27</td>
<td>Integrate TcCOM object</td>
<td>44</td>
</tr>
<tr>
<td>28</td>
<td>Properties of the TcCOM object</td>
<td>45</td>
</tr>
<tr>
<td>29</td>
<td>Parameterise the transformation in the channel parameter list</td>
<td>46</td>
</tr>
<tr>
<td>30</td>
<td>Switch over to debug configuration</td>
<td>46</td>
</tr>
<tr>
<td>31</td>
<td>Activate real-time debugging</td>
<td>47</td>
</tr>
<tr>
<td>32</td>
<td>Breakpoint in the transformation</td>
<td>48</td>
</tr>
<tr>
<td>33</td>
<td>Setting the constructor after generation using TwinCAT3 templates</td>
<td>49</td>
</tr>
<tr>
<td>34</td>
<td>Adapted constructor due to high number of axes</td>
<td>49</td>
</tr>
<tr>
<td>35</td>
<td>Adapting the number of inputs/outputs</td>
<td>55</td>
</tr>
<tr>
<td>36</td>
<td>Interfaces for adaptation to various callers</td>
<td>57</td>
</tr>
<tr>
<td>37</td>
<td>Displaying the additive transformation position</td>
<td>58</td>
</tr>
</tbody>
</table>
1 Kinematic transformation (TRAFO)

1.1 Introduction

This function is available as of TwinCAT 3 and higher.

Definition of McCOM

Motion Control Component Object Model is a binary interface standard for machine tool controllers. Based on the Microsoft COM standard, McCOM defines how various software components developed and generated independently cooperate in real time.

Figure 1: Example of a kinematic transformation
Kinematic structure

To simplify workpiece programming, the kinematic transformation encapsulates the machine’s kinematic structure and abstracts the motions in a simple Cartesian coordinate system.

Licensing note

Please note that the use of the transformation interface is an additional option and subject to the purchase of a license.

Forward/backward transformation

Depending on the machines' kinematics, the CNC needs the transformation between axis coordinates and programming coordinates to calculate motions. With the aid of this kinematic transformation, the coordinates of the NC program (forward transformation, ACS -> MCS) are calculated from the physical positions of the axes. Conversely, backward transformation calculates the axis positions from the programmed NC positions (MCS -> ACS).

Selecting/deselecting

Transformation is selected by means of an NC command in the NC program, for example.

```
N10 #KIN ID[65]
N20 #TRAFO ON
N30 X100...
N90 #TRAFO OFF
```

Figure 2: Function of kinematic transformation

Extendibility

You can create a user-defined transformation and make it available to the CNC under a selected number (ID). The range [65; 69] is provided as the transformation numbers for this operation.
1.2 Coordinate systems

WCS: workpiece coordinate system

PCS: part coordinate system

MCS: machine coordinate system

ACS: axes coordinate system

Figure 3: Coordinate systems in detail
Subroutine Coordinate System PCS

This coordinate system is used for geometry description based on the DIN 66025 programming syntax language. The data in a subroutine represents program coordinates.

Workpiece Coordinate System WPCS

This coordinate system refers to a fixed point of the workpiece. The coordinate description of the workpiece refers to this system.

The workpiece coordinate system without offsets is used as the basic coordinate system (WPCS₀).

Machine Coordinate System MCS

The machine coordinate system represents an abstract coordinate system that is defined by the machine manufacturer. All other coordinate systems refer to this system.

If the machine has no Cartesian axis structure (e.g. robot), the machine coordinate system is only virtual.

Axis Coordinate System ACS

Each axis has a separate coordinate system. Each axis is either fitted to the machine bed itself or to another axis. This means that the machine bed or the associated axis forms the basis. Therefore, the axis coordinate system is defined related to its fixed point.
1.3 Position offsets

Offset management in the PCS – WPCS transformations

If an offset needs to be activated between the programmed coordinates PCS and the actual physical axis positions ACS, you have a number of options as user.

CNC-programmed offsets (G54, G92, etc.) are taken into consideration between PCS and WPCS.

WPCS – ACS

If the kinematics of a machine require offsets on the axis coordinate system, this is taken into consideration in the transformation.

Figure 4: Coordinate systems in detail
Use of axis-specific offsets in kinematic transformation

N010 G54 ; activate zero point offsets at ACS=PCS level
N020 G0 X0 Y0 Z0 B0 C0 ; move to zero at PCS level
; ...
N090 G53 ; deactivate PCS offsets
; ...
N120 V.G.KIN[65].PARAM[40] = <x_offset in [0.1 µm]>
N130 V.G.KIN[65].PARAM[43] = <b_offset in [0.0001 degree]>
N140 V.G.KIN[65].PARAM[44] = <c_offset> in [0.0001 degree]

N200 #KIN ID[65] ; select kinematic type
N210 #TRAFO ON ; ACS offsets are considered inside transformation
N220 G01 X100 C90
; ...
N240 G92 X400 C180 ; activate additional offset at PCS level
N250 G01 X12 C0
; ...
N340 G56 ; activate additional offset at PCS level
N350 G01 X2 C50
; ...
N999 M30

Access to kinematic parameters

If kinematic parameters are initialised in the CNC program, they are forwarded to the forward/backward algorithms as transformation input parameters (the parameter index used is transformation-specific).

Figure 5: Access to kinematic parameters
### 1.4 Modulo setting of axes

**MCS – ACS**

Depending on the axis properties, the kinematic transformation must define the modulo calculation of the positions. Modulo handling within the transformation must use the same modulo interval as the CNC caller function.

The specified MCS modulo setting is automatically adopted by the CNC caller functionality.

MCS linear / mod[-180;180]

The specified ACS modulo setting is used for a plausibility check. The CNC checks whether the setting matches the axis property configured.

ACS linear / mod[-180;180] / mod[0;360]

![Diagram](image)

**Figure 6:** Modulo handling of an axis
2 Interfacing transformation via TcCOM

In TwinCAT 3, transformation can be interfaced to the CNC via the TcCOM infrastructure.

Figure 7: Interfacing kinematic transformation via TcCOM in TwinCAT3

**NOTE**

The transformation is used in different "timing" phases of NC program execution within one NC channel, possibly simultaneously. This is why the kinematic transformation must be created with reentrant capability and may not use any global data.

**NOTE**

Concatenation of the forwards and backward transformation must again result in the identical starting position. The positions transferred are supplied in [0.1 um]. The transformation results must be available in this resolution range.
2.1 TcCOM transformation interface

TcCOM – TwinCAT Component Object Model

Further information on the TcCOM concept is contained in the TwinCAT3 Help.

2.1.1 Transformation methods

The following methods must be implemented when creating a transformation (TcNcKinematicsInterfaces.h).

- virtual HRESULT TCOMAPI Forward (PTcCncTrafoParameter p)=0;
- virtual HRESULT TCOMAPI Backward (PTcCncTrafoParameter p)=0;
- virtual HRESULT TCOMAPI TrafoSupported (PTcCncTrafoParameter p, bool fwd)=0;
- virtual HRESULT TCOMAPI GetDimensions (PULONG pForwardInput, PULONG pForwardOutput)=0;

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward</td>
<td>Transformation of the axis positions into the programming coordinate system.</td>
<td>PTcCncTrafoParameter *p Current parameters of the transformation</td>
</tr>
<tr>
<td>Backward</td>
<td>Transformation of the programming coordinates into the axis coordinate system.</td>
<td>PTcCncTrafoParameter *p Current parameters of the transformation</td>
</tr>
<tr>
<td>GetDimension</td>
<td>When the transformation is selected, a configuration request (required axis numbers) is executed once.</td>
<td>ULONG * pForwardInput Number of forward transformation input coordinates (= number of reverse transformation output coordinates)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ULONG * pForwardInput Number of forward transformation output coordinates (= number of reverse transformation input coordinates)</td>
</tr>
<tr>
<td>TrafoSupported</td>
<td>Initialising the transformation and requesting options</td>
<td>PTcCncTrafoParameter *p Current parameters of the transformation bool fwd</td>
</tr>
</tbody>
</table>

The corresponding member variables must be initialised in the constructor of the "ITcCncTrafo" class to dimension the input and output coordinates.

Figure 8: Dimensioning the input and output coordinates

```cpp
// Constructor
ChyKinTrafo::ChyKinTrafo(): m_forwardNbrIn[5], m_forwardNbrOut[5]
{
```
### 2.1.2 Working (instance data) of the transformation

#### 2.1.2.1 Basic working data: TcNcTrafoParameter

**Parameters of the methods**

**Type** = EcNcTrafoParameter_Base

The parameters for the individual methods are passed on in encapsulated form via the following structure `TcNcTrafoParameter` (TcNcKinematicsInterfaces.h).

```c
EcNcTrafoParameter  type;
ULONG dim_i;        // dim of input vectors (i, d_i, dd_i)
ULONG dim_o;        // dim of output vectors (o, d_o, dd_o, torque)
ULONG dim_para;     // dim of additional parameter (para)

const double* i;   // input values parameter (dim_i)
const double* d_i;
const double* dd_i;

double* o;         // output values parameter (dim_i)
double* d_o;
double* dd_o;

double* torque;
const double* para; // additional parameter (dim_p)

double payload;    // weight in kg
double tool_len;   // actual tool length in [mm]
```

**Note:**

The CNC does not use the variables in italics.
2.1.2.2 Extended working data: TcNcTrafoParameterExtCnc

Parameters of the methods

Type = EcNcTrafoParameter_ExtCnc

The parameters for the individual methods are transferred via the following extended structure TcCnCTrafoParameter. The data structure provided by the CNC is identified by this parameter type.

Type = EcNcTrafoParameter_ExtCnc

struct TcCncTrafoParameter : public TcNcTrafoParameter, TcCncParam

unsigned short kin_id; // in: used kinematic ID
unsigned long control; // in: control trafo calculation, e.g. EcCncTrafoCtrl_cartesianTrafoInactive
EcCncTrafoOption ret_option; // out: select option of transformation during TrafoSupported()
TcCncVersion CncInterfaceVersion; // Interface version TcCncVersionMajor.TcCncVersionMinor

// orientation
EcCnc_TrafoOriModeActual actual_orientation_mode; // Treatment of orientation, actual rotation sequence
EcCnc_TrafoOriModeSupported supported_orientation_modes; // Orientation modes supported by the TcCom transformation

// modulo configuration
ULONG dim_modulo; // dim of modulo vector
EcCnc_McsModulo * mcs_modulo;
EcCnc_AcsModulo * acs_modulo;

Caller identification

The active kinematic transformation is currently used at several points in the CNC: The different callers are noted in the working data transferred to the transformation.

0 : EcCncTrafoCallerID_Undefined
1 : EcCncTrafoCallerID_Decode
2 : EcCncTrafoCallerID_ToolRadiusCorrection
3 : EcCncTrafoCallerID_PathPreparation
4 : EcCncTrafoCallerID_Interpolation
5 : EcCncTrafoCallerID_Display
6 : EcCncTrafoCallerID_BlockSearch

The caller's identification (caller_id) is used to calculate the transformation at various points with variants, e.g.:

- dynamic variables can be added during the interpolation
- a simplified backward transformation can be calculated for display.

Transformation options

While the transformation is initialised (TrafoSupported method), you can select individual CNC options. These options change the CNC interface management and may supply additional parameters. Each of the options is predefined by the CNC and must match the corresponding transformation. The following options are available:

0 : EcCncTrafoOption_None
1 : EcCncTrafoOption_Interpolation_AddInput

Control input

The following data is transferred cyclically to the kinematic transformation

0x0000 0001 EcCncTrafoCtrl_cartesianTrafoInactive
Version number of CNC interface

In the TcCncVersion data item, the CNC transfers the version number of the transformation interface it uses:

```c
struct TcCncVersion {
    Long   major;
    Long   minor;
};
```

Rotation sequence

In the actual_rotation_mode data item, the CNC transfers the active rotation sequence of the orientation axes:

- `EcCncTrafoOri_YPR` = 0x1
- `EcCncTrafoOri_CBC1` = 0x2
- `EcCncTrafoOri_CBA` = 0x4
- `EcCncTrafoOri_CAB` = 0x8

The rotation sequences supported in the transformation are transferred to the CNC in the supported_rotation_modes data item:

```c
typedef struct _EcCnc_TrafoOriModeSupported {
    unsigned long   f_YPR : 1;
    unsigned long   f_CBC1 : 1;
    unsigned long   f_CBA : 1;
    unsigned long   f_CAB : 1;
} EcCnc_TrafoOriModeSupported;
```

Modulo settings

The CNC supplies the dimension of the axis-specific objects mcs_modulo and acs_modulo in the object dim_modulo. Modulo handling in the MCS coordinate system is transferred to the CNC in the axis-specific data item mcs_modulo:

- `EcCnc_McsModulo_None` = 0
- `EcCnc_McsModulo_180_180` = 1

The CNC transfers the extended modulo setting of an axis in the ACS coordinate system in the acs_modulo data item:

- `EcCnc_AcsModulo_None` = 0
- `EcCnc_AcsModulo_180_180` = 1
- `EcCnc_AcsModulo_0_360` = 2

Disabling intersection calculation if #CS is inactive

For example, if the kinematic transformation varies depending on whether a higher-level Cartesian transformation is active or not, you can select this operation by means of the input bit. This is indicated by the controller.

Kinematic transformation when intersection calculation is active (`EcCncTrafoCtrl_cartesianTrafoInactive deleted`)
Interfacing transformation via TcCOM

Figure 9: Kinematic transformation when intersection calculation is active

Kinematic transformation when intersection calculation is inactive

(EcCncTrafoCtrl_cartesianTrafoInactive set)

NOTE
Forward transformation must always be inverse to backward transformation.
position = forward(backward(position))
If the transformation varies is dependent on the caller (caller_id), then disable this variation when the controller is initialised at standstill.
Interfacing transformation via TcCOM

Figure 11: Identification of the transformation callers
2.1.3 Configuring and registering the transformation with the CNC

Registering the transformation

The following data is used to register a TcCOM (TcCncServices.h)

- **Type** 1 (see TCCNC_REGISTEROBJECT_TYPE_TRAFO) is defaulted
- **Group** Channel number of transformation [1;12] der Transformation [1;12] selectable at configuration
- **Index** Number of the kinematic [65;69] selectable at configuration

The transformation is registered via the following TcCOM interface, which is defined in the TcCncInterfaces.h file.

- virtual HRESULT TCOMAPI RegisterObject (TcCncRegisterObject& id, IUnknown* ipUnk)=0;
- virtual HRESULT TCOMAPI UnregisterObject (TcCncRegisterObject& id)=0;

Supplying the transformation manually

After the transformation is generated, two files must be provided for its uses.

The transformation is described in the TCM file TcCncTrafo1.tmc and is stored out of the source code directory in the following target directory.

<TwinCAT>\3.1\CustomConfig\Modules

The generated device driver (e.g. TcCncTrafo1.sys) is stored by

<TwinCAT>\3.1\SDK\products\TwinCAT RT (x86)\Release

in

<TwinCAT>\3.1\Driver\AutoInstall

. For debugging, the generated device driver (e.g. TcCncTrafo1.sys) and the symbol file (e.g. TcCncTrafo1.pdb) are stored by

<TwinCAT>\3.1\SDK\products\TwinCAT RT (x86)\Debug

in

<TwinCAT>\3.1\Driver\AutoInstall

. 
Configuration of the transformation

When the transformation is configured, the TcCOM object is selected in the System Manager and the channel (group) and the transformation ID (index) are initialised.

Figure 12: Inserting a transformation module

Figure 13: Configuring kinematic transformation via TcCOM in TwinCAT3
3 Parametrisation

Transformation parameters

Users can parameterise the transformation via channel and/or tool-specific values. The parameters' meanings depend purely on the implementation of the transformation. The parameters can be initialised in the following areas and have different validity periods:

- **CNC channel**
  The channel parameters can be set for each channel. They apply in the CNC configuration until this channel data is updated (download in System Manager).

- **Tool**
  Tool parameters are included with the tool request (D programming in the NC program). They apply until the tool is selected in the NC program. The parameters can be initialised individually for each tool.

- **TcCOM**
  Global parameters can be specified when the kinematics are configured. These apply for as long as the transformation is loaded, i.e. as long as TwinCAT is active.

### 3.1 CNC parameters: Channel and tool

The transformation parameters for the CNC channel and the tools are supplied to the transformation by means of transfer parameters (pointer p to structure TcNcTrafoParameter).

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>p-&gt;para[0]</td>
<td>1088000</td>
</tr>
<tr>
<td>p-&gt;para[1]</td>
<td>1987000</td>
</tr>
</tbody>
</table>

If a tool is selected (D word, see [PROG//Tool geometry compensation], the sum of the kinematic parameters is transferred from the channel parameter list and the tool.

**Example:**

Channel parameter list: `kinematic[65].param[2]` 300000


**NOTE**

The transformation parameter with index 0 (`kinematic[..].param[0]`) always acts in the direction of the 3rd main axis (normally the Z axis) and is also included in the calculation of the tool length. Therefore, if the unchanged length of the tool is required for the transformation, this parameter should not be used.
### 3.1.1 Transformation parameters of the tool

The tool parameters can be managed in the CNC or in an external tool management system (e.g., in the PLC). If the tool parameters are managed in the CNC, i.e., the channel parameter ext_wzv_vorhanden = 0 (see P-CHAN-00016) is set, the tab “Tool Para” and the tool parameter list are available in the TwinCAT3 project.

**Tool parameterisation example**

For parameterisation in the tool list, see P-TOOL-00009

```
wz[5].kinematic.param[0] 1538000   # Head offset 1: 153.8 mm
wz[5].kinematic.param[1] 25000    # Head offset 2: 2.5 mm
wz[5].kinematic.param[2] 0        # Head offset 3: 0 mm
wz[5].kinematic.param[5] 900000   # Head offset 6; 90 mm
```

The kinematic parameters of the tool can only be specified for default step = 0.

![Figure 14: Transformation parameters of the tool](image)

### 3.1.2 Channel parameter

**Channel parameterisation example**

For parameterisation in the tool list, see P-CHAN-00094

```
kinematik_id 65 standard kinematic: 65

kinematik[65].param[0] 1088000
kinematik[65].param[1] 1987000

kinematik[66].param[0] 1538000
kinematik[66].param[1] 25000
```
Figure 15: Channel transformation parameter

```plaintext
Parametrisation

kinematik[66].param[2] 0
# kinematik[67].param[0] 1487000
kinematik[67].param[1] 25000
kinematik[67].param[2] 0
```
3.2 TcCOM parameters

TcCOM transformation parameter

Besides the channel or tool-specific parameters, which are made available via the CNC, further individual parameters can be passed on to the transformation. These are initialised during configuration of the TcCOM object.

Figure 16: Transformation parameters via TcCOM
The TcCOM parameters required for the transformation can be defined in the TMC Editor:

*Figure 17: TMC Editor*

Use the TwinCAT TMC Code Generator (right-click on the TcCnCTrafo project -> TwinCAT TMC Code Generator) to automatically add the parameters in the TMC file to the class for the transformation CtcCncTrafo1 as member variables, e.g. `m_Parameter`. They can then be used in forward and backward transformation.

```c
///<AutoGeneratedContent id="Members">
    TcCncRegisterObject m_CncObjectRef;
    double m_Parameter[10];
///</AutoGeneratedContent>
```
4 Error handling and diagnosis

4.1 Error message

<table>
<thead>
<tr>
<th>Administration errors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>292019</td>
<td>Programmed transformation is not loaded, i.e. possibly not configured in TwinCAT</td>
</tr>
<tr>
<td>292020</td>
<td>Not enough memory for transformation (system error)</td>
</tr>
<tr>
<td>292021</td>
<td>Transfer unknown channel number internally (system error)</td>
</tr>
<tr>
<td>292022</td>
<td>Programmed transformation is unknown internally (loaded), i.e. possibly not configured in TwinCAT</td>
</tr>
<tr>
<td>292023</td>
<td>The backward transformation is not inverse to the forward transformation.</td>
</tr>
<tr>
<td>292030</td>
<td>Error on request of configuration data of kinematic transformation (see GetDimension())</td>
</tr>
<tr>
<td>292031</td>
<td>Error on initialisation of kinematic transformation (see TrafoSupported()).</td>
</tr>
<tr>
<td>292032</td>
<td>Error on kinematic forward transformation (see Forward()).</td>
</tr>
<tr>
<td>292033</td>
<td>Error on kinematic backward transformation (see Backward()).</td>
</tr>
<tr>
<td>292034</td>
<td>Current MCS-input position of the kinematic forward-transformation.</td>
</tr>
<tr>
<td>292035</td>
<td>Current WPCS-output position of the kinematic forward-transformation.</td>
</tr>
<tr>
<td>292036</td>
<td>Current WPCS-input position of the kinematic backward-transformation.</td>
</tr>
<tr>
<td>292037</td>
<td>Current MCS-output position of the kinematic backward-transformation.</td>
</tr>
<tr>
<td>292044</td>
<td>The transformation interface of the CNC is too old and does not match the TcCOM object.</td>
</tr>
<tr>
<td>292045</td>
<td>The orientation type selected is not supported by the transformation.</td>
</tr>
</tbody>
</table>

Example of default error: Logging in diagnostic data

(Date/Time): 07.09.2012 / 11:37:38
Version: V3.00.3012.04   Module: DECU_TRF.C   Cycle: 3108
------------------------------------------------------------
ERRTXT: Backward transformation after forward transformation results in different position.
------------------------------------------------------------
Error ID : 292023  BF type : 9  Channel ID : 1
Multiple ID : 1  Line : 2213  Commu ID : 42
Recovery class: 2  Reaction class: 2  Body type: 1
NC file : log. path no. 65535 -> D:\TwinCAT3\test.nc
NC program: trafo65test
NC prog. info:
Block number : 20  File offset: 55  Block offset: 14
--------------------- NC_block ------------------------------
Output not possible! log_pfad_nr not in assignment table.
Value_1: Current value is 65 [-]
Value_2: Error value is 1005 [-]

--------- end of error message -----------------------------

User-specific transformation errors

Besides the standard transformation errors, you can issue user-defined errors by using the function return value (0 = OK) with several methods see italics, bold error IDs).

HRESULT CTrafo::Forward(PTcNcTrafoParameter p)
{
    if (...)
    {
        return 123; // raise error
    }
Error handling and diagnosis

... return S_OK;
}

**Error messages in TcCncUsers-Events.xml**

In the event of an error (bold, italic IDs), the user-defined return value of the method can be transferred to the error message evaluation via the PLC (ChannelError()-Manager). The error texts are supplemented accordingly in the XML error text file for each language (C:\TwinCAT\3.1\Target\Resource):

```xml
<Event>
  <Id>123</Id>
  <Message LcId="1033">kinematic transformation reports error 123</Message>
  <Message LcId="1031">kinematic transformation reports error 123</Message>
</Event>
```

**Output in event logger**

**Extended error return values**

If the extended transformation parameter `TcNcTrafoParameterExtCnc` is used, additional error values may be returned in the event of an error. These values are displayed in the error message.

```c
double ret_value1; // out: error value
double ret_value2; // out: error value
char ret_text[24]; // out: additional error info
```

**User-specific errors**

```xml
20.06.2013 16:31:06:019 (11862) Version: .00.3017.00
Error : 292033 - Error on kinematic backward transformation
Program : trafo65test
Path : D:\TwinCAT3\ (No: 65535)
File : _trafo65-error-test.nc
Block no: N60 Fileoffset: 151
Line : N060 Y42 ; util_error_Id = -12

Channel : (No.: 1)
Value 1 : Actual value : 65
Class : ERROR (5) Reaction : PROGRAM_ABORT (2)
-----------------------------------------------
Value 2 : Actual value : 0
Value 3 : Actual value :
-----------------------------------------------
Utility : Error 123 - ...
Module : Line : 0
-----------------------------------------------
Config : ONE_CHANNEL_CONFIGURATION / ...
Module : BAVO_5AX.C Line : 6438
BF-Type : BAVO (5) Commu: BAVO_1 (44) Multiple ID: 0
Content : NC_PROGRAM (1)
-----------------------------------------------

20.06.2013 16:31:06:019 (11862) Version: .00.3017.00
Error : 292036 - Current WPCS output position of the kinematic forward transformation.
```
4.2 Diagnostic data

Log of axis positions

The <n> input/output positions of the kinematic transformation last used are logged. When diagnostic data is requested (see dump.bat), these values are logged in the diagnostic data diag_data.txt. The following transformations are recorded in the diagnostic data:

- positions of the decoder forward transformation
- positions of the backward transformation during interpolation

Logging in diagnostic data

<table>
<thead>
<tr>
<th>TIME</th>
<th>ID0</th>
<th>ID1</th>
<th>IN[00]</th>
<th>IN[01]</th>
<th>IN[02]</th>
<th>IN[03]</th>
</tr>
</thead>
<tbody>
<tr>
<td>250907</td>
<td>87</td>
<td>0</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>250908</td>
<td>87</td>
<td>0</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>250909</td>
<td>87</td>
<td>0</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>250939</td>
<td>87</td>
<td>0</td>
<td>822000.000</td>
<td>0.000</td>
<td>0.000</td>
<td>-814000.000</td>
</tr>
<tr>
<td>250946</td>
<td>86</td>
<td>0</td>
<td>822000.000</td>
<td>0.000</td>
<td>0.000</td>
<td>-814000.000</td>
</tr>
<tr>
<td>250947</td>
<td>86</td>
<td>0</td>
<td>822000.000</td>
<td>0.000</td>
<td>0.000</td>
<td>-814000.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VALUE</th>
<th>Actual value</th>
<th>0 / 1.05E+005 / 0 [0.1*10^-3 mm or ░]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value 1 :</td>
<td>Actual value</td>
<td>0 / 0 / 0 [0.1*10^-3 mm or ░]</td>
</tr>
<tr>
<td>Value 2 :</td>
<td>Actual value</td>
<td>0 / 0 / 0 [0.1*10^-3 mm or ░]</td>
</tr>
<tr>
<td>Value 3 :</td>
<td>Actual value</td>
<td>0 / 0 / 0 [0.1*10^-3 mm or ░]</td>
</tr>
<tr>
<td>Value 4 :</td>
<td>Actual value</td>
<td>0 / 0 / 0 [0.1*10^-3 mm or ░]</td>
</tr>
<tr>
<td>Value 5 :</td>
<td>Actual value</td>
<td>0 / 0 / 0 [0.1*10^-3 mm or ░]</td>
</tr>
</tbody>
</table>

Config : ONE_CHANNEL_CONFIGURATION / ...
Module : BAVO 5AX.C Line : 6438
BF-Type : BAVO (5) Commu: BAVO_1 (44) Multiple ID: 2
Content : NC_PROGRAM (1)
5 Working data of the transformation

Definition of working data

Implementation of the transformation can provide any parameters as working data. Make sure that the transformation is used in several timing phases of the CNC. This is why the CNC must be written as re-entrant. Therefore, the working data must not contain a state of the transformation that is used for subsequent calculation.
6 Concatenating transformations, multistep transformations

Multistep capability - Additive kinematic transformation

Normally, only one kinematic transformation is used but the CNC offers the option of cascading several partial kinematic transformations. At present, an additional transformation can be concatenated to the normal transformation

Using this option, you can structure your transformations independently:

- Standard kinematic transformation (Step=0): maps the basic kinematic chain of the machine (Configuration type = TCCNC_REGISTEROBJECT_TYPE_TRAFO)
- Additive kinematic transformation (Step=1): compensates, e.g. dynamic effects of the machine (Configuration type = TCCNC_TCCNC_REGISTEROBJECT_TYPE_TRAFO_ADD)

Figure 18: Concatenating kinematic transformations

Initialising kinematic parameters

The kinematic parameters for each step of the kinematic transformation can be initialised in the channel list in the following form:

| kin_step[0].id[83].param[0] | 10000 |
| kin_step[1].id[51].param[0] | 55000 |
| kin_step[1].id[51].param[1] | 80000 |

Initialising the standard transformation

The standard transformation of each step can be defined in the channel list in the following form.

| default_id_of_kin_step[0] | 83 |
| default_id_of_kin_step[1] | 51 |

Accessing parameters in the NC program

The kinematic parameters of each step can be addressed in the NC program in the following way,

\[
\text{N10 V.G.KIN_STEP[1].ID[1].PARAM[0]} = 55000 \\
\text{N20 V.G.KIN_STEP[1].ID[1].PARAM[1]} = 80000
\]

Activating a transformation for every step

Each of the kinematic steps can be selected by the following NC commands:

\[
\text{#TRAFO [<kin-id-step0>, <kin-id-step1>]}\]
Concatenating transformations, multistep transformations

#TRAFO [KIN_ID_DEF, KIN_ID_DEF]
  ; KIN_ID_DEF = default parameter default_id_of_kin_step

#TRAFO [ OFF, <kin-id-step1>]
#TRAFO [<kin-id-step0>, OFF]

#TRAFO [ OFF, OFF]
#TRAFO OFF
7 Generating a transformation

When you generate a TcCOM object using the TwinCAT3 template, a so-called extended transformation is created by default.

7.1 System requirements

1. TwinCAT3 installed
2. Visual Studio 2010 installed with Service Pack 1
3. WinSDK 2010 must be installed. The environment variable WINSDK must be assigned to the DDK directory (normally C:\WinSDK\7600.16385.1).
4. Refer to the TwinCAT3 Help in the chapter TwinCAT3 C/C++ for more information on system requirements and the development of C/C++ objects in TwinCAT3.

7.2 Generation process

The transformation is generated using a WinCAT3 template

Execute the following steps:

- Create or open a TwinCAT3 XAE project with integrated CNC configuration
- Create the scope for transformation using templates as shown in the example below.
- Generate user C++ code for transformation (this step can also take place later but then a new driver must also be generated)
- Generate driver (MyTrafo.sys)
- Integrate the transformation into the XAE project Configuration as TcCOM object
- Activating the configuration

The MyTrafo.sys driver is copied automatically to the \<TwinCAT\>\3.1\Driver\Autoinstall directory when the configuration is activated.

All additive drivers are placed in this directory.
7.2.1 Create project and transformation

Step 1- TwinCAT3 XAE project with CNC configuration

Figure 19: Create a project
Figure 20: Generate CNC configuration
Generating a transformation

Figure 21: Create channel
Figure 22: Create axes
Step 2: Create user transformation using TwinCAT3 templates.

Figure 23: Create TwinCAT driver project
Figure 24: Create transformation class
Figure 25: Name transformation class

This defines the framework for the TcCOM object in Visual Studio 2010.
Step 3: Create the driver

Right-click on the project to "Create" the driver.

Figure 26: Create driver
Integrate the transformation into the existing CNC configuration as follows:

Figure 27: Integrate TcCOM object

The integration is completed when you press the “OK” button to confirm.

Double-click on the TcCOM object to display the properties.
Figure 28: Properties of the TcCOM object

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Permissible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>1</td>
<td>Type = 1 specifies that the TcCOM object is a kinematic transformation.</td>
</tr>
<tr>
<td>Group</td>
<td>0 &lt;= group &lt;= number of channels</td>
<td>Which CNC channel may access the transformation is specified in the group parameter. If group = 0, the transformation is available for all CNC channels.</td>
</tr>
<tr>
<td>Index</td>
<td>65 &lt;= index &lt;= 69 or 500 &lt;= index &lt;= 999</td>
<td>Via the index parameter, the transformation receives a unique ID in the CNC channel by means of which it can be addressed in the CNC, e.g. in the NC program with the command #KIN ID [65].</td>
</tr>
</tbody>
</table>

Parameterise the transformation in the CNC

The created kinematic must still be parameterised in the CNC: This is executed in the default channel parameter list or in a particular channel parameter list.
7.2.3 Debugging the transformation

To debug, switch the transformation in the TwinCAT3 project over to debug and activate the real-time debugger.

Figure 29: Parameterise the transformation in the channel parameter list

The transformation ID to be entered in the index of the TcCOM object.

Figure 30: Switch over to debug configuration
Figure 31: Activate real-time debugging

- The MyTrafo.sys debug driver and the associated pdb file are copied automatically to the Autoinstall directory when the configuration is activated.

You can start debugging the transformation project after starting TwinCAT in the "RUN" state and setting the associated breakpoints.
Generating a transformation

Figure 32: Breakpoint in the transformation
7.2.4 Source code extension/encoding

To complete the generation process, integrate your user transformation equations in the functions

- Forward
- Backward
- TrafoSupported
- GetDimensions

They are already created as examples in the MyKinTrafo.cpp using the TwinCAT3 template.

User tip

If the transformation requires more than 5 axes, adapt the constructor as follows. If there are fewer than 5 axes, the values must be reduced correspondingly.

```cpp
// Constructor
CMyKinTrafo::CMyKinTrafo(): m_forwardNbrIn(5), m_forwardNbrOut(5)
{
```

Figure 33: Setting the constructor after generation using TwinCAT3 templates

```cpp
// Constructor
CMyKinTrafo::CMyKinTrafo(): m_forwardNbrIn(7), m_forwardNbrOut(7)
{
```

///<AutoGeneratedContent id="MemberInitialization">

Figure 34: Adapted constructor due to high number of axes

If you enter a value in the constructor that is higher than the number of axes in the channel, the error message 20658 is output. This error message is also output if there are gaps in the configuration of axes in the channel.

Possible solutions:

- Check and correct the gaps in the configuration
- Adapt the constructor to the number of the axes in the channel used

After implementing the functions, recreate the driver and re-activate the configuration.
7.3 Differences between extended transformation / standard transformation

When you use TwinCAT3 templates, the extended transformation is created by default. It is exclusively designed for use with the CNC.

The standard transformation is used when both the CNC and the NCI are used.

(NCI – controller solution from Beckhoff)

<table>
<thead>
<tr>
<th>Extended transformation</th>
<th>Standard transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended interface: ITcCncTrafo</td>
<td>Default interface: ITcNcTrafo</td>
</tr>
<tr>
<td>GUID extended interface: IID_ITcCncTrafo</td>
<td>GUID standard interface: IID_ITcNcTrafo</td>
</tr>
<tr>
<td>Extended transformation parameter:</td>
<td>Default parameter:</td>
</tr>
<tr>
<td>PTcCncTrafoParameter</td>
<td>PTcNcTrafoParameter</td>
</tr>
</tbody>
</table>
8 Parameter

Description of transformation-specific parameter

The parameterisation of kinematic transformations is described in the documentation of the channel parameter [CHAN].

The tool parameters for kinematic transformations are described in the documentation of the tool data [TOOL].
# Additional options of extended transformation

## Version identifier of transformation interface

In future, the transformation interface will be extended by new functions and therefore has a unique version identifier (<Major>,<Minor>). The CNC version number is supplied to the TcCOM transformation in the data item p->CncInterfaceVersion. The TcCOM object can request the unique version number using the GetInterfaceVersion() method. The CNC transformation interface is downwards compatible, i.e. TcCOM objects of an older interface version can continue to be used together with more recent CNC versions. However, the opposite does not apply: The interface version of the CNC must be at least as up-to-date as the transformation interface of the TcCOM object. Otherwise, the CNC generates the error message P-ERR-292044.

```c
HRESULT <UserTrafo>::TrafoSupported(PTcCncTrafoParameter p, bool fwd) {
    ...
    TcCncVersion TcCOMInterfaceVersion;
    this->GetInterfaceVersion(&TcCOMInterfaceVersion);
    if (   (TcCOMInterfaceVersion.major <= p->CncInterfaceVersion.major)
        && (TcCOMInterfaceVersion.minor <= p->CncInterfaceVersion.minor))
        return S_OK;
};
```

### Rotation sequence

When transformations are complete, the sequence of rotations executed about the 3 rotary axes can be defined to meet the requirements (see P-CHAN-00112). If this is required, the TcCOM transformation must also take this into consideration. Therefore, the current setting is transferred to the transformation in the p->actual_orientation_mode parameter. The rotation sequences supported in the transformation can be sent to the CNC in the data item p->supported_orientation_modes When the transformation is selected, the CNC checks the setting in P-CHAN-00112 for plausibility and generates the error message P-ERR-292045 if the transformation does not support the selected rotation sequence.

### CNC --> TcCOM transformation:

<table>
<thead>
<tr>
<th>p-&gt;actual_orientation_mode</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>EcCncTrafoOri_YPR</td>
<td>Yaw-Pitch-Roll rotation sequence 1. Rotation about Z, 2. Negative rotation about Y, 3. Rotation about X</td>
</tr>
<tr>
<td>EcCncTrafoOri_CBC1</td>
<td>Euler rotation sequence: 1. Rotation about Z, 2. Rotation about Y, 3. Rotation about Z'</td>
</tr>
<tr>
<td>EcCncTrafoOri_CBA</td>
<td>1. Rotation about Z, 2. Rotation about Y, 3. Rotation about X</td>
</tr>
<tr>
<td>EcCncTrafoOri_CAB</td>
<td>1. Rotation about Z, 2. Rotation about X, 3. Rotation about Y</td>
</tr>
</tbody>
</table>

### TcCOM transformation --> CNC:

<table>
<thead>
<tr>
<th>p-&gt;supported_orientation_modes</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>.f_YPR = TRUE</td>
<td>Transformation supports rotation sequence YPR</td>
</tr>
<tr>
<td>.f_CBC1 = TRUE</td>
<td>Transformation supports rotation sequence CBC'</td>
</tr>
<tr>
<td>.f_CBA = TRUE</td>
<td>Transformation supports rotation sequence CBA</td>
</tr>
<tr>
<td>.f_CAB = TRUE</td>
<td>Transformation supports rotation sequence CAB</td>
</tr>
</tbody>
</table>
By default, the CNC uses the setting EcCncTrafoOri_YPR (Yaw->Pitch->Roll). Accordingly, the data item p->supported_orientation_mode.f_YPR is set to the value TRUE by default.

```csharp
HRESULT <UserTrafo>::TrafoSupported(PTcCncTrafoParameter p, bool fwd)
{
    /* Transformation supports YPR and Euler rotation sequence. */
    p->supported_orientation_modes.f_YPR = TRUE;
    p->supported_orientation_modes.f_CBC1 = TRUE;
    ...
    return S_OK;
}
```

```csharp
HRESULT <UserTrafo>::Backward(PTcCncTrafoParameter p)
{
    ...
    if (EcCncTrafoOri_CBC1 == p->actual_orientation_mode)
    {
        /* Rotation sequence acc. to Euler active */
    }
    else
    {
        ...
    }
    return S_OK;
}
```

### 9.3 Modulo handling of axis positions

Normally, the positions in the MCS coordinate system is handled linearly by the CNC, i.e. no modulo correction takes place. If the transformation expects the MCS positions in the modulo interval [-180° - +180°] (e.g. for shortest way programming), a modulo correction can be activated for an axis in the MCS coordinate system in the TrafoSupported() function by the data item mcs_modulo.

<table>
<thead>
<tr>
<th>p-&gt;mcs_modulo[i]</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>EcCnc_McsModulo_None</td>
<td>Linear MCS positions, no modulo calculation for this axis</td>
</tr>
<tr>
<td>EcCnc_McsModulo_180_180</td>
<td>Modulo calculation of the MCS positions for this axis in the interval [-180°, +180°].</td>
</tr>
</tbody>
</table>

The calculated ACS coordinates must match the axis properties. If the axis uses modulo positions, the ACS coordinates in the transformation must also execute a modulo correction. Therefore, the modulo setting in the axis-specific data item acs_modulo used in the transformation is sent to the CNC. The CNC then checks whether the transformation matches the axis properties. If not, it generates the error message P-ERR-50534.

<table>
<thead>
<tr>
<th>p-&gt;acs_modulo[i]</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>EcCnc_AcsModulo_None</td>
<td>Linear ACS positions: no modulo handling is required for this axis.</td>
</tr>
<tr>
<td>EcCnc_AcsModulo_180_180</td>
<td>For this axis a modulo calculation of the ACS positions is required in the interval [-180°, +180°].</td>
</tr>
<tr>
<td>EcCnc_AcsModulo_0_360</td>
<td>For this axis a modulo calculation of the ACS positions is required in the interval [0°, 360°].</td>
</tr>
</tbody>
</table>

**Modulo handling of axis positions**

```csharp
HRESULT <UserTrafo>::TrafoSupported(PTcCncTrafoParameter p, bool fwd)
{
    ...
    /* 3 axes linear MCS positions, modulo handling for the 4th axis */
    p->mcs_modulo[0] = EcCnc_McsModulo_None
    p->mcs_modulo[1] = EcCnc_McsModulo_None
    p->mcs_modulo[2] = EcCnc_McsModulo_None
    p->mcs_modulo[3] = EcCnc_McsModulo_180_180
}
/* 2 axes linear ACS positions, modulo handling for 2 axes */
$p->acs_modulo[0] = EcCnc_AcsModulo_None
$p->acs_modulo[1] = EcCnc_AcsModulo_180_180
$p->acs_modulo[2] = EcCnc_AcsModulo_0_360
$p->acs_modulo[3] = EcCnc_AcsModulo_None
}

9.4 Use of extended parameters
The example below shows the use of extended transformation parameters.

```c
HRRESULT <User.Trafo>::TrafoSupported(PITcNc.TrafoParameterExtCnc p, bool fwd)
{
  if (p == NULL)
  {
    return E_POINTER;
  }
  if (p->type != EcNc.TrafoParameter_ExtCnc)
  {
    p->ret_value1 = (double)p->type;
    strcpy(p->ret_text, "EcNc.TrafoParameter");
    return S_FALSE;
  }
  if (p->i == NULL || p->o == NULL)
  {
    return E_INVALIDARG;
  }
  if (p->dim_i != m_forwardNbrIn)
  {
    p->ret_value1 = p->dim_i;
    p->ret_value2 = m_forwardNbrIn;
    strcpy(p->ret_text, "m_forwardNbrIn");
    return S_FALSE;
  }
  if (p->dim_o != m_forwardNbrOut)
  {
    p->ret_value1 = p->dim_o;
    p->ret_value2 = m_forwardNbrOut;
    strcpy(p->ret_text, "m_forwardNbrOut");
    return S_FALSE;
  }

  /* request addition input parameters ... if backward interpolation trafo */
  if ((FALSE == fwd) && (p->caller_id == EcCnc.TrafoCallerID_Interpolation))
  {
    ...
  }
  return S_OK;
}
9.5 Use of extended options

Number of inputs/outputs

Normally, the number of inputs and outputs is symmetrical in the forward and backward directions. This basic number is defined by the method GetDimension.

For special requirements, the transformation can evaluate additional inputs. The method TrafoSupported can match the number of inputs/outputs to the requirements.

- CNC option (ret_option)
- Number of additional input values (dim_i)

In this case, the CNC must supply the additional values to the interface. If the CNC does not support this function, an error message is output.

Figure 35: Adapting the number of inputs/outputs

The transformation options can be set during a configuration request (TrafoSupported method).
Additional options of extended transformation

HRESULT <User::Trafo>::TrafoSupported(PITcNcTrafoParameterExtCnc p, bool fwd)
{
    /*
    | request addition input parameters ... if backward interpolation trafo
    +--------------------------------------------------+
    */
    if ((FALSE == fwd) && (p->caller_id == EcCncTrafoCallerID_Interpolation))
    {
        p->ret_option = EcCncTrafoOption_Interpolation_AddInput;
        p->dim_i += 8;
    }

    return S_OK;
}

Forward/backward adaptation

Adaptation can be carried out for each forward/backward transformation. In addition, this can also be carried out depending on the callers within the CNC (decoder, tool radius compensation etc.).
Figure 36: Interfaces for adaptation to various callers.
10 Display the position of the additive transformation

During the interpolation, the additive transformation is called for display purposes (caller ID = 5 = EcCncTrafoCallerID_Display). These position values are accessible for each axis via ADS.

mc_ax_<i>_add_kin_pos_r

Figure 37: Displaying the additive transformation position

Additive transformation positions can also be verified by the COM task in the ISG object browser.