SIEMENS

Kinematic Transformation

Configuration of kinematic transformations with the SIMOTION library *LKTrans*
(**L**ibrary **K**inematic **T**ransformation)

Ausgabe 2011/1

Subject to technical product changes without prior notice.

Copyright

Definitions and warnings

Qualified personnel

In the context of this documentation these are personnel who are familiar with the assembly, installation, commissioning, operation, and maintenance of the Siemens AG products to be used, and who have the appropriate qualifications for their job.
For example:

- Training/instruction and authorization to switch on/off, ground, and mark circuits and devices in accordance with established safety procedures.
- Training/instruction in the proper care and use of protective equipment in accordance with established safety standards.
- Trained in first aid.

Warnings are not explicitly given in this documentation. However, you are strongly advised to observe the warning information contained in the operating instructions for each product.

Disclaimer of liability

The standard applications are provided to you free of charge. They may be copied and used, and be passed on to third-parties. However, in the case of them being passed on to a third party, they must be in their full and unmodified form, and all copyright conditions must be observed. Any commercial distribution to third parties (e.g. for shareware/freeware distribution) is subject to the express written approval of Siemens AG. SINCE THE APPLICATION EXAMPLES ARE PROVIDED FREE OF CHARGE; THE AUTHORS AND COPYRIGHT HOLDERS CANNOT ACCEPT LIABILITY FOR THIS. THEIR USE IS THE SOLE RISK AND RESPONSIBILITY OF THE USER. THE AUTHORS AND COPYRIGHT HOLDERS SHALL ONLY BE HELD LIABLE FOR MALICIOUS INTENT AND GROSS NEGLIGENCE. ANY FURTHER CLAIMS ARE EXCLUDED. IN PARTICULAR, THE AUTHORS AND COPYRIGHT HOLDERS CANNOT ACCEPT LIABILITY FROM ANY DAMAGES ARISING FROM DEFECTS OR CONSEQUENCES ARISING FROM SUCH DEFECTS. Should you discover any errors in the application examples, please inform us.

Valid conditions

If nothing else was negotiated, the "Terms and Conditions for Deliveries and Services for Siemens Internal Transactions" apply in their current valid version at the time of purchase of the equipment.

Note on trademarks

SIMOTION® is a trademark of the Siemens AG.
SINAMICS® is a trademark of the Siemens AG.
MASTERDRIVES® is a trademark of the Siemens AG.
SIMODRIVE® is a trademark of the Siemens AG.
SIMATIC® is a trademark of the Siemens AG.

Notice regarding export identification codes

AL: N
ECCN: N

Preface

This manual describes the use of function blocks of the SIMOTION "LKTrans" application library. This documentation is intended exclusively for qualified commissioning and service personnel. This manual is intended as a supplement to the SIMOTION standard documentation.

The following is required when working with SIMOTION and the "LKTrans" application:

- Basic knowledge of the SIMOTION system
- Familiarity with the SIMOTION SCOUT engineering system
- SIMATIC STEP 7: Knowledge of the HW Config and NetPro
- Expertise in the parameterization and optimization of axis control loops
- Knowledge of the used converters with respect to parameterization of the required settings and communication via PROFIBUS DP, as well as optimization of the closed-loop speed control

The "LKTrans" library does not require a license.

Real axes require licenses. Details can be found in the SIMOTION documentation.

Table of contents

List of figures

List of abbreviations

FB	Function block
FC	Function
TO	Technology object
LKTRANS	Constant prefix for Library Kinematic Transformation
LKTRANS_ERR	Constant prefix for Library Kinematic Transformation error messages

Coding standard specifications

The following prefixes apply within data structures:

1.) Floating-point variables are identified by the prefix r32 or r64.
2.) Time Variables are defined with the prefix t.
3.) BOOL bit variables are identified by the prefix bo.
4.) Data structure variables are identified by the prefix s.
5.) Enumerator variables are identified by the prefix e.
6.) Array variables are are identified by the prefix a.
7.) BYTE variables are identified by the prefix b8, WORD variables by the prefix b16 and DWORD variables by the prefix b32.
8.) DINT variables are identified by the prefix i32, INT variables by the prefix i16, SINT variables by the prefix i8, UDINT variables by the prefix u32, UINT variables by the prefix u16, USINT variables by the prefix u8.
9.) These specifications refer to the **internal view of the FBs described here**

1 Scope of functions

1.1 Representation of the field of application

Simple kinematic transformations, such as handling or robotics applications can be configured with the blocks of the standard "Kinematic transformation with SIMOTION" library. Use of the standard applications “Top loading with SIMOTION” or “Handling with SIMOTION” is not absolutely necessary for utilization of this function. However, these standard applications provide the user with all functions required to create complete handling projects. Details can be found in the respective documentation.
The blocks described here complement the technology functions of SIMOTION and the standard applications mentioned above with the following functionality:

- FBLKTransTransformation function block with uniform interface to accommodate the kinematic transformation created by the user
- FBLKTransKinTransformation function block for controlling real axes with cyclic position setpoints via the motion interface
- Before activating the motion interface, the input axes are set to the actual, transformed positions of the real output axes in order to prevent jumps
- Monitoring of individual real axes using setpoint monitoring
- Monitoring of Cartesian actual values and setpoints using path deviation

In this case, the FBLKTransTransformation function block contains the mathematical transformation equations, the FBLKTransKinTransformation takes over the control of the axes; its functions are mainly configured.

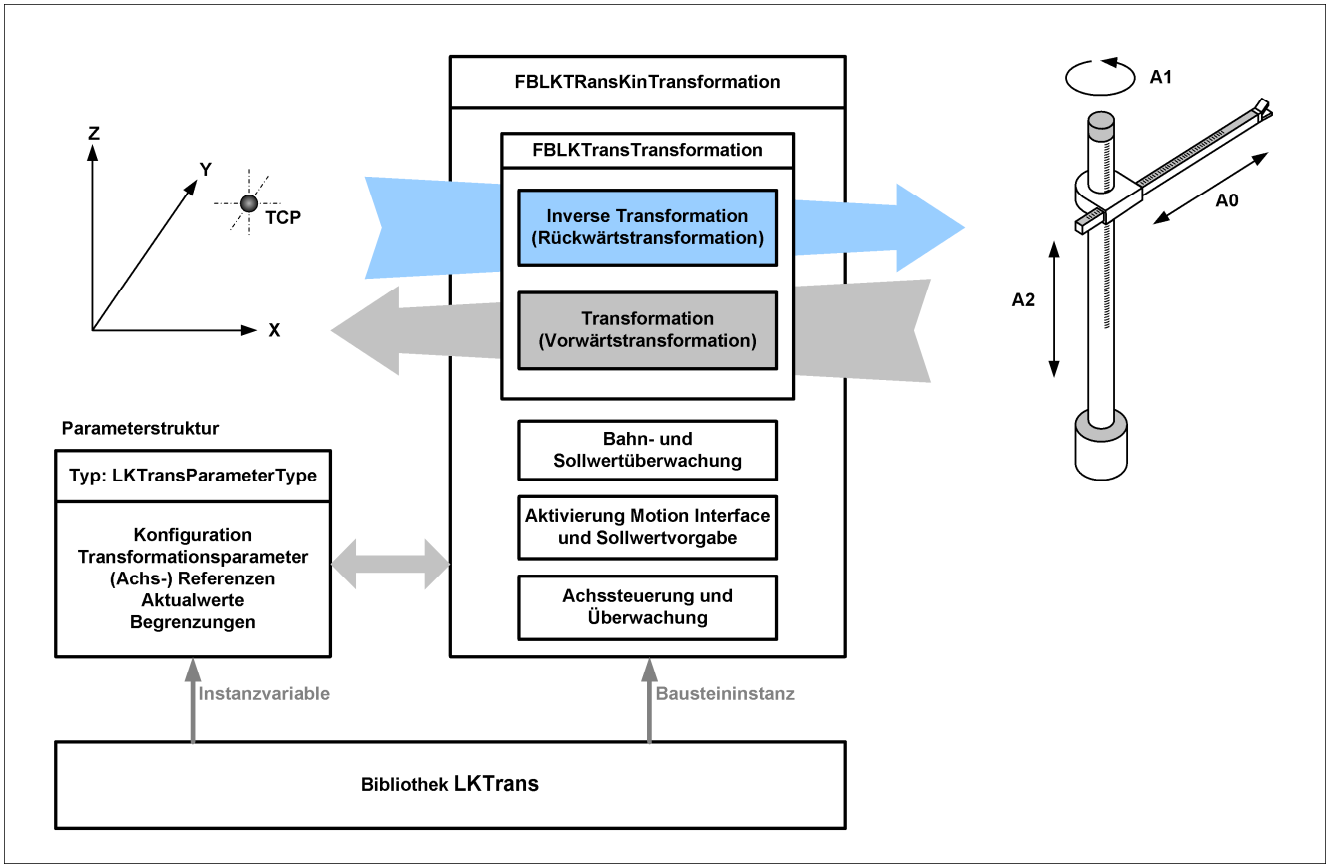


Fig. 1-1: Structure of FBLKTransKinTransformation and FBLKTransTransformation

1.2 Limitations

1.2.1 Can be used for:

- Simple, user-developed multi-axis kinematics
- Virtual axes as input variables

1.2.2 Cannot be used for:

- Replacing motion control on the output axes
- Superimposed motion control on the output axes
- Synchronous couplings in which the output axes are used as master for further axes

2 Requirements

2.1 Hardware

The library has been developed for SIMOTION and can be used on all controller versions (SIMOTION C, D or P).

2.2 Software

The library has been created with SIMOTION SCOUT / SIMOTION Runtime V 4.1 SP 2 and can used as of this version. However, no restrictions are currently known that would exclude use on a firmware version as of V4.0 (language-dependent expansion and, if required, path object functionality).

2.3 Project

- The axes must be completely parameterized and optimized (position control loop, speed control loop).
- The relevant axes must be computed in the same cycle as the FBLKTransKinTransformation function block.
- The homing procedure for axes requiring homing must be configured and tested.
- The absolute encoder adjustment of axes with absolute encoders must have been performed.
- The release of the controller is not part of the blocks described here and must be implemented by the user.

3 General description of functions

3.1 Terminology

3.1.1 Real machine axes

Machine axes are the real, actually available axes of a handling device. In conjunction with the FBLKTransKinTransformation, only the configuration of the output axes as real axes makes sense.

3.1.2 Path object or path interpolator

The path object provides the functionality for the path interpolation of two or three path axes and for further tasks associated with the path interpolation. It also contains the kinematic transformations implemented in the system. In conjunction with the FBLKTransKinTransformation, a path object can be used to generate the Cartesian position setpoints, whereby the path object is configured as Cartesian gantry. The path object, however, is not a prerequisite for use of the blocks described here.

3.1.3 Input and output axes

The term input axes in conjunction with the FBLKTransKinTransformation is used for the virtual, mostly Cartesian axes, which represent the positions X, Y and Z. Output axes are the real axes, whose motion interface is supplied cyclically with the calculated values of the inverse transformation.

3.1.4 Frame transformation

The term refers here to the mathematical function for alignment of the coordinate system. Rotations and offsets can be implemented. The input coordinate system can be adapted with this before it is converted into axis coordinates by means of the inverse transformation. The coordinate system, which is calculated from the axis coordinates via the transformation, is also adapted in order to correspond to the input coordinate system.

3.2 Library structure

The LKTrans library (**L**ibrary **K**inematik **Trans**formation) is supplied in XML format.
This can be imported into the project by right-clicking the library folder in the project configurator (menu command: Import object).
It is divided into the following (library) units:

Table 1: Library overview of the data structures

Unit	Know-how protection	Meaning
aVersion	Open	Information about changes and the current versions.
cPublic	Open	The unit contains important **constant definitions** (number of input and output values, error numbers).
dProtected	Protected	Enumerators and **type definitions** for internal and external use.
fKinTrans	Protected	Contains the **FBLKTransKinTransformation** user function block, the functions **FCLKTransFrameTrans** and **FCLKTransInvFrametrans** and other necessary auxiliary functions, which however cannot be used externally.
fTrans	Open	Contains the **FBLKTransTransformation** user function block, in which the kinematic transformation is programmed, as well as required auxiliary functions.

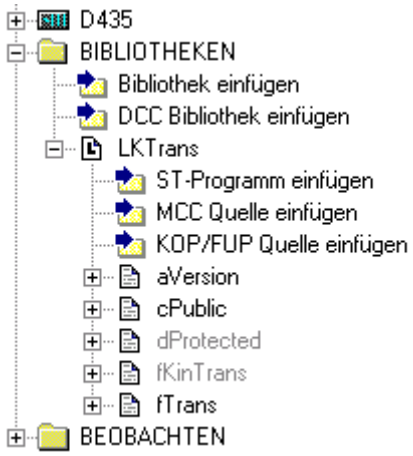


Fig. 3-1: View of the library in SCOUT project navigator

4 Commissioning

The commissioning of the FBLKTRansKinTransformation is explained using the kinematic 4000, a pillar robot, as an example. This kinematic is already contained in the FBLKTransTransformation. The integration of user kinematics is explained in detail in Section 0.

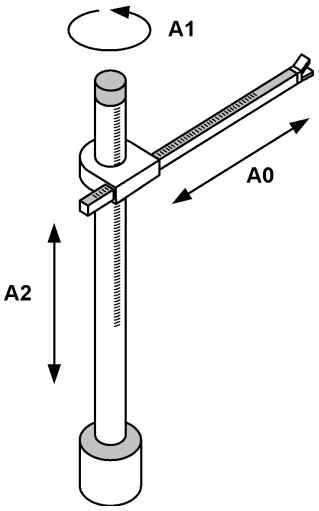


Fig. 4-1: View of kinematic example (transId := 4000)

4.1 Requirements in the project

The required axes have already been created in the project. Fig. 4-2 shows an example of six configured axes. The axes *Axis_X*, *Axis_Y* and *Axis_Z* are virtual linear axes. The axes *AxisOut_0*, *AxisOut_1* and *AxisOut_2* have been configured as real positioning axes. The axes *AxisOut_0* and *AxisOut_2* have been configured as linear axes, the axes *AxisOut_1* as rotary axes with modulo function 0..360°.

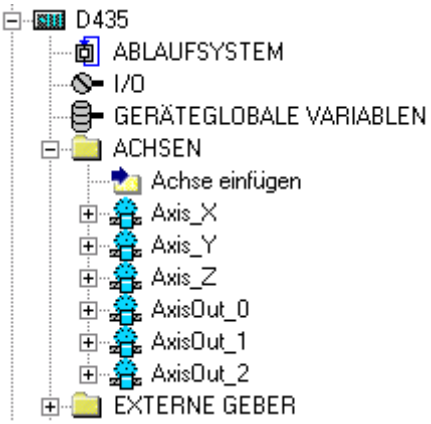


Fig. 4-2: Axes configured in the project navigator

The LKTrans library has been imported by right-clicking the LIBRARIES folder in the project navigator and then compiled on the respective target system.

4.2 Adapting the library constants

In the first step, the parameter structure and the FBLKTransKinTransformation are adapted to the actually used number of input and output axes. Selection of a number that is too high is permitted, but with regard to memory requirement and clarity inappropriate. The constants can be adapted by the user in the *pPublic* library unit. In the example shown, three input axes and three output axes are used.
The processing cycle clock of the function block can also be changed here. This must correspond to the processing cycle clock of the relevant axes. By specifying the processing level, the block reads out the set cycle time and calculates the required velocities and accelerations from this. The block has a ring buffer in which error messages are stored. The number of entries in the buffer can be adapted.

```
//constants for FBLKTransKinTransformation
//-------------------------------------------------------
    //number of used input axes - range: 1..LKTRANS_NUMBER_OF_INPOSITIONS
    LKTRANS_NUMBER_OF_USED_INPUT_AXES         : USINT := 3;

    //number of used output axes - range: 1..LKTRANS_NUMBER_OF_OUTPOSITIONS
    LKTRANS_NUMBER_OF_USED_OUTPUT_AXES        : USINT := 3;

    //execution level of fb and technology object: IPO, IPO_2, SERVO - default IPO
    LKTRANS_EXECUTIONLEVEL                    : EnumToExecutionLevel := IPO;

    //number of items in diagnostic buffer, range: 1..USINT#MAX
    LKTRANS_NUMBER_OF_DIAGBUFFER_ITEMS      : USINT := 20;
```

Note:
The maximum number corresponds to the constants LKTRANS_NUMBER_OF_INPOSITIONS and LKTRANS_NUMBER_OF_OUTPOSITIONS of the FBLKTransTransformation. If required, these must also be increased.

4.3 Call example in ST

A new ST program source file is inserted in the PROGRAMS folder.

4.3.1 Creating the parameter structure

The parameter structure is connected via an input/output parameter to the FBLKTransKinTransformation and contains all of the required settings, limits, setpoints and actual values.
The following code example shows the creation of the parameter structure, an instance variable of the *sLKTransParameterType* type. The structure is described in detail in Section 5.2.1. The name can be freely selected, *gsParameter* has been used here. The connection to the library is established with the *USELIB LKTrans* statement.

```
INTERFACE
//------------- Import -----------------------------------------------------
    USELIB LKTrans; //connection to library
//------------- Device Global Variables ------------------------------------
    VAR_GLOBAL
        //instance of parameter struct
        gsParameter : sLKTransParameterType;
    END_VAR
//--------------------------------------------------------------------------
END_INTERFACE
```

After compilation of the unit, the parameter structure in the SCOUT symbol browser is as shown in Fig. 4-3.

D435.pKinTrans: Symbolbrowser

	Name	Datentyp	Anzeigeformat	Anfangswert
	Alle	Alle	Alle	Alle
1	gsParameter	'sLKTransParameterType'		
2	sGeneral	'sLKTransGeneralType'		
3	sCmdPathTolerance	'sLKTransGeneralCmdPathToleranceType'		
4	sActPathTolerance	'sLKTransGeneralActPathToleranceType'		
5	sActPathFollowingError	'sLKTransGeneralFollowingErrorType'		
6	eMVecCalcMode	'eLKTransMICalculationType'		[0] STANDARD_CALC
7	sTransformation	'sLKTransType'		
8	i32TransID	DINT	DEC	0
9	ar64Parameter	'ARRAY [0..19] OF LREAL'		
10	sCoordTrans	'sLKTransFrameTransType'		
11	asInputAxis	'ARRAY [0..2] OF sLKTransInputAxisType'		
12	asInputAxis[0]	'sLKTransInputAxisType'		
13	asInputAxis[1]	'sLKTransInputAxisType'		
14	asInputAxis[2]	'sLKTransInputAxisType'		
15	asOutputAxis	'ARRAY [0..3] OF sLKTransOutputAxisType'		
16	asOutputAxis[0]	'sLKTransOutputAxisType'		
17	asOutputAxis[1]	'sLKTransOutputAxisType'		
18	asOutputAxis[2]	'sLKTransOutputAxisType'		
19	asOutputAxis[3]	'sLKTransOutputAxisType'		

Fig. 4-3: View of the parameter structure in the symbol browser

4.3.2 Calling the function block

The following code example shows the call of the function block in a user program (with the example name pKinTransIpo). An instance of the block is created under the freely selected name *myFBTrans*. The input parameter *enable* is assigned the global variable *gboKinTransEnable*. This enables the block as well as the calculation of the transformation to be activated. The input parameter *enableMI* is assigned the global variable *gboKinTransEnableMI*. This activates the motion interface of
the axes and then supplies them with cyclic setpoints. The input/output parameter *parameter* is connected to the already created *gsParameter* parameter structure. The output parameters of the FB are not yet assigned in the code section shown.

```
PROGRAM pKinTransIpo
//------------------------------------------------------------------------------
    VAR
        myFBKTrans : FBLKTransKinTransformation; //instance of function block
    END_VAR
    //call instance of function block
    //-------------------------------------------------------
    myFBKTrans(
        enable := gboKinTransEnable
        ,enableMI := gboKinTransEnableMI
        ,parameter := gsParameter
        // ,valid =>
        // ,state =>
        // ,error =>
        // ,errorId =>
        // ;transErrorId =>
        // ,diagnostics =>
        );

        IF myFBKTrans.error THEN
            //error handling
```

```
            ;
        END_IF;
END_PROGRAM
```

4.3.3 Task integration in the execution system

The program must be called in the same processing cycle clock as the relevant axes.
The program must be assigned in the execution system of the *IPOSynchronousTask* if IPO has been configured as processing cycle clock on the axes. The program must be assigned in the execution system of *IPOSynchronousTask_2* if IPO_2 has been configured as processing cycle clock on the axes.

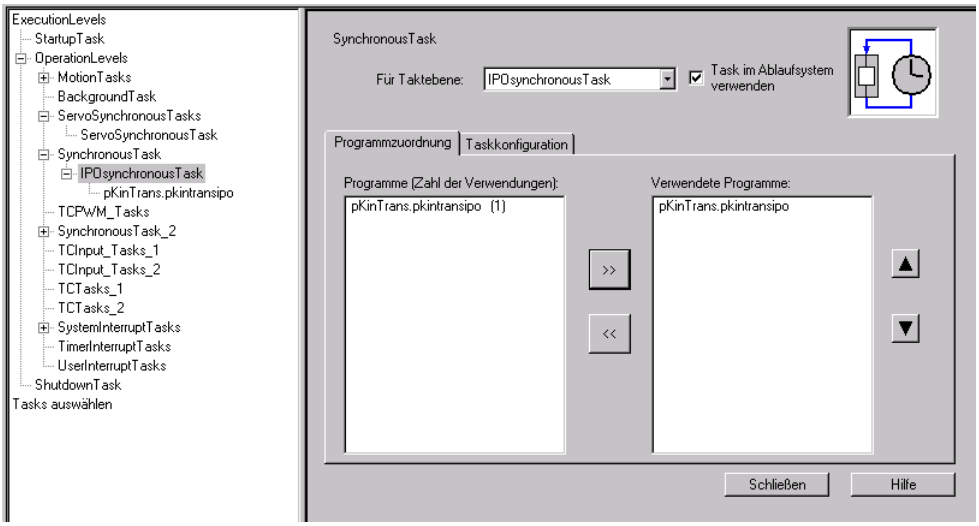


Fig. 4-4: Call of the program in the execution system

4.3.4 Value assignment of the parameter structure

The parameter structure contains all relevant settings of the FBLKTransKinTransformation.
The required values must be set here for the configuration. The user has various options in SIMOTION:

- Initialization program in the StartupTask
- Import of the values from the CF card
- Creating the parameter structure as retain variable, online change of values
- Initialization in the first execution cycle of a cyclic task

The assignment of the values in the first processing cycle is shown here as an example.

```
VAR
    boFirstCycle : BOOL := TRUE; //first cycle in mode run
END_VAR

//initialisation of variables in first cycle
IF boFirstCycle THEN
    //general parameter-----------------------------------------------------
    //the motionIn will activated direct under this limit , 0.0 = inactive
    gsParameter.sGeneral.sActPathTolerance.r64ActivateMIDirectLimit := 0.1; //mm
    //the (output-) axes will pulled to setpoint under this limit
    //before activating the MI, 0.0 = inactive
    gsParameter.sGeneral.sActPathTolerance.r64ActivateMIPullAxesLimit := 20.0; //mm
    //an active motionIn will deactivated when reach this limit, 0.0 = inactive
    gsParameter.sGeneral.sActPathFollowingError.r64Limit := 10.0; //mm
    //an active motionIn  will deactivated when reach this LIMIT, 0.0 = inactive
    gsParameter.sGeneral.sCmdPathTolerance.r64Limit := 1.0 //mm
```

```
    //calculation mode for motion vector
    gsParameter.sGeneral.eMVecCalcMode := STANDARD_CALC;
   //transformation parameter-------------------------------------------
    //tranformation id - example 4000
    gsParameter.sTransformation.i32TransID := 4000;
    //mm, minimum length OF actualOutPosition[0]
    gsParameter.sTransformation.ar64Parameter[0] := 10.0;
    //mm, maximum length of actualOutPosition[0]
    gsParameter.sTransformation.ar64Parameter[1] := 2000.0;
    //mm, shiftet axis: distance d between axis Z and actualOutPosition[0]
    gsParameter.sTransformation.ar64Parameter[2] := 0.0;
    //angle in deg for commandOutPosition[1] when X=0.0 and Y=0.0
    gsParameter.sTransformation.ar64Parameter[4] := 0.0;
    //input axes---------------------------------------------------------
    gsParameter.asInputaxis[0].toAx := Axis_X; //reference
    gsParameter.asInputaxis[1].toAx := Axis_Y; //reference
    gsParameter.asInputaxis[2].toAx := Axis_Z; //reference
    //output axes--------------------------------------------------------
    //axis A0 – liner axis:
    gsParameter.asOutputaxis[0].toAx := AxisOut_0; //reference
    //limit for commandvalue tolerance
    gsParameter.asOutputaxis[0].sCommandValueTolerance.r64Limit := 1.0; //mm
    //dynamic limits to open motion in (e.g. axes limits)
    gsParameter.asOutputaxis[0].sMIDynamics.r64Velocity := 1000.0; //mm/s
    gsParameter.asOutputaxis[0].sMIDynamics.r64Acceleration := 10000.0; //mm/s²
    gsParameter.asOutputaxis[0].sMIDynamics.r64Deceleration := 10000.0; //mm/s²
    //dynamic limits for Stop, e.g. in case of error
    gsParameter.asOutputaxis[0].sStopDynamics.r64Deceleration := 5000.0; //mm/s²
    gsParameter.asOutputaxis[0].sStopDynamics.r64Jerk := 10000.0; //mm/s³
    //dynamic limits for Stop, e.g. in case of error
    gsParameter.asOutputaxis[0].sPullDynamics.r64Velocity := 100.0; //mm/s
    gsParameter.asOutputaxis[0].sPullDynamics.r64Acceleration := 1000.0; //mm/s²
    gsParameter.asOutputaxis[0].sPullDynamics.r64Deceleration := 1000.0; //mm/s²
    gsParameter.asOutputaxis[0].sPullDynamics.r64Jerk := 10000.0; //mm/s³
    //axis A1 – rotary axis:
    gsParameter.asOutputaxis[1].toAx := AxisOut_1; //reference
    gsParameter.asOutputaxis[1].sCommandValueTolerance.r64Limit := 1.0; //°
    gsParameter.asOutputaxis[1].sMIDynamics.r64Velocity := 1000.0; //°/s
    gsParameter.asOutputaxis[1].sMIDynamics.r64Acceleration := 10000.0; //°/s²
    gsParameter.asOutputaxis[1].sMIDynamics.r64Deceleration := 10000.0; //°/s²
    gsParameter.asOutputaxis[1].sStopDynamics.r64Deceleration := 5000.0; //°/s²
    gsParameter.asOutputaxis[1].sStopDynamics.r64Jerk := 10000.0; //°/s³
    gsParameter.asOutputaxis[1].sPullDynamics.r64Velocity := 100.0; //°/s
    gsParameter.asOutputaxis[1].sPullDynamics.r64Acceleration := 1000.0; //°/s²
    gsParameter.asOutputaxis[1].sPullDynamics.r64Deceleration := 1000.0; //°/s²
    gsParameter.asOutputaxis[1].sPullDynamics.r64Jerk := 10000.0; //°/s³
    //axis A2 – linear axis:
    gsParameter.asOutputaxis[2].toAx := AxisOut_2; //reference
    gsParameter.asOutputaxis[2].sCommandValueTolerance.r64Limit := 1.0; //mm
    gsParameter.asOutputaxis[2].sMIDynamics.r64Velocity := 1000.0; //mm/s
    gsParameter.asOutputaxis[2].sMIDynamics.r64Acceleration := 10000.0; //mm/s²
    gsParameter.asOutputaxis[2].sMIDynamics.r64Deceleration := 10000.0; //mm/s²
    gsParameter.asOutputaxis[2].sStopDynamics.r64Deceleration := 5000.0; //mm/s²
    gsParameter.asOutputaxis[2].sStopDynamics.r64Jerk := 10000.0; //mm/s³
    gsParameter.asOutputaxis[2].sPullDynamics.r64Velocity := 100.0; //mm/s
    gsParameter.asOutputaxis[2].sPullDynamics.r64Acceleration := 1000.0; //mm/s²
    gsParameter.asOutputaxis[2].sPullDynamics.r64Deceleration := 1000.0; //mm/s²
    gsParameter.asOutputaxis[2].sPullDynamics.r64Jerk := 10000.0; //mm/s³
    //-------------------------------------------------------------------
    boFirstCycle := FALSE;
END_IF;
```

4.4 Deactivating system errors

The external transformation uses the motion interface on the TO axis to transfer the setpoints of the transformation to the axis. In addition to the position (setpoint), the axis also requires the velocity and acceleration. As the transformation equation only supplies the positions, the velocity and acceleration have to be calculated.
Since this calculation is inaccurate for performance reasons, the TO axis may output the following errors:

40002 The system is limiting the programmed velocity to the maximum permissible velocity.

40020 Dynamic response of the setpoints on the motion interface (type: 2) cannot be maintained (reason: 1) or (reason: 2)

Normally the technology object does not respond to these errors since the pre-assignment of the local response is set to NONE (= no response by the axis - only Information).
It is recommended that messages 40002 and 40020 be deactivated for all output axes of the transformation and the monitoring functions of the function block (Section 6.2.8) be used.

The deactivation of alarms in SIMOTION SCOUT is performed in the alarm configuration.
This can be accessed in the TechnologicalFaultTask in the execution system (see Fig. 4-5).

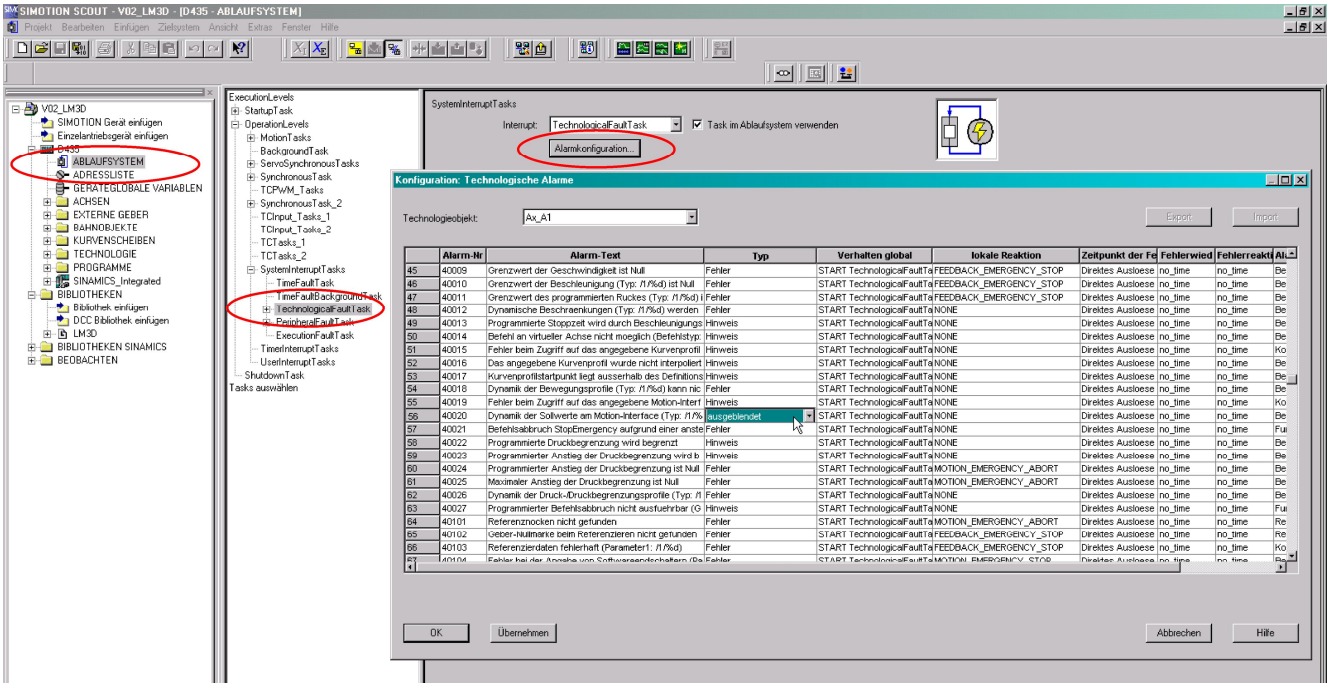


Fig. 4-5: Change of the alarm response in SIMOTION SCOUT

5 Constants and data structures

5.1 Global constants

The quantity structure of the function blocks can be set with the following constants (number of inputs and outputs, number of axes, etc.). These constants are defined in the **cPublic** (library) unit. This unit also contains constants that can be used as error numbers. These are listed
and described in Section 0.

Name	Data type	Meaning
LKTRANS_NUMBER_OF_USED_INPUT_AXES	USINT	Number of input axes on the FBLKTransKinTransformation block. Range of values: 1.. LKTRANS_NUMBER_OF_INPOSITIONS, default 3 (e.g. for X, Y, Z)
LKTRANS_NUMBER_OF_USED_OUTPUT_AXES	USINT	Number of output axes on the FBLKTransKinTransformation block. Range of values: 1.. LKTRANS_NUMBER_OF_OUTPOSITIONS , default 4
LKTRANS_EXECUTIONLEVEL	ENUM	Processing cycle clock of the FB and the relevant axes. Type: *EnumToExecutionLevel* (SIMOTION system enum): IPO, IPO_2 or SERVO
LKTRANS_NUMBER_OF_DIAGBUFFER_ITEMS	USINT	Number of diagnostic buffer entries in the error buffer. Range of values: 1..USINT#MAX, default 20
LKTRANS_NUMBER_OF_INPOSITIONS	USINT	Number of (input) positions on the FBLKTransTransformation block. Range of values: 1..USINT#MAX, default 4
LKTRANS_NUMBER_OF_OUTPOSITIONS	USINT	Number of (output) positions on the FBLKTransTransformation block. Range of values: 1..USINT#MAX, default 10
LKTRANS_NUMBER_OF_TRANS_PARAMETER	USINT	Number transformation parameters on the FBLKTransTransformation block. Range of values: 1..USINT#MAX, default 20

Explanation:
The constants LKTRANS_NUMBER_OF_INPOSITIONS and LKTRANS_NUMBER_OF_OUTPOSITIONS are the number of input and output coordinates of the mathematical block FBLKTransTransformation. They can be greater than the number of axes actually used in the project, which are determined by the constants LKTRANS_NUMBER_OF_USED_INPUT_AXES and LKTRANS_NUMBER_OF_USED_OUTPUT_AXES.

5.2 Technological parameters

5.2.1 Parameter structure

This is the central data interface of the FBLKTransKinTransformation. This has an input/output parameter of this type. The structure is divided into several areas.

sLKTransParameterType – technological parameters

Name	Type	Meaning
sGeneral	sLKTransGeneralType	General parameters (Section 5.2.2)
sTransformation	sLKTransType	Transformation parameters (Section 5.2.3)
asInputAxis	ARRAY[0..n -1] OF sLKTransInputAxisType	Parameters of the (Cartesian) input axes (Section 5.2.4)
asOutputAxis	ARRAY[0..m -1] OF sLKTransOutputAxisType	Parameters of the real output axes (Section 5.2.5)

Note: The variable n is determined by the constant LKTRANS_NUMBER_OF_USED_INPUT_AXES.
The variable m is determined by the constant LKTRANS_NUMBER_OF_USED_OUTPUT_AXES.

5.2.2 General parameters (sLKTransGeneralType)

This area contains general parameters, such as the current path deviation, as well as relevant limits.

sLKTransGeneralType – general parameters

Name	Type	Meaning
sCmdPathTolerance	sLKTransGeneralCmdPathToleranceType	Parameters of the setpoint deviation (Section)
sActPathTolerance	sLKTransGeneralActPathToleranceType	Parameters of the actual value deviation (Section)
sActPathFollowingError	sLKTransGeneralFollowingErrorType	Parameters of the Cartesian following error (Section)
eMVecCalcMode	eLKTransMICalculationType	Calculation type of the motion vector (STANDARD_CALC, QUADRATIC_CALC)

5.2.2.1 Parameters of the setpoint deviation (sLKTransGeneralCmdPathToleranceType)

sLKTransGeneralCmdPathToleranceType – parameters of the setpoint deviation

Name	Type	Meaning
r64ActualValue	LREAL	Current deviation if monitoring is active
r64Limit	LREAL	Limit value, 0.0 = monitoring is deactivated

5.2.2.2 Parameters of the actual value deviation (sLKTransGeneralActPathTolerance Type)

sLKTransGeneralActPathToleranceType – parameters of the actual value deviation

Name	Type	Meaning
r64ActualValue	LREAL	Current deviation if monitoring is active
r64ActivateMIDirectLimit	LREAL	Limit value, 0.0 = monitoring is deactivated. Below the limit, coupling is performed directly
r64ActivateMIPullAxesLimit	LREAL	Limit value, 0.0 = monitoring is deactivated. Below the limit, the (output) axes are positioned at the setpoint

5.2.2.3 Parameters of the following error (sLKTransGeneralFollowingError**Type)**

sLKTransGeneralFollowingErrorType – parameters of the following error

Name	Type	Meaning
r64ActualValue	LREAL	Current deviation if monitoring is active
r64Limit	LREAL	Limit value, 0.0 = monitoring is deactivated

5.2.3 Transformation parameters (sLKTransTransformationType)

This area contains both the transformation number as well as all transformation parameters. The parameters are for specifying mechanical dimensions, offset or link positions. Current values determined by the transformation (e.g. link positions) can also be displayed.

sLKTransTransformationType – transformation parameters

Name	Type	Meaning
i32TransId	DINT	Transformation number
ar64Parameter	ARRAY[0..n] OF LREAL	Transformation parameters
sCoordTrans	sLKTransFrameTransType	Frame transformation parameters (Section 5.2.3.1)

Note: n is determined by the constant LKTRANS_NUMBER_OF_TRANS_PARAMETER.

5.2.3.1 Parameters of the frame transformation parameters (sLKTransType)

sLKTransFrameTransType – frame transformation parameters

Name	Type	Meaning
ar64Translation	ARRAY[0..2] OF LREAL	Coordinate offset
ar64Rotation	ARRAY[0..2] OF LREAL	Coordinate rotation

5.2.4 Parameters of the input axes (sLKTransInputAxisType)

This area contains all parameters of the input axes.

sLKTransInputAxisType – parameters of the actual value deviation

Name	Type	Meaning
r64CommandPosition	LREAL	Current setpoint of the axis
r64ActualPosition	LREAL	Current actual value from the transformation
toAx	Posaxis	Reference of the axis, TO#NIL=inactive

Name	Type	Meaning
i32State	DWORD	Status word of the axis
boGetValWithoutAx	BOOL	The current setpoint is also used even without an assigned axis reference

5.2.5 Parameters of the output axes (sLKTransOutputAxisType)

This area contains all parameters of the output axes.

sLKTransOutputAxisType – parameters of the actual value deviation

Name	Type	Meaning
r64MotionInPosition	LREAL	Current setpoint from the transformation
r64CommandPosition	LREAL	Current setpoint of the axis
r64ActualPosition	LREAL	Current actual value from the transformation
toAx	Posaxis	Reference of the axis
i32State	DWORD	Status word of the axis
sCommandValueTolerance	sLKTransOutputAxCmdValueToleranceType	Parameters of the setpoint monitoring (Section 5.2.5.1)
sMIDynamics	sLKTransOutputAxMIDynamikType	Dynamic response limits of the motion interface MI (Section5.2.5.2)
sStopDynamics	sLKTransOutputAxStopDynamikType	Dynamic response limits at stop, e.g. in the event of an error (Section5.2.5.3)
sPullDynamics	sLKTransOutputAxPullDynamikType	Dynamic response limits for pulling the axes during activation (Section 5.2.5.4)

5.2.5.1 Parameters of the setpoint monitoring (sLKTransOutputAxCmdValueToleranceType)

sLKTransOutputAxCmdValueToleranceType – parameters of the setpoint monitoring

Name	Type	Meaning
r64ActualValue	LREAL	Current deviation if monitoring is active
r64Limit	LREAL	Limit value, 0.0 = monitoring is deactivated

5.2.5.2 Dynamic response limits of the motion Interface (sLKTransOutputAxMIDynamikType)

sLKTransOutputAxMIDynamikType – parameters of the setpoint monitoring

Name	Type	Meaning
r64Velocity	LREAL	Maximum velocity
r64Acceleration	LREAL	Maximum acceleration
r64Deceleration	LREAL	Maximum deceleration

5.2.5.3 Dynamic response limits at stop (sLKTransOutputAxStopDynamikType)

sLKTransOutputAxStopDynamikType – dynamic response limits at stop

Name	Type	Meaning
r64Deceleration	LREAL	Maximum deceleration

Name	Type	Meaning
r64Jerk	LREAL	Jerk specification

5.2.5.4 Dynamic response limits when pulling the axes (sLKTransOutputAxPullDynamikType)

sLKTransOutputAxPullDynamikType – dynamic response limits when pulling the axes

Name	Type	Meaning
r64Velocity	LREAL	Maximum velocity
r64Acceleration	LREAL	Maximum acceleration
r64Deceleration	LREAL	Maximum deceleration
r64Jerk	LREAL	Jerk specification

5.2.6 Diagnostics structure (sLKTransDiagnosticsType)

Detailed information about errors that have occurred are saved in the diagnostics structure. This helps the user to find the cause of the error. See also Section 8.

sLKTransOutputAxisType – parameters of the diagnostics structure

Name	Type	Meaning
u8AxNumber	USINT	Relevant (axis) index
i32Detail1	DINT	Detail information 1
b32Detail2	DWORD	Detail information 2
b32Detail3	LREAL	Detail information 3
i32BufferIndex	DINT	Index of the current (last) error message in the error buffer, -1=no entered error
asBuffer	ARRAY[0..n-1] OF sLKTransDiagnosticItemType	Error buffer, implemented as ring buffer (Section 5.2.6.1)

Note: n is determined by the constant LKTRANS_NUMBER_OF_DIAGBUFFER_ITEMS.

5.2.6.1 Parameters of a diagnostic buffer entry (sLKTransDiagnosticItemType)

sLKTransDiagnosticItemType – parameters of the diagnostic buffer entry

Name	Type	Meaning
dtPointOfTime	DATE_AND_TIME	Time when the error occurred
b32ErrorId	DWORD	Error number
u8AxNumber	USINT	Relevant (axis) index
i32Detail1	DINT	Detail information 1
b32Detail2	DWORD	Detail information 2
b32Detail3	LREAL	Detail information 3

5.2.7 Status words of the axes

The current status of the axis in the form of individual bits can be read out on the status word ***b32state*** within the axis parameters (Sections 5.2.4 and 5.2.5). A status word is available for each implemented axis.

LowWord -> dynamic states	HighWord -> static
Bit 0: Emergency stop inactive	Bit 16: Synchronous operation instance created
Bit 1: No technological axis error	Bit 17: Modulo axis created
Bit 2: Axis in closed-loop control	Bit 18: Path object created
Bit 3: Axis homed	Bit 19: Path object connected
Bit 4: Velocity setpoint = 0	Bit 20: Path motion active
Bit 5: Synchronization status	Bit 21: Synchronization of a path object to a master value is active
Bit 6: Active gearing	
Bit 7: Active camming	

For performance reasons, only the status bits required by the FB are updated on the axis.
For the input axes, this is bit 4 and for the output axes these are bits 0, 1, 2, 3 and 4.

5.2.8 Enumerators

5.2.8.1 Block state (eLKTransStateType)

The output parameter *state* displays the current state of the motion interface on the FBLKTransKinTransformation block. The enum of type eLKTransStateType is used for this purpose:

Identifier	Value	Meaning
NONE	0	Initial value after loading, check of elementary constants.
INACTIVE	10	Idle state, the motion interface is not active.
REDEFINE_AXES	20	Preparation for the activation of the motion interface: The input axes are set to the current position values of the transformation.
PULL_AXES	25	Preparation for the activation of the motion interface: The output axes are pulled to the setpoint of the inverse transformation using positioning.
ACTIVATING_MI	30	The motion interface is currently being activated on all output axes.
ACTIVE	50	The motion interface has been activated on all output axes and is being supplied with cyclic position values of the inverse transformation.
DEACTIVATING_MI	60	The motion interface is currently being deactivated on all output axes. This can be due to the input parameters or as a response to an error.
STOP_PULL_AXES	65	The PULL_AXES state has been aborted by the user or due to an error. Positioning of the axes is now stopped.

5.2.8.2 Transformation calculation type (eLKTransCalcModeType)

This input parameter on the FBLKTransTransformation determines what is to be calculated by the block: The transformation (from ActualOutPosition to ActualInPosition), the inverse transformation (from CommandInPosition to CommandOutPosition) or both directions.

Identifier	Value	Meaning
BOTH	0	Both directions are calculated.
TRANS	1	Only the transformation is calculated.
INV_TRANS	2	Only the inverse transformation is calculated.

5.2.8.3 Motion vector calculation type (eLKTransMICalculationType)

Within the FBLKTransKinTransformation, the complete motion vector (position, velocity, acceleration) is calculated from the position setpoint of the axis which comes from the inverse transformation. The calculation method can be set in the parameter structure. The enum of type eLKTransStateType is used for this purpose:

Identifier	Value	Meaning
STANDARD_CALC	0	It is derived numerically from the current position and the last motion vector.
QUADRATIC_CALC	2	Determination of a parabola from the current and the last two positions. The motion vector is read out from this.

6 Description of the function blocks

6.1 FBLKTransTransformation function block

6.1.1 Task

The FBLKTRansTransformation function block provides uniform interfaces for the conversion of position values. In the handling/robotics field, the Cartesian position values X, Y, Z can be converted to the (machine) axis positions of the used kinematics. If both the transformation and inverse transformation are implemented, the block calculates the geometric distance between the specified Cartesian position setpoint and the transformed Cartesian actual position.
The user can thus develop specific kinematics that are not supported by the TO path interpolation. Several kinematics have already been implemented as examples. The number of input and output positions and the number of transformation parameters can be set with library constants. The FB can also be used in the user program as information function.

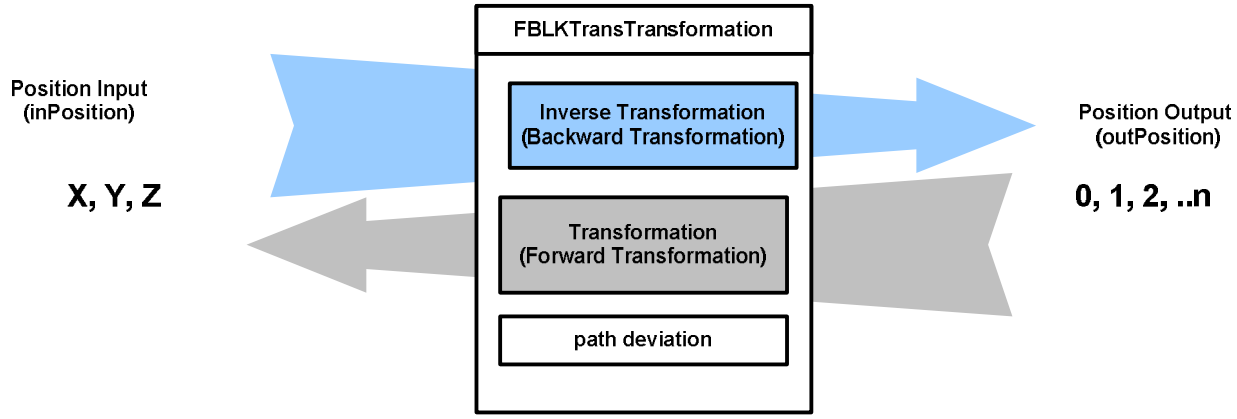


Fig. 6-1: Task of the FBLKTransTransformation

6.1.2 Task integration

In principle, integration is possible in all tasks. However, if setpoints are generated with the block, typically the call is in a cyclic, if required, synchronous task.

6.1.3 Schematic LAD/FBD diagram

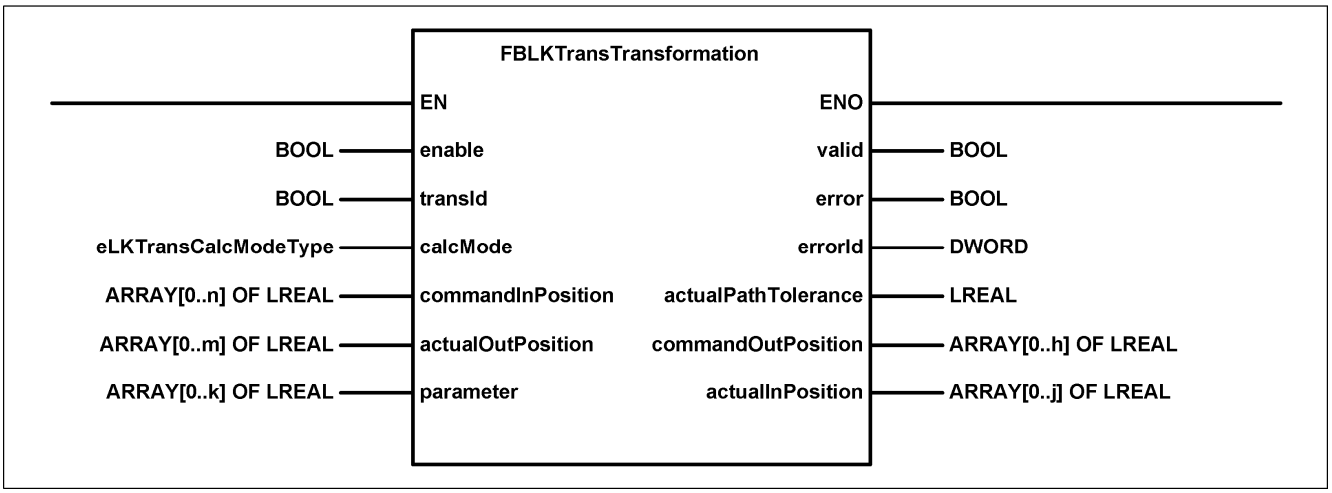


Fig. 6-2: LAD view of the FBLKTransTransformation

6.1.4 Parameters of the blocks

Name	P type	Data type	Meaning
enable	IN	BOOL	Decides on the execution of the block
transId	IN	DINT	Technology structure (see Section 5.2.1)
calcMode	IN	eLKTransCalcModeType	Calculation type / direction: TRANS, INV_TRANS, BOTH
commandInPosition	IN	ARRAY[0..n] OF LREAL	(Setpoint) positions before inverse transformation
actualOutPosition	IN	ARRAY[0..m] OF LREAL	(Actual) positions before transformation
parameter	IN_OUT	ARRAY[0..k] OF LREAL	Transformation parameters
valid	OUT	BOOL	Output values are valid
error	OUT	BOOL	Pending error
errorId	OUT	DWORD	Error number
actualPathTolerance	OUT	LREAL	Current geometric deviation from commandInPosition[0,1,2] and actualInPosition[0,1,2]
commandOutPosition	OUT	ARRAY[0..h] OF LREAL	(Setpoint) positions after inverse transformation
actualInPosition	OUT	ARRAY[0..j] OF LREAL	(Actual) positions after transformation

Notes:
n is determined by the constant LKTRANS_NUMBER_OF_INPOSITIONS
m is determined by the constant LKTRANS_NUMBER_OF_OUTPOSITIONS
k is determined by the constant LKTRANS_NUMBER_OF_TRANS_PARAMETER
h is determined by the constant LKTRANS_NUMBER_OF_OUTPOSITIONS
j is determined by the constant LKTRANS_NUMBER_OF_INPOSITIONS

6.1.5 Description of functions

The processing of the function block is activated by setting the *enable* input parameter. This is indicated by the set output parameter *valid*. The transformation equation is now calculated and can be checked.
The number of the transformation can be selected with the *transId* input parameter. The calculation type can be specified with the *calcMode* input parameter: With the selection TRANS, only the transformation is calculated, with INV_TRANS only the inverse transformation is calculated. With the selection BOTH, both directions and the path deviation are calculated.
The *commandInPosition* input parameter consists of an array of position values, which are processed by the inverse transformation. The results are also available in an array of position values at the *commandOutPosition* output parameter.
The *actualOutPosition* input parameter consists of an array of position values, which are processed by the transformation. The results are also available in an array of position values at the *actualInPosition* output parameter.
The first three array indices ([0], [1] and [2]) of the parameters *commandInPosition* and *actualInPosition* are interpreted as Cartesian coordinates X, Y, Z. The difference between the parameters is calculated (as absolute value) and provided at the *actualPathTolerance* output parameter.
If an error occurs (e.g. invalid transformation number) the *error* output parameter is set and an error number output at the *errorId* parameter. Different errors (e.g. undefined value ranges) may occur depending on the transformation equation.

Important note:
It is the responsibility of the creator of a transformation equation to catch and report all possible errors. If arithmetic errors, such as division by zero are not caught, this can result in a processing error in the controller, which then switches to the STOP mode. Machine damage can occur depending on the configured response of the drives.

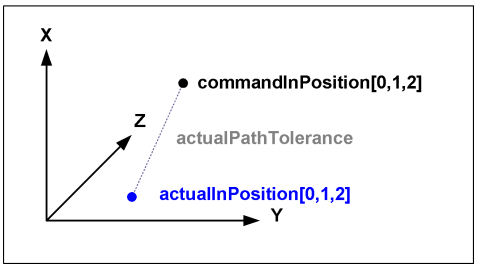


Fig. 6-3: Path deviation

6.2 FBLKTransKinTransformation function block

6.2.1 Task

The FBLKTransKinTransformation function block can be used to configure a kinematic transformation between (Cartesian) input axes and (real) output axes.
The FBLKTransTransformation (Section 6.1) is used for the calculation and called cyclically.
The function block reads in the setpoints of the configured input axes which correspond to the Cartesian coordinates X, Y and Z. Setpoints are calculated from these setpoints using the inverse transformation and when the block is active (enable := TRUE; enableMI := TRUE), written to the motion interface of the real machine axes. The actual values of these output axes are transformed by the block and are available as information and for the monitoring.

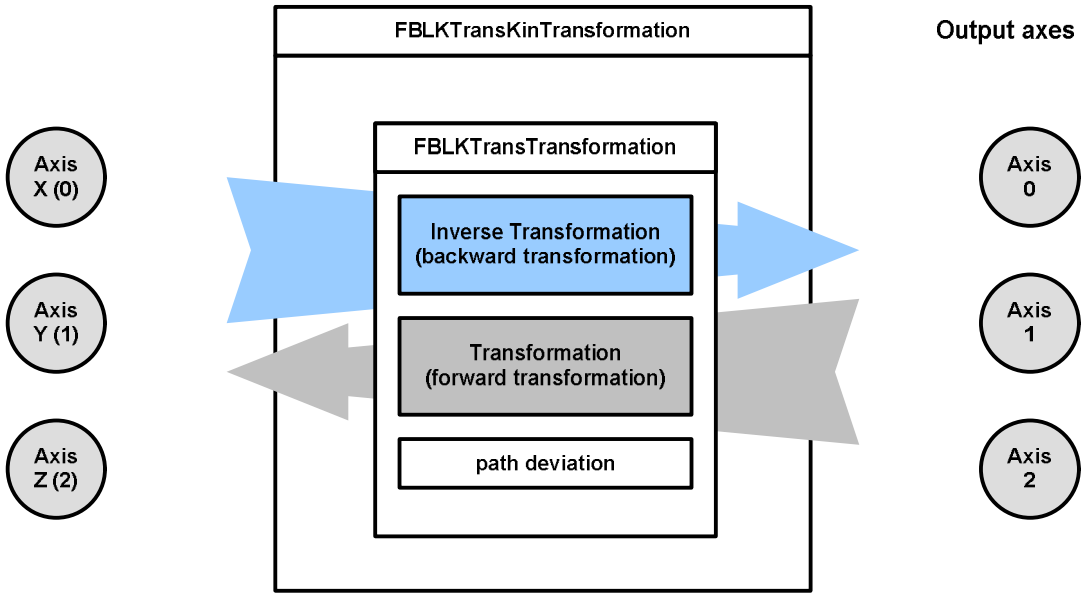


Fig. 6-4: Use of the FBKintranformation

6.2.2 Task integration

The block must be called in the same processing cycle clock as the relevant axes.
The program must be assigned in the execution system of the *IPOSynchronousTask* if IPO has been configured as processing cycle clock on the axes. The program must be assigned in the execution system of *IPOSynchronousTask_2* if IPO_2 has been configured as processing cycle clock on the axes. Calling in the ServoSynchronousTask is not advisable for performance reasons and is not recommended.
The FB must also be informed in which task it will be called by means of the constant *LKTRANS_EXECUTIONLEVEL* (see Section 5.1).

6.2.3 Schematic LAD/FBD diagram

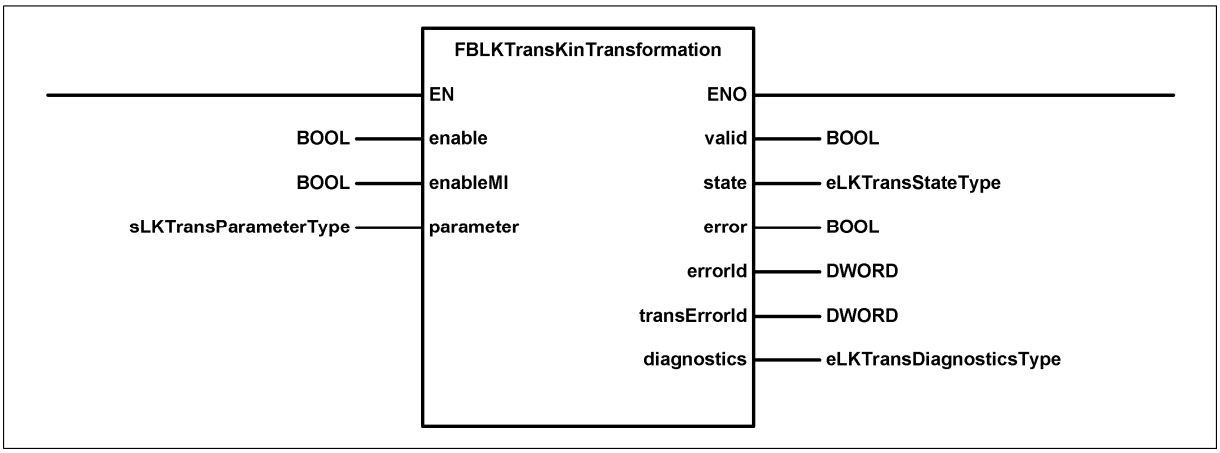


Fig. 6-5: LAD view of the FBLKTransKinTransformation

6.2.4 Parameters of the blocks

Name	P type	Data type	Meaning
enable	IN	BOOL	Activates the processing and transformation calculation
enableMI	IN	BOOL	Activates the motion interface and supplies this with cyclic setpoints
parameter	IN_OUT	sLKTransParameterType	Parameter structure (see Section 5.2.1)
valid	OUT	BOOL	Valid block state
state	OUT	eLKTransStateType	State of the motion interface (see Section 5.2.8.1)
error	OUT	BOOL	Pending block error
errorId	OUT	DWORD	Error number (see Section 8)
transErrorId	OUT	DWORD	Error number of the transformation
diagnostics	OUT	sLKTransDiagnosticsType	Diagnostics structure (see Section 5.3.3)

Note: The *transErrorId* output parameter supplies the pending error number of the contained FBLKTransTransformation. However, this is only considered as an error in the activated state of the motion interface.
In this way, error-free operation of the transformation can be checked during the commissioning phase with inactive motion iInterface.

6.2.5 FBLKTransKinTransformation flow diagram

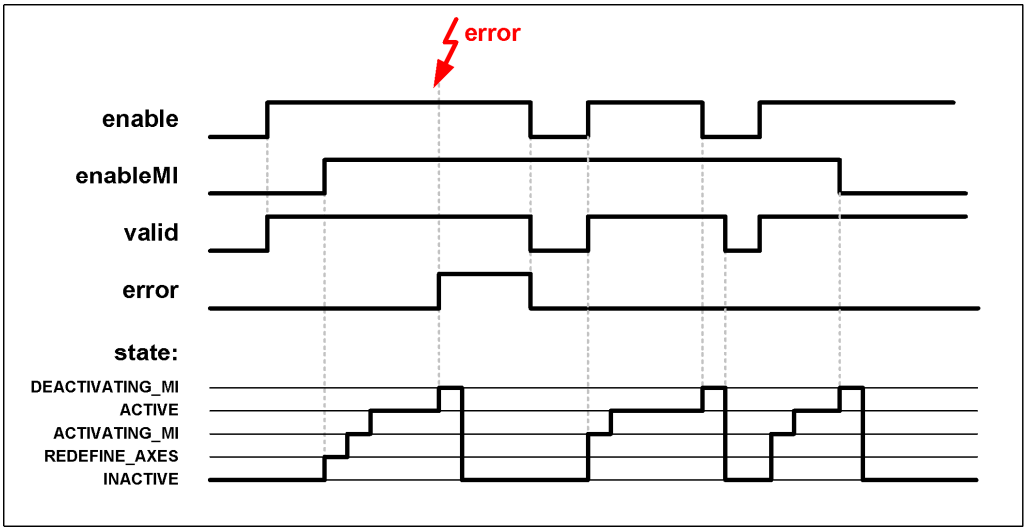


Fig. 6-6: FBLKTRansKinTransformation flow diagram

If an error occurs during processing, the motion interface is deactivated. The *enable* input parameter must be reset to acknowledge a pending error.

6.2.6 FBLKTransKinTransformation state model

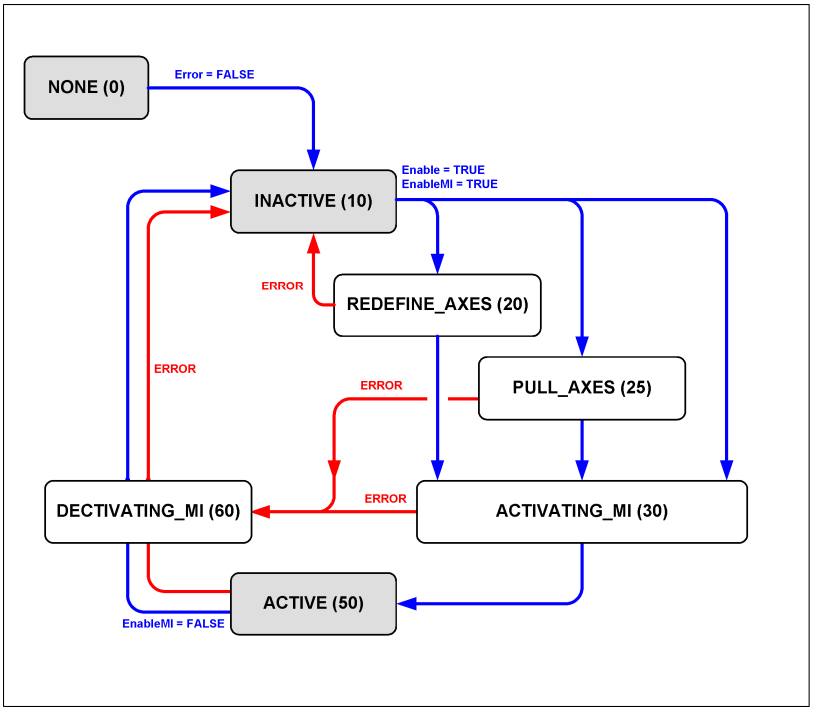


Fig. 6-7: FBLKTransKinTransformation state model

The current state of the FBLKTRansKinTransformation can be read out at the *state* output parameter. This is an enum of the type eLKTransStateType (Section 5.2.8.1).
After the first call, the block changes from the initial state NONE to INACTIVE, when no error is detected in the constant values.

6.2.7 Activation of the motion interface

The calculation of the transformation as well as the cyclic reading of axis values are already active when the *enable* input parameter is set.
To activate the motion interface the *enableMI* input parameter must also be set.
When all requirements have been satisfied, the function block starts the activation, otherwise an error is output.
The following requirements must be satisfied:

- The output axes have been homed
- Neither a fatal error nor an emergency stop is active on the output axes
- The input and output axes are stationary
- The controllers of the output axes have been enabled

During activation, the block now checks whether the motion interface can be directly activated. This should only be performed when there is just a slight deviation between the actual position and the position setpoint.
Parameter *sGeneral.sActPathTolerance.r64ActivateMIDirectLimit* in the parameter structure specifies the maximum permissible path deviation (distance between the Cartesian position setpoint and the Cartesian actual position). If 0.0 is specified as the limit, there is no direct activation.
If the limit is undershot, the motion interface is activated, resulting compensating movements are performed with the *.asOutputAxis[..].sMIDynamics* dynamic responses of the parameter structure.
If the limit is exceeded or the limit is deactivated, a check is made as to whether the output axes can be pulled to the setpoint using positioning.
Parameters*General.sActPathTolerance.r64ActivateMIPullAxesLimit* of the parameter structure specifies the maximum permissible path deviation during activation. If 0.0 is specified as the limit, the limit is deactivated. If the limit is exceeded, the axes are positioned at the setpoint with the *.asOutputAxis[..].sPullDynamics* dynamic responses of the parameter structure.

If both limits have been exceeded, or they are deactivated, the input axes are set to the real positions (_redefinePosition system function).

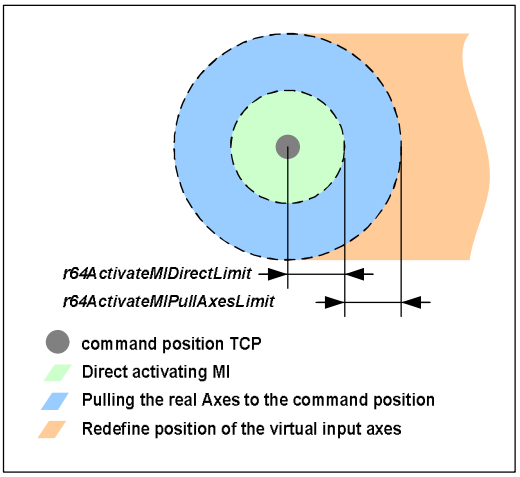


Fig. 6-8: Limits during activation of the motion interface

6.2.7.1 Example

If the input axes are virtual axes, they display the position 0.0 after ramp-up. However, the real output axes are at a real position. Fig. 6-9 illustrates this state (for simplification, there is no conversion of the positions in the transformation).

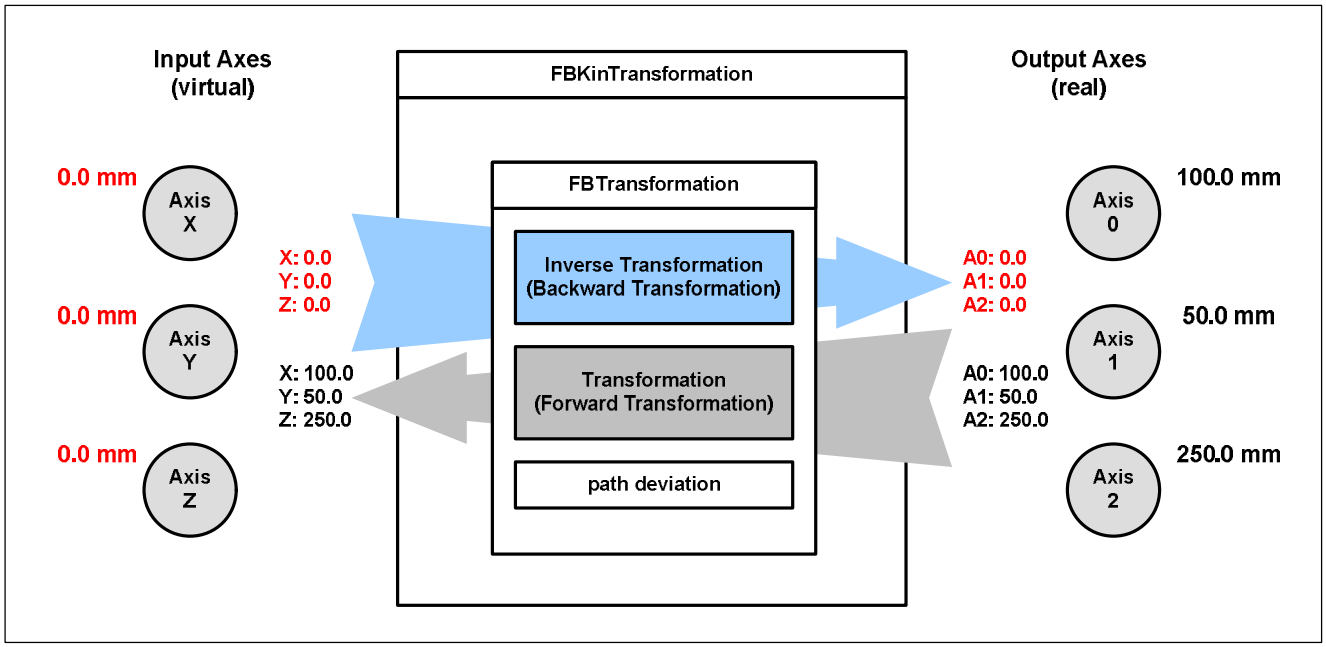


Fig. 6-9: Typical initial state after ramp-up

If the motion interface was now activated directly, the real output axes would compensate this setpoint jump by an uncoordinated movement with maximum dynamic response.
If the two adjustable limits are exceeded in this state, the positions of the input axes are set to the (real) transformed positions. The inverse transformation now supplies suitable values for the real axes. This state is shown in Fig. 6-10.
The motion interface can now be activated without any compensating movement.

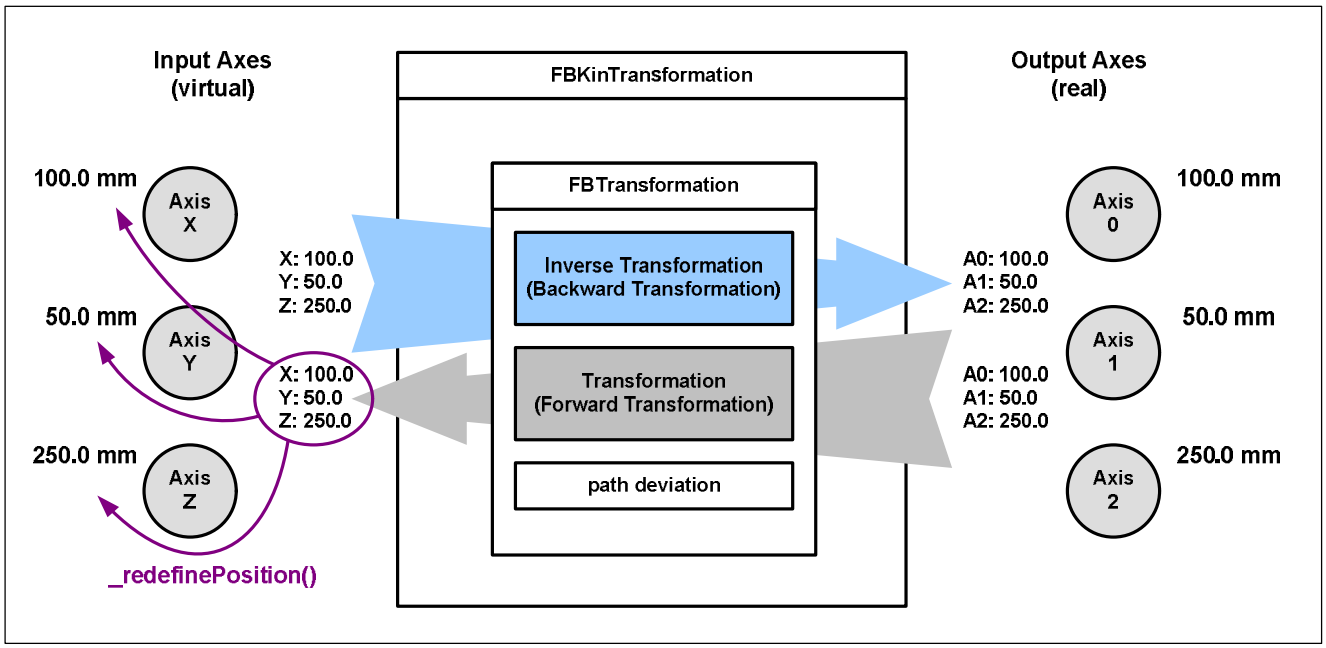


Fig. 6-10: Setting of the input axes positions

If no error occurs, the FB switches to the ACTIVE state and now monitors the motion interface.
The path setpoint monitoring, path following error monitoring and axis setpoint monitoring are now active.
In this state, motion of the input axes can be started, e.g. with a path object.

6.2.8 Monitoring functions

The FBLKTransKinTransformation() function block monitors the correct operation. For this purpose, different characteristic values are calculated and monitored with limit values.

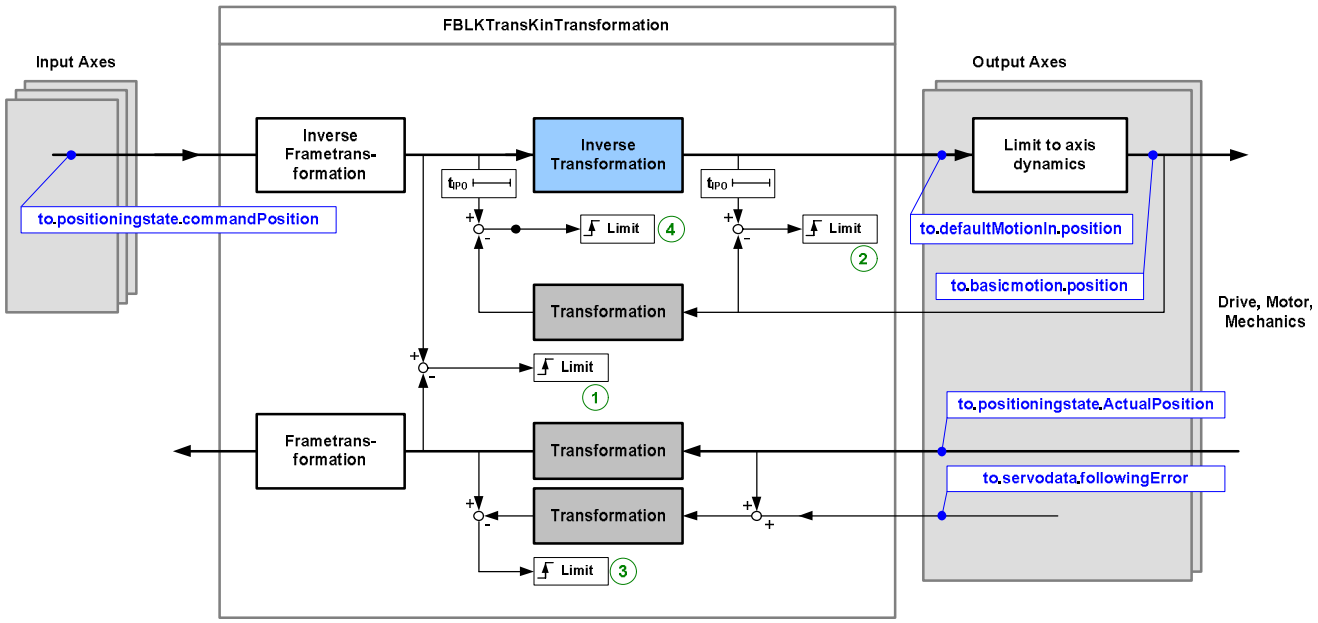


Fig. 6-11: Overview of FBLKTRansKinTransformation with monitoring functions

The following monitoring functions are implemented in the FB:

- Monitoring of the current path deviation
  (marked with 1 in Fig. 6-11, details are described in Section 6.2.8.1)
- Monitoring of the path setpoint deviation
  (marked with 4 in Fig. 6-11, details are described in Section 6.2.8.2)
- Monitoring of the path following error
  (marked with 3 in Fig. 6-11, details are described in Section 6.2.8.3)
- Monitoring of the axis setpoint
  (marked with 2 in Fig. 6-11, details are described in Section 6.2.8.4)

6.2.8.1 Path deviation

The path deviation determines the deviation between the current, Cartesian position setpoint and actual position. The current deviation is output under *sGeneral.sActPathTolerance.r64ActualValue* in the parameter structure.
However, as in principle there is always a time delay between the setpoint and returned actual axis position with the TO axis (bus system + interpolation cycle), this value does not correspond to the actual deviation on the TCP at the same time. There is a velocity-dependent deviation.
This deviation is therefore only used at standstill during the activation of the transformation.
Details are described in Section 6.2.7.

6.2.8.2 Path setpoint deviation

To determine the path setpoint deviation, the limited axis positions are transformed and compared with the position setpoint of the last call. The current deviation is output under *sGeneral.sCmdPathTolerance.r64ActualValue* in the parameter structure, when monitoring is active. The limit value is specified under *sGeneral.sCmdPathTolerance.r64Limit*; a value of 0.0 deactivates the monitoring. If the limit is exceeded while the block is in the ACTIVE state, an error is output. Due to the error, the block changes to the DEACTIVATING_MI state and the output axes are stopped individually with the specified dynamic responses (for details, see Section 5.2.5).

A deviation results when the transformation and inverse transformation do not match (error in the transformation equation), or when the TO axis cannot follow the required motion and limits this.

6.2.8.3 Path following error

The path following error is determined by also forming the associated position setpoint for the same time and transforming it in addition to the transformation of the current axis position. The position setpoint is formed by adding the following error and the current position. The current deviation is output under *sGeneral.sActPathFollowingError.r64ActualValue* in the parameter structure, when monitoring is active. The limit value is specified under *sGeneral.sActPathFollowingError.r64Limit*; a value of 0.0 deactivates the monitoring. If the limit is exceeded while the block is in the ACTIVE state, an error is output. Due to the error, the block changes to the DEACTIVATING_MI state and the output axes are stopped individually with the specified dynamic responses (for details, see Section 5.2.5).

A larger deviation results when the power limit of the drives has been reached or if the axis and drive controllers have not been set adequately.

6.2.8.4 Setpoint monitoring of the axis

A setpoint monitoring can be activated separately for each axis on the real output axes. The FBLKTransKinTransformation function block writes the cyclic setpoints of the inverse transformation to the motion interface (MotionIn) of the axis. If these values exceed the configured, maximum dynamic response values of the motion interface, they are limited to these values. This results inevitably in deviation from the specified path. The TO axis now signals alarm 40020, which however does not result in an error response. The following error monitoring of the axis also does not respond, because this only compares the setpoint and actual value on the position controller.

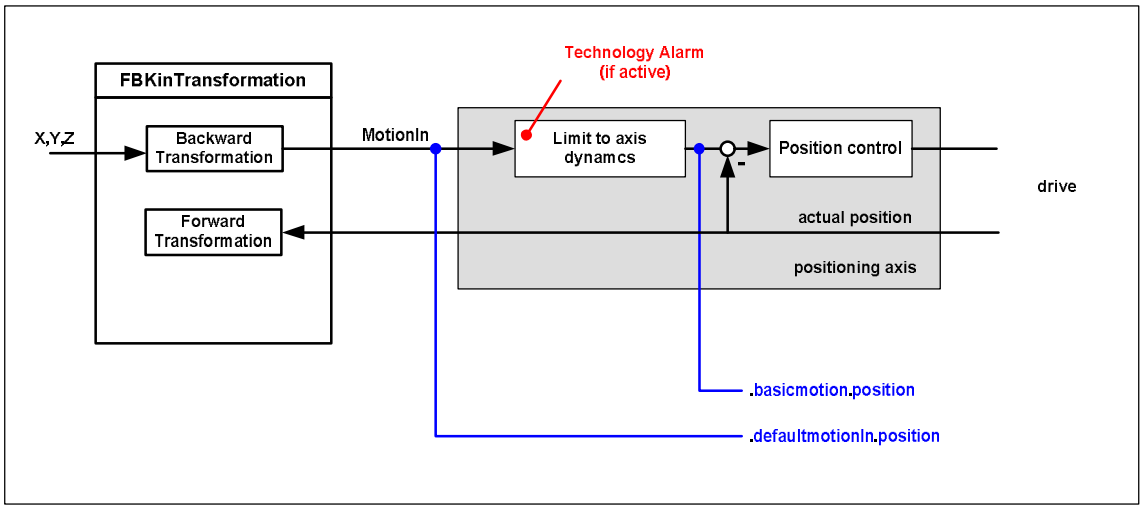


Fig. 6-12: Setpoint monitoring structure

The setpoint monitoring of the block compares the system variables *(axis).defaultmotionIn.position* and *(axis).basicmotion.position* with each other. The offset of one IPO cycle is taken into account. If the difference exceeds the configured limit in the ACTIVE state, an error is output, all axes are stopped individually and the block is deactivated (for details, see Section 5.2.5).
The limit is set in the *asOutputaxis[n].sCmdValueTolerance.r64Limit* parameter. A value of 0.0 results in deactivation of the monitoring on this axis.

6.2.9 Setting of the dynamic response parameters

The following grading of the dynamic response limits must be observed when using the FBLKTransKinTransformation.

The configured axis limits (of the real positioning axis) should not be exceeded.
The velocity limit is set on the configuration variable *typeOfAxis.maxVelocity.maximum* and the system variable *plusLimitsOfDynamics.velocity* (as well as *minusLimitsOfDynamics.velocity*) - the currently lower value is effective here. The same applies for the acceleration with the variables *typeOfAxis.maxAcceleration.maximum* and *plus/minusLimitsOfDynamics.positiveAccel*, (as well as *plus/minusLimitsOfDynamics.negativeveAccel*).

The motion interface can be activated as maximum with the axis limits.
For this purpose, the same or smaller values must be specified in parameters *asOutputAxis[n].sMIDynamics.r64Velocity*, *asOutputAxis[n]. sMIDynamics.r64Acceleration* and *asOutputAxis[n]. sMIDynamics.r64Deceleration* in the parameter structure. If the values exceed those set on the axis, alarm 40020 is output by the system when the motion interface is activated and all cyclic position specifications are limited accordingly. The error is present continuously, but operation is not affected.

The cyclic position values and the resulting velocity and acceleration should not exceed either the axis limits or the maximum values for the motion interface. If this happens nevertheless, alarm 40020 is also output and the values limited.

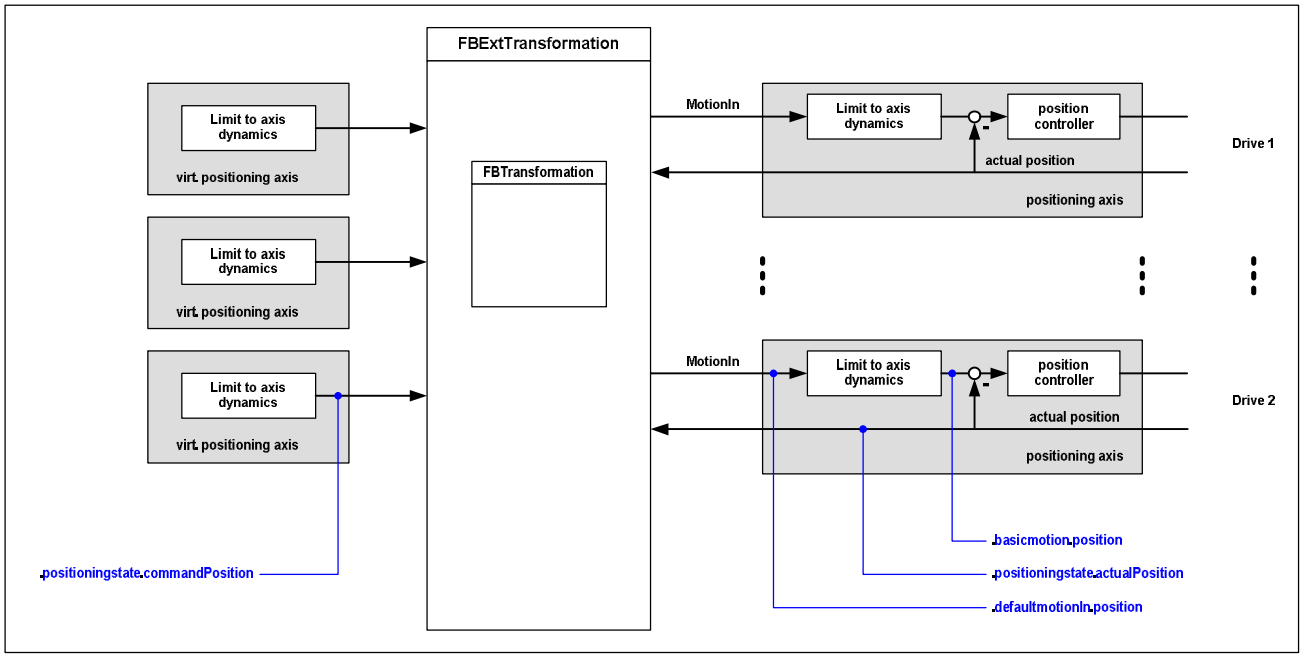


Fig. 6-13: Dynamic response limits when using the FBKintranformation

6.2.10 Example of dynamic responses and monitoring functions

Dynamic response of an output axis
The motion interface is to be operated with the maximum dynamic responses.
For this reason, the configured axis limits are also to be used as the dynamic response limits for the motion interface. Values above the axis limits are not permitted.

Limitation	Dynamic response limits			
	Velocity	Acceleration	Deceleration	Jerk
TO axis (technology object limitation)	2000.0 mm/s	20.000 mm/s²	20.000 mm/s²	200.000 mm/ s³
sMIDynamics	2000.0 mm/s	20.000 mm/s²	20.000 mm/s²	-
sStopDynamics	-	-	10.000 mm/s²	200.000 mm/ s³
sPullDynamics	100.0 mm/s	1.000 mm/s²	1.000 mm/s²	10.000 mm/ s³

The dynamic responses used for stopping in the event of an error must be below the axis limits.
The dynamic responses with which the axes are pulled for deviations are selected much lower.
Monitoring of the axis
A limitation by the axis during operation is to be monitored. Only very small deviations are to be tolerated; therefore the limit is set small.
gsParameter.asOutputaxis[..].sCommandValueTolerance.r64Limit := 0.5; //mm

Monitoring during activation of the motion interface
During activation of the motion interface, direct activation is only to be performed with very small deviations of the TCP, because resulting compensating movements are performed with the high dynamic responses of the motion interface.
gsParameter.sGeneral.sActPathTolerance.r64ActivateMIDirectLimit := 0.1; //mm
Up to a deviation of 50 mm, the output axes should be positioned at the position setpoint.
gsParameter.sGeneral.sActPathTolerance.r64ActivateMIPullAxesLimit := 50.0; //mm
The input axes are also set to the actual position.

Monitoring of the motion interface during operation
The path setpoint monitoring is also set to a small value, because setpoint deviations should not occur.
gsParameter.sGeneral.sCmdPathTolerance.r64Limit := 1.0; //mm
A larger value is permitted for the path following error, because this is determined by the system (controller setting).
gsParameter.sGeneral.sActPathFollowingError.r64Limit := 10.0; //mm

6.2.11 Frame transformation

A frame transformation for spatial adaptation has been integrated in the function block. The orientation of the geometric coordinate system can be rotated and offset with this frame. In order to make a clear distinction from the term "frame" as it is used in the NC language, the following description explains the meaning of the term "frame" in relation to FBLKTransKinTransformation.

Frame
A frame can be used to translate one coordinate system into another.
A differentiation is made between translation and rotation. In the forward direction, first the rotation and then the translation is calculated.
All settings described here are in the parameter structure in the *sTransformation.sCoordTrans* area (see Section 5.2.3).

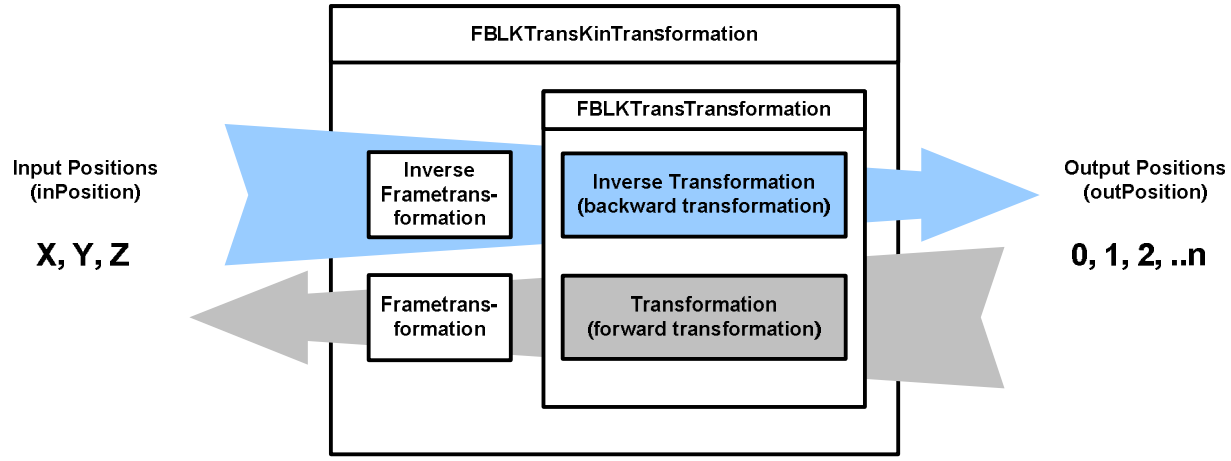


Fig. 6-14: Frame transformation in the FBKin transformation

6.2.11.1 Translation

The X, Y and Z coordinates are used to describe the translation.
They are defined so as to result in a right-handed coordinate system.
The variables ar64Translation[1 .. 3] are used to define the position of the embedded coordinate system in the basic coordinate system (position of the coordinate origin of the embedded coordinate system in the basic coordinate system).

ar64Translation[1]: X position
ar64Translation[2]: Y position
ar64Translation[3]: Z position

6.2.11.2 Rotation

The RPY angles ar64Rotation[1], ar64Rotation[2], ar64Rotation[3] are used for the rotation (RPY stands for Roll Pitch Yaw). This means that rotation is first performed around the X axis, then around the (already rotated) Y axis and then around the (already twice-rotated) Z axis. The positive direction of rotation is specified by the "right-hand" rule, i.e. if thumb of the right hand points in the direction of the rotary axis, the remaining fingers indicate the positive angle direction.
Note that it makes sense to select aRot[1] and aRot[3] in the interval [–180; +180] and aRot[2] in the interval [–90; +90] in order to exclude ambiguities.

The angles are defined as follows within the kinematics structure:

ar64Rotation[1]: **Roll** around the X axis
ar64Rotation[2]: **Pitch** around the (rotated) Y axis
ar64Rotation[3]: **Yaw** around the (already twice-rotated) Z axis

Fig. 6-15 shows an example for the rotation through the specified angles. In this case, the initial coordinate system X1, Y1, Z1 is first rotated through the ar64Rotation[1] angle around the X1 axis, then through the ar64Rotation[2] angle around the Y2 axis, and finally through the ar64Rotation[3] angle around the Z3 axis.

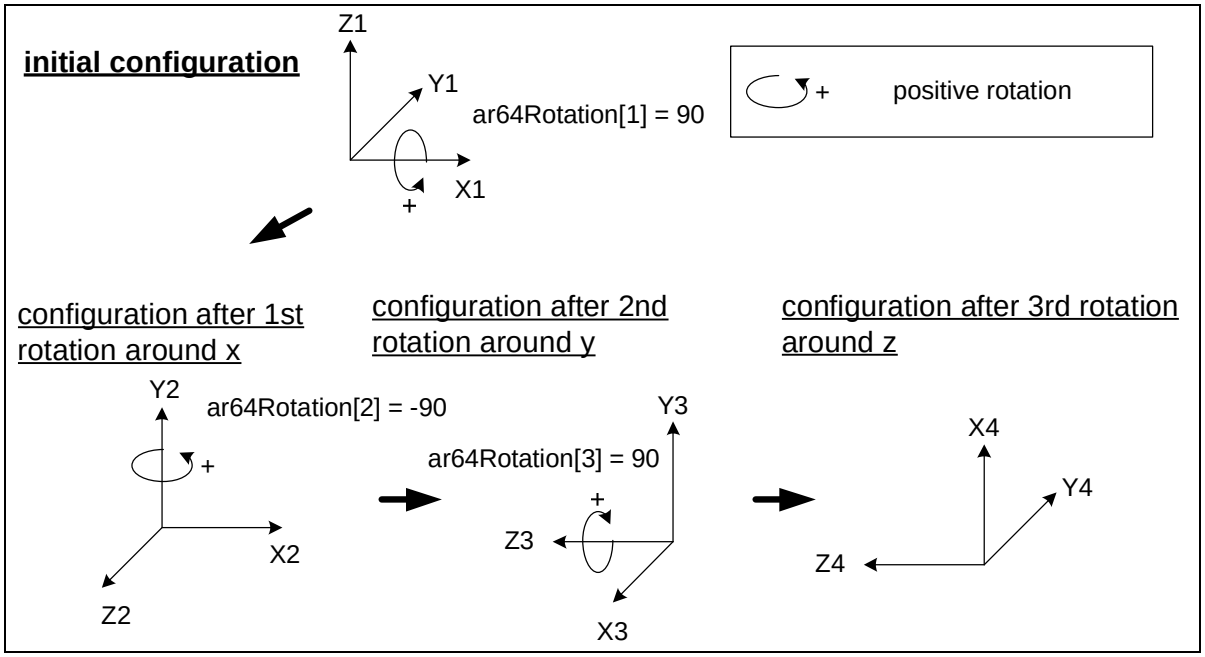


Fig. 6-15: Rotation of the coordinate system

The functions FCLKTRansFrameTrans and FCLKTRansInvFrameTrans are used internally for the frame transformation. These can also used by the user and are described in Section 6.3.

6.3 Functions FCLKTransFrameTrans and FCLKTransInvFrameTrans

6.3.1 Task

The functions FCLKTRansFrameTrans and FCLKTRansInvFrameTrans can be used to adapt coordinate systems using frame transformation. A distinction is made between translation and rotation.
The original coordinate system is transferred as an array of the data type LREAL. Whereby the indices 0, 1 and 2 are interpreted as X, Y and Z. The FCLKTRansFrameTrans function adapts the coordinate system by the specified values and returns it as a return value (also as array). The FCLKTRansInvFrameTrans acts in the same way, only that it transforms an adapted coordinate system back to the original system. The frame parameters are transferred as a structure of the *sLKTransFrameTransType* type (see 5.2.3) at the *coordIn* input parameter. Details about the principle of operation can be found in Section 6.2.11. In the FCLKTRansFrameTrans function, the rotation is calculated first, then the translation; in the FCLKTRansInvFrameTrans, first the translation, then the rotation.

6.3.2 Schematic LAD/FBD diagram

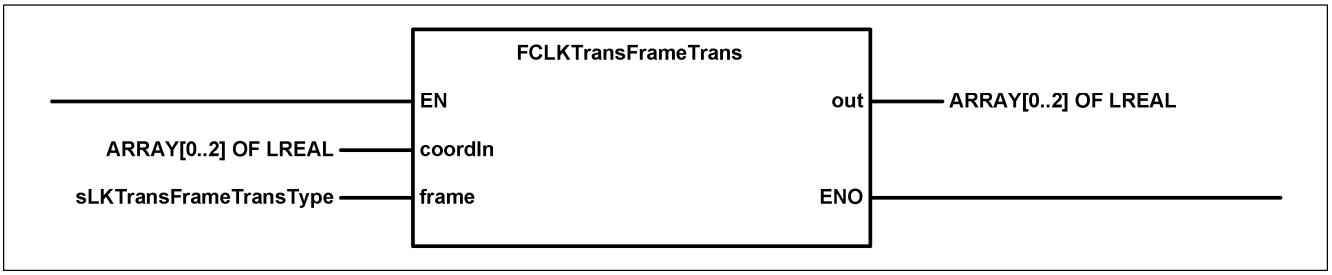


Fig. 6-16: LAD view of the FCLKTRansFrameTrans

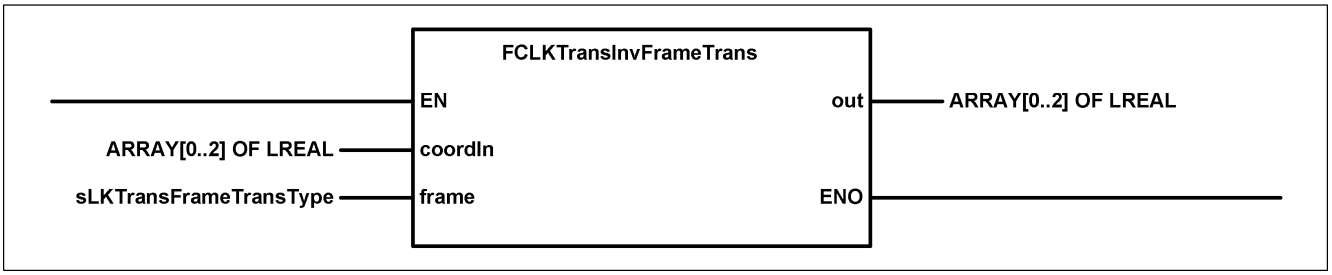


Fig. 6-17: LAD view of the FCLKTRansInvFrameTrans

6.3.3 Parameters of the functions

Name	P type	Data type	Meaning
coordIn	IN	ARRAY[0..2] OF LREAL	Coordinate system (X, Y, Z)
frame	IN	sLKTransFrameTransTyp e	Frame structure, contains parameters for offsets and rotations
out	OUT	ARRAY[0..2] OF LREAL	Return value: Adapted coordinate system (X, Y, Z)

7 Integration of own kinematics

To create a separate kinematic transformation, the user should perform the following steps.

7.1 Adaptation of constants

A check must be made as to whether there are sufficient input/output values and transformation parameters available or whether the number must be increased using the constants. A reduction of the already defined number usually does not make sense as implemented kinematics may use these listed elements and compilation of the program code is then not possible.
The constants are defined in the *cPublic* library unit.
Per default, four input positions (e.g. for X, Y, Z and W) and ten output positions are already specified for the FBLKTransTransformation. A list of twenty transformation parameters (e.g. for mechanical dimensions, link constellations and offset positions) has been provided.

```
//constants for FBLKTransTransformation
//-----------------------------------------------------
    //number of input values, range: 3..USINT#MAX
    LKTRANS_NUMBER_OF_INPOSITIONS            : USINT := 4; //default: 4

    //number of output values, range: 1..USINT#MAX
    LKTRANS_NUMBER_OF_OUTPOSITIONS           : USINT := 10; //default: 10

    //number of transformation parameter, range: 1..USINT#MAX
    LKTRANS_NUMBER_OF_TRANS_PARAMETER        : USINT := 20;
```

Fig. 7-1: Code section for the FBLKTransTransformation constant definition in the cPublic unit

The number of input and output axes actually used on the FBLKTransKinTransformation can be be set separately.

7.2 Creating the transformation and inverse transformation

In order to use the entire functionality, the user must implement both the transformation and the inverse transformation.
In the inverse transformation, the machine axis positions (commandOutPosition[0,..,n]) are calculated from the Cartesian position setpoints (commandInPosition[0, 1, 2] = [X,Y,Z]).
In the transformation, the current machine axis positions (actualOutPosition[0,.., n]) are converted to the Cartesian actual positions (actualInPosition[0,1,2] = [X,Y,Z]).
The integration of mechanical dimensions, offsets or similar is performed via the *parameter[n]* input/output parameter which is made up of an array of the LREAL type. This can be found under *.sTransformation.ar64Parameter[n]* in the parameter structure of the FBLKTransKinTransformation. A separate transformation number must be defined for each new transformation.
If the transformation is also to be used as information function in the user program, the calcMode input parameter with its three states TRANS, INV_TRANS and BOTH must be taken into account.

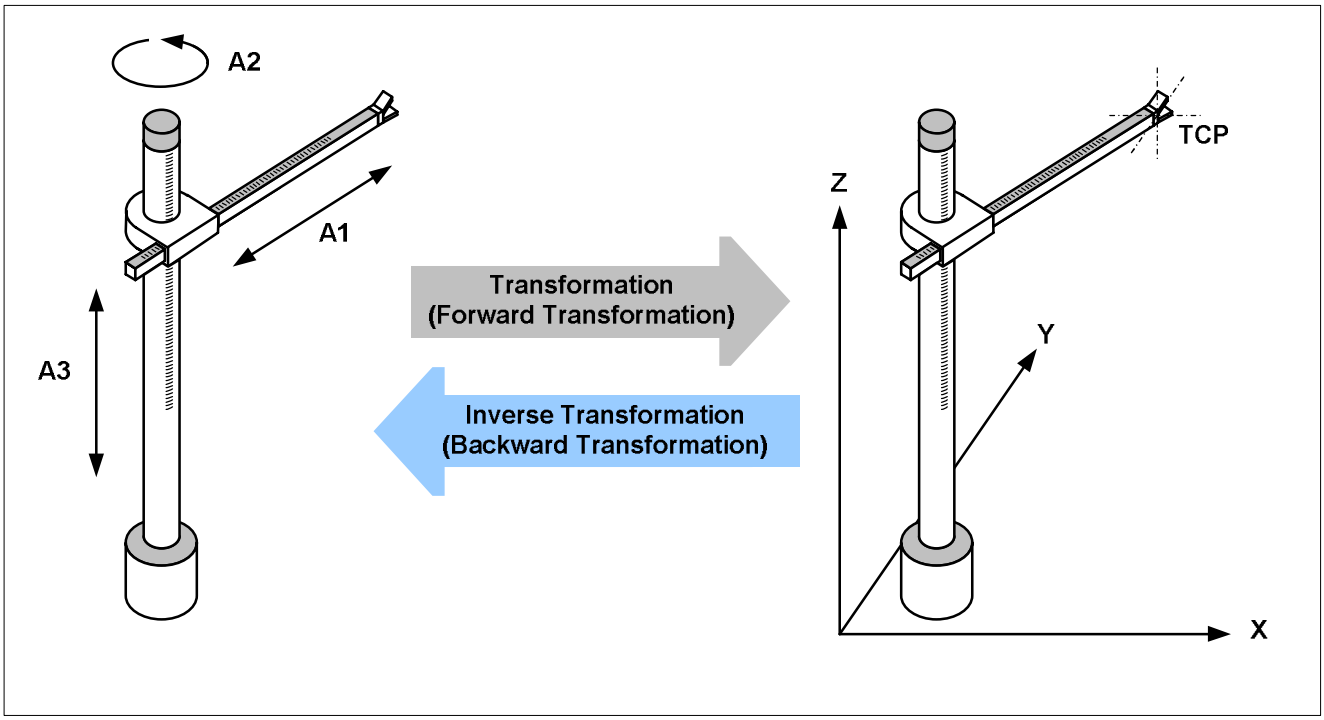


Fig. 7-2: Transformation and inverse transformation

A simple transformation example is shown in Fig. 7-3.

```
IF calcMode = TRANS OR calcMode = BOTH THEN
    //transformation
    //-------------------------------------------------------------
    actualInPosition[X] := actualOutPosition[0] * COS(DEGRAD * actualOutPosition[1]);
    actualInPosition[Y] := actualOutPosition[0] * SIN(DEGRAD * actualOutPosition[1]);
    actualInPosition[Z] := actualOutPosition[2];
END_IF;
```

Fig. 7-3: Simple transformation example

To increase the readability, the constants X, Y and Z are used in the displayed code for the indices of the Cartesian positions. Appropriate constants such as DEGRAD are available for easy conversion of ° (radian measure, 0..2π) to ° (angular degrees, 0..360).

```
VAR_GLOBAL CONSTANT
    //Constant PI
    PI      : LREAL := 3.14159265358979323846264338327795;
    //Angle[RAD]=DEGRAD*Angle[DEG]
    DEGRAD  : LREAL := 1.74532925199432957692369076848e-2;
    //Angle[DEG]=RADDEG*Angle[RAD]
    RADDEG  : LREAL := 57.295779513082320876798154814105;
    //use for axes
    X       : USINT := 0;
    Y       : USINT := 1;
    Z       : USINT := 2;
    W       : USINT := 3;
END_VAR
```

Fig. 7-4: Available user constant

The user must ensure that errors are detected. Fig. 7-5 shows the call of an error by the user. The error output variable is set and the error number assigned as LKTRANS_ERR_XY_UNDEFINED constant to the errorId output parameter. You can create your own error constants in the cPublic library unit. There can now be a response to the error message outside of the block.

```
//inverse transformation
IF  parameter[2] <= SQRT(
    commandInPosition[X] ** 2.0
    + commandInPosition[Y] ** 2.0 )
    THEN
    //calculate inverse transformation
    //...


ELSE
    //error
    error := TRUE;
    errorId := LKTRANS_ERR_XY_UNDEFINED; //error number 4002
END_IF;
```

Fig. 7-5: Example (section) of an error query

8 Errors and warnings

The following table describes all the errors that can occur with the function blocks **FBLKTransTransformation** and **FBLKTransKinTransformation**. Errors on the FBLKTransTransformation are determined in each cycle and do not have to be acknowledged.
If an error occurs on the FBLKTransKinTransformation block, further details are displayed in the diagnostics structure on the *diagnostics* block parameter. Parameters *i8AxNumber*, *i32Detail1*, *b32Detail2*, *r32Detail3* are available for this.
There is also an error buffer (*asBuffer*) in the form a ring buffer in the diagnostics structure which contains the past errors together with the time of occurrence. The current (last) entry is displayed by the *i32BufferIndex* parameter.
Errors at this block must be acknowledged by resetting the enable input parameter.

Table 2: Error list

Error no. hex (dec) **short designation**	**Meaning/details**
0000_4000 (16384) LKTRANS_ERR_UNKNOWN_TRANSID	An unknown transformation number has been selected on the FBLKTransTransformation block.
0000_4001 (16385) LKTRANS_ERR_OUTAX_0_POSITION	The current position of output axis 0 is outside the valid range.
0000_4002 (16386) LKTRANS_ERR_XY_UNDEFINED	The specified, Cartesian position (X,Y) cannot be reached with the current parameterization.
0000_4003 (16387) LKTRANS_ERR_SINGULARITY	An error has occurred while calculating the inverse transformation.
0000_4004 (16388) LKTRANS_ERR_OUT_OF_RANGE	The kinematics cannot reach the position with the specified geometric dimensions.
0000_4003 .. 0000_4999	Reserved for further user errors in the FBLKTransTransformation.
0000_5000 (20480) LKTRANS_ERR_TRANSFORMATION	FBLKTransKinTransformation: An error has occurred in the transformation block (FBLKTransTransformation). Detailed information: **b32Detail2**: Error number of the transformation block
0000_5001 (20481) LKTRANS_ERR_STATE_REDEFINE_ COMMAND	Error while setting the input axes using the _redefinePosition() system function. Detailed information: **u8AxNumber**: Index of the relevant input axis **i32Detail1**: 1 = error, 2 = abort **b32Detail2**: Error number in accordance with PLCopen

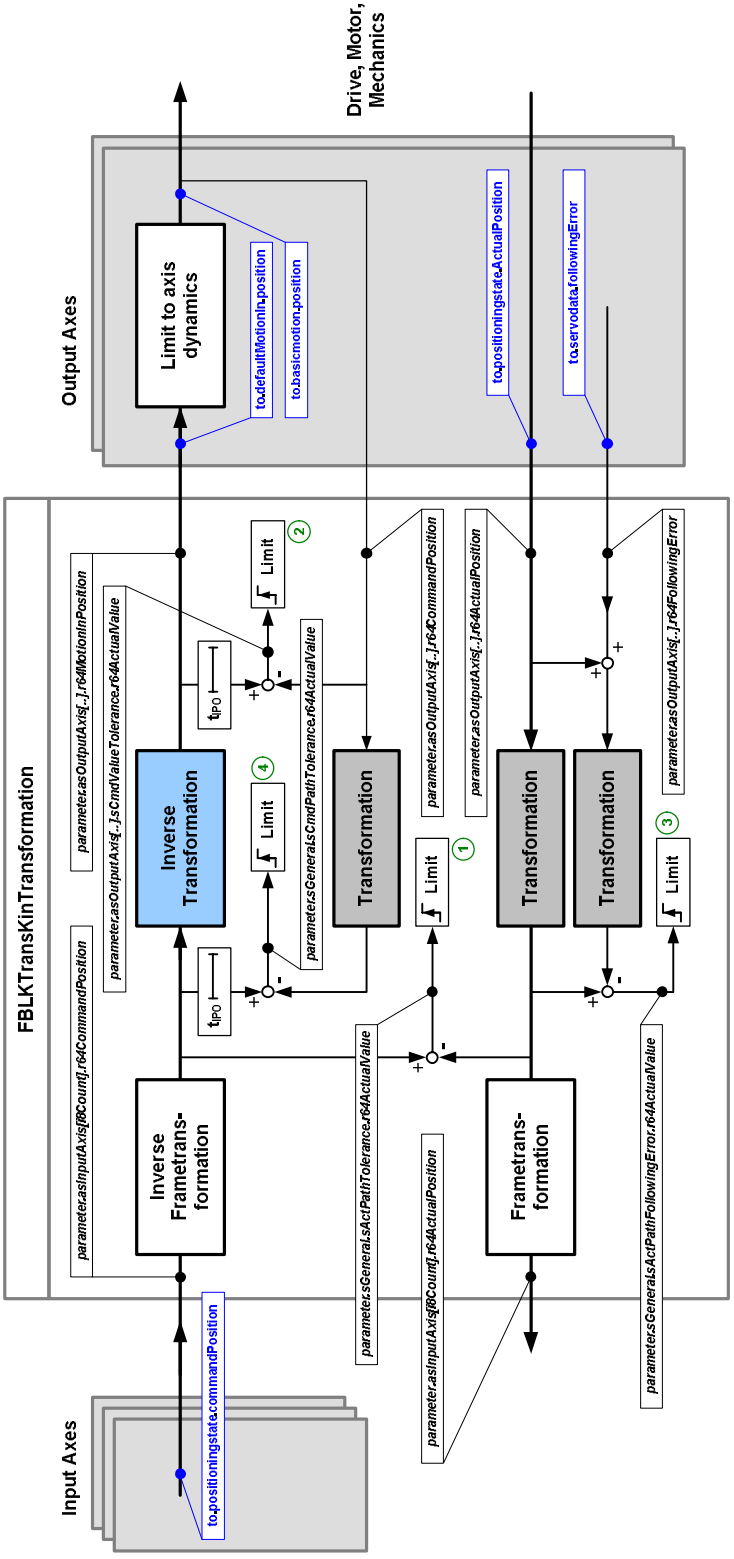
Error no. hex (dec) short designation	Meaning/details
0000_5002 (20482) LKTRANS_ERR_MOTIONIN	Error while activating or monitoring the motion interface using the _runPositionBasedMotionIn() system function. Detailed information: **u8AxNumber**: Index of the relevant output axis **i32Detail1**: 1 = error, 2 = abort **b32Detail2**: Error number in accordance with PLCopen
0000_5003 (20483) LKTRANS_ERR_CONSTANTS	The values for the user constants (see Section5.1) of the cPublic library unit are invalid or not matched. Detailed information: **i32Detail1**: 1 = NUMBER_OF_TRANS_PARAMETER must be greater than 0 2 = NUMBER_OF_INPOSITIONS must be greater than 0 3 = NUMBER_OF_OUTPOSITIONS must be greater than 0 4 = NUMBER_OF_INPOSITIONS must be greater than NUMBER_OF_USED_INPUT_AXES 5 = NUMBER_OF_OUTPOSITIONS must be greater than NUMBER_OF_USED_OUTPUT_AXES
0000_5004 (20484) LKTRANS_ERR_LIMIT_REACHED_ AFTER_REDEF	One or more limits have been exceeded after setting the input axes to transformed positions, and therefore the activation of the motion interfaces aborted. The block changes to the INACTIVE state. The cause of this error can be limits that have been selected too low or deviations between the transformation and the inverse transformation. Detailed information: **u8AxNumber**: Index of the relevant axis (if Details1 = 4) **i32Detail1**: Exceeded monitoring 1 = limit for direct activation of the MI (sActPathTolerance.r64ActivateMIDirectLimit) 2 = limit for pulling the axes. Parameter (sActPathTolerance.r64ActivateMIPullAxesLimit) 3= monitoring in accordance with 1 and 2 have been exceeded 4 = setpoint monitoring of the axes. Parameter (asOutputAxis[..].sCommandValueTolerance.r64Limit) **r32Detail3**: Deviation from the triggering time
0000_5005 (20485) LKTRANS_ERR_AX_CMDVAL_TOLERANC E	With an activated motion interface, the setpoint monitoring has been exceeded on at least one output axis, (parameter sCmdValueTolerance.r64Limit). As response, the motion interface is deactivated. Detailed information: **u8AxNumber**: Index of the relevant output axis **r32Detail3**: Set (exceeded) tolerance
0000_5006 (20486) LKTRANS_ERR_PATH_TOLERANCE	With an activated motion interface, the configured path tolerance has been exceeded (parameter r64ActualPathToleranceLimit). As response, the motion interface is deactivated. Detailed information: **r32Detail3**: Set (exceeded) tolerance

Error no. hex (dec) short designation	Meaning/details
0000_5007 (20487) LKTRANS_ERR_AXES_ PRECONDITION	The requirements of at least one axis have not been satisfied when activating the motion interface. Detailed information: **u8AxNumber**: Index of the relevant axis **i32Detail1**: 1 = input axis, 2 = output axis **b32Detail2**: Current status word (see Section 5.2.7) of the relevant axis (The following bits must be set: 0, 1, 2, 3, 4)
0000_5008 (20488) LKTRANS_ERR_OUTPUTAXIS	A condition has not been satisfied on at least one output axis during operation of the motion interface. Detailed information: **u8AxNumber**: Index of the relevant axis **i32Detail1**: 1 = input axis, 2 = output axis **b32Detail2**: Current status word (see Section 5.2.7) of the relevant axis (The following bits must be set: 0, 1, 2, 3)
0000_5009 (20489) LKTRANS_ERR_NO_AXES_USED	No axis references could be read out for use from the parameter structure. At least one axis must be used. Detailed information: **i32Detail1**: 1 = input axes, 2 = output axes
0000_500A (20490) LKTRANS_ERR_ STATE_POS_COMMAND	An error has occurred in the PULL_AXES state. The positioning of one or more axes has been aborted. The system function block _MC_MoveAbsolute() is used for this. **u8AxNumber**: Index of the relevant axis **i32Detail1**: 1 = error, 2 = abort (commandAborted) **b32Detail2**: Error number in accordance with PLCOpen
0000_500B (20491) LKTRANS_ERR_ LIMIT_REACHED_AFTER_PULL	After pulling the output axes (PULL_AXES), the limit of a monitoring function is still being exceeded. The block changes to the INACTIVE state. The configured limit values should be checked. **u8AxNumber**: Index of the relevant axis (if Details1 = 4) **i32Detail1**: Exceeded monitoring 1 = limit for direct activation of the MI (sActPathTolerance.r64ActivateMIDirectLimit) 2 = limit for pulling the axes. Parameter (sActPathTolerance.r64ActivateMIPullAxesLimit) 3= monitoring in accordance with 1 and 2 have been exceeded 4 = setpoint monitoring of the axes. Parameter (asOutputAxis[..].sCommandValueTolerance.r64Limit) **r32Detail3**: Deviation from the triggering time
0000_500C (20492) LKTRANS_ERR_ INPUT_PARAMETER	An internal error has occurred in the calculation of the motion vector. The function FBLKTransCalcMotionVec() is used for this. Please contact your SIEMENS representative. **i32Detail1**: Current calculating mode (eLKTransMICalculationType)
0000_500D (20493) LKTRANS_ERR_ SPECIFIC_POINT	An internal error has occurred in the calculation of the motion vector. The function FBLKTransCalcMotionVec() is used for this. Please contact your SIEMENS representative. **i32Detail1**: Current calculating mode (eLKTransMICalculationType)

Error no. hex (dec) short designation	Meaning/details
0000_500E (20494) LKTRANS_ERR_ PATH_CMDVAL_TOLERANCE	The monitoring of the path setpoint has detected a violation of the limit value. The block changes to the INACTIVE state. **i32Detail1**: Currently set limit value (from parameter.sGeneral.sCmdPathTolerance.r64Limit)
0000_500F (20495) LKTRANS_ERR_ PATH_FOLLOWING_ERROR	The monitoring of the path setpoint has detected a violation of the limit value. The block changes to the INACTIVE state. **i32Detail1**: Currently set limit value (from parameter.sGeneral.sActPathFollowingError.r64Limit)
0000_5010 (20496) LKTRANS_ERR_ UNKNOWN_CALCULATION_MODE	A calculation mode has been selected that is not supported. The configuration (sGeneral.eMVecCalcMode) should be checked. **i32Detail1**: Selected (not supported) calculation mode (eLKTransMICalculationType)
0000_5011 (20497) LKTRANS_ERR_ TASK_CYCLE_TIME	An internal error has occurred in the calculation of the motion vector because the cycle time is faulty. The function FBLKTransCalcMotionVec() is used for this. Please contact your SIEMENS representative. **i32Detail1**: Current calculating mode (eLKTransMICalculationType)
0000_5012 (20498) LKTRANS_ERR_ STATE_STOP_COMMAND	An error has occurred in the STOP_PULL_AXES state. The stopping of one or more axes has been aborted. The system function block_MC_Stop() is used for this. **u8AxNumber**: Index of the relevant axis **i32Detail1**: 1 = error, 2 = abort (commandAborted) **b32Detail2**: Error number in accordance with PLCOpen

9 Appendix

9.1 Monitoring functions



9.2 Contact partners

Siemens AG
Industry Sector
I DT MC PM APC
Frauenauracher Strasse 80
D - 91056 Erlangen
Fax.: +49 9131 98 1297
E-mail: applications.erlf.aud@siemens.com

9.3 Internet addresses

Additional information on various topics can be found at the following Internet addresses:

SIMOTION (www.siemens.com/simotion)

SINAMICS (www.siemens.com/sinamics)

Handling (www.automation.siemens.com/mc-app/handling/html_00/00-handling.htm)

TopLoading (http://support.automation.siemens.com/WW/view/de/37584177)

Motion control / Application Center (www.siemens.com/motioncontrol/apc)

10 Index

INDEX