DOCUMENTATION ISG-kernel

McCOM - Online adaptation

Short description:
McCOM - OAD

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

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

Documentation version: 1.0
20/10/2025

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

ISG®, ISG kernel®, ISG virtuos®, ISG dirigent®, TwinStore® 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

Preface **2**

General and safety instructions **3**

1 Overview **7**

2 Online tool radius compensation **8**

2.1 Overview 8
2.2 Description 9
2.2.1 Overview of commands 13
2.2.2 Properties of tool radius compensation 14
2.2.2.1 Reaction with block transition at angles greater than 180° 15
2.2.2.2 Reaction with block transitions at angles less than 180° 18
2.3 Adding Online Tool Radius Compensation via TcCOM 20
2.3.1 Interface methods 20
2.3.2 Instance data 20
2.3.3 Configuring and registering a TcCOM object 22
2.4 Programming 23
2.4.1 TRC options for online TRC and 2-path 24
2.4.2 TRC option GEN_CIR_BLOCK_IN_CORNER 27
2.4.3 TRC option G236_LIN 27
2.4.4 TRC option PERPENDICULAR_RADIUS_CHANGE 28
2.5 Parameter 33
2.6 Error messages 34

3 Geometric feed adaptation **35**

3.1 Overview 35
3.2 Description 36
3.2.1 Integration in the NC channel 36
3.3 Adding a geometric feed adaptation via TcCOM 38
3.3.1 Interface methods 38
3.3.2 Instance data 38
3.3.3 Configuring and registering a TcCOM object 39
3.4 Programming 40
3.5 Parameter 41
3.6 Error messages 43

4 Dynamic Contour Control **44**

4.1 Overview 44
4.2 Description 45
4.2.1 External calculation mode 47
4.3 Programming 49
4.4 Adding Dynamic Contour Control via TcCOM 50
4.4.1 Interface of the methods 50
4.4.2 Instance data 51
4.4.3 Configuring and registering a TcCOM object 52
4.5 Parameter 53
4.5.1 Parameterisation example 54

4.6 Error messages ... 55
5 Process to generate a TcCOM object ... 56
5.1 Create new project ... 56
5.2 Geometric feed adaptation ... 59
5.2.1 Create an object for geometric feed adaptation ... 59
5.2.2 Integrate geometric feed adaptation object ... 61
5.3 Dynamic Contour Control ... 63
5.3.1 Create an object for Dynamic Contour Control ... 63
5.3.2 Integrate Dynamic Contour Control object ... 65
5.4 Online tool radius compensation ... 68
5.4.1 Creating an object for online tool radius compensation ... 68
5.4.2 Integrating online tool radius compensation object ... 71
6 Appendix ... 73
6.1 Suggestions, corrections and the latest documentation ... 73
Index ... 74

List of figures

Fig. 1: Placing online tool radius compensation in the NC channel ... 9
Fig. 2: Principle sequence in the NC channel ... 10
Fig. 3: 2-path programming flow with online TRC ... 11
Fig. 4: 2-path application with inclined wire ... 12
Fig. 5: Workpiece with static tool radius compensation ... 14
Fig. 6: Tangent rotation at the outside corners of a geometry ... 15
Fig. 7: Online TRC path with inclined tool at an outside corner ... 16
Fig. 8: Changing the tool radius at an outside corner ... 17
Fig. 9: Response at the inside corner of a geometry at block transitions with a angle of 90° ... 18
Fig. 10: Change in tool radius at the inside corner of a geometry at block transitions with an angle of 180° ... 19
Fig. 11: Orthogonal extension of a tool radius change ... 28
Fig. 12: Geometric feed adaptation in the NC channel ... 36
Fig. 13: Integrating the object in the NC channel ... 37
Fig. 14: Geometric mode of operation of programmed and corrected path ... 45
Fig. 15: Arrangement of Dynamic Contour Control within the system ... 46
Fig. 16: Dynamic Contour Control with implicit superimposition ... 47
Fig. 17: Dynamic Contour Control with explicit offset superimposition in the first kinematic step ... 48
Fig. 18: Create a new project ... 56
Fig. 19: Configure the new project ... 57
Fig. 20: Create a CNC configuration ... 57
Fig. 21: Create a channel ... 58
Fig. 22: Create an axis ... 58
Fig. 23: Drive project for geometric feed adaptation ... 59
Fig. 24: Define class ... 59
Fig. 25: Name class ... 60
Fig. 26: Create driver ... 60
Fig. 27: Integrating the TcCOM object ... 61
Fig. 28: Properties of the TcCOM object ... 61
Fig. 29: Create driver project for Dynamic Contour Control ... 63
Fig. 30: Define class ... 64
Fig. 31: Name class ... 64
Fig. 32: Create driver ... 65
Fig. 33: Integrating the TcCOM object ... 65
Fig. 34: Properties of the TcCOM object ... 66
Fig. 35: Driver project for online tool radius compensation ... 68
Fig. 36: Define transformation class ... 69
Fig. 37: Name transformation class ... 69
Fig. 38: Create driver ... 70
Fig. 39: Integrating the TcCOM object ... 71
Fig. 40: Properties of the TcCOM object ... 71

1 Overview

This document is a summary of the following three functionalities:

- Online tool radius compensation
- Dynamic Contour Control
- Geometric feed adaptation

Task

Online tool radius compensation permits the integration of technology-specific tool compensation in real time. It is specially designed for EDM wire erosion with an inclined wire in 2-path applications.

Dynamic Contour Control (DCC) compensates for deviations between the programmed and the actually produced contour, for example, resulting from the physical deformations of an erosion wire. The aim is to adapt the actual contour to the nominal contour.

Geometric feed adaptation allows the user to influence the feed rate by using custom (geometry-dependent) algorithms. With EDM wire erosion, this is specially designed to ensure constant surface removal.

Properties

Online tool radius compensation: Only applies when tool radius compensation (G41/G42) is enabled. Requires a C2 continuous parallel path taken from static tool radius compensation. Supports 2-path configuration with independent dynamic planning and synchronised output.

Dynamic Contour Control: Modifies the interpolated tool centre path as a function of current and previous contour elements, tool parameters and PLC influence (e.g. velocity dependency).

Geometric feed adaptation: Calculates a factor that is multiplied by the current command velocity and the override factor. The calculation is based on a cyclically corrected radius centre point path where the tool radius consists of the actual tool radius and a gap value.

Programming

Please refer to the relevant sub-sections for a description of the exact programming:

- Online tool radius compensation [▷ 23]
- Dynamic Contour Control [▷ 49]
- Geometric feed adaptation [▷ 40]

Mandatory note on references to other documents

For the sake of clarity, links to other documents and parameters are abbreviated, e.g. [PROG] for the Programming Manual or P-AXIS-00001 for an axis parameter.

For technical reasons, these links only function in the Online Help (HTML5, CHM) but not in pdf files since pdfs do not support cross-linking.

2 Online tool radius compensation

2.1 Overview

Task

The online tool radius compensation functionality allows users to integrate technology-specific tool compensation data.

The functionality is particularly useful for EDM wire erosion with an inclined wire in 2-path applications.

Release Note

The functionality is available as of CNC Build V3.1.3108.

Notice

This function is an additional option requiring a license.

Properties

Online TRC (TRC = Tool Radius Compensation) is only enabled when tool radius compensation is active. The functionality is activated by G41 or G42 and deactivated by G40.

Parameter

The P-CHAN-00550 [▶ 33] parameter must be set when a 2-path application is used.

Programming

The functionality is activated and deactivated using the standard tool radius compensation commands (G40/G41/G42). Online tool radius compensation is then defined using the #TRC command [▶ 24].

Links to other documents

For the sake of clarity, links to other documents and parameters are abbreviated, e.g. [PROG] for the Programming Manual or P-AXIS-00001 for an axis parameter.

For technical reasons, these links only function in the Online Help (HTML5, CHM) but not in pdf files since pdfs do not support cross-document linking.

2.2 Description

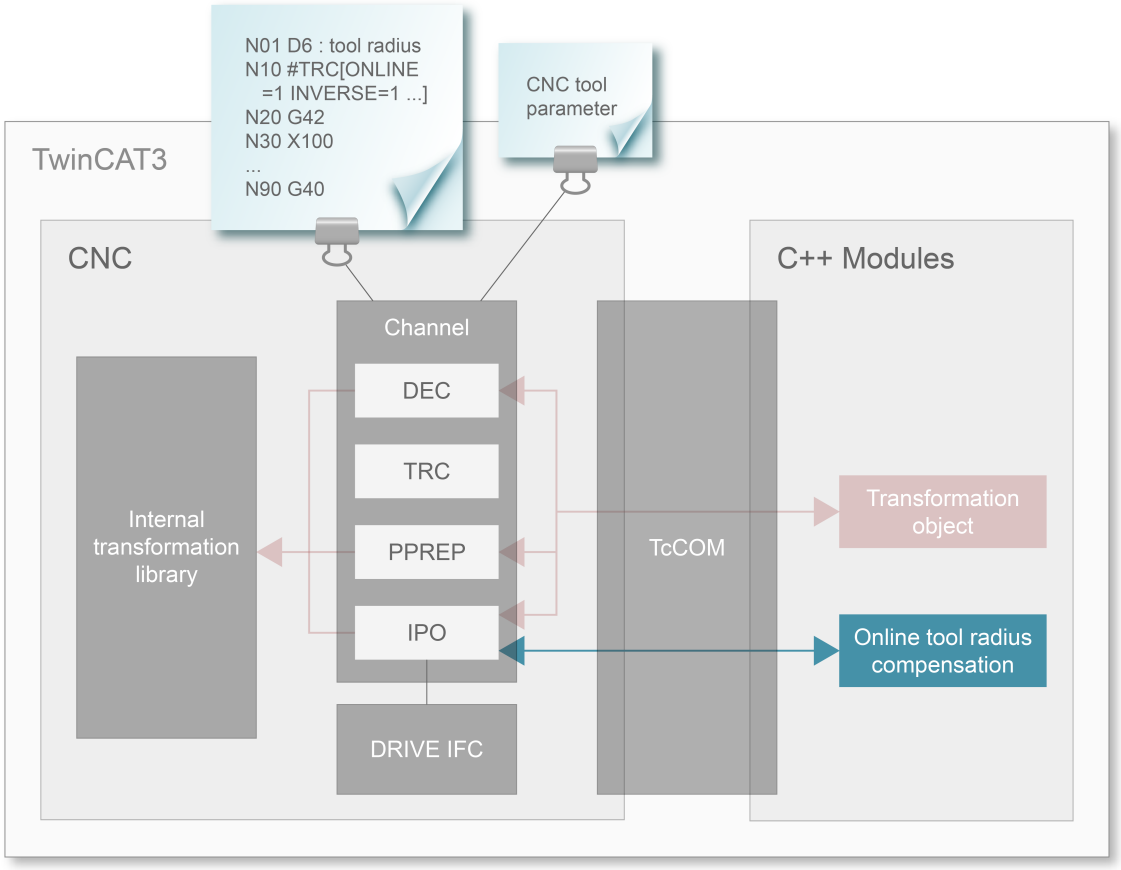


Fig. 1: Placing online tool radius compensation in the NC channel

Prerequisite:

In order to use online tool radius compensation, the generated parallel path of the static tool radius compensation in the non-real-time part of the CNC must supply a C2 continuous contour (i.e. curvature continuous). For this reason, the selection of commands is restricted to static tool radius compensation.

The list of available and impermissible commands for tool radius compensation is in the sub-section Overview of commands [▶ 13].

Sequence in the NC channel

The figures below describe the principle sequence.

The NC program generated is used to determine a parallel path in the non-real-time part of the CNC and for which dynamic planning is carried out:

Dynamic planning using this parallel path is only adequately precise for online TRC for small radii (approx. < 1 mm) and small tool inclinations (approx. < 15 °). A deviating path generated by online TRC is not taken into account by dynamic planning. In extreme cases, this can result in acceleration overshoots.

This dynamically planned path is used to recalculate the programmed path in the real-time part of the CNC. This is then followed by the online tool radius compensation that calculates the compensation data in each cycle.

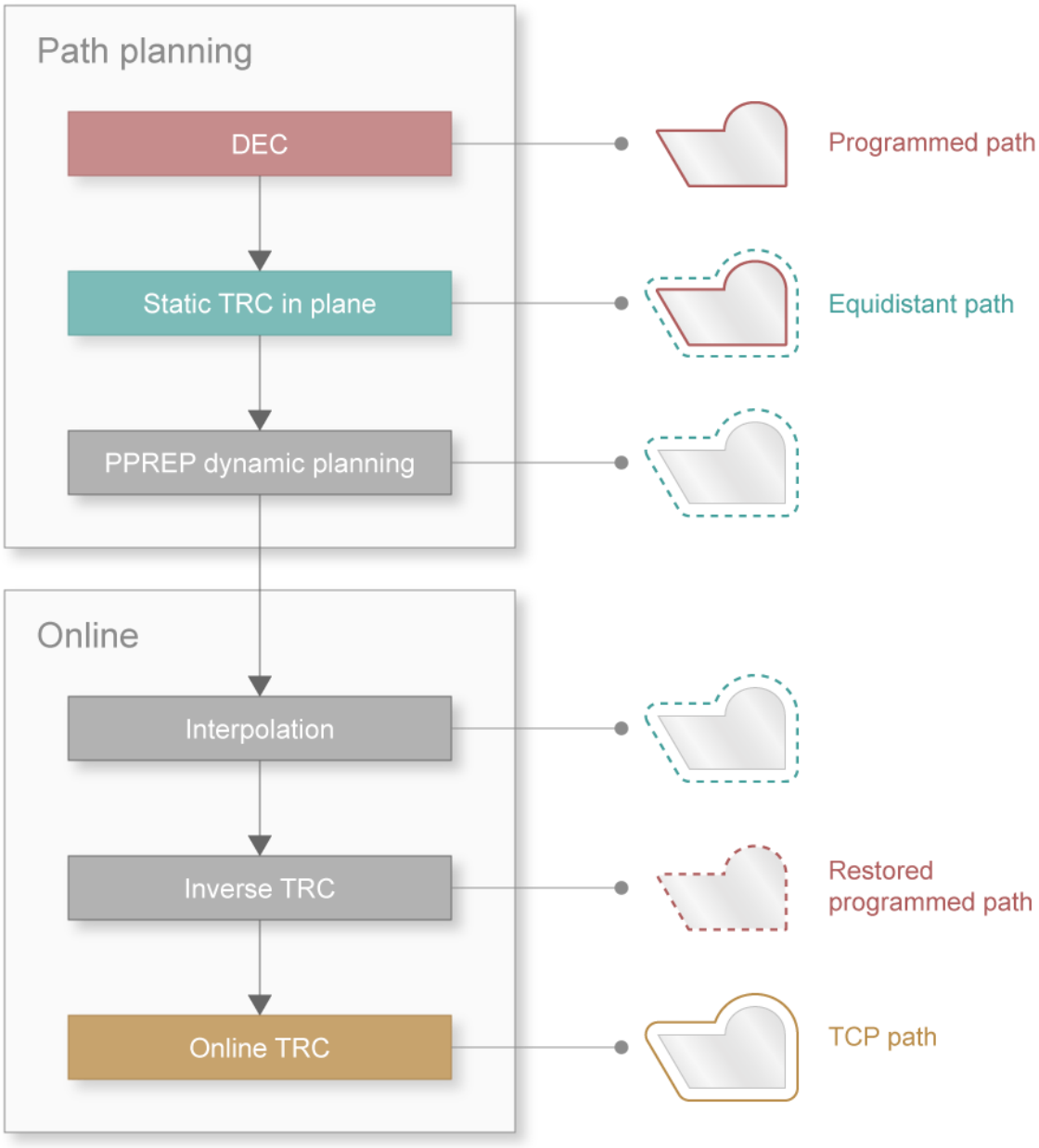


Fig. 2: Principle sequence in the NC channel

Multi-path configuration and tool radius compensation

Tool radius compensation allows 2 paths to be programmed in an NC line, see the PROG section: 2-path programming. The parallel path and dynamic planning are executed in each path independently of any other path. At the time of interpolation, the two paths are output again synchronised.

This takes place provided the parameter P-CHAN-00550 [▶ 33] is set:

```
configuration.tool_radius_comp.function MULTI_PATH
```

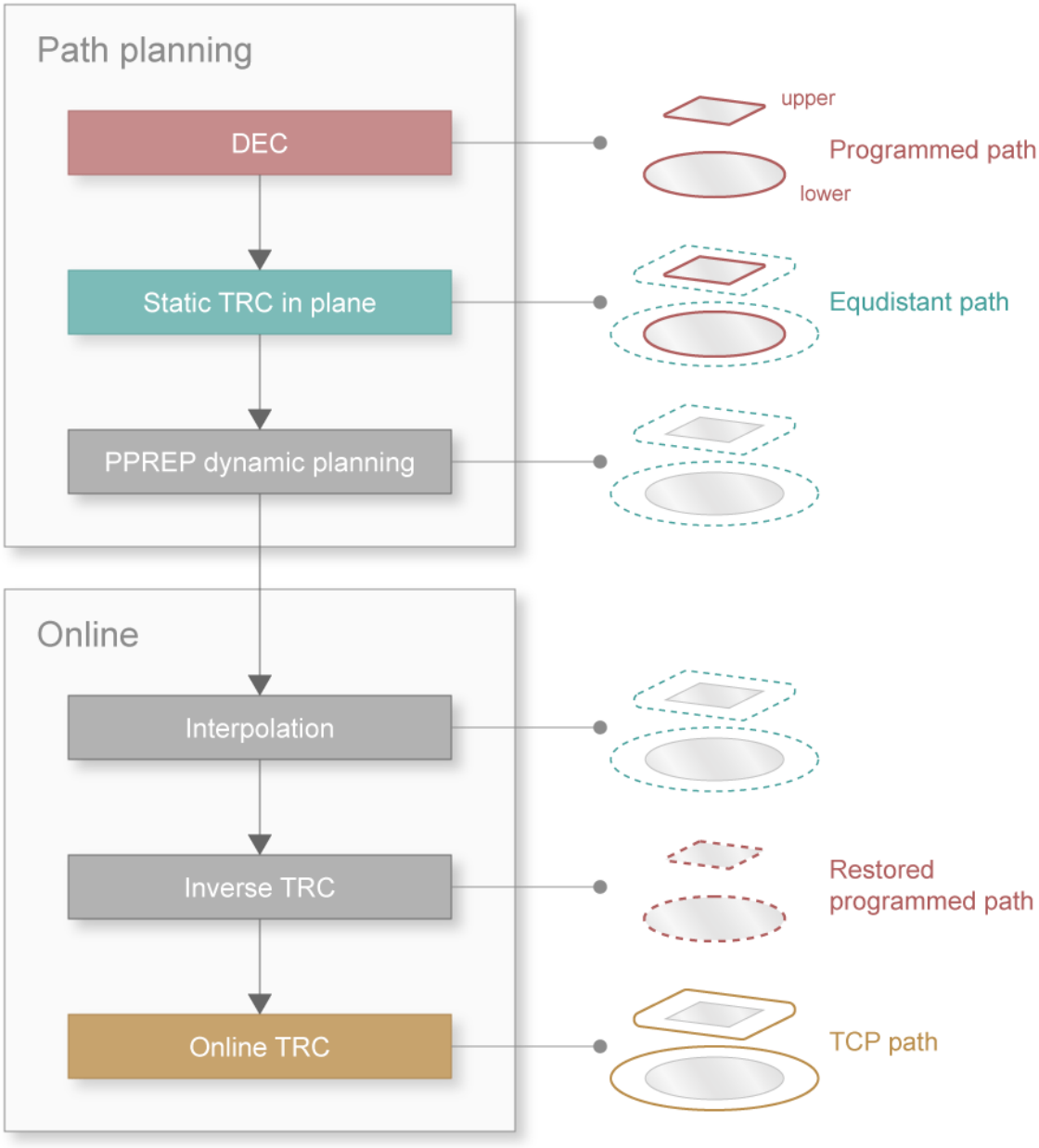


Fig. 3: 2-path programming flow with online TRC

You can integrate your own online tool radius compensation based on the “Extracted programmed path”.

Application example

A possible application is to use a TcCOM object in conjunction with a 2--path application. When the tool (wire) is inclined, an ellipse is then produced from the tool radius in the plane.

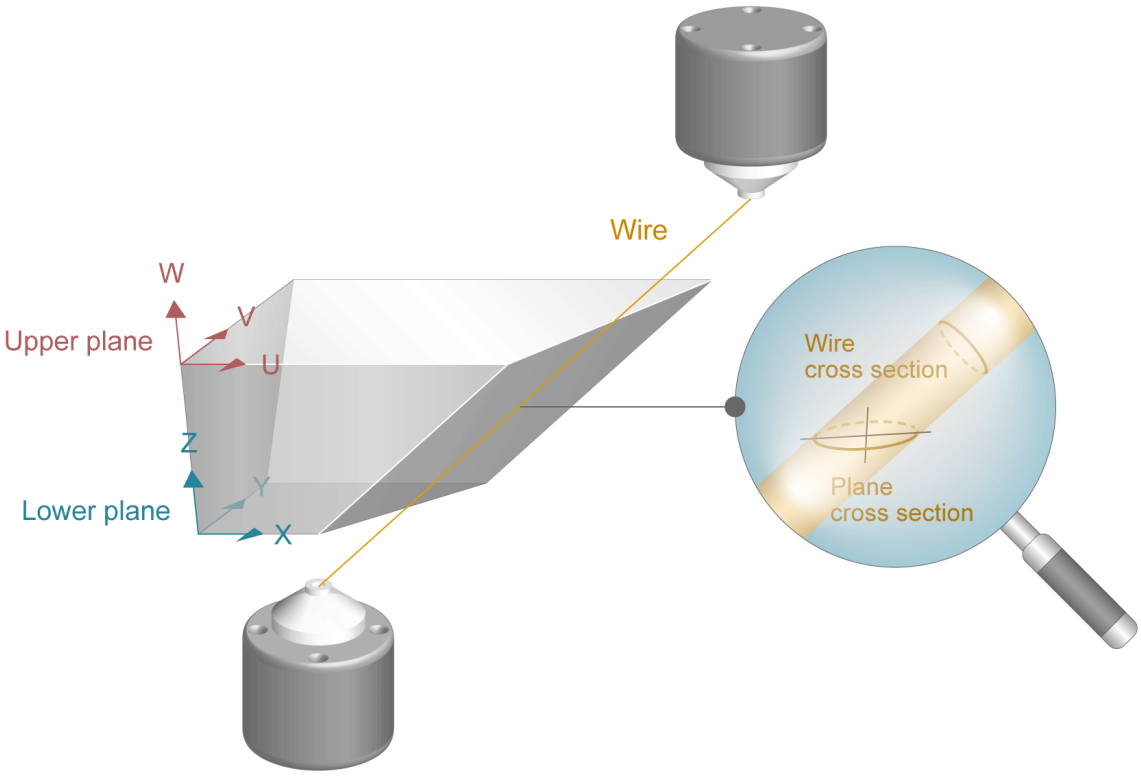


Fig. 4: 2-path application with inclined wire

2.2.1 Overview of commands

The list below contains the G commands that affect tool radius compensation in the NC program divided into permitted and impermissible commands:

Permitted commands

NC command	Meaning
G26	Circular transitions with TRC
G40	Deselect tool radius compensation
G41	Activate TRC/CRC on left of contour
G42	Activate TRC/CRC on right of contour
G138	Mode: Direct TRC selection
G139	Mode: Indirect TRC selection
G140	Deselecting contour masking
G141	Selecting contour masking
G236	Mode: Direct TRC selection/deselection on the path
G237	Mode: Direct TRC selection/deselection on the path
G239	Mode: Selecting/deselecting TRC directly without block

Impermissible commands

NC command	Meaning
G05	Tangential TRC selection/deselection
G25	Linear transitions with TRC
G238	Mode: Selecting inside corner of TRC
#TRC [REMOVE_MASKED_BLOCKS]	Remove contour loops

If an impermissible command is used, error ID 90166 is output.

2.2.2 Properties of tool radius compensation

The following information is always transferred to the McCOM object of the online tool radius compensation functionality:

- the current tool radius,
- the position,
- the path tangent vector and the
- tool direction vector

The response to block transitions and to selection and deselection of the static tool radius compensation functionality is only intended as an aid to understanding.

The initial situation for online tool radius compensation is to calculate static tool radius compensation.

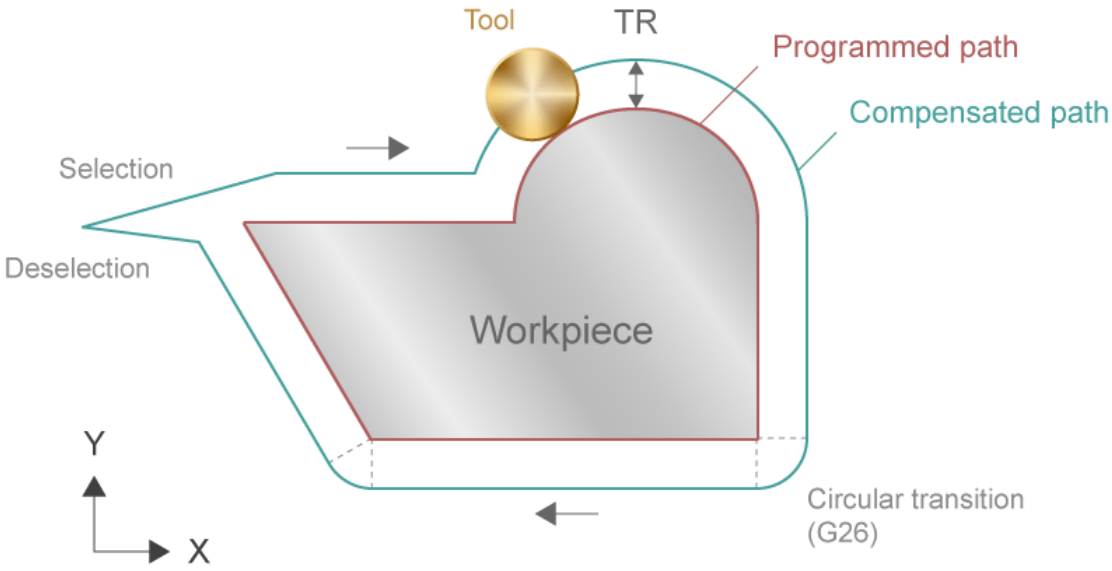


Fig. 5: Workpiece with static tool radius compensation

2.2.2.1 Reaction with block transition at angles greater than 180°

Static tool radius compensation adds circular elements to close contours when G26 and block transitions with transition angles greater than 180° are used. An inserted circular element makes a parallel path tangentially continuous.

A contour tangent continuously changes in a circular element.

The time for changing a contour tangent is approximately the traversed circular path divided by the velocity attained.

The inverse TRC is reduced to a single point when an added circular element is interpolated.

Based on this inverse TRC path, online tool radius compensation calculates the required TRC compensation in every tracing cycle.

Perpendicular tool

With a perpendicular tool, the resulting online TRC path is identical to the planned 2D TRC path.

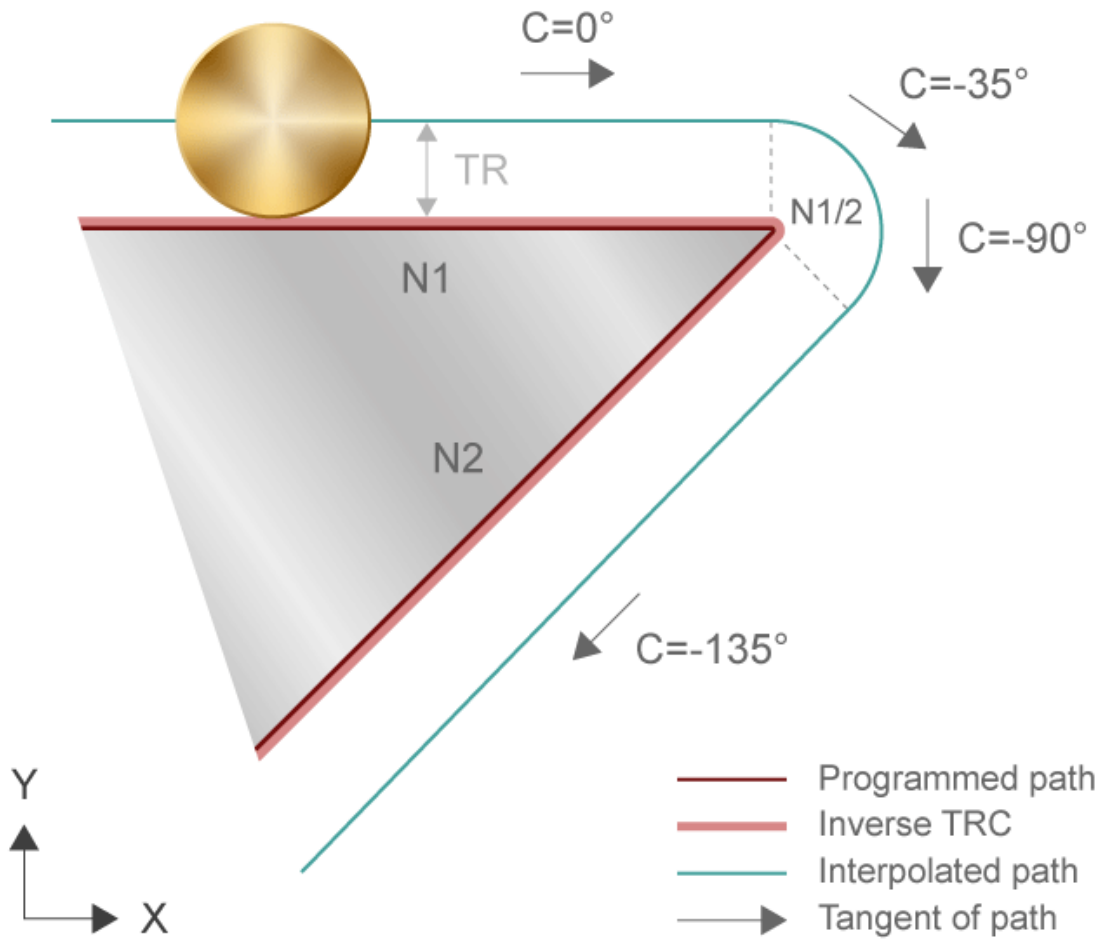


Fig. 6: Tangent rotation at the outside corners of a geometry

Response with a perpendicular tool

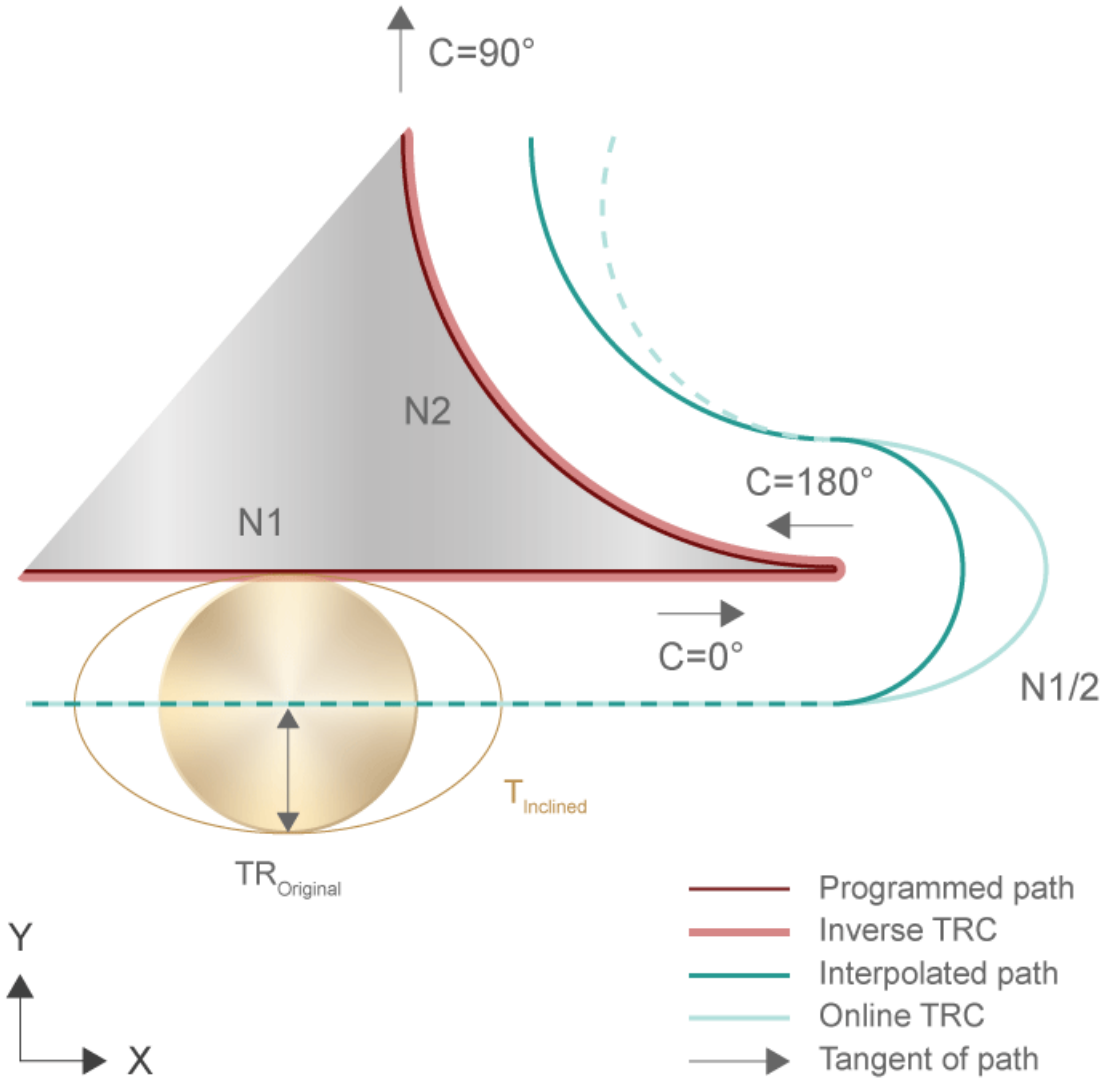


Fig. 7: Online TRC path with inclined tool at an outside corner

Other block transitions are contained in [PROG, section: Generating compensation blocks].

Change in tool radius:

If the tool radius is changed when tool radius compensation is active, the parameter can be influenced by the TRC option “PERPENDICULAR_RADIUS_CHANGE” [▶ 28].

By default, if an option is inactive, the changed tool radius is extended in the following motion block (see Reaction to change in tool radius).

The tool radius is then changed to the set option at the outside corner.

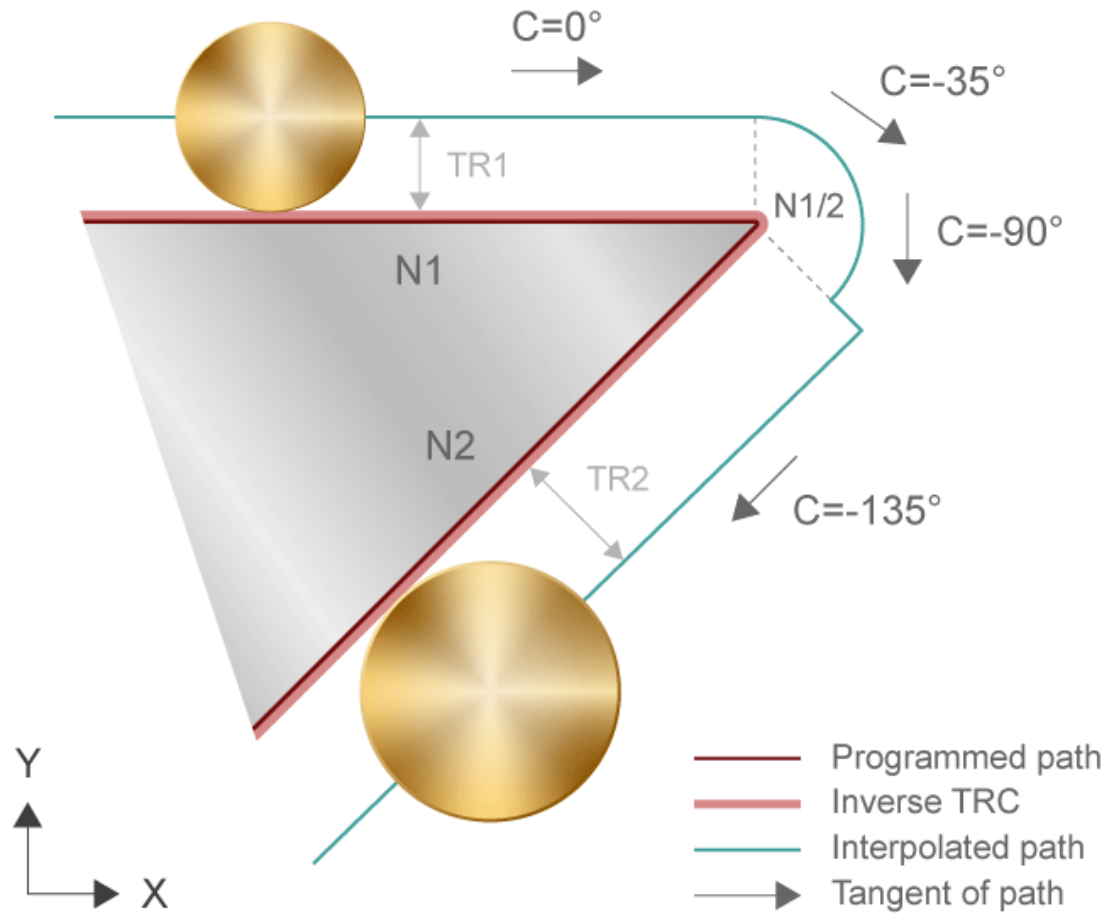


Fig. 8: Changing the tool radius at an outside corner

2.2.2.2 Reaction with block transitions at angles less than 180°

For block transitions with transition angles less than 180°, the two block elements are shortened.

An inverse TRC shifts the two involved blocks back to the programmed contour. The profile of an inverse TRC at this point is neither C0 continuous nor C1 continuous. During the interpolation, the profile of an inverse path jumps from the end of the first block to the start of the second block.

Based on this inverse TRC path, online tool radius compensation calculates the real TRC path.

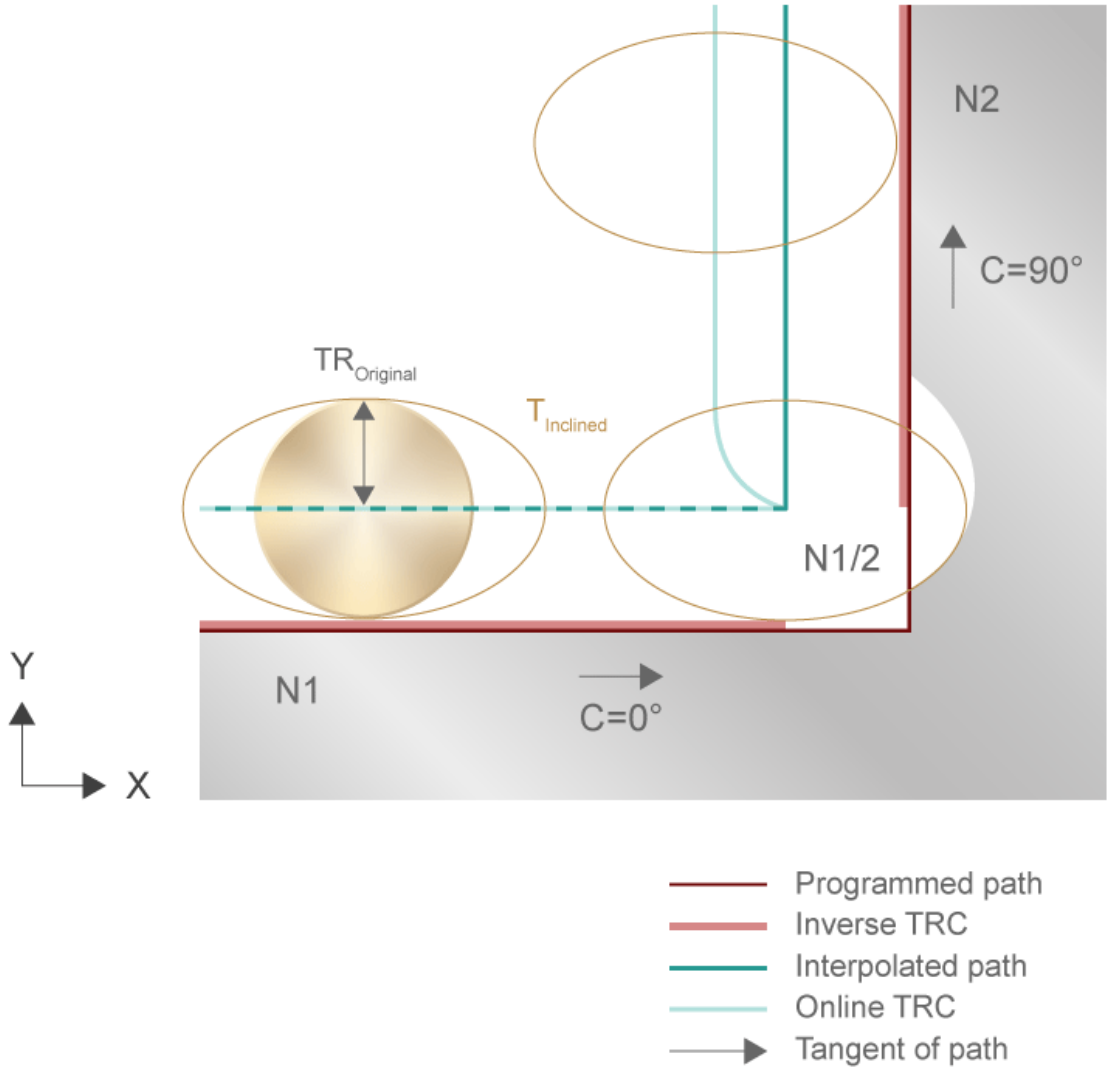


Fig. 9: Response at the inside corner of a geometry at block transitions with a angle of 90°

Other block transitions are contained in [PROG, section: Generating compensation blocks].

Change in tool radius:

If the tool radius is changed when tool radius compensation is active, the parameter can be influenced by the TRC option “PERPENDICULAR_RADIUS_CHANGE” [▶ 28].

By default, if an option is inactive, the changed tool radius is extended in the following motion block (see Reaction to change in tool radius).

If this option is set, the following profile results at the inside corner.

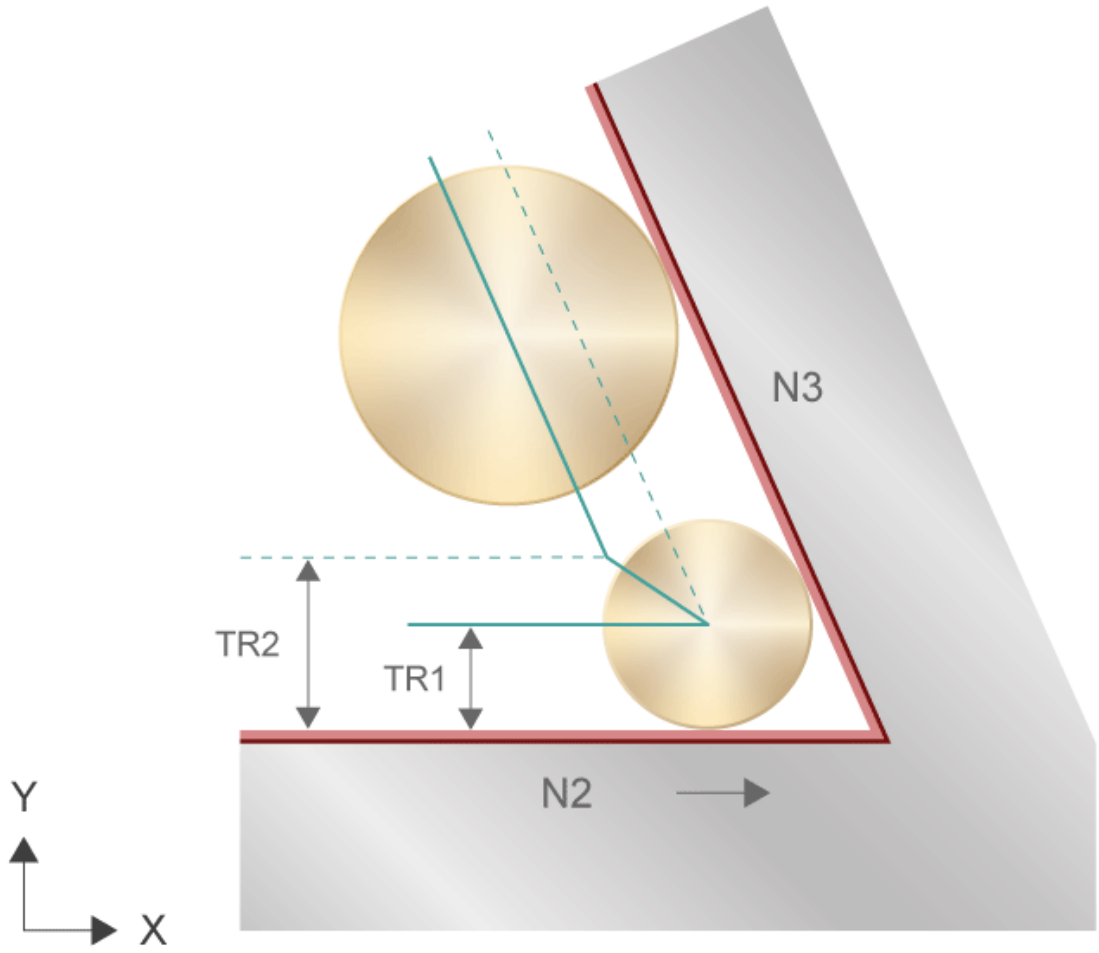


Fig. 10: Change in tool radius at the inside corner of a geometry at block transitions with an angle of 180°

2.3 Adding Online Tool Radius Compensation via TcCOM

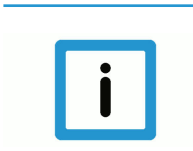
2.3.1 Interface methods

Methods to be implemented

The following method must be implemented for online tool radius compensation (TcCncToolRadiusCompInterfaces.h):

- virtual HRESULT TCOMAPI CalculateTrcPath (PTcToolRadiusCompParam trc)=0;

CalculateTrcPath	Calculating the compensated path using the specified tool radius in the programming coordinate system. Input: Current parameters of tool radius compensation.



Notice

Member variables must be initialised in the class constructor to dimension the input and output coordinates.

2.3.2 Instance data

Working data – TcToolRadiusCompParam

Parameters of the methods

The parameters for each of the methods are transferred in encapsulated form by the structure TcCncDynContourCtrlParam (TcCncDynContourCtrlInterfaces.h):

```
struct TcToolRadiusCompParam
{
  EcToolRadiusComp   type;

  int dimension; // dimension of positions, 3 or 6 for 2 paths
  int dim_radius; // dimension of radius, 1 or 2 for 2 paths

  const double * i_path; // input: position of path, see dimension
  const double * i_tangent; // input: tangent of path, see dimension
  const double * i_tool_direction; // input: direction of tool, see di-
mension
  const double * i_radius; // input: actual tool radius, see dim_radius

  double * o_path; // output: position of compensated path
}
```

TcCncParam structure

The following parameters are generally used in every TcCOM interface:

```
struct TcCncParam
{
  // Input data
  EcCncCallerID  caller_id;     // identification of caller
  unsigned long  block_number;  // block number of actual executed
                                // CNC program, MDI(not yet supported)
  unsigned short  cnc_channel;  // calling CNC channel
  // Output data
  double          ret_value1;   // additional error value
  double          ret_value2;   // additional error value
  char            ret_text[24]; // out: additional error text, max. 24
byte
}
```

Caller ID

Online tool radius compensation is called at various points in the interpolation module. The following identifiers are available for identification:

Identification number	Caller ID
1000	EcCncTrcCallerID_Display
1001	EcCncTrcCallerID_Interpolation
1002	EcCncTrcCallerID_GeometricFeedAdaption
1003	EcCncTrcCallerID_TargetPos

A TcCOM object can use caller IDs to implement various algorithms for the interpolation and display functionality, e.g.:

- With an interpolation, the tool radius specified for this path is used in both paths.
- The same (e.g. secondary) radius is used for display in both paths.

2.3.3 Configuring and registering a TcCOM object

Registering in TwinCAT

The following data is used to register a TcCOM object belonging to online tool radius compensation (TcCncServices.h):

- Type 3 (CCNC_REGISTEROBJECT_TYPE_TOOL_RADIUS_COMP) is default.
- If the TcCOM object does not use instance-specific objects, set the Group entry to 0.
- If the TcCOM object uses instance-specific variables, set the Group entry to the respective channel number [1;12] assigned to the object.
  Maximum one object per channel.
- Index is not used.

An online tool radius compensation object is registered via the following TcCOM interface which is defined in the file TcCncInterfaces.h

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

Supplying the TcCOM object

After online tool radius compensation is generated, there should be 2 files available:

1. TMC file
2. Driver file

The description of online tool radius compensation is contained in the TMC file, e.g. TcCncMyOnlineToolRadiusComp.tmc.

The file is located in the work directory of the solution.

The drive file directory is dependent on Release or Debug;

- <TwinCAT> \3.1\sdk_products\TwinCAT RT (x64)\Release or
- <TwinCAT> \3.1\sdk_products\TwinCAT RT (x64)\Debug

When the configuration is activated, the respective driver file is automatically copied to the directory <TwinCAT>\3.1\Driver\AutoInstall.

Based on the above example name: TcCncMyOnlineToolRadiusComp.sys

Notice

You only need to trigger the generation (Debug/Release) and activate the associated configuration.

The procedure for debugging the created online tool radius compensation is analogous to debugging an McCOM transformation. This procedure is described in [McCOM transformation, section: Debugging the transformation].

Loading the object

Loading the object for online tool radius compensation is described in “Integrating online tool radius compensation object [▶ 71]”.

2.4 Programming

The use of online tool radius compensation (online TRC) is defined in the programming of the INVERSE and ONLINE options [▶ 24]. The selected setting becomes effective when tool radius compensation (TRC) is activated by G41 or G42. Online TRC is deselected when TRC is deactivated with G40.

Attention

In order to use online TRC, both options
#TRC [INVERSE=1 ONLINE=1] must be set.

Otherwise an error message with ID 22125 is output.

Programing Example

Selecting TRC with options set

```
N10 V.G.WZ_AKT.R = 0.3
N20 G26
N30 G138
; …
N70 #TRC [INVERSE=1 ONLINE=1] ( Set options)

N100 G0 X0 Y0 U0 V0

N280 G41      ( Select TRC)
( of the contour)
; …
N500 G40      ( Deselect TRC)
; ...
N550 M30
```

Programming the tool radius

Identical tool radii in both paths

```
N.. V.G.WZR=0.15
```

Alternatively:

```
N.. V.G.WZR=0.15 : X100 Y20: U100 V20
```

Different tool radii in both paths

```
N.. : V.G.WZR=0,139 : V.G.WZR=0.15
Alternatively:
N.. : V.G.WZR=0.139 X100 Y20 : V.G.WZR=0.15 U100 V20
```

Define the tool radius in the reference path

```
N.. : V.G.WZR=0,134 :
```

Define the tool radius in the second path

```
N.. : : V.G.WZR=0,151
```

2.4.1 TRC options for online TRC and 2-path

Online TRC

The following parameters can influence online tool radius compensation. They can only be set when tool radius compensation is active (G41/G42).

The INVERSE parameter is used to extract the originally programmed path using the offset parallel path generated by tool radius compensation.

When the INVERSE parameter is set, the ONLINE parameter permits settings for online tool radius compensation. Online tool radius compensation can be implemented by a TcCOM object.

Attention

The ONLINE parameter may only be set to a value unequal to 0 if the INVERSE parameter is active. If INVERSE is deactivated, error ID 22125 is output.

Syntax:

#TRC [[INVERSE=..] [**ONLINE=**..] [**ONLINE_BY_VECTOR=**..]]

INVERSE=.. Extract parallel path created by tool radius compensation.

0: no extraction (default).

1: Extract the parallel path.

ONLINE=.. Set online tool radius compensation:

0: No calculation of online tool radius compensation.

1: Calculate the compensated path via TcCOM interface.

2: Simple calculation of the parallel path for each plane in the CNC (corresponds to the path taken from tool radius compensation, only for testing)

3: Calculate the parallel path by considering tool geometry in the CNC.

ONLINE_BY_VECTOR=.. This parameter determines the method used to backward transform a 3D TRC online compensated position on the coordinate system plane.

0: The shift along the wire direction must be implemented in a TcCOM object of the kinematic transformation.

1: The shift along the wire direction is executed directly after 3D TRC compensation by the CNC.

2: Combined solution: the generation of cyclic command values uses the implementation in the TcCOM object of the kinematic transformation; the display uses the calculation executed by the CNC directly after 3D TRC compensation.

Basic radius for tool radius compensation

This option can be used to specify whether the tool radius compensation for individual paths is used to calculate the parallel path using the tool radius specifically defined for this path. Alternatively, the tool radius of the path that can be defined using this option.

This option can be used with EDM wire erosion to display the ideal path on an HMI, whereas taking wire wear is considered in the calculation of the nominal geometry.

Syntax:

#TRC [[**MULTI_PATH_RADIUS=**..] **]**

MULTI_PATH_RADIUS=..	This parameter specifies the tool radius that is to be used by the static tool radius compensation to calculate the parallel path.	
	INDIVIDUAL/ DEFAULT	The tool radii programmed in each path are used (default).
	REFERENCE	The tool radius of the reference path is used.
	SECONDARY	The tool radius of the secondary path is used.

Dynamic offset overlapping

Dynamic offset overlapping is a user-specific overlapping of the TCP path acting in the online TRC.

Therefore, overlapping axis motions can only be considered to a limited extent when maximum velocity and acceleration are planned in the CNC.

Overlaps can lead to acceleration overshoots, especially at inside corners. To prevent this, the CNC analyses the tangent rotation at inside corners and uses this to determine the maximum permissible dynamics.

The function in the online TRC is enabled/disabled by the PLC. The calculation of the permissible dynamics for offset overlapping at inside corners is enabled by the following programming command:

Syntax:

#TRC [[**DYNAMIC_VARIATION_MAX_OFFSET=**..] **]**

DYNAMIC_VARIATION_MAX_OFFSET =..	Define the maximum dynamic offset in [0.1µm]
	0: Disable Dynamic Offset Overlapping (default).
	> 0: Enable; maximum offset is used to calculate the dynamics.

TAPERLINK

The TAPERLINK option permits synchronisation between the reference path and the s3econdary path in a 2-path configuration in order to obtain the optimum wire angle. See [FCT-C49, section: Description].

The condition for using this function is a 2-path configuration and selection of tool radius compensation using G41 or G42.

Syntax:

#TRC [[**TAPERLINK=**..] **]**

TAPERLINK =.. Define mode for the taper link function.

0: Taper link function inactive (default).

1: Taper link active: Compensation is active on both paths; automatic detection.

2: Taper link active: Reference path compensates the secondary path.

3: Taper link active: Secondary path compensates the reference path.

2.4.2 TRC option GEN_CIR_BLOCK_IN_CORNER

This option integrates a virtual circle (radius 0) in inside corners when tool radius compensation is active. Inside corners are transitions between motion blocks whose transition angle is less than 180°.

Only possible with a tool radius unequal to 0.

This option must be used to avoid acceleration overshoots.

Syntax:

#TRC [[**GEN_CIR_BLOCK_IN_CORNER=**..] **]**

GEN_CIR_BLOCK_IN_CORNER=.. Insert virtual circular blocks of radius 0 to inside corners of the tool radius compensation.

0: Do not insert virtual circular blocks (default)

1: Insert virtual circular blocks

2.4.3 TRC option G236_LIN

This option only takes effect when G236 selection mode is used and when the transition angle is within the selection and deselection range of the tool radius compensation between 90° and 180°

Syntax:

#TRC [[**G236_LIN =**..] **]**

G236_LIN=.. Define whether a circular or linear block is inserted in the described angle range of 90° to 180°.

0: Insert a circular block (default)

1: Insert a linear block

2.4.4 TRC option PERPENDICULAR_RADIUS_CHANGE

This option directly extends a programmed change in tool radius by inserting a motion orthogonal to the programmed contour.

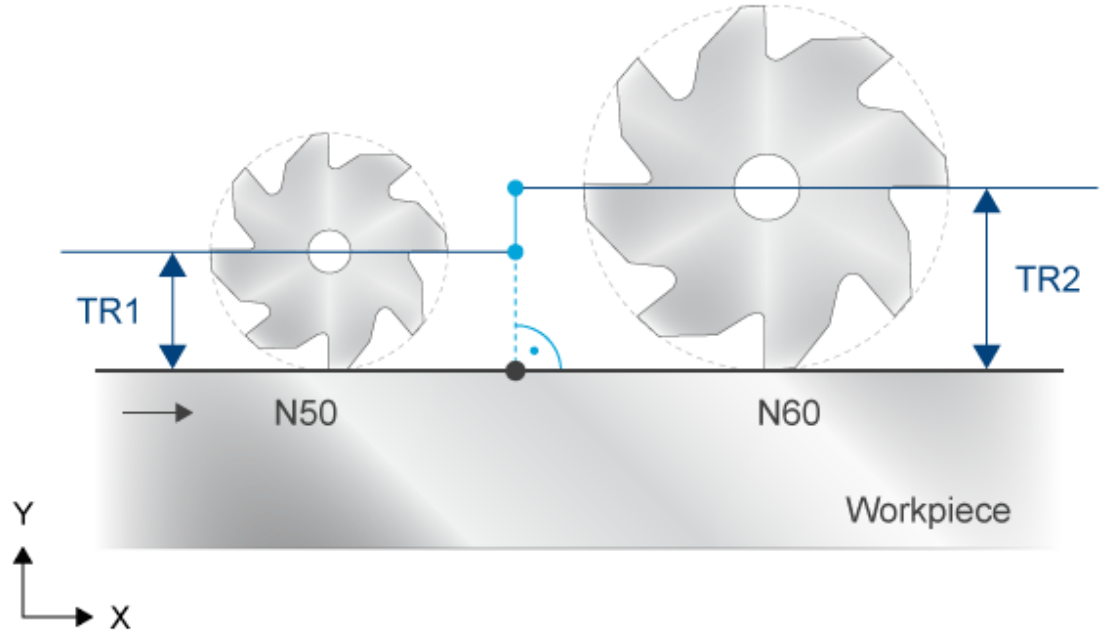


Fig. 11: Orthogonal extension of a tool radius change

Syntax:

#TRC [[PERPENDICULAR_RADIUS_CHANGE=..] **]**

PERPENDICULAR_ RADIUS_CHANGE=..

This parameter can extend the change in tool radius perpendicular to the contour.

0: No perpendicular extension for the change in the new tool radius (default)

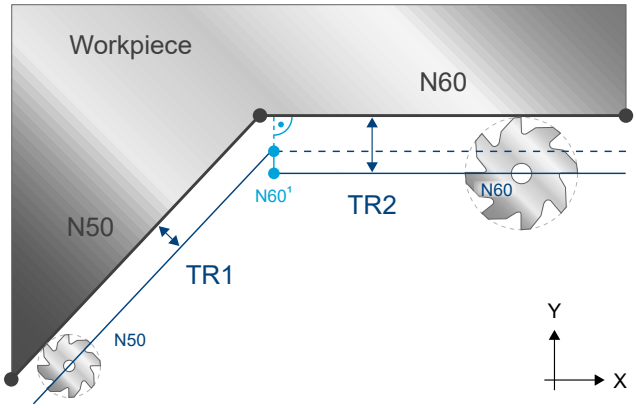
1: Perpendicular extension for the change in the new tool radius

In the program examples, a change is made to the tool radius between the programmed motion blocks N50 and N60.

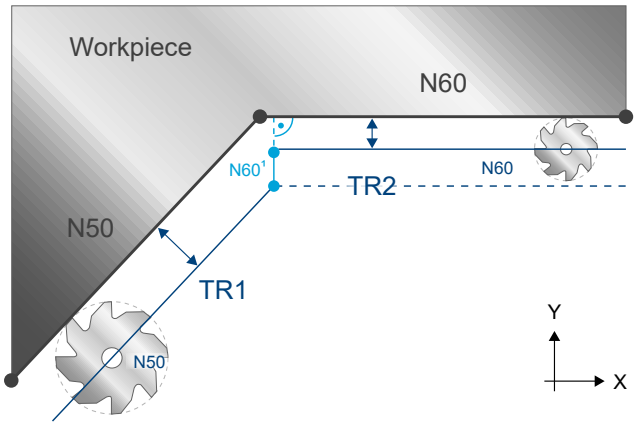
To illustrate the response, a very large change in tool radius is chosen. Normally, a change in tool radius involves very minor compensation, e.g. caused by wear.

Several block transitions are shown in the examples below. All combinations of linear and circular blocks are permitted.

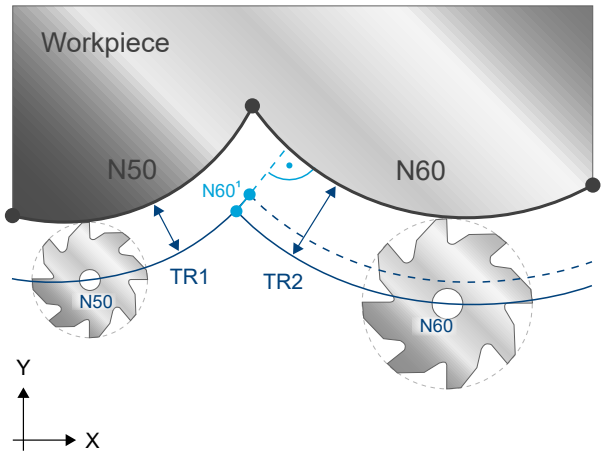
Change of tool radius at inside corner



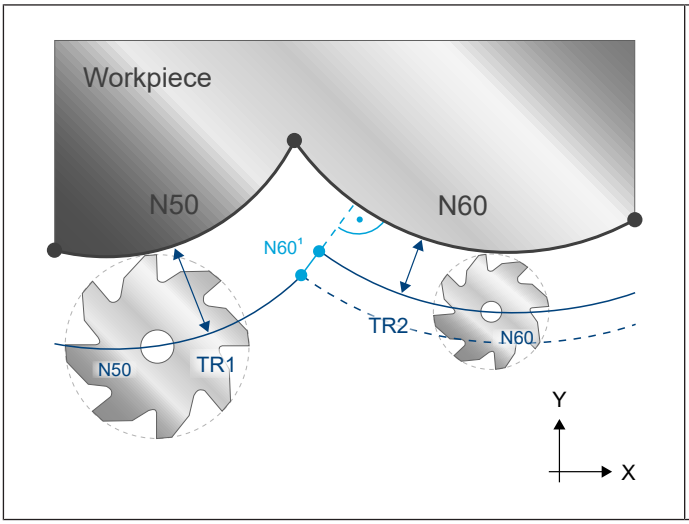

```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 5
N20 G42
N40 G01 X20 Y0
N50 G01 X50 Y50
N55 V.G.WZ_AKT.R = 6
N60 G01 X100
N70 G40 X120 Y0
N99 M30
```




```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 6
N20 G42
N40 G01 X20 Y0
N50 G01 X50 Y50
N55 V.G.WZ_AKT.R = 5
N60 G01 X100
N70 G40 X120 Y0
N99 M30
```

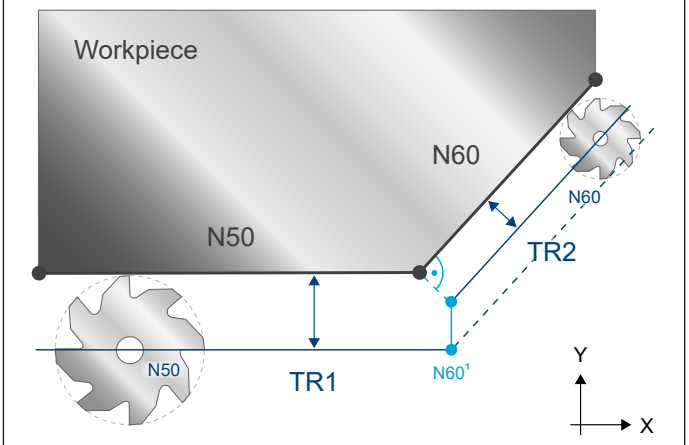



```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 5
N20 G42
N30 G01 X10 Y20
N40 G01 X20
N50 G03 X50 Y25 R30
N55 V.G.WZ_AKT.R = 6
N60 G03 X90 Y18 R40
N70 G01 X100
N80 G40 X120 Y0
N99 M30
```

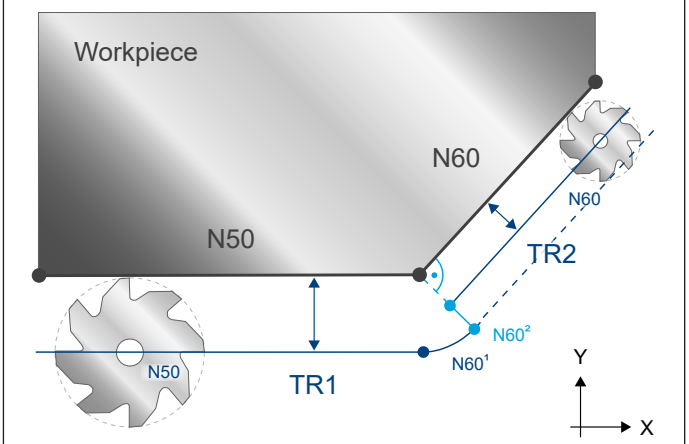



```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 6
N20 G42
N30 G01 X10 Y20
N40 G01 X20
N50 G03 X50 Y25 R30
N55 V.G.WZ_AKT.R = 5
N60 G03 X90 Y18 R40
N70 G01 X100
N80 G40 X120 Y0
N99 M30
```

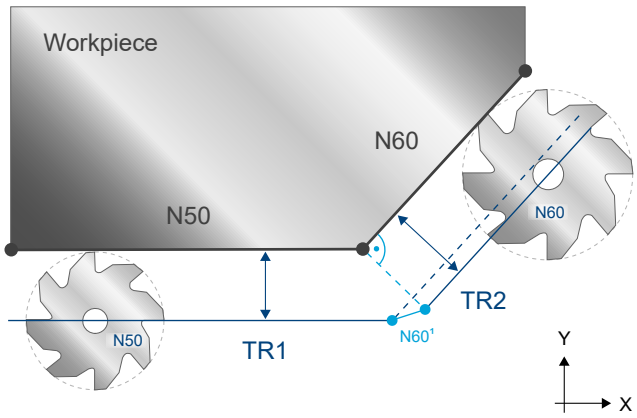
Change of tool radius at outside corner



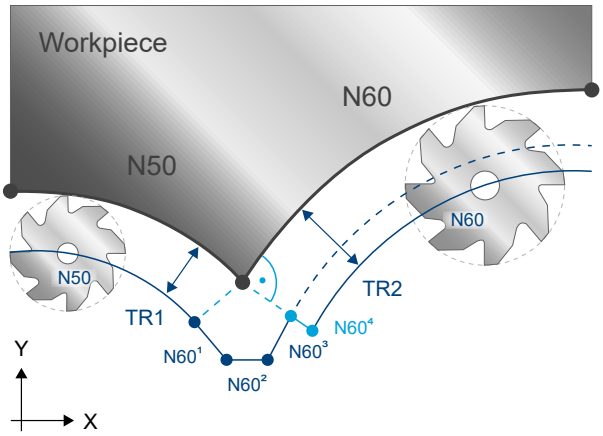

```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 6
N20 G42 G25
N40 G01 X20 Y20
N50 G01 X50
N55 V.G.WZ_AKT.R = 5
N60 G01 X80 Y50
N70 G01 X120
N80 G40 X140 Y0
N99 M30
```



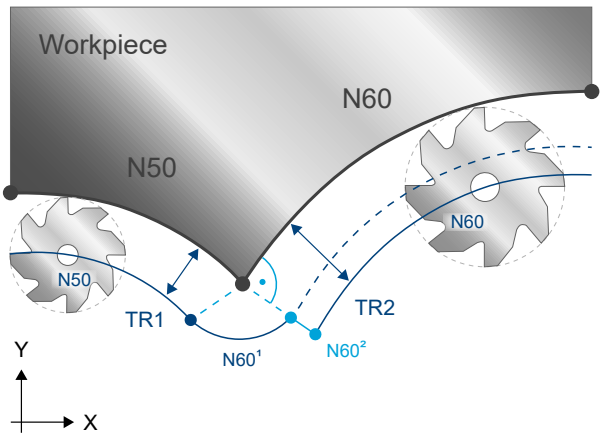

```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 6
N20 G42 G26
N40 G01 X20 Y20
N50 G01 X50
N55 V.G.WZ_AKT.R = 5
N60 G01 X80 Y50
N70 G01 X120
N80 G40 X140 Y0
N99 M30
```



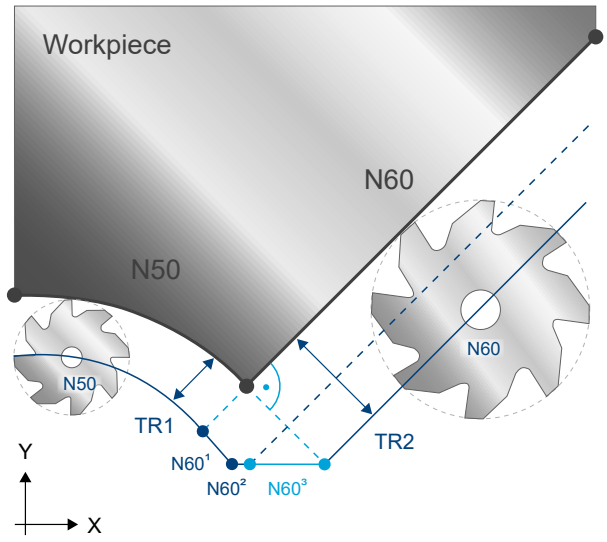
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [**PERPENDICULAR_RADIUS_CHANGE=1**]
N10 **V.G.WZ_AKT.R= 5**
N20 G42 **G25**
N40 G01 X20 Y20
N50 G01 X50
N55 **V.G.WZ_AKT.R = 6**
N60 G01 X80 Y50
N70 G01 X120
N80 G40 X140 Y0
N99 M30



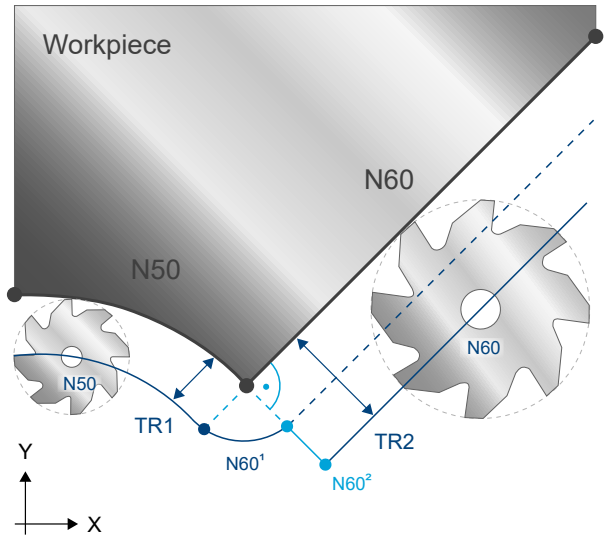
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [**PERPENDICULAR_RADIUS_CHANGE=1**]
N10 **V.G.WZ_AKT.R= 5**
N20 **G25**
N30 G42 X10 Y10
N40 G01 X20
N50 G02 X60 Y0 R30
N55 **V.G.WZ_AKT.R = 6**
N60 G02 X90 Y12 R50
N70 G01 X100
N80 G40 X120 Y0
N99 M30



%Test.nc
N05 G0 X0 Y0 F1000
#TRC [**PERPENDICULAR_RADIUS_CHANGE=1**]
N10 **V.G.WZ_AKT.R= 5**
N20 **G26**
N30 G42 X10 Y10
N40 G01 X20
N50 G02 X60 Y0 R30
N55 **V.G.WZ_AKT.R = 6**
N60 G02 X90 Y12 R50
N70 G01 X100
N80 G40 X120 Y0
N99 M30




```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 5
N20 G25
N30 G42 X10 Y10
N40 G01 X20
N50 G02 X60 Y0 R30
N55 V.G.WZ_AKT.R = 6
N60 G01X80 Y40
N70 G01 X100
N80 G40 X120 Y0
N99 M30
```




```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 5
N20 G26
N30 G42 X10 Y10
N40 G01 X20
N50 G02 X60 Y0 R30
N55 V.G.WZ_AKT.R = 6
N60 G01X80 Y40
N70 G01 X100
N80 G40 X120 Y0
N99 M30
```

2.5 Parameter

P-CHAN-00550	Definition of functionalities for tool radius compensation
Description	This parameter defines individual functionalities for tool radius compensation.
Parameter	configuration.tool_radius_comp.function
Data type	STRING
Data range	MULTI_PATH: 2-path configuration and 2-path programming active -: No functionalities defined.
Dimension	----
Default value	*
Remarks	Parameter is available as of the following Builds: V2.11.2040.04 ; V2.11.2810.02 ; V3.1.3079.17 ; V3.1.3107.10 * Note: The default value of variables is a blank string. The parameters P-CHAN-00555 and P-CHAN-00556 can be used to define functions depending on the machining mode.

2.6 Error messages

User-specific compensation errors

Besides the standard transformation errors, the user can issue user-defined errors for some methods (e.g. error ID 123) by using the function return value (0 = OK).

```
HRESULT CToolRadiusComp::CalculationTrcPath(PTcCncToolRadiusCompParam trc){
  if (…)
    return 123; // raise error
…
  return S_OK;
}
```

Error messages in TcCncUsersEvents.xml

In the event of an error, the user-defined return value of the method can be transferred to the error message evaluation via the PLC or the TwinCAT Event Logger, see also FCT-M7: TwinCAT3 error output.
Error messages must be supplemented in the corresponding language in the XML error message file. (C:\TwinCAT\3.1\Target\Resource):

```
<Event>
  <Id>123</Id>
  <Message LcId="1033">Online tool radius compensation: error 123</Message>
  <Message LcId="1031">Online tool radius compensation: Error 123</Message>
</Event>
```

The error is output by the Event Logger.

3 Geometric feed adaptation

3.1 Overview

Task

The Geometric Feed Adaptation functionality offers the option of ensuring constant surface removal with EDM wire erosion by using a factor that acts cyclically on the command feed rate

Release Note

The functionality is available as of CNC Build V3.1.3108.

Notice

This function is an additional option requiring a license.

Properties

The functionality can be used via a user-specific TcCOM object.

Geometric feed adaptation is used to determine a factor that is multiplied by the current command velocity and the override factor.

Programming and parametrisation

The functionality can be configured, activated and deactivated using the NC command #GEO FEED ADAPT [▶ 40] and the associated channel parameters [▶ 35].

Links to other documents

For the sake of clarity, links to other documents and parameters are abbreviated, e.g. [PROG] for the Programming Manual or P-AXIS-00001 for an axis parameter.

For technical reasons, these links only function in the Online Help (HTML5, CHM) but not in pdf files since pdfs do not support cross-document linking.

3.2 Description

Geometric feed adaptation is used to calculate a factor for wire erosion to ensure constant surface removal.

The calculations are based on a cyclically corrected radius centre point path. The tool radius consists of the actual tool radius and an additional parameter, i.e. the gap value.
Corrected path positions are cyclically supplied to the TcCOM object.

The additional parameters required can be transferred by the PLC. The parameters can also be defined in the NC program.

The result of the calculations is a factor that is multiplied in the commanded feed rate.

The arrangement of contour control in the NC channel is shown below:

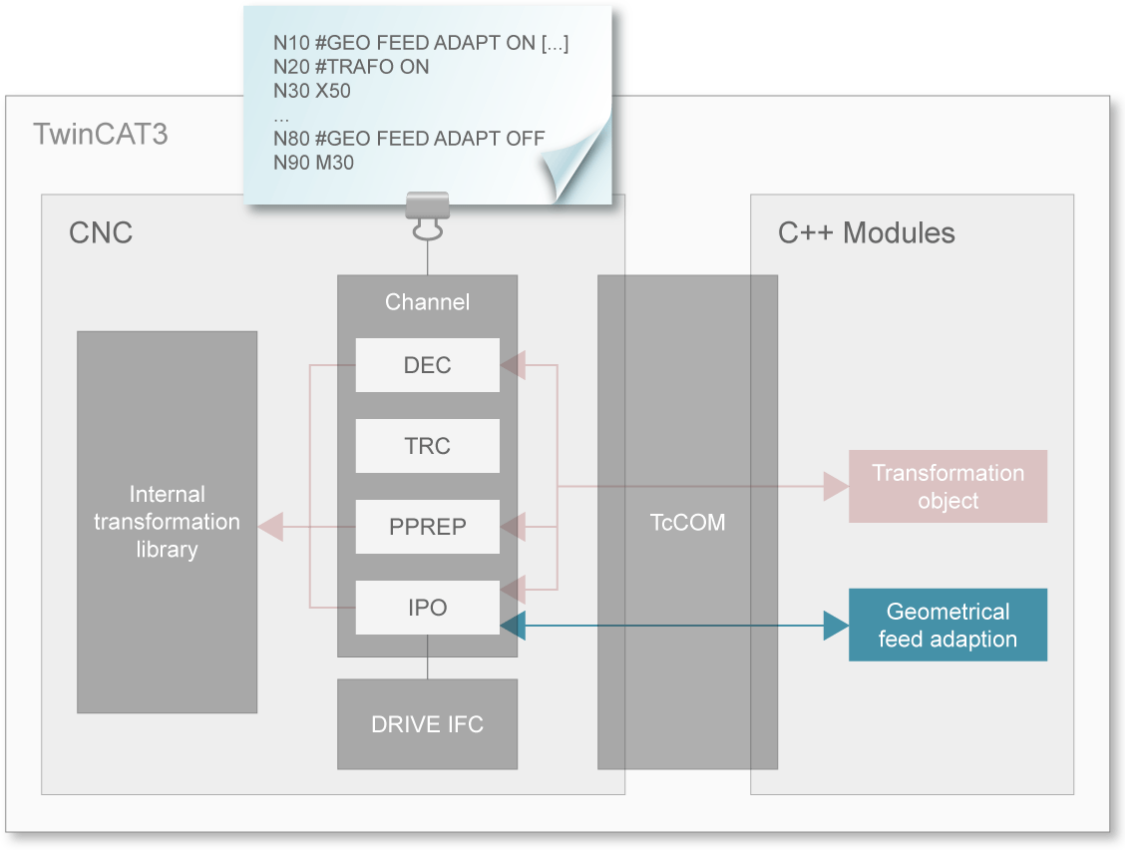


Fig. 12: Geometric feed adaptation in the NC channel

Notice

With a TcCOM object with instance-specific variables, the object must be configured in the respective CNC channel.

3.2.1 Integration in the NC channel

Geometric feed adaptation consists of multiple elements:

1. Call the online TRC with caller ID "Geometric feed adaptation". The gap value taken from the NC program (pre-assignment in the channel list) is then added to the wire radius.
2. Transform the positions from PCS to MCS coordinates. The positions of the second plane are calculated relative to the reference coordinate system.
3. Call the TcCOM object "Geometric feed adaptation". Calculate the additional override factor which is then multiplied by the override value from the PLC.

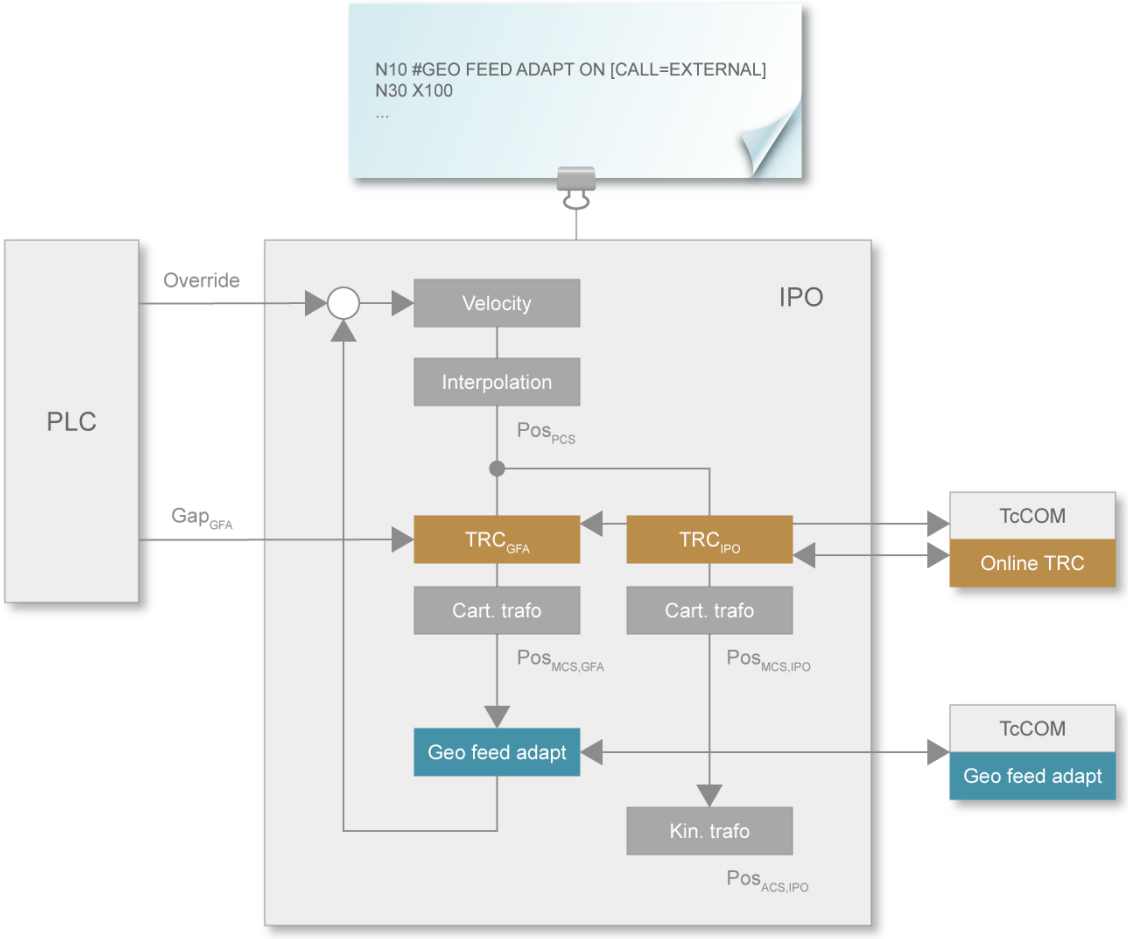


Fig. 13: Integrating the object in the NC channel

3.3 Adding a geometric feed adaptation via TcCOM

3.3.1 Interface methods

Methods to be implemented

The following methods must be implemented for a geometric feed adaptation (TcCncGeoFeedAdaptInterfaces.h):

- virtual HRESULT TCOMAPI CalculateFeedAdaption(PTcGeoFeedAdaptParam gfa)=0;

CalculateFeedAdaption	When the functionality is in its active state, the calculation takes place in each interpolation cycle. The method calculates an override factor dependent on the surface ratio. Input: Current parameters for geometric feed adaptation

3.3.2 Instance data

Working data - PTcGeoFeedAdaptParam

Parameters of the methods:

The parameters for each of the methods are transferred in encapsulated form by the structure PTcGeoFeedAdaptParam (TcCncGeoFeedAdaptInterfaces.h).

```
struct PTcGeoFeedAdaptParam
{
  // config: EcGeoFeedAdapt_ParamStandard
  EcGeoFeedAdapt     type;
  // config: dimension of path : 1 or 2
  int                dim_path;
  // config: dimension of tool parameters
  int                dim_tool_param;
  // tool parameters
  const double       * i_tool_param;
  // feed adaption factor
  double             * o_feed_adaption;

  struct TcGeoFeedAdaptPath path[2]; //
}
```

Path-specific parameters:

```
struct TcGeoFeedAdaptPath
{
  // input: 3-dim GFA position X,Y,Z
  const double       * i_position_gfa;
  // Input: 3-dim display position X,Y,Z
  const double       * i_position_display
}
```

3.3.3 Configuring and registering a TcCOM object

Registering in TwinCAT

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

- Type 5 (TCCNC_REGISTEROBJECT_TYPE_GEO_FEED_ADAPT) is default.
- If the TcCOM object does not use instance-specific objects, set the Group entry to 0.
- If the TcCOM object uses instance-specific variables, set the Group entry to the respective channel number [1;12] assigned to the object. Maximum one object per channel.
- Index is not used.

A geometric feedforward control object is registered via the following TcCOM interface which is defined in the file TcCncInterfaces.h:

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

Supplying the TcCOM object

After geometric feed adaptation is generated, there should be 2 files available:

1. TMC file
2. Driver file

Geometric feed adaptation is described in the TMC file, e.g. TcCncMyGeoFeedAdapt.tmc.

The file is located in the work directory of the solution.

The drive file directory is dependent on Release or Debug;

- <TwinCAT> \3.1\sdk_products\TwinCAT RT (x64)\Release or
- <TwinCAT> \3.1\sdk_products\TwinCAT RT (x64)\Debug

When the configuration is activated, the respective driver file is automatically copied to the directory <TwinCAT>\3.1\Driver\AutoInstall.

Based on the above example name: TcCncMyGeoFeedAdapt.sys

Notice

You only need to trigger the generate function (Debug/Release) and activate the associated configuration.

The procedure for debugging the created geometric feed adaptation is analogous to debugging an McCOM transformation. This procedure is described in [McCOM transformation, section: Debugging the transformation].

Loading the object

Loading the object for geometric feed adaptation is described in “Integrate geometric feed adaptation object [▶ 61]”.

3.4 Programming

Syntax:

#GEO FEED ADAPT [ON/OFF] [CALL=.. [I<i>=..] [F<j>=..] { I<i>=..}{ F<j>=..}]

ON Activate geometric feed adaptation, analogous to P-CHAN-00386 [▶ 41].

OFF Deactivate geometric feed adaptation, analogous to P-CHAN-00386 [▶ 41].

CALL=.. Type of method used for geometric feed adaptation.

Permitted identifiers:

- DEFAULT - The method in P-CHAN-00387 [▶ 41] is used (default)
- EXTERNAL - A customer-specific method is used

I<i>=.. SGN32 value, analogous to the P-CHAN-00391 [▶ 42] parameter.

where i=0 - 3.

The equals sign is mandatory,
e.g. I2=17

F<j>=.. REAL64 value, analogous to the P-CHAN-00390 [▶ 41] parameter.

where j=0 - 3.

Cation:
F0=<gap_value> [0.1 µm] is default.

The equals sign is mandatory,
e.g. F2=3.45

The following V.G. variables can be accessed in the NC program:

Variable	Meaning	Data type	Input/output unit	Permitted access: Read/ Write
V.G.GEO_FEED_ADAPT.ACTIVE	Indicates whether geometric feed adaptation is active.	Boolean	0 / 1	L
V.G.GEO_FEED_ADAPT.CALL	Indicates the method used for geometric feed adaptation. 0: EXTERNAL	SGN32	0 / 1	L

Programing Example

External geometric feed adaptation in 2-path application

```
N180 #GEO FEED ADAPT ON [CALL=EXTERNAL F0=12.3]
N260 G141
N270 G139 G26
N270 G01 G42 : X10 Y0 : U12 V2
; …
N400 G40 : X0 Y0 : U0 V0
N410 #GEO FEED ADAPT OFF
N420 M30
```

3.5 Parameter

P-CHAN-00386	Select/deselect the constant surface feed rate function
Description	This parameter selects/deselects the constant surface feed rate.
Parameter	geo_feed_adapt.active
Data type	BOOLEAN
Data range	0/1
Dimension	----
Default value	0
Remarks	Available as of V3.1.3108

P-CHAN-00387	Selection of calculation method (surface feed rate)
Description	This parameter selects whether the constant surface feed rate in a COM object is calculated or whether the algorithm integrated in the CNC is used.
Parameter	geo_feed_adapt.call
Data type	STRING
Data range	EXTERNAL, BUILTIN
Dimension	----
Default value	EXTERNAL
Remarks	Available as of V3.1.3108 BUILTIN is only for internal tests

P-CHAN-00390	REAL64 input parameters for surface feed rate
Description	A total of four REAL64 input parameters can be defined for surface feed rate. **Caution**: f[0]=<gap_value> in [0.1 µm] This entry is default.
Parameter	geo_feed_adapt.param.f[i] where i = 0…3
Data type	REAL64
Data range	MIN(REAL64) ≤ f[i] ≤ MAX(REAL64)
Dimension	----
Default value	0.0
Remarks	Available as of V3.1.3108

P-CHAN-00391	**SGN32 input parameters for surface feed rate**
Description	A total of four SGN32 input parameters can be defined for surface feed rate.
Parameter	geo_feed_adapt.param.i[i] where i = 0…3
Data type	SGN32
Data range	MIN(SGN32) ≤ i[i] ≤ MAX(SGN32)
Dimension	----
Default value	0
Remarks	Available as of V3.1.3108

3.6 Error messages

Error ID	Description
ID 50752	Geometric feed adaptation returns error during activation.
ID 293100	Requested COM-interface has not been configured.
ID 293101	Insufficient memory for management of COM interface.
ID 293102	The directory of the TcCom-CNC-interface objects not exist.
ID 293103	ISGCtrl not yet initialised.
ID 293104	Specified ID of COM interface is not configured.
ID 293105	Specified COM interface cannot be stored internally.

4 Dynamic Contour Control

4.1 Overview

Task

Physical tool deformation can result in differences to the programmed contour. The differences can be compensated at runtime using dynamic contour control.

Release Note

The functionality is available as of CNC Build V3.1.3108.

Notice

This function is an additional option requiring a license.

Properties

Depending on the calculation mode selected, the functionality can be used

- internally with CNC functions or
- externally by a user-specific TcCOM object

.

Programming and parameterisation

The functionality can be configured, activated and deactivated using the NC command #DCC [▶ 49] and the associated channel parameters [▶ 53].

Links to other documents

For the sake of clarity, links to other documents and parameters are abbreviated, e.g. [PROG] for the Programming Manual or P-AXIS-00001 for an axis parameter.

For technical reasons, these links only function in the Online Help (HTML5, CHM) but not in pdf files since pdfs do not support cross-document linking.

4.2 Description

Due to a physical deformation of the tool (e.g. the erosion wire), the resulting contour may deviate from the programmed contour. To compensate for this error, Dynamic Contour Control (DCC) modifies the interpolated tool centre point path as a function of the current and previous contour elements, the tool parameters and an online influence of the PLC (e.g. velocity dependency).

The operating principle of Dynamic Contour Control is shown below:
The task of contour control is to influence the nominal contour by suitably modifying it so that the real actual contour corresponds to the programmed nominal contour.

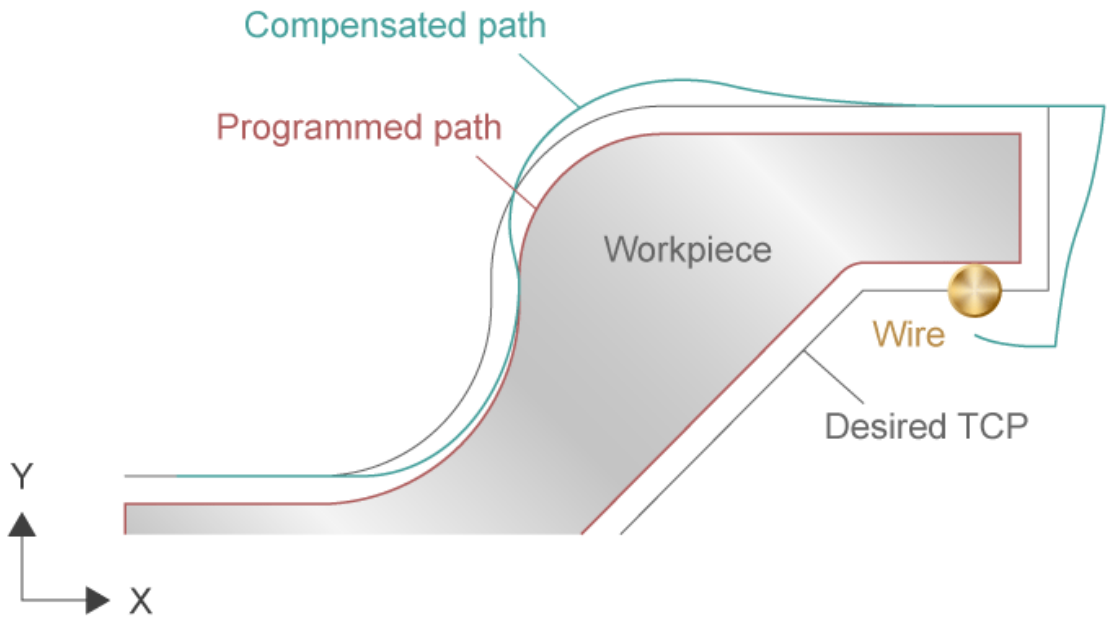


Fig. 14: Geometric mode of operation of programmed and corrected path

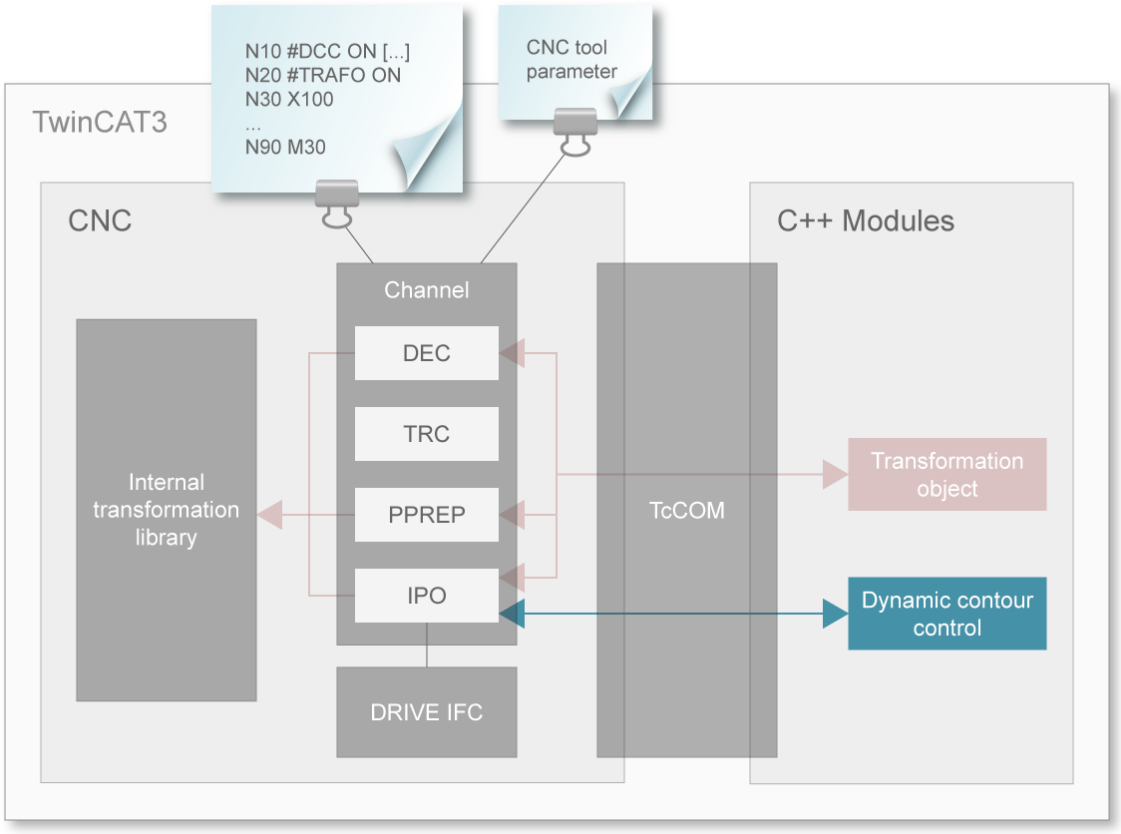


Fig. 15: Arrangement of Dynamic Contour Control within the system

Notice

With a TcCOM object with instance-specific variables, the object must be configured in the respective CNC channel.

The function is defined either with P-CHAN-00385 [▶ 53] or in the NC command #DCC [▶ 49] [**CALL**=..].

- EXTERNAL, external input
- BUILTIN, CNC-internal algorithm

4.2.1 External calculation mode

#DCC[CALL=EXTERNAL]

Calculating offsets and superimposing them on the axis positions can be executed in the methods of the TcCOM object.

The arrangement in the system of a 2-path configuration is shown below. The offset is superimposed in the CNC:

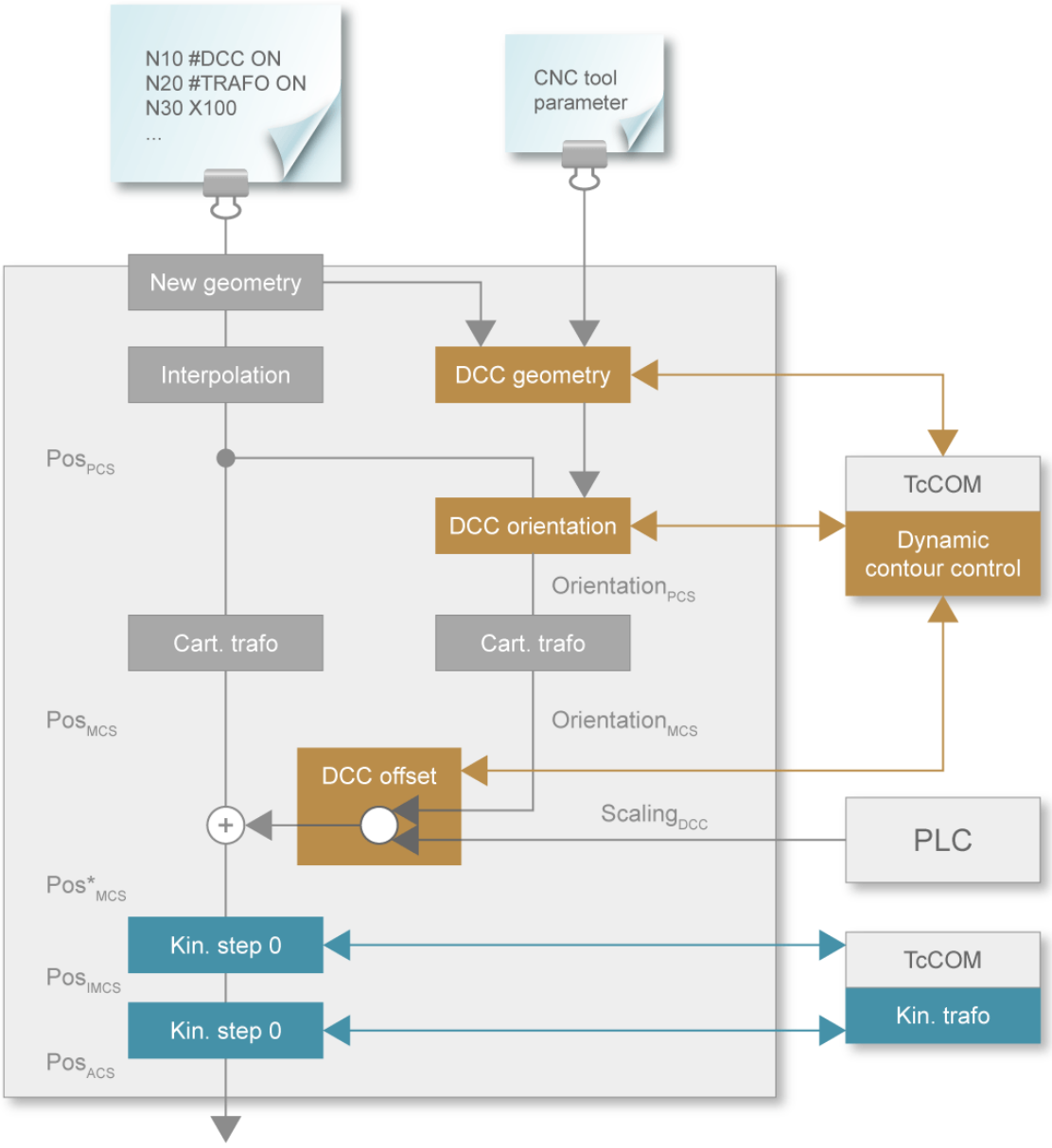


Fig. 16: Dynamic Contour Control with implicit superimposition

Motion in the ACS system is calculated using a kinematic transformation. In addition to the default parameters, the orientation of the corrected geometry in the MCS is also specified for kinematic backward transformation.

The TcCOM object can be provided with a scaling factor by the PLC via the cyclical input.

Its arrangement in the system is shown below together with the superimposition of offsets in the TcCOM object of the kinematic transformation:

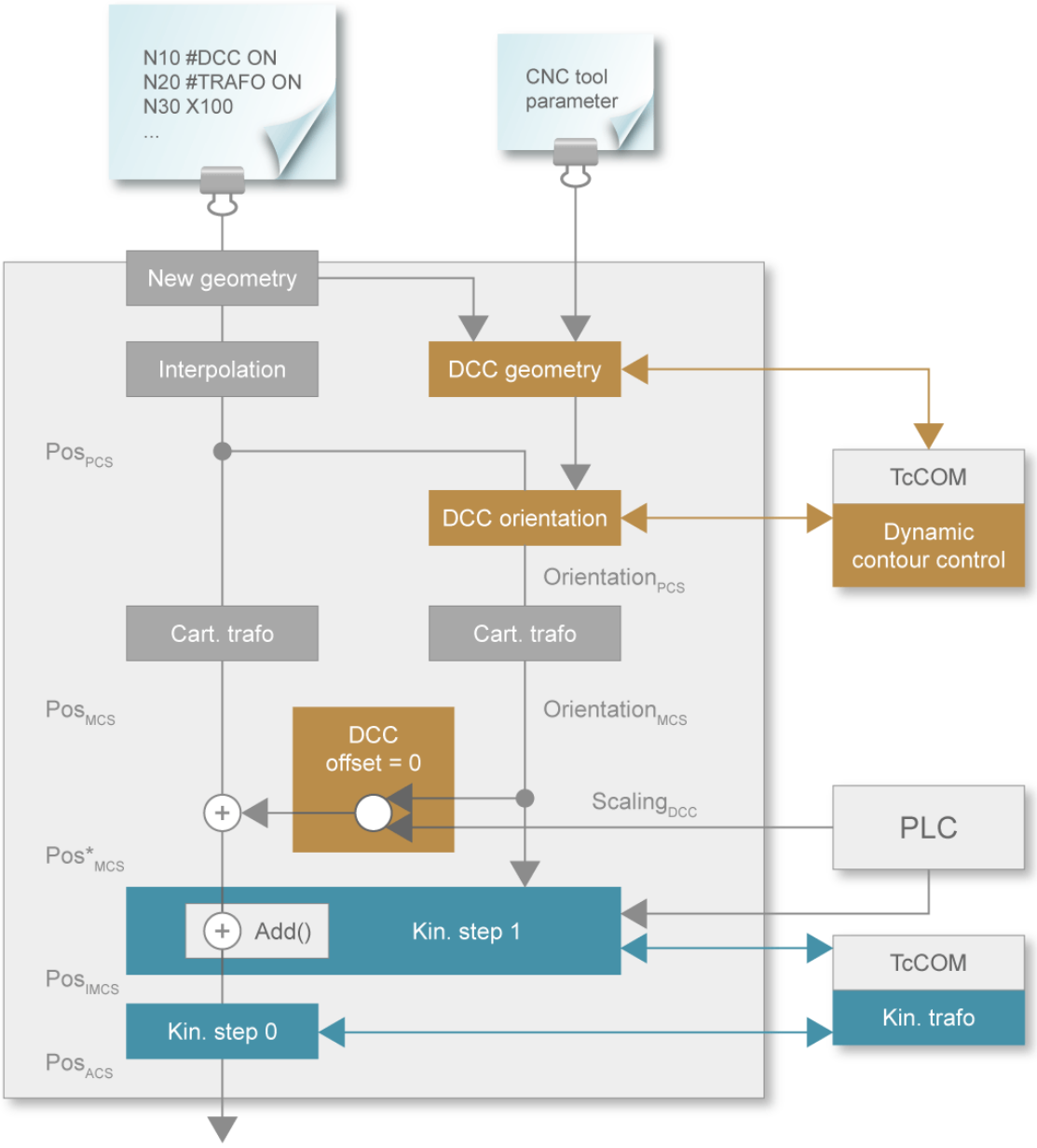


Fig. 17: Dynamic Contour Control with explicit offset superimposition in the first kinematic step

Notice

If the CNC is not required to superimpose Dynamic Contour Control offsets implicitly, the "CalculateOffset()" function must return offsets with the value 0.

You can calculate and superimpose offsets in a TcCOM object of the "kinematic transformation" type.

4.3 Programming

Syntax:
#DCC [ON/OFF] [CALL=.. [I<i>=..] [F<j>=..] { I<i>=..}{ F<j>=..}]

ON	Activate Dynamic Contour Control, analogous to P-CHAN-00384 [▶ 53].
OFF	Deactivate Dynamic Contour Control, analogous to P-CHAN-00384 [▶ 53].
CALL=..	Type of methods used for Dynamic Contour Control. Permitted identifiers: • DEFAULT - The method in P-CHAN-00385 [▶ 53] is used (default) • EXTERNAL - A customer-specific method is used
I<i>=..	SGN32 value, analogous to the P-CHAN-00389 [▶ 54] parameter. where i=0 - 3. The equals sign is mandatory, e.g. I2=17
F<j>=..	REAL64 value, analogous to the P-CHAN-00388 [▶ 53] parameter. where j=0 - 3. The equals sign is mandatory, e.g. F2=3.45

The following V.G. variables can be accessed in the NC program:

Variable	Meaning	Data type	Unit for input/output	Permitted access: Read/ Write
V.G.DCC.ACTIVE	Indicates whether Dynamic Contour Control is active.	Boolean	0 / 1	L
V.G.DCC.CALL	Indicates the method used for Dynamic Contour Control. 0: EXTERNAL	SGN32	0 / 1	L

Programing Example

#DCC Dynamic Contour Control variants

```
; use default settings
N10 #DCC ON ; activate
N15 #DCC OFF ; deselect DCC

; Use external method
N20 #DCC ON [CALL=EXTERNAL] ; activate
N30 #DCC [CALL=DEFAULT] ; reset to default
N40 #DCC ON ; activate using current method

; external method with parameters
N50 #DCC ON [CALL=EXTERNAL I0=1 F0=2.2 I1=7]
```

4.4 Adding Dynamic Contour Control via TcCOM

4.4.1 Interface of the methods

Methods to be implemented

The following methods must be implemented for Dynamic Contour Control (TcCncDynContourControlInterface.h):

- virtual HRESULT TCOMAPI CalculateOnNewGeometry(PTcDynContourCtrlParam dcc)=0;
- virtual HRESULT TCOMAPI CalculateOrientation(PTcDynContourCtrlParam dcc)=0;
- virtual HRESULT TCOMAPI CalculateOffset(PTcDynContourCtrlParam dcc)=0;

CalculateOnNewGeometry	The calculation takes place for every new geometrical element. The method calculates a normalisation factor depending on the current geometrical transition (see table below) and the tool parameters. Input: Current parameters of Dynamic Contour Control
CalculateOrientation	The calculation takes place for every new position The method calculates the new orientation compensation depending on the current position. Input: Current parameters of Dynamic Contour Control
CalculateOffset	The calculation takes place for every new position The calculation of the axis offsets in the MCS is dependent on the previously calculated compensation orientation. Axis offsets are superimposed on the axis positions. Input: Current parameters of Dynamic Contour Control **Note**: If offsets are superimposed in a kinematic transformation, the offsets of this function must have the value 0.

Geometrical transition value	Meaning
0	Position change
1	Standstill in linear motion
2	Standstill in circular motion
3	Transition from linear motion to linear motion
4	Transition from circular motion to linear motion
5	Transition from linear motion to circular motion
6	Transition from circular motion to circular motion

4.4.2 Instance data

Working data - TcCncDynContourCtrlParameter

Parameters of the methods

The parameters for each of the methods are transferred in encapsulated form by the structure TcCncDynContourCtrlParam (TcCncDynContourCtrlInterfaces.h):

```
struct TcDynContourCtrlParam
{
  // config: EcDynContourCtrl_ParamStandard
  EcDynContourCtrl   type;
  // config: dimension of path : 1 or 2
  int                dim_path;
  // config: dimension of tool parameter
  int                dim_tool_param;

  // input : <n>-dimensional actual parameter of tool
  const double       * i_tool_param;
  // input : path velocity [m/s]
  double             i_velocity;
  struct TcDynContourCtrlPath path[2];
}
```

Path-specific parameters

```
struct TcDynContourCtrlPath
{
  // calculation on new block: CalculateOnNewGeometry()
  // input: type of element, see enum EtDynContourCtrl
  int                 i_element_type;
  // input: radius of circle
  double              i_circle_radius;
  // input : actual tangent change
  //(== 0 for C2-continuous contour elements), in radiant
  double              i_tangent_variation;
  // output: normalization factor, dependent on geometry and technology
(tool parameters): X0
  double              * o_norm_factor;

  // calculation orientation on new sample
  // position: CalculateOrientation()
  // input : normalization factor: X0
  double             i_norm_factor;
  // input : 3-dim actual position of path: X,Y,Z
  const double      * i_position;
  // in-/output : 3-dim normalized orientation
  double            * o_orientation_pcs;

  // calculation offset: CalculateOffset()
  // input : 3-dim normalized orientation
  double            * i_orientation_mcs;
  // output: 3-dim offset of path
  double            * o_offset;
}
```

4.4.3 Configuring and registering a TcCOM object

Registering in TwinCAT

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

- Type 4 (TCCNC_REGISTEROBJECT_TYPE_DYN_CONTOUR_CTRL) is default.
- If the TcCOM object does not use instance-specific objects, set the Group entry to 0.
- If the TcCOM object uses instance-specific variables, set the Group entry to the respective channel number [1;12] assigned to the object.
  Maximum one object per channel.
- Index is not used.

The Dynamic Contour Control object is registered via the following TcCOM interface which is defined in the file TcCncInterfaces.h:

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

Supplying the TcCOM object

After Dynamic Contour Control is generated, there should be 2 files available:

1. TMC file
2. Driver file

Dynamic Contour Control is described in the TMC file, e.g. TcCncMyDynContCtrl.tmc.

The file is located in the work directory of the solution.

The drive file directory is dependent on Release or Debug;

- <TwinCAT> \3.1\sdk_products\TwinCAT RT (x64)\Release or
- <TwinCAT> \3.1\sdk_products\TwinCAT RT (x64)\Debug

When the configuration is activated, the respective driver file is automatically copied to the directory <TwinCAT>\3.1\Driver\AutoInstall.

Based on the above example name: TcCncMyDynContCtrl.sys

Notice

You only need to trigger the generation (Debug/Release) and activate the associated configuration.

The procedure for debugging the created Dynamic Contour Control is analogous to debugging an McCOM transformation. This procedure is described in [McCOM transformation//Debugging the transformation].

Loading the object

Loading the object for Dynamic Contour Control is described in “Integrate Dynamic Contour Control object [▶ 65]”.

4.5 Parameter

P-CHAN-00384	Select/deselect the Dynamic Contour Control function
Description	This parameter selects/deselects the Dynamic Contour Control function.
Parameter	dcc.active
Data type	BOOLEAN
Data range	0/1
Dimension	----
Default value	0
Remarks	Available as of V3.1.3108

P-CHAN-00385	Select calculation method (Dynamic Contour Control)
Description	This parameter defines that the Dynamic Contour Control function is calculated with a TcCOM object.
Parameter	dcc.call
Data type	STRING
Data range	EXTERNAL, BUILTIN
Dimension	----
Default value	EXTERNAL
Remarks	Available as of V3.1.3108 BUILTIN is only for internal tests

P-CHAN-00388	REAL64 dynamic contour control input parameters (Dynamic Contour Control)
Description	A total of four REAL64 input parameters can be defined for Dynamic Contour Control.
Parameter	dcc.param.f[i] where i = 0…3
Data type	REAL64
Data range	MIN(REAL64) ≤ f[i] ≤ MAX(REAL64)
Dimension	----
Default value	0.0
Remarks	Available as of V3.1.3108

P-CHAN-00389	SGN32 Dynamic Contour Control input parameters
Description	A total of four SGN32 input parameters can be defined for Dynamic Contour Control.
Parameter	dcc.param.i[i] where i = 0…3
Data type	SGN32
Data range	MIN(SGN32) ≤ i[i] ≤ MAX(SGN32)
Dimension	----
Default value	0
Remarks	Available as of V3.1.3108

4.5.1 Parameterisation example

All Dynamic Contour Control settings can be executed in the channel parameters. They are valid at every program start and can be modified by the #DCC command [▶ 49] in the NC program.

Default settings:

```
dcc.active FALSE
dcc.call EXTERNAL
```

Parameterisation example with additional values:

```
dcc.active TRUE
dcc.call EXTERNAL
dcc.i0 1
dcc.f0 2.3
dcc.i1 2
dcc.f1 17.3
```

4.6 Error messages

Error ID	Description
ID 50732	Reserved memory for TwinCAT3 TcCOM dynamic contour control interface too small.
ID 50733	Dynamic contour control returns error during activation.
ID 50734	The option for dynamic contour control is not included in the current software version.
ID 50735	CalculateOnNewGeometry() of dynamic contour control returned an error.
ID 50736	CalculateOrientation() of dynamic contour control returned an error.
ID 50737	CalculateOffset() of dynamic contour control returned an error.
ID 293100	Requested COM interface has not been configured.
ID 293101	Insufficient memory for management of COM interface.
ID 293102	The directory of the TcCom-CNC interface objects not exist.
ID 293103	ISGCtrl not yet initialised.
ID 293104	Specified ID of COM interface is not configured.
ID 293105	Specified COM interface cannot be stored internally.

5 Process to generate a TcCOM object

Minimum requirements for using McCOM assistants

- TwinCAT3 Version 4024
- Microsoft Visual Studio 2019 Professional/Enterprise: During installation, additionally select the option “Desktop development with C++”.

5.1 Create new project

The procedure below is an example of how to create a user-defined kinematic transformation using a TcCOM object and was generated with Visual Studio 2019.

TwinCAT3 XAE project with CNC configuration

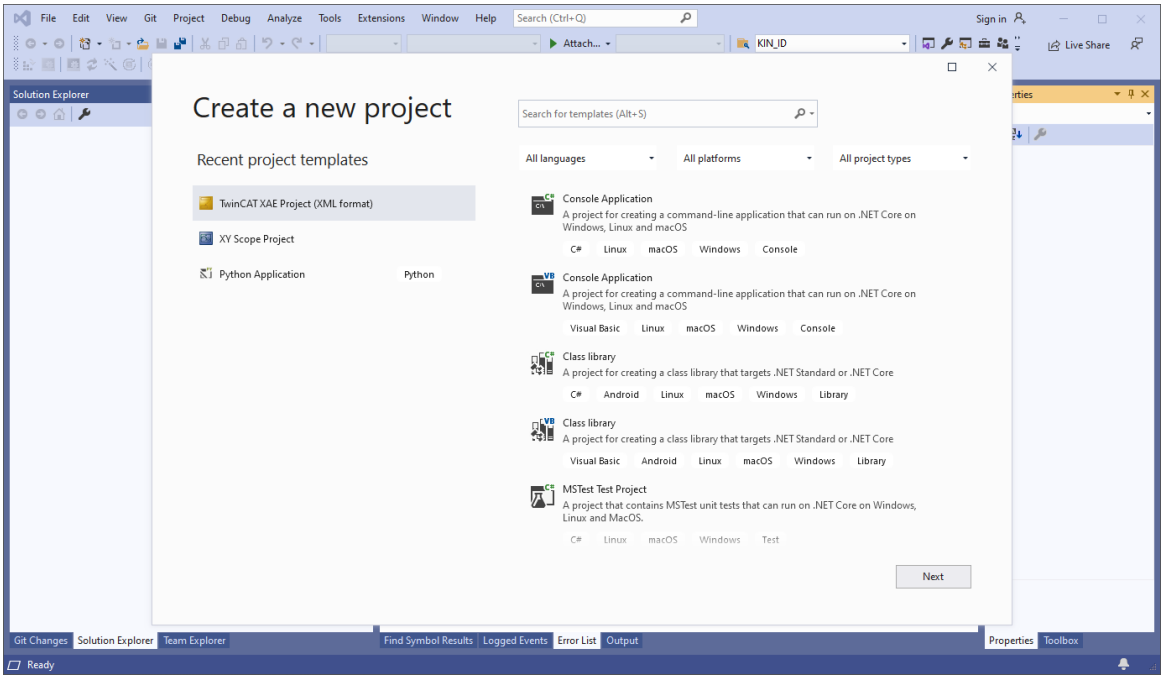


Fig. 18: Create a new project

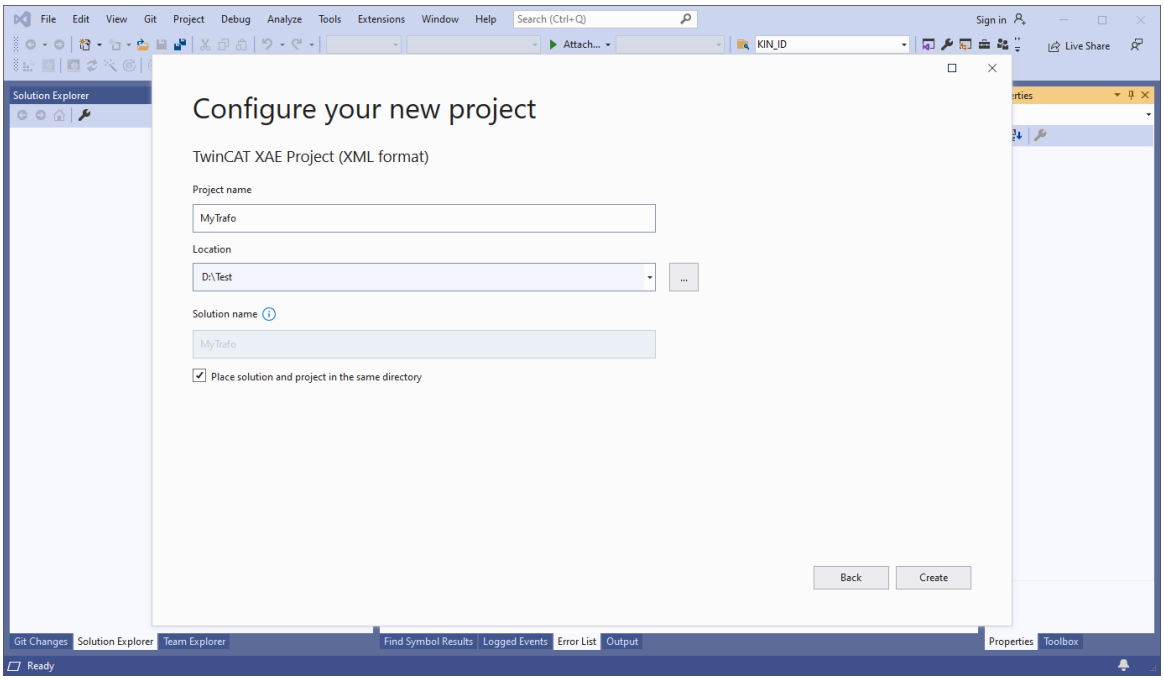


Fig. 19: Configure the new project

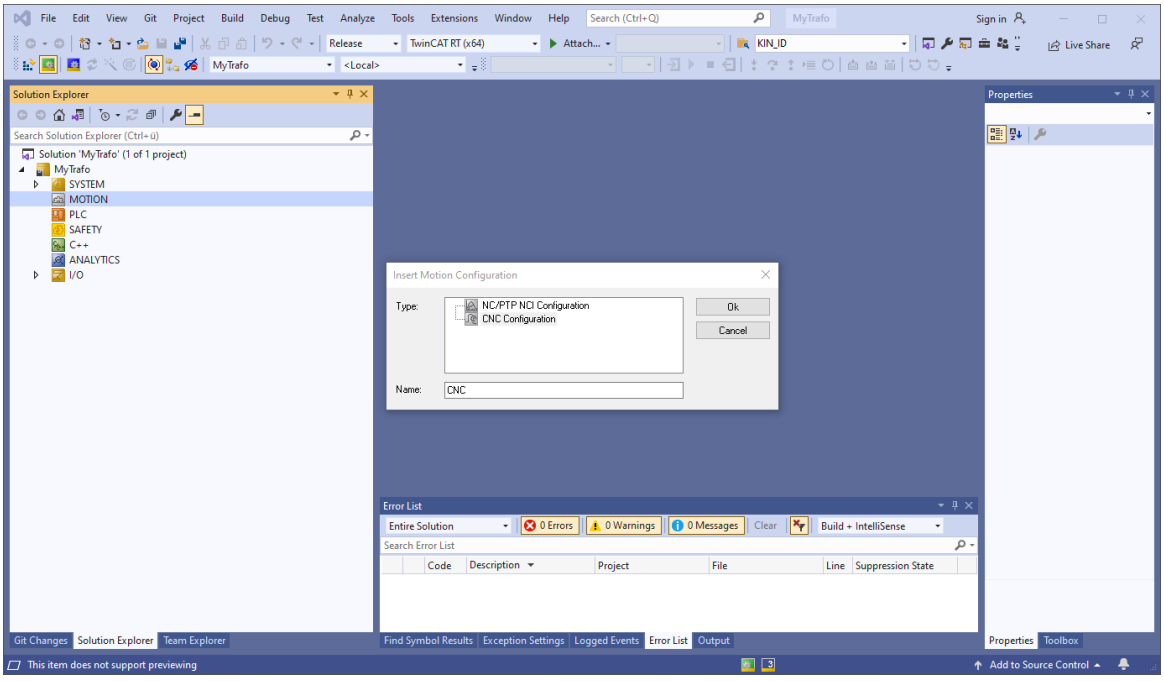


Fig. 20: Create a CNC configuration

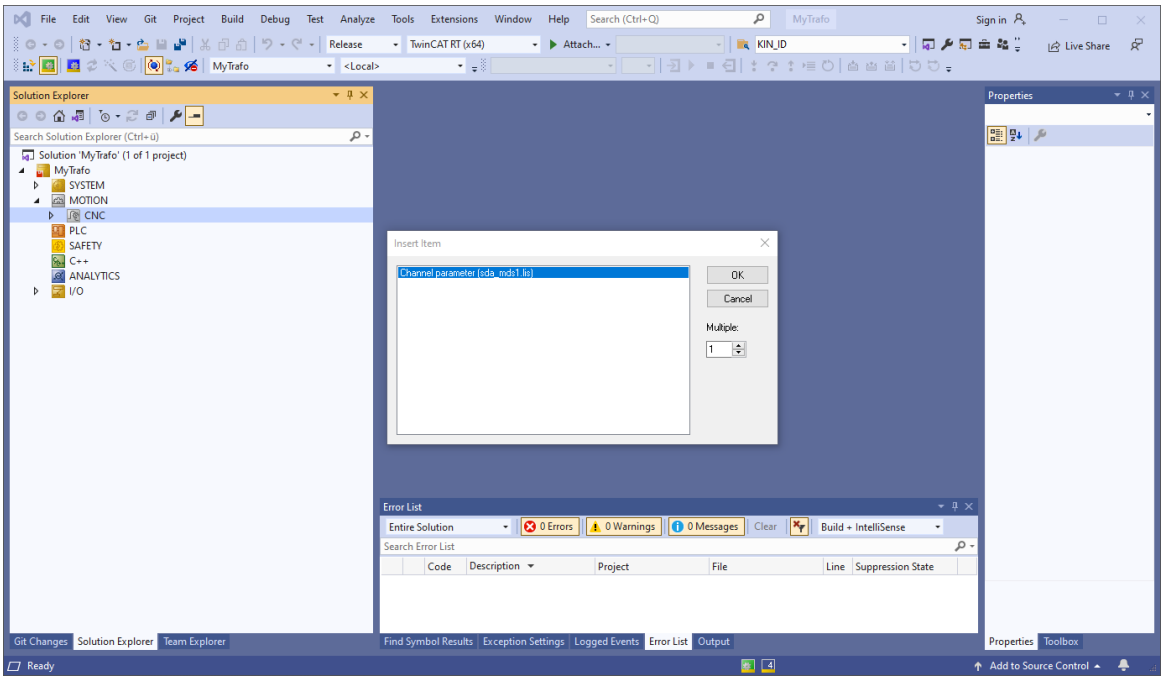


Fig. 21: Create a channel

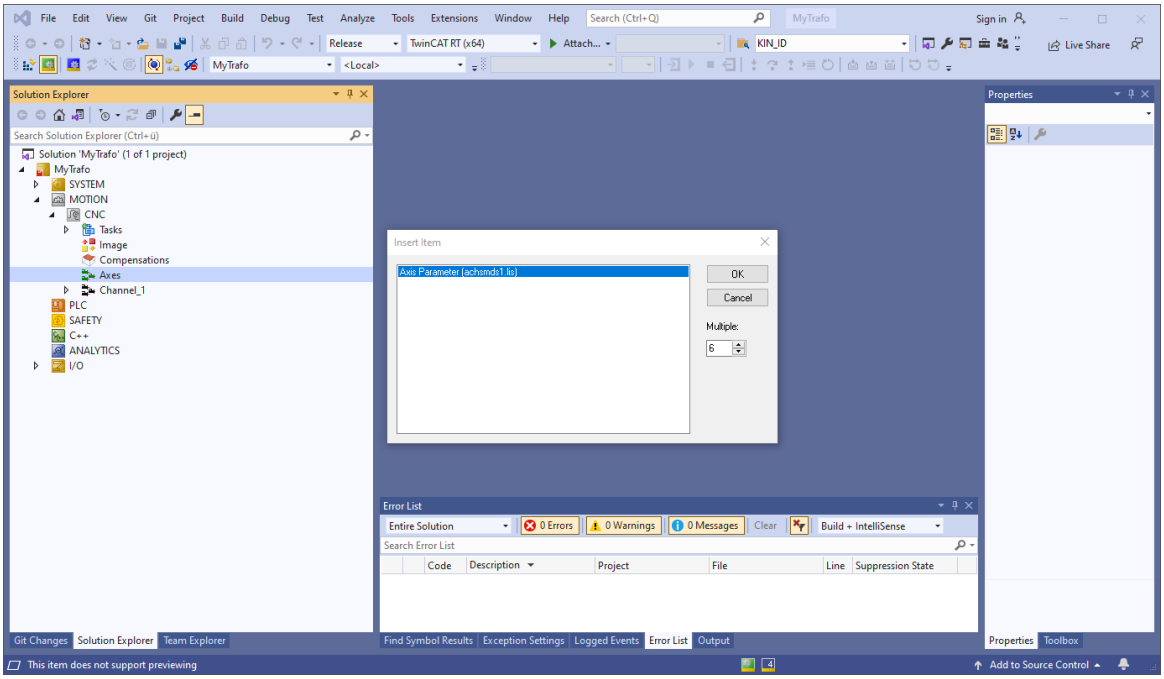


Fig. 22: Create an axis

5.2 Geometric feed adaptation

5.2.1 Create an object for geometric feed adaptation

The procedure below is an example of creating a user-defined geometric feed adaptation object using Visual Studio 2019,

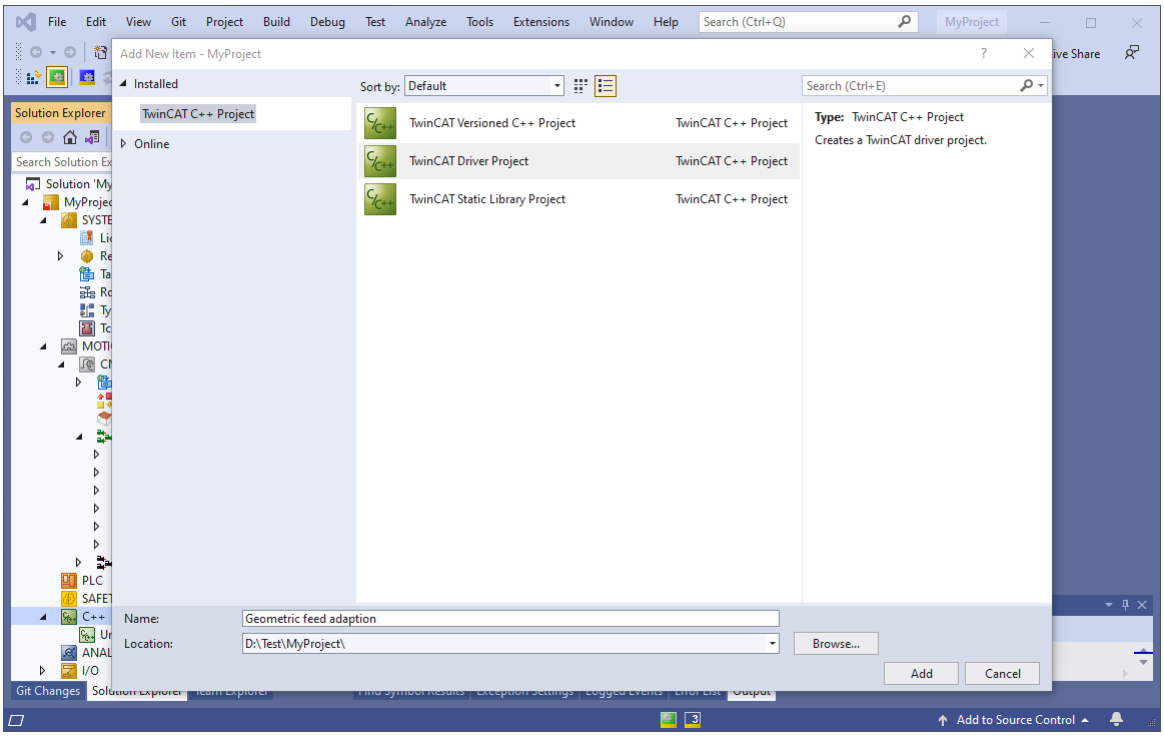


Fig. 23: Drive project for geometric feed adaptation

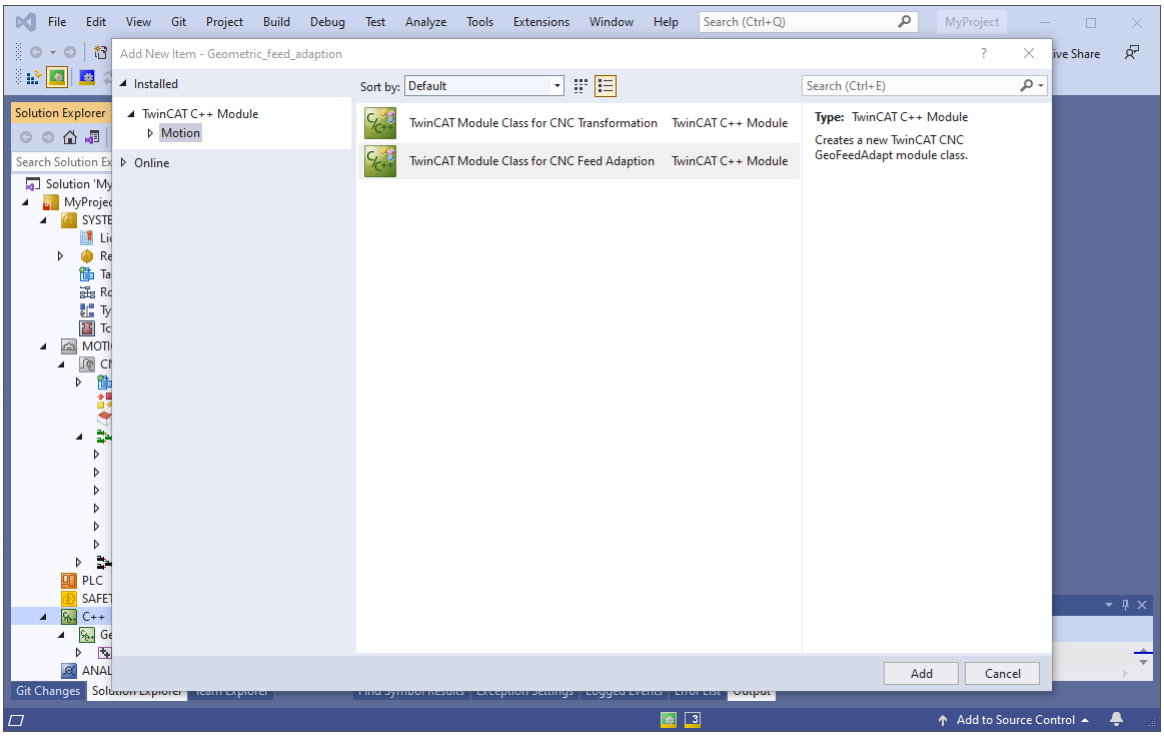


Fig. 24: Define class

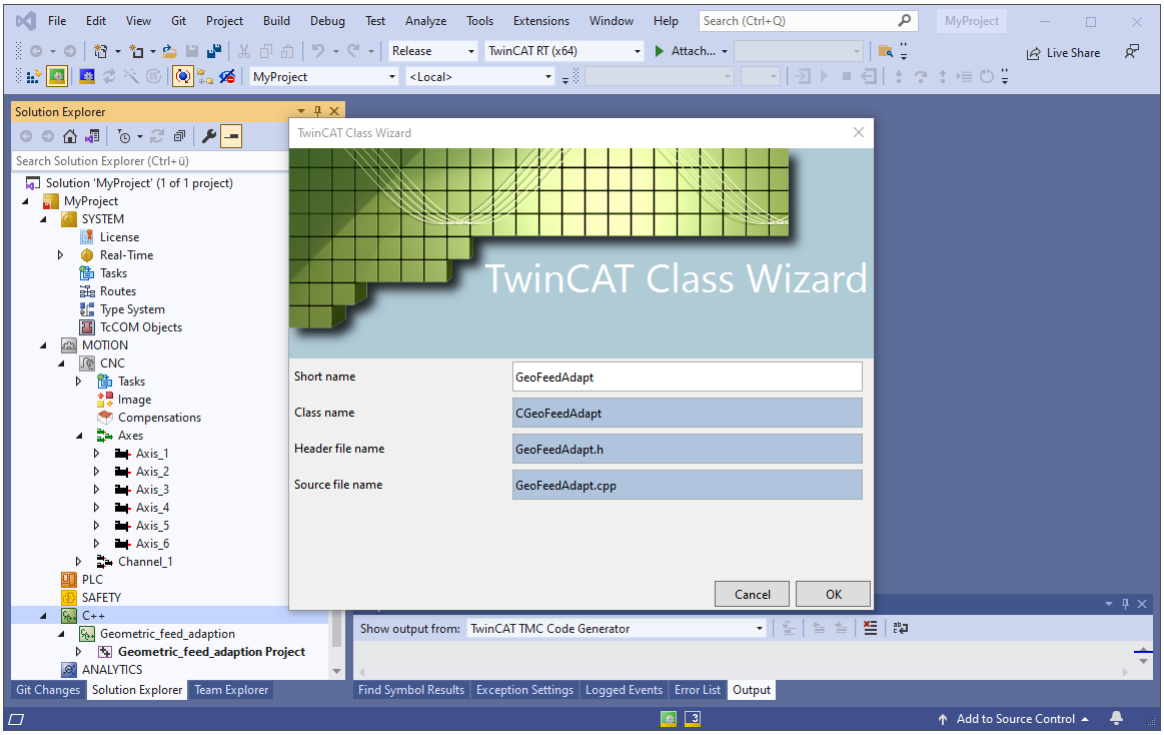


Fig. 25: Name class

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

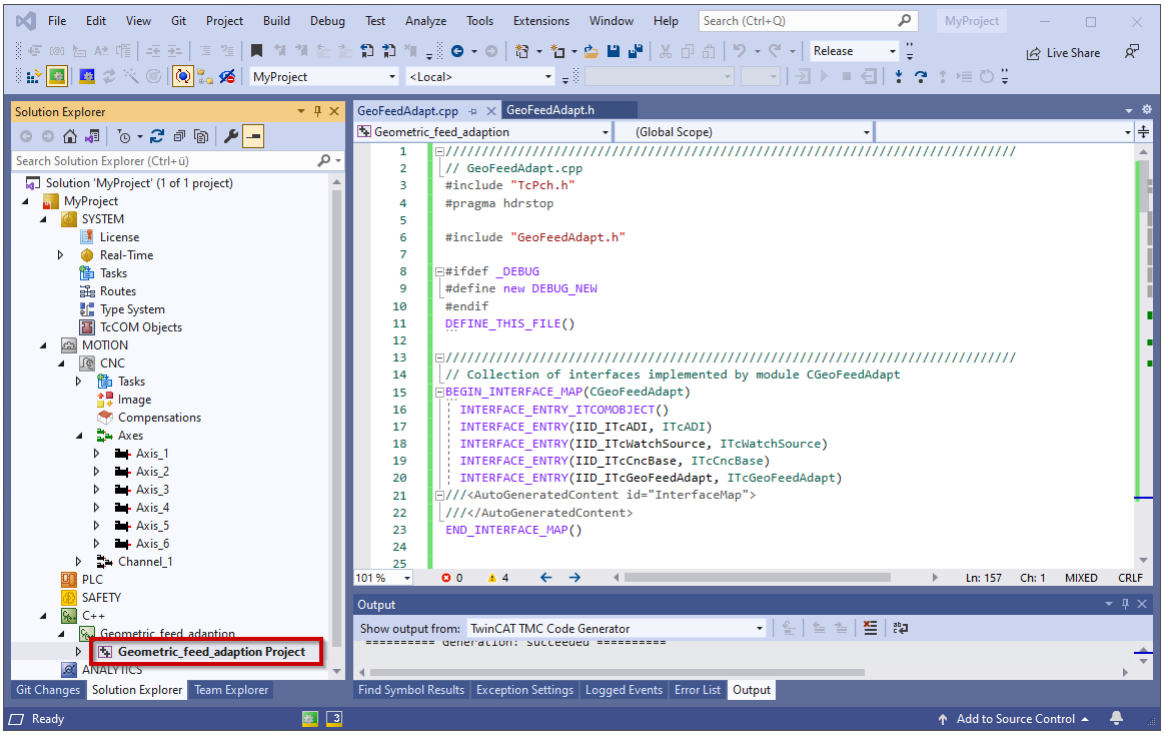


Fig. 26: Create driver

5.2.2 Integrate geometric feed adaptation object

After the geometric feed adaptation object is created, it must be integrated into the existing CNC configuration as follows:

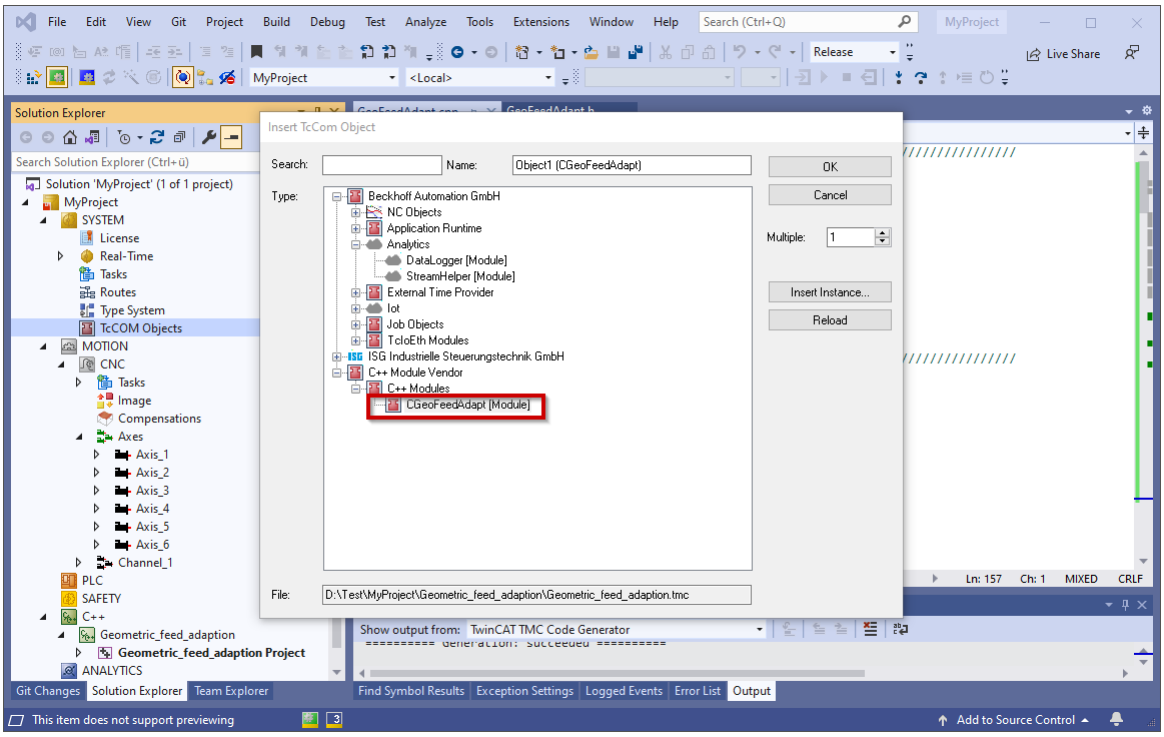


Fig. 27: Integrating the TcCOM object

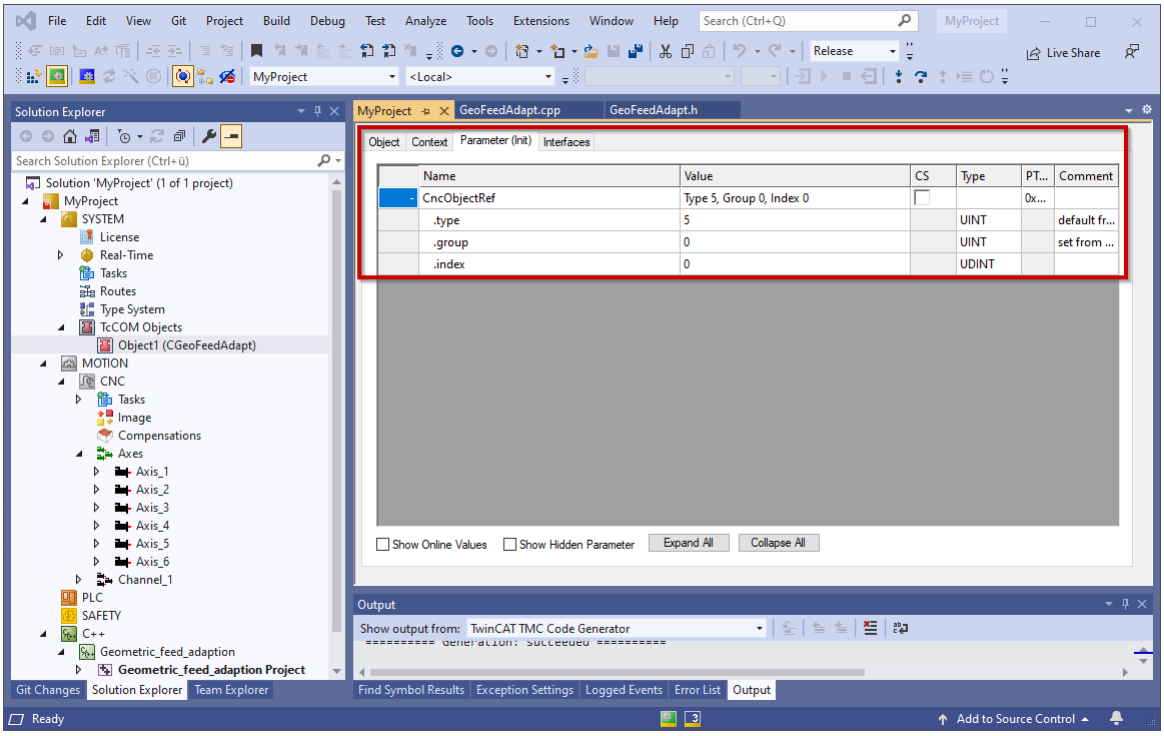


Fig. 28: Properties of the TcCOM object

The type of the geometric feed adaptation object is 5.

The channel specification is dependent on whether instance-specific variables are used in the TcCOM object.

- With instance-specific data, the group corresponds to channel number [1;12].
- If there are no instance-specific data, assign the group the value 0.

The index of each object is not used.

5.3 Dynamic Contour Control

5.3.1 Create an object for Dynamic Contour Control

The procedure below is an example of creating a user-defined Dynamic Contour Control object using Visual Studio 2019,

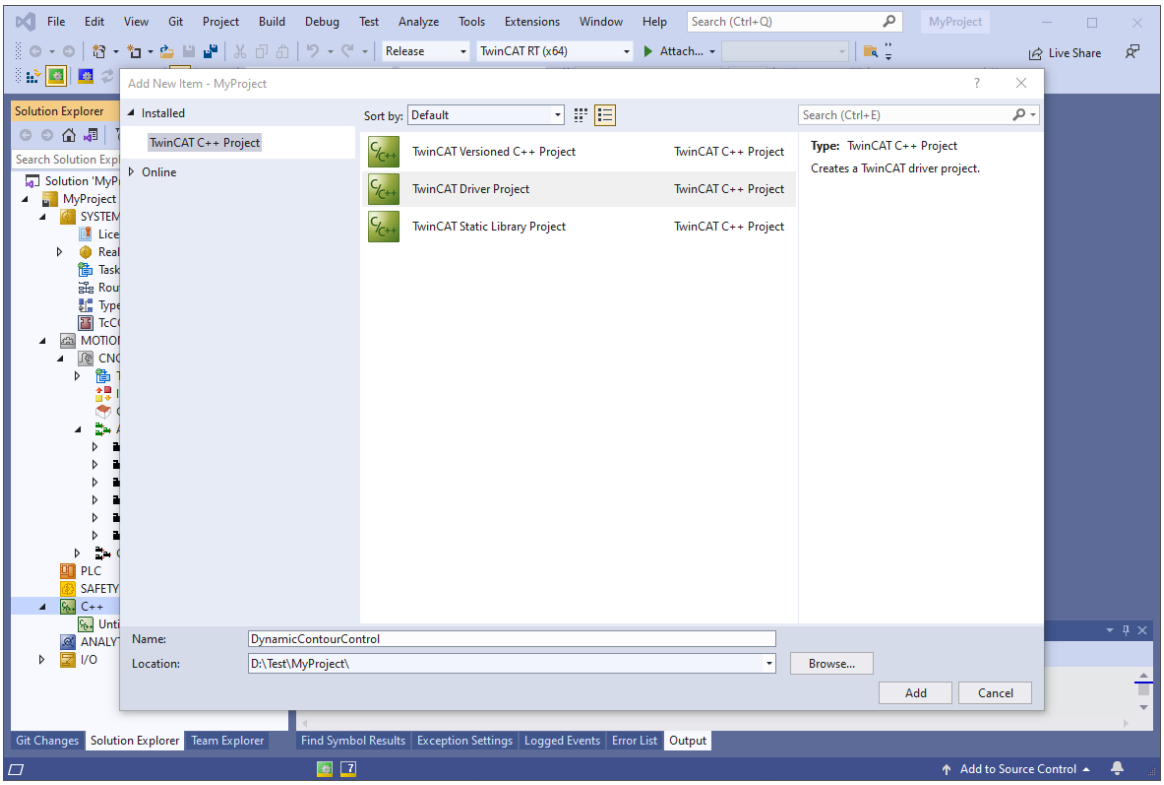


Fig. 29: Create driver project for Dynamic Contour Control

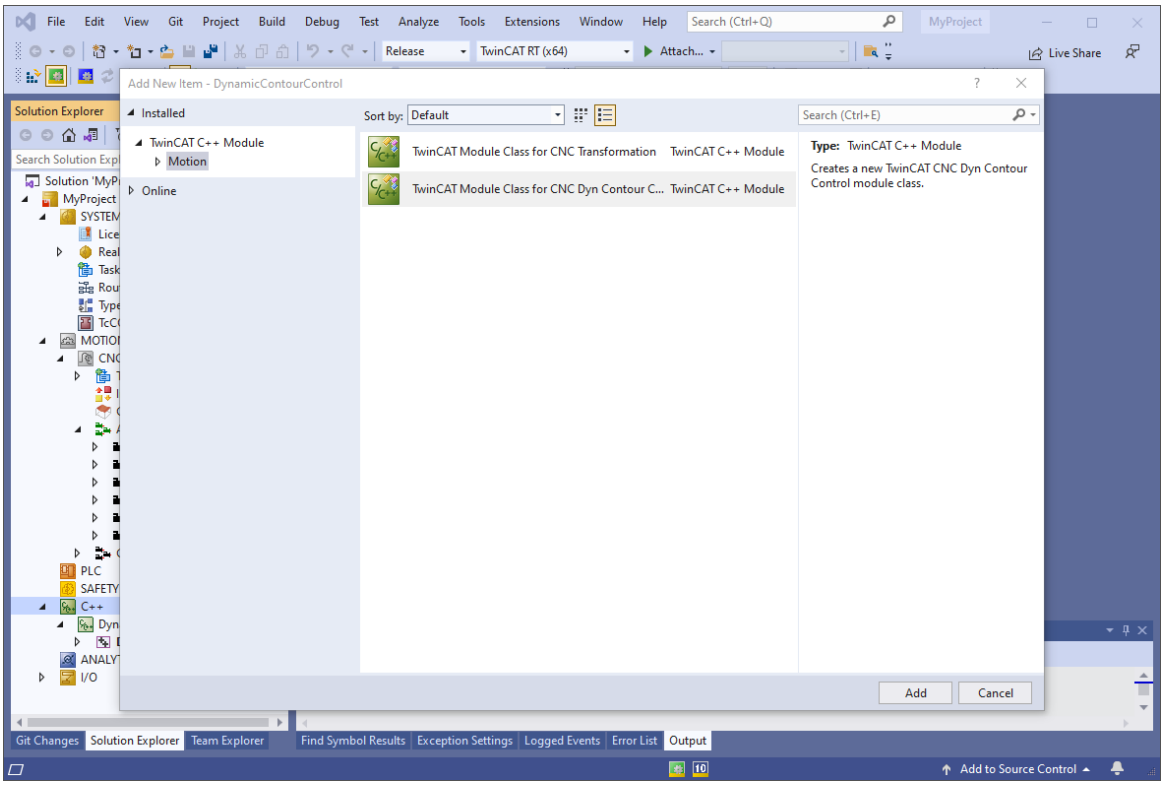


Fig. 30: Define class

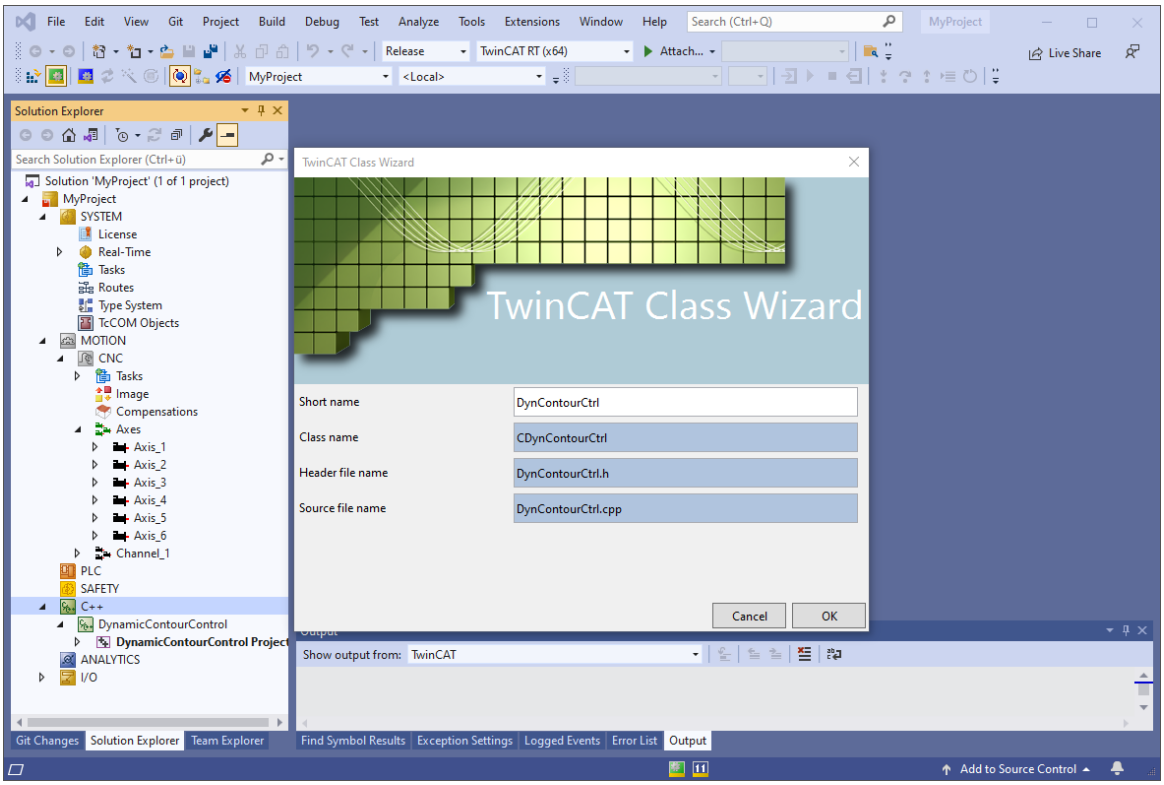


Fig. 31: Name class

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

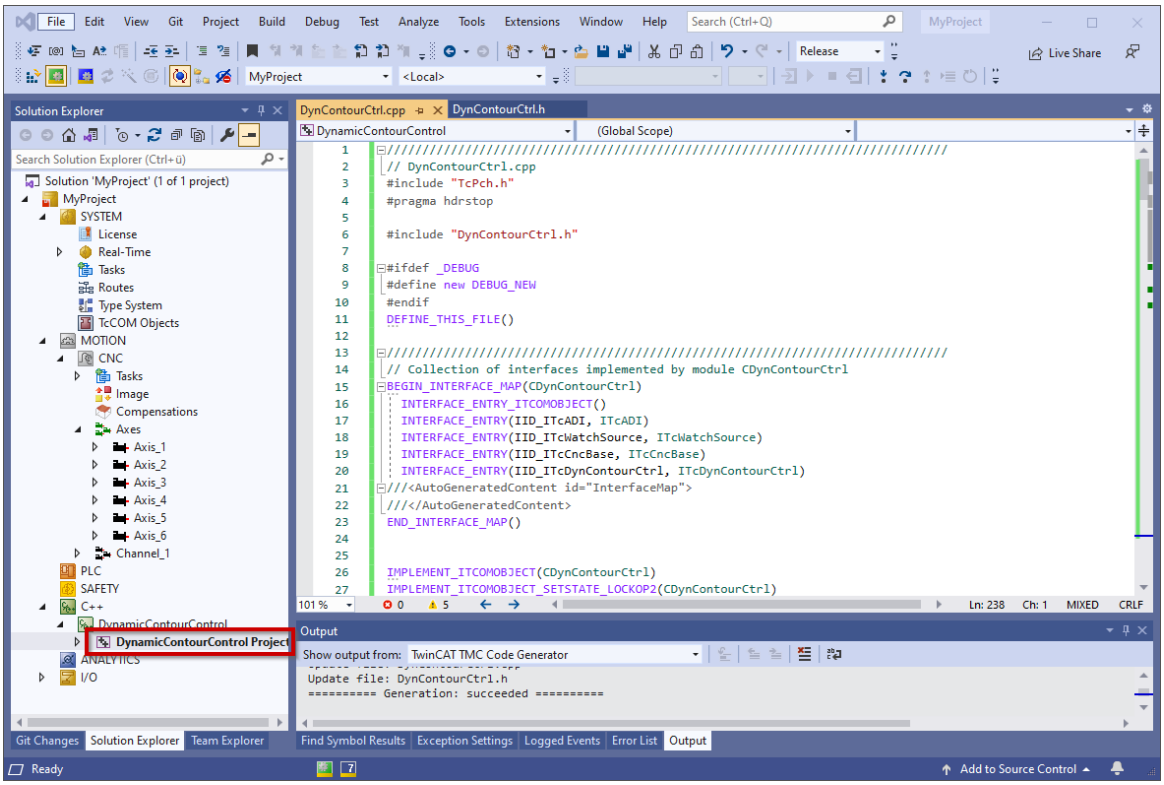


Fig. 32: Create driver

5.3.2 Integrate Dynamic Contour Control object

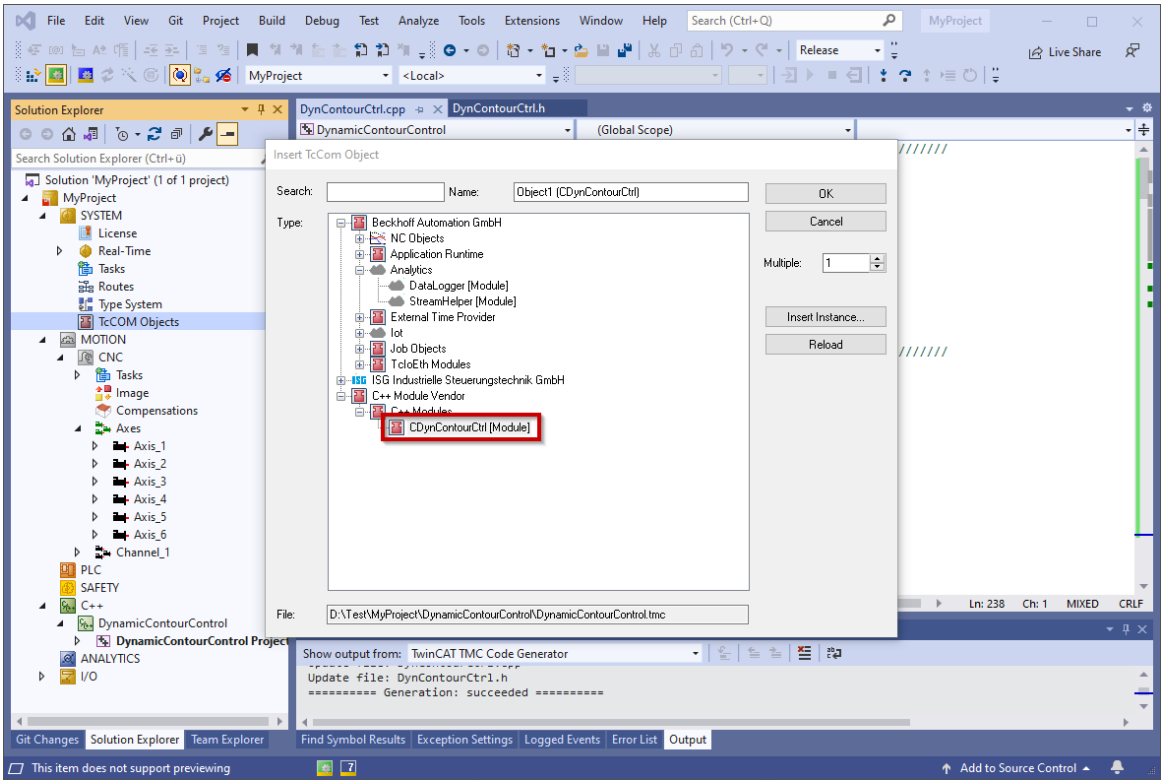


Fig. 33: Integrating the TcCOM object

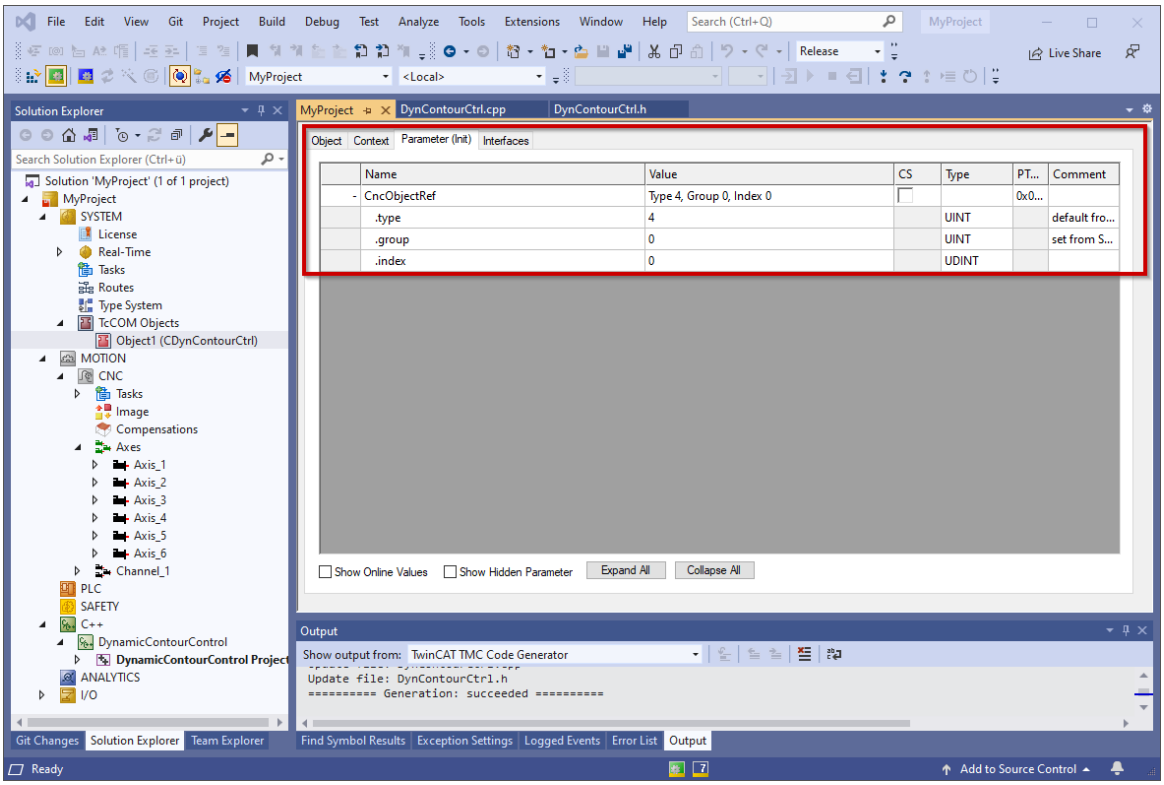


Fig. 34: Properties of the TcCOM object

The type of the Dynamic Contour Control object is 4.

The channel specification is dependent on whether instance-specific variables are used in the TcCOM object.

- With instance-specific data, the group corresponds to channel number [1;12].
- With no instance-specific data, assign the group the value 0.

The index of each object is not used.

Notice

The TcCOM object must be digitally signed, otherwise it is not loaded. For more information on TcCOM objects and how to digitally sign an object, go to Beckhoff Information System.

Only one dynamic model can be used for each channel.

5.4 Online tool radius compensation

5.4.1 Creating an object for online tool radius compensation

The procedure below is an example of creating a user-defined online tool radius compensation object using Visual Studio 2019,

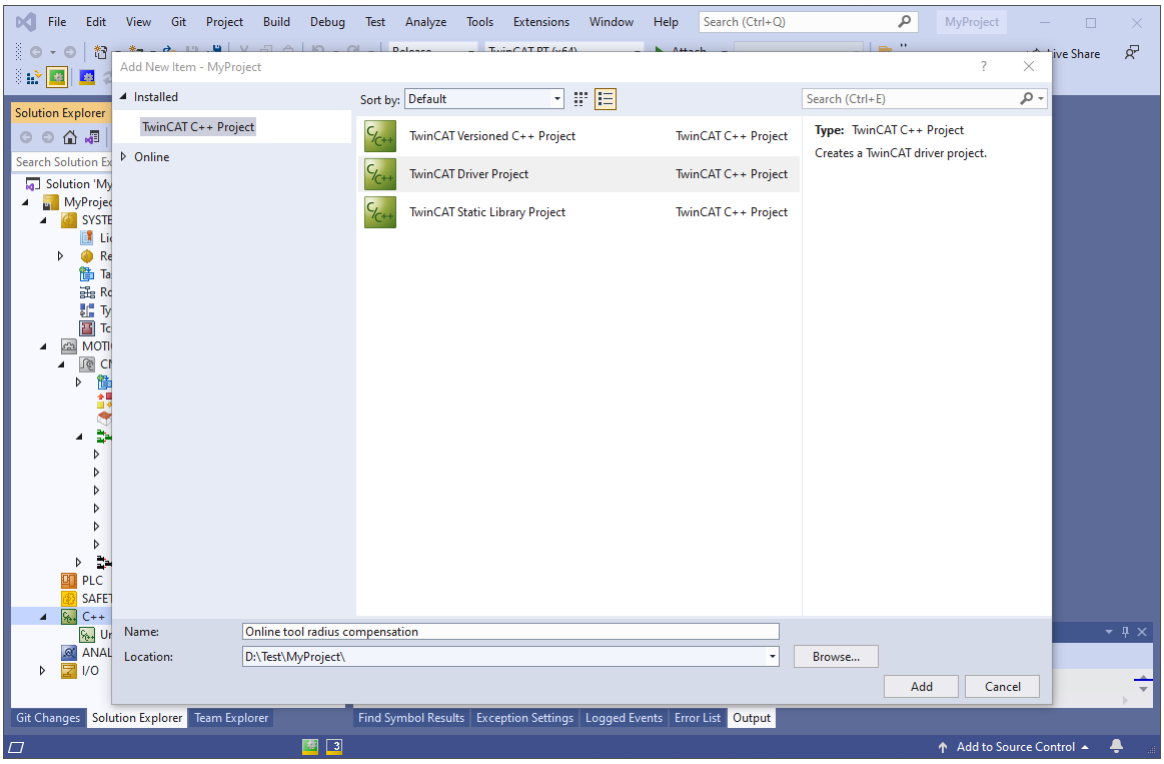


Fig. 35: Driver project for online tool radius compensation

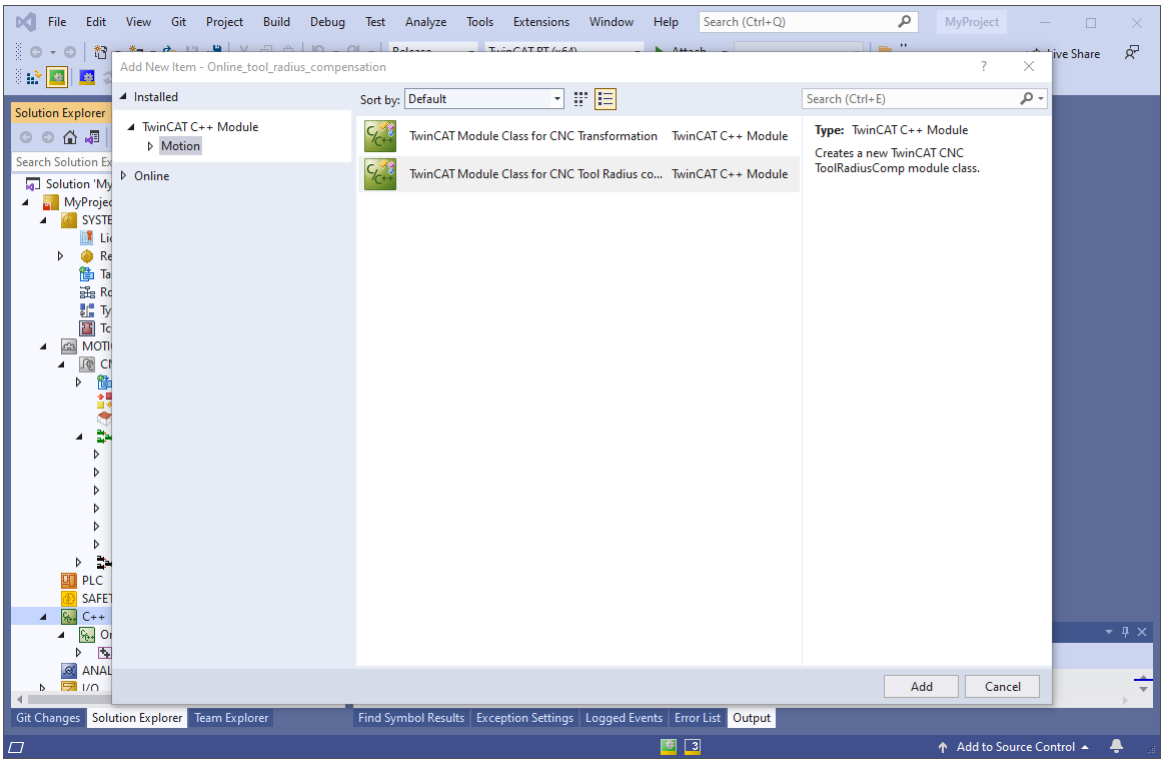


Fig. 36: Define transformation class

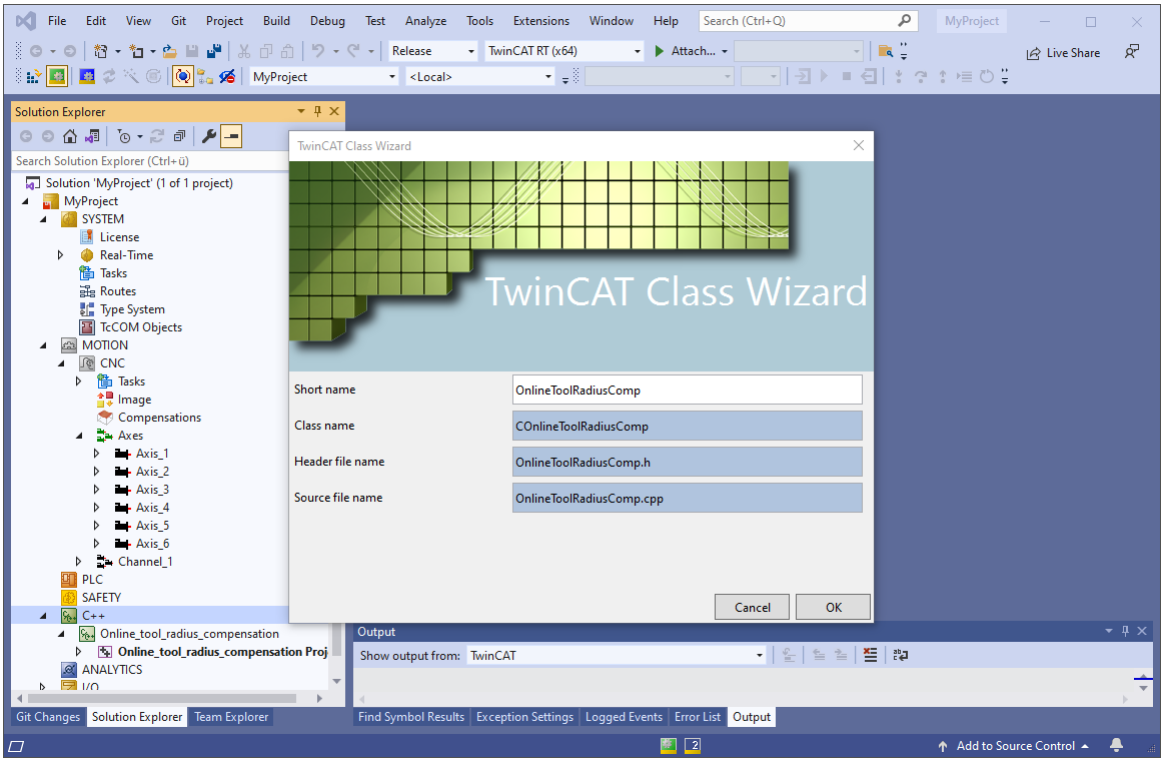


Fig. 37: Name transformation class

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

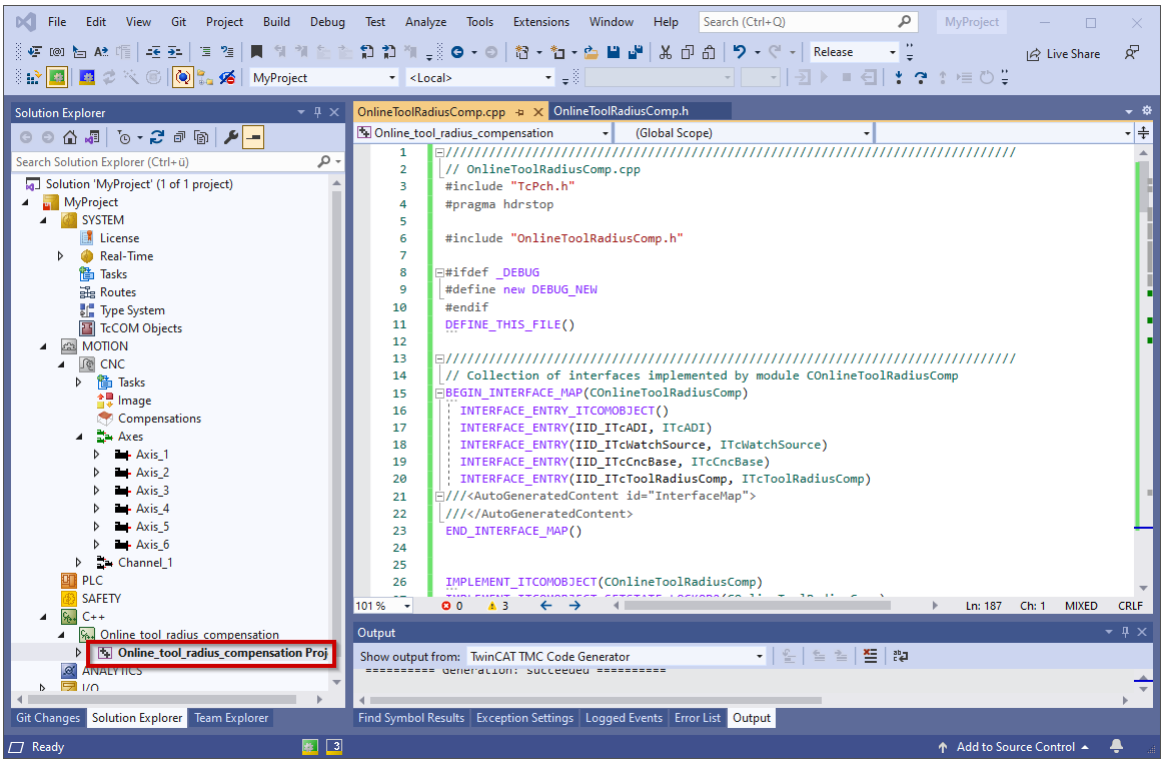


Fig. 38: Create driver

5.4.2 Integrating online tool radius compensation object

The created online tool radius compensation object must be integrated into the existing CNC configuration as follows:

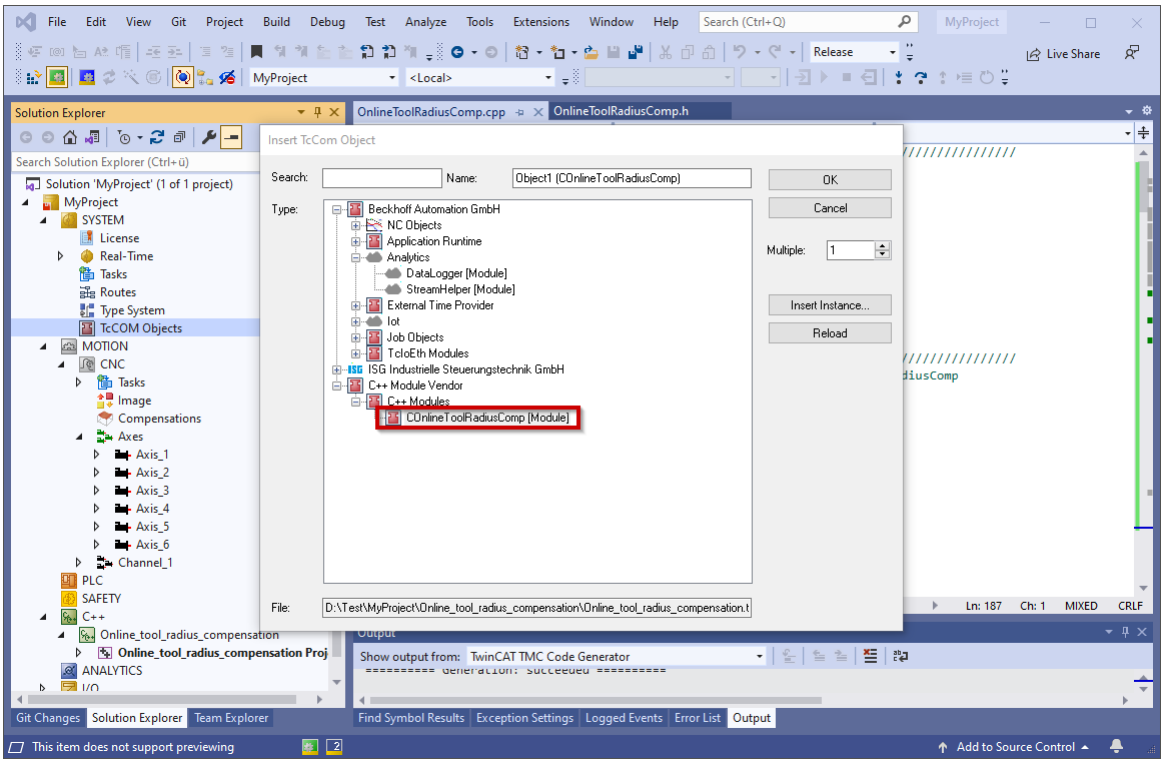


Fig. 39: Integrating the TcCOM object

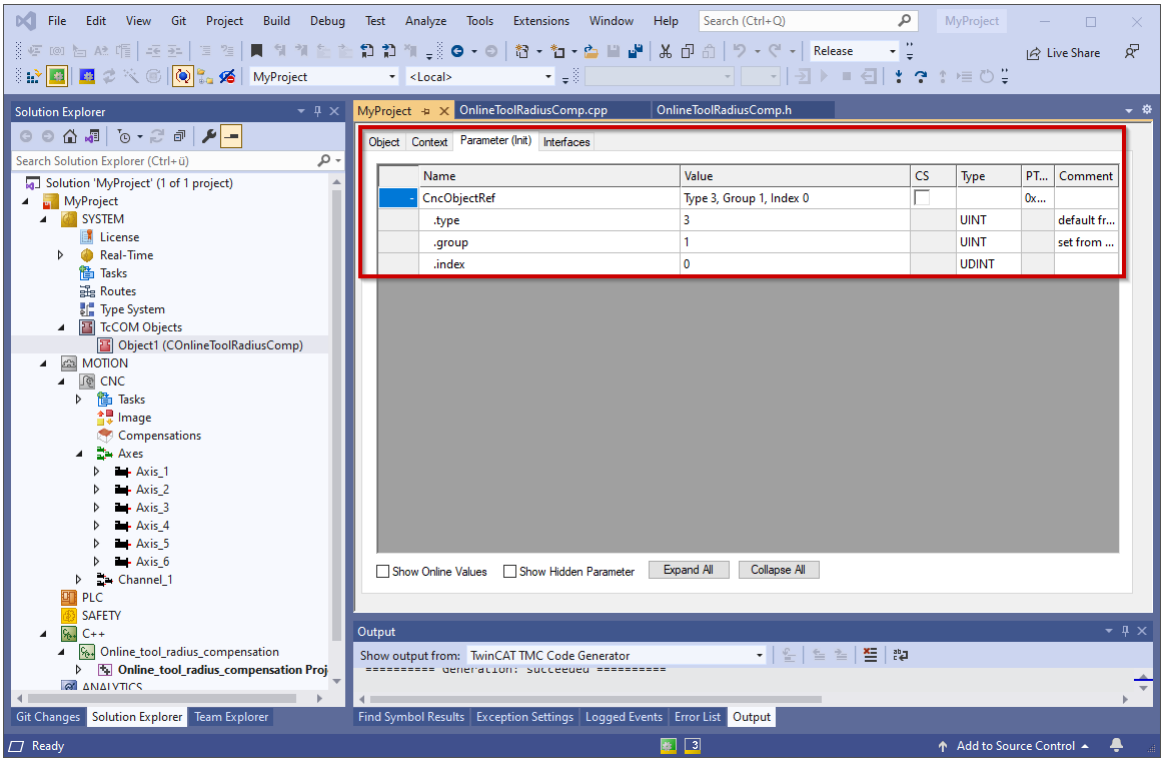


Fig. 40: Properties of the TcCOM object

The type for the online tool radius compensation object is 3.

The channel specification is dependent on whether instance-specific variables are used in the TcCOM object.

- With instance-specific data, the group corresponds to channel number [1;12].
- With no instance-specific data, assign the group the value 0.

The index of each object is not used.

Notice

The TcCOM object must be digitally signed, otherwise it is not loaded. For more information on TcCOM objects and how to digitally sign an object, go to Beckhoff Information System.

Only one online tool radius compensation object can be used for each channel.

6 Appendix

6.1 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

Email: documentation@isg-stuttgart.de

Index

P

P-CHAN-00384 53
P-CHAN-00385 53
P-CHAN-00386 41
P-CHAN-00387 41
P-CHAN-00388 53
P-CHAN-00389 54
P-CHAN-00390 41
P-CHAN-00391 42
P-CHAN-00550 33

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

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