

TANGO-DLL Documentation

Of TANGO DLL
Version 1.520



In der Murch 15
35579 Wetzlar
Germany
Tel.: +49/6441/9116-0
www.marzhauser.com

Table of Contents

1.	Introduction	9
1.1.	Functional Range.....	9
1.2.	System Requirements	9
1.3.	Supported Development Environments	9
2.	DLL-Interface.....	10
2.1.	General Information.....	10
2.2.	Default API for C++	11
2.3.	Integration in Visual C++	12
2.4.	Integration in Visual Basic	12
2.5.	Integration in LabVIEW.....	13
2.6.	Integration in MATLAB	16
3.	General Information of DLL Usage	17
3.1.	Initialization of Controller.....	18
3.2.	Own Program Section	20
3.3.	API State Diagram.....	21
4.	Functions.....	22
4.1.	Quick Reference	22
4.2.	DLL Configuration / Interface.....	32
	CreateLSID	32
	ConnectSimple	32
	ConnectEx	33
	LoadConfig	34
	Connect	34
	SaveConfig	34
	Disconnect.....	34
	FreeLSID	35
	SetShowProt.....	35
	ClearProtocolWindow	35
	SetLanguage	35
	GetCommandTimeout	36
	SetCommandTimeout.....	36
	EnableCommandRetry	36
	SetDllNumOfAxes.....	36
	GetSwapZA	37
	SetSwapZA.....	37
	FlushBuffer	37
	GetDLLVersionString.....	37
	SendString.....	38
	SendStringPosCmd	39
	SetAbortFlag.....	39
4.3.	Controller Information.....	40
	GetSerialNr	40
	GetTangoVersion	40
	GetVersionStr	40
	GetVersionStrDet.....	40
	GetVersionStrInfo	41

	GetStageSN	41
4.4.	Status Requests.....	42
	GetError.....	42
	GetErrorString	42
	GetPos	42
	GetPosEx	42
	GetPosSingleAxis	43
	SetPos.....	43
	ClearPos.....	43
	GetStatus	43
	GetStatusAxis.....	44
	GetStatusLimit	44
	GetStA.....	45
	SetAutoStatus.....	46
	GetAutoStatus	46
	IsVel	47
	IsVelSingleAxis	47
4.5.	Settings	48
	GetPowerAmplifier.....	48
	SetPowerAmplifier	48
	GetActiveAxes	48
	SetActiveAxes	48
	GetAxisDirection	49
	SetAxisDirection	49
	GetCalibOffset	49
	SetCalibOffset	49
	GetRMOffset.....	50
	SetRMOffset	50
	GetCalibBackSpeed	50
	SetCalibBackSpeed.....	50
	GetCalibrateDir	51
	SetCalibrateDir	51
	GetDimensions	51
	SetDimensions	52
	GetResolution.....	52
	SetResolution	53
	GetPitch.....	53
	SetPitch	53
	GetGear.....	54
	SetGear	54
	GetMotorSteps	54
	SetMotorSteps.....	54
	GetMotorCurrent.....	55
	SetMotorCurrent	55
	GetReduction.....	55
	SetReduction	55
	GetCurrentDelay	56
	SetCurrentDelay	56
	GetSpeedPoti	56
	SetSpeedPoti.....	56
	GetStopPolarity	57
	SetStopPolarity.....	57
	GetVel	57
	SetVel.....	57
	SetVelSingleAxis	58

GetSecVel	58
SetSecVel.....	58
SetSecVelSingleAxis	58
GetVelFac	59
SetVelFac.....	59
GetAccel.....	60
SetAccel	60
SetAccelSingleAxis.....	60
GetAccelFunc	60
SetAccelFunc	60
GetStopAccel.....	61
SetStopAccel.....	61
GetBISmoothSingleAxis.....	61
SetBISmoothSingleAxis	61
LStepSave	62
SoftwareReset	62
4.6. Move Commands and Positioning Management.....	63
Calibrate	63
CalibrateEx	63
RMeasure	63
RMeasureEx.....	64
GetDelay	64
SetDelay	64
MoveAbs	65
MoveAbsSingleAxis	65
MoveEx	66
MoveRel	66
MoveRelSingleAxis.....	67
MoveRelShort.....	67
GetDistance.....	67
SetDistance	67
Go	68
GoSingleAxis	68
GoEx	68
GetDigJoySpeed	69
SetDigJoySpeed.....	69
StopAxes	69
StopAxesEx.....	69
WaitForAxisStop.....	70
4.7. Joystick and Handwheel	71
SetJoystickOff.....	71
SetJoystickOn.....	71
GetJoystickDir	71
SetJoystickDir.....	72
GetJoystick.....	72
GetJoyChangeAxis	73
JoyChangeAxis.....	73
GetHandWheel	73
SetHandWheelOff.....	73
SetHandWheelOn.....	74
GetJoystickWindow	74
SetJoystickWindow.....	74
GetHwFactor	75
SetHwFactor	75
GetHwFactorSingleAxis.....	75

	SetHwFactorSingleAxis	75
	GetHwFactorB	76
	SetHwFactorB	76
	GetHwFactorBSingleAxis	76
	SetHwFactorBSingleAxis	76
	GetZwTravel	77
	SetZwTravel	77
	GetHdiKeys	77
	GetKey	77
	GetKeyLatch	78
	ClearKeyLatch	78
	GetHdiSpeedIndex	79
	SetHdiSpeedIndex	79
	GetHdiSpeedIndexSingleAxis	79
	SetHdiSpeedIndexSingleAxis	79
4.8.	BPZ Console with Trackball and Joyspeed Keys	80
	GetBPZ	80
	SetBPZ	80
	GetBPZJoyspeed	80
	SetBPZJoyspeed	80
	GetBPZTrackballBackLash	81
	SetBPZTrackballBackLash	81
	GetBPZTrackballFactor	81
	SetBPZTrackballFactor	81
4.9.	Limit Switches (Hardware and Software)	82
	GetAutoLimitAfterCalibRM	82
	SetAutoLimitAfterCalibRM	82
	GetLimit	82
	SetLimit	83
	GetLimitControl	83
	SetLimitControl	83
	GetSwitchActive	84
	SetSwitchActive	84
	GetSwitchPolarity	84
	SetSwitchPolarity	85
	GetSwitchType	85
	SetSwitchType	85
	GetSwitches	86
4.10.	Digital and Analog Inputs and Outputs	87
	GetAnalogInput	87
	SetAnalogOutput	87
	SetLedBright	87
	GetAnalogOutputMode	88
	SetAnalogOutputMode	88
	SetAuxDigitalOutput	88
	GetAuxDigitalInput	89
	GetDigitalInputs	89
	GetDigitalInputsE	89
	SetDigitalOutput	90
	SetDigitalOutputs	90
	SetDigitalOutputE	90
	SetDigitalOutputsE	90
	SetDigIO_Distance	91
	SetDigIO_EmergencyStop	91

	SetDigIO_Off	91
	SetDigIO_Polarity	92
4.11.	TVR Clock & Direction IO	93
	GetTVRMode.....	93
	SetTVRMode	93
	GetFactorTVR	93
	SetFactorTVR	93
4.12.	Encoder Settings	94
	ClearEncoder.....	94
	GetEncoder	94
	GetEncoderActive.....	94
	SetEncoderActive	95
	GetEncoderMask	95
	SetEncoderMask	95
	GetEncoderSingleAxis	96
	SetEncoderSingleAxis	96
	GetEncoderPeriod	97
	SetEncoderPeriod.....	97
	GetEncoderPosition.....	97
	SetEncoderPosition	97
	GetEncoderRefSignal	98
	SetEncoderRefSignal	98
	GetRefSpeed.....	98
	SetRefSpeed	98
4.13.	Closed Loop Settings	99
	GetController	99
	SetController.....	99
	GetControllerCall	99
	SetControllerCall	100
	GetControllerFactor	100
	SetControllerFactor.....	100
	GetControllerFactorSingleAxis.....	101
	SetControllerFactorSingleAxis	101
	GetControllerSteps	102
	SetControllerSteps.....	102
	GetControllerTimeout	102
	SetControllerTimeout.....	102
	GetControllerTWDelaySingleAxis	103
	SetControllerTWDelaySingleAxis.....	103
	GetControllerTWDelay.....	103
	SetControllerTWDelay	103
	GetTargetWindow.....	104
	SetTargetWindow	104
	SetCtrFastMoveOff	104
	SetCtrFastMoveOn	104
	GetCtrFastMove	105
	GetCtrFastMoveCounter	105
	ClearCtrFastMoveCounter	106
4.14.	Trigger Output	107
	GetTrigger	107
	SetTrigger.....	107
	GetTriggerMode	107
	SetTriggerMode.....	107
	GetTriggerPar	108

SetTriggerPar	108
GetTrigCount	108
SetTrigCount	108
GetTriggerAxis	109
SetTriggerAxis	109
GetTriggerSignalLength	109
SetTriggerSignalLength	109
GetTriggerDistance	109
SetTriggerDistance	110
GetTriggerCompensation	110
SetTriggerCompensation	110
GetTriggerEncoder	110
SetTriggerEncoder	110
GetTriggerFrequency	111
SetTriggerFrequency	111
GetTriggerOutput	111
SetTriggerOutput	111
Get2ndTriggerDelay	112
Set2ndTriggerDelay	112
Get2ndTriggerWidth	112
Set2ndTriggerWidth	112
Get2ndTriggerFrequency	112
Set2ndTriggerFrequency	113
GetTriggerRange	113
SetTriggerRange	113
GetTriggerPositionList	114
SetTriggerPositionList	114
GetTriggerPositionListIndex	114
SetTriggerPositionListIndex	114
GetTriggerPositionListEntries	115
SetTriggerPositionListEntries	115
GetTriggerLevel	115
SetTriggerLevel	115
4.15. Snapshot Input	116
GetSnapshot	116
SetSnapshot	116
GetSnapshotMode	116
SetSnapshotMode	116
GetSnapshotCount	117
SetSnapshotCount	117
GetSnapshotFilter	117
SetSnapshotFilter	117
GetSnapshotPar	118
SetSnapshotPar	118
GetSnapshotPos	118
GetSnapshotPosArray	119
SetSnapshotPosArray	119
ClearSnapshotPosArray	119
GetSnapshotIndex	120
SetSnapshotIndex	120
5. SlideExpress Functions	121
Eject	122
Insert	122
SlideSeated	122

	MagazinSeated.....	122
	GetGripper.....	123
	SetGripper.....	123
	GetClipType.....	123
	GetSlide.....	124
	PutSlide.....	124
	GetPrioHandlerPos.....	124
	SetPrioHandlerPos.....	124
6.	TrayExpress Functions	125
	Eject	125
	Insert	125
	SlideSeated	125
	MagazinSeated.....	125
	GetGripper.....	126
	SetGripper.....	126
	GetTray	126
	PutTray.....	126
	GetRFID	127
	SetRFID.....	127
	GetNumberOfSlots	127
	GetNumberOfMagazines	127
7.	Additional Handling System Functions	128
	GetLoaderType.....	128
	GetNumberOfRows	128
	GetNumberOfColumns	128
	GetTraySN.....	128
	GetTrayType.....	129
	SetTrayType	129
	SetCabinLED.....	130
	GetCabinLED	130
	SetLabelLED	130
	GetLabelLED	130
8.	xPos Module (POS3 3 axis extension)	131
	Xpos3GetPosSingleAxis	131
	Xpos3SetPosSingleAxis	131
	Xpos3MoveAbsSingleAxis	131
	Xpos3MoveRelSingleAxis.....	131
9.	Error Codes	132
9.1.	Tango Error Messages	132
9.2.	Error Messages for SlideExpress and TrayExpress	133
9.3.	Error Messages of the RFID Interface	133
9.4.	Error Messages of the Piezo Z-Stage	133
9.5.	Error Messages of Custom Handling Systems.....	134
9.6.	DLL Error Messages.....	135
10.	Document Revision History	136

1. Introduction

The TANGO-DLL (programming interface for TANGO controllers) is designed to help software developers writing applications for 2/4-phase stepper motors fast and effectively without the need of hardware-oriented programming. The TANGO-DLL supports all commands of the TANGO controller.

1.1. Functional Range

- Windows DLL 32-bit and 64-bit
- Supports TANGO stepper motor controllers
- Control via RS232, Virtual COM Port (USB, PCI and PCI-E) or Ethernet
- Supports most controller commands directly
- Up to 4 axes per TANGO
- Up to 8 TANGO controllers per DLL

1.2. System Requirements

The Tango-DLL can be used on all Windows PCs from Windows 7 to Windows 11.

It requires the *Microsoft Visual C++ 2017 Redistributable Package* to be installed, which often is already installed on Windows PCs. If not, it can be downloaded from the Microsoft website:

<https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-170>

--> 32 bit: https://aka.ms/vs/17/release/vc_redist.x86.exe

--> 64 bit: https://aka.ms/vs/17/release/vc_redist.x64.exe

1.3. Supported Development Environments

The Tango-DLL is available as 32 Bit and 64 Bit version.

It has been tested on operating systems Windows 7, 8, 10 and 11 using following development tools:

- Microsoft Visual Studio 2010 languages Visual Basic, C# and C++
- Microsoft Visual Studio 2017 language C++
- Java
- Python
- MATLAB
- National Instruments LabVIEW
- Embarcadero Delphi 2007 and Delphi XE
- Compatibility is assumed for all other programming environments which can use a DLL.

(DLL = Dynamic Link Library, generally means a dynamic library. In programming, a software library is a collection of program functions for tasks belonging together. Other than programs, libraries are not independently operating units, but auxiliary modules, which are made available to programs.)

2. DLL-Interface

Main part of the Tango DLL is the data file `Tango_DLL.dll`. Use this file for developing own programs to configure the TANGO, calibrate and move, send commands, retrieve input- and output states, etc.

2.1. General Information

The DLL will require one or two more files (.h and/or .c) that are provided with the API, depending on how the DLL is used. Refer to the following examples. It is important to use the right DLL, for 64 bit programs use the 64 bit DLL and use the 32 bit version for 32 bit x86 programs.

All DLL functions return a 32-bit integer value, where 0 (zero) indicates the error free execution of a function. For other integer values refer to chapter **Error Codes**. [`GetErrorString`](#) can be used to translate an error code into a short English explanation text.

The functions described in this document (chapter 4) use the "LSX_" commands, in which the first value stands for the TANGO ID (LSID). This ID is used to address up to 8 controllers simultaneously through one DLL. But it is recommended to use the "LS_" instructions and open the DLL for each TANGO separately. Then, in function calls the first value (the TANGO ID) is skipped, and CreateLSID / FreeLSID is not required.

Example

"LS" Function Call:

```
LS_MoveAbs(50.0, 50.0, 50.0, 10.0, TRUE);
```

"LSX" Function Call:

```
LSX_MoveAbs(1, 50.0, 50.0, 50.0, 10.0, TRUE);  
// the first value is the LSID, which is not needed with "LS_" function calls
```

With functions such as `LSX_MoveAbs`, values of 4 axes must be passed to the function.

If the controller only provides 1-3 axes, values of the not available axes are ignored and can be set to 0.

2.2. Default API for C++

In C++, direct access to LSX DLL functions is easily possible via the `TangoLSX_API.h` header file, but it is recommended to access each TANGO by an individual DLL as described in the next chapter.

Example for LSX functions: Access two TANGOs with one DLL

- The `TangoLSX_API.h` header file from the TANGO DLL folder must be included (by `#include`)
- In the C++ project, the `Tango_DLL.lib` file must be added by going to: Project Properties / Linker Input / Additional Dependencies - and adding the `Tango_Dll.lib`; there.
- It might be useful to copy the TANGO files from the TANGO CD or USB-Stick's "API & DLL" in a small folder structure to the project, e.g. in a "TANGO-DLL" folder with two "32" and "64" named subfolders.

The `TangoLSX_API.h` directly in the TANGO-DLL folder and the corresponding 32 and 64 bit `Tango_DLL.dll` DLLs and their `Tango_DLL.lib` files in the 32 and 64 named sub-folders. Then, the 32 and 64 bit project properties can have their `Tango_DLL.lib` file added as `TANGO_DLL\32\Tango_DLL.lib` and the 64 bit build as `TANGO_DLL\64\Tango_DLL.lib`.

The DLL example "VS2017_Cpp_64bit" is made this way - the DLL, LIB and the API header files are there in an own "Distrib" folder, and in the Project Properties, the entry of the `Tango_DLL.lib` file can be found (for 32 and for 64 bit program compile versions, the corresponding folder is specified).

```
#include "..\MyDllFolder\TangoLSX_API.h" // Include the API Header file for LSX function calls

int IdTangoXY = 0;
int IdTango_Z = 0;
int result;
char text[128] = "";

result = LSX_CreateLSID(&IdTangoXY);
result = LSX_CreateLSID(&IdTango_Z);

result = LSX_ConnectSimple(IdTangoXY, 1, "COM11", 57600, TRUE); //FALSE); // Show Protocol
result = LSX_ConnectSimple(IdTango_Z, 1, "COM7", 57600, TRUE); //FALSE); // Show Protocol

result = LSX_GetDLLVersionString(IdTangoXY, text, sizeof(text)-1); // DLL Version

result = LSX_GetTangoVersion(IdTangoXY, text, sizeof(text)-1); // Tango Version of COM11
result = LSX_GetTangoVersion(IdTango_Z, text, sizeof(text)-1); // Tango Version of COM7

result = LSX_Disconnect(IdTangoXY);
result = LSX_Disconnect(IdTango_Z);

result = LSX_FreeLSID(IdTangoXY);
result = LSX_FreeLSID(IdTango_Z);

IdTangoXY = 0;
IdTango_Z = 0;
```

2.3. Integration in Visual C++

A TANGO class is provided for C++, which loads the DLL and all pointers on function calls dynamically. There is no „LS_“ or „LSX_“ prefix in the function names of the Tango object.

Example: `pTango->Calibrate()` instead of `LS_Calibrate()` or `LSX_Calibrate(1)`.

Only one instance of the CTango class should be created, and the Tango-DLL loaded only once.

The required files for a C/C++ Application, `Tango.h` and `Tango.cpp` can be found in the folder: `\Software\API & DLL\DLL Files`.

In some cases, the `Tango.cpp` file must be adapted by replacing `#include "stdafx.h"` with `"pch.h"`.

Required files:

- `Tango_DLL.dll`
- `Tango.h`
- `Tango.cpp` (must also be added to the project's source code files by "add existing file")

Visual C++ example for controlling a TANGO:

```
#include "Tango.h"
...

CTango* pTango;

pTango = new CTango();
...

pTango->ConnectSimple(1, "COM3", 57600, TRUE);
pTango->MoveAbs(30, 50, 70, 0, TRUE);
pTango->Disconnect();
delete pTango;
```

2.4. Integration in Visual Basic

To use the functions of the Tango-DLL, the file `Tango.vb` must be added to the project.

The file `Tango.vb` can be found on the CD: `\Software\ API & DLL\DLL Examples\Visual_Basic\SourceCode`.

Required files: `Tango_DLL.dll` and `Tango.vb`

Visual Basic example for controlling a Tango:

```
Dim return value1 As Integer
Dim return value2 As Integer
Dim return value3 As Integer

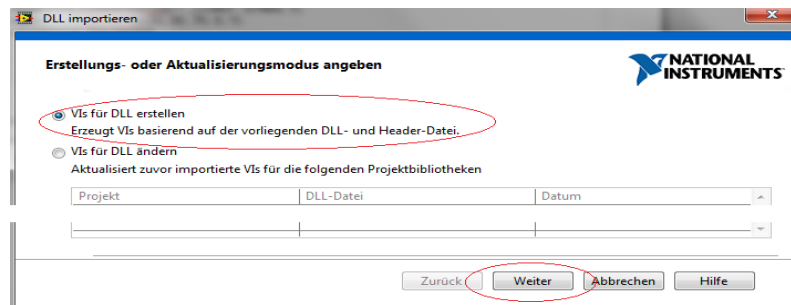
...
Return value1 = LS_ConnectSimple(1, "COM3", 57600, 1)
Return value2 = LS_MoveAbs(30, 50, 70, 0, 1)
Return value3 = LS_Disconnect();
```

2.5. Integration in LabVIEW

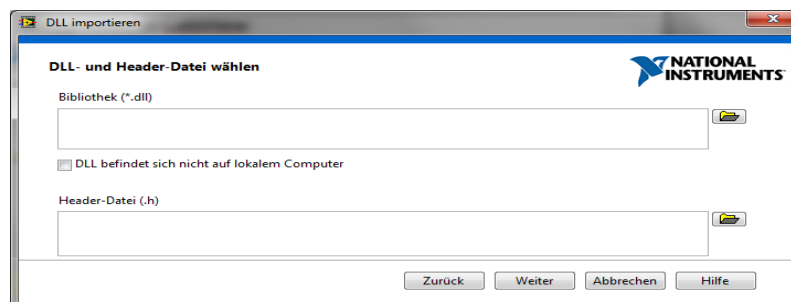
This DLL import description can be used with every LabVIEW Version, which supports DLL import functionality.

To use the TANGO-DLL functions with LabVIEW, the TANGO-DLL has to be imported to LabVIEW. Therefore, follow the steps listed below:

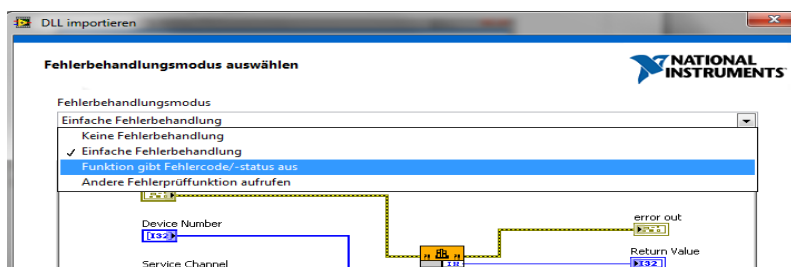
- 1) Start LabVIEW
- 2) In LabVIEW window: Tools → Import → DLL select the first radio button and press next.



- 3) In the 2 corresponding fields select files "TANGO_DLL.dll" and "TANGOLSX_API.h" from CD directory / Software / API&DLL / LabVIEW.

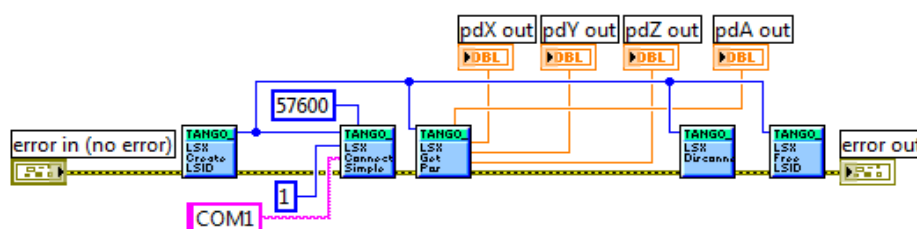


- 4) "Including Paths" in the next window need not to be configured.
- 5) In the next window the included functions of the TANGO_DLL.dll are listed and selectable. It is recommended to select all functions. You may notice, that only half of the functions included in TANGO_DLL.dll are found in the TANGOLSX_API.h which is correct, because all functions exist in "LS_function" and in "LSX_function" notation.
- 6) The TANGOLSX_API.h defines just the "LSX" functions, which should be preferred to use anyway.



- 7) After selecting the path and name for the project library the error handling mode should at least contain a simple error handling or even an error handling with return function of TANGO_DLL.dll included.
- 8) The configuration of the VIs should not be changed and the import process can start.

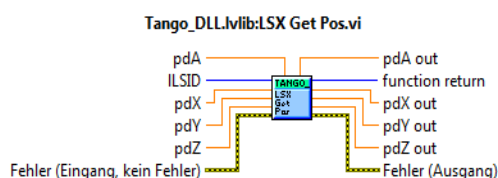
LabVIEW starting example for controlling a TANGO:



This example creates a TANGO-ID number to select the TANGO, which is addressed for the command. A connection to the TANGO is established with virtual COM-Port 1 and Baud-Rate 57600. The actual position of all axes is read out and the TANGO is disconnected. Last step is to free the created TANGO-ID number.

Remark:

“Get” functions defined in TANGO_DLL.dll often have pointers as parameters. These pointers are displayed as inputs and outputs in LabVIEW VIs because LabVIEW is not able to detect whether this pointer is needed as input or output.



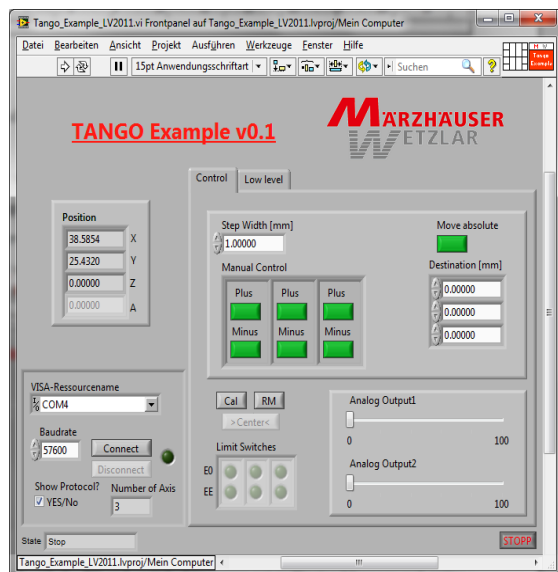
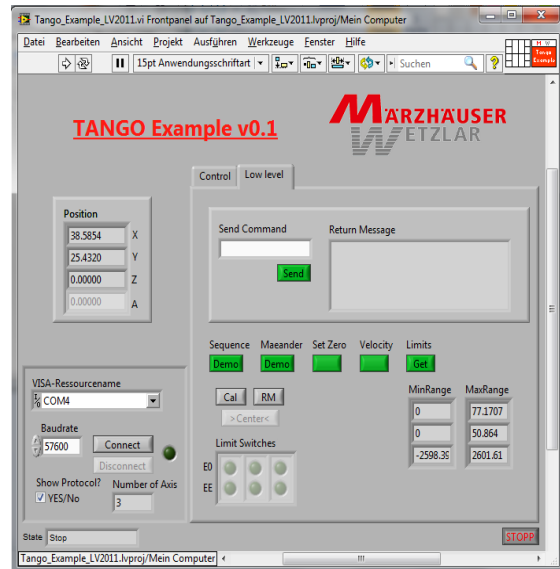
TANGOAPI int TANGOCALL LSX_GetPos (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);

In all such "Get" VIs just connect only required output parameters. It is useless to connect input parameters because they will be ignored anyway and won't have any effect.

Program Example:

Required LabVIEW-Version: LabVIEW 2011 and newer.

An example program of controlling a TANGO via LabVIEW can be found on CD in directory Software/API&DLL/LabVIEW/TANGO_Example_LV2011. This example is implemented in LV2011 and is not compatible with elder versions. It gives an overview of how the TANGO_DLL.dll can be used with LabVIEW and how the TANGO can be controlled with a LabVIEW environment.



This example VI looks for a TANGO (connected with the PC and switched to power on) in Device Manager and writes the corresponding COM-Port in VISA-Ressourcenname as a pre-selection. The default baud-rate is 57600. After selecting the correct COM-Port the user is able to connect to TANGO.

The program gives you an overview over the actual position of all active axes, the values for analog outputs and if a limit switch is active or not (limit switches can only be active, as long as no calibration and range measure drive has been performed).

Functions included in TANGO example VI:

- Calibrate (looks for the backward limit switches)
- Range Measure (looks for the forward limit switches)
- Center Drive (Drives all axes with a limit switch into its middle position → range measure is required as precondition)
- Manual Control (Move a single axis with configured step width)
- Move Absolute (Moves all active axes to an absolute position entered in destination)
- Change value of analog output 1 & 2
- Directly send commands like "?pos" or "?version" (Please be careful, here you have full access to all parameters of the controller)
- Movement demos like "Sequence" or "Meander"
- Set the actual position of all axes to zero
- Check and change "velocity" and "acceleration" of every axis
- Display the range values for limit switches (calibration and range measure is required before)

2.6. Integration in MATLAB

The Tango-DLL can be used in MATLAB. Therefore,

- the Tango_DLL.dll (usually the 64 Bit version)
- and the Tango_DLL.h

must be included in the project (folder).

Example programs can be found on the TANGO-CD:

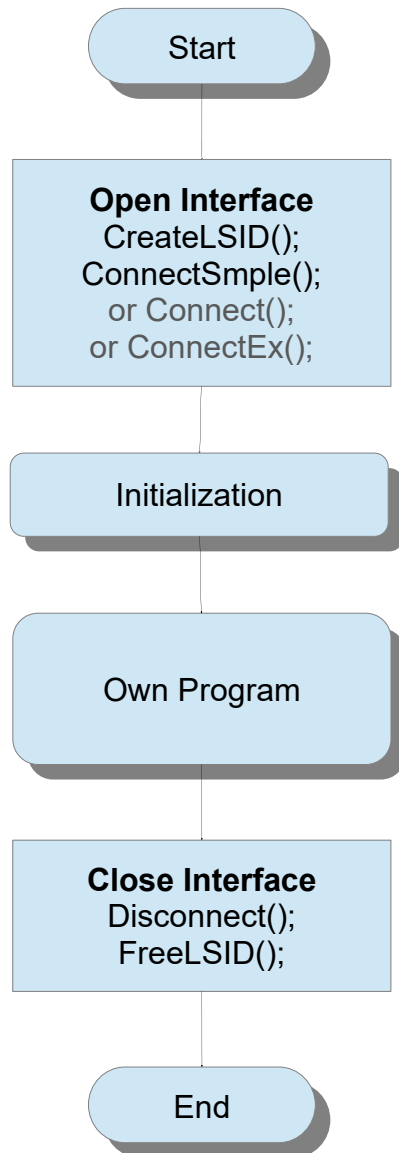
[\Software\ API & DLL \DLL Examples\MatLab\](#)

3. General Information of DLL Usage

The following flow chart shows how to establish and end a DLL communication to the TANGO and is valid for all communication interfaces like RS232, USB, PCI, PCI-E or Ethernet.

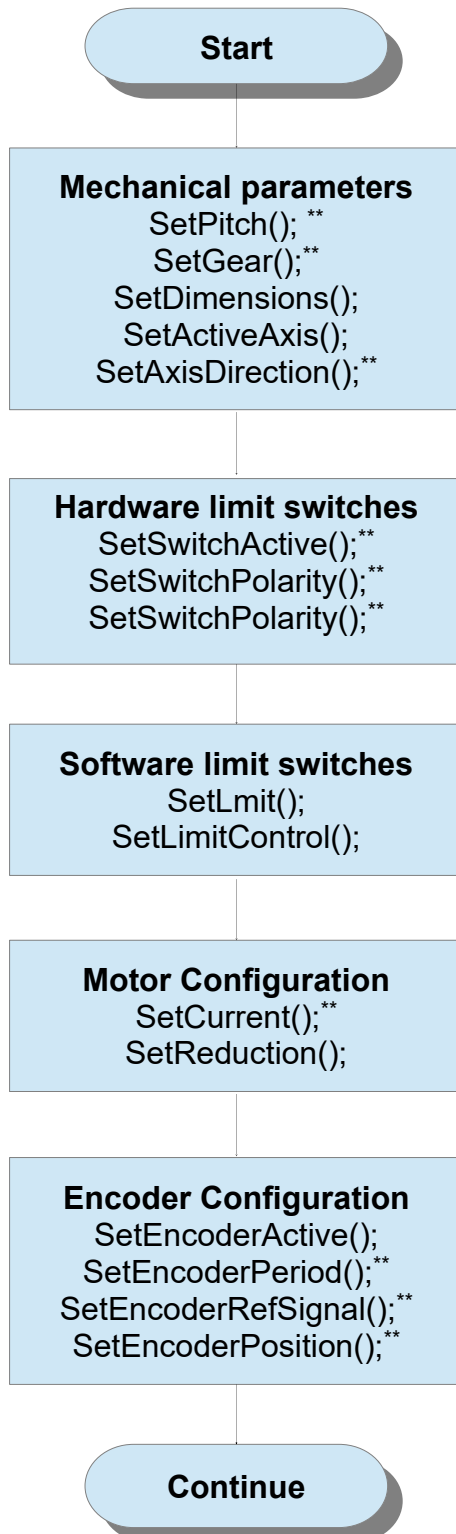
The guideline is equal for all possible programming languages.

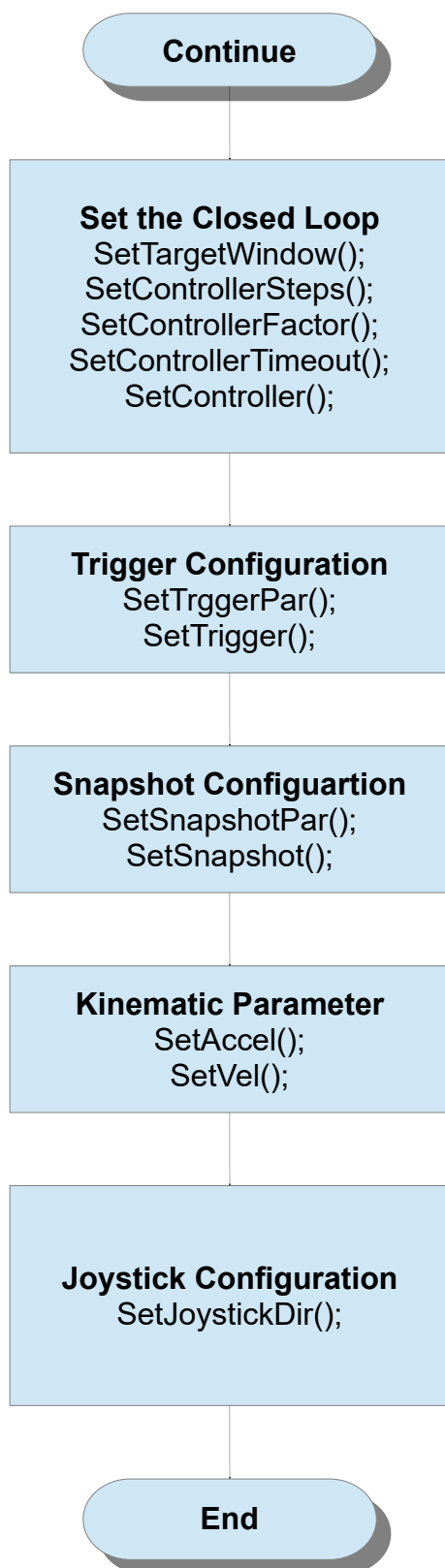
The available DLL functions are listed and described in the following chapters. For further information, the TANGO Instruction Set Documentation can be used. Therefore, the DLL function description also includes the TANGO instruction that stands behind the function call, e.g. SetPitch (!pitch).



3.1. Initialization of Controller

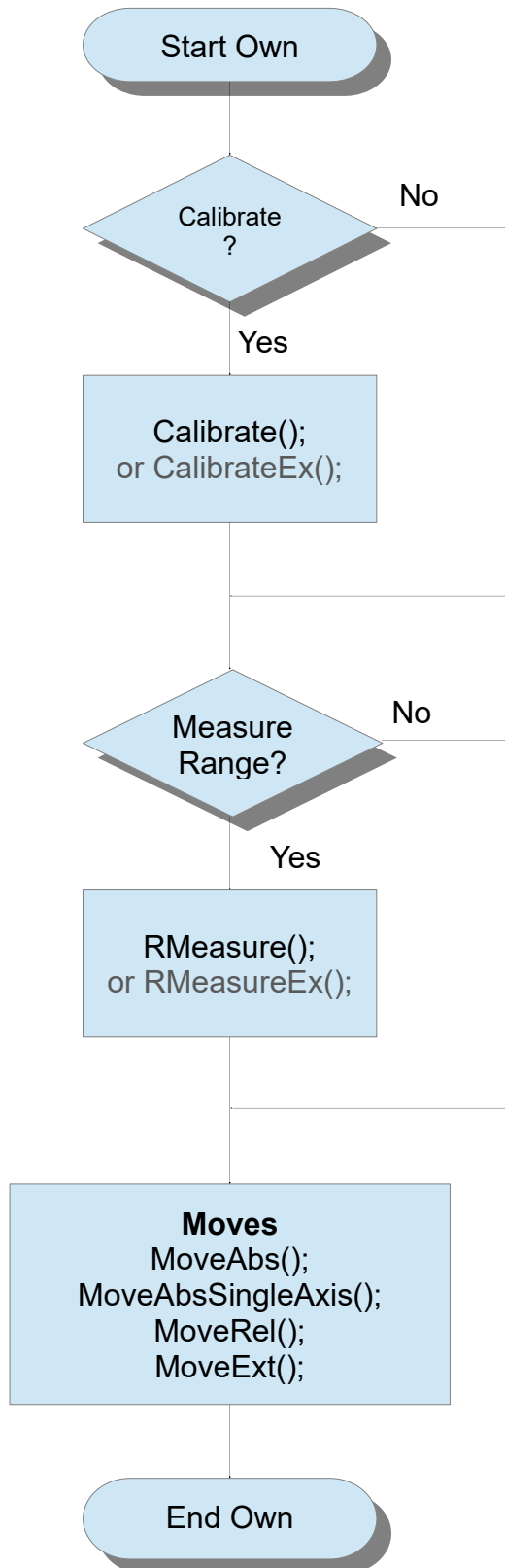
Most Märzhäuser stages and axes are ETS coded. The TANGO then uses the ETS data for initialization. The affected settings become write protected, meaning they cannot and do not need to be modified: **See ****. Märzhäuser products will run “from scratch”, if connected to a TANGO. Individual settings like the measuring unit (SetDimension), velocity, closed loop etc. could be made as shown on the next page.
Note: Mechanics may be damaged if wrong parameters are used. If no Märzhäuser axis is used, please take care the correct settings (**) for the axes are made (e.g. pitch, gear, direction, limit switches, encoder).





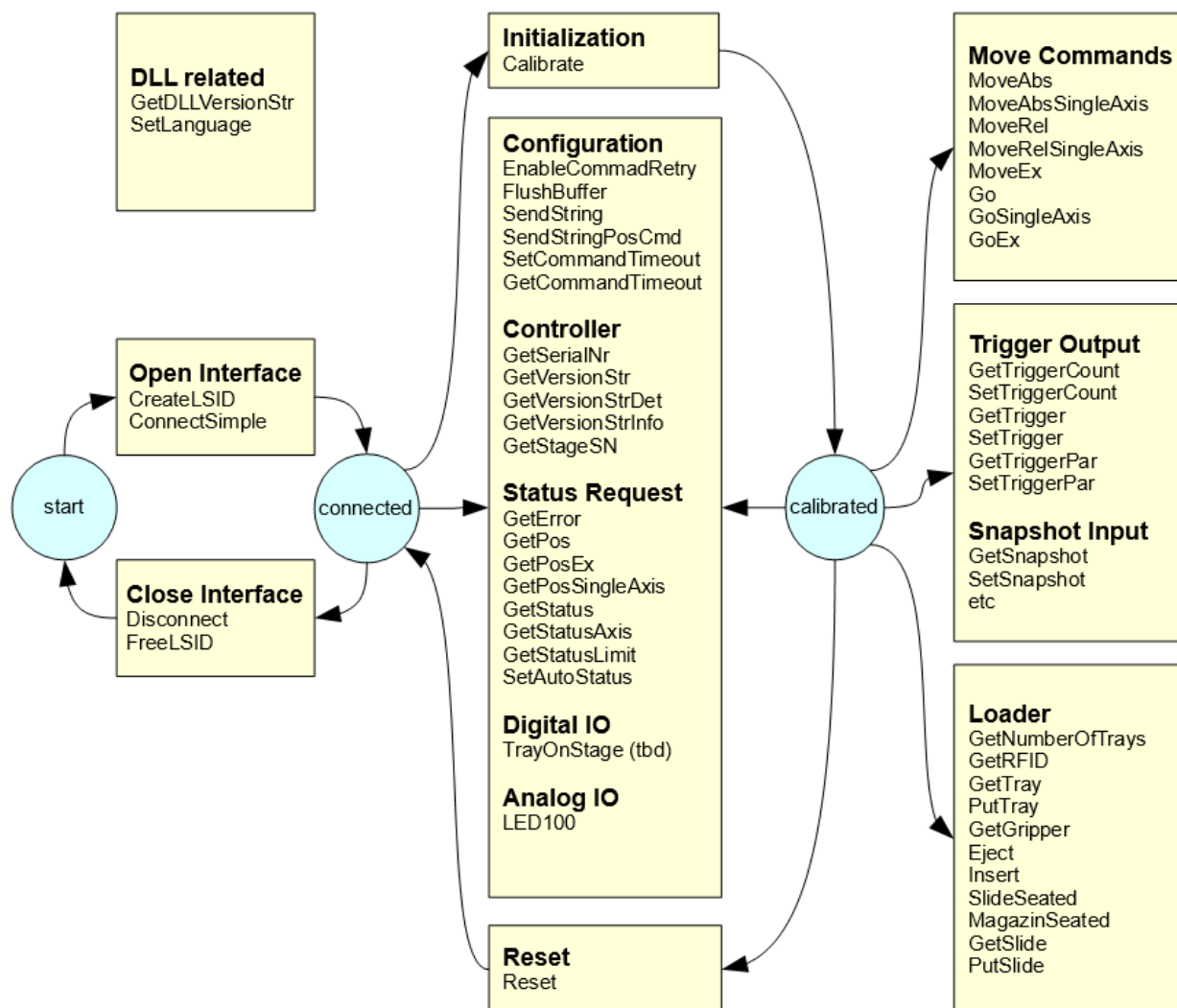
3.2. Own Program Section

In the own program section, the user can program desired functionality of the controller. This includes movements, if desired depending on status of digital I/Os as well as setting trigger signals depending on the position, etc.



3.3. API State Diagram

The API state shows which DLL functions usually require an initialisation as precondition. This means the axes must be moved at least to a reference point. Usually, the lower limit switch is used as reference.



4. Functions

4.1. Quick Reference

DLL Configuration / Interface:

Command	Brief Description	Page
CreateLSID	Creates a Tango-ID number	32
ConnectSimple	Connect to Tango using default controller settings	32
ConnectEx	Connect to Tango using the TLS_ControlInitPar structure	33
LoadConfig	Load configuration from ini file	34
Connect	Connect to Tango using data from LoadConfig ini file	34
SaveConfig	Save configuration to ini file	34
Disconnect	Disconnects Tango Controller from DLL	34
FreeLSID	Releases the previously created Tango ID-Number	35
SetShowProt	Switches protocol window for communication monitoring on/off	35
ClearProtocolWindow	Delete the list content of the protocol window	35
SetLanguage	Set language of protocol window	35
GetCommandTimeout	Read current DLL timeout for read, move and calibration functions	35
SetCommandTimeout	Set DLL timeout for read, move and calibration function calls	35
EnableCommandRetry	Enable/disable repeated command sending in case of comm. errors	36
SetDllNumOfAxes	Manipulate DLL-internal number of Tango axes	36
GetSwapZA	Read if Z and A axes are swapped or not	37
SetSwapZA	Swap Z and A axis assignment for DLL instructions	37
GetDLLVersionString	Read version string of the DLL	37
FlushBuffer	Clears the receive buffer from possibly remaining data fragments	37
SendString	Sends strings to Tango (allows using all commands as ASCII text)	38
SendStringPosCmd	Send an ASCII move command and wait for completion reply	39
SetAbortFlag	Set internal DLL flag to abort a (hanging) communication	39
ReadControlPars	Reserved / future use: Read actual setup parameter from Tango	
SetControlPars	Reserved / future use: Send setup parameter to Tango controller	
SetWriteLogText	Reserved / future use: Activate Data Logging (generate log-file)	

Controller information:

Command	Brief Description	Page
GetSerialNr	Read out the Controller serial number	40
GetTangoVersion	Read out the Tango version information	40
GetVersionStr	Provides the backward-compatible firmware information	40
GetVersionStrDet	Read detailed firmware version information	40
GetVersionStrInfo	Read additional information to current version number	41
GetStageSN	Read stage serial number (if available)	41

Status Requests:

Command	Brief Description	Page
GetError	Provides error state of the Tango (error number as listed in chapter 9.1)	42
GetErrorString	Provides information text about the specified error number	42
GetPos	Read the position of all axes	42
GetPosEx	Read encoder- or motor-positions of all axes	42
GetPosSingleAxis	Read position of one axis	43
GetStatus	Provides current Controller status	43
GetStatusAxis	Read status of one axis	44
GetStatusLimit	Read status of software limits of all axes	44

Command	Brief Description	Page
GetStA	Read the detailed axis state (flags that include almost all states at once)	45
SetAutoStatus	Switches Auto-Status mode (e.g., status reply on/off)	46
GetAutoStatus	Read current Auto-Status mode	46
IsVel	Read actual velocities at which the axes are currently travelling	47
IsVelSingleAxis	Read actual velocity of specified axis	47

Controller Settings:

Command	Brief Description	Page
GetPowerAmplifier	Retrieves actual state of power amplifier	48
SetPowerAmplifier	Set required state of power amplifier (on/off)	48
GetActiveAxes	Retrieve axes state	48
SetActiveAxes	Set axes state	48
GetAxisDirection	Read axis direction	49
SetAxisDirection	Set axis direction	49
GetCalibOffset	Read calibration offset	49
SetCalibOffset	Set calibration offset	49
GetRMOffset	Read range measure offset	50
SetRMOffset	Set range measure offset	50
GetCalibBackSpeed	Read calibration backward speed	50
SetCalibBackSpeed	Set calibration backward speed	50
GetCalibrateDir	Read calibration direction	51
SetCalibrateDir	Set calibration direction	51
GetDimensions	Read the measuring units of all axes	51
SetDimensions	Set measuring units of all axes	52
GetResolution	Get digits after decimal point	52
SetResolution	Set digits after decimal point	53
GetPitch	Read actual spindle pitch	53
SetPitch	Set required spindle pitch	53
GetGear	Read gear ratio	54
SetGear	Set gear ratio	54
GetMotorSteps	Read number of motor steps	54
SetMotorSteps	Set number of motor steps	54
GetMotorCurrent	Read electrical motor current	55
SetMotorCurrent	Set electrical current of motor	55
GetReduction	Read actual current reduction	55
SetReduction	Set current reduction	55
GetCurrentDelay	Provides time delay for motor current reduction	56
SetCurrentDelay	Sets the time delay, after which the motor current is reduced	56
GetSpeedPoti	Read speed potentiometer setting	56
SetSpeedPoti	Set speed potentiometer	56
GetStopPolarity	Read stop polarity	57
SetStopPolarity	Set stop polarity	57
GetVel	Read actual velocity	57
SetVel	Set required velocity	57
SetVelSingleAxis	Set velocity for a single axis	58
GetSecVel	Read actual secure velocity	58
SetSecVel	Set required secure velocity	58
SetSecVelSingleAxis	Set secure velocity for a single axis	58
GetVelFac	Read velocity factor	59
SetVelFac	Set velocity factor	59
GetAccel	Read actual acceleration	60

Command	Brief Description	Page
SetAccel	Set required acceleration	60
SetAccelSingleAxis	Set acceleration for a single axis	60
GetAccelFunction	Read actual acceleration function	60
SetAccelFunction	Set acceleration function trapezoidal or sinusoidal	60
GetStopAccel	Read stop acceleration	61
SetStopAccel	Set stop acceleration	61
GetBlSmoothSingleAxis	Read backlash smoothing mode for open loop systems	61
SetBlSmoothSingleAxis	Set backlash smoothing mode for open loop systems	61
LStepSave	Save all actual parameter in controller	62
SoftwareReset	Reset and reboot the controller	62

Move Commands and Position Management:

Command	Brief Description	Page
Calibrate	Calibrate enabled axes to the CAL limit switches	63
CalibrateEx	Calibrates selected or single axes	63
ClearPos	Sets position values to zero	43
RMeasure	Measure maximum travel range of all axes	63
RMeasureEx	Measure max. travel range of axes selected by the axis bit mask	64
GetDelay	Read the optional delay of vector start	64
SetDelay	Set a delay for move vector start	64
MoveAbs	Absolute positioning - Directs all axes to the specified absolute position	65
MoveAbsSingleAxis	Absolute positioning - Directs one axis to the specified absolute pos.	65
MoveEx	Extended move/move relative command with axis bit mask	66
MoveRel	Relative positioning - Let axes travel by the specified distance	66
MoveRelSingleAxis	Relative positioning - let one axis travel by the specified distance	67
MoveRelShort	Relative positioning (short command, refer to SetDistance)	67
GetDistance	Read distances used by MoveRelShort	67
SetDistance	Set distances for MoveRelShort	67
SetPos	Set current position to the desired value	43
Go	Move command designed to be used with mouse drag events	68
GoSingleAxis	Go for single axis	68
GoEx	Extended Go command	68
GetDigJoySpeed	Read current “digital joystick speed” (of the speed move instruction)	69
SetDigJoySpeed	Start axis move at constant speed (called “digital joystick”)	69
StopAxes	Stops all moving axes	69
StopAxesEx	Stop the specified axis	69
WaitForAxisStop	Function returns as when all axes in bit mask have stopped/arrived	70

HDI Input Devices (Joystick etc.):

Command	Brief Description	Page
SetJoystickOff	Globally disable HDI device (e.g., joystick)	71
SetJoystickOn	Globally enable HDI device (e.g., joystick)	71
GetJoystickDir	Read HDI (joystick) directions	71
SetJoystickDir	Set HDI (joystick) directions (per axis: default, reversed off)	72
GetJoyChangeAxis	Read joystick X-Y assignment swap state	73
JoyChangeAxis	Set joystick X-Y assignment swap state	73
GetJoystick	Read joystick status	72
GetHandWheel	Reserved / future use: Read handwheel status	73
SetHandWheelOff	Reserved / future use: Switches handwheel off	73
SetHandWheelOn	Reserved / future use: Switches handwheel on	74
GetJoystickWindow	Read analog joystick window (not used for digital HDI)	74
SetJoystickWindow	Set analog joystick window (not used for digital HDI)	74
GetHwFactor	Read handwheel factor of all axes	75
SetHwFactor	Set handwheel factor of all axes	75
GetHwFactorSingleAxis	Read handwheel factor of one axis	75
SetHwFactorSingleAxis	Set handwheel factor of one axis	75
GetHwFactorB	Read second handwheel factor of all axes	76
SetHwFactorB	Set second handwheel factor of all axes	76
GetHwFactorBSingleAxis	Read second handwheel factor of one axis	76
SetHwFactorBSingleAxis	Set second handwheel factor of one axis	76
GetZwTravel	Read z-wheel (multifunction wheel) travel distances	77
SetZwTravel	Set z-wheel (multifunction wheel) travel distances	77
GetHdiKeys	Read key states	77
GetKey	Read key states	77
GetKeyLatch	Read and clear latched key states	78
ClearKeyLatch	Clear latched key states of an individual key or of all keys	78
GetHdiSpeedIndex	Read selected HDI speed index (switched by HDI keys or by Set)	79
SetHdiSpeedIndex	Manipulate HDI speed index	79
GetHdiSpeedIndexSingleAxis	Read selected HDI speed index of the specified axis	79
SetHdiSpeedIndexSingleAxis	Manipulate HDI speed index of the specified axis	79

Custom Control Console with Trackball and Joyspeed Keys (BPZ device):

Command	Brief Description	Page
GetBPZ	Read status of control console	80
SetBPZ	Switches control console on / off	80
GetBPZJoyspeed	Read control console joystick speed	80
SetBPZJoyspeed	Set control console joystick speed	80
GetBPZTrackballBackLash	Read control console trackball backlash	81
SetBPZTrackballBackLash	Set control console trackball backlash	81
GetBPZTrackballFactor	Read control console trackball factor	81
SetBPZTrackballFactor	Set control console trackball factor	81

Limit Switches, Hard and Soft Limits:

Command	Brief Description	Page
GetAutoLimitAfterCalibRM	Provides, whether internal software limits are set when calibrating or measuring stage travel range	82
SetAutoLimitAfterCalibRM	Prevents setting internal software limits by calibration or range measure	82
GetLimit	Provides travel range limits of single axes	82
SetLimit	Sets travel range limits of single axes	83
GetLimitControl	Retrieves whether area control is switched on or off	83
SetLimitControl	Switches area control on / off	83
GetSwitchActive	Provides, whether limit switches are active	84
SetSwitchActive	Enable/disable limit switches	84
GetSwitchPolarity	Retrieves polarity of limit switches	84
SetSwitchPolarity	Sets polarity of limit switches	85
GetSwitchType	Retrieves status of pull up or pull-down resistor array (NPN or PNP)	85
SetSwitchType	Set resistor pull-up or pull down to match NPN or PNP switches	85
GetSwitches	Retrieves status of all limit switches	86

Digital and Analog Inputs and Outputs:

Command	Brief Description	Page
GetAnalogInput	Retrieves current level of analogue input signals	87
SetLedBright	Set the brightness of the LED100 illumination OFF/0-100%	87
SetAnalogOutput	Set analogue output voltage	87
GetAnalogOutputMode	Read analog output anamode function	88
SetAnalogOutputMode	Set analog output anamode function	88
SetAuxDigitalOutput	Set individual digital outputs of AUX-I/O connector	88
GetDigitalInputs	Retrieve all digital input pin levels of IO1 interface	89
GetDigitalInputsE	Retrieve digital inputs of IO2 / Multi-IO interface	89
SetDigitalOutput	Set individual digital output of IO1-Module	90
SetDigitalOutputs	Set digital outputs 0-7 of IO1-Module	90
SetDigitalOutputE	Set individual digital output of IO2 / Multi-IO module	90
SetDigitalOutputsE	Set digital outputs 0-7 of IO2 / Multi-IO module	90
SetDigIO_Distance	Reserved / future use: Activate an output, depending on set distance before or after reaching determined position	91
SetDigIO_EmergencyStop	Reserved / future use: Assign Emergency-Stop pin	91
SetDigIO_Off	Reserved / future use: Switch off digital I/O functionality	91
SetDigIO_Polarity	Reserved / future use: Set polarity	92

TVR Clock & Direction Input and Output:

Command	Brief Description	Page
GetTVRMode	Read TVR mode (?tvr)	93
SetTVRMode	Set TVR mode (!tvr)	93
GetFactorTVR	Read AUX-IO TVR clock and direction input factor (?tvrif)	93
SetFactorTVR	Set AUX-IO TVR clock and direction input factor (!tvrif)	93
GetTVROutMode	Reserved / future use	
SetTVROutMode	Reserved / future use	
GetTVROutResolution	Reserved / future use	
SetTVROutResolution	Reserved / future use	
GetTVROutPitch	Reserved / future use	
SetTVROutPitch	Reserved / future use	
GetVelTVRO	Reserved / future use	
SetVelTVRO	Reserved / future use	
GetAccelTVRO	Reserved / future use	
SetAccelTVRO	Reserved / future use	
SetAccelSingleAxisTVRO	Reserved / future use	
GetPosTVRO	Reserved / future use	
SetPosTVRO	Reserved / future use	
MoveAbsTVRO	Reserved / future use	
MoveAbsSingleAxisTVRO	Reserved / future use	
MoveRelTVRO	Reserved / future use	
MoveRelSingleAxisTVRO	Reserved / future use	
GetStatusTVRO	Reserved / future use	
SetTVRInPulse	Reserved / future use	

Encoder Settings:

Command	Brief Description	Page
ClearEncoder	Set encoder TTL-count position to zero	94
GetEncoder	Read all encoder TTL-count positions	94
GetEncoderActive	Read which encoder is activated after calibration (" <i>encmask</i> ")	94
SetEncoderActive	Select encoder to be activated after calibration (" <i>encmask</i> ")	95
GetEncoderMask	Read current activation status of encoders (" <i>enc</i> " command!)	95
SetEncoderMask	Activates / deactivates encoders (" <i>enc</i> ")	95
GetEncoderSingleAxis	Read the main settings of the encoder (type, period, ref-signal)	96
SetEncoderSingleAxis	Set the main settings of the encoder (type, period, ref-signal)	96
GetEncoderPeriod	Read length of encoder signal period	97
SetEncoderPeriod	Set length of encoder period	97
GetEncoderPosition	Provides, whether encoder- or motor-position is displayed	97
SetEncoderPosition	Switches encoder value display on / off	97
GetEncoderRefSignal	Read if reference signal from encoder shall be used when calibrating	98
SetEncoderRefSignal	Set if encoder reference signal shall be used when calibrating	98
GetRefSpeed	Read velocity for searching the encoder reference mark (old cmd.)	98
SetRefSpeed	Set velocity for searching the encoder reference mark (old cmd.)	98

Closed Loop Settings:

Command	Brief Description	Page
GetController	Read controller mode	99
SetController	Set controller mode	99
GetControllerCall	Read controller call interval	99
SetControllerCall	Set controller call time	100
GetControllerFactor	Read setting of closed loop controller factor (old, backward compatib.)	100
SetControllerFactor	Set closed loop controller factor (old, backward compatible)	100
GetControllerFactorSingleAxis	Read setting of closed loop controller factors (recommended function)	101
SetControllerFactorSingleAxis	Set closed loop controller factor (recommended function)	101
GetControllerSteps	Read controller steps	102
SetControllerSteps	Set controller steps	102
GetControllerTimeout	Read setting of closed loop control global timeout	102
SetControllerTimeout	Set closed loop control global timeout	102
GetControllerTWDelaySingleAxis	Read closed loop control time to remain in target window of individual axes	103
SetControllerTWDelaySingleAxis	Set closed loop control time to remain in target window of individual axes	103
GetControllerTWDelay	Old: Read closed loop control time to remain in target window	103
SetControllerTWDelay	Old: Set closed loop control time to remain in target window	103
GetTargetWindow	Read target windows of all axes	104
SetTargetWindow	Set controller target windows	104
SetCtrFastMoveOff	Switch off FastMove function	104
SetCtrFastMoveOn	Switch on FastMove function	104
GetCtrFastMove	Read whether FastMove function is switched on or off	105
GetCtrFastMoveCounter	Read number of executed FastMove functions	105
ClearCtrFastMoveCounter	Resets number of executed FastMove functions to 0	106

Trigger Output:

Command	Brief Description	Page
GetTrigger	Read trigger enable state	107
SetTrigger	Switch trigger on / off	107
GetTriggerMode	Read trigger mode	107
SetTriggerMode	Set trigger mode	107
GetTriggerPar	Read trigger parameters	108
SetTriggerPar	Set trigger parameters	108
GetTrigCount	Read trigger counter value	108
SetTrigCount	Set trigger counter value	108
GetTriggerAxis	Read to which controller axis the trigger unit is assigned	109
SetTriggerAxis	Assign the trigger unit to an axis (for position-dependent trigger)	109
GetTriggerSignalLength	Read the signal length of an active trigger pulse (pulse width in μ s)	109
SetTriggerSignalLength	Set the signal length for an active trigger pulse (pulse width in μ s)	109
GetTriggerDistance	Read the travel distance between trigger signals in modes 0...11	109
SetTriggerDistance	Set the axis position distance between trigger pulses	110
GetTriggerCompensation	Read the trigger compensation value (look forward time in μ s)	110
SetTriggerCompensation	Set the trigger compensation value (look forward time in μ s)	110
GetTriggerEncoder	Read, if the position trigger is based on encoder position	110
SetTriggerEncoder	Set the position trigger to encoder- or to motor position	110
GetTriggerFrequency	Read the trigger frequency of periodic trigger modes 100, 101	111
SetTriggerFrequency	Set the trigger frequency for periodic trigger modes 100, 101	111
GetTriggerOutput	Read the trigger output modes	111
SetTriggerOutput	Set the trigger output mode and 2 nd output functionality	111
Get2ndTriggerDelay	Read precise delay of secondary trigger output signal TAKT_OUT	112
Set2ndTriggerDelay	Set precise delay for secondary trigger output signal TAKT_OUT	112
Get2ndTriggerWidth	Read precise width of secondary trigger output signal TAKT_OUT	112
Set2ndTriggerWidth	Set precise width for secondary trigger output signal TAKT_OUT	112
Get2ndTriggerFrequency	Read precise frequency of secondary trigger output TAKT_OUT	112
Set2ndTriggerFrequency	Set precise frequency for secondary trigger output TAKT_OUT	113
GetTriggerRange	Read the trigger range settings for trigger range mode	113
SetTriggerRange	Set trigger range mode (from position to position, num. of pulses)	113
GetTriggerPositionList	Read the trigger position list (for trigger modes 20,21)	114
SetTriggerPositionList	Set individual trigger positions, trigger mode turns to 20,21 autom.	114
GetTriggerPositionListIndex	Read the current index of the position list (where the trigger is)	114
SetTriggerPositionListIndex	Manipulate the current trigger list index	114
GetTriggerPositionListEntries	Read number of position entries in the trigger list (of mode 20,21)	115
SetTriggerPositionListEntries	Delete the list (0) or reduce the number of list entries	115
GetTriggerLevel	Read the trigger level for trigger modes 20,21	115
SetTriggerLevel	Set the trigger level for trigger modes 20,21 to active high or low	115

Snapshot-Input:

Command	Brief Description	Page
GetSnapshot	Retrieve current on/off status of Snapshot	116
SetSnapshot	Switch Snapshot on / off	116
GetSnapshotMode	Retrieve Snapshot mode	116
SetSnapshotMode	Set Snapshot mode	116
GetSnapshotCount	Read Snapshot counter (number of PosArray entries)	117
SetSnapshotCount	Set Snapshot counter to less entries (truncate/discard the last entries)	117
GetSnapshotFilter	Retrieve input filter debounce delay	117
SetSnapshotFilter	Set input filter debounce delay	117
GetSnapshotPar	Retrieve Snapshot parameters (signal polarity and modes 0,1)	118
SetSnapshotPar	Set Snapshot parameters (signal polarity and modes 0,1)	118
GetSnapshotPos	Retrieve current Snapshot position	118
GetSnapshotPosArray	Retrieve a Snapshot position from the position array	119
SetSnapshotPosArray	Add or change a position of the position array	119
ClearSnapshotPosArray	Delete all position array entries	119
GetSnapshotIndex	Read Snapshot index (current pointer position in array (0...n-1))	120
SetSnapshotIndex	Set Snapshot index (current pointer position in array (0...n-1))	120

SlideExpress Interface:

Command	Brief Description	Page
Eject	Eject magazines	122
Insert	Magazines are inserted and tested if seated on which slides are present.	122
SlideSeated	Query if slide is present (seated) or not or unknown.	122
MagazinSeated	Query if magazine is present (seated) or not or unknown.	122
GetGripper	Set input filterQuery gripper status information. Returns status of gripper 1 and 2.	123
SetGripper	Set gripper status information. (Possibly useful for slide sorting tasks)	123
GetClipType	Read the clip type that is currently in the gripper	123
GetSlide	Get slide(s) from addressed position in magazine or priority handler	124
PutSlide	Put slide(s) back to addressed position in magazine or priority handler	124
GetPrioHandlerPosition	Query actual priority handler position.	124
SetPrioHandlerPosition	Enables user to shift priority handler to required position. Handler is locked at destination or after 30s timeout	124

TrayExpress Interface:

Command	Brief Description	Page
Eject	Eject magazine	125
Insert	Magazine is inserted and tested if seated and which trays are present	125
GetGripper	Retrieve gripper status, e.g. which tray is gripped	126
SetGripper	Set gripper status information	126
GetTray	Get tray from a magazine slot and put it e.g. under the microscope	126
PutTray	Put tray back to a magazine slot	126
GetRFID	Retrieve RFID of addressed tray (if properly seated in magazine)	127
GetNumberOfSlots	Retrieve max available number of slots in magazine	127
GetNumberOfMagazines	Retrieve max available number of magazines	127

Additional Handling System Functions:

Command	Brief Description	Page
GetLoaderType	Read configured type of handling system	128
GetNumberOfRows	Read available number of slide or tray slots	128
GetNumberOfColumns	Read available number of slides per tray or row	128
GetTraySN	Read serial number of the tray	128
GetTrayType	Read type of tray	129
SetTrayType	Set type of tray	129
SetCabinLED	Switch cabin LED on or off	130
GetCabinLED	Read state of cabin LED	130
SetLabelLED	Switch barcode LED on or off	130
GetLabelLED	Read state of barcode LED	130

xPos Module Functions:

Command	Brief Description	Page
Xpos3GetPosSingleAxis	Read axis position from xPos module	131
Xpos3SetPosSingleAxis	Set axis position of xPos module	131
Xpos3MoveAbsSingleAxis	Move xPos module axis to a position	131
Xpos3MoveRelSingleAxis	Move xPos axis by relative distance	131

4.2. DLL Configuration / Interface

CreateLSID	
Description	When using LSX functions, CreateLSID must always be the first command before establishing a new connection. CreateLSID requests a unique ID from the DLL, which must be used as first parameter of all LSX functions to identify the connection. This way, up to eight individual TANGO controllers can be accessed through one DLL. After disconnecting, a call to FreeLSID frees the occupied ID for further connections. (When using the LS functions, only one TANGO can be connected and the LSID parameter is neither required nor available in the LS function calls)
C++	<code>int LSX_CreateLSID(int *pLSID);</code>
Parameters	LSID: Returns a new Tango ID-Number (1 to 8) after calling CreateLSID. If 0 is returned, then no new ID could be created (all IDs already occupied). The returned ID must be used for all subsequent commands belonging to this device.
Example	<code>int Tango1, Tango2; pTango->CreateLSID(&Tango1); // create ID for first Tango pTango->CreateLSID(&Tango2); // create ID for second Tango</code>

ConnectSimple	
Description	The default way to connect to a Tango (other options to connect are: ConnectEx or LoadConfig+Connect). In case of LSX functions, CreateLSID() must be called before connecting. The DLL can connect to RS232, USB and PCI/PCI-E ports via a COM port interface either by specifying the COM port number and using InterfaceType = 1 or by auto-connect to the first TANGO USB/PCI/PCI-E interface the DLL finds on the computer: Then use InterfaceType = -1 The DLL can also connect to a TANGO Desktop HE via Ethernet (IPv4). - Therefore, instead of the ComName must contain the TANGO IP-Address xxx.xxx.xxx.xxx instead of COMx and the InterfaceType must be 6 instead of 1. - In the special case of "Bootloader Connect" (InterfaceType 5, the DLL decides automatically between the interfaces COM or Ethernet.
C++	<code>int LSX_ConnectSimple(int lLSID, int lAnInterfaceType, char *pcAComName, int lABaudRate, BOOL bAShowProt);</code>
Parameters	AnInterfaceType: Interface type = 1 (always 1 for RS232, PCI, PCI-E and USB) Interface type = -1 (connects the DLL to the first USB or PCI TANGO found on the computer, without specifying a COM port) Interface type = 6 (connects the DLL via Ethernet) AComName: Name of COM-Interface, e.g. "COM2" ABaudRate: e.g. 57600 Baud (only used for RS232, else don't care) AShowProt: Show (TRUE) or hide (FALSE) the DLL protocol window
Example	<code>pTango->ConnectSimple(1, 1, "COM2", 57600, TRUE); // Connect to COM2 pTango->ConnectSimple(1, -1, NULL, 57600, TRUE); // Auto-connect with the first found USB or PCI TANGO in the system pTango->ConnectSimple(1, 6, "192.168.1.162", 57600, TRUE); // Connect to IPv4</code>

ConnectEx	
Description	Another way to connect to a Tango ConnectEx requires the “TLS_ControlInitPar” parameter structure to connect, as defined in “Tango.h”. This structure must contain the required connection setup. (other options to connect are: ConnectSimple or LoadConfig+Connect). Hint: Use parameter ID given from command CreateLSID(), when LSX commands shall be used, CreateLSID() is not required for the LS commands. Without connection setup, connection is not possible. The DLL can also connect to a TANGO Desktop HE via Ethernet (IPv4). - Therefore, instead of the ComName must contain the TANGO IP-Address xxx.xxx.xxx.xxx instead of COMx and the InterfaceType must be 6 instead of 1. - In the special case of “Bootloader Connect” (InterfaceType 5”, the DLL decides automatically between the interfaces COM or Ethernet.
C++	<code>int LSX_ConnectEx(int llsid, TLS_ControlInitPar *pAControlInitPar);</code>
Parameters	AControlInitPar: Structure with baud rate, port, protocol etc. information
Example	<code>pTango->ConnectEx (1, &ControlInitPar);</code>

LoadConfig	
Description	Load configuration data file (SwitchBoard “.ini” file) If the file was not found or has invalid content, the function returns error 4001.
C++	<code>int LSX_LoadConfig (int ILSID, char *pcFileName);</code>
Parameters	pcFileName → file name to be used to read data from. The ini file data structure should be generated from SwitchBoard (ASCII text).
Example	<pre> REQUIRED(LSX_CreateLSID(&g_LSID) == 0); char* inifile = "C:\\Users\\me\\Desktop\\mytest.ini"; REQUIRED(LSX_LoadConfig(g_LSID, inifile) == 0); REQUIRED(LSX_Connect(g_LSID) == 0); //LSX_SetShowProt(g_LSID, TRUE); //overwritten from ini file REQUIRED(LSX_SetLanguage(g_LSID, "germ") == 0); </pre>

Connect	
Description	The 3 rd way to connect to a Tango (other options to connect are: ConnectSimple or ConnectEx). Connect using previously loaded configuration data. The COM Port is taken from the loaded ini file and the ini setup parameters are sent to the Tango controller after connecting.
C++	<code>int LSX_Connect (int ILSID);</code>
Parameters	-
Example	<pre> pTango->CreateLSID(&Tango1); // create ID for first Tango pTango->LoadConfig (Tango1, inifile_name_string); pTango->Connect (Tango1); // Connect with the ini file informations of LoadConfig </pre>

SaveConfig	
Description	Save configuration data to certain file. As of TANGO DLL 1.403 not supported yet.
C++	<code>int LSX_SaveConfig (int ILSID, char *pcFileName);</code>
Parameters	pcFileName → file name to be used to write data to. Data is simple ASCII text only.
Example	<code>pTango->SaveConfig (ILSID, pcFileName);</code>

Disconnect	
Description	Disconnect from Tango. After calling this function, commands can no longer be sent to the Tango Controller. This function should be called just before closing the program.
C++	<code>int LSX_Disconnect(int ILSID);</code>
Parameters	-
Example	<pre> pTango->Disconnect(1); // Disconnect the controller number 1 (LSID 1) pTango->FreeLSID(1); // And free the LSID (if LSX_+LSID is used, not for LS_) </pre>

FreeLSID	
Description	Free a Tango ID-Number that was created by CreateLSID. FreeLSID should only be called after executing Disconnect. The LSID is used as an additional parameter in Tango-DLL LSX commands to select the Tango to which command is aimed at from a range of connected Tangos.
C++	<code>int LSX_FreeLSID(int lLSID);</code>
Parameters	LSID: The given Tango ID-Number, which is to be set free. The ID must not be used after FreeLSID has been executed.
Example	<code>int Tango1;</code> <code>pTango->CreateLSID(&Tango1);</code> <code>pTango->ConnectSimple(Tango1, ...);</code> <code>...</code> <code>pTango->Disconnect(Tango1);</code> <code>pTango->FreeLSID(Tango1);</code>

SetShowProt	
Description	Switches the interface protocol window on / off.
C++	<code>int LSX_SetShowProt (int lLSID, BOOL bShowProt);</code>
Parameters	ShowProt: TRUE = show Interface Protocol window FALSE = hide Interface Protocol window
Example	<code>pTango->SetShowProt(1, TRUE);</code> <i>// Show interface protocol for Tango1, in case not already visible</i>

ClearProtocolWindow	
Description	Clears the content of the protocol window.
C++	<code>int LSX_ClearProtocolWindow (int lLSID);</code>
Parameters	-
Example	<code>pTango->ClearProtocolWindow (1);</code> <i>// Delete protocol window list content of Tango1</i>

SetLanguage	
Description	Set language of protocol window
C++	<code>int LSX_SaveConfig (int lLSID, char *pcPLN);</code>
Parameters	pcPLN: if string contains “germ” or “deut” language is switched to german if string contains “fren” or “fran” language is switched to french all other strings switch to english
Example	<code>pTango->SaveConfig (1, "french");</code> <i>// Switch protocol 1 language to french</i>

GetCommandTimeout	
Description	Read current DLL timeout for read, move and calibration
C++	<code>int LSX_GetCommandTimeout (int ILSID, int *toRead, int *toMove, int *toCal);</code>
Parameters	toRead: DLL standard timeout to get a reply from the controller (default 1000 ms) toMove: DLL timeout for axes moves in [ms] toCal: DLL timeout for calibration in [ms]
Example	<code>pTango->GetCommandTimeout(1, &tR, &tM, &tC);</code>

SetCommandTimeout	
Description	Set DLL timeout for read, move and calibration
C++	<code>int LSX_SetCommandTimeout (int ILSID, int toRead, int toMove, int toCal);</code>
Parameters	toRead: do not modify DLL standard timeout default 1000 ms toMove: timeout for move in [ms] (consider speed and acceleration) toCal: timeout for calibration in [ms] (consider axes length, speed and acceleration)
Example	<code>pTango->SetCommandTimeout(1, tR, tM, tC);</code>

EnableCommandRetry	
Description	This function enables/disables repeated sending of commands in case of errors (Default = enabled).
C++	<code>int LSX_EnableCommandRetry (int ILSID, BOOL bAValue);</code>
Parameters	AValue: TRUE → in case of errors, the Tango DLL repeats sending certain command (especially in case of WaitForAxisStop) FALSE → disable repeated sending
Example	<code>pTango->EnableCommandRetry(1, FALSE);</code>

SetDllNumOfAxes	
Description	Manipulate the DLL-internal information about the available Tango axes. E.g., instructions sent from the DLL to the Tango will be limited to the corresponding amount of axis parameters. It is not recommended to manipulate DLL-internal states except for good reasons.
C++	<code>int LSX_SetDllNumOfAxes (int ILSID, int INumOfAxes);</code>
Parameters	NumOfAxes: 1, 2, 3 or 4
Example	<code>pTango->SetDllNumOfAxes(1, 3); // Force the DLL to use 3 axes</code>

GetSwapZA	
Description	Read if the axis Z and A are swapped by the Tango DLL (Z parameters redirected to A and vice versa).
C++	<code>int LSX_GetSwapZA (int ILSID, BOOL *pbValue);</code>
Parameters	Value: TRUE = Z and A are swapped, FALSE = not swapped (default)
Example	<code>pTango->GetSwapZA (1, &bSwapped); // Read the Tango DLL Z-A swap setting</code>

SetSwapZA	
Description	Set if the axis Z and A should be swapped by the Tango DLL (Z parameters redirected to A and vice versa) or not.
C++	<code>int LSX_SetSwapZA (int ILSID, BOOL bValue);</code>
Parameters	Value: TRUE = swap Z and A, FALSE = no swapped (default)
Example	<code>pTango->GetSwapZA (1, &bSwapped); // Read the Tango DLL Z-A swap setting</code>

FlushBuffer	
Description	Clear communication input buffer. Can be used in error situations to remove no longer needed feedback messages from the input buffer.
C++	<code>int LSX_FlushBuffer (int ILSID, int lAValue);</code>
Parameters	AValue: not used, can be set to 0
Example	<code>pTango->FlushBuffer(1, 0);</code>

GetDLLVersionString	
Description	Get DLL version string
C++	<code>int LSX_GetDLLVesionString (int ILSID, char *pcVers, int lMaxLen);</code>
Parameters	pcVers → Buffer, containing return message from DLL lMaxLen → Limits the max. number of characters to be copied into buffer
Example	<code>char cVersionString[128]; pTango->GetDLLVesionString (ILSID, cVersionString, 127); // Example reply: "DLL64 Version 1.403, Oct 12, 2021, 12:34:33"</code>

SendString	
Description	Send an ASCII string to the Tango or send and receive or receive only.
C++	<pre> int LSX_SendString (int ILSID, char *pcStr, char *pcRet, int lMaxLen, BOOL bReadLine, int lTimeout); </pre>
Parameters	Str → Zero-terminated string, which is to be sent to controller. String must end with a carriage return (\r). Ret → Buffer, containing return message from Tango, in case ReadLine = TRUE or also ZERO (NULL), in case ReadLine = FALSE; MaxLen → Max. amount of characters allowed to be copied into buffer ReadLine → TRUE = read return message from Tango FALSE = don't wait for return message Timeout → Max. waiting period for return message [ms]
Example	<pre> pTango->SendString(1, „?version\r“, pcVer, 256, TRUE, 1000); // Read version number, allow max. 256 characters, 1 Second Timeout pTango->SendString(1, „!baud 115200\r“, NULL, 0, FALSE, 0); // set max. baud rate for RS232 pTango->SendString(1, NULL, pcVer, 256, TRUE, 250); // just read, wait 250ms </pre>

SendStringPosCmd	
Description	Send move command to Tango as a string and wait for return message. The parameters allow options of wait / do not wait, return the reply or not required (the reply string might be of interest in case of @, E, T, A, D, S ... was returned by the move) With ReadLine TRUE, the function always waits until a reply was sent from the Tango. And if a return buffer is specified, the reply is copied into this buffer, else not. If autostatus is set to 0, no reply will be sent by the Tango, and the wait will timeout.
C++	<pre>int LSX_SendStringPosCmd (int ILSID, char *pcStr, char *pcRet, int lMaxLen, BOOL bReadLine, int lTimeOut);</pre>
Parameters	Str → Zero-terminated ASCII string, which is to be sent to the controller The last character must be an “\r” (the CR character) Ret → Buffer, containing return message from Tango, in case ReadLine = TRUE or ZERO (NULL), in case the autostatus “@@@” reply string is not required MaxLen → Max. amount of characters allowed copied into pcRet buffer ReadLine → TRUE = wait for return message (@@@ reply) from Tango FALSE = don’t wait for return message / move complete TimeOut → Max. waiting period for return message [ms]
Example	<pre>char reply_text[16]; pTango->SetAutoStatus(1, 1); // Ensure the Tango will reply on the move (@@@ reply) pTango->SendStringPosCmd(1, “!moa 1.5 8.3\r”, &reply_text[0], 16, TRUE, 10000); pTango->SendStringPosCmd(1, “!mor z -0.05\r”, NULL, 0, TRUE, 2000); pTango->SendStringPosCmd(1, “!cal x\r”, &reply_text[0], 16, TRUE, 60000);</pre>

SetAbortFlag	
Description	Set flag so that communication with Tango is cut off. A function, which when calling LSX_SetAbortFlag is still waiting for return message from controller (e.g. drive commands), then returns with an error message. The use of this function especially makes sense for programs with message processing routines or with multiple threads, in case, for example, a drive movement shall be stopped quickly.
C++	<pre>int LSX_SetAbortFlag (int ILSID);</pre>
Parameters	-
Example	<pre>pTango->SetAbortFlag(1); pTango->StopAxes(1); // closes communication with Tango and sends stop command for all axes</pre>

4.3. Controller Information

GetSerialNr	
Description	Reads out the TANGO serial number (?readsn).
C++	<code>int LSX_GetSerialNr (int ILSID, char *pcSerialNr, int IMaxLen);</code>
Parameters	SerialNr: Pointer to a buffer, in which the serial number will be returned MaxLen: Limits the max. number of characters to be copied into buffer
Example	<pre>char TangoSN[16]; // The SN consists of 9 ASCII characters (0-9, A-Z) pTango->GetSerialNr(1, TangoSN, 15); // Example reply: 190103001 // 190103001 = 19 = YY, 01 = WW, 0 = Controller Type, 3 = 3Axes max., 001 Index</pre>

GetTangoVersion	
Description	Get TANGO version string (?version). Read the TANGO Controller Version information as ASCII text.
C++	<code>int LSX_GetTangoVesion (int ILSID, char *pcVers, int IMaxLen);</code>
Parameters	pcVers: Pointer to the char buffer, in which the TANGO version text is returned IMaxLen: Limits the max. number of characters to be copied into buffer
Example	<pre>char cVersionString[128]; pTango->GetTangoVesion (ILSID, cVersionString, 127); // Example reply: "TANGO-Desktop, Version 1.73, Mar 11 2021 , 13:28:26"</pre>

GetVersionStr	
Description	Returns the backward-compatible firmware, axis and motorcurrent information (?ver).
C++	<code>int LSX_GetVersionStr (int ILSID, char *pcVers, int IMaxLen);</code>
Parameters	pcVers: Pointer to the char buffer, in which the TANGO version text is returned IMaxLen: Limits the max. number of characters to be copied into buffer
Example	<code>pTango->GetVersionStr(1, pcVers, 64); // retrieve compatible version information</code>

GetVersionStrDet	
Description	Retrieves detailed configuration of Tango (?det) as decimal ASCII digits.
C++	<code>int LSX_GetVersionStrDet (int ILSID, char *pcVersDet, int IMaxLen);</code>
Parameters	VersDet: Pointer to a buffer, in which the string will be returned IMaxLen: Limits the max. number of characters to be copied into buffer
Example	<code>pTango->GetVersionStrDet(1, pcVersDet, 16); // retrieve detailed configuration</code>

GetVersionStrInfo	
Description	Provides optional internal information on the controller version (?iver).
C++	<code>int LSX_GetVersionStrInfo (int ILSID, char *pcVersInfo, int lMaxLen);</code>
Parameters	VersInfo: Pointer to a buffer, in which the string will be returned lMaxLen: Limits the max. number of characters to be copied into buffer
Example	<code>pTango->GetVersionStrInfo(1, pcVersInfo, 16);</code>

GetStageSN	
Description	Provides optional internal information on the stage serial number (?stagesn -1).
C++	<code>int LSX_GetStageSN (int ILSID, char *pcSN, int lMaxLen);</code>
Parameters	pcSN: Pointer to a buffer, in which the string will be returned lMaxLen: Limits the max. number of characters to be copied into buffer
Example	<code>pTango->GetVersionStrInfo(1, pcSN, 16);</code>

4.4. Status Requests

GetError	
Description	Readout the current error state of the controller (?err).
C++	<code>int LSX_GetError (int ILSID, int *pLErrorCode);</code>
Parameters	ErrorCode: Error number (as described in chapter 9.1)
Example	<code>pTango->GetError(1, &ErrorCode);</code>

GetErrorString	
Description	Provides ASCII text explanation of the here specified error number (?help). TANGO errors 0...255 can be requested as well as DLL error codes >= 4001.
C++	<code>int LSX_GetErrorString (int ILSID, int lError, char *pcErrorString, int lMaxLen);</code>
Parameters	lError: Error number of which the explanation shall be returned 0..255, 4001... pcErrorString: Character array pointer to receive the text MaxLen: Limits the max. number of characters to be copied into the array
Example	<code>pTango->GetErrorString(1, 29, &text_array[0], 64); // read explanation of error 29</code>

GetPos	
Description	Retrieves current position of all axes (?pos). Also refer to SetEncoderPosition .
C++	<code>int LSX_GetPos (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	X, Y, Z, A: Axis position values
Example	<code>pTango->GetPos(1, &X, &Y, &Z, &A);</code>

GetPosEx	
Description	Retrieves encoder or motor positions of all axes (!encpos + ?pos). If an axis is not available, 0.0 is returned.
C++	<code>int LSX_GetPosEx (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA, BOOL bEncoder);</code>
Parameters	X, Y, Z, A: Position parameter Encoder = TRUE → Provide encoder positions (if encoder connected, else motor pos.) = FALSE → Provide motor position values
Example	<code>pTango->GetPosEx(1, &X, &Y, &Z, &A, TRUE);</code>

GetPosSingleAxis	
Description	Retrieves current position of a single axis (?pos). If the motor or encoder position is returned depends on SetEncoderPosition . If the axis is not available, 0.0 is returned.
C++	<code>int LSX_GetPosSingleAxis (int ILSID, int lAxis, double *pdPos);</code>
Parameters	Axis: Axis of which the position parameters shall be retrieved from, 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) Pos: Positions
Example	<code>pTango->GetPosSingleAxis(1, 2, &Pos); // retrieves position of Y-Axis</code>

SetPos	
Description	Set position (!pos).
C++	<code>int LSX_SetPos (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: Min- / max. range of travel, command depends on dimension
Example	<code>pTango->SetPos(1, 10, 10, 0, 0); // Set current position to this values</code>

ClearPos	
Description	Sets current position and internal position counter to 0 (!clearpos). This function is needed for endless axes, as controller can only process $\pm 1,000$ motor revolutions within its parameters. This instruction will be ignored for axes with encoders.
C++	<code>int LSX_ClearPos (int ILSID, int IFlags);</code>
Parameters	Flags: Bit mask Bit 0=X, Bit 1=Y, Bit 2=Z, Bit 3=A Bit 0 = 1 → position of X-Axis is set to zero. Bit 1 = 0 → function is not executed for Y-Axis.

GetStatus	
Description	Provides current status of the controller (?status).
C++	<code>int LSX_GetStatus (int ILSID, char *pcStat, int IMaxLen);</code>
Parameters	Stat: Pointer to a buffer, in which the status string will be returned MaxLen: Limits the max. number of characters to be copied into buffer
Example	<code>pTango->GetStatus(1, &Stat, 16);</code>

GetStatusAxis

Description	Provides current status of the axes (?statusaxis). Only the character M indicates a moving axis. At all other characters, the axis is stopped.
C++	<code>int LSX_GetStatusAxis (int ILSID, char *pcStatusAxisStr, int IMaxLen);</code>
Parameters	StatusAxisStr: Pointer to a buffer in which status string will be returned MaxLen: Limits the max. number of characters to be copied into buffer e.g.: @ -- M -- J -- C -- S -- A -- D -- U -- T @ = Axis stands still M = Axis is in motion - = Axis is not enabled J = Joystick switched on C = Axis is in closed loop A = Return message after calibration (cal) E = Error when calibrating (limit switch not cleared correctly) D = Return message after measuring stage travel range (rm) U = Setup mode T = Timeout
Example	<code>pTango->GetStatusAxis(1, &StatusAxisStr, 16);</code>

GetStatusLimit

Description	Provides current status of software limits of each axis (?statuslimit).
C++	<code>int LSX_GetStatusLimit (int ILSID, char *pcLimit, int IMaxLen);</code>
Parameters	Limit: Pointer to a buffer, in which the status of the axes will be returned e.g.: AAA-DD-- LLLL A = Axis has been calibrated D = Stage travel range has been measured (rm) L = Software limit has been set - = Software limit remains unchanged MaxLen: The max. number of characters that can be copied into the buffer
Example	<code>pTango->GetStatusLimit(1, &Limit, 32);</code>

GetStA	
Description	Read the detailed axis states (sta). Returns all states in which the axis can be as a set of 32 individual flags. For more details, refer to the sta description of the TANGO Instruction Set and the corresponding TANGO firmware version. <pre> 00000001 !axis is set to 0 or 1 (motor current is on) 00000002 !axis is set to 1 (enabled) 00000004 motor power amplifier is on 00000008 motor power amplifier error 00000010 corresponds to statusaxis 'M' state of the axis 00000020 the axis travels due to HDI deflection (e.g.joystick) 00000040 cal is running 00000080 rm is running 00000100 cal already executed ('A' in statuslimit) 00000200 rm already executed ('D' in statuslimit) 00000400 lower limit switch E0 actuated (1 in readsw) 00000800 upper limit switch EE actuated (1 in readsw) 00001000 axis move waits for snapshot signal 00002000 calrequired prevents axis move, no cal/rm yet 00004000 1D position correction active 00008000 2D position correction active (@ Z reply: 2D+z) 00010000 encoder is active (?enc) 00020000 encoder was activated, even if currently disabled 00040000 reserved 00080000 encoder error (encerr) 00100000 closed loop is on 00200000 closed loop is active, regulating 00400000 closed loop is in target window (set by twi) 00800000 closed loop is in lock-in range (set by ctrs) 01000000 stop signal is active 02000000 HDI is enabled for this axis (joy+joydir) 04000000 Thermal compensation temperature is updatet, no error 04000000 Thermal compensation is applied, was activated by cal 10000000 Macro execution (macro is running) 2nd gen. TANGOs 20000000 Cal Learn data for encoder (callrnpos) Firmw. ≥ 1.782 40000000 Cal Learn data for motor (callrnposmot) Fw. ≥ 1.782 80000000 reserved Example: sta x returns a hex number for the x axis state flags: 02030307 --> axis is not traveling, no closed loop, no errors +- axis is on (!axis x 1, !pal) +--- cal and rm are executed +----- encoder is active (and was/is active) +----- HDI is enabled (joy+joydir) </pre>
C++	<code>int LSX_GetStA (int ILSID, int *plStaX, int *plStaY, int *plStaZ, int *plStaA);</code>
Parameters	<i>plSta</i> : Pointers to integer, in which the state flags of the axes are returned
Example	<code>pTango->GetStA(1, &lStaX, &lStaY, &lStaZ, &lStaA);</code>

SetAutoStatus	
Description	Switches Auto-Status on/off (!autostatus). Please note: AutoStatus mode should not be changed as Tango DLL sets correct mode for travel commands etc. (except the SendStringPosCmd, where AutoStatus 1 is required, if the wait for reply option is used).
C++	<code>int LSX_SetAutoStatus (int ILSID, int IValue);</code>
Parameter	Value: AutoStatus mode: 0 → Controller sends no status 1 → Controller automatically sends „Position reached“ messages (the @@@, default) 2 → Controller automatically sends „Position reached“ and status messages This mode might cause false behaviour of the TANGO DLL. 3 → There is only one carriage return sent for „Position reached“
Example	<code>pTango->SetAutoStatus(1, 1);</code>

GetAutoStatus	
Description	Read current Auto-Status mode (?autostatus).
C++	<code>int LSX_GetAutoStatus (int ILSID, int *plStatus);</code>
Parameter	Status: Pointer to integer, in which the current setting of Auto-Status will be returned 0 → Controller sends no status 1 → Controller automatically sends „Position reached“ messages 2 → Controller automatically sends „Position reached“ and status messages 3 → There is only one carriage return sent for „Position reached“
Example	<code>pTango->GetAutoStatus(1, &autostatus);</code>

IsVel	
Description	Read the actual velocities at which the axes are currently travelling. Unlike ' ?vel ' or ' ?speed ' this instruction returns the currently travelled (true) speed of the axes, even when controlled by a HDI device (?isvel).
C++	<code>int LSX_IsVel (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	pdX, pdY, pd Z, pdA: actual axes velocities in [mm/s]
Example	<code>pTango->IsVel(1, &vx, &vy, &vz, &va);</code>

IsVelSingleAxis	
Description	Read the actual velocity at which an axis is currently travelling. Unlike ' ?vel ' or ' ?speed ' this instruction returns the currently travelled (true) speed of the axes, even when controlled by a HDI device (?isvel).
C++	<code>int LSX_IsVelSingleAxis (int ILSID, int lAxis, double *pdVel);</code>
Parameters	lAxis: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) pdVel: actual axis velocity in [mm/s]
Example	<code>pTango->IsVel(1, 2, &vel); // returns actual velocity of y axis</code>

4.5. Settings

GetPowerAmplifier

Description	Provides the amplifier states, on or off (?pa).
C++	<code>int LSX_GetPowerAmplifier (int ILSID, BOOL *pbAmplifier);</code>
Parameters	Amplifier: TRUE → Amplifiers are switched on FALSE → Amplifiers are switched off
Example	<code>pTango->GetPowerAmplifier(1, &Amplifier);</code>

SetPowerAmplifier

Description	Switch amplifier on / off (!pa 2 / !pa 0).
C++	<code>int LSX_SetPowerAmplifier (int ILSID, BOOL bAmplifier);</code>
Parameters	Amplifier: TRUE → Switch amplifiers on FALSE → Switch amplifiers off
Example	<code>pTango->SetPowerAmplifier(1, TRUE); // switches amplifiers on</code>

GetActiveAxes

Description	Provides the axis enable states (?axis).
C++	<code>int LSX_GetActiveAxes (int ILSID, int *plFlags);</code>
Parameters	Flags: 32-Bit Integer. After calling this function the axis bitmask is returned in Bits 0-4 Bit 0 = X, Bit 1 = Y, Bit 2 = Z, Bit 3 = A / 1=axis is on, 0 = axis off or disabled
Example	<code>pTango->GetActiveAxes(1, &Flags);</code> <code>if (Flags & 0x01) ...; // Flags Bit 0 = 1 → X-Axis is on</code> <code>if ((Flags & 0x04) == 0) ...; // Flags Bit 2 = 0 → Z-Axis is off or disabled</code>

SetActiveAxes

Description	Enable or disable axes (!axis).
C++	<code>int LSX_SetActiveAxes (int ILSID, int lFlags);</code>
Parameters	Flags: Bit mask, bits 0 to 3 represent axes X (0x01) to A (0x08) Bit 0 = 1 → X-Axis enabled , Bit 1 = 2→ Y-Axis enabled Bit 2 = 4 → Z-Axis enabled , Bit 3 = 8 → A-Axis enabled
Example	<code>pTango->SetActiveAxes(1, 3);</code> <code>// X- and Y-Axes are enabled (Bits 0 and 1 set),</code> <code>Z-Axis and A-Axis switched off (Bit 2 = 0, Bit 3 = 0)</code>

GetAxisDirection

Description	Retrieves axis directions (?axisdir).
C++	<code>int LSX_GetAxisDirection (int ILSID, int *plXD, int *plYD, int *plZD, int *plAD);</code>
Parameters	XD, YD, ZD, AD: 4 32-Bit Integers 0 → normal turning direction 1 → reversed turning direction
Example	<code>pTango->GetAxisDirection(1, &XD, &YD,&ZD,&AD);</code>

SetAxisDirection

Description	Set axis directions (!axisdir).
C++	<code>int LSX_SetAxisDirection (int ILSID, int lXD, int lYD, int lZD, int lAD);</code>
Parameters	XD, YD, ZD, AD: 4 32-Bit Integers 0 → normal motor turning direction 1 → reverse reversed motor turning direction
Example	<code>pTango->SetAxisDirection(1, 1, 0, 0, 0);</code> <i>// Set direction of X-Axis to reversed (1), other axes not reversed</i>

GetCalibOffset

Description	Retrieves zero position offset of axes (?caliboffset).
C++	<code>int LSX_GetCalibOffset (int ILSID,</code> <code>double *pdX,</code> <code>double *pdY,</code> <code>double *pdZ,</code> <code>double *pdA)</code>
Parameters	X, Y, Z, A: zero position offset from cal switch, depending on dimensions
Example	<code>pTango->GetCalibOffset(1, &X, &Y, &Z, &A);</code>

SetCalibOffset

Description	Sets zero position offset of axes (!caliboffset). The axis zero position is moved from the hardware cal limit switch by this amount.
C++	<code>int LSX_SetCalibOffset (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: typically 0-5 [mm]
Example	<code>pTango->SetCalibOffset(1, 1, 1, 1, 1);</code> <i>// when calibrating, axes X, Y, Z and A are each moved for 1mm (at dimension 2 2 2 2) from zero limit switch towards stage center and then zero position is set (software limit)</i>

GetRMOffset	
Description	Retrieves axis position offsets to RM limit switch (?rmoffset).
C++	<code>int LSX_GetRMOffset (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	X, Y, Z, A: Limit switch position offset, depending on measuring unit (dimension).
Example	<code>pTango->GetRMOffset(1, &X, &Y, &Z, &A);</code>

SetRMOffset	
Description	Sets RM position offset of axes (!rmoffset). The axis stops this amount before the hardware RM endswitch.
C++	<code>int LSX_SetRMOffset (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: typically 0-5 [mm]
Example	<code>pTango->SetRMOffset(1, 1, 1, 1, 1);</code> <i>// limit positions of axes are each moved for 1mm (at dimension 2 2 2 2) towards stage center</i>

GetCalibBackSpeed	
Description	Retrieves revolving speed at which axes are driven from limit switches when calibrating. Speed is equivalent to issued value * 0.01 rev/sec (?calbspeed).
C++	<code>int LSX_GetCalibBackSpeed (int ILSID, int *plSpeed);</code>
Parameters	Speed: Speed value in 1/100 revolutions/second
Example	<code>pTango->GetCalibBackSpeed(1, &lSpeed);</code>

SetCalibBackSpeed	
Description	Sets revolving speed at which axes are driven from limit switches when calibrating. Speed is equivalent to issued value * 0.01 rev/sec (!calbspeed).
C++	<code>int LSX_SetCalibBackSpeed (int ILSID, int lSpeed);</code>
Parameters	Speed: Speed value in 1/100 revolutions/second (within parameters of 1 to 100)
Example	<code>pTango->SetCalibBackSpeed(1, 10);</code> <i>// when calibrating, limit switches are left at 0.1 rev/sec</i>

GetCalibrateDir	
Description	Retrieves calibrating direction (?caldir).
C++	<code>int LSX_GetCalibrateDir (int ILSID, int *plXD, int *plYD, int *plZD, int *plAD);</code>
Parameters	<i>XD, YD, ZD, AD</i>: 32-Bit Integer 0 → normal calibration direction 1 → (reserved, don't use!) 2, ... reserved for center reference modes, refer to TANGO Instruction Set
Example	<code>pTango->GetCalibrateDir(1, &XD, &YD,&ZD,&AD);</code>

SetCalibrateDir	
Description	Set calibrating direction (!caldir).
C++	<code>int LSX_SetCalibrateDir (int ILSID, int lXD, int lYD, int lZD, int lAD);</code>
Parameters	<i>XD, YD, ZD, AD</i>: 32-Bit Integer 0 → normal calibration direction 1 → (reserved, don't use!) 2, ... reserved for center reference modes, refer to TANGO Instruction Set
Example	<code>pTango->(1, 0, 0, 0, 0); // Set all axes to caldir = 0</code>

GetDimensions	
Description	Provides the applied measuring units of axes (?dim)
C++	<code>int LSX_GetDimensions (int ILSID, int *plXD, int *plYD, int *plZD, int *plAD);</code>
Parameters	<i>XD, YD, ZD, AD</i>: Dimension units 0 → Microsteps 1 → μm 2 → mm 3 → Degree 4 → Revolutions 5 → cm 6 → m 7 → Inch 8 → mil (1/1000 Inch) 9 → position in mm and speed in mm/s (also the preferred setting) 10 → position in μm and speed in mm/s (behaves as dim 1 at a 1mm pitch)
Example	<code>pTango->GetDimensions(1, &XD, &YD,&ZD,&AD);</code>

SetDimensions	
Description	Set measuring units of axes (!dim).
C++	<code>int LSX_SetDimensions (int ILSID, int IXD, int IYD, int IZD, int IAD);</code>
Parameters	<i>XD, YD, ZD, AD</i>: Dimension units 0 → Microsteps 1 → μm 2 → mm (Pre-set) 3 → Degree 4 → Revolutions 5 → cm 6 → m 7 → Inch 8 → mil (1/1000 Inch) 9 → position in mm and speed in mm/s 10 → position in μm and speed in mm/s (behaves as dim 1 at a 1mm pitch)
Example	<pre>pTango->SetDimensions(1, 3, 2, 2, 1); // X-Axis in degree, Y- and Z-Axis in mm and A-Axis in μm</pre>

GetResolution	
Description	Provides the applied number of “digits after the millimetre” (?resolution). This command is used for dimensions 1, 2, 9 or 10 (mm and μm). The Tango default is “4 digits after the millimetre” (0.1μm resolution). The Tango transmits (replies) Position values with this resolution to the DLL.
C++	<code>int LSX_GetDimensions (int ILSID, int *plValue);</code>
Parameters	<i>Value</i>: Resolution units 3 → 1 μm resolution 4 → 0.1 μm resolution (Default) 5 → 10 nm resolution 6 → 1 nm resolution
Example	<pre>pTango->GetResolution(1, &resolution);</pre>

SetResolution	
Description	Sets the applied number of “digits after the millimetre” (!resolution). This command is used for dimensions 1, 2, 9 or 10 (mm and µm). The Tango default is “4 digits after the millimetre” (1/10µm resolution). The Tango transmits (replies) Position values with this resolution to the DLL. Specify 5 (=10nm) or 6 (=1nm) digits to get higher position resolution from Tango.
C++	<code>int LSX_SetDimensions (int ILSID, int IValue);</code>
Parameters	Value: Resolution units 3 → 1 µm resolution 4 → 0.1 µm resolution (Default) 5 → 10 nm resolution 6 → 1 nm resolution
Example	<code>pTango->SetResolution(1, 5); // set 5 digits after the decimal point for all axes</code>

GetPitch	
Description	Provides spindle pitch (?pitch).
C++	<code>int LSX_GetPitch (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	X, Y, Z, A: Spindle pitch [mm]
Example	<code>pTango->GetPitch(1, &X, &Y, &Z, &A);</code>

SetPitch	
Description	Set spindle pitch (!pitch). If a Märzhäuser stage is used, the pitch of the axes X+Y is usually pre-defined and does not need to be set.
C++	<code>int LSX_SetPitch (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: 0.0001 to 100 [mm per motor revolution]
Example	<code>pTango->SetPitch(1, 4, 4, 1, 2); // Set spindle pitch of X+Y to 4mm, Z to 1mm, A to 2mm</code>

GetGear	
Description	Retrieves gear ratio (?gear).
C++	<code>int LSX_GetGear (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	X, Y, Z, A: Gear ratio values
Example	<code>pTango->GetGear(1, &X, &Y, &Z, &A);</code>

SetGear	
Description	Set gear ratio (!gear). If a Märzhäuser stage or axis is used, the gear of the axes X+Y is usually pre-defined and does not need to be set. The default for most applications is 1.
C++	<code>int LSX_SetGear (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: 0.01 – 1000, default = 1
Example	<code>pTango->SetGear(1, 4.0, 2.0, 1.0, 1.0);</code> <i>// programs gear ratios ¼ for Z, ½ for Y and 1/1 for Z and A</i>

GetMotorSteps	
Description	Retrieves number of motor steps (?motorsteps).
C++	<code>int LSX_GetMotorSteps (int ILSID, int *IX, int *IY, int *IZ, int *IA);</code>
Parameters	X, Y, Z, A: Number of motor steps
Example	<code>pTango->GetMotorSteps(1, &X, &Y, &Z, &A);</code>

SetMotorSteps	
Description	Set number of motor steps. Default 200 for 1,8° stepper motors (!motorsteps).
C++	<code>int LSX_SetMotorSteps (int ILSID, int IX, int IY, int IZ, int IA);</code>
Parameters	X, Y, Z, A: Motor steps X, Y, Z and A-Axis
Example	<code>pTango->SetMotorCurrent(1, 200, 200, 400, 24);</code> <i>// set X, Y to default 200, Z to 400 and A to 24 steps motor type (1.8°, 0.9° and 15° mot.)</i>

GetMotorCurrent

Description	Retrieves electrical motor current (?cur).
C++	<code>int LSX_GetMotorCurrent (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	X, Y, Z, A: Electrical motor currents in [A]
Example	<code>pTango->GetMotorCurrent(1, &X, &Y, &Z, &A);</code>

SetMotorCurrent

Description	Set electrical current of motor (!cur).
C++	<code>int LSX_SetMotorCurrent (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: Motor current X, Y, Z and A-Axis in [A]
Example	<code>pTango->SetMotorCurrent(1, 1.0, 1.0, 0.8, 0.8);</code> <i>// motor current X- and Y-Axis 1 Ampere; Z- and A-Axis 0.8 Ampere</i>

GetReduction

Description	Retrieves motor current reduction factor for idle axes (?reduction). Info: A reduction of 0.7 (to 70%) will reduce the power and produced heat by 50% and a reduction of 0.5 (50%) will reduce the power and heat of idle axes to about 25%.
C++	<code>int LSX_GetReduction (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA)</code>
Parameters	X, Y, Z, A: Electrical motor current reduction 0.00 ... 1.00 (= 0...100%)
Example	<code>pTango->GetReduction(1, &X, &Y, &Z, &A);</code>

SetReduction

Description	Set reduction factor of motor current (!reduction). Info: A reduction of 0.7 (to 70%) will reduce the power and produced heat by 50% and a reduction of 0.5 (50%) will reduce the power and heat of idle axes to about 25%. If the reduction is set below 0.3 (< 30%), the closed loop will be disabled while applied.
C++	<code>int LSX_SetReduction (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: Electrical motor current reduction 0.00 ... 1.00 (= 0...100%)
Example	<code>pTango->SetReduction(1, 0.1, 0.7, 0.5, 0.5);</code> <i>// standby current X-Axis = 0.1*rated current, Y-Axis = 0.7*rated current, Z- and A-Axis = 0.5*rated current</i>

GetCurrentDelay	
Description	Provides time delay for motor current reduction (?curdelay).
C++	<code>int LSX_GetCurrentDelay (int ILSID, int *plX, int *plY, int *plZ, int *plA);</code>
Parameters	<i>X, Y, Z, A</i> : Time delay [ms]
Example	<code>pTango->GetCurrentDelay(1, &X, &Y,&Z,&A);</code>

SetCurrentDelay	
Description	Sets the time delay, after which the motor current reduction is applied (!curdelay).
C++	<code>int LSX_SetCurrentDelay (int ILSID, int IX, int IY, int IZ, int IA);</code>
Parameters	<i>X, Y, Z, A</i> : 0...65000 [ms] (A delay of 0 disables the current reduction = default)
Example	<code>pTango->SetCurrentDelay(1, 100, 300, 1000, 0);</code>

GetSpeedPoti	
Description	Shows whether the speed potentiometer functionality is switched on or off (?pot). Speed potentiometer shall not be used. It slows down the controller execution times and is not supported on all Tangos and firmware versions.
C++	<code>int LSX_GetSpeedPoti (int ILSID, BOOL *pbSpePoti);</code>
Parameter:	The SpePoti flag shows, whether potentiometer is switched on (1) or off (0)
Example	<code>pTango->GetSpeedPoti(1, &flag);</code>

SetSpeedPoti	
Description	Switches Speed Potentiometer functionality on or off (!pot). Speed potentiometer shall not be used. It slows down the controller execution times and is not supported on all Tangos and firmware versions.
C++	<code>int LSX_SetSpeedPoti (int ILSID, BOOL bSpeedPoti);</code>
Parameters	<i>SpeedPoti</i> = FALSE → pre-set speed (velocity) is used as movement speed = TRUE → pre-set speed (velocity) can be reduced depending on the speed-potentiometer deflection
Example	<code>pTango->SetSpeedPoti(1, TRUE); // switch potentiometer mode on</code>

GetStopPolarity	
Description	Retrieves active polarity of the stop input signal (?stoppol).
C++	<code>int LSX_GetStopPolarity (int ILSID, BOOL *pbHighActiv);</code>
Parameters	HighActiv: TRUE → stop input is high active FALSE → stop input is low active
Example	<code>pTango->GetStopPolarity(1, &HighActiv);</code>

SetStopPolarity	
Description	Set polarity for active stop input signal (!stoppol). As the stop input has a pull up resistor to 5V, ensure that switches contact to ground. A normally open contact will require a low active setting while a normally closed contact requires the high active setting.
C++	<code>int LSX_SetStopPolarity (int ILSID, BOOL bHighActiv);</code>
Parameters	HighActiv: TRUE → stop input high active FALSE → stop input low active
Example	<code>pTango->SetStopPolarity(1, FALSE);</code> <i>// stop input is low active (e.g. normally open switch to ground)</i>

GetVel	
Description	Retrieves velocity of all axes (?vel).
C++	<code>int LSX_GetVel (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	pdX, pdY, pdZ, pdA: Velocity values [depending on dimension: rev/sec or mm/s]
Example	<code>pTango->GetVel(1, &X, &Y, &Z, &A);</code>

SetVel	
Description	Set velocity of all axes (!vel).
C++	<code>int LSX_SetVel (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: > 0 ... max. speed [depending on dimension: rev/sec or mm/s]
Example	<code>pTango->SetVel(1, 20.0, 15.0, 0.5, 10);</code>

SetVelSingleAxis	
Description	Set velocity of a single axis (!vel x,y,z,a).
C++	<code>int LSX_SetVelSingleAxis (int lLSID, int lAxis, double dVel);</code>
Parameters	Axis: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) Vel: >0 to max. speed [depending on dimension: rev/sec or mm/s]
Example	<code>pTango->SetVelSingleAxis(1, 2, 10.0); // sets speed of Y-Axis to 10 [rev/s or mm/s]</code>

GetSecVel	
Description	Retrieves secure velocity of all axes (?secvel). The secure velocity limits the axis velocity to secvel until the travel range of the axis is known (calibration and range measure executed).
C++	<code>int LSX_GetSecVel (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	pdX, pdY, pdZ, pdA: Velocity values [mm/s]
Example	<code>pTango->GetSecVel(1, &X, &Y, &Z, &A);</code>

SetSecVel	
Description	Set secure velocity of all axes (!secvel). The secure velocity limits the axis velocity to secvel until the travel range of the axis is known (calibration and range measure executed). Default = 10mm/s, max. = 100mm/s.
C++	<code>int LSX_SetSecVel (int lLSID, double dX, double dY, double dZ, double dA);</code>
Parameters	dX, dY, dZ, dA: >0 ... max. speed [mm/s]
Example	<code>pTango->SetSecVel(1, 20.0, 15.0, 0.5, 10);</code>

SetSecVelSingleAxis	
Description	Set secure velocity of a single axis (!secvel x,y,z,a). The secure velocity limits the axis velocity to secvel until the travel range of the axis is known (calibration and range measure executed). Default = 10mm/s, max. = 100mm/s.
C++	<code>int LSX_SetSecVelSingleAxis (int lLSID, int lAxis, double dVel);</code>
Parameters	Axis: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) Vel: >0 ... max. speed [mm/s]
Example	<code>pTango->SetSecVelSingleAxis(1, 2, 10.0); // sets secure speed of Y-Axis to 10 mm/s</code>

GetVelFac

Description	Retrieves velocity reduction factor of all axes (?velfac). A velocity factor which is multiplied to the velocity setting. Should be left at the default and only used for applications where it is required. Else just set !vel accordingly.
C++	<code>int LSX_GetVelFac (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	dX, dY, dZ, dA: Velocity factor, default = 1
Example	<code>pTango->GetVelFac(1, &X, &Y, &Z, &A);</code>

SetVelFac

Description	Set velocity reduction factor (!velfac). A velocity factor which is multiplied to the velocity setting. Should be left at the default and only used for applications where it is required. Else just set !vel accordingly.
C++	<code>int LSX_SetVelFac (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	dX, dY, dZ, dA: Velocity reduction factor, within parameters 0.01 -- 1.00
Example	<code>pTango->SetVelFac(1, 1, 1, 0.1, 0.1);</code> <i>// reduces velocity of Z and A axes to 1/10 of nominal velocity</i>

GetAccel	
Description	Retrieves acceleration (?accel).
C++	<code>int LSX_GetAccelFunc (double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	dX, dY, dZ, dA: Acceleration values [m/s ²]
Example	<code>pTango->GetAccel(1, &X, &Y, &Z, &A);</code>

SetAccel	
Description	Set acceleration (!accel).
C++	<code>int LSX_SetAccel (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	dX, dY, dZ, dA: 0.01 - 20.00 [m/s ²]
Example	<code>pTango->SetAccel(1, 1.0, 1.5, 0, 0);</code>

SetAccelSingleAxis	
Description	Set acceleration of a single axis (!accel).
C++	<code>int LSX_SetAccelSingleAxis (int ILSID, int IAxis, double dAccel);</code>
Parameters	Axis: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) Accel: Acceleration 0.01 - 20.00 [m/s ²]
Example	<code>pTango->SetAccelSingleAxis(1, 3, 1.0); // sets acceleration of Z-Axis to 1.0 m/s²</code>

GetAccelFunc	
Description	Retrieves acceleration function (?accelfunc).
C++	<code>int LSX_GetAccelFunc (int ILSID, int *IX, int *IY, int *IZ, int *IR);</code>
Parameters	IX, IY, IZ, IR: Acceleration function of the axes (pointer to variable) 0 = trapezoidal acceleration 1 = s-curve acceleration
Example	<code>pTango->GetAccelFunc(1, &IX, &IY, &IZ, &IR);</code>

SetAccelFunc	
Description	Sets acceleration function: 0 for trapezoidal, 1 for s-curve (!accelfunc).
C++	<code>int LSX_SetAccelFunc (int ILSID, int IX, int IY, int IZ, int IR);</code>
Parameters	IX, IY, IZ, IR: Acceleration function for the axes 0 = trapezoidal acceleration 1 = s-curve acceleration
Example	<code>pTango->SetAccel(1, IX, IY, IZ, IR);</code>

GetStopAccel

Description	Provides deceleration for error conditions (?stopaccel).
C++	<code>int LSX_GetStopAccel (int ILSID, double *pdXD, double *pdYD, double *pdZD, double *pdAD);</code>
Parameters	<i>XD, YD, ZD, AD</i> : Deceleration values [m/s ²]
Example	<code>pTango->GetStopAccel(1, &XD, &YD, &ZD, &AD);</code>

SetStopAccel

Description	Deceleration value used when moving into a limit switch or causing a stop condition. If the axis acceleration (set with LSX_SetAccel) is higher, then this higher value will be used (!stopaccel).
C++	<code>int LSX_SetStopAccel (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	<i>X, Y, Z, A</i> : Brake acceleration, within parameters 0.01 to 20 [m/s ²]
Example	<code>pTango->SetStopAccel(1, 1.5, 1.5, 1.5, 1.5);</code>

GetBlSmoothSingleAxis

Description	Read the currently used backlash smoothing mode of an axis (?blsmooth). Backlash and backlash smoothing are used on open loop systems without encoders. As the backlash individually compensates a deviation that occurs depending on travel direction, the blsmooth can be used to lower the impact on the compensation (like introduced shake).
C++	<code>int LSX_GetBlSmoothAxis (int ILSID, int lAxis, int *plBlSmooth);</code>
Parameters	<i>lAxis</i> 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) <i>lBlSmooth</i> Pointer to int for returning blsmooth mode of the specified axis
Example	<code>pTango->LSX_GetBlSmoothSingleAxis(1, 2, &lBlSmooth); // Get blsmooth of Y</code>

SetBlSmoothSingleAxis

Description	Set the currently used backlash smoothing mode of an axis (!blsmooth). Backlash and backlash smoothing are used on open loop systems without encoders. As the backlash individually compensates a deviation that occurs depending on travel direction, the blsmooth can be used to lower the impact on the compensation (like introduced shake).
C++	<code>int LSX_SetBlSmoothSingleAxis (int ILSID, int lAxis, int lSmooth);</code>
Parameters	<i>lAxis</i> 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) <i>lBlSmooth</i> New blsmooth mode for the axis
Example	<code>pTango->LSX_SetBlSmoothSingleAxis (1, 2, 0); // Set blsmooth of Y to 0</code>

LStepSave	
Description	Save current configuration in Tango EEPROM (!save). All settings made in the Tango (velocity, limit switch etc.) will be stored permanently. If the save command failed, the function returns error 4002.
C++	<code>int LSX_LStepSave (int ILSID);</code>
Parameters	-
Example	<code>pTango->LStepSave(1);</code>

SoftwareReset	
Description	Tango is reset and reboots (!reset).
C++	<code>int LSX_SoftwareReset (int ILSID);</code>
Parameters	-
Example	<code>pTango->SoftwareReset(1);</code>

4.6. Move Commands and Positioning Management

Calibrate	
Description	All enabled axes will be calibrated (!cal). Axes are driven towards smaller position values until reaching the cal limit switch and then driven with reduced speed in opposite direction until limit switch is no longer active. If a position offset is configured, the axis continues traveling for that distance. Then the zero point is set.
C++	<code>int LSX_Calibrate (int ILSID);</code>
Parameters	-
Example	<code>pTango->Calibrate(1);</code>

CalibrateEx	
Description	Calibrates selected or single axes (!cal x / !cal y / !cal z / !cal a / !cal [1...15]). Only calibrates axes with corresponding Bit set in transferred Integer value.
C++	<code>int LSX_CalibrateEx (int ILSID, int IFlags);</code>
Parameters	Flags: Bit mask for the axes to be calibrated Bit 0=X, Bit 1=Y, Bit 2=Z, Bit 3=A If Bit 2 = 1 → calibrate Z-Axis If Bit 2 = 0 → do not calibrate Z-Axis
Example	<code>pTango->CalibrateEx(1, 6); // only calibrate Y- and Z-Axis (Bit 1 and 2 set = 2+4 = 6)</code>

RMeasure	
Description	Travels to maximum position of all enabled axes (!rm). Axes are driven towards larger position values until reaching rm limit switch and then driven with reduced speed in opposite direction until limit switch is no longer active. If a rm position offset is configured, the axis continues traveling for that distance. Then the max. possible travel range is set. Only to be executed when the stage features limit switches on either end. After this command the controller remembers the switch position and disables a possible security speed limitation.
C++	<code>int LSX_RMeasure (int ILSID);</code>
Parameters	-
Example	<code>pTango->RMeasure(1);</code>

RMeasureEx	
Description	Measure maximum position of axes (!rm x / !rm y / !rm z / !rm a / !rm [1...15]). Moves the stage towards the RM limit switch only for the axes whose corresponding axis bit mask is set.
C++	<code>int LSX_RMeasureEx (int lLSID, int lFlags);</code>
Parameters	Flags: Bit mask Bit 2 = 1 → calibrate Z-Axis Bit 2 = 0 → Do not calibrate Z-Axis ...
Example	<code>pTango->RMeasureEx(1, 3);</code> // measure maximum position of X- and Y-Axis (1+2=3)

GetDelay	
Description	Retrieves time delay (wait time) until a commanded move is executed (?delay).
C++	<code>int LSX_GetDelay (int lLSID, int *plDelay);</code>
Parameters	Delay: Delay [ms]
Example	<code>pTango->GetDelay(1, &Delay);</code>

SetDelay	
Description	Sets the time for which move commands are delayed (!delay). Before each positioning the controller waits for this period of time delay, default = 0.
C++	<code>int LSX_SetDelay (int lLSID, int lDelay);</code>
Parameters	Delay: 0 - 10000 [ms]
Example	<code>pTango->SetDelay(1, 1000);</code> // 1 Second delay until a move command is executed

MoveAbs	
Description	All axes are moved absolute positions (!moa). Axes X, Y, Z and A are positioned at transferred position values.
C++	<code>int LSX_MoveAbs (int ILSID, double dX, double dY, double dZ, double dA, BOOL bWait);</code>
Parameters	X, Y, Z, A: $\pm$ Travel range, command depends on measuring unit Wait: Determines, whether function shall return after reaching position (= TRUE) or directly after sending the command (= FALSE)
Example	<code>pTango->MoveAbs(1, 10.0, 10.0, -10.0, 10.0, TRUE);</code>

MoveAbsSingleAxis	
Description	Positions a single axis to the specified absolute position (!moa). (Remarks: As autostatus here is always 0, this instruction will not generate a “trigger after position reached”, which requires autostatus being set to 1 or 3)
C++	<code>int LSX_MoveAbsSingleAxis (int ILSID, int lAxis, double dValue, BOOL bWait);</code>
Parameters	Axis: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) Value: Position, command depends on measuring unit (dimension) Wait: TRUE = Wait until the specified axis has reached its position (The DLL will do by polling the axis **) FALSE = No wait, function returns after sending the move ** The wait will only check for the specified axis. Therefore, autostatus will be set to 0 and the axis state is polled.
Example	<code>pTango->MoveAbsSingleAxis(1, 2, 10.0, TRUE);</code> <i>// Moves the the Y-Axis to 10mm (mm if dim=2 or 9) and waits until reached</i>

MoveEx	
Description	Extended move command (!moa or !mor). Function LSX_MoveEx can execute relative and absolute travel commands, synchronously as well as asynchronously. The number of axes, which are to be moved, can be determined by using AxisCount parameter. For example, this function can be used to move X and Y.
C++	<pre>int LSX_MoveEx (int ILSID, double dX, double dY, double dZ, double dA, BOOL bRelative, BOOL bWait, int lAxisCount);</pre>
Parameters	X, Y, Z, A: Position vectors Relative: FALSE = values of X, Y, Z and A are interpreted as absolute coordinates TRUE = values are interpreted as relative coordinates to current position Wait: TRUE = the function doesn't return before reaching the target position, FALSE = function returns immediately after sending the command to the Tango. AxisCount: Number of axes, which are to be moved e.g. if AxisCount = 1, only X is moved e.g. if AxisCount = 2, X and Y are moved ...
Example	<pre>pTango->MoveEx(1, 2.0, 3.0, 0, 0, TRUE, TRUE, 2); // X and Y are moved relatively by 2 or 3, function call returns when positions are reached</pre>

MoveRel	
Description	Move relative position (!mor). Axes X, Y, Z and A are moved by the transmitted distances. All axes reach their destinations simultaneously.
C++	<pre>int LSX_MoveRel (int ILSID, double dX, double dY, double dZ, double dA, BOOL bWait);</pre>
Parameters	X, Y, Z, A: +/- Travel range, command depends on measuring unit (dimension) Wait: TRUE = function waits until position is reached FALSE = function does not wait
Example	<pre>pTango->MoveRel(1, 10.0, 10.0, -10.0, 10.0, TRUE);</pre>

MoveRelSingleAxis

Description	Move single axis relative (!mor). Similar to MoveAbsSingleAxis , but for relative dist.
C++	<code>int LSX_MoveRelSingleAxis (int ILSID, int lAxis, double dValue, BOOL bWait);</code>
Parameters	Axis: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) Value: Distance, command depends on set measuring unit, Wait: TRUE or FALSE
Example	<code>pTango->MoveRelSingleAxis(1, 3, 5,0, TRUE);</code> <i>// Z-Axis is moved by 5mm in positive direction, the function waits until reached</i>

MoveRelShort

Description	Relative positioning with short command (m). This command may be used to execute several fast equal distance relative moves without communication overhead. Distances must be pre-set once with LSX_SetDistance or are taken from the most recent LSX_MoveRel distances.
C++	<code>int LSX_MoveRelShort (int ILSID);</code>
Parameters	-
Example	<code>pTango->SetDistance(1, 1.0, 1.0, 0, 0);</code> <code>for (i = 0; i < 10; i++) pTango->MoveRelShort(1);</code> <i>// position X- and Y-Axis 10 times relatively by 1mm</i>

GetDistance

Description	Retrieve distance values last used for LSX_MoveRelShort (?distance).
C++	<code>int LSX_GetDistance (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	X, Y, Z, A: Current distances of all axes, depending on corresponding measuring unit.
Example	<code>pTango->GetDistance(1, &X, &Y, &Z, &A);</code>

SetDistance

Description	Set distance (!distance). Sets distance parameters for command LSX_MoveRelShort. This enables very fast equal distance relative positioning without the need of communication overhead.
C++	<code>int LSX_SetDistance (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: Min-/max- travel range, values depend on measuring unit.
Example	<code>pTango->SetDistance(1, 1, 2, 0, 0);</code> <i>// sets distances for axes X to 1mm and Y to 2mm (if dimension=2), Z and A are not moved when calling function LSX_MoveRelShort</i>

Go	
Description	All axes are moved to given absolute positions (!go). You may send Go while preceding Go is in progress. This command is designed to be called directly from mouse events to move axes. Axes X, Y, Z and A are positioned at transferred position values.
C++	<code>int LSX_Go (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: $\pm$ Travel range, command depends on measuring unit
Example	<code>pTango->Go(1, 10.0, 10.0, -10.0, 10.0);</code>

GoSingleAxis	
Description	One axis is moved to given absolute position (!go). You may send GoSingleAxis while preceding GoSingleAxis is in progress. This command is designed to be called directly from mouse events to move axes. Addressed Axis X, Y, Z or A is positioned to transferred position.
C++	<code>int LSX_GoSingleAxis (int ILSID, int IAxis, double dValue);</code>
Parameters	Axis: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) Value: $\pm$ Travel range, command depends on measuring unit
Example	<code>pTango->GoSingleAxis(1, 2, 12.34); //move Y to target position 12.34</code>

GoEx	
Description	Similar like Go() command with additional parameter (!go). The number of axes, which are to be moved, can be determined by using AxisCount parameter. For example this function can be used to move X and Y.
C++	<code>int LSX_GoEx (int ILSID, double dX, double dY, double dZ, double dA, int IAxisCount);</code>
Parameters	X, Y, Z, A: Position vectors AxisCount: Number of axes, which are to be moved e.g. if AxisCount = 1, only X is moved e.g. if AxisCount = 2, X and Y are moved ...
Example	<code>pTango->GoEx(1, 2.0, 3.0, 56.78, 67.89, 2);</code> <i>// X and Y are moved relatively by 2 or 3 while Z and A will not move</i>

GetDigJoySpeed	
Description	Retrieves current travel speed as initiated by SetDigJoySpeed (?speed).
C++	<code>int LSX_GetDigJoySpeed (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	X, Y, Z, A: Speed values [motor rev./sec] or [mm/s in case of Dimension 9 and 10]
Example	<code>pTango->GetDigJoySpeed(1, &X, &Y, &Z, &A);</code>

SetDigJoySpeed	
Description	This command moves axes at a constant speed (!speed). The speed values can be overwritten without the need to stop. To stop a speed move, the speed of the axes to stop can be set to 0. Else the constant speed is maintained until approaching a limit switch or StopAxes (abort, a) is executed. The Speed function is disabled by setting SetJoyDir to 0.
C++	<code>int LSX_SetDigJoySpeed (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: Speed, within parameter range +- max. speed Depending on Dimension in [motor revolutions/sec] or [mm/s] in Dimensions 9 and 10
Example	<code>pTango->SetDigJoySpeed(1, 0, 10.0, 25.0, 0);</code> <i>// Axes X and A - speed 0 and Joystick operation „OFF“,</i> <i>Axis Y - speed 10.0 r/sec and Joystick operation „ON“,</i> <i>Axis Z -speed 25.0 r/sec and Joystick operation „ON“</i>

StopAxes	
Description	Abort (a). Stops all moving axes with their stop acceleration.
C++	<code>int LSX_StopAxes (int ILSID);</code>
Parameters	-
Example	<code>pTango->StopAxes(1);</code>

StopAxesEx	
Description	Abort (a). Stops the specified moving axis/axes with their stop acceleration. Axes are specified as integer bitmask (1=X, 2=Y, 4=Z, 8=A)
C++	<code>int LSX_StopAxesEx (int ILSID, int IFlags);</code>
Parameters	IFlags: Axis bits of the axes to stop, 0 ... 15 (0...0x0F)

Example	pTango->StopAxesEx(1, 3); // 3: Stop X+Y axis
----------------	---

WaitForAxisStop

Description	Function returns as soon as the axes selected by the bit mask “lAFlags” have reached their target positions or the timeout is exceeded. LSX_WaitForAxisStop uses '?statusaxis' to poll axis status, therefore it will not generate a “trigger after position reached”, if that trigger mode is required.
C++	<pre>int LSX_WaitForAxisStop (int lLSID, int lAFlags, int lATimeoutValue, BOOL *pbATimeout);</pre>
Parameters	AFlags: Bit mask Bit 0: X-Axis Bit 1: Y-Axis Bit 2: Z-Axis Bit 3: A-Axis ATimeoutValue: Timeout in milliseconds WaitForAxisStop returns latest after this period of time pbATimeout is set to “TRUE”, if axes are still in motion. Setting lATimeoutValue = 0 disables the Timeout (wait infinite) pbATimeout Flag: Shows whether a Timeout has occurred (TRUE) or not (FALSE)
Example	<pre>pTango->WaitForAxisStop(1, 3, 1000, pbATimeout); // wait until X- and Y-Axes have stopped, wait will end (timeout) after 1 second pTango->WaitForAxisStop(1, 7, 20000, pbATimeout); // wait until X-, Y- and Z-Axis have stopped, wait will end (timeout) after 20 sec.</pre>

4.7. Joystick and Handwheel

SetJoystickOff	
Description	Switch Joystick and in general the HDI off (!joy 0).
C++	<code>int LSX_SetJoystickOff (int ILSID);</code>
Parameters	-
Example	<code>pTango->SetJoystickOff(1);</code>

SetJoystickOn	
Description	Switch Joystick and in general the HDI on (!joy 1 / !joy 2 / !joy 4).
C++	<code>int LSX_SetJoystickOn (int ILSID, BOOL bPositionCount, BOOL bEncoder);</code>
Parameters	PositionCount = TRUE → position count on = FALSE → position count off Encoder = TRUE → encoder values, if encoders available
Example	<code>pTango->SetJoystickOn(1, TRUE, TRUE);</code> <i>// switch on joystick with position count (encoder values)</i>

GetJoystickDir	
Description	Retrieves axis direction for the Joystick and other HDI input devices (?joydir). Individual axes can be reversed or disabled.
C++	<code>int LSX_GetJoystickDir (int ILSID, int *plXD, int *plYD, int *plZD, int *plAD);</code>
Parameters	XD, YD, ZD, AD: 0 → Axis disabled for Joystick (deflection ignored) 1 → positive axis direction, current reduction disabled (treated by TANGO as 2) -1 → negative axis direction, current reduction disabled (treated by TANGO as -2) 2 → positive axis direction with current reduction (default) -2 → negative axis direction with current reduction
Example	<code>pTango->GetJoystickDir(1, &XD, &YD, &ZD, &AD);</code>

SetJoystickDir

Description	Sets axis direction for Joystick and other HDI input devices (!joydir). Individual axes can be reversed or disabled. Setting JoystickDir to 0 also prevents speed moves of the axes.
C++	<code>int LSX_SetJoystickDir (int llsid, int lxd, int lzd, int lzd, int lzd);</code>
Parameters	<i>XD, YD, ZD, AD:</i> 0 → Axis disabled for Joystick (deflection ignored) 1 → positive axis direction, current reduction disabled (treated by TANGO as 2) -1 → negative axis direction, current reduction disabled (treated by TANGO as -2) 2 → positive axis direction with current reduction (default) -2 → negative axis direction with current reduction
Example	<pre>pTango->SetJoystickDir(1, 1, 1, -1, 0); // X- and Y-Axis positive direction, Z-Axis negative direction, A-Axis blocked</pre>

GetJoystick

Description	Retrieves Joystick and in general the HDI status (?joy).
C++	<code>int LSX_GetJoystick (int llsid,</code> <code>BOOL *pbJoystickOn,</code> <code>BOOL *pbManual,</code> <code>BOOL *pbPositionCount,</code> <code>BOOL *pbEncoder);</code>
Parameters	<i>JoystickOn:</i> TRUE = Joystick switched on <i>Manual:</i> FALSE = Joystick switch set on automatic TRUE = Joystick is switched on manually via switch <i>PositionCount:</i> TRUE = position count switched on <i>Encoder:</i> TRUE = encoder values, if available
Example	<pre>pTango->GetJoystick(1, &JoystickOn, &Manual, &PositionCount, &Encoder);</pre>

GetJoyChangeAxis	
Description	Retrieves Joystick X<->Y axis change state (?joychangeaxis).
C++	<code>int LSX_GetJoyChangeAxis (int ILSID, BOOL *pbChangedXY);</code>
Parameters	bChangedXY: TRUE = Joystick X and Y axes assignment swapped FALSE = Normal operation: X controls X, Y controls Y (default)
Example	<code>pTango->GetJoyChangeAxis(1, &bChangedXY);</code>

JoyChangeAxis	
Description	Set Joystick X<->Y axis change state (!joychangeaxis).
C++	<code>int LSX_SetJoyChangeAxis (int ILSID, BOOL bChangeXY);</code>
Parameters	bChangeXY: TRUE = Joystick X and Y axes assignment swapped FALSE = Normal operation: X controls X, Y controls Y (default)
Example	<code>pTango->JoyChange(1, bChangeXY);</code>

GetHandWheel	
Description	Retrieves hand wheel status (?hw). Not supported by TANGO controllers! Use GetJoystick().
C++	<code>int LSX_GetHandWheel (int ILSID, BOOL *pbHandWheelOn, BOOL *pbPositionCount, BOOL *pbEncoder);</code>
Parameters	HandWheelOn: TRUE = hand wheel switched on FALSE = hand wheel switched off PositionCount: TRUE = position count switched on FALSE = position count switched off Encoder: TRUE = encoder values, if available
Example	<code>pTango->GetHandWheel(1, &HandWheelOn, &PositionCount, &Encoder);</code>

SetHandWheelOff	
Description	Switch hand wheel off (!hw 0). Not supported by TANGO controllers! Use SetJoystickOff().
C++	<code>int LSX_SetHandWheelOff (int ILSID);</code>
Parameters	-
Example	<code>pTango->SetHandWheelOff(1);</code>

SetHandWheelOn	
Description	Switch hand wheel on (!hw 1 / !hw 2 / !hw 3). Not supported by TANGO controllers! Use SetJoystickOn().
C++	<code>int LSX_SetHandWheelOn (int ILSID, BOOL bPositionCount, BOOL bEncoder);</code>
Parameters	<i>PositionCount</i> = TRUE → position counter on = FALSE → position counter off <i>Encoder</i> = TRUE → encoder values, if encoders available
Example	<code>pTango->SetHandWheelOn(1, TRUE, TRUE);</code> <i>// switch on hand wheel with position count (encoder values)</i>

GetJoystickWindow	
Description	Retrieves idle window for analogue Joysticks (?joywindow).
C++	<code>int LSX_GetJoystickWindow (int ILSID, int *plAValue);</code>
Parameters	<i>AValue</i> : Analogue signal range (as digits) in which axes do not move.
Example	<code>pTango->GetJoystickWindow(1, &AValue);</code>

SetJoystickWindow	
Description	Set Joystick idle window for analogue Joysticks (!joywindow). A value in digits which configures an angle where an analogue Joystick deflection has no effect. Used to compensate for mechanical and signal noise effects which else would cause a minor motion of the axes. Does not apply to TANGOs with digital HDI.
C++	<code>int LSX_SetJoystickWindow (int ILSID, int lAValue);</code>
Parameters	<i>AValue</i> : Analogue signal range (as digits) in which axes do not move. 0 ... 100
Example	<code>pTango->SetJoystickWindow(1, 30);</code>

GetHwFactor	
Description	Read hand wheel factor of all axes, in [mm per knob rotation] (?hwfactor).
C++	<code>int LSX_GetHwFactor (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	X, Y, Z, A: Pointer to double
Example	<code>pTango->GetHwFactor(1, &dX, &dY, &dZ, &dA);</code>

SetHwFactor	
Description	Set hand wheel factor for all axes, in [mm per knob rotation] (!hwfactor).
C++	<code>int LSX_SetHwFactor (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: Floating point values (double) in mm/rev
Example	<code>pTango->SetHwFactor(1, dX, dY, dZ, dA);</code>

GetHwFactorSingleAxis	
Description	Read hand wheel factor of the specified axis, in [mm per knob rotation] (?hwfactor).
C++	<code>int LSX_GetHwFactorSingleAxis (int ILSID, int lAxis, double *pdFactor);</code>
Parameters	lAxis: Axis number 1, 2, 3, 4 (corresponding to axes X, Y, Z, A) Factor: Pointer to double
Example	<code>pTango->GetHwFactorSingleAxis(1, 2, &dFactor);</code>

SetHwFactorSingleAxis	
Description	Set hand wheel factor of the specified axis, in [mm per knob rotation] (!hwfactor).
C++	<code>int LSX_SetHwFactorSingleAxis (int ILSID, int lAxis, double dFactor);</code>
Parameters	lAxis: Axis number 1, 2, 3, 4 (corresponding to axes X, Y, Z, A) Factor: Floating point value (double) in mm/rev
Example	<code>pTango->SetHwFactorSingleAxis (1, 2, 14.4); // Set Factor of Y-axis to 14.4mm/rev</code>

GetHwFactorB	
Description	Read second hand wheel factor of all axes, in [mm per knob rotation] (?hwfactorb).
C++	<code>int LSX_GetHwFactorB (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	Pointer to double for all axes
Example	<code>pTango->GetHwFactorB(1, &dX, &dY, &dZ, &dA);</code>

SetHwFactorB	
Description	Set second hand wheel factor for all axes, in [mm per knob rotation] (!hwfactorb).
C++	<code>int LSX_SetHwFactorB (int lLSID, double dX, double dY, double dZ, double dA)</code>
Parameters	Double values for all axes
Example	<code>pTango->SetHwFactorB(1, dX, dY, dZ, dA);</code>

GetHwFactorBSingleAxis	
Description	Read hand wheel factor B of the specified axis, in [mm per knob rotation] (?hwfactorb).
C++	<code>int LSX_GetHwFactorBSingleAxis (int lLSID, int lAxis, double *pdFactor);</code>
Parameters	lAxis: Axis number 1, 2, 3, 4 (corresponding to axes X, Y, Z, A) Factor: Pointer to double
Example	<code>pTango->GetHwFactorBSingleAxis(1, 2, &dFactorB);</code>

SetHwFactorBSingleAxis	
Description	Set hand wheel factor B of the specified axis, in [mm per knob rotation] (!hwfactorb).
C++	<code>int LSX_SetHwFactorBSingleAxis (int lLSID, int lAxis, double dFactorB);</code>
Parameters	lAxis: Axis number 1, 2, 3, 4 (corresponding to axes X, Y, Z, A) Factor: Floating point value (double) in mm/rev
Example	<code>pTango->SetHwFactorBSingleAxis (1, 2, 0.5); // Set Factor B of Y-axis to 0.5mm/rev</code>

GetZwTravel	
Description	Read z-wheel travel distances, in [mm per knob rotation] (?zwtravel).
C++	<code>int LSX_GetZwTravel (int ILSID, int IIndex, double *pdDistance);</code>
Parameters	IIndex: 1: Get setting for standard distance 2: Get setting for slow distance 3: Get setting for fast distance dDistance: Pointer to double
Example	<code>pTango->GetZwTravel (1, IIndex, &dDistance);</code>

SetZwTravel	
Description	Set z-wheel travel distances, in [mm per knob rotation] (!zwtravel).
C++	<code>int LSX_SetZwTravel (int ILSID, int IIndex, double dDistance);</code>
Parameters	IIndex: 1: Set standard distance 2: Set slow distance 3: Set fast distance dDistance: Double value in mm/rev
Example	<code>pTango->SetZwTravel (1, IIndex, dDistance);</code>

GetHdiKeys	
Description	Get HDI device key states, e.g. Joystick or ERGODRIVE (?key).
C++	<code>int LSX_GetHdiKeys (int ILSID, int *plKey1, int *plKey2, int *plKey3, int *plKey4);</code>
Parameters	Pointers to int, 1=Key is currently pressed, 0=key not pressed
Example	<code>pTango->GetHdiKeys(1, &lKey[0], &lKey[1], &lKey[2], &lKey[3]);</code>

GetKey	
Description	Get HDI device key states, e.g. Joystick or ERGODRIVE (?key). Same as GetHdiKey(), but uses BOOL pointers instead of int.
C++	<code>int LSX_GetKey (int ILSID, BOOL *pbKey1, BOOL *pbKey2, BOOL *pbKey3, BOOL *pbKey4);</code>
Parameters	Pointers to BOOL, TRUE=Key is currently pressed
Example	<code>pTango->GetKey(1, &bKey[0], &bKey[1], &bKey[2], &bKey[3]);</code>

GetKeyLatch	
Description	Get and clear HDI device key states (?keyl).
C++	<code>int LSX_GetKeyLatch (int ILSID, BOOL *pbKey1, BOOL *pbKey2, BOOL *pbKey3, BOOL *pbKey4);</code>
Parameters	Pointers to BOOL, TRUE=Key was or is pressed
Example	<code>pTango->GetKeyLatch(1, &bKey[0], &bKey[1], &bKey[2], &bKey[3]);</code>

ClearKeyLatch	
Description	Clear latched key state(s) of one specified (1-4) or all (0) keys. No bitmask. (!keyl)
C++	<code>int LSX_ClearKeyLatch (int ILSID, int lKey);</code>
Parameters	lKey: 0 = clear latched key state of all 4 keys 1 = clear latched key state of key 1 only 2 = clear latched key state of key 2 only 3 = clear latched key state of key 3 only 4 = clear latched key state of key 4 only
Example	<code>pTango->ClearKeyLatch(1, 0); // Clear all</code>

GetHdiSpeedIndex	
Description	Read the currently used HDI speed index of all axes (?hdisi). The speed index is the currently (by HDI F-key) selected speed level index e.g., from ERGODRIVE or the Multifunction Wheel
C++	<code>int LSX_GetHdiSpeedIndex (int ILSID, int *pLSi1, int *pLSi2, int *pLSi3, int *pLSi4);</code>
Parameters	Pointers to int for returning the speed index
Example	<code>pTango->LSX_GetHdiSpeedIndex (1, &lSpeedIndex);</code>

SetHdiSpeedIndex	
Description	Manipulate the currently used HDI speed index of all axes (!hdisi). The speed index usually is the currently (by HDI F-key) selected speed level index e.g., from ERGODRIVE or the Multifunction Wheel
C++	<code>int LSX_SetHdiSpeedIndex (int ILSID, int lSi1, int lSi2, int lSi3, int lSi4);</code>
Parameters	New speed index for the HDI axes
Example	<code>pTango->LSX_SetHdiSpeedIndex (1, 0, 0, 0, 0); // Set speed index to 0</code>

GetHdiSpeedIndexSingleAxis	
Description	Read the currently used HDI speed index of an axes (?hdisi). The speed index is the currently (by HDI F-key) selected speed level index e.g., from ERGODRIVE or the Multifunction Wheel
C++	<code>int LSX_GetHdiSpeedIndexSingleAxis (int ILSID, int lAxis, int *pLSi);</code>
Parameters	lAxis: As number 1, 2, 3, 4 corresponding to X, Y, Z, A axes lSi: Pointer to int for returning the speed index of the specified axis
Example	<code>pTango->LSX_GetHdiSpeedIndexSingleAxis(1, 2, &lSpeedIndex); // Get Y index</code>

SetHdiSpeedIndexSingleAxis	
Description	Manipulate the currently used HDI speed index of all axes (!hdisi). The speed index usually is the currently (by HDI F-key) selected speed level index e.g., from ERGODRIVE or the Multifunction Wheel
C++	<code>int LSX_SetHdiSpeedIndexSingleAxis (int ILSID, int lAxis, int lSi);</code>
Parameters	lAxis: As number 1, 2, 3, 4 corresponding to X, Y, Z, A axes lSi: New speed index for the HDI axis
Example	<code>pTango->LSX_SetHdiSpeedIndexSingleAxis (1, 2, 0); // Set speed index of Y to 0</code>

4.8. BPZ Console with Trackball and Joyspeed Keys

GetBPZ	
Description	Retrieves status of a custom-built control console with trackball (?bpz).
C++	<code>int LSX_GetBPZ (int ILSID, int *pAValue);</code>
Parameters	AValue: 0 → control console is OFF 1 → control console active, trackball operated at 0,1µm step resolution. 2 → control console active, trackball operated with trackball factor.
Example	<code>pTango->GetBPZ(1, &AValue);</code>

SetBPZ	
Description	Switches custom-built control console on / off (!bpz).
C++	<code>int LSX_SetBPZ (int ILSID, int lAValue);</code>
Parameters	AValue: 0...2 0 → switch control console OFF 1 → activate control console and operate trackball at 0,1µm step resolution. 2 → activate control console and operate trackball with trackball factor.
Example	<code>pTango->SetBPZ(1, 1);</code>

GetBPZJoyspeed	
Description	Retrieves custom-built control console Joystick speed (?joyspeed).
C++	<code>int LSX_GetBPZJoyspeed (int ILSID, int lAPar, double *pdAValue);</code>
Parameters	APar: 1, 2 or 3 (console keys for speed selection: slow, medium, fast) AValue: max. speed [r/sec]
Example	<code>pTango->GetBPZJoyspeed(1, &AValue); // retrieve set speed of key 1 (slow)</code>

SetBPZJoyspeed	
Description	Set custom-built control console joystick speed (!joyspeed).
C++	<code>int LSX_SetBPZJoyspeed (int ILSID, int lAPar, double dAValue);</code>
Parameters	APar: 1, 2 or 3 (console keys for speed selection: slow, medium, fast) AValue: ±max. speed [r/sec]
Example	<code>pTango->SetBPZJoyspeed(1, 1, 25); // Set key 1 parameter (slow) to speed 25</code>

GetBPZTrackballBackLash

Description	Retrieves custom-built control console trackball backlash (?bpzbl).
C++	<code>int LSX_GetBPZTrackballBackLash (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	X, Y, Z A: backlash [mm]
Example	<code>pTango->GetBPZTrackballBackLash(1, &X, &Y, &Z, &A);</code>

SetBPZTrackballBackLash

Description	Set custom-built control console trackball backlash (!bpzbl).
C++	<code>int LSX_SetBPZTrackballBackLash (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: 0.001 to 0.15 mm
Example	<code>pTango->SetBPZTrackballBackLash(1, 0.01, 0.01, 0.01, 0.01); // Set backlash for all axes to 10µm</code>

GetBPZTrackballFactor

Description	Retrieves control console trackball factor (?bpztf).
C++	<code>int LSX_GetBPZTrackballFactor (int ILSID, double *pdAValue);</code>
Parameters	AValue: Trackball factor e.g. AValue of 3 means that one trackball pulse results in 3 motor increments.
Example	<code>pTango->GetBPZTrackballFactor(1, &AValue);</code>

SetBPZTrackballFactor

Description	Set custom-built control console trackball factor (!bpztf).
C++	<code>int LSX_SetBPZTrackballFactor (int ILSID, double dAValue);</code>
Parameters	AValue: 0.01 ... 100 AValue = 1 → Trackball factor = 1, i.e. one trackball impulse results in one motor increment
Example	<code>pTango->SetBPZTrackballFactor(1, 1,0);</code>

4.9. Limit Switches (Hardware and Software)

GetAutoLimitAfterCalibRM	
Description	Provides, whether internal software limits are set when calibrating 'cal' or measuring stage travel range 'rm' (?nosetlimit).
C++	<code>int LSX_GetAutoLimitAfterCalibRM (int ILSID, int *plFlags);</code>
Parameters	Flags: Bit mask: Bit0=X, Bit1=Y, Bit2=Z, Bit3=A Bit 0 = 1 → no travel range limits are set from X-Axis calibration or range measure Bit 1 = 0 → software limits are set for Y-Axis (cal/rm)
Example	<code>pTango->GetAutoLimitAfterCalibRM(1, &Flags);</code>

SetAutoLimitAfterCalibRM	
Description	Prevents setting of internal software limits when calibrating or measuring travel range (!nosetlimit).
C++	<code>int LSX_SetAutoLimitAfterCalibRM (int ILSID, int lFlags);</code>
Parameters	Flags: Bit mask: Bit0=X, Bit1=Y, Bit2=Z, Bit3=A Bit 0 = 1 → no travel range limits are set from X-Axis calibration or range measure Bit 1 = 0 → software limits are set for Y-Axis (cal/rm)
Example	<code>pTango->SetAutoLimitAfterCalibRM(1, Flags);</code>

GetLimit	
Description	Provides soft travel range limits, as set by SetLimit or cal+rm (?lim).
C++	<code>int LSX_GetLimit (int ILSID,</code> <code>int lAxis,</code> <code>double *pdMinRange,</code> <code>double *pdMaxRange);</code>
Parameters	Axis: Axis from which travel range limits are to be retrieved (as number 1, 2, 3, 4 corresponding to X, Y, Z, A axes) MinRange: lower travel range limit, unit depends on dimension MaxRange: upper travel range limit, unit depends on dimension
Example	<code>pTango->GetLimit(1, &MinRange, &MaxRange);</code>

SetLimit	
Description	Set soft travel range limits (!lim).
C++	<code>int LSX_SetLimit (int lLSID, int lAxis, double dMinRange, double dMaxRange);</code>
Parameters	Axis: Axis from which travel range limits are to be retrieved (as number 1, 2, 3, 4 corresponding to X, Y, Z, A axes) MinRange: lower travel range limit, unit depends on dimension MaxRange: upper travel range limit, unit depends on dimension
Example	<pre>pTango->SetLimit(1, 1, -10.0, 20.0); // assign X-Axis -10 as lower and 20 as upper travel range limits</pre>

GetLimitControl	
Description	Retrieves, whether area control (limits) is enabled or ignored (?limctr).
C++	<code>int LSX_GetLimitControl (int lLSID, int lAxis, BOOL *pbActive);</code>
Parameters	Axis: As number 1, 2, 3, 4 corresponding to X, Y, Z, A axes Active: TRUE = area control of corresponding axis is active FALSE = area control of corresponding axis is deactivated
Example	<code>pTango->GetLimitControl(1, 3, &Active); // Read limit control enable state of Z-Axis</code>

SetLimitControl	
Description	Switches area control on / off (!limctr).
C++	<code>int LSX_SetLimitControl (int lLSID, int lAxis, BOOL bActive);</code>
Parameters	Axis: As number 1, 2, 3, 4 corresponding to X, Y, Z, A axes Active: TRUE = activate area control of corresponding axis FALSE = disable area control of corresponding axis (no limits checked)
Example	<code>pTango->SetLimitControl(1, 2, TRUE); // Enable area control of Y-Axis is active</code>

GetSwitchActive	
Description	Provides, whether hardware limit switches are enabled (?swact).
C++	<code>int LSX_GetSwitchActive (int ILSID, int *plXA, int *plYA, int *plZA, int *plAA);</code>
Parameters	A bit mask is supplied for each axis: Bit 0 → zero limit switch (cal, “E0”) Bit 1 → reference limit switch (unused) Bit 2 → end limit switch (rm, “EE”) The limit switch is enabled if the corresponding bit is set.
Example	<code>pTango->GetSwitchActive(1, &XA, &YA, &ZA, &AA);</code>

SetSwitchActive	
Description	Switches limit switches on / off (!swact).
C++	<code>int LSX_SetSwitchActive (int ILSID, int lXA, int lYA, int lZA, int lAA);</code>
Parameters	A bit mask is supplied for each axis: Bit 0 → zero limit switch (cal, “E0”) Bit 1 → reference limit switch (unused) Bit 2 → end limit switch (rm, “EE”) The limit switch is enabled if the corresponding bit is set.
Example	<code>pTango->SetSwitchActive(1, 7, 1, 5, 0);</code> <i>// X-Axis: All limit switches enabled, Y-Axis: Only Zero limit switch enabled,</i> <i>// Z-Axis: E0 and EE switches enabled (default,) A-Axis: All limit switches ignored</i>

GetSwitchPolarity	
Description	Retrieves polarity of limit switches (?swpol).
C++	<code>int LSX_GetSwitchPolarity (int ILSID, int *plXP, int *plYP, int *plZP, int *plAP);</code>
Parameters	A bit mask is supplied for each axis: Bit 0 → zero limit switch (cal, “E0”) Bit 1 → reference limit switch (unused) Bit 2 → end limit switch (rm, “EE”) If bit is set (1), the corresponding switch is interpreted active when high. If bit is reset (0), the corresponding switch is active low.
Example	<code>pTango->GetSwitchPolarity(1, &XP, &YP, &ZP, &AP);</code>

SetSwitchPolarity

Description	Sets polarity of limit switches (!swpol).
C++	<code>int LSX_SetSwitchPolarity (int ILSID, int IXP, int IYP, int IZP, int IAP);</code>
Parameters	A bit mask is supplied for each axis: Bit 0 → zero limit switch (cal, “E0”) Bit 1 → reference limit switch (unused) Bit 2 → end limit switch (rm, “EE”) If bit is set (1), the corresponding switch is interpreted active when high. If bit is reset (0), the corresponding switch is active low.
Example	<pre>pTango->SetSwitchPolarity(1, 7, 0, 0, 0); // all limit switches of X-Axis are high active, // all limit switches of Y-, Z- and A-Axis are low active</pre>

GetSwitchType

Description	Retrieves type of limit switches (?swtyp).
C++	<code>int LSX_GetSwitchType (int ILSID, int *plXP, int *plYP, int *plZP, int *plAP);</code>
Parameters	A bit mask is supplied for each axis: Bit 0 → zero limit switch (cal, “E0”) Bit 1 → reference limit switch (unused) Bit 2 → end limit switch (rm, “EE”) If bit is set (1), input is for NPN type limit switch. If bit is reset (0), input is for PNP type limit switch (default).
Example	<code>pTango->GetSwitchType(1, &XP, &YP, &ZP, &RP);</code>

SetSwitchType

Description	Sets type of limit switches (!swtyp).
C++	<code>int LSX_SetSwitchType (int ILSID, int IXP, int IYP, int IZP, int IAP);</code>
Parameters	A bit mask is supplied for each axis: Bit 0 → zero limit switch (cal, “E0”) Bit 1 → reference limit switch (unused) Bit 2 → end limit switch (rm, “EE”) If bit is set (1), input is configured for NPN type limit switch using pull-up resistor. If bit is reset (0), input is configured for PNP type limit switch with pull down resistor (default).
Example	<code>pTango->SetSwitchType(1, XP, YP, ZP, AP);</code>

GetSwitches

Description	Retrieves actuation status of all limit switches (?readsw). The 4 bits of the "Ref" switch are only used in systems with center reference.																							
C++	int LSX_GetSwitches (int ILSID, int *plFlags);																							
Parameters	Flags: Pointer on Integer Value, which includes status of all limit switches as bit mask In bit mask, status of limit switches is encoded as follows: <table><tr><td>Limit switch</td><td>EE (rm)</td><td>Ref.</td><td>E0 (cal)</td></tr><tr><td>Axis</td><td>AZYX</td><td>AZYX</td><td>AZYX</td></tr><tr><td>Bits MSB→LSB:</td><td>0000</td><td>0000</td><td>0000</td></tr></table> <table><tr><td>Example: 0x203 =</td><td>0010</td><td>0000</td><td>0011</td></tr><tr><td colspan="4">→ EE of Y-Axis is actuated, E0 of X- and Y-Axis are actuated</td></tr></table>				Limit switch	EE (rm)	Ref.	E0 (cal)	Axis	AZYX	AZYX	AZYX	Bits MSB→LSB:	0000	0000	0000	Example: 0x203 =	0010	0000	0011	→ EE of Y-Axis is actuated, E0 of X- and Y-Axis are actuated			
Limit switch	EE (rm)	Ref.	E0 (cal)																					
Axis	AZYX	AZYX	AZYX																					
Bits MSB→LSB:	0000	0000	0000																					
Example: 0x203 =	0010	0000	0011																					
→ EE of Y-Axis is actuated, E0 of X- and Y-Axis are actuated																								
Example	pTango->GetSwitches(1, &Flags);																							

4.10. Digital and Analog Inputs and Outputs

GetAnalogInput	
Description	Retrieves current A/D conversion result of an analogue channel (?anain). Each TANGO controller might have its individual channels to read. Check the TANGO Instruction Set for channel numbers and their assignment.
C++	<code>int LSX_GetAnalogInput (int ILSID, int IIndex, int *pIValue);</code>
Parameters	Index: 0...15 (analog channel), 0...9 = HDI connector, pins 1...10 10 = ANAIN0 of AUX-IO connector Value: Pointer to Integer value, to which the channel's A/D conversion result is written. 0...5V analog = 0...1023
Example	<code>pTango->GetAnalogInput(1, 0, &Input); // Read channel 0</code>

SetAnalogOutput	
Description	Set analogue output signals (!anaout).
C++	<code>int LSX_SetAnalogOutput (int ILSID, int IIndex, int IValue);</code>
Parameters	Index: 0, 1 (analogue output index) Value: 0...100 [%]
Example	<code>pTango->SetAnalogOutput(1, 0, 100);</code> <i>// set analogue output 0 to max. voltage (10V) (or 0 to 5V for Tango mini 3)</i>

SetLedBright	
Description	Set the brightness of the LED100 illumination, when connected in the default configuration (ANOUT0 and TAKT_OUT) to the AUX I/O or AUX mini port (!adigout + !anaout). The SetLedBright function also controls the TAKT_OUT digital pin in order to entirely switch of LED100 with the LED-DR1 driver.
C++	<code>int LSX_SetLedBright (int ILSID, double dBright);</code>
Parameters	dBright: Brightness of the LED100 -1 = OFF A negative value <0 switches the LED entirely off (digital pin) 0 ... 100 Brightness in %, up to 3 fractional digits supported
Example	<code>pTango->SetLedBright(1, -1); // set led off</code> <code>pTango->SetLedBright(1, 0); // set led to lowest possible brightness</code> <code>pTango->SetLedBright(1, 12.345); // set led to 12.345% brightness</code> <code>pTango->SetLedBright(1, 100); // set led to max. brightness</code>

GetAnalogOutputMode	
Description	Read the AUX-IO analog output mode (?anamode).
C++	<code>int LSX_GetAnalogOutputMode (int ILSID, int *plMode);</code>
Parameters	Mode: int pointer to return the anamode setting (0 ... 5)
Example	<code>pTango->GetAnalogOutputMode(1, 0, &Mode); // Read AnaMode</code>

SetAnalogOutputMode	
Description	Set the AUX-IO analog output mode (!anamode).
C++	<code>int LSX_SetAnalogOutputMode (int ILSID, int lMode);</code>
Parameters	Mode: Integer number of the desired AnaMode (0 ... 5)
Example	<code>pTango->SetAnalogOutputMode(1, 5); // Activate AnaMode 5</code>

SetAuxDigitalOutput	
Description	Set the specified digital output pin of the AUX-I/O port (!adigout).
C++	<code>int LSX_SetAuxDigitalOutput (int ILSID, int lIndex, BOOL bValue);</code>
Parameters	Index: 0 to 3 for the output to set TANGO 3 mini: 0 = Bit 0: AUX mini Pin 6 (TAKT_OUT, default LED100 on/off pin) 1 = Bit 1: AUX mini Pin 7 (VR_OUT) 2 = Bit 2: AUX mini Pin 8 (SHUTTER_OUT) 3 = Bit 3: AUX mini Pin 9 (TRIGGER_OUT) Other TANGO controllers: 0 = Bit 0: AUX I/O Pin 5 (TAKT_OUT, default LED100 on/off pin) 1 = Bit 1: AUX I/O Pin 6 (VR_OUT) 2 = Bit 2: AUX I/O Pin 7 (SHUTTER_OUT) 3 = Bit 3: AUX I/O Pin 8 (TRIGGER_OUT) Value: FALSE = set pin to low TRUE = set pin to high
Example	<code>pTango->SetAuxDigitalOutput(1, 0, TRUE); // set output 0 to high</code>

GetAuxDigitalInput	
Description	Get state of the specified digital input of the AUX-I/O port (?adigin).
C++	<code>int LSX_GetAuxDigitalInput (int ILSID, int IIndex, BOOL *bValue);</code>
Parameters	Index: 0 to 3 for the digital input pin TANGO 3 mini: 0 = Bit 0: AUX mini Pin 1 (TAKT_IN) 1 = Bit 1: Motor Connector Pin 7 (TRIN1) 2 = Bit 2: [not available] 3 = Bit 3: AUX mini Pin 2 (SNAPSHOT_IN) Other TANGO controllers: 0 = Bit 0: AUX I/O Pin 1 (TAKT_IN) 1 = Bit 1: AUX I/O Pin 2 (VR_IN) 2 = Bit 2: AUX I/O Pin 3 (STOP) 3 = Bit 3: AUX I/O Pin 4 (SNAPSHOT2) Value: Pin level FALSE = low TRUE = high
Example	<code>pTango->GetAuxDigitalInput(1, 3, &bState); // get input 3 state</code>

GetDigitalInputs	
Description	Retrieve signal level of all 24 “I/O1 extension” digital input pins (?digin).
C++	<code>int LSX_GetDigitalInputs (int ILSID, int *plValue);</code>
Parameters	Value: Pointer to Integer value, to which the status of all inputs is written (as bit mask). LSB = Digital input 0
Example	<pre>int inputs; pTango->GetDigitalInputs(1, &inputs); if (Inputs & 16) ... // if input 4 is set ...</pre>

GetDigitalInputsE	
Description	Retrieve signal level of all 12 “IO2 / Multi-IO” digital inputs (?edigin).
C++	<code>int LSX_GetDigitalInputsE (int ILSID, int *plValue);</code>
Parameters	Value: Pointer on a 32-Bit Integer, which returns the inputs 16...31 in the bits 0...15
Example	<pre>int ext_inputs; pTango->GetDigitalInputsE(1, &ext_inputs);</pre>

SetDigitalOutput

Description	Set individual digital output pin of IO1 extension (!digout).
C++	<code>int LSX_SetDigitalOutput (int ILSID, int IIndex, BOOL bValue);</code>
Parameters	Index: 0 to 7 Value: Set pin level to FALSE = low TRUE = high
Example	<code>pTango->SetDigitalOutput(1, 2, TRUE); // set output pin 2 to '1'</code>

SetDigitalOutputs

Description	Set all digital output pins (0-7) of the I/O1 extension port (!digout).
C++	<code>int LSX_SetDigitalOutputs (int ILSID, int IValue);</code>
Parameters	Value: Bit mask, bits 0-7 determine value that is set for outputs 0-7
Example	<code>pTango->SetDigitalOutputs(1, 3); // 3 = set outputs 0 and 1 to 1, remaining pins to 0</code>

SetDigitalOutputE

Description	Set individual digital output pin of Multi I/O / IO2 port (!edigout).
C++	<code>int LSX_SetDigitalOutputE (int ILSID, int IIndex, BOOL bValue);</code>
Parameters	Index: 0 to 7 Value: Set pin level to FALSE = low TRUE = high
Example	<code>pTango->SetDigitalOutputE(1, 2, TRUE); // set output pin 2 to '1'</code>

SetDigitalOutputsE

Description	Set digital outputs of the TANGO PCI-E or Desktop Multi I/O / IO2 port (!edigout).
C++	<code>int LSX_SetDigitalOutputsE (int ILSID, int IValue);</code>
Parameters	Value: Bit mask, bits 0-7 determine value that is set for outputs 0-7
Example	<code>pTango->SetDigitalOutputsE(1, 5); // 5 = set outputs 0 and 2 to 1, remaining pins to 0</code>

SetDigIO_Distance	
Description	NOT SUPPORTED BY TANGO Function of digital inputs / outputs. Activate an output depending on preset distance before or after reaching designated position.
C++	int LSX_SetDigIO_Distance (int lLSID, int lIndex, BOOL bFkt, double dDist, int lAxis);
Parameters	Index: 0 to 15 (output pin) Fkt = FALSE → activation of an output depending on set distance before reaching determined position Fkt = TRUE → activation of an output depending on set distance after start position Dist: Distance, depends on selected dimension (unit) Axis: Axis number 1, 2, 3, 4 corresponding to X, Y, Z, A axes
Example	pTango->SetDigIO_Distance(1, 7, FALSE, 78.9, 3); // output 7 is activated 78.9mm before reaching final position (Z-Axis)

SetDigIO_EmergencyStop	
Description	NOT SUPPORTED BY TANGO Function of digital inputs / outputs. Assignment of Emergency-Stop pin functionality.
C++	int LSX_SetDigIO_EmergencyStop (int lLSID, int lIndex);
Parameters	Index: 0 to 15 (input/output)
Example	pTango->SetDigIO_EmergencyStop(1, 15); // Pin 15 is used for Emergency-Stop

SetDigIO_Off	
Description	NOT SUPPORTED BY TANGO Switch off digital inputs / outputs function. (Does not affect inputs / outputs states).
C++	int LSX_SetDigIO_Off (int lLSID, int lIndex);
Parameters	Index: 0 to 15 (individual Input/Output pins), 16 (all 16 port pins)
Example	pTango->SetDigIO_Off(1, 0); // Function of I/O pin 0 is switched 'Off'

SetDigIO_Polarity	
Description	NOT SUPPORTED BY TANGO Set polarity of digital inputs / outputs (!digfkt).
C++	int LSX_SetDigIO_Polarity (int lLSID, int lIndex, BOOL bHigh);
Parameters	<i>Index</i> : 0 to 15 (individual I/O pin), 16 (all 16 port pins) <i>High</i> = TRUE → high active <i>High</i> = FALSE → low active
Example	pTango->SetDigIO_Polarity(1, 3, TRUE); // input pin / output pin 3 high active

4.11. TVR Clock & Direction IO

GetTVRMode	
Description	Retrieve the TVR mode of the clock & direction IO (?tvr)
C++	<code>int LSX_GetTVRMode (int lLSID, int *pIValue);</code>
Parameters	Value: Pointer to a 32-Bit Integer, which returns the TVR mode
Example	<pre>int mode; pTango->GetTVRMode (1, &mode);</pre>

SetTVRMode	
Description	Set the TVR mode of the clock & direction IO (!tvr)
C++	<code>int LSX_SetTVRMode (int lLSID, int lMode1, int lMode2, int lMode3, int lMode4);</code>
Parameters	Value: Required TVR mode of the axes 0 = disabled (1) = enabled without tvrf factor (2) = enabled with tvrf factor (3) = enabled without tvrf factor, requires ext. start/stop (4) = enabled with tvrf factor, requires ext. start/stop 5 = enabled with tvrjoyf factor
Example	<code>pTango->SetTVRMode(1, 0, 0, 5, 0); // only set tvr mode 5 for Z</code>

GetFactorTVR	
Description	Retrieve the TVR factor for TVR modes 1-4 (?tvrf)
C++	<code>int LSX_GetFactorTVR (int lLSID, double *pdVal1, double *pdVal2, double *pdVal3, double *pdVal4);</code>
Parameters	Val1...3: Pointers to double, which hold the returned TVR factors
Example	<pre>double factor[4]; pTango->GetFactorTVR (1, &factor[0] , &factor[1] , &factor[2] , &factor[3]);</pre>

SetFactorTVR	
Description	Set the TVR factor for TVR modes 1-4 (!tvrf)
C++	<code>int LSX_SetFactorTVR (int lLSID, double dVal1, double dVal2, double dVal3, double dVal4);</code>
Parameters	Val1...Val4: Required TVR factor for all axes 1...4
Example	<code>pTango->SetFactorTVR (1, 0.2 0.2 0.05, 50);</code>

4.12. Encoder Settings

ClearEncoder	
Description	Reset encoder TTL counter to zero (!clearhwcount).
C++	<code>int LSX_ClearEncoder (int lLSID, int lAxis);</code>
Parameters	Axis: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes)
Example	<code>pTango->ClearEncoder(1, 2); // reset encoder counter of Y-Axis to zero</code>

GetEncoder	
Description	Retrieves all encoder TTL counter positions (?hwcount).
C++	<code>int LSX_GetEncoder (int lLSID, double *pdXP, double *pdYP, double *pdZP, double *pdAP);</code>
Parameters	XP, YP, ZP, AP: Counter values, 4x interpolated
Example	<code>pTango->GetEncoder(1, &XP, &YP, &ZP, &AP);</code>

GetEncoderActive	
Description	Retrieves which encoder will be activated after calibration (?encmask). Please note: This function is corresponding to the „?encmask“ command! Not “?enc”.
C++	<code>int LSX_GetEncoderActive (int lLSID, int *plFlags);</code>
Parameters	Flags: Encoder mask (flags) Bit 0 = X encoder will be activated Bit 1 = Y encoder will be activated Bit 2 = Z encoder will be activated
Example	<code>pTango->GetEncoderActive(1, &Flags);</code>

SetEncoderActive	
Description	Sets which encoder is activated after calibration (!encmask) Please note: This function is corresponding to „!encmask“ command, not “!enc”.
C++	<code>int LSX_SetEncoderActive (int ILSID, int IFlags);</code>
Parameters	Value: Encoder mask (flags) Bit 0 = X encoder will be activated Bit 1 = Y encoder will be activated Bit 2 = Z encoder will be activated
Example	<code>pTango->SetEncoderActive(1, 0); // No encoder will be used</code> <code>pTango->SetEncoderActive(1, 2); // encoder of Y-Axis will be activated after calibration</code>

GetEncoderMask	
Description	Retrieve status of encoders (?enc). Please note: This function is corresponding to „?enc“ command, not “?encmask”!
C++	<code>LSX_GetEncoderMask (int ILSID, int *pIFlags);</code>
Parameters	Flags: Active encoder mask (flags) Bit 0 = X encoder is active / inactive Bit 1 = Y encoder is active / inactive Bit 2 = Z encoder is active / inactive
Example	<code>int EncMask;</code> <code>pTango->GetEncoderMask(1, &EncMask);</code> <code>if (EncMask & 2) ...</code> <code>// if encoder of Y-Axis connected + active ...</code>

SetEncoderMask	
Description	Activates / deactivates encoders manually (!enc). Please note: This function is corresponding to „!enc“ command, not “!encmask”! Do not use in closed loop. Encoders should always be activated with Calibrate command.
C++	<code>int LSX_SetEncoderMask (int ILSID, int IValue);</code>
Parameters	Value: Active encoder mask (flags) Bit 0 = (activate)/deactivate X encoder Bit 1 = (activate)/deactivate Y encoder Bit 2 = (activate)/deactivate Z encoder
Example	<code>pTango->SetEncoderMask(1, 0); // deactivate all encoders</code> <code>pTango->SetEncoderMask(1, 2); // deactivate X and Z encoders, activate Y-Axis encoder</code>

GetEncoderSingleAxis

Description	Read the encoder settings of Encoder Type, Signal Period and Reference Signal of the specified axis (?enctype, ?encref, ?encperiod).
C++	<pre>int LSX_GetEncoderSingleAxis (int lLSID, int *plAxis, double *pdPeriod, int *plReference);</pre>
Parameters	lAxis: The axis number 1-4 dPeriod: Pointer to return the encoder signal period length [mm] dReference: Pointer to return the usage of encoder reference (0 or 1)
Example	<code>pTango->GetEncoderSingleAxis(1, 3, &lEncType, &dPeriod, &lReference); // axis 3(Z)</code>

SetEncoderSingleAxis

Description	Set the Encoder Type, Signal Period and availability of Reference Signal of the specified axis (!enctype, !encref, !encperiod).
C++	<pre>int LSX_SetEncoderSingleAxis (int lLSID, int lAxis, double dPeriod, int lReference);</pre>
Parameters	lAxis: The axis number 1-4 dPeriod: The encoder signal period length [mm] dReference: Usage of encoder reference signal (0=no signal or 1=has a ref signal)
Example	<code>pTango->SetEncoderSingleAxis(1, 3, 2, 0.02, 0); // axis 3(Z) = Type 2, 20µm, no ref</code>

GetEncoderPeriod	
Description	Retrieves encoder signal period length (?encperiod). A NULL pointer can be used for not required axes.
C++	<code>int LSX_GetEncoderPeriod (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	X, Y, Z, A: Period length [mm]
Example	<code>double X,Y,Z,A; pTango->GetEncoderPeriod(1, &X, &Y, &Z, &A); pTango->GetEncoderPeriod(1, &X, &Y, NULL, NULL); // Ignore Z+A axes</code>

SetEncoderPeriod	
Description	Set encoder signal period length (!encperiod).
C++	<code>int LSX_SetEncoderPeriod (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: 0.0001 – 4 mm
Example	<code>pTango->SetEncoderPeriod(1, 0.5, 0.5, 0.5, 0.5); // set encoder signal period of all axes to 0.5mm</code>

GetEncoderPosition	
Description	Retrieves position response type (?encpos).
C++	<code>int LSX_GetEncoderPosition (int ILSID, BOOL *pbValue);</code>
Parameters	Value: TRUE → axis position values will be read from the encoder, if activated. (If no encoders are active, the position is taken from the motor) FALSE → Position will be taken from the motor position only.
Example	<code>pTango->GetEncoderPosition(1, &Value);</code>

SetEncoderPosition	
Description	Select readout of encoder- or motor-related position values (!encpos).
C++	<code>int LSX_SetEncoderPosition (int ILSID, BOOL bValue);</code>
Parameters	Value: TRUE → axis position values will be read from the encoder, if activated. (If no encoders are active, the position is taken from the motor) FALSE → Position will be taken from the motor position.
Example	<code>pTango->SetEncoderPosition(1, TRUE);</code>

GetEncoderRefSignal

Description	Retrieves whether the encoder reference signal is used when calibrating (?encref).
C++	<code>int LSX_GetEncoderRefSignal (int ILSID, int *plXR, int *plYR, int *plZR, int *plAR);</code>
Parameters	1 → encoder reference signal is evaluated while calibrating 0 → reference signal is not evaluated, zero position is set at the CAL end switch
Example	<code>pTango->GetEncoderRefSignal(1, &X, &Y, &Z, &A);</code>

SetEncoderRefSignal

Description	Use reference signal from encoder when calibrating (!encref).
C++	<code>int LSX_SetEncoderRefSignal (int ILSID, int lXR, int lYR, int lZR, int lAR);</code>
Parameters	XR, YR, ZR, AR: 0 (encoder reference signal is evaluated while calibrating) or 1 (reference signal is not evaluated, zero position is set at the CAL end switch)
Example	<code>pTango->SetEncoderRefSignal(1, 1, 1, 0, 0);</code> <i>// when calibrating, reference signals of encoders X and Y are evaluated</i>

GetRefSpeed

Description	Old method for reading the velocity for calibrating to the encoder reference mark or to the reference switch (?calrefspeed). It is recommended to use ?encrefvel.
C++	<code>int LSX_GetRefSpeed (int ILSID, int *plSpeed);</code>
Parameters	Pointer to int for returning the velocity value (in 1/100 revolutions/s, one for all axes)
Example	<code>pTango->GetRefSpeed (1, &Speed);</code>

SetRefSpeed

Description	Old method of setting the velocity for calibrating to the encoder reference mark or to the reference switch (!calrefspeed). It is recommended to use !encrefvel.
C++	<code>int LSX_SetRefSpeed (int ILSID, int lSpeed);</code>
Parameters	Velocity in 1/100 motor revolutions/s, one velocity applies to all axes
Example	<code>pTango->SetRefSpeed (1, 50);</code> <i>// search the reference switch (or the reference mark)</i> <i>search velocity of all axes to 50/100 motor revolutions/s (=0.5 rev/s)</i>

4.13. Closed Loop Settings

GetController	
Description	Retrieve Closed Loop mode (?ctr).
C++	<code>int LSX_GetController (int ILSID, int *plXC, int *plYC, int *plZC, int *plRC);</code>
Parameters	Controller mode XC, YC, ZC, AC: 0 → controller „OFF“ 1 → controller „OFF after reaching target position“ 2 → controller „Always ON“ 3 → controller „OFF after reaching designated end position“ with current reduction 4 → controller „Always ON“ with current reduction
Example	<code>pTango->GetController(1, &X, &Y, &Z, &A);</code>

SetController	
Description	Set Closed Loop mode (!ctr).
C++	<code>int LSX_SetController (int ILSID, int lXC, int lYC, int lZC, int lAC);</code>
Parameters	Controller mode XC, YC, ZC, AC: 0 → controller „OFF“ 1 → controller „OFF after reaching target position“ 2 → controller „Always ON“ 3 → controller „OFF after reaching designated end position“ with current reduction 4 → controller „Always ON“ with current reduction
Example	<code>pTango->SetController(1, 2, 2, 0, 0); // Enable permanent closed loop for X and Y axes</code>

GetControllerCall	
Description	Read Closed Loop interval time (?ctrc). Should remain at the default setting (3 or 5 ms).
C++	<code>int LSX_GetControllerCall (int ILSID, int *plCtrCall);</code>
Parameter:	CtrCall: Controller call time [ms] (default 3 or 5ms, depending on the TANGO type)
Example	<code>pTango->GetControllerCall(1, &CtrCall);</code>

SetControllerCall	
Description	Set Closed Loop interval time (!ctrc). Applies to all axes. The interval, in which the closed loop processes the deviation. Should remain at the controller default setting (3 or 5 ms).
C++	<code>int LSX_SetControllerCall (int ILSID, int lCtrCall);</code>
Parameters	CtrCall: Controller call time [ms]
Example	<code>pTango->SetControllerCall(1, 5);</code> <i>// CtrCall = 5 means: Closed Loop controller is called every 5 milliseconds</i>

GetControllerFactor	
Description	FOR COMPATIBILITY ONLY! Retrieve Closed Loop controller factors (?ctrf). Note: The TANGO supports 2 factors per axis (idle & move), therefore it is highly recommended to use GetControllerFactorSingleAxis() !
C++	<code>int LSX_GetControllerFactor (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	X, Y, Z, A: Closed Loop factors
Example	<code>pTango->GetControllerFactor(1, &X, &Y, &Z, &A);</code>

SetControllerFactor	
Description	FOR COMPATIBILITY ONLY! Set Closed Loop controller factor (!ctrf). Note: The TANGO supports 2 factors per axis (idle & move), therefore it is highly recommended to use SetControllerFactorSingleAxis() !
C++	<code>int LSX_SetControllerFactor (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: Position difference amplification factor 1 – 64
Example	<code>pTango->SetControllerFactor(1, 2, 2, 2, 0);</code> <i>//Closed Loop amplification is set to 2 for X, Y and Z axes</i>

GetControllerFactorSingleAxis	
Description	Retrieve Closed Loop controller factors of the specified axis (?ctrff). Note: The TANGO supports 2 factors per axis (idle & move)
C++	<code>int LSX_GetControllerFactorSingleAxis (int ILSID, int lAxis, double * pdFidle, double * dFmove);</code>
Parameters	<i>lAxis</i> = Axis number 1,2,3,4 (for axes X,Y,Z,A) <i>dFidle</i> = returns 0.0 ...25.4 <i>dFmove</i> = returns 0.0 ...25.4
Example	<code>pTango->GetControllerFactorSingleAxis(1, 2, &idlefactor, &movefactor); // Read Y</code>

SetControllerFactorSingleAxis	
Description	Set Closed Loop controller factor (!ctrff). The TANGO supports 2 factors per axis (idle & move).
C++	<code>int LSX_SetControllerFactorSingleAxis (int ILSID, int lAxis, double dFidle, double dFmove);</code>
Parameters	<i>lAxis</i> = Axis number 1,2,3,4 (for axes X,Y,Z,A) <i>dFidle</i> = 0.0 ...25.4 <i>dFmove</i> = 0.0 ...25.4
Example	<code>pTango->SetControllerFactorSingleAxis(1, 3, 8, 2.5, 0); // Set Closed Loop amplification for Z to 8 at idle and 2.5 at move</code>

GetControllerSteps	
Description	Retrieves length of controller steps (ctr). The TANGO uses ctrs as the “Lock-In Range”, where after exceeding a certain behavior can be specified with ctrfm.
C++	<code>int LSX_GetControllerSteps (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	X, Y, Z, A: Lock-In Range [mm]
Example	<code>pTango->GetControllerSteps(1, &X, &Y, &Z, &A);</code>

SetControllerSteps	
Description	Set controller steps (!ctr). The TANGO uses ctrs as the “Lock-In Range”, where after exceeding a certain behavior can be specified with ctrfm.
C++	<code>int LSX_SetControllerSteps (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: Lock-In Range, >= 0.01 [mm]
Example	<code>pTango->SetControllerSteps(1, 4, 5, 7, 9);</code>

GetControllerTimeout	
Description	Read the closed loop control timeout (?ctr). The maximum wait for TWI timeout.
C++	<code>int LSX_GetControllerTimeout (int ILSID, int *plACtrTimeout);</code>
Parameters	ActrTimeout: Timeout [ms], If the Closed Loop controller is unable to settle in the target window for this time, the move is aborted (move function calls return with error code 4013).
Example	<code>pTango->GetControllerTimeout(1, &ActrTimeout);</code>

SetControllerTimeout	
Description	Set the closed loop control timeout (!ctr). The maximum wait for TWI timeout.
C++	<code>int LSX_SetControllerTimeout (int ILSID, int lACtrTimeout);</code>
Parameters	ActrTimeout: Timeout 0 – 10000 ms, If the Closed Loop controller is unable to settle in the target window for this time, the move is aborted (move function calls return with error code 4013). This time should be set longer than the target window delay (TWDelay).
Example	<code>pTango->SetControllerTimeout(1, 500);</code> <i>// Abort after trying to settle in the target window for 500ms</i>

GetControllerTWDelaySingleAxis

Description	Read controller delay (?ctrd). Time to remain in TWI until “reached” state is assigned. For individual axes.
C++	<code>int LSX_GetControllerTWDelaySingleAxis (int lLSID, int lAxis, int *plCtrTWDelay);</code>
Parameters	Axis: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) CtrTWDelay: Controller delay [ms]
Example	<code>pTango->GetControllerTWDelaySingleAxis (1, 3, &CtrTWDelay); // Read Z</code>

SetControllerTWDelaySingleAxis

Description	Set controller delay (!ctrd). Time to remain in TWI until “reached” state is assigned. For individual axes.
C++	<code>int LSX_SetControllerTWDelay (int lLSID, int lAxis, int lCtrTWDelay);</code>
Parameters	Axis: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) CtrTWDelay: Controller delay 0 – 250 ms Time for which the axis must remain in the target window. Moves are delayed by at least this time.
Example	<code>pTango->SetControllerTWDelaySingleAxis(1, 3, 0);</code> <i>// controller delay in Z switched off, closed loop end position will be inaccurate</i>

GetControllerTWDelay

Description	FOR COMPATIBILITY ONLY! Read controller delay (?ctrd). Time to remain in TWI until “reached” state is assigned. OLD Function! As the TANGO allows to set ctrd for individual axes, use GetControllerTWDelaySingleAxis() or ctrd by SendString.
C++	<code>int LSX_GetControllerTWDelay (int lLSID, int *plCtrTWDelay);</code>
Parameters	CtrTWDelay: Controller delay [ms]
Example	<code>pTango->GetControllerTWDelay(1, &CtrTWDelay);</code>

SetControllerTWDelay

Description	FOR COMPATIBILITY ONLY! Set controller delay (!ctrd). Time to remain in TWI until “reached” state is assigned. OLD Function! As the TANGO allows to set ctrd for individual axes, use SetControllerTWDelaySingleAxis() or ctrd by SendString.
C++	<code>int LSX_SetControllerTWDelay (int lLSID, int lCtrTWDelay);</code>
Parameters	CtrTWDelay: Controller delay 0 – 250 ms Time for which the axis has to remain in the target window. Moves are delayed by at least this time.
Example	<code>pTango->SetControllerTWDelay(1, 0);</code> <i>// controller delay switched off, closed loop end position will be inaccurate</i>

GetTargetWindow

Description	Retrieves closed loop target windows of all axes (?twi).
C++	<code>int LSX_GetTargetWindow (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	X, Y, Z, A: Target window, depends on selected dimension
Example	<code>pTango->GetTargetWindow(1, &X, &Y, &Z, &A);</code>

SetTargetWindow

Description	Set closed loop controller target windows (!twi). The closed loop controller has to settle within $\pm$ this window size for the specified delay time.
C++	<code>int LSX_SetTargetWindow (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	X, Y, Z, A: 1 – 25000 (motor increments) 0.1 – 1000 (μ m) 0.0001 – 1 (mm) (values depend on dimension)
Example	<code>pTango->SetTargetWindow(1, 1.0, 0.001, 0.001, 0.0005);</code>

SetCtrFastMoveOff

Description	Not supported by TANGO. FastMove function deactivated (!ctrfm 0).
C++	<code>int LSX_SetCtrFastMoveOff (int ILSID);</code>
Parameters	-
Example	<code>pTango->SetCtrFastMoveOff(1);</code>

SetCtrFastMoveOn

Description	Not supported by TANGO. Activate FastMove function, meaning a new vector is started if controller position difference is larger than the lock-in range (!ctrfm 1).
C++	<code>int LSX_SetCtrFastMoveOn (int ILSID);</code>
Parameters	-
Example	<code>pTango->SetCtrFastMoveOn(1);</code>

GetCtrFastMove

Description	Not supported by TANGO. Retrieves setting of FastMove function (?ctrfm).
C++	int LSX_GetCtrFastMove (int ILSID, BOOL *pbActive);
Parameters	Active: TRUE → FastMove function active
Example	pTango->GetCtrFastMove(1, &Active);

GetCtrFastMoveCounter

Description	Not supported by TANGO. If position difference is larger than lock-in range, a new vector will be started and corresponding counter will be increased by one. Function provides Fast Move counts (?ctrfmc).
C++	int LSX_GetCtrFastMoveCounter (int ILSID, int *plXC, int *plYC, int *plZC, int *plAC);
Parameters	XC, YC, ZC, AC: Number of carried out Fast Move functions
Example	pTango->GetCtrFastMoveCounter(1, &XC, &YC,&ZC,&AC);

ClearCtrFastMoveCounter

Description	Not supported by TANGO. If position difference is larger than lock-in range, a new vector will be started and corresponding counter will be increased by one (!ctrfmc).
C++	<code>int LSX_ClearCtrFastMoveCounter (int lLSID);</code>
Parameters	-
Example	<code>pTango->ClearCtrFastMoveCounter(1);</code>

4.14. Trigger Output

GetTrigger	
Description	Retrieve trigger globally enable setting (?trig).
C++	<code>int LSX_GetTrigger (int ILSID, BOOL *pbATrigger);</code>
Parameters	Atrigger: TRUE → trigger is on (enabled) FALSE → trigger is off (disabled)
Example	<code>pTango->GetTrigger(1, &Atrigger);</code>

SetTrigger	
Description	Switch trigger ON / OFF (!trig).
C++	<code>int LSX_SetTrigger (int ILSID, BOOL bATrigger);</code>
Parameters	Atrigger = TRUE → switch trigger on = FALSE → switch trigger off
Example	<code>pTango->SetTrigger(1, TRUE); // Globally enable the trigger (also defines start pos.)</code>

GetTriggerMode	
Description	Retrieve trigger mode (?trigm).
C++	<code>int LSX_GetTriggerMode (int ILSID, int *plMode);</code>
Parameters	lMode: Pointer to variable where the mode should be returned to
Example	<code>pTango->GetTriggerMode(1, &Mode);</code>

SetTriggerMode	
Description	Select trigger mode (!trigm).
C++	<code>int LSX_SetTriggerMode (int ILSID, int lMode);</code>
Parameters	lMode = Desired Trigger Mode
Example	<code>pTango->SetTriggerMode(1, 5); // Set Trigger Mode to 5</code>

GetTriggerPar	
Description	Retrieves trigger parameters (?triga / ?trigm / ?trigs / ?trigd).
C++	<code>int LSX_GetTriggerPar (int ILSID, int *plAxis, int *plMode, int *plSignal, double *pdDistance);</code>
Parameters	Axis: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) Mode: Trigger mode (see command !trigm) Signal: aTrigger signal (see command !trigs) Distance: Trigger distance (see command !trigd)
Example	<code>pTango->GetTriggerPar(1, &Axis, &Mode, &Signal, &Distance);</code>

SetTriggerPar	
Description	Set trigger parameters (!triga / !trigm / !trigs / !trigd). Sets several parameters at once.
C++	<code>int LSX_SetTriggerPar (int ILSID, int lAxis, int lMode, int lSignal, double dDistance);</code>
Parameters	Axis: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) Mode: Trigger mode (see command !trigm) Signal: Trigger signal (see command !trigs) Distance: Trigger distance (see command !trigd)
Example	<code>pTango->SetTriggerPar(1, 1, 3, 2, 5.0);</code>

GetTrigCount	
Description	Retrieve trigger counter value (?trigcount).
C++	<code>int LSX_GetTrigCount (int ILSID, int *plValue);</code>
Parameters	Value: Number of executed triggers
Example	<code>pTango->GetTrigCount(1, &Value);</code>

SetTrigCount	
Description	Set trigger counter value (!trigcount).
C++	<code>int LSX_SetTrigCount (int ILSID, int lValue);</code>
Parameters	Value: 0 to 2147483647
Example	<code>pTango->SetTrigCount(1, 0); // Clear trigger counter</code>

GetTriggerAxis	
Description	Read the selected axis for position trigger (?triga).
C++	<code>int LSX_GetTriggerAxis (int ILSID, int *plAxis);</code>
Parameters	Axis: Axis for position-dependent trigger modes (axis number 1, 2, 3, 4 = X...A)
Example	<code>pTango->GetTriggerAxis (1, &Axis);</code>

SetTriggerAxis	
Description	Set the axis for position-dependent triggering (!triga).
C++	<code>int LSX_SetTriggerAxis (int ILSID, int lAxis);</code>
Parameters	Axis: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes)
Example	<code>pTango->SetTriggerAxis (1, 2); // Trigger uses position of Y-axis (2)</code>

GetTriggerSignalLength	
Description	Read the trigger output signal length / pulse width (?trigs).
C++	<code>int LSX_GetTriggerSignalLength (int ILSID, int *plLength);</code>
Parameters	Length: Trigger pulse width 0 to 2500000 [μ s]
Example	<code>pTango->GetTriggerSignalLength (1, &Length);</code>

SetTriggerSignalLength	
Description	Set the trigger output signal length / pulse width (!trigs).
C++	<code>int LSX_SetTriggerSignalLength (int ILSID, int lLength);</code>
Parameters	Length: Trigger pulse width 0 to 2500000 [μ s] in increments of 40 μ s (0,40,80,...)
Example	<code>pTango->SetTriggerSignalLength (1, 120); // Trigger signal is 120μs long</code>

GetTriggerDistance	
Description	Read the position interval for trigger pulses (?trigd).
C++	<code>int LSX_GetTriggerDistance (int ILSID, double *pdDistance);</code>
Parameters	Distance: >0.0 to 5000000 [position unit depends on Dimension]
Example	<code>pTango->GetTriggerDistance (1, &Distance);</code>

SetTriggerDistance	
Description	Set the position interval for trigger pulses (!trigd).
C++	<code>int LSX_SetTriggerDistance (int ILSID, double dDistance);</code>
Parameters	Distance: >0.0 to 5000000 [position unit depends on Dimension]
Example	<code>pTango->SetTriggerDistance (1, 12.5); // Set the position interval to 12.5 (mm, μm, ...)</code>

GetTriggerCompensation	
Description	Read the trigger look-ahead timing compensation (?trigcomp). Compensates for delays within the trigger chain by looking ahead. Useful for bidirectional scanning in order to compensate comb effect of the lines.
C++	<code>int LSX_GetTriggerCompensation (int ILSID, int *plTime);</code>
Parameters	Time: -10000 ... +10000 [μ s] in increments of 10 μ s (0,10,20,...)
Example	<code>pTango->GetTriggerCompensation (1, &Time);</code>

SetTriggerCompensation	
Description	Set the trigger look-ahead timing compensation (!trigcomp). Compensates for delays within the trigger chain by looking ahead. Useful for bidirectional scanning in order to compensate comb effect of the lines.
C++	<code>int LSX_SetTriggerCompensation (int ILSID, int lTime);</code>
Parameters	Time: -10000 ... +10000 [μ s] in increments of 10 μ s (0,10,20,...)
Example	<code>pTango->SetTriggerCompensation (1, 40); // Trigger compensation looks 40μs ahead</code>

GetTriggerEncoder	
Description	Read if the trigger unit uses the encoder position (?trigenc).
C++	<code>int LSX_GetTriggerEncoder (int ILSID, int *plEncoder);</code>
Parameters	Encoder: 0 = trigger on motor position, 1 = trigger on encoder position
Example	<code>pTango->GetTriggerEncoder (1, &Encoder);</code>

SetTriggerEncoder	
Description	Select trigger on encoder (1) or motor (0) position (!trigenc).
C++	<code>int LSX_SetTriggerEncoder (int ILSID, int lEncoder);</code>
Parameters	Encoder: 0 = trigger on motor position, 1 = trigger on encoder position
Example	<code>pTango->SetTriggerEncoder (1, 0); // Trigger on motor position</code>

GetTriggerFrequency	
Description	Read the trigger output frequency for trigger modes 100, 101 (?trigf).
C++	<code>int LSX_GetTriggerFrequency (int ILSID, double *pdFrequency);</code>
Parameters	Frequency: 0.01 to 25000 Hz
Example	<code>pTango->GetTriggerFrequency (1, &Frequency);</code>

SetTriggerFrequency	
Description	Set the trigger output frequency for trigger modes 100, 101 (!trigf).
C++	<code>int LSX_SetTriggerFrequency (int ILSID, double dFrequency);</code>
Parameters	Frequency: 0.01 to 25000 Hz
Example	<code>pTango->SetTriggerFrequency (1, 1000); // Set the trigger frequency to 1kHz</code>

GetTriggerOutput	
Description	Read the trigger output setting (?trigo).
C++	<code>int LSX_GetTriggerOutput (int ILSID, int *plValue);</code>
Parameters	Value: 0 = no output used, 1=TRIGGER_OUT, and further combinations (refer to trigo)
Example	<code>pTango->GetTriggerOutput (1, &Value);</code>

SetTriggerOutput					
Description	Select trigger output setting (!trigo). Select output 1 and/or 2 and output mode.				
C++	int LSX_SetTriggerOutput (int ILSID, int lValue);				
Parameters	Value: 0 = no output used, 1=TRIGGER_OUT, and further combinations (refer to trigo)				
	Output / Mode	STANDARD	PREC.WIDTH2	PREC.DELAY2	PREC.FREQUENCY2
	No output No signal out	0	(4)	(8 PCI-E)	(12)
	Primary TRIGGER OUT	1	(5)	(9 PCI-E)	(13)
	Secondary ** TAKT OUT	2	6	10 (PCI-E)	14
	Both, P&S ** TRIGGER+TAKT	3	7	11 (PCI-E)	15
Example	pTango->SetTriggerOutput (1, 1); // Use default trigger output 1 (TRIGGER_OUT) only				

Get2ndTriggerDelay	
Description	Read the precise edge delay of the 2 nd output related to the trigger output (?trigbdelay).
C++	<code>int LSX_Get2ndTriggerDelay (int ILSID, double *pdValue);</code>
Parameters	Value: 0.00 to 32500000 [μs], internal resolution is 1/132μs
Example	<code>pTango->Get2ndTriggerDelay (1, &Value);</code>

Set2ndTriggerDelay	
Description	Set the precise edge delay of the 2 nd output related to the trigger output (!trigbdelay).
C++	<code>int LSX_Set2ndTriggerDelay (int ILSID, double dValue);</code>
Parameters	Value: 0.00 to 32500000 [μs], internal resolution is 1/132μs
Example	<code>pTango->Set2ndTriggerDelay (1, 0.05); // Set the 2nd output delay to 50ns</code>

Get2ndTriggerWidth	
Description	Read the precise pulse width of the 2 nd trigger output signal (?trigbwidth).
C++	<code>int LSX_Get2ndTriggerWidth (int ILSID, double *pdValue);</code>
Parameters	Value: 0.00 to 32500000 [μs], internal resolution is 1/132μs
Example	<code>pTango->Get2ndTriggerWidth (1, &Value);</code>

Set2ndTriggerWidth	
Description	Set the precise pulse width of the 2 nd trigger output signal (!trigbwidth).
C++	<code>int LSX_Set2ndTriggerWidth (int ILSID, double dValue);</code>
Parameters	Value: 0.00 to 32500000 [μs], internal resolution is 1/132μs
Example	<code>pTango->Set2ndTriggerWidth (1, 1.05); // Set the 2nd output pulse width to 1.05μs</code>

Get2ndTriggerFrequency	
Description	Read the precise frequency of the 2 nd trigger output signal (?trigbf).
C++	<code>int LSX_Get2ndTriggerFrequency (int ILSID, double *pdValue);</code>
Parameters	Value: 0.010 ... 66000000 Hz
Example	<code>pTango->Get2ndTriggerFrequency (1, &Value);</code>

Set2ndTriggerFrequency

Description	Set the precise frequency of the 2 nd trigger output signal (!trigbf).
C++	<code>int LSX_Set2ndTriggerFrequency (int ILSID, double dValue);</code>
Parameters	Value: 0.010 ... 66000000 Hz
Example	<code>pTango->Set2ndTriggerFrequency (1, 1000000); // Set the 2nd output frequency to 1MHz</code>

GetTriggerRange

Description	Read back the trigger range settings which were set by LSX_SetTriggerRange (?trigr).
C++	<code>int LSX_GetTriggerRange (int ILSID, double *pdStartPos, double *pdEndPos, int *plNumberOfTriggerPulses);</code>
Parameters	Axis: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) StartPos: Position where the triggering starts (first pulse) EndPos: Position where the trigger ends (last pulse) NumberOfTriggerPulses: To achieve the required trigger distance (interval). It must be considered that the first pulse is before the first interval, so N+1 pulses must be set
Example	<code>pTango->GetTriggerRange (1, &StartPos, &EndPos, &NumberOfTriggerPulses);</code>

SetTriggerRange

Description	Set the trigger range, which defines a trigger start position, end position and the number of triggers from start to end (!trigr). The unit (µm,mm,...) depends on the Dimension. Info: SetTriggerRange creates an internal equidistant position list, so the TriggerPositionList functions can also be used with TriggerRange, if required. To define the triggered axis, use SetTriggerAxis once before using TriggerRange(s).
C++	<code>int LSX_SetTriggerRange (int ILSID, double dStartPos, double dEndPos, int lNumberOfTriggerPulses);</code>
Parameters	Axis: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) StartPos: Position where the triggering starts (first pulse) EndPos: Position where the trigger ends (last pulse) NumberOfTriggerPulses: To achieve the required trigger distance (interval). It must be considered that the first pulse is before the first interval, so N+1 pulses must be set
Example	<code>pTango->SetTriggerRange (1, 10.5, 20.5 101); // Set 101 pulses from 10.5 to 20.5mm // which is the 1st pulse at 10.5mm plus 100 pulses until and including 20.5mm that lead to a (20.5-10.5)/100 = 0.1mm distance between the pulses</code>

GetTriggerPositionList	
Description	Read back the trigger position list from SetTriggerPositionList (?trigp). This can also read back the internal list set by SetTriggerRange.
C++	<code>int LSX_GetTriggerPositionList (int ILSID, int lIndex, double *pdPos);</code>
Parameters	Index: 1... GetTriggerPositionListEntries Pos: Position list entry of the selected trigger axis (depends on Dimension)
Example	<code>pTango->GetTriggerPositionList (1, &Index, &Pos);</code>

SetTriggerPositionList	
Description	Create a trigger position list with individual positions, e.g. not equidistant (!trigp). The positions can be individual but must be either constantly increasing or decreasing.
C++	<code>int LSX_SetTriggerPositionList (int ILSID, int lIndex, double dPosition);</code>
Parameters	Index: 1...MAX_ENTRIES Pos: Position list entry for the selected trigger axis (depends on Dimension)
Example	<code>pTango->SetTriggerPositionList (1, 1, 10.5); // Set first pulse at 10.5mm pTango->SetTriggerPositionList (1, 2, 17.5031); // Set 2nd pulse at 17.5031mm pTango->SetTriggerPositionList (1, 3, 51.8774); // Set 3rd pulse at 51.8774mm</code>

GetTriggerPositionListIndex	
Description	Read at which entry (index) the position list currently is (?trigi). This also reads back the current index in case of SetTriggerRange.
C++	<code>int LSX_GetTriggerPositionListIndex (int ILSID, int *plIndex);</code>
Parameters	Index: 1...N
Example	<code>pTango->GetTriggerPositionListIndex (1, &Index);</code>

SetTriggerPositionListIndex	
Description	Manipulate the position list index for the next trigger pulse (!trigi). Usually, the trigger goes forward through the position list, but by manipulating the current list index the trigger can skip some positions by manipulating the index forward.
C++	<code>int LSX_SetTriggerPositionListIndex (int ILSID, int lIndex);</code>
Parameters	Index: 1...N
Example	<code>pTango->SetTriggerPositionListIndex (1, 17); // Set the next trigger position at list entry number 17 (to skip the trigger positions in between)</code>

GetTriggerPositionListEntries

Description	Read the number of entries in the trigger position list (?trigc). This also is the index of the last entry in the trigger position list: 1...Entries And can be used e.g. to append positions at Index “entries+1” etc.
C++	<code>int LSX_GetTriggerPositionListEntries (int ILSID, int *plNumberOfEntries);</code>
Parameters	<i>NumberOfEntries</i> : 0...N
Example	<code>pTango->GetTriggerPositionListEntries (1, &NumberOfEntries);</code>

SetTriggerPositionListEntries

Description	Manipulate the amount of position list entries (!trigc). The number 0 can be used to clear the entire position list (to allow entering a new list).
C++	<code>int LSX_SetTriggerPositionListEntries (int ILSID, int lNumberOfEntries);</code>
Parameters	<i>NumberOfEntries</i> : 0, 1...N
Example	<code>pTango->SetTriggerPositionListEntries (1, 5); // Reduce the number of entries to 5</code> <code>pTango->SetTriggerPositionListEntries (1, 0); // Delete the entire trigger position list</code>

GetTriggerLevel

Description	Read the trigger level used exclusively by trigger modes 20 and 21 (?trigl). All other modes specify their level (pulse polarity) by the mode itself, e.g. 100, 101.
C++	<code>int LSX_GetTriggerLevel (int ILSID, int *plLevel);</code>
Parameters	<i>Level</i> : 0 = active low, 1 = active high trigger pulse
Example	<code>pTango->GetTriggerLevel (1, &Level);</code>

SetTriggerLevel

Description	Set the trigger level used exclusively by TriggerRange and PositionList (?trigl). All other modes specify their level (pulse polarity) by the mode itself, e.g. 100, 101.
C++	<code>int LSX_SetTriggerLevel (int ILSID, int lLevel);</code>
Parameters	<i>Level</i> : 0 = active low, 1 = active high trigger pulse
Example	<code>pTango->SetTriggerLevel (1, 0); // Set trigger to active low (0)</code>

4.15. Snapshot Input

GetSnapshot	
Description	Provides current Snapshot state, if it is ON/enabled or OFF/disabled (?sns).
C++	<code>int LSX_GetSnapshot (int ILSID, BOOL *pbASnapshot);</code>
Parameters	<i>Asnapshot:</i> TRUE → Snapshot is “On” (enabled) FALSE → Snapshot is “Off” (disabled)
Example	<code>pTango->GetSnapshot(1, &Asnapshot);</code>

SetSnapshot	
Description	Switch Snapshot functionality ON or OFF (!sns).
C++	<code>int LSX_SetSnapshot (int ILSID, BOOL bASnapshot);</code>
Parameters	<i>Asnapshot:</i> TRUE → switch Snapshot “On” (enable) FALSE → switch Snapshot “Off” (disable)
Example	<code>pTango->SetSnapshot(1, TRUE); // Globally enable the snapshot functionality</code>

GetSnapshotMode	
Description	Provides the current Snapshot mode (?snsm).
C++	<code>int LSX_GetSnapshotMode (int ILSID, int*plMode);</code>
Parameters	<i>Mode:</i> 0-12 (refer to snsmd documentation in TANGO Instruction Set)
Example	<code>pTango->GetSnapshotMode(1, &Mode);</code>

SetSnapshotMode	
Description	Sets the Snapshot mode/functionality (!snsm).
C++	<code>int LSX_SetSnapshotMode (int ILSID, int lMode);</code>
Parameters	<i>Mode:</i> 0-12 (refer to snsmd documentation in TANGO Instruction Set)
Example	<code>pTango->SetSnapshotMode(1, 0); // Set mode to 0 = capture positions @ HDI F2 key</code>

GetSnapshotCount

Description	Snapshot counter (?snc). It counts the snapshot events = number of captured positions / entries in the position array (see SnapshotPosArray).
C++	<code>int LSX_GetSnapshotCount (int ILSID, int *plSnsCount);</code>
Parameters	SnsCount: Amount of captured Snapshots (= available position array entries)
Example	<code>pTango->GetSnapshotCount(1, &SnsCount);</code>

SetSnapshotCount

Description	Manipulate Snapshot counter (captured positions), truncate position array entries (!snc).
C++	<code>int LSX_SetSnapshotCount (int ILSID, int lSnsCount);</code>
Parameters	SnsCount: Amount of available position array entries
Example	<code>pTango->SetSnapshotCount(1, 5); // Truncate position array to 5 entries.</code>

GetSnapshotFilter

Description	Retrieve input filter times for signal chatter (?snsf).
C++	<code>int LSX_GetSnapshotFilter (int ILSID, int *plTime);</code>
Parameters	Time: Filter time [ms]
Example	<code>pTango->GetSnapshotFilter(1, &Time);</code>

SetSnapshotFilter

Description	Set input debounce filter (!snsf). If a mechanical switch is connected, a typical debounce time is around 10ms. As the debounce filter slows down the processing speed (interval), for a true digital signal (which does not bounce), the filter might be set to 0 ms.
C++	<code>int LSX_SetSnapshotFilter (int ILSID, int lTime);</code>
Parameters	Time: Filter time, within 0-100 ms
Example	<code>pTango->SetSnapshotFilter(1, 0); // no filter, fast response (e.g. for TTL signals)</code>

GetSnapshotPar

Description	Retrieve Snapshot parameters (?snsl + ?snsm 0/1). Does not support reading of higher snapshot modes! Only 0 or “not 0”. → Use GetSnapShotMode instead.
C++	<code>int LSX_GetSnapshotPar (int ILSID, BOOL *pbHigh, BOOL *pbAutoMode);</code>
Parameters	High: TRUE → snapshot is high active FALSE → snapshot is low active AutoMode: TRUE → snapshot „Automatic“: Position is automatically moved to after first snapshot pulse (corresponds to SnapshotMode 1) FALSE → snapshot capture mode (corresponds to SnapshotMode 0)
Example	<code>pTango->GetSnapshotPar(1, &High, &AutoMode);</code>

SetSnapshotPar

Description	Set Snapshot parameters (polarity and only snapshot mode 0 or 1: !snsl + !snsm 0/1). The AutoMode might interfere with a previously set SnapshotMode, if that was set to a mode higher than 1) Does not support setting of higher snapshot modes! Only 0 or 1. → Use SetSnapShotMode instead.
C++	<code>int LSX_SetSnapshotPar (int ILSID, BOOL bHigh, BOOL bAutoMode);</code>
Parameters	High: TRUE → snapshot is high active FALSE → snapshot is low active AutoMode: TRUE → snapshot „Automatic“: Position is automatically moved to after first snapshot pulse (corresponds to SnapshotMode 1) FALSE → snapshot capture mode (corresponds to SnapshotMode 0)
Example	<code>pTango->SetSnapshotPar(1, TRUE, FALSE);</code>

GetSnapshotPos

Description	Retrieve position that was captured on the most recent Snapshot event (?snsp).
C++	<code>int LSX_GetSnapshotPos (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</code>
Parameters	X, Y, Z, A: Position values
Example	<code>pTango->GetSnapshotPos(1, &X, &Y, &Z, &A);</code>

GetSnapshotPosArray	
Description	Retrieve Snapshot position from Array (?snsa).
C++	<pre>int LSX_GetSnapshotPosArray (int ILSID, int IIndex, double *pdX, double *pdY, double *pdZ, double *pdA);</pre>
Parameters	Index: Index of snapshot positions (from =1 to SnapshotCount, max. entries is 1024) X, Y, Z, A: Position values
Example	<pre>pTango->GetSnapshotPosArray(1, 2, &X, &Y, &Z, &A); // 2 = Read positions captured on the second snapshot event (second array entry)</pre>

SetSnapshotPosArray	
Description	Set, append or change entries of the position array (!snsa).
C++	<pre>int LSX_SetSnapshotPosArray (int ILSID, int IIndex, double dX, double dY, double dZ, double dA);</pre>
Parameters	Index: Index of snapshot positions (1-1024) Index must be within the number of existing entries (or one above to append) appending is also possible by using Index = -1, which is easier to handle X, Y, Z, A: Position values
Example	<pre>pTango->SetSnapshotPosArray(1, -1, 0.55, 2.4, 0.0, 0.0); // Append a position array entry by software</pre>

ClearSnapshotPosArray	
Description	Deletes the entire position array, clears all entries (!snc 0) and checks if the array was cleared by ?snc == 0.
C++	<pre>int LSX_ClearSnapshotPosArray (int ILSID,);</pre>
Parameters	-
Example	<pre>pTango->ClearSnapshotPosArray(1); // Delete the entire PosArray</pre>

GetSnapshotIndex

Description	Read the current Snapshot index (?snsi), e.g. to identify where it is in “Automatic” mode. Remarks: The index goes from 0 to SnapshotCount-1, so index “0” is PosArray(1).
C++	<code>int LSX_GetSnapshotIndex (int lLSID, int *plSnsIndex);</code>
Parameters	SnsIndex: Current position of the index pointer within the position array
Example	<code>pTango->GetSnapshotIndex(1, &SnsIndex);</code>

SetSnapshotIndex

Description	Manipulate Snapshot index: set index to a different position array entry (!snsi) Remarks: The index goes from 0 to SnapshotCount-1, so index “0” is PosArray(1).
C++	<code>int LSX_SetSnapshotIndex (int lLSID, int lSnsIndex);</code>
Parameters	SnsIndex: Required position of the index pointer within the PosArray, e.g. for SnapshotMode “Automatic”
Example	<code>pTango->SetSnapshotIndex(1, 5); // Set pointer to Index 5</code>

5. SlideExpress Functions

This chapter describes additional DLL functions for the SlideExpress. From application point of view there are only few differences between previous top loader and new front loader systems SlideExpress (1) and SlideExpress 2.

Constant Name	Meaning	Top Loader	Front Loader
MAXMAGA	number of magazines	4	3
MAXROW	number of rows	50	30
MAXCOL	Number of columns	4	4

SlideExpress 2: Organization of slides in rows and columns

Despite the organization of **clips and magazines**, the SlideExpress instructions still handle slides as **rows and columns**.

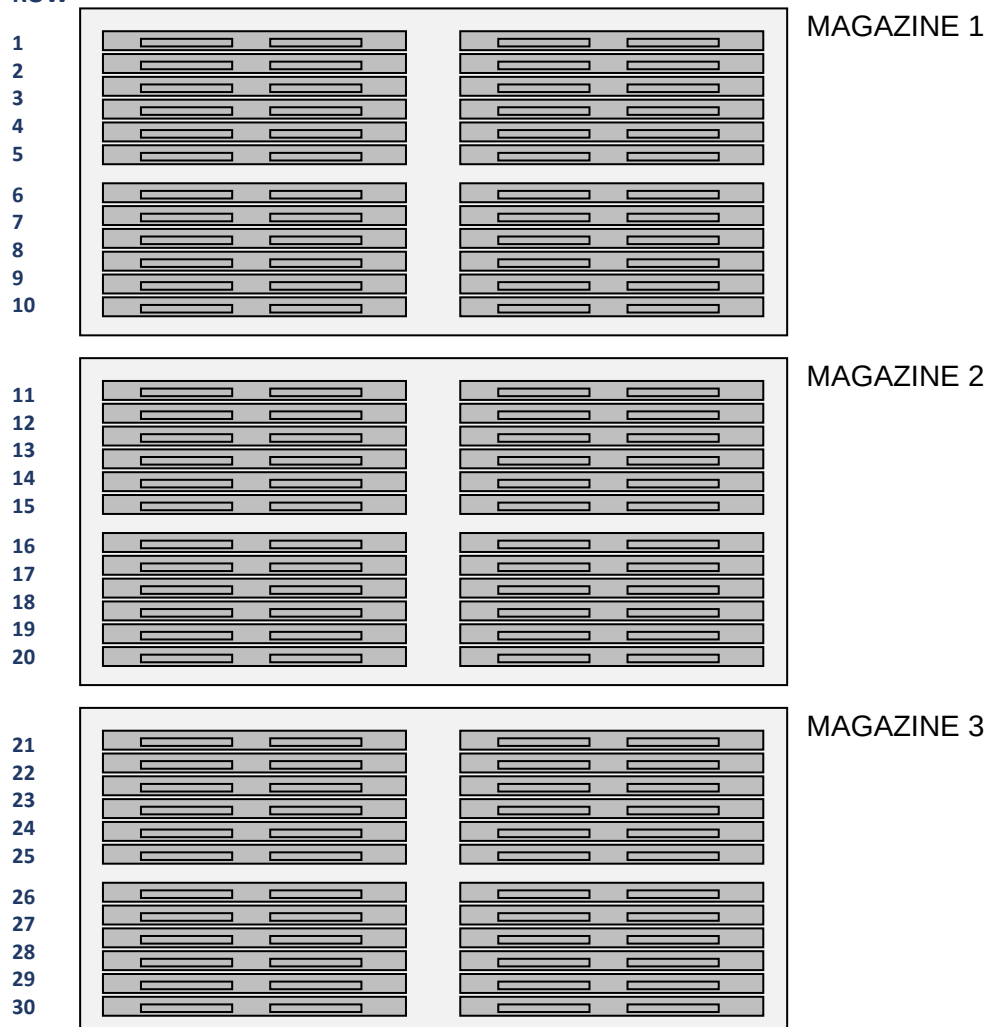
For the new SlideExpress 2, there are 3 magazines 1,2,3 stacked over each other.

Each magazine has 10 rows with two clip columns, where a clip usually contains 2 slides.

This leads to the following organization:

COLUMN: 1 2 3 4

ROW



Eject	
Description	Move magazine(s) to a position that allows user access (!eject).
C++	<code>int LSX_Eject (int ILSID, int maga, int keep);</code>
Parameters	maga → magazine number [1..MAXMAGA] keep → 0 to empty gripper before eject magazine(s) or 1 to keep slide(s) in gripper
Example	<code>pTango->Eject(1, 1, 0);</code>

Insert	
Description	Magazine(s) are inserted and tested if seated and which slides are present (!insert). This function is precondition to use SlideSeated() and MagazinSeated().
C++	<code>int LSX_Insert (int ILSID);</code>
Parameters	-
Example	<code>pTango->Insert(1);</code>

SlideSeated	
Description	Query if slide is present or not or unknown (?insert).
C++	<code>int LSX_SlideSeated (int ILSID, int col, int row, int *status);</code>
Parameters	col → col number [1..MAXCOL] row → row number [1..MAXROW] status → returns slide status (-1 = unknown, 0 = empty, 1 = seated)
Example	<code>pTango->SlideSeated (1, 4, 30, &status);</code>

MagazinSeated	
Description	Query if magazine is present or not or unknown (?insert).
C++	<code>int LSX_MagazinSeated (int ILSID, int maga, int *status);</code>
Parameters	maga → magazine number [1..MAXMAGA] status → returns magazine status (-1 = unknown, 0 = empty, 1 = seated)
Example	<code>pTango->MagazinSeated (1, 1, &status); // check if magazine 1 is seated</code>

GetGripper

Description	Query gripper status information. Returns status of gripper 1 and 2 (?gripper). Status information like unknown, empty or origin of slide in gripper.
C++	<code>int LSX_GetGripper (int ILSID, int *c1, int *r1, int *c2, int *r2);</code>
Parameters	c1 → column number [-1, 0, 1..MAXCOL] of slide 1 in gripper r1 → row number [-1, 0, 1..MAXROW] of slide 1 in gripper c2 → column number [-1, 0, 1..MAXCOL] of slide 2 in gripper r2 → row number [-1, 0, 1..MAXROW] of slide 2 in gripper
Example	pTango->GetGripper (1, &c1, &r1, &c2, &r2); // check status of gripper 1 and 2 c1, c2 → -1 = unknown, 0 = empty or 1 to 4 for magazine number r1, r2 → -1 = unknown, 0 = empty or 1 to 50 for slot number c1=1,r1=0 indicates priority slide 1 in gripper (obsolete for front loader) c2=1,r2=0 indicates priority slide 2 in gripper (obsolete for front loader)

SetGripper

Description	Set gripper forces an overwrite of the gripper status (!gripper). Usually the status represents empty, unknown or slide/tray origin state. The status is managed internally and shall only be manipulated to solve unknown gripper state after power loss or to manipulate the origin where it comes from and so where it will be returned to, e.g. for custom sorting.
C++	<code>int LSX_SetGripper (int ILSID, int c1, int r1, int c2, int r2);</code>
Parameters	c1 → column number [-1, 0, 1..MAXCOL] of slide 1 in gripper r1 → row number [-1, 0, 1..MAXROW] of slide 1 in gripper c2 → column number [-1, 0, 1..MAXCOL] of slide 2 in gripper r2 → row number [-1, 0, 1..MAXROW] of slide 2 in gripper
Example	pTango->SetGripper (1, 0, 0, 0, 0); // set gripper to “empty”

GetClipType

Description	GetClipType returns the clip type that is currently in the gripper (?cliptype). For SlideExpress handling system.
C++	<code>int LSX_GetClipType (int ILSID, int *pClipType);</code>
Parameters	pClipType → pointer to int, returns the clip type
Example	pTango->GetClipType (1, pClipType);

GetSlide	
Description	Get slide(s) from addressed position in magazine (!getslide).
C++	<code>int LSX_GetSlide (int ILSID, int col, int row, int mode);</code>
Parameters	col → column number [1..MAXCOL] row → row number [1..MAXROW] mode → (0 = inspection, 1 = bar code reader, 2 = liquid dispenser “oiler”)
Example	<code>pTango->GetSlide (1, 1, 1, 0);</code>

PutSlide	
Description	Put slide(s) back to addressed position in magazine or priority handler (!putslide). To return the slide to its original position (where it was taken from by GetSlide), set both col and row parameters to 0.
C++	<code>int LSX_PutSlide (int ILSID, int col, int row);</code>
Parameters	col → column [1..MAXCOL] row → slot number [1..MAXROW] (obsolete: or [0] for priority handler) If both parameters are 0 the DLL transmits !putslide without arguments. In this case Tango uses known gripper information to put slides back (if any).
Example	<code>pTango->PutSlide (1, 4, 20);</code> // put slide to magazine 4 slot 20. <code>pTango->PutSlide (1, 0, 0);</code> // put slide back to where it was taken from.

Obsolete:

GetPrioHandlerPos	
Description	Read actual priority handler position (?priohand).
C++	<code>int LSX_GetPrioHandlerPos (int ILSID, int *php);</code>
Parameters	php → return value of actual priority handler position (55 = unknown, 0 = middle, -1 = shift in, 1 = pulled out)
Example	<code>pTango->GetPrioHandlerPos (1, &php);</code>

Obsolete:

SetPrioHandlerPos	
Description	Enables user to shift priority handler to required position (!priohand). Handler is locked at destination or after 30s timeout.
C++	<code>int LSX_SetPrioHandlerPos (int ILSID, int php);</code>
Parameters	php → specify destination 0 = middle, -1 = shift in, 1 = pulled out
Example	<code>pTango->SetPrioHandlerPos (1, 1);</code> //enable user to pull out priority handler

6. TrayExpress Functions

This chapter describes optional DLL functions to be used in conjunction for TrayExpress.

Eject	
Description	Eject magazine (!eject). The TrayExpress moves magazine downwards and opens front cover to allow user operations like removing trays or loading trays.
C++	<code>int LSX_Eject (int ILSID, int maga, int keep);</code>
Parameters	maga → magazine number [1] (currently only 1 allowed) keep → 0 to empty gripper before eject magazine or 1 to keep tray in gripper
Example	<code>pTango->Eject(1, 1, 0);</code>

Insert	
Description	Front Cover is closed and magazine is inserted and tested if seated and which trays are present (!insert). This function is precondition to use SlideSeated() and MagazinSeated().
C++	<code>int LSX_Insert (int ILSID);</code>
Parameters	-
Example	<code>pTango->Insert(1);</code>

SlideSeated	
Description	Query if tray is present or not or unknown (?insert).
C++	<code>int LSX_SlideSeated (int ILSID, int maga, int slot, int *status);</code>
Parameters	maga → magazine number [1] slot → slot number [1..50] status → returns slide status (-1 = unknown, 0 = empty, 1 = seated)
Example	<code>pTango->SlideSeated (1, 1, 1, &status);</code>

MagazinSeated	
Description	Query if magazine is present or not or unknown (?seated).
C++	<code>int LSX_MagazinSeated (int ILSID, int maga, int *status);</code>
Parameters	maga → magazine number [1] status → returns magazine status (-1 = unknown, 0 = empty, 1 = seated)
Example	<code>pTango->MagazinSeated (1, 1, &status); //check if magazine 1 is seated</code>

GetGripper

Description	Query gripper status information (?gripper). Returns status of gripper.
C++	<code>int LSX_GetGripper (int ILSID, int *c1, int *s1, int *c2, int *s2);</code>
Parameters	c1 → magazine number [-1, 0, 1..4] of slide in gripper s1 → slot number [-1, 0, 1..24] of slide in gripper c2 → dummy for compatibility with slide express s2 → dummy for compatibility with slide express
Example	pTango->GetGripper (1, &c1, &s1, &c2, &s2); //check status of gripper 1 and 2 c1 → -1 = unknown, 0 = empty or 1 (magazine number) s1 → -1 = unknown, 0 = empty or 1 to 24 for slot number

SetGripper

Description	Set gripper status information (!gripper). The status is managed internally and shall only be manipulated in case of solving gripper states after power loss or for tray sorting tasks.
C++	<code>int LSX_SetGripper (int ILSID, int c1, int s1, int c2, int s2);</code>
Parameters	c1 → magazine number [-1, 0, 1..4] of slide in gripper s1 → slot number [-1, 0, 1..50] of slide in gripper c2 → dummy for compatibility with slide express s2 → dummy for compatibility with slide express
Example	pTango->SetGripper (1, 0, 0, 0, 0); //set gripper to “empty”

GetTray

Description	Get tray from the specified magazine position (!gettray).
C++	<code>int LSX_GetTray (int ILSID, int slot, int mode);</code>
Parameters	slot → slot number [1..35*] mode → place tray for [0 = microscope, 1 = liquid dispenser, 2 = bar code reader *] * The maximum slot number and availability of modes depends on hardware
Example	pTango->GetTray (1, 2, 0); // get tray from slot 2 and put it under the microscope

PutTray

Description	Put tray back to the specified magazine position (!puttray). or to the position where it came from → by setting slot = 0.
C++	<code>int LSX_PutTray (int ILSID, int slot);</code>
Parameters	slot → slot number [1..35*] or 0 to return it back to the original position of GetTray * The maximum slot number (24, 35, ...) depends on hardware
Example	pTango->PutTray (1, 10); // put tray to magazine slot 10. pTango->PutTray (1, 0); // put tray back to where it was taken from by GetTray.

GetRFID	
Description	Get RFID of addressed tray, when tray is properly seated in magazine (?rfid)
C++	<code>int LSX_GetRFID (int ILSID, int slot, int bank, int *plRFID);</code>
Parameters	slot → slot number [1..MAXSLOT] bank → bank number [0 to 64] plRFID → pointer to int returns data stored in RFID transponder device
Example	<code>pTango->GetRFID (1, 1, 0, plRFID);</code>

SetRFID	
Description	Store RFID data into addressed tray (!rfid)
C++	<code>int LSX_SetRFID (int ILSID, int slot, int bank, int rfdata);</code>
Parameters	slot → slot number [1..MAXSLOT] bank → bank number [2 to 64] (bank 0 and 1 are not writeable) rfdata → int contains customer data to be coded into RFID transponder device
Example	<code>pTango->SetRFID (1, 1, 0, rfdata);</code>

GetNumberOfSlots	
Description	Get number of available slots per magazine (?separ 15)
C++	<code>int LSX_GetNumberOfSlots (int ILSID, int *plSlots);</code>
Parameters	plSlots → returns number of slots per magazine
Example	<code>pTango->GetNumberOfSlots (1, plSlots);</code>

GetNumberOfMagazines	
Description	Get number of available magazines (?maxmaga). Returns always 1 and is available for compatibility to SlideExpress only.
C++	<code>int LSX_GetNumberOfMagazines (int ILSID, int *plMagazines);</code>
Parameters	plMagazines → pointer to int returns number [1]
Example	<code>pTango->GetNumberOfMagazines (1, plMagazines);</code>

7. Additional Handling System Functions

Additional commands for SlideExpress, TrayExpress and other handling systems.

GetLoaderType	
Description	Get loader type (?loadertype) Response depends on system configuration.
C++	<code>int LSX_GetLoaderType (int ILSID, int *plLoaderType);</code>
Parameters	plLoaderType → pointer to int returns loader type 1 → SlideExpress 2 → Custom handling system: standalone base unit 3 → Custom handling system: loader master base unit 4 → Custom handling system: loader slave (magazine)
Example	<code>pTango->GetLoaderType (1, plLoaderType);</code>

GetNumberOfRows	
Description	Get number of magazine rows, e.g. max. number of slots to insert trays (?separ 15). Response is number of magazine rows.
C++	<code>int LSX_GetNumberOfRows (int ILSID, int *plRows);</code>
Parameters	plRows → pointer to int returns number of available slots or magazine rows
Example	<code>pTango->GetNumberOfRows (1, plRows);</code>

GetNumberOfColumns	
Description	Get number of magazine columns, e.g. max number of slide sensors per slot/tray (?separ 16). Response is number of magazine columns.
C++	<code>int LSX_GetNumberOfColumns (int ILSID, int *plCols);</code>
Parameters	plCols → pointer to int returns number of magazine column (6 manual, 6 for loader system)
Example	<code>pTango->GetNumberOfColumns (1, plCols);</code>

GetTraySN	
Description	Get tray SN returns unique tray RFID serial number of addressed slot / tray (?traysn). For TrayExpress and custom handling system.
C++	<code>int LSX_GetTraySN (int ILSID, int slot, int *plTraySN);</code>
Parameters	plTraySN → pointer to int returns unique tray RFID serial number
Example	<code>pTango->GetTraySN (1, 1, plTraySN);</code>

GetTrayType	
Description	GetTrayType returns tray type of addressed tray (?traytype). For TrayExpress and custom handling system.
C++	<code>int LSX_GetTrayType (int ILSID, int slot, int *plTrayType);</code>
Parameters	plTrayType → pointer to int returns tray type (user coded data)
Example	<code>pTango->GetTrayType (1, 1, plTrayType);</code>

SetTrayType	
Description	SetTrayType stores tray type into RFID transponder of addressed slot / tray (!traytype). For TrayExpress and custom handling system, only used for factory programming .
C++	<code>int LSX_SetTrayType (int ILSID, int slot, int aTrayType);</code>
Parameters	aTrayType → int data contains information of required tray type
Example	<code>int aTrayType = 0x0100010a; //see customer specification requirements for explanation pTango->SetTrayType (1, 1, aTrayType);</code>

SetCabinLED	
Description	SetCabinLED on or off (!cabinled). For handling systems with internal illumination.
C++	<code>int LSX_SetCabinLED (int lOn);</code>
Parameters	lOn → 0 to switch OFF or 1 to switch ON the loader illumination
Example	<code>pTango->SetCabinLED (1, 1); // switch ON illumination</code> <code>pTango->SetCabinLED (1, 0); // switch OFF</code>

GetCabinLED	
Description	GetCabinLED returns actual state of cabin illumination, on or off (?cabinled). For handling systems with internal illumination.
C++	<code>int LSX_GetCabinLED (int lLSID, int *plState);</code>
Parameters	plState → Pointer to int returns illumination state 0 (off) or 1 (on)
Example	<code>pTango->GetCabinLED (1,plState);</code>

SetLabelLED	
Description	SetLabelLED on or off. (!labelled) For handling systems with barcode illumination.
C++	<code>int LSX_SetLabelLED (int lOn);</code>
Parameters	lOn → 0 to switch OFF or 1 to switch ON the label illumination
Example	<code>pTango->SetLabelLED (1, 1); // switch ON illumination</code> <code>pTango->SetLabelLED (1, 0); // switch OFF</code>

GetLabelLED	
Description	GetLabelLED returns actual state of label illumination (?labelled). For handling systems with barcode illumination.
C++	<code>int LSX_GetLabelLED (int lLSID, int *plState);</code>
Parameters	plState → pointer to int returns illumination state (0=off, 1=on)
Example	<code>pTango->GetLabelLED (1,plState);</code>

8. xPos Module (POS3 3 axis extension)

Additional commands for the 3 auxiliary axes of the optional xPos / POS3 module.

Xpos3GetPosSingleAxis	
Description	Read axis position from an xPos axis (?xp)
C++	<code>int LSX_Xpos3GetPosSingleAxis (int ILSID, int IAxis, double *pIPos);</code>
Parameters	IAxis → xPos axis 1, 2 or 3 IPos → variable to return the position to
Example	<code>pTango->Xpos3GetPosSingleAxis(1, 2, &Pos); // Read Position of the 2nd xPos axis</code>

Xpos3SetPosSingleAxis	
Description	Set/Change axis position of an xPos axis (!xp)
C++	<code>int LSX_Xpos3SetPosSingleAxis (int ILSID, int IAxis, double IPos);</code>
Parameters	IAxis → xPos axis 1, 2 or 3 IPos → desired position value
Example	<code>pTango->Xpos3SetPosSingleAxis(1, 3, 12.345); // Set xPos 3rd axis position to 12.345</code>

Xpos3MoveAbsSingleAxis	
Description	Absolute Move for an xPos axis (!xma)
C++	<code>int LSX_Xpos3MoveAbsSingleAxis (int ILSID, int IAxis, double ITargetPos BOOL bWait);</code>
Parameters	IAxis → xPos axis 1, 2 or 3 ITargetPos → Absolute Position value to move to bWait → function shall return after reaching position (= TRUE) or directly after sending the command (= FALSE)
Example	<code>pTango->Xpos3MoveAbsSingleAxis (1, 2, 10.5, FALSE); // Move 2nd xPos axis to 10.5</code>

Xpos3MoveRelSingleAxis	
Description	Relative Move for an xPos axis (!xmr)
C++	<code>int LSX_Xpos3MoveRelSingleAxis (int ILSID, int IAxis, double IDistance BOOL bWait);</code>
Parameters	IAxis → xPos axis 1, 2 or 3 IDistance → desired position value bWait → function shall return after reaching position (= TRUE) or directly after sending the command (= FALSE)
Example	<code>pTango->Xpos3MoveRelSingleAxis(1, 3, -0.5, FALSE); // Move 3rd xPos axis 0.5mm backwards</code>

9. Error Codes

9.1. Tango Error Messages

0	no error
1	no valid axis name
2	no executable instruction
3	too many characters in command line
4	invalid instruction
5	number is not inside allowed range
6	wrong number of parameters
7	! or ? is missing or not allowed
8	no TVR possible, while axis active
9	no ON or OFF of axis possible, while TVR active
10	function not configured
11	no move instruction possible, while joystick enabled
12	limit switch actuated
13	function not executable, because encoder detected
14	error during calibration (limit switch not released)
15	error during calibration (opposing limit switch actuated)
21	multiple axis moves are forbidden (e.g. during initialization)
22	automatic or manual move is not allowed (e.g. door open or initialization)
27	emergency STOP is active
29	servo amplifiers are disabled (switched OFF)
30	safety circuit out of order
32	move discarded target outside limit
70	wrong CPLD data
71	ETS error
72	parameter is write protected (check lock bits)
73	internal error, e.g. eeprom data corruption
74	closed loop switched off due to parameter change, deviation or enc. Error
75	could not enable axis correction, or axis correction was disabled
76	io extension error (output overload on IO1 or Multi-IO connector)
77	io/xPos internal bus communication error
78	HDI input device error
79	xPos module error
80	internal error: HDI ISR not running
81	internal error: Encoder ISR not running
82	overload on motor connector +5V (PCI-E/DT-E: also on +5V of AUX I/O)
83	overload on AUX I/O +5V supply
84	overload on encoder +5V supply
85	overload on AUX I/O +12V supply or AUX mini +24V supply
86	low brake output voltage
87	overload on motor 4 connector +5V
88	overload on a supply output pin (latched overload state), clear by “!err”
89	not executable while in standby mode
90	temperature error
91	encoder error

9.2. Error Messages for SlideExpress and TrayExpress

Error	Meaning	Explanation / Solution
100	hardware missing (IO1)	PCB option IO1 not installed inside TANGO
101	magazine not correctly seated	proof if magazine is seated correctly
102	magazine slot is empty	the slide index points to an empty slot
103	magazine slot is occupied	the slide index points to an occupied slot
104	sensor reports get failure	the slides are still visible from glass sensor
105	sensor reports put failure	the glass sensor did not detect the slide
106	sensor overmodulation	
107	magazine unknown	
108	ejector timeout	proof if ejector is mechanically blocked
109	priority handler is rear	
110	priority handler is in front	
111	priority handler is not locked	
112	priority handler position not clear	
113	priority handler timeout (front)	
114	priority handler timeout (middle)	
115	priority handler timeout (rear)	
116	front door is open	proof if front door is completely closed
117	timeout close door	
118	no priority handler available	slide index for row value must not be zero
119	gripper is not empty	proof signal from clip detector inside gripper
120	gripper contains unknown clip or slide(s)	
121	system not yet initialized	calibrate at first
122	clip not correctly seated in gripper	proof signal from clip detector inside gripper
123	clamp not open	
124	tray on stage	
125	no tray in gripper	
126	step mode finished	
127	POS3 stop input	
128	no tray on stage	
129	amplifier OFF from crash detection	

9.3. Error Messages of the RFID Interface

130	RF connect
131	RF timeout
132	RF address
133	RF NAK
134	RF sync
135	RF cancel
136	RF not OK
137	RF length
138	RF checksum

9.4. Error Messages of the Piezo Z-Stage

140	Piezo connect
141	Piezo timeout
142	Piezo address

9.5. Error Messages of Custom Handling Systems

Error	Meaning	Explanation / Solution
150	HL connect	the master is not yet configured
151	HL timeout	none or too slow response from slave
152	HL cal X	slave scara arm calibration problem
153	HL cal Y	slave magazine calibration problem
154	HL cal Z	slave vertical axis calibration problem
155	HL insert	
156	HL eject	
157	HL get tray	
158	HL put tray	
159	HL protocol	
160	HL hall tray detection	none of the 2 outer tray hall sensors actuated (tray not present?)
161	clamp electronic	no response from stage ETS
162	no tray on stage but RFID read	inconsistent hardware information clamp vs RFID
163	escape position Z	The Z axis is not in the safe escape position
164	HL Tray alignment	At least one tray is misaligned in the magazine
165	tray on stage but no RFID	inconsistent hardware information clamp vs RFID
166	HM motor flap timeout	The motorized flap is stuck
167	HM door open	The loading door is open
168	HM door just opened	movement of the motorized flap was interrupted from opening the loading door
169	HL laser alignment	The position of comb tines is unexpected
170	HL laser count	The number of detected comb tines is not correct
171	HL slot alignment	At least one comb tine is misaligned
172	gripper not open	Gripper did not open completely
173	cal error pos3	Error while calibrating pos3 module axis
174	loader at barcode positions	Some instructions are not possible in this position and so report error 174
175	limit switch not reached	Axis should be moved into a limit switch but did not reach it
176	IO2 hardware missing	The required IO2 module is not present (not installed or defective)
177	gripping timeout	Timeout while trying to close the gripper
178	unable to free gripper	Unable to free gripper in magazine while cal
179	error stop latch	Stop- was detected -> Cal required
180	magazine not empty	Magazine must be empty, e.g. for the “!mol” instruction

9.6. DLL Error Messages

As returned from DLL function calls.

0	no error	
4001	A passed pointer is NULL, a file error or failed save	
4002	A parameter value or string length is out of range	(in the function call or in a controller reply)
4003	Function call with wrong LSID or wrong axis	(Axis number or LSID out of range)
4004	Unknown interface type	(parameter of Connect/ConnectEx/ConnectSimple)
4005	Error while initializing interface	(connecting to the interface failed)
4006	No connection to the controller	(e.g., function used before connecting to controller)
4007	Timeout while reading from interface	(no reply from the controller)
4008	Error during command transmission to the controller	(send or receive error, received data error)
4009	Command aborted by SetAbortFlag call	
4010	Command is not supported by the controller	(due to firmware version or controller type)
4011	Manual Joystick mode switched on	(can occur with SetJoystickOn/Off function)
4012	No move command possible, because manual joystick enabled	
4013	Closed Loop Controller Timeout	(could not settle within target window)
4014	Error while calibrating	(Limit switch not released, timeout or cal/rm error)
4015	Limit switch actuated in travel direction	(prevents or stops axis travel)
4016	Repeated vector start	(here: if the axis is already traveling)
4031	Not possible to switch on joystick, because move active	
4032	Software limits undefined	

Errors from 4100 are used to forward error numbers from the controller.

Then, the DLL adds +4100 to the controller's error number (e.g., 5 → 4105).

Please refer to the [TANGO Error Messages](#) above, chapters 9.1 to 9.5.

4100	no error
4101	No valid axis name
4102	No executable instruction
4103	Too many characters in command line
4104	Invalid instruction
4105	Number is not inside allowed range
4106	Wrong number of parameters
4107	Either ! or ? is missing
4108	No TVR possible, while axis active
4109	No ON or OFF of axis possible while TVR active
4110	Function not configured
4111	No move instruction possible while joystick enabled
4112	Limit switch actuated
4113	Function not executable, because encoder detected

10. Document Revision History

No.	Revision	Date	Changes	Remarks
01	A	26. Feb. 2009	Initial version	
02	B	27. Oct. 2011	New MW logo and appearance, Added new Error Codes, Added HwFactor, HwFactorB, ZwFactor, GetKey, GetKeyLatch, ClearKeyLatch	
03	C	22. Mar. 2013	Added: GetAccelFunc, SetAccelFunc GetSwitchType, SetSwitchType GetMotorSteps, SetMotorSteps Chapter 5: SlideExpress Interface	
04	D	08. Nov. 2013	Added: Chapter 2.4 LabVIEW Support	
05	E	24. Mar. 2014	Chapter 2.4 reformatted to Arial text	
06	F	18. Sep. 2014	Added: GetCommandTimeout SetCommandTimeout	
07	G	11. Jul. 2016	general review Chapter 6: TrayExpress interface	
08	H	04. Jul. 2017	Added: GetSnapshotMode SetSnapshotMode SetSnapshotCount SetSnapshotPosArray ClearSnapshotPosArray GetSnapshotIndex SetSnapshotIndex Updated Error Codes Added ConnectSimple Interface Type -1	Based on Tango_DLL 1.384 (ML)
09	I	16. Aug. 2017	Added: SetAuxDigitalOutput Corrected IO descriptions	Based on Tango_DLL 1.385 (ML)
10	J	19. Oct. 2017	Added: SetLedBright	Based on Tango_DLL 1.387 (ML)
11	K	01. Nov. 2017	Added: Chapter 3.3 API State Diagram	
12	L	22. Jan. 2018	new: Chapter 7 Express IFC Extensions	Implemented since version 1.388 (FD)
13	M	28. Aug. 2018	Update of Chapter 8	
14	N	18. Dec. 2018	new: SetCabinLED / GetCabinLED SetLabelLED / GetLabelLED	Implemented since version 1.397 (FD)
15	O	19. Feb. 2019	Calibrate returns more specific error code GetLoaderType return value expanded prevent endless loop at removed USB	Implemented since version 1.398 (FD)
16	P	8. Mar. 2019	Update list of Tango error messages	
17	Q	2. Nov. 2020	Added: GetResolution / SetResolution Bugfix: USB endless wait after loosing USB connection (e.g. unplug or Tango power down or Tango reset)	implemented since version 1.399 (FD)
18	R	13. Apr. 2021	Added: GetAutoStatus Bugfix: LSX_Connect, LSX_LoadConfig correct description LSX_SetActiveAxes()	implemented since version 1.401 (FD)
19		06.Jan. 2022	Document type changed from .odt to .docx, new Table of Contents	(ML)
20	S	18. Jan. 2022	Entirely revised DLL documentation, added connecting over Ethernet.	Based on Tango_DLL 1.403 (ML)

21		19. Jan. 2022	Added GetTangoVersion, GetErrorString, SetControllerFactorSingleAxis, GetControllerFactorSingleAxis Improved explanations	(ML)
22		20. Jan. 2022	Added JoyChangeAxis, GetJoyChangeAxis, GetHdiKeys, ConnectEx, SetAnalogOutputMode, SetAnalogOutputMode, StopAxesEx	(ML)
23		21. Jan. 2022	Added additional handling system functions (10) and xPos functions (4) Listed all available trigger functions	(ML)
24		25. Jan. 2022	Added further functions and TVR section	Based on Tango_DLL 1.403 (ML)
25		31. Jan. 2022	Documented HdiSpeedIndex and TVR	Based on Tango_DLL 1.403 (ML)
26	T	31. Jan. 2022	Release for DLL 1.403 and 1.404	Based on Tango_DLL 1.403/1.404 (ML)
27	U	10. Mar. 2022	Added GetControllerTWDelaySingleAxis, SetControllerTWDelaySingleAxis, GetBISmoothSingleAxis, SetBISmoothSingleAxis, GetStA Switched IsVel descriptions to the Status Request chapter	Based on Tango_DLL 1.405 (ML)
28		28. Mar. 2022	Added SetWriteLogText Release for DLL 1.406	Based on Tango_DLL 1.406 (ML)
29		29. Mar. 2022	Added description for Get / Set TriggerAxis, Get / Set TriggerSignalLength, Get / Set TriggerDistance, Get / Set TriggerCompensation, Get / Set TriggerEncoder, Get / Set TriggerFrequency, Get / Set TriggerOutput, Get / Set 2ndTriggerDelay, Get / Set 2ndTriggerWidth, Get / Set 2ndTriggerFrequency	(ML)
30	V	30. Mar. 2022	Added Get / Set TriggerRange, Get / Set TriggerPositionList, Get / Set TriggerPositionListIndex, Get / Set TriggerPositionListEntries, Get / Set TriggerLevel Error Messages 172 to 180 Corrected the description of GetRefSpeed, SetRefSpeed Updated, corrected several descriptions	Release for Tango_DLL 1.406 (ML)
31		27. Jun. 2022	Added GetSecVel, SetSecVel, SetSecVelSingleAxis Corrected vel units (to not only r/sec)	Based on Tango_DLL 1.409 (ML)
32		22. Aug. 2022	Corrected GetSlide "mode" parameter	
33	W	25. Nov. 2022	Added GetAuxDigitalInput	Based on Tango_DLL 1.410 (ML)
34		21. Dec. 2022	Added SetDigitalOutputE	Based on Tango_DLL 1.412 (ML)
35		17. Jan. 2023	Added ClearProtocolWindow	Based on Tango_DLL 1.413 (ML)
36		26. Jan. 2023	Changed GetTray/Puttray number of slots	(ML)
37	X	01. Feb. 2023	Changed DLL Version to 1.414	Release for Tango_DLL 1.414 (ML)
38		15. Feb 2023	Corrected error messages description, according to new DLL version 1.415 Corrected and more clearer function descriptions	Based on Tango_DLL 1.415 (ML)
39	Y	21. Feb. 2023	Added Get / Set EncoderSingleAxis	Based on Tango_DLL 1.415 (ML)



40	Z	20. Nov. 2023	Updated DLL Interface Integration info	Release for Tango_DLL 1.418 (ML)
41	ZA	27. Nov. 2023	Added GetClipType for SlideExpress	Release for Tango_DLL 1.419 (ML)
42		28. Mar. 2024	Improved introduction (Chapters 1 and 2)	(ML)
43	ZB	02. April 2024	Released	(ML)
44	ZC	16. April 2024	Updated information of MSVC runtime, changed 2010 to 2017, and their download	Release for Tango_DLL 1.500 (ML)
45	ZD	16. May 2024	Improved and corrected description of DLL usage and initialization, improved pitch and gear documentation	(ML)
46	ZE	13. Aug. 2024	Corrected description of SetTriggerRange / GetTriggerRange Added MATLAB support	Based on TANGO DLL 1.520 (ML)
47	ZF	19. Sep. 2024	Improved SendStringPosCmd description Improved descriptions of several functions	Based on TANGO DLL 1.520 (ML)