DOCUMENTATION ISG-kernel

McCOM - Interface to a kinematic transformation

Short Description:
McCOM-Trafo

www.isg-stuttgart.de
support@isg-stuttgart.de

Documentation version
13/12/2023

Preface

Legal information

This documentation was produced with utmost care. The products and scope of functions described are under continuous development. We reserve the right to revise and amend the documentation at any time and without prior notice.

No claims may be made for products which have already been delivered if such claims are based on the specifications, figures and descriptions contained in this documentation.

Personnel qualifications

This description is solely intended for skilled technicians who were trained in control, automation and drive systems and who are familiar with the applicable standards, the relevant documentation and the machining application.

It is absolutely vital to refer to this documentation, the instructions below and the explanations to carry out installation and commissioning work. Skilled technicians are under the obligation to use the documentation duly published for every installation and commissioning operation.

Skilled technicians must ensure that the application or use of the products described fulfil all safety requirements including all applicable laws, regulations, provisions and standards.

Further information

Links below (DE)

https://www.isg-stuttgart.de/produkte/softwareprodukte/isg-kernel/dokumente-und-downloads

or (EN)

https://www.isg-stuttgart.de/en/products/softwareproducts/isg-kernel/documents-and-downloads

contains further information on messages generated in the NC kernel, online help, PLC libraries, tools, etc. in addition to the current documentation.

Disclaimer

It is forbidden to make any changes to the software configuration which are not contained in the options described in this documentation.

Trade marks and patents

The name ISG®, ISG kernel®, ISG virtuos®, ISG dirigent® and the associated logos are registered and licensed trade marks of ISG Industrielle Steuerungstechnik GmbH.

The use of other trade marks or logos contained in this documentation by third parties may result in a violation of the rights of the respective trade mark owners.

Copyright

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

➢ 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.

Attention

Restriction or error

This icon describes restrictions or warns of errors.

Notice

Tips and other notes

This icon indicates information to assist in general understanding or to provide additional information.

Example

General example

Example that clarifies the text.

Programing Example

NC programming example

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

Release Note

Specific version information

Optional or restricted function. The availability of this function depends on the configuration and the scope of the version.

Contents

List of figures

1 Kinematic transformation (TRAFO)

1.1 Introduction

Release Note

This function is available as of TwinCAT 3 and higher.

Definition of McCOM

Motion **C**ontrol **C**omponent **O**bject **M**odel is a binary interface standard in machine tool controllers.

Based on the Microsoft COM standard, McCOM defines how various software components developed and generated independently can cooperate in real time.

The acronym TcCOM stands for the **T**win**C**AT **C**omponent **O**bject **M**odel concept.

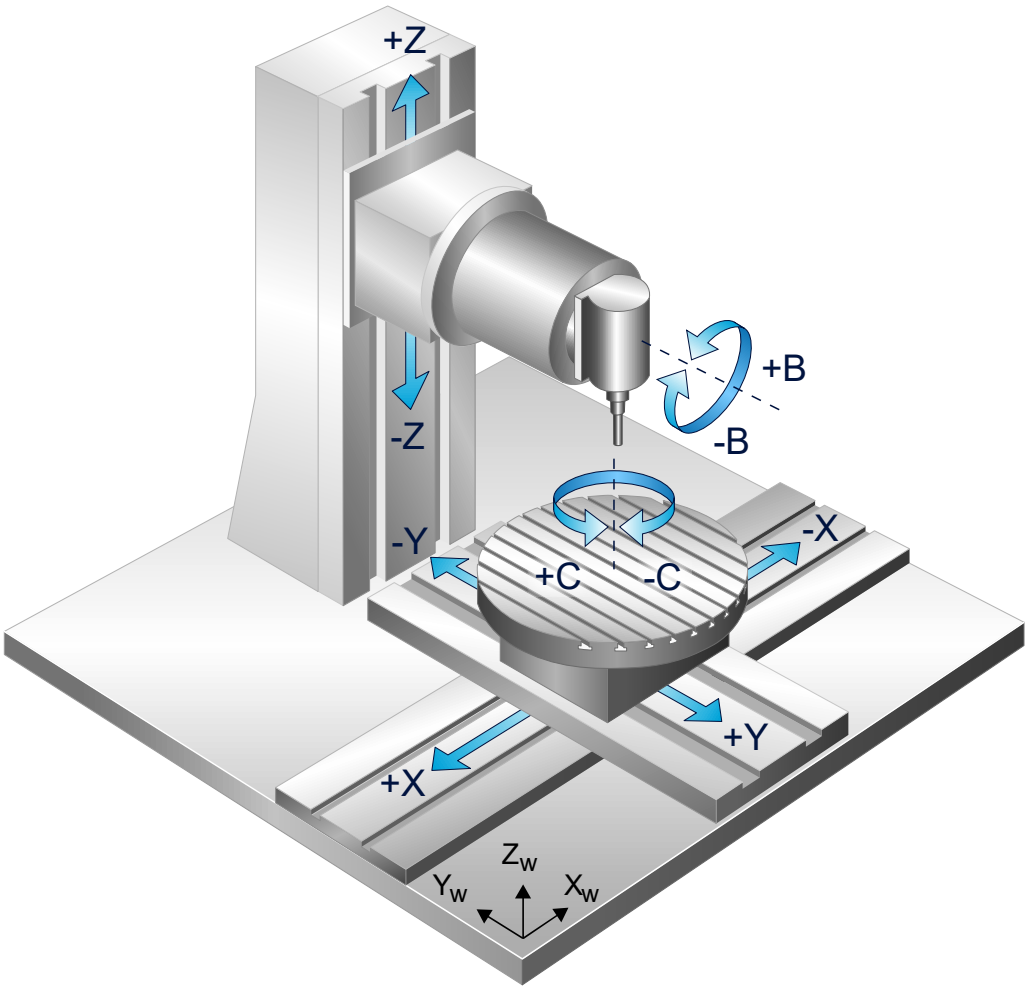

Fig. 1: Example of a kinematic transformation

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 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.

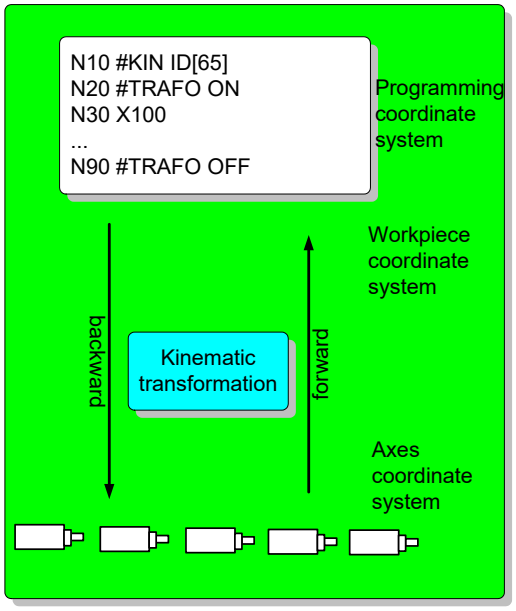

Fig. 2: Function of kinematic transformation

Extendibility

You can create a user-defined transformation and make it available to the CNC under a selected number (ID). The range [500; 999] is provided as the transformation numbers for this operation. The range [65; 69] continues to be provided for compatibility reasons.

1.2 Coordinate systems

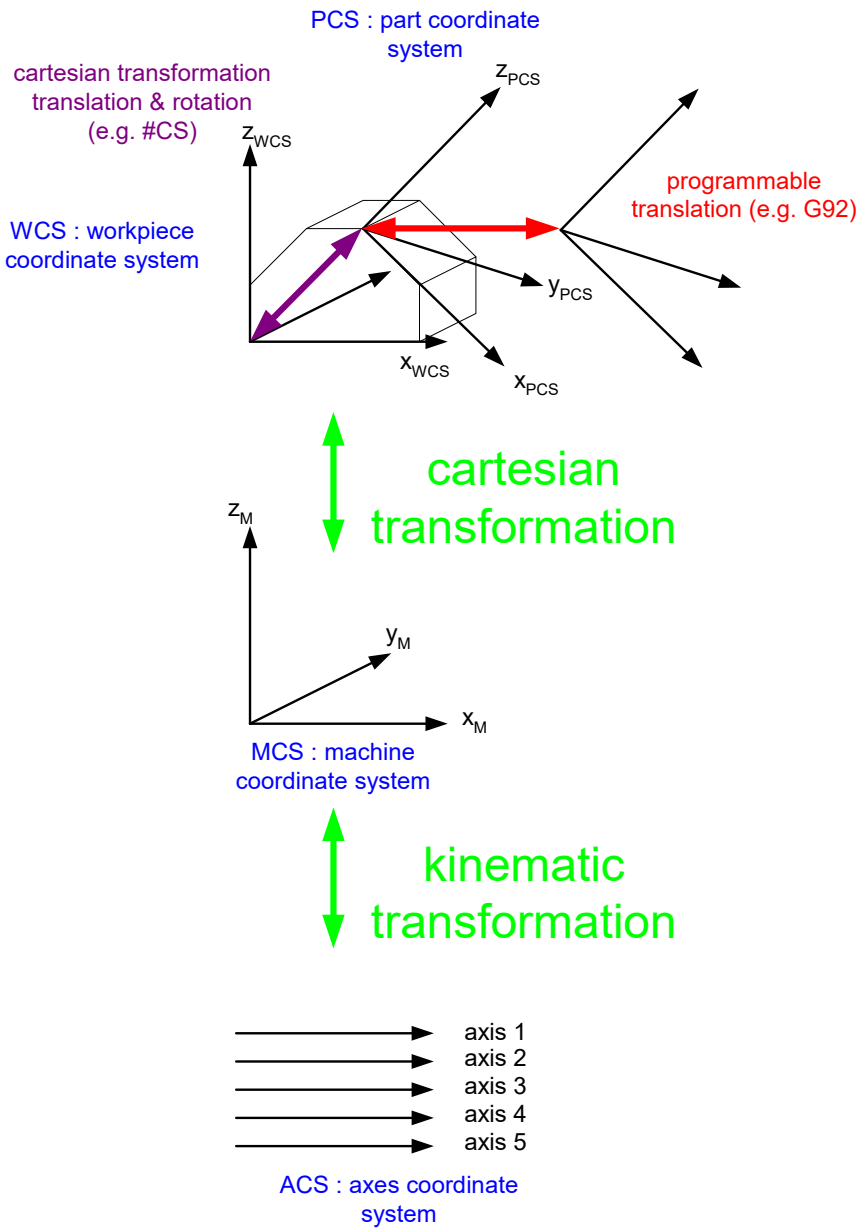

Fig. 3: Coordinate systems in detail

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 ($WPCS_0$).

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.

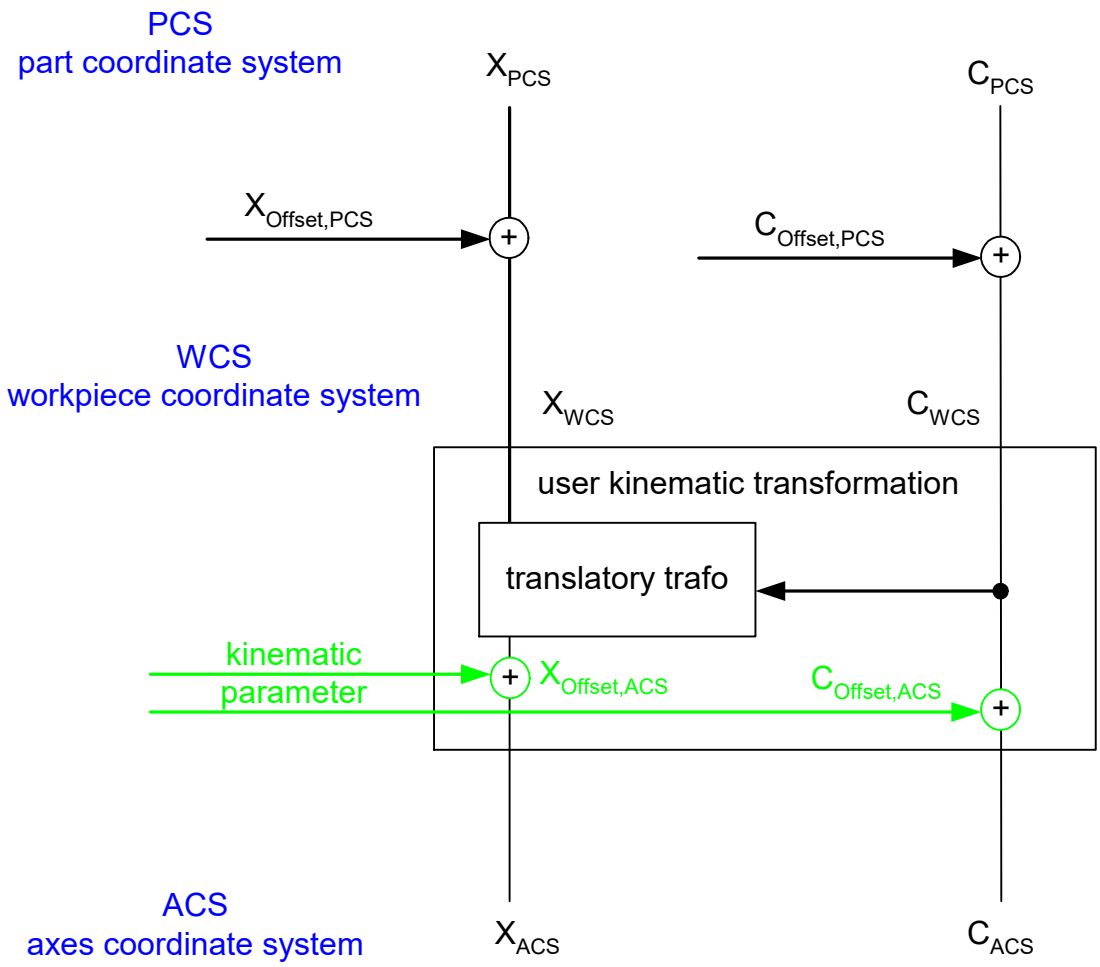

Fig. 4: Coordinate systems in detail

Programing Example

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 trans-
formation
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).

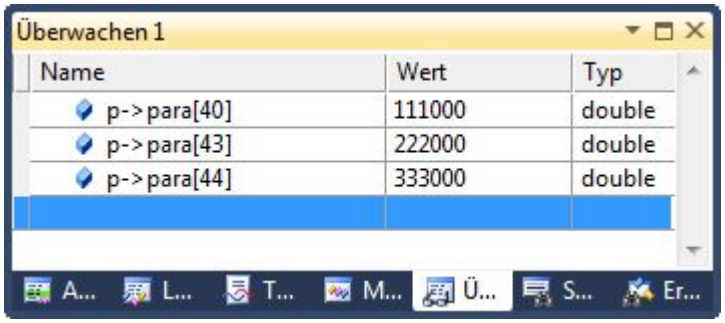

Fig. 5: Access to kinematic parameters

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]

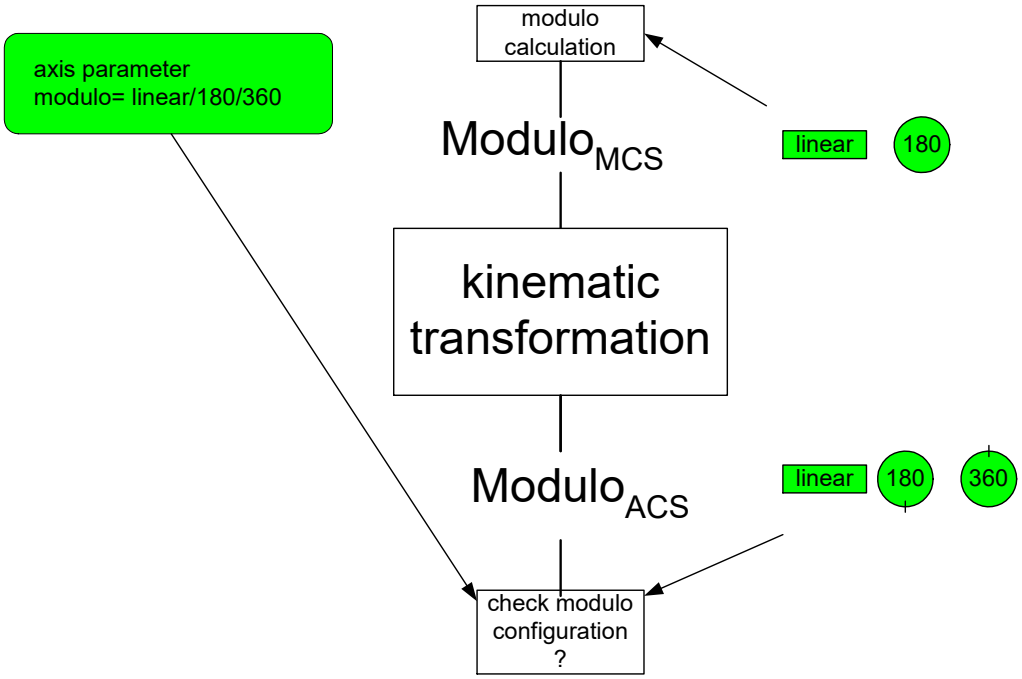

Fig. 6: Modulo handling of an axis

2 Interfacing transformation via TcCOM

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

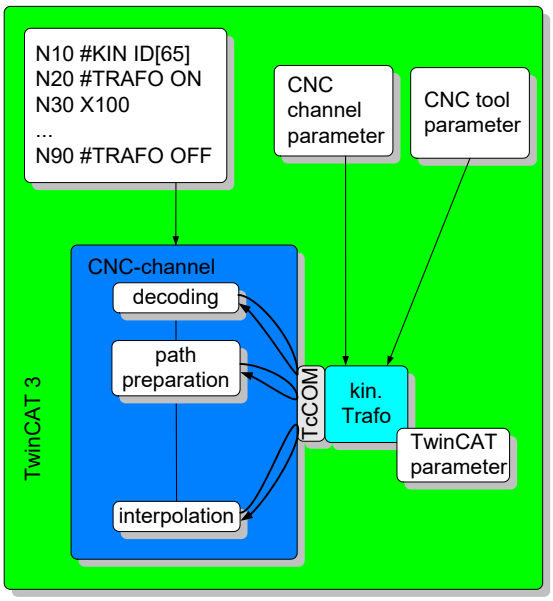

Fig. 7: Interfacing kinematic transformation via TcCOM in TwinCAT3

Attention

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.

Attention

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;

Forward	Transformation of the axis positions into the programming coordinate system.
PTcCncTrafoParameter *p	Current parameters of the transformation

Backward	Transformation of the programming coordinates into the axis coordinate system.
PTcCncTrafoParameter *p	Current parameters of the transformation

GetDimension	When the transformation is selected, a configuration request (required axis numbers) is executed once.
ULONG * pForwardInput	Number of forward transformation input coordinates (= number of reverse transformation output coordinates)
ULONG * pForwardInput	Number of forward transformation output coordinates (= number of reverse transformation input coordinates)

TrafoSupported	Initialising the transformation and requesting options
PTcCncTrafoParameter *p	Current parameters of the transformation
bool fwd	

Notice

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

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

Fig. 8: Dimensioning the input and output coordinates

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).

```
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, TcCnc-
Param

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 trans-
formation during TrafoSupported()
TcCncVersion CncInterfaceVersion; // Interface version
TcCncVersionMajor.TcCncVersionMinor

// orientation
EcCnc_TrafoOriModeActual actual_orientation_mode; // Treatment
of orientation, actual rotation sequence
EcCnc_TrafoModeSupported supported_modes; // modes supported by
the TcCOM transformation
```

Warning:

The structure element EcCnc_TrafoModeSupported supported_modes replaces the previous element EcCnc_TrafoOriModeSupported supported_orientation_modes. However, the data item is still supported for downward compatibility reasons.

```
// 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:

```
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_None = 0
EcCncTrafoOri_YPR  = 1
EcCncTrafoOri_CBC1 = 2
EcCncTrafoOri_CBA  = 3
EcCncTrafoOri_CAB  = 4
EcCncTrafoOri_AB   = 5
EcCncTrafoOri_BA   = 6
```

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

```
typedef struct _EcCnc_TrafoModeSupported
{
  unsigned long   f_YPR   : 1;
  unsigned long   f_CBC1  : 1;
  unsigned long   f_CBA   : 1;
  unsigned long   f_CAB   : 1;
  unsigned long   f_UniqueTrafo  : 1;
  unsigned long   f_AB    : 1;
  unsigned long   f_BA    : 1;
```

```
} EcCnc_TrafoModeSupported;
```

Unique CNC --> TcCOM transformation

The flag f_UniqueTrafo in the data item supported_modes allows the user to mark the TcCOM transformation as unique in forward and backward directions. As user, you can set the flag in the TafoSupported method. By default, the CNC handles TcCOM transformations as not unique. The flag f_UniqueTrafo is checked when the transformation is initialised.

Setting the flag accelerates by a few cycles all operations where the CNC must read the positions of the drives. Such operations include selecting or deselecting the transformation, changing coordinate systems when a TcCOM transformation is active or using V.A.ACS.ABS variables.

Example code to set the flag:

```
virtual HRESULT TCOMAPI TrafoSupported(PTcCncTrafoParameter p,
bool fwd)
{
  p->supported_modes.f_UniqueTrafo = TRUE;
  return S_OK;
};
```

s

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,
```

Example

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)

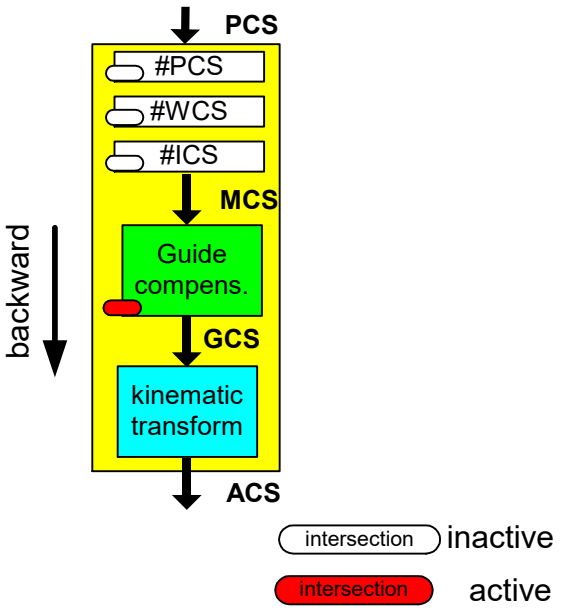

Fig. 9: Kinematic transformation when intersection calculation is active

Kinematic transformation when intersection calculation is inactive
(EcCncTrafoCtrl_cartesianTrafoInactive set)

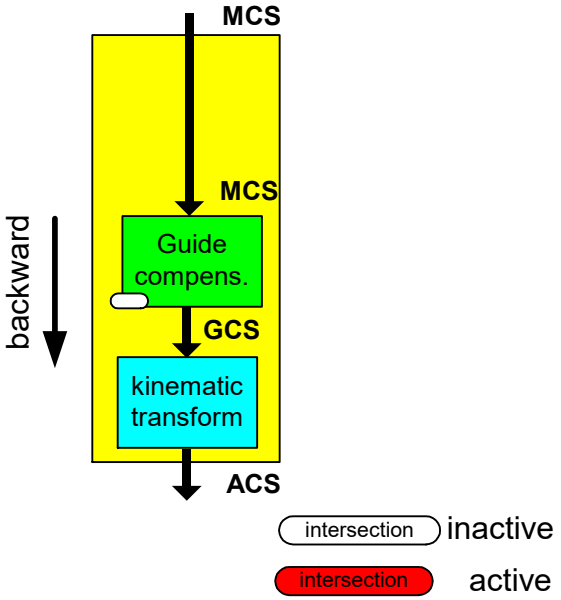

Fig. 10: Kinematic transformation when intersection calculation is inactive

Attention

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.

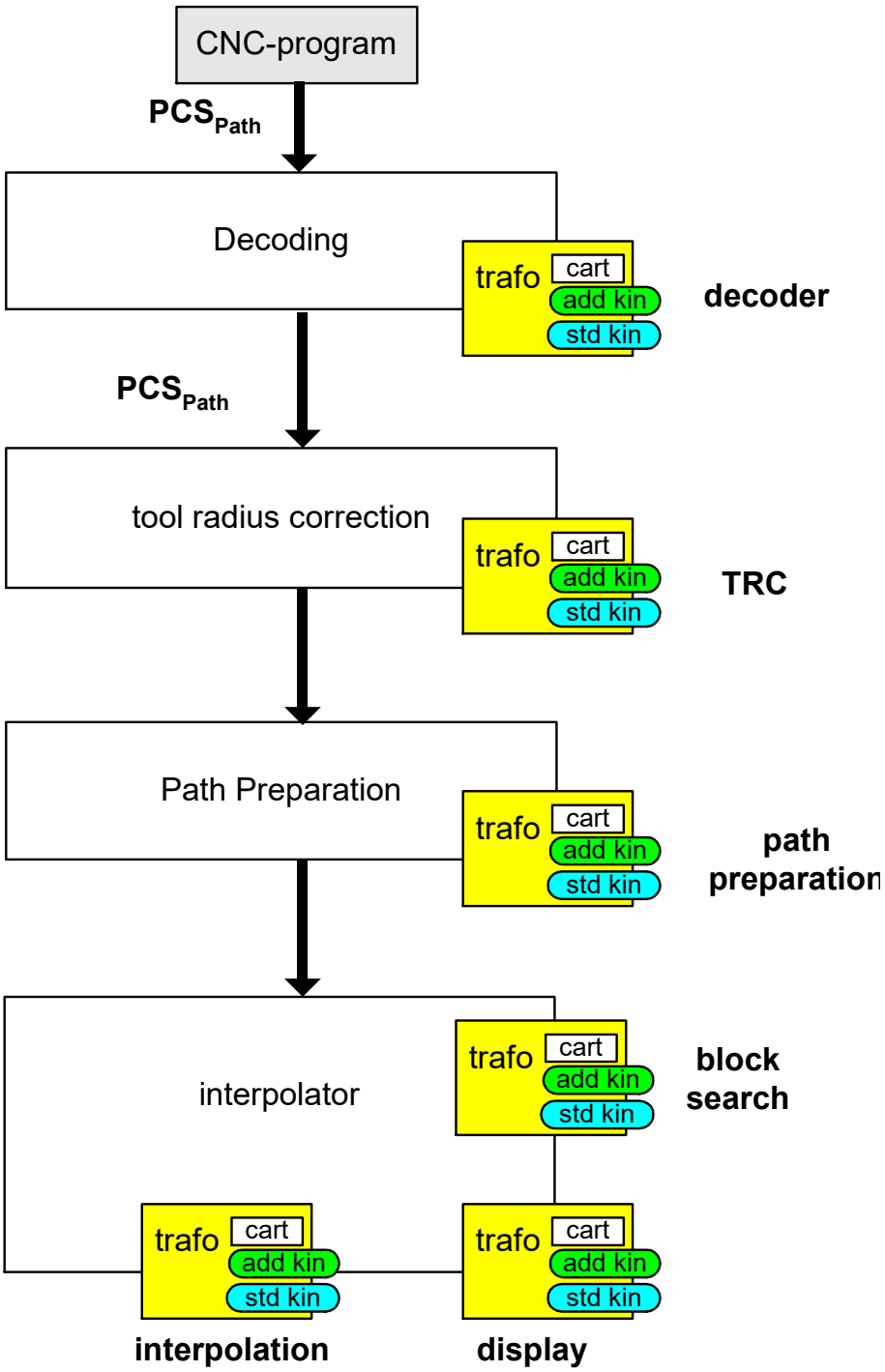

Fig. 11: Identification of the transformation callers

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] selectable at configuration
- Index Number of the kinematic [500;99] selectable at configuration; also permitted for compatibility reasons [65;69]

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

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

Manual transformation provision

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

The transformation is described in the TMC 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.

The procedure is illustrated in Integrate transformation [▶ 43].

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).

Name	Wert
p->para[0]	1088000
p->para[1]	1987000
p->para[2]	342000

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
Selected tool: wz[5].kinematic.param[2]       500000
```

Transferred transformation parameter: p->para[2] = 800000

Attention

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.

Example

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
```

Notice

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

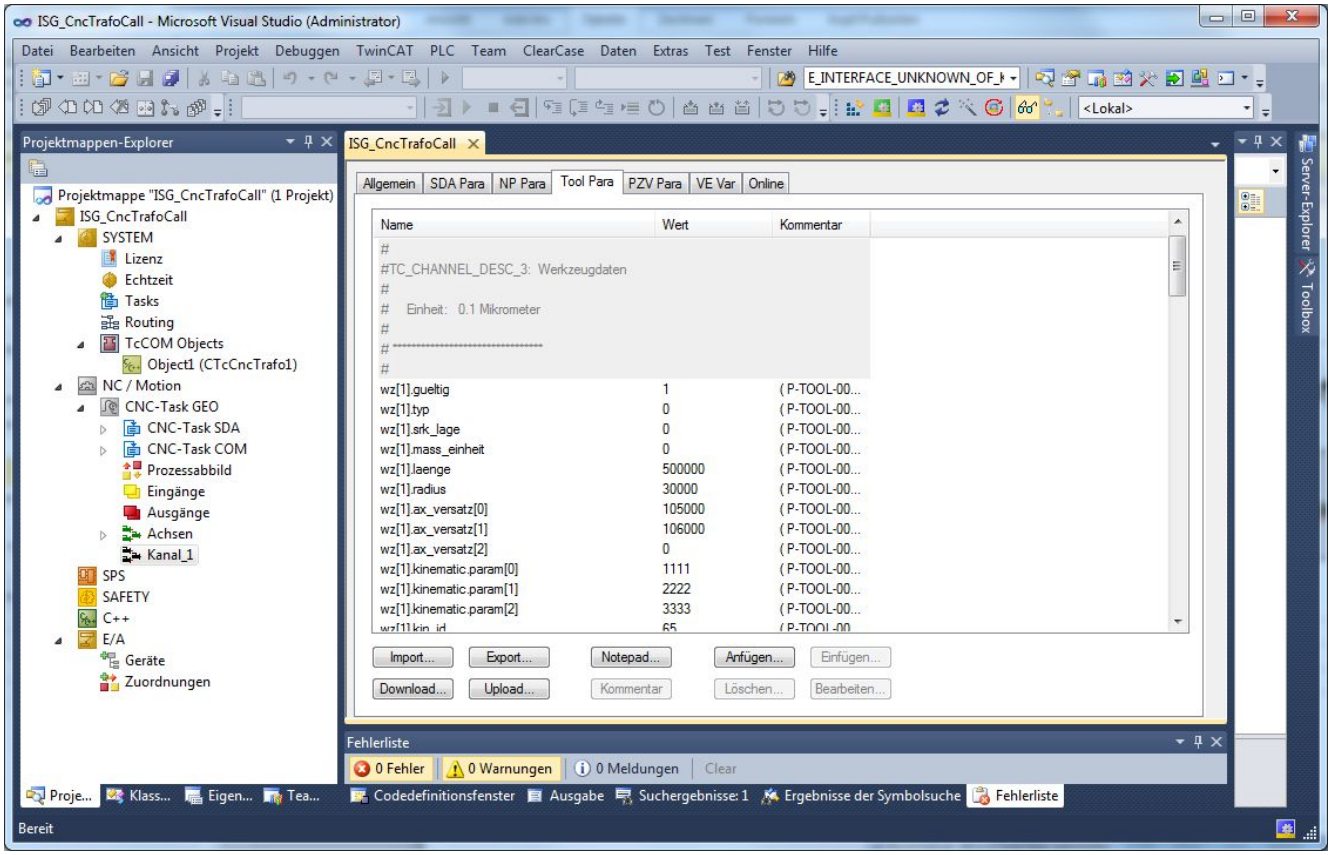

Fig. 12: Transformation parameters of the tool

3.1.2 Channel parameter

Example

Channel parameterisation example

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

```
# -- TcCOM Transformation
#
trafo[0].id             500
trafo[0].param[0]       1088000
trafo[0].param[1]       342000
trafo[0].param[2]       150
trafo[0].param[3]       0
trafo[0].param[4]       0
trafo[0].param[5]       0
trafo[0].param[6]       0
#
trafo[1].id             9
trafo[1].param[0]       120000
trafo[1].param[1]       100000
trafo[1].param[2]       120
trafo[1].param[3]       0
trafo[1].param[4]       0
```

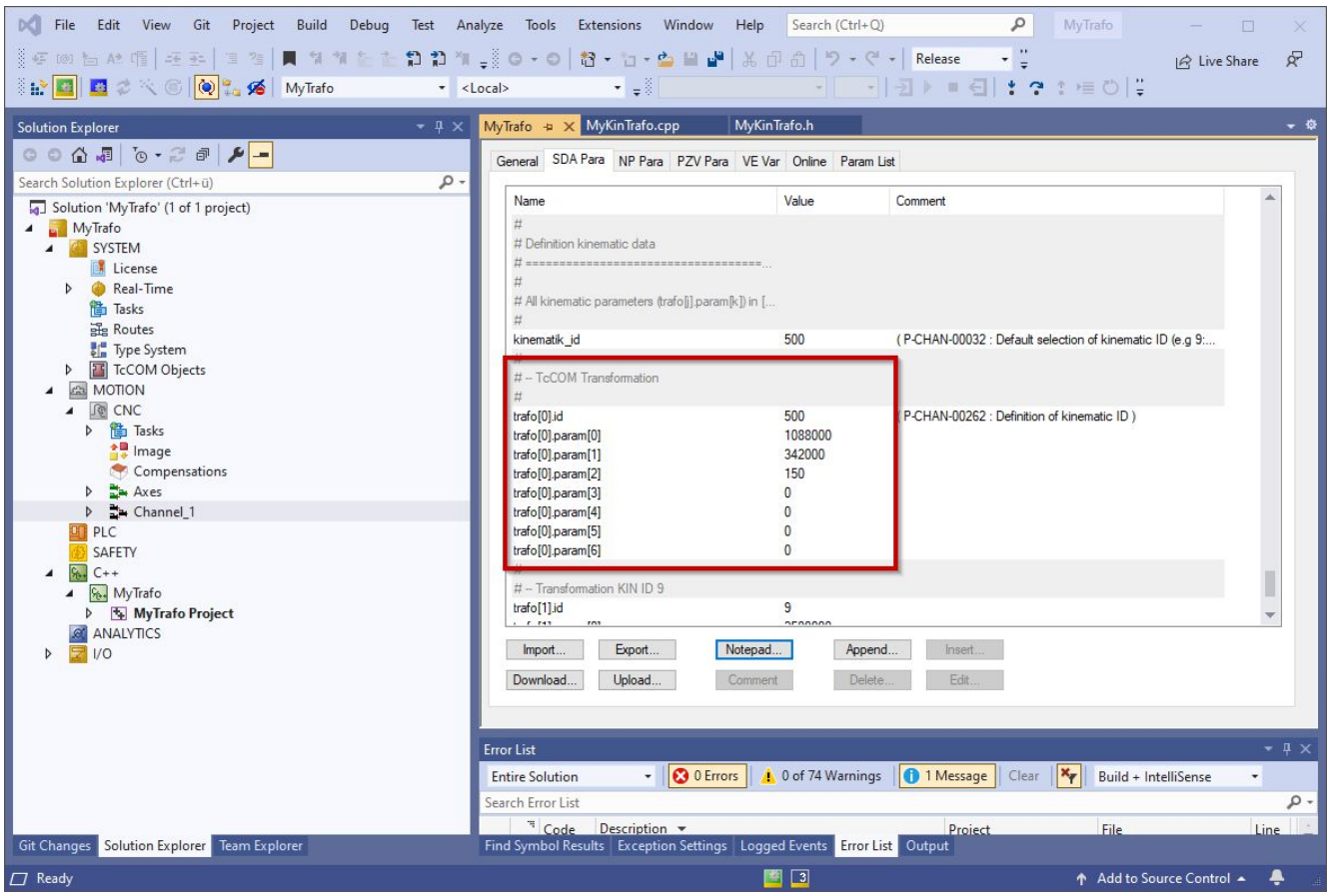

Fig. 13: Channel transformation parameter

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.

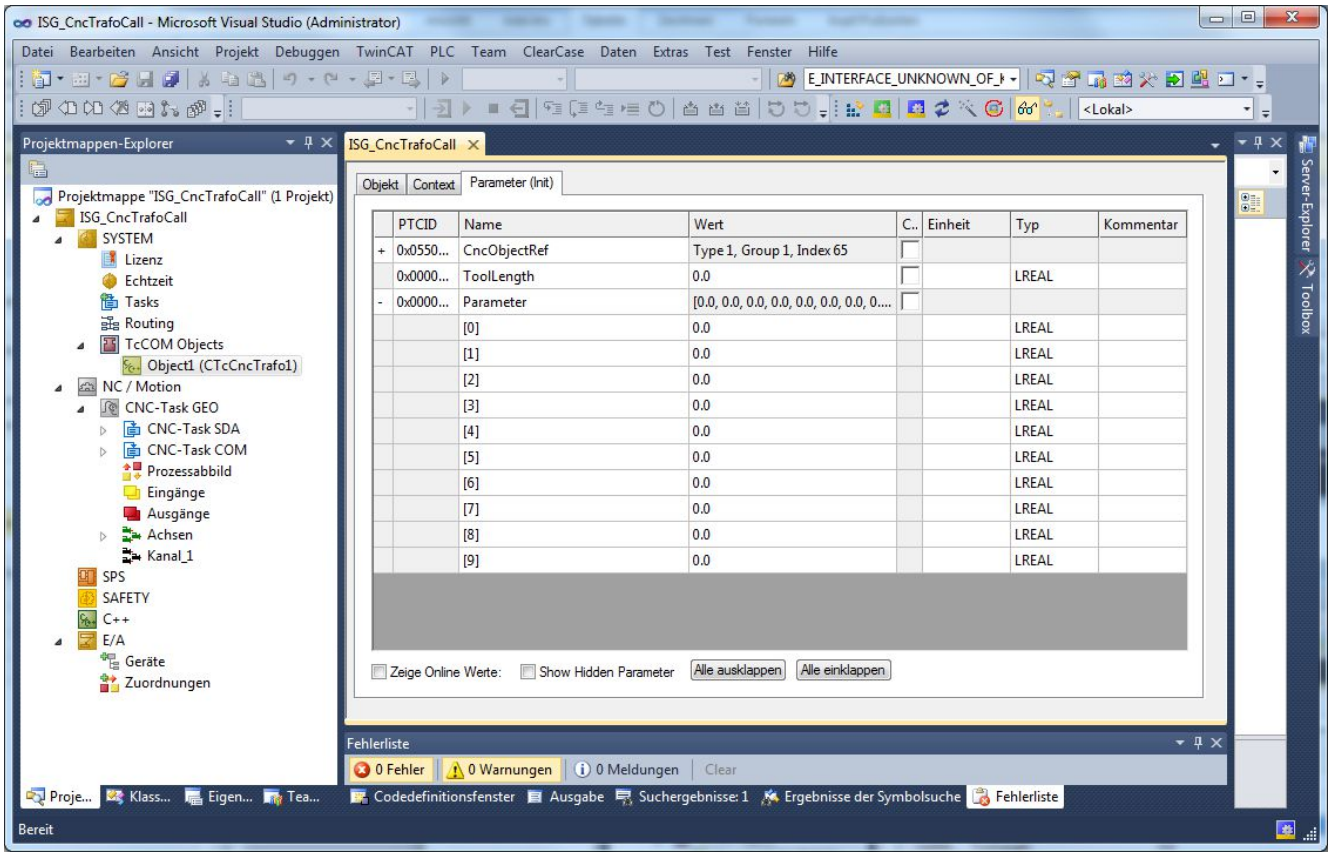

Fig. 14: Transformation parameters via TcCOM

The TcCOM parameters required for the transformation can be defined in the TMC Editor:

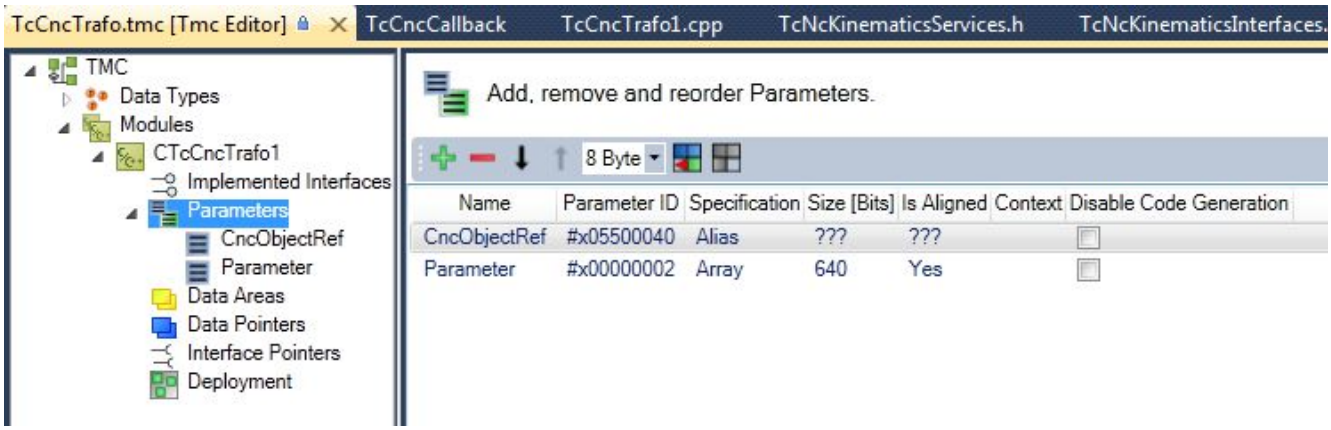

Fig. 15: TMC Editor

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.

```
///<AutoGeneratedContent id="Members">
  TcCncRegisterObject m_CncObjectRef;
  double m_Parameter[10];
///</AutoGeneratedContent>
```

4 Error handling and diagnosis

4.1 Error message

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

Example

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
…
  return S_OK;
}
```

Error messages in TcCncUsersEvents.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):

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

The error is output by the 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.

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

Example

User-specific errors

```
<<-------------------------------------------------------------
---
20.06.2013 16:31:06:019 (11862)                        Ver-
sion: V3.00.3017.00
-------------------------------------------------------------
---
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   : 65
Class   : ERROR (5)                   Reaction : PROGRAM_ABORT
(2)
=============================================================
===
Value 1 : Actual value         : 65
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:
V3.00.3017.00
-------------------------------------------------------------
---
Error   : 292036 - Current WPCS output position of the kin-
ematic forward 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   : 000 [mm]
```

```
Class   : WARNING (0)                    Reaction : PRO-
GRAM_ABORT (2)
================================================================
===
Value 1 : Actual value         : 0 / 1.05E+005 / 0 [0.1*10^-3
mm or ▒]
Value 2 : Actual value         : 0 / 0 / 0 [0.1*10^-3 mm or ▒]
Value 3 : Actual value         : 0 / 0 / 0 [0.1*10^-3 mm or ▒]
Value 4 : Actual value         : 0 / 0 / 0 [0.1*10^-3 mm or ▒]
Value 5 : Actual value         : 0 / 0 / 0 [0.1*10^-3 mm or ▒]
----------------------------------------------------------------
---
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)
```

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

Example

Logging in diagnostic data

```
DECODER : KIN TRAFO IO DIAGNOSIS DATA CHANNEL NO.:  1
CYCL_COUNTER : 274703
================================================================
===
    TIME ID0
ID1         IN[00]         IN[01]         IN[02]         IN[03]
  250907  87   0         0.000         0.000
0.000         0.000
  250908  87   0         0.000         0.000
0.000         0.000
  250909  87   0         0.000         0.000
0.000         0.000
  250939  87   0    822000.000         0.000         0.000
-814000.000
  250946  86   0    822000.000         0.000         0.000
-814000.000
  250947  86   0    822000.000         0.000         0.000
-814000.000
  250956  86   0    800000.000         0.000         0.000
-300000.000


PATH : KIN TRAFO IO DIAGNOSIS DATA CHANNEL NO.:  1
CYCL_COUNTER : 263179
```

```
=================================================================
===
    TIME ID0
ID1        IN[00]        IN[01]        IN[02]        IN[03]….
  263153  87   0    753617.664   -376316.335         0.000
-197893.497
  263154  87   0    754593.192   -374570.386         0.000
-196254.241
  263155  87   0    755561.728   -372820.548         0.000
-194608.676
  263156  87   0    756523.257   -371066.850         0.000
-192956.796
  263157  87   0    757477.763   -369309.320         0.000
-191298.596
  263158  87   0    758425.232   -367547.986         0.000
-189634.068
  263159  87   0    759365.648   -365782.876         0.000
-187963.208
  263160  87   0    760298.996   -364014.018         0.000
-186286.009
  263161  87   0    761225.261   -362241.442         0.000
-184602.466
  263162  87   0    762144.428   -360465.174         0.000
-182912.574
  263163  87   0    763056.483   -358685.244         0.000
-181216.327
  263164  87   0    763961.411   -356901.681         0.000
-179513.720
  263165  87   0    764859.197   -355114.511         0.000
-177804.748
  263166  87   0    765749.828   -353323.765         0.000
-176089.406
  263167  87   0    766633.289   -351529.471         0.000
-174367.689
  263168  87   0    767509.565   -349731.657         0.000
-172639.592
  263169  87   0    768378.643   -347930.352         0.000
-170905.111
  263170  87   0    769240.509   -346125.586         0.000
-169164.241
  263171  87   0    770095.149   -344317.386         0.000
-167416.979
  263172  87   0    770942.549   -342505.783         0.000
-165663.319
```

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)

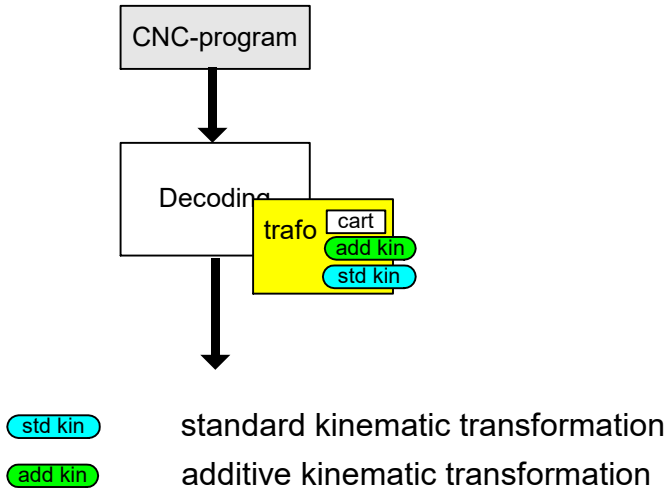

Fig. 16: Concatenating kinematic transformations

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.,

```
N10 V.G.KIN_STEP[1].ID[1].PARAM[0] = 55000
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:

```
#TRAFO [<kin-id-step0>, <kin-id-step1>]
#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. Min. Visual Studio 2010 installed with Service Pack 1
3. WinDDK 2010 must be installed. The environment variable WINDDK7 must be assigned to the DDK directory (normally C:\WinDDK\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

Notice

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

The procedure below is an example of creating a user-defined kinematic transformation using Visual Studio 2019,

Step 1- TwinCAT3 XAE project with CNC configuration

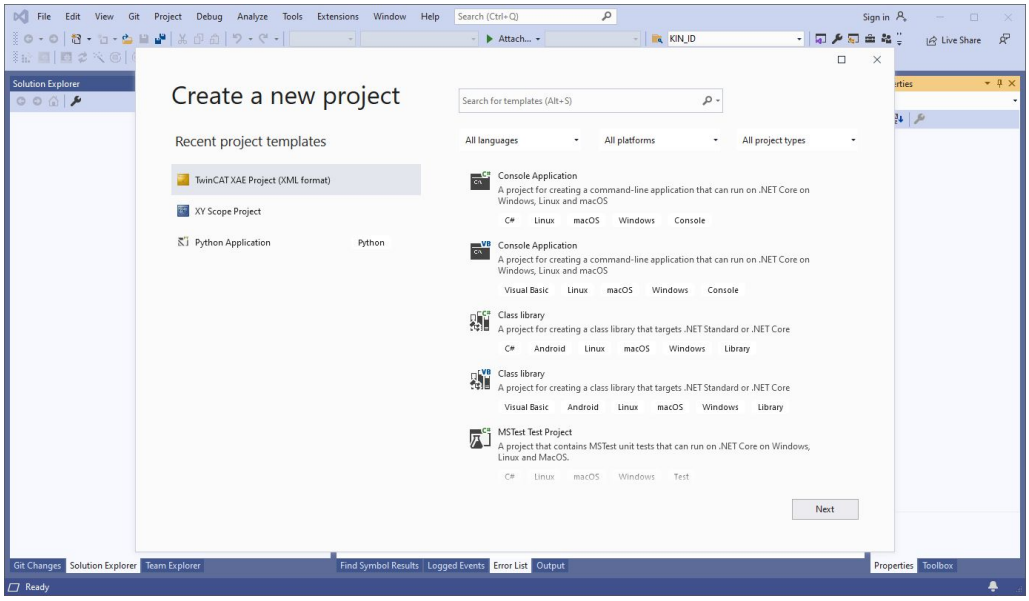

Fig. 17: Create new project

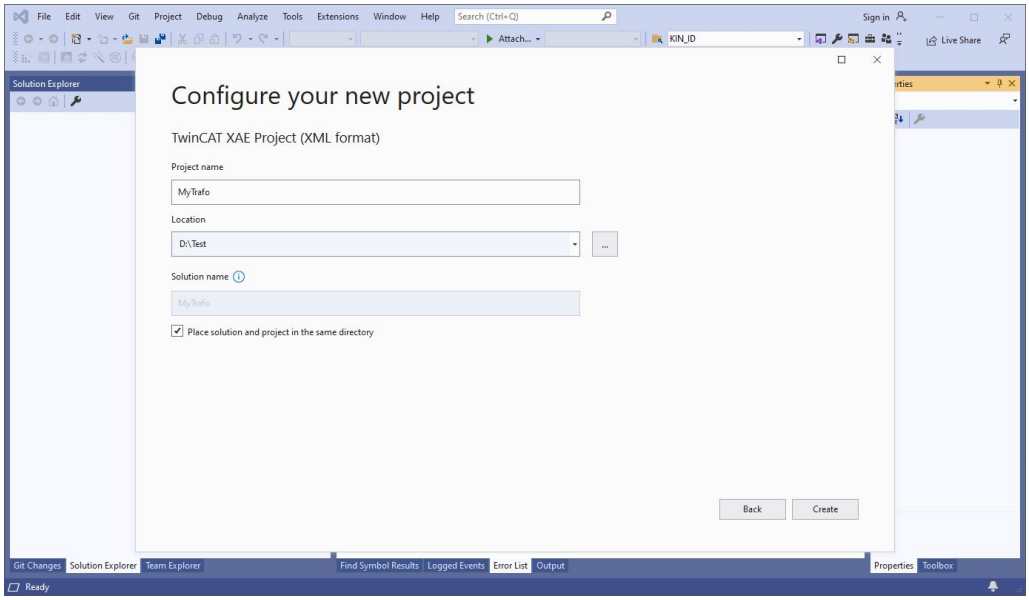

Fig. 18: Configure new project

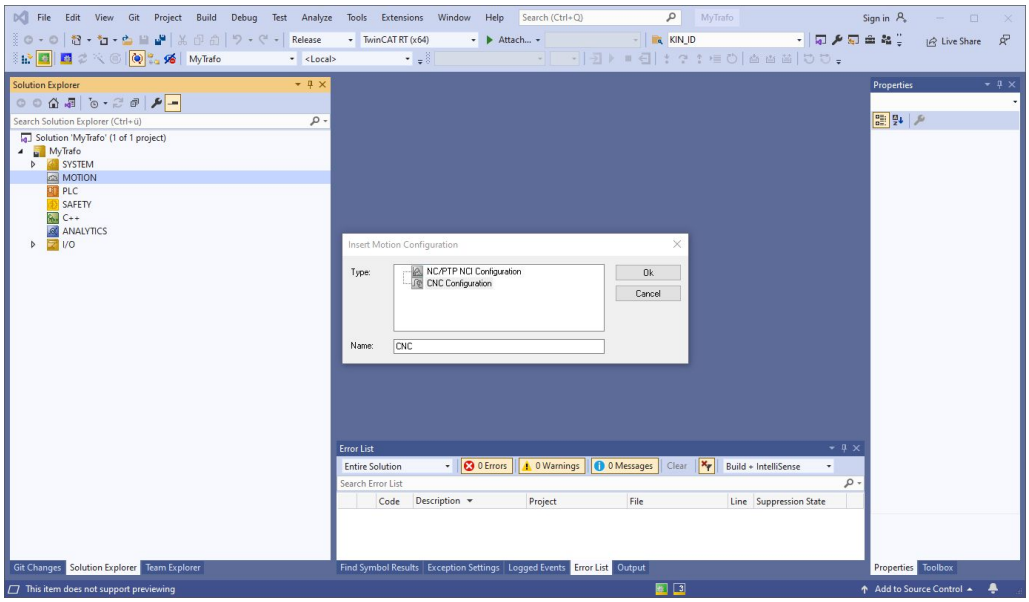

Fig. 19: Generate CNC configuration

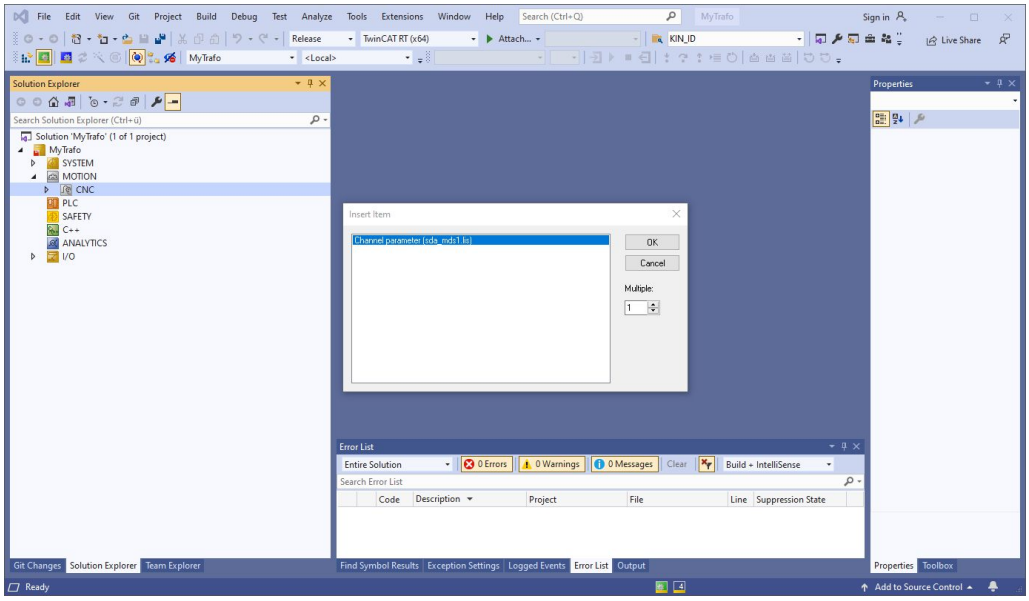

Fig. 20: Create channel

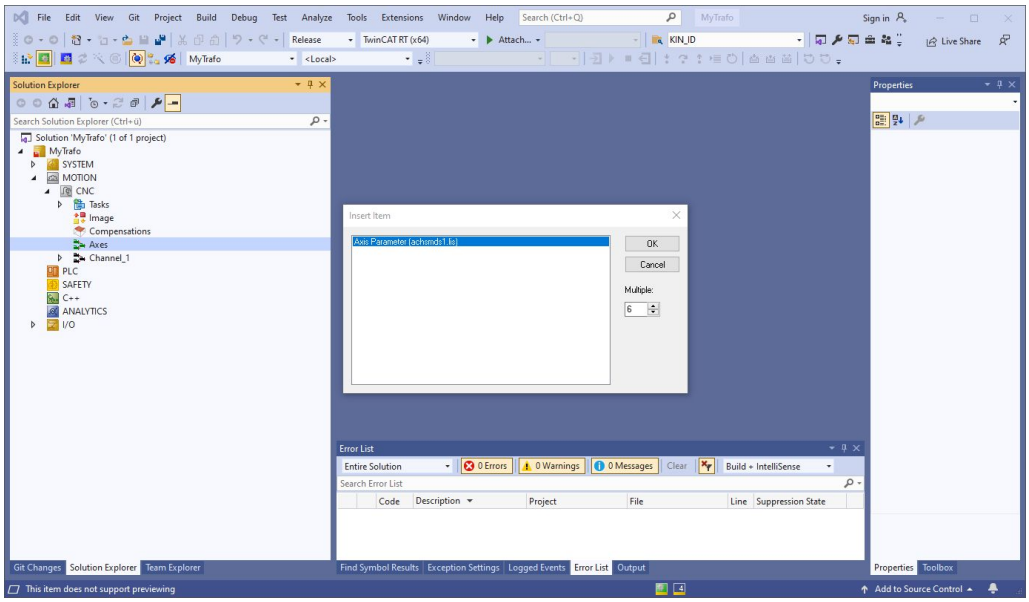

Fig. 21: Create axes

Step 2: Create user transformation using TwinCAT3 templates.

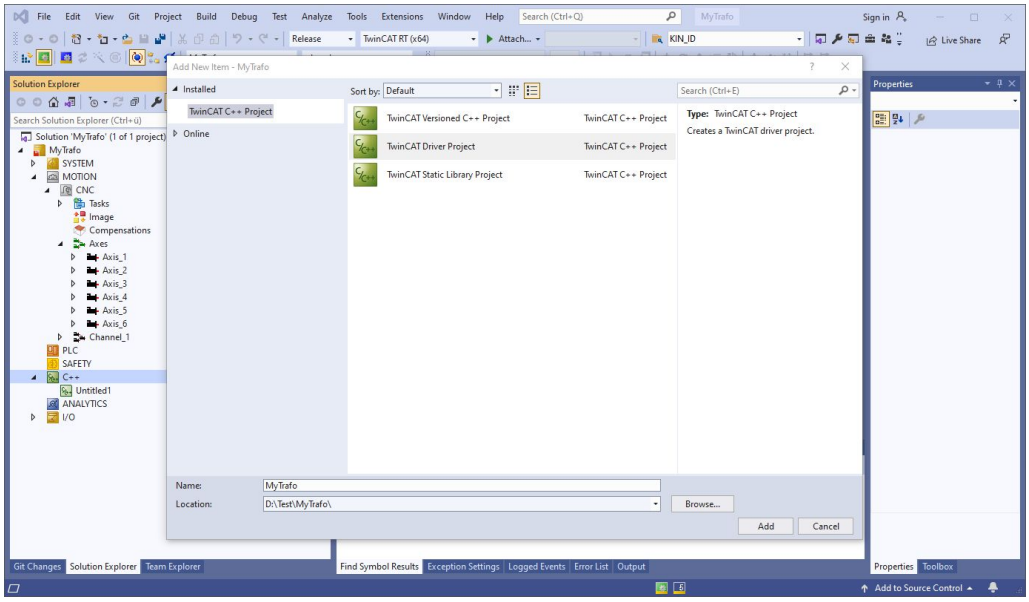

Fig. 22: Create TwinCAT driver project

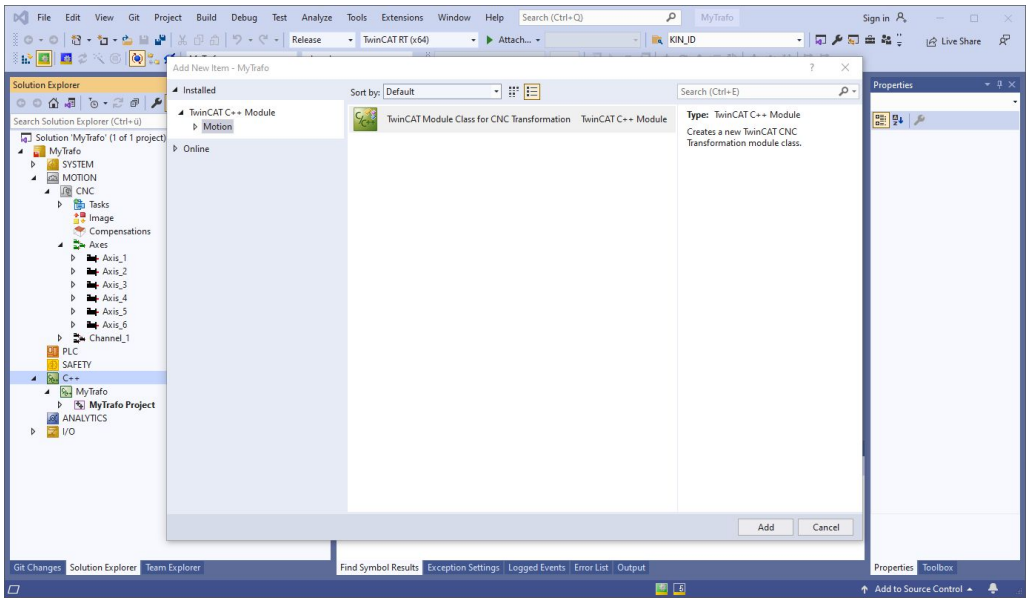

Fig. 23: Create transformation class

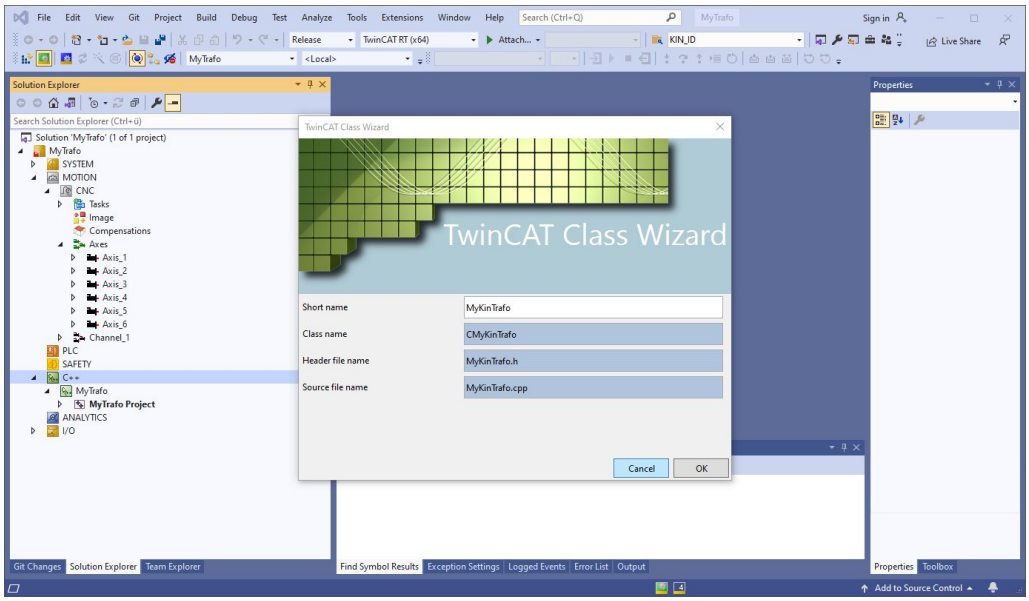

Fig. 24: Name transformation class

This defines the framework for the TcCOM object in Visual Studio.

Step 3: Create the driver

Right-click on the project to “Create” the driver.

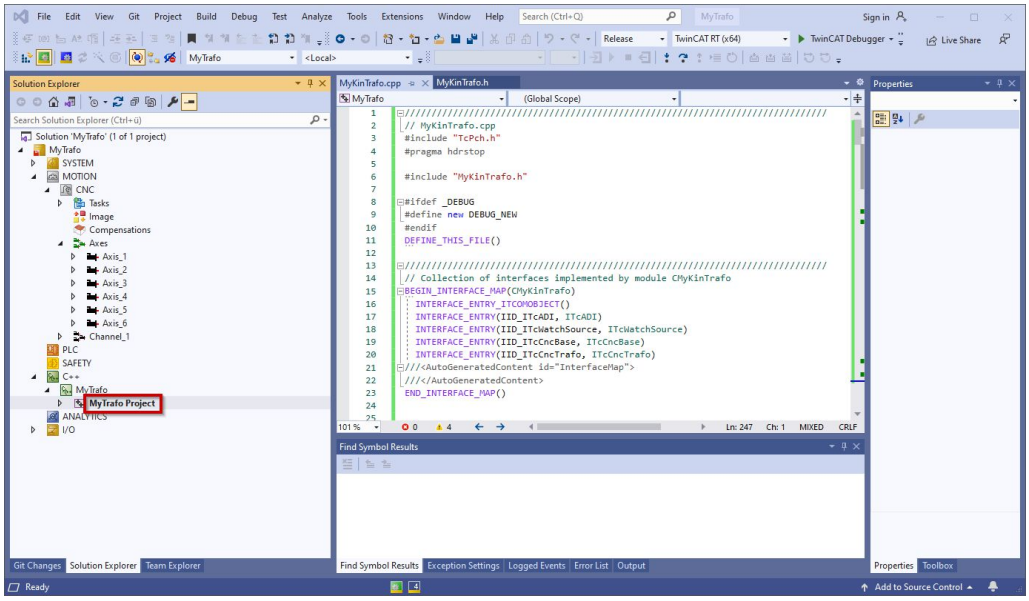

Fig. 25: Create driver

7.2.2 Integrate transformation

Integrate the transformation into the existing CNC configuration as follows:

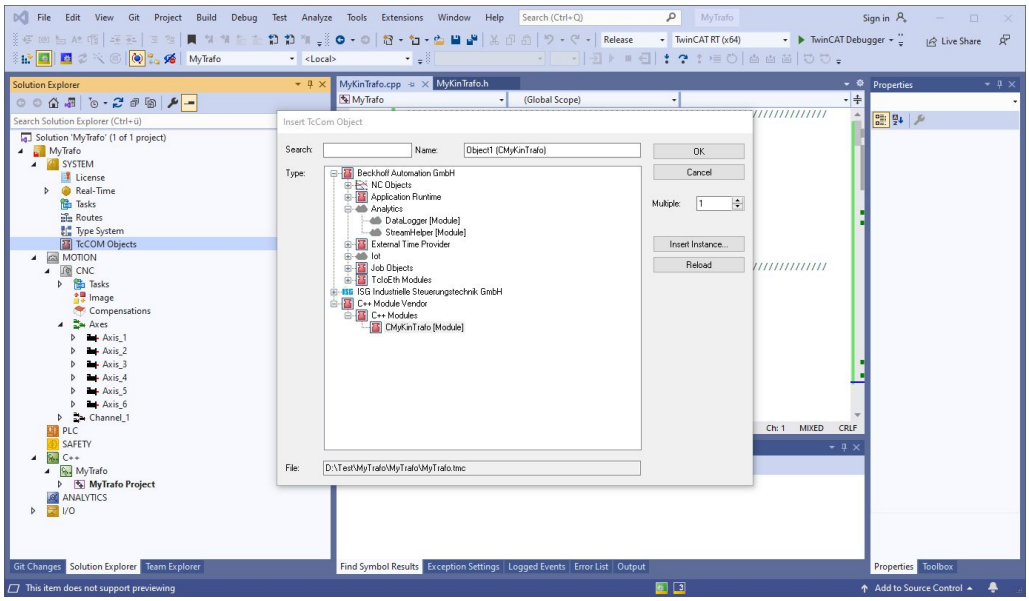

Fig. 26: Integrate TcCOM object

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

Double-click on the TcCOM object to display the properties.

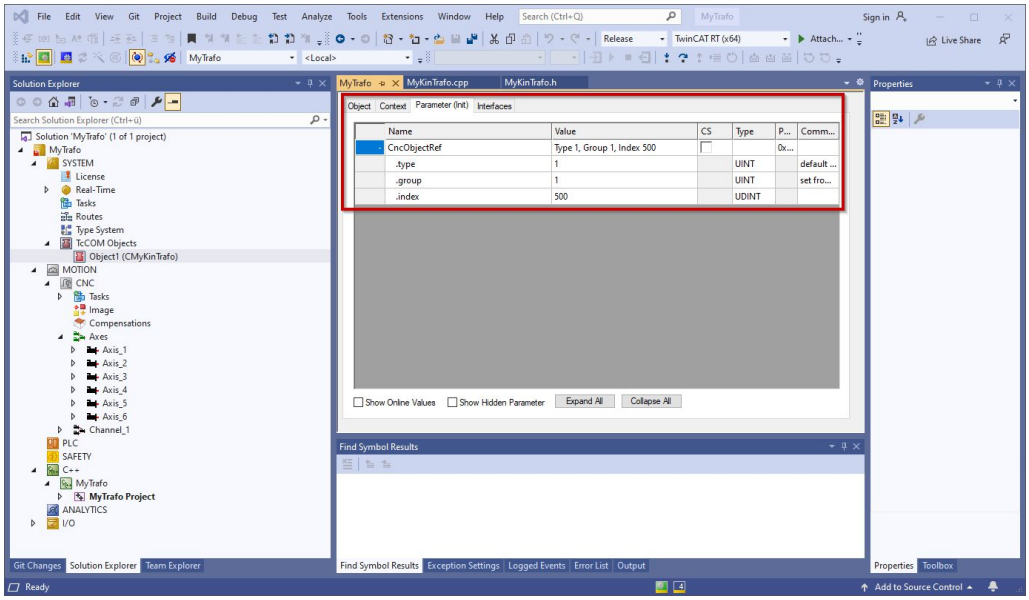

Fig. 27: Properties of the TcCOM object

Parameter	Permissible values	Description
Type	1	Type = 1 specifies that the TcCOM object is a kinematic transformation.
Group	0 <= group <= number of channels	Which CNC channel may access the transformation is specified in the group parameter. If group = 0, the transformation is available for all CNC channels.
Index	65 <= index <= 69 or 500 <= index <= 999	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].

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.

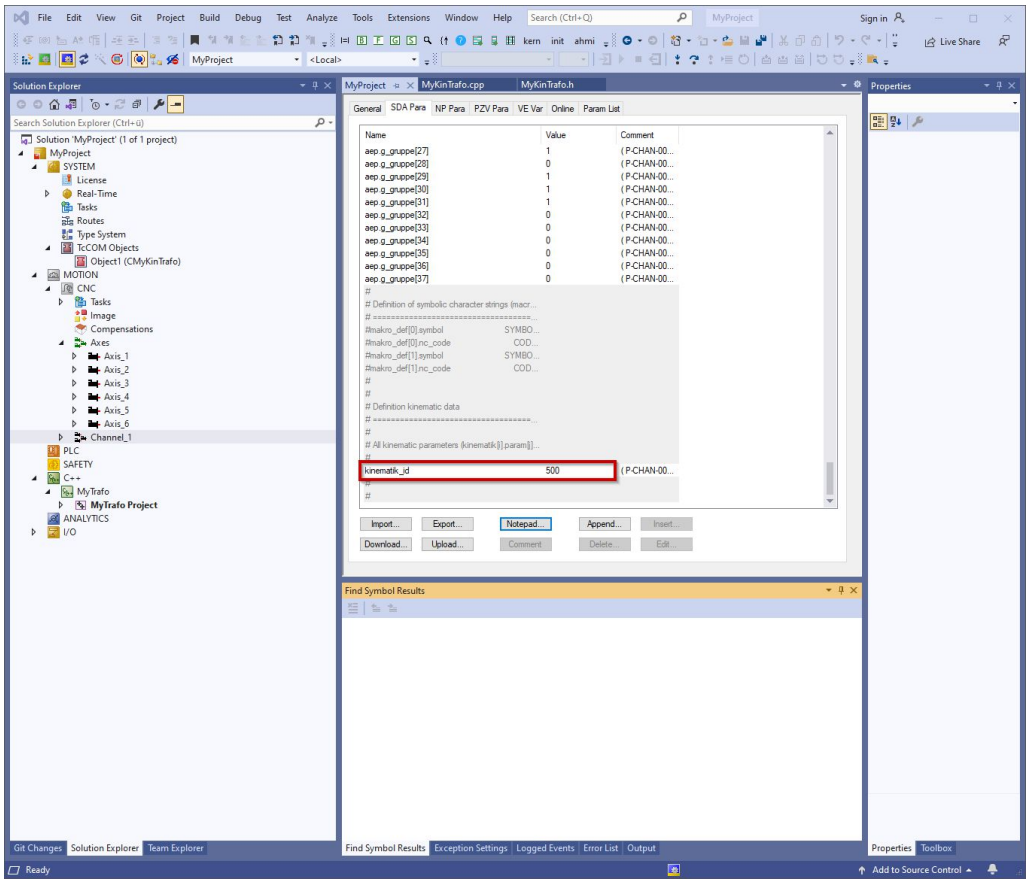

Fig. 28: Parameterise the transformation in the channel parameter list

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

7.2.3 Debugging the transformation

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

Fig. 29: Switch over to debug configuration

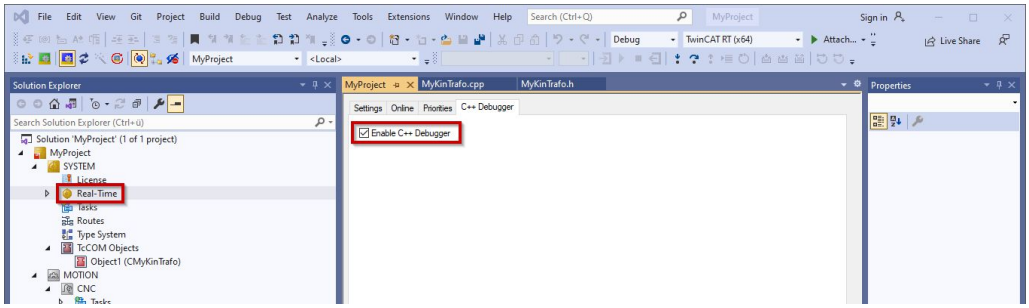

Fig. 30: Activate real-time debugging

Notice

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.

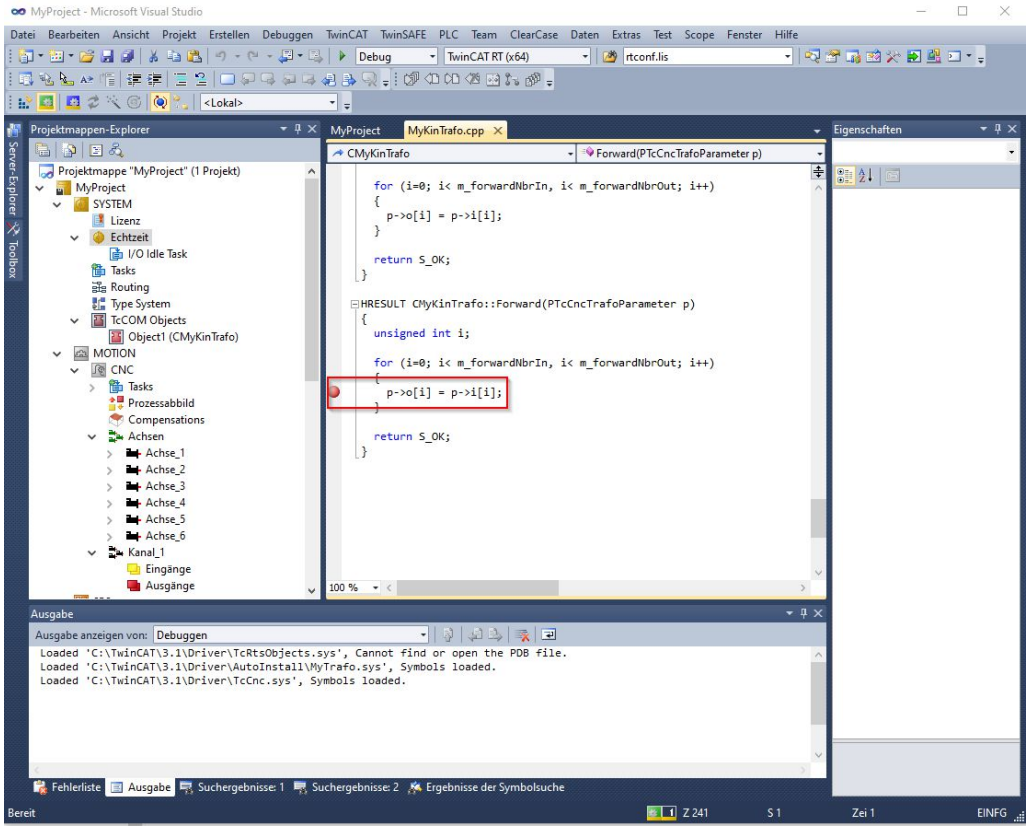

Fig. 31: Breakpoint in the transformation

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.

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

Fig. 32: Setting the constructor after generation using TwinCAT3 templates

```
////////////////////////////////////////////////////////////////////////////////
// Constructor
CMyKinTrafo::CMyKinTrafo(): m_forwardNbrIn(7), m_forwardNbrOut(7)
{
///<AutoGeneratedContent id="MemberInitialization">
```

Fig. 33: 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)

Extended transformation	**Standard transformation**
Extended interface: **ITcCncTrafo** GUID extended interface: **IID_ITcCncTrafo**	Default interface: **ITcNcTrafo** GUID standard interface: **IID_ITcNcTrafo**
Extended transformation parameter: **PTcCncTrafoParameter**	Default parameter: **PTcNcTrafoParameter**

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].

9 Additional options of extended transformation

9.1 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.

Example

HRESULT <UserTrafo>::TrafoSupported(PTcCncTrafoParameter p, bool fwd)

```
{
  ...
  TcCncVersion TcCOMInterfaceVersion;
  this->GetInterfaceVersion(&TcCOMInterfaceVersion);

  if (   (TcCOMInterfaceVersion.major <= p->CncInterfaceVer-
sion.major)
      && (TcCOMInterfaceVersion.minor <= p->CncInterfaceVer-
sion.minor))
  {
    return S_OK;
  };

}
```

9.2 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:

p->actual_orientation_mode	Meaning
EcCncTrafoOri_None	No rotation
EcCncTrafoOri_YPR	Yaw-Pitch-Roll rotation sequence: 1st rotation about Z, 2nd negative rotation about Y, 3rd rotation about X
EcCncTrafoOri_CBC1	Euler rotation sequence: 1st rotation about Z, 2nd rotation about Y, 3rd rotation about Z‘
EcCncTrafoOri_CBA	1st rotation about Z, 2nd rotation about Y, 3rd rotation about X
EcCncTrafoOri_CAB	1st rotation about Z, 2nd rotation about X, 3rd rotation about Y
EcCncTrafoOri_AB	1st rotation about X, 2nd rotation about Y
EcCncTrafoOri_BA	1st rotation about Y, 2nd rotation about X

TcCOM transformation --> CNC:

p->supported_orientation_modes	Meaning
.f_YPR	= TRUE, transformation supports rotation sequence YPR
.f_CBC1	= TRUE, transformation supports rotation sequence CBC‘
.f_CBA	= TRUE, transformation supports rotation sequence CBA
.f_CAB	= TRUE, transformation supports rotation sequence CAB
.f_AB	= TRUE, transformation supports rotation sequence AB
.f_BA	= TRUE, transformation supports rotation sequence BA

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.

Example

```
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;
}

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.

p->mcs_modulo[i]	Meaning
EcCnc_McsModulo_None	Linear MCS positions, no modulo calculation for this axis
EcCnc_McsModulo_180_180	Modulo calculation of the MCS positions for this axis in the interval [-180°, +180°[.

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.

p->acs_modulo[i]	Meaning
EcCnc_AcsModulo_None	Linear ACS positions: no modulo handling is required for this axis.
EcCnc_AcsModulo_180_180	For this axis a modulo calculation of the ACS positions is required in the interval [-180°, +180°[.
EcCnc_AcsModulo_0_360	For this axis a modulo calculation of the ACS positions is required in the interval [0°, 360°[.

Programing Example

Modulo handling of axis positions

```
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.

```
HRESULT <UserTrafo>::TrafoSupported(PTcNcTrafoParameterExtCnc p, bool fwd)
{
  if ( p == NULL )
  {
    return E_POINTER;
  }
  if ( p->type != EcNcTrafoParameter_ExtCnc )
  {
    p->ret_value1 = (double)p->type;
    strcpy(p->ret_text, "EcNcTrafoParameter");
    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 == EcCncTrafoCallerID_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.

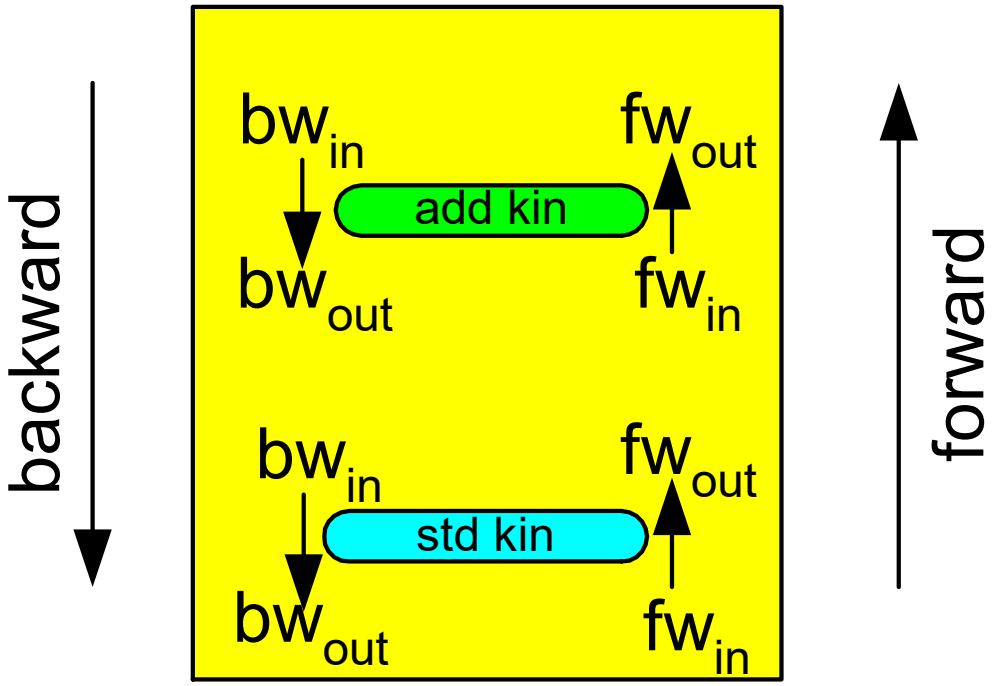

Fig. 34: Adapting the number of inputs/outputs

The transformation options can be set during a configuration request (TafoSupported method).

```
HRESULT <UserTrafo>::TrafoSupported(PTcNcTrafoParameterExtCnc 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.).

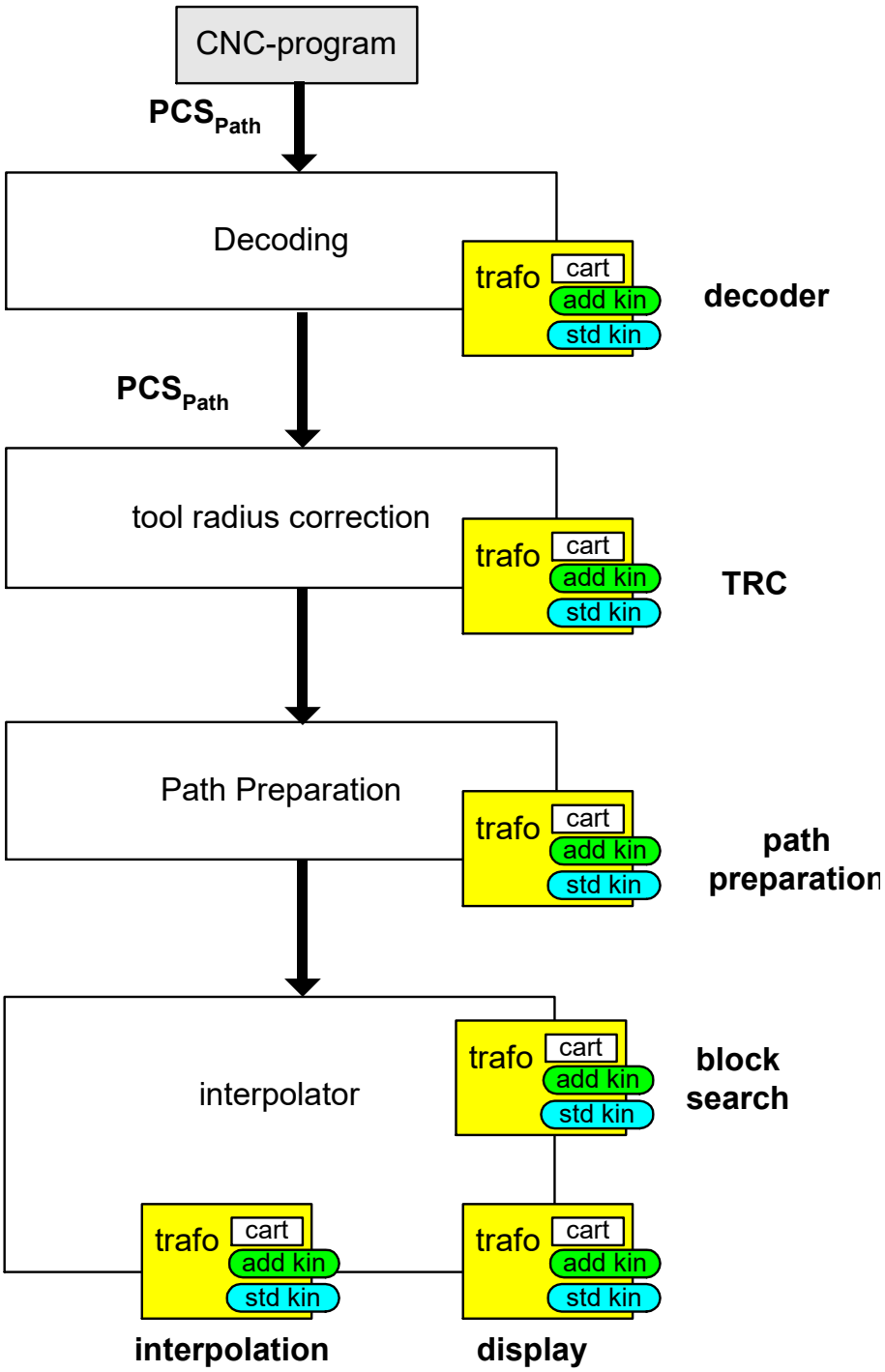

Fig. 35: Interfaces for adaptation to various callers.

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`

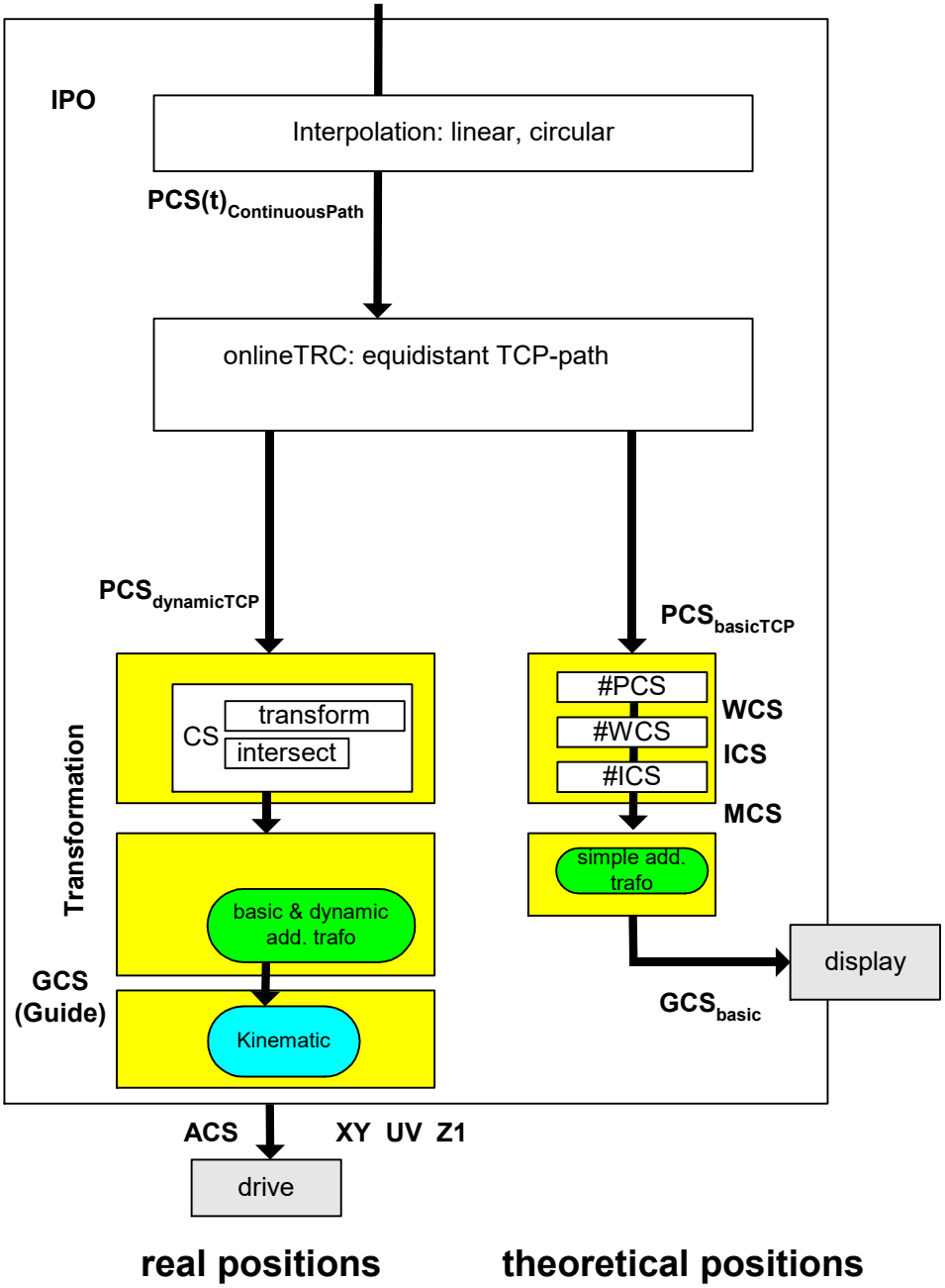

Fig. 36: Displaying the additive transformation position

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

11 Appendix

11.1 Suggestions, corrections and the latest documentation

Did you find any errors? Do you have any suggestions or constructive criticism? Then please contact us at documentation@isg-stuttgart.de. The latest documentation is posted in our Online Help (DE/EN):

QR code link: https://www.isg-stuttgart.de/documentation-kernel/

The link above forwards you to:

https://www.isg-stuttgart.de/fileadmin/kernel/kernel-html/index.html

Notice

Change options for favourite links in your browser;

Technical changes to the website layout concerning folder paths or a change in the HTML framework and therefore the link structure cannot be excluded.

We recommend you to save the above "QR code link" as your primary favourite link.

PDFs for download:

DE:
https://www.isg-stuttgart.de/produkte/softwareprodukte/isg-kernel/dokumente-und-downloads

EN:
https://www.isg-stuttgart.de/en/products/softwareproducts/isg-kernel/documents-and-downloads

E-Mail: documentation@isg-stuttgart.de

Index

ISG Industrielle Steuerungstechnik GmbH
STEP, Gropiusplatz 10
D-70563 Stuttgart

www.isg-stuttgart.de
support@isg-stuttgart.de