TANGO-DLL Documentation

Of TANGO DLL

Version 1.419

and

Version 1.500

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

Table of Contents

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++
- National Instruments LabVIEW
- Embarcadero Delphi 2007 and Delphi XE
- Java

- 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.

Required files:

- Tango_DLL.dll
- Tango.h
- Tango.cpp

Visual C++ example for controlling a Tango:

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

CTango*  pTangoLSID;

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 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:

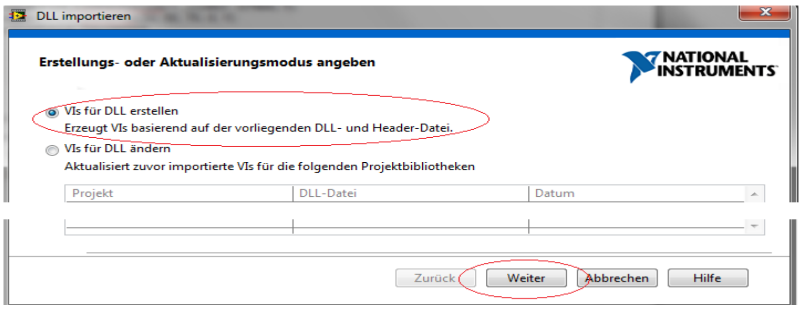
```
Dim return value1 As Integer
Dim return value2 As Integer
Dim return value3 As Integer

...
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.



DLL importieren

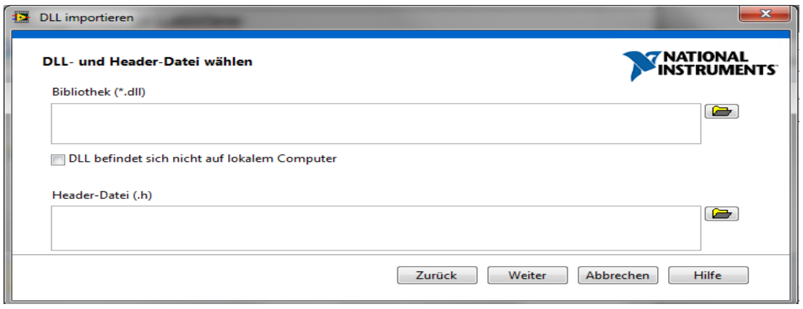
Erstellungs- oder Aktualisierungsmodus angeben

[✓ VIs für DLL erstellen]
Erzeugt VIs basierend auf der vorliegenden DLL- und Header-Datei.

VIs für DLL ändern
Aktualisiert zuvor importierte VIs für die folgenden Projektbibliotheken

Projekt	DLL-Datei	Datum

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



DLL importieren

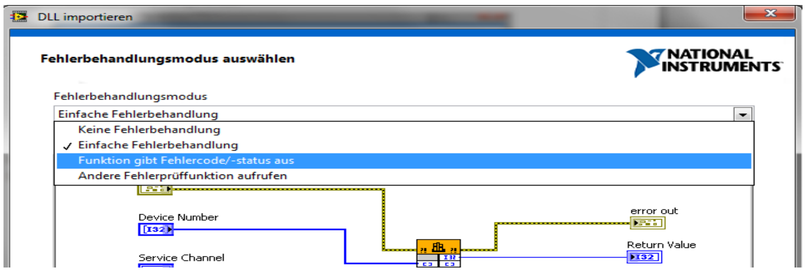
DLL- und Header-Datei wählen

Bibliothek (*.dll)

DLL befindet sich nicht auf lokalem Computer

Header-Datei (.h)

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.



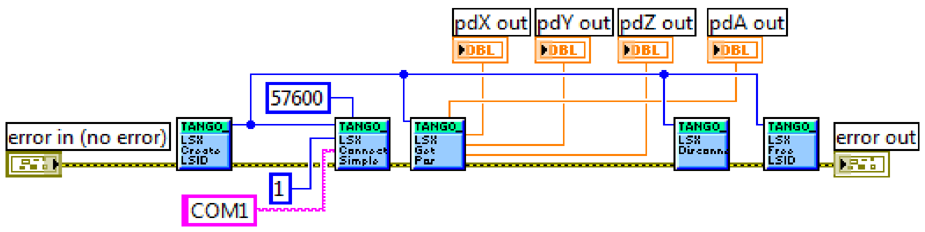
DLL importieren

Fehlerbehandlungsmodus auswählen

Fehlerbehandlungsmodus
Einfache Fehlerbehandlung
Keine Fehlerbehandlung
✓ Einfache Fehlerbehandlung
Funktion gibt Fehlercode/-status aus
Andere Fehlerprüffunktion aufrufen

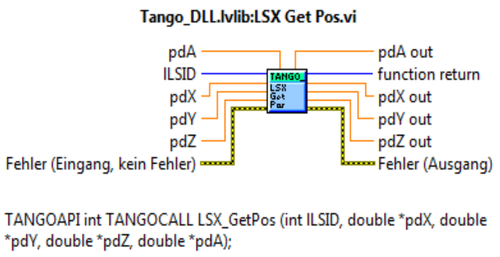
7) After selecting the path and name for the project library the error handling mode should at least contain a simple error handling or even an error handling with return function of TANGO_DLL.dll included.
8) The configuration of the VIs should not be changed and the import process can start.

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.



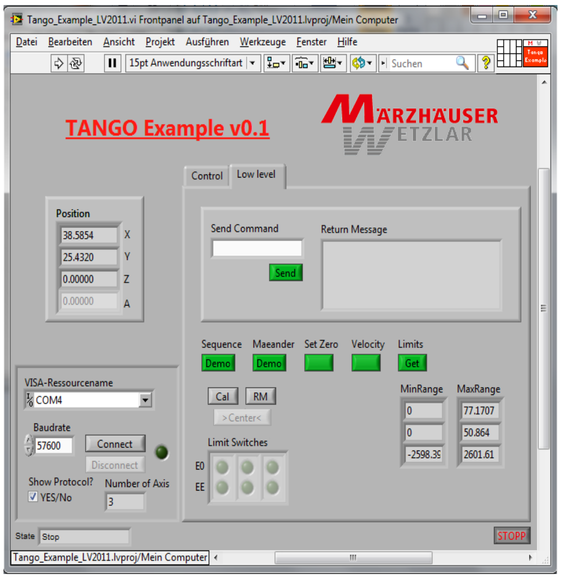
TANGOAPI int TANGOCALL LSX_GetPos (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);

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.


Tango_Example_LV2011.vi Frontpanel auf Tango_Example_LV2011.lvproj/Mein Computer

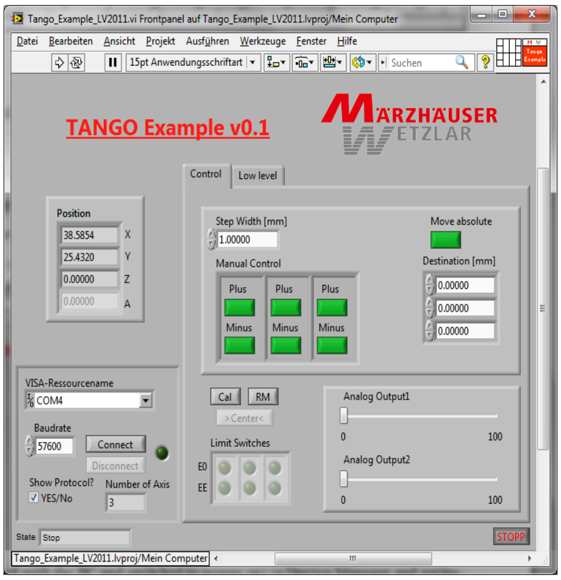
TANGO Example v0.1

Position: X 38.5854, Y 25.4320, Z 0.00000, A 0.00000

VISA-Ressourcename: COM4
Baudrate: 57600
Show Protocol? ✓ YES/No
Number of Axis: 3

Low level tab: Send Command, Return Message
MinRange: 0, 0, -2598.39
MaxRange: 77.1707, 50.864, 2601.61

State: Stop


Tango_Example_LV2011.vi Frontpanel auf Tango_Example_LV2011.lvproj/Mein Computer

TANGO Example v0.1

Position: X 38.5854, Y 25.4320, Z 0.00000, A 0.00000

VISA-Ressourcename: COM4
Baudrate: 57600
Show Protocol? ✓ YES/No
Number of Axis: 3

Control tab: Step Width [mm] 1.00000
Destination [mm]: 0.00000, 0.00000, 0.00000
Analog Output1: 0 – 100
Analog Output2: 0 – 100

State: Stop

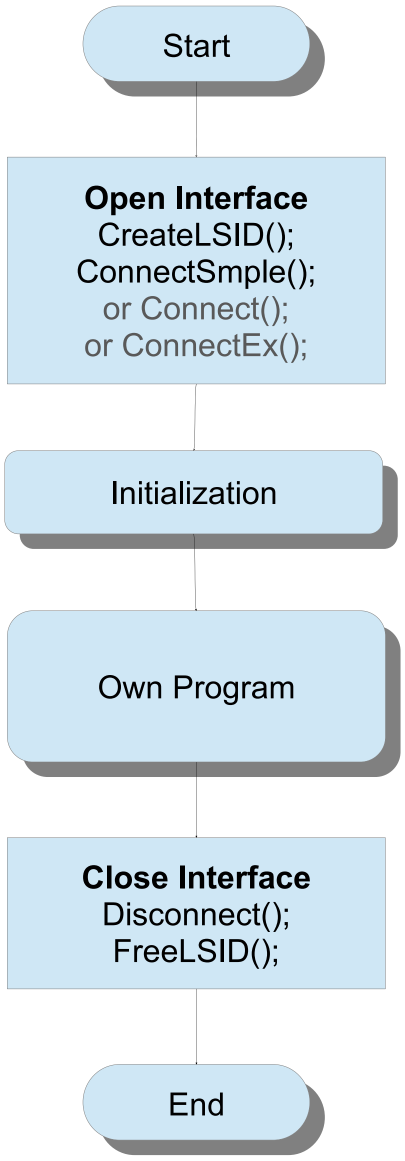
This example VI looks for a TANGO (connected with the PC and switched to power on) in Device Manager and writes the corresponding COM-Port in VISA-Ressourcename as a pre-selection. The default baud-rate is 57600. After selecting the correct COM-Port the user is able to connect to TANGO.
The program gives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

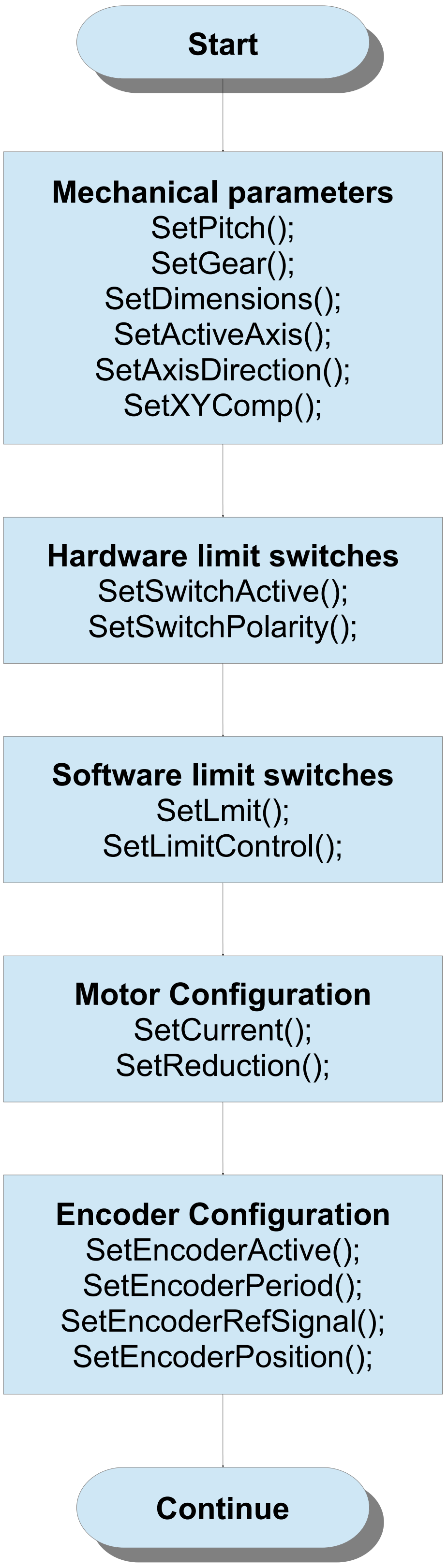
3. General Information of DLL Usage

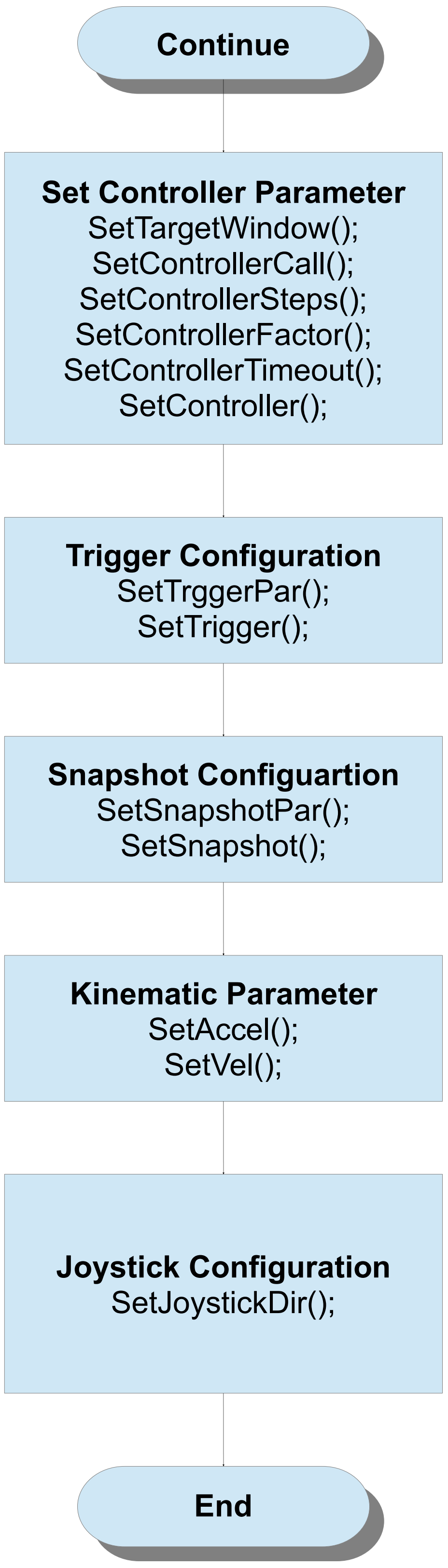
The following flow chart shows how to establish and end Tango communication and is valid for all different physical layer like RS232, USB, PCI, PCI-E or Ethernet. All Tango application programs, independent of chosen and involved programming language, should follow this guideline.
DLL functions are listed and described in detail in the next chapters.



3.1. Initialization of Controller

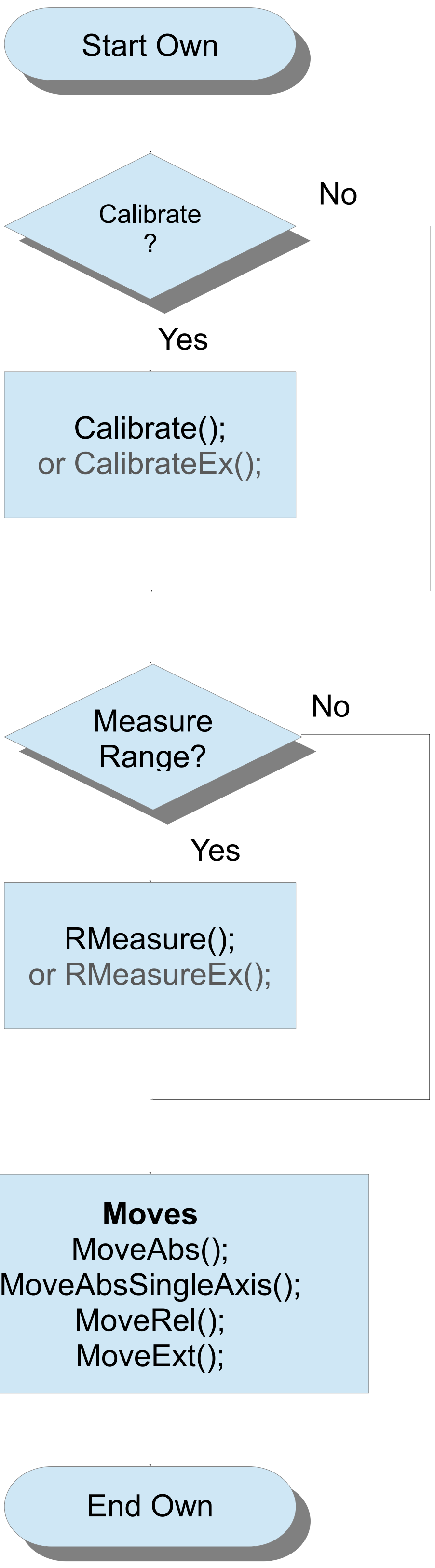
Most Märzhäuser stages are ETS coded. The Tango uses the available ETS data for stage initialization. Several parameters then are correctly predefined and write protected.
Note: Any mechanics may be damaged if wrong parameters are used. Pleas be careful to use correct stage data to prevent any damage. Follow below flow chart to transmit individual settings.




Continue
Set Controller Parameter
SetTargetWindow();
SetControllerCall();
SetControllerSteps();
SetControllerFactor();
SetControllerTimeout();
SetController();
Trigger Configuration
SetTrggerPar();
SetTrigger();
Snapshot Configuartion
SetSnapshotPar();
SetSnapshot();
Kinematic Parameter
SetAccel();
SetVel();
Joystick Configuration
SetJoystickDir();
End

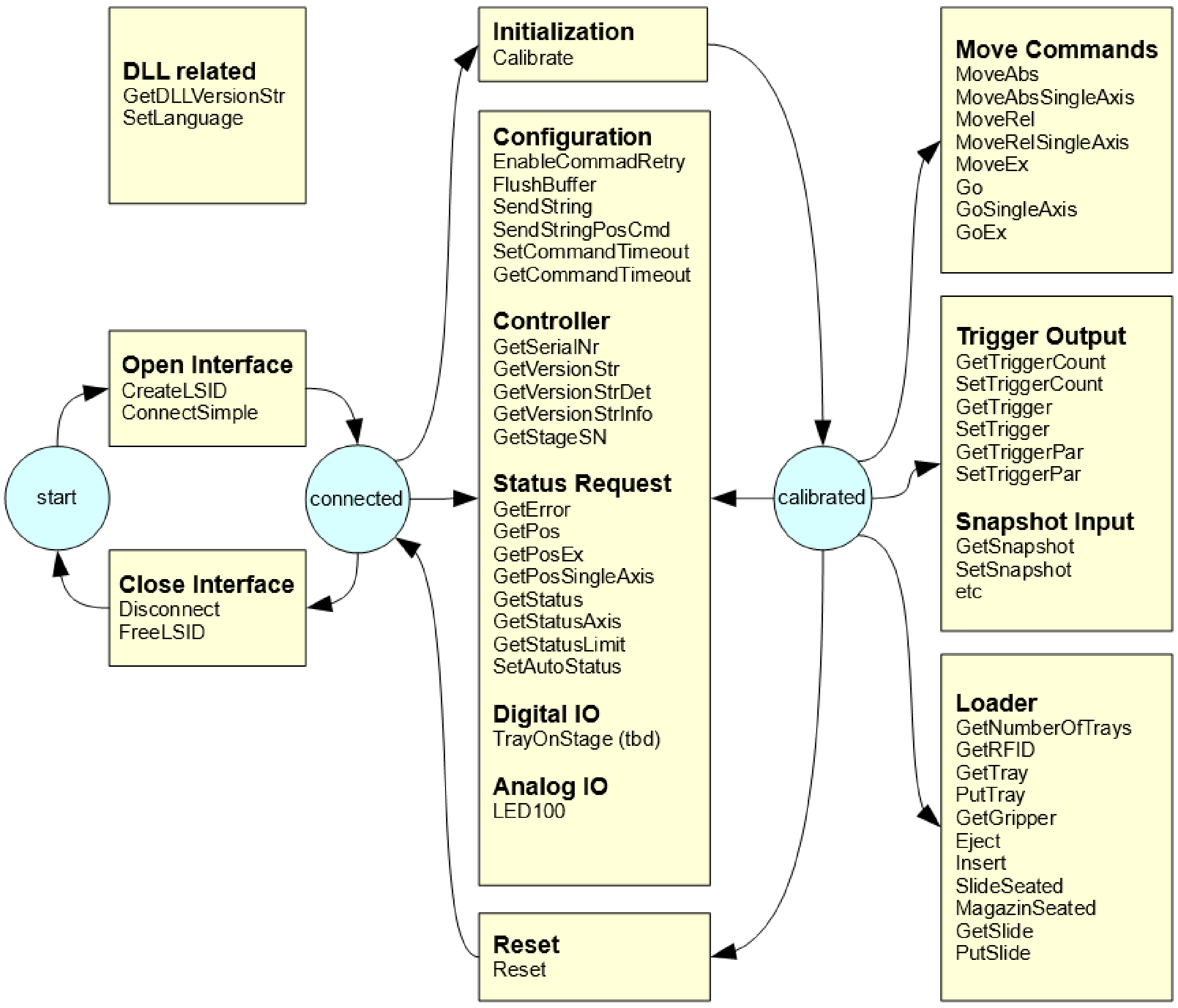
3.2. Own Program Section

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	38
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

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 (?tvrf)	93
SetFactorTVR	Set AUX-IO TVR clock and direction input factor (!tvrf)	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 (*"encmask"!*)	94
SetEncoderActive	Select encoder to be activated after calibration (*"encmask")*	95
GetEncoderMask	Read current activation status of encoders *("enc" command!*)	95
SetEncoderMask	Activates / deactivates encoders ("*enc")*	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 µs)	109
SetTriggerSignalLength	Set the signal length for an active trigger pulse (pulse width in µs)	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 µs)	110
SetTriggerCompensation	Set the trigger compensation value (look forward time in µs)	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 2nd 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++	int LSX_CreateLSID(int *plLSID);
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	int Tango1, Tango2; pTango->CreateLSID(&Tango1); *// create ID for first Tango* pTango->CreateLSID(&Tango2); *// create ID for second Tango*

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++	int LSX_ConnectSimple(int lLSID, int lAnInterfaceType, char *pcAComName, int lABaudRate, BOOL bAShowProt);
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	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

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++	int LSX_ConnectEx(int lLSID, TLS_ControlInitPar *pAControlInitPar);
Parameters	***AControlInitPar***: Structure with baud rate, port, protocol etc. information
Example	pTango->ConnectEx (1, &ControlInitPar);

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++	int LSX_LoadConfig (int lLSID, char *pcFileName);
Parameters	***pcFileName*** → file name to be used to read data from. The ini file data structure should be generated from SwitchBoard (ASCII text).

Example	`REQUIRE(LSX_CreateLSID(&g_LSID) == 0);`
	`char* inifile = "C:\\Users\\me\\Desktop\\mytest.ini";`
	`REQUIRE(LSX_LoadConfig(g_LSID, inifile) == 0);`
	`REQUIRE(LSX_Connect(g_LSID) == 0);`
	`//LSX_SetShowProt(g_LSID, TRUE); //overwritten from ini file`
	`REQUIRE(LSX_SetLanguage(g_LSID, "germ") == 0);`

Connect	
Description	The 3rd 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++	int LSX_Connect (int lLSID);
Parameters	-
Example	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

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

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++	int LSX_Disconnect(int lLSID);
Parameters	-

Example	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_)

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++	int LSX_FreeLSID(int lLSID);
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	int Tango1;
	
	pTango->CreateLSID(&Tango1);
	pTango->ConnectSimple(Tango1, …);
	…
	pTango->Disconnect(Tango1);
	pTango->FreeLSID(Tango1);

SetShowProt

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

ClearProtocolWindow

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

SetLanguage

Description	Set language of protocol window
C++	int LSX_SaveConfig (int lLSID, char *pcPLN);
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	pTango->SaveConfig (1, "french"); // Switch protocol 1 language to french

GetCommandTimeout

Description	Read current DLL timeout for read, move and calibration
C++	int LSX_GetCommandTimeout (int lLSID, int *toRead, int *toMove, int *toCal);
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	pTango->GetCommandTimeout(1, &tR, &tM, &tC);

SetCommandTimeout

Description	Set DLL timeout for read, move and calibration
C++	int LSX_SetCommandTimeout (int lLSID, int toRead, int toMove, int toCal);
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	pTango->SetCommandTimeout(1, tR, tM, tC);

EnableCommandRetry

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

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++	int LSX_SetDllNumOfAxes (int lLSID, int lNumOfAxes);
Parameters	***NumOfAxes***: 1, 2, 3 or 4
Example	pTango->SetDllNumOfAxes(1, 3); *// Force the DLL to use 3 axes*

GetSwapZA

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

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++	int LSX_SetSwapZA (int lLSID, BOOL bValue);
Parameters	***Value***: TRUE = swap Z and A, FALSE = no swapped (default)
Example	pTango->GetSwapZA (1, &bSwapped); *// Read the Tango DLL Z-A swap setting*

FlushBuffer

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

GetDLLVersionString

Description	Get DLL version string
C++	int LSX_GetDLLVesionString (int lLSID, char *pcVers, int lMaxLen);
Parameters	***pcVers*** → Buffer, containing return message from DLL ***lMaxLen*** → Limits the max. number of characters to be copied into buffer

Example	char cVersionString[128];
	pTango->GetDLLVesionString (lLSID, cVersionString, 127);
	// Example reply: “DLL64 Version 1.403, Oct 12, 2021, 12:34:33”

SendString	
Description	Send an ASCII string to the Tango or send and receive or receive only.
C++	int LSX_SendString (int lLSID, char *pcStr, char *pcRet, int lMaxLen, BOOL bReadLine, int lTimeOut);
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	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

SendStringPosCmd	
Description	Send move command to Tango as a string and wait for return message.
C++	int LSX_SendStringPosCmd (int lLSID, char *pcStr, char *pcRet, int lMaxLen, BOOL bReadLine, int lTimeOut);
Parameters	***Str*** → Zero-terminated ASCII string, which is to be sent to the controller ***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 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	pTango->SendStringPosCmd(1, “!moa 1 2\r”, pcRet , 256, TRUE, 10000);

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++	int LSX_SetAbortFlag (int lLSID);
Parameters	-

Example	pTango->SetAbortFlag(1);
	pTango->StopAxes(1);
	// closes communication with Tango and sends stop command for all axes

4.3. Controller Information

GetSerialNr	
Description	Reads out the TANGO serial number (?readsn).
C++	int LSX_GetSerialNr (int lLSID, char *pcSerialNr, int lMaxLen);
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	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

GetTangoVersion	
Description	Get TANGO version string (?version). Read the TANGO Controller Version information as ASCII text.
C++	int LSX_GetTangoVesion (int lLSID, char *pcVers, int lMaxLen);
Parameters	***pcVers:*** Pointer to the char buffer, in which the TANGO version text is returned ***lMaxLen***: Limits the max. number of characters to be copied into buffer

Example	char cVersionString[128];
	pTango->GetTangoVesion (lLSID, cVersionString, 127);
	// Example reply: “TANGO-Desktop, Version 1.73, Mar 11 2021 , 13:28:26”

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

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

GetVersionStrInfo	
Description	Provides optional internal information on the controller version (?iver).
C++	int LSX_GetVersionStrInfo (int lLSID, char *pcVersInfo, int lMaxLen);
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	pTango->GetVersionStrInfo(1, pcVersInfo, 16);

GetStageSN	
Description	Provides optional internal information on the stage serial number (?stagesn -1).
C++	int LSX_GetStageSN (int lLSID, char *pcSN, int lMaxLen);
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	pTango->GetVersionStrInfo(1, pcSN, 16);

4.4. Status Requests

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

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++	int LSX_GetErrorString (int lLSID, int lError, char *pcErrorString, int lMaxLen);
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	pTango->GetErrorString(1, 29, &text_array[0], 64); *// read explanation of error 29*

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

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

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++	int LSX_GetPosSingleAxis (int lLSID, int lAxis, double *pdPos);
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	pTango->GetPosSingleAxis(1, 2, &Pos); *// retrieves position of Y-Axis*

SetPos

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

ClearPos

Description	Sets current position and internal position counter to 0 (!clearpos). This function is needed for endless axes, as controller can only process ±1,000 motor revolutions within its parameters. This instruction will be ignored for axes with encoders.
C++	int LSX_ClearPos (int lLSID, int lFlags);
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++	int LSX_GetStatus (int lLSID, char *pcStat, int lMaxLen);
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	pTango->GetStatus(1, &Stat, 16);

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++	int LSX_GetStatusAxis (int lLSID, char *pcStatusAxisStr, int lMaxLen);
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	pTango->GetStatusAxis(1, &StatusAxisStr, 16);

GetStatusLimit

Description	Provides current status of software limits of each axis (?statuslimit).
C++	int LSX_GetStatusLimit (int lLSID, char *pcLimit, int lMaxLen);
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	pTango->GetStatusLimit(1, &Limit, 32);

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.
	
	`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 reserved`
	`20000000 reserved`
	`40000000 reserved`
	`80000000 reserved`
	
	`Example: StaX returns the number (here displayed as hex):`
	`02030307  --> axis is not traveling, no closed loop, no errors`
	` | | | |`
	` | | | +- axis is on (!axis x 1, !pa1)`
	` | | +--- cal and rm are executed`
	` | +----- encoder is active (and was/is active)`
	` +------- HDI is enabled (joy+joydir)`

C++	int LSX_GetStA (int lLSID, int *plStaX, int *plStaY, int *plStaZ, int *plStaA);
Parameters	***plSta***: Pointers to integer, in which the state flags of the axes are returned
Example	pTango->GetStA(1, &lStaX, &lStaY, &lStaZ, &lStaA);

SetAutoStatus	
Description	Switches Auto-Status on/off (!autostatus). Please note: As a rule, AutoStatus mode should not be changed as Tango DLL sets correct mode for travel commands etc., changing Autostatus manually to a value of 0, 2 or 3 could cause errors.
C++	int LSX_SetAutoStatus (int lLSID, int lValue);
Parameter	***Value***: AutoStatus mode: 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	pTango->SetAutoStatus(1, 1);

GetAutoStatus	
Description	Read current state of Auto-Status (?autostatus).
C++	int LSX_GetAutoStatus (int lLSID, int *plAstatus);
Parameter	astatus: Pointer to integer, in which the current state of Auto-State 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	pTango->GetAutoStatus(1, &autostatus);

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++	`int LSX_IsVel (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);`
Parameters	***pdX, pdY, pd Z, pdA***: actual axes velocities in [mm/s]
Example	`pTango->IsVel(1, &vx, &vy, &vz, &va);`

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++	`int LSX_IsVelSingleAxis (int lLSID, int lAxis, double *pdVel);`
Parameters	**lAxis:** 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) **pdVel:** actual axis velocity in [mm/s]
Example	`pTango->IsVel(1, 2, &vel); // returns actual velocity of y axis`

4.5. Settings

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

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

GetActiveAxes	
Description	Provides the axis enable states (?axis).
C++	int LSX_GetActiveAxes (int lLSID, int *plFlags);
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	pTango->GetActiveAxes(1, &Flags);
	if (Flags & 0x01) …; *// Flags Bit 0 = 1 → X-Axis is on*
	if ((Flags & 0x04) == 0) …; *// Flags Bit 2 = 0 → Z-Axis is off or disabled*

SetActiveAxes	
Description	Enable or disable axes (!axis).
C++	int LSX_SetActiveAxes (int lLSID, int lFlags);
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	pTango->SetActiveAxes(1, 3); *// X- and Y-Axes are enabled (Bits 0 and 1 set),* *Z-Axis and A-Axis switched off (Bit 2 = 0, Bit 3 = 0)*

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

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

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

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++	int LSX_SetCalibOffset (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: typically 0-5 [mm]
Example	pTango->SetCalibOffset(1, 1, 1, 1, 1); *// 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)*

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

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

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++	int LSX_GetCalibBackSpeed (int lLSID, int *plSpeed);
Parameters	***Speed***: Speed value in 1/100 revolutions/second
Example	pTango->GetCalibBackSpeed(1, &lSpeed);

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++	int LSX_SetCalibBackSpeed (int lLSID, int lSpeed);
Parameters	***Speed***: Speed value in 1/100 revolutions/second (within parameters of 1 to 100)
Example	pTango->SetCalibBackSpeed(1, 10); *// when calibrating, limit switches are left at 0.1 rev/sec*

GetCalibrateDir

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

SetCalibrateDir

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

GetDimensions

Description	Provides the applied measuring units of axes (?dim)
C++	int LSX_GetDimensions (int lLSID, int *plXD, int *plYD, int *plZD, int *plAD);
Parameters	***XD, YD, ZD, AD***: 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	pTango->GetDimensions(1, &XD, &YD,&ZD,&AD);

SetDimensions

Description	Set measuring units of axes (!dim).
C++	int LSX_SetDimensions (int lLSID, int lXD, int lYD, int lZD, int lAD);
Parameters	***XD, YD, ZD, AD***: 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	pTango->SetDimensions(1, 3, 2, 2, 1); *// X-Axis in degree, Y- and Z-Axis in mm and A-Axis in µm*

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++	int LSX_GetDimensions (int lLSID, int *plValue);
Parameters	***Value***: Resolution units 3 → 1 µm resolution 4 → 0.1 µm resolution (Default) 5 → 10 nm resolution 6 → 1 nm resolution
Example	pTango->GetResolution(1, &resolution);

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. You may specify 5 (10nm) or 6 (1nm) digits to get higher resolution from Tango.
C++	int LSX_SetDimensions (int lLSID, int lValue);
Parameters	***Value***: Resolution units 3 → 1 µm resolution 4 → 0.1 µm resolution (Default) 5 → 10 nm resolution 6 → 1 nm resolution
Example	pTango->SetResolution(1, 5); *// set 5 digits after the decimal point for all axes*

GetPitch

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

SetPitch

Description	Set spindle pitch (!pitch).
C++	int LSX_SetPitch (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: 0.0001 to 72 [mm per motor revolution]
Example	pTango->SetPitch(1, 4, 4, 4, 4); *// Set spindle pitch of all axes to 4mm*

GetGear

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

SetGear

Description	Set gear ratio (!gear).
C++	int LSX_SetGear (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A:*** 0.01 - 1000
Example	pTango->SetGear(1, 4.0, 2.0, 1.0, 1.0); *// programs gear ratios ¼ for Z, ½ for Y and 1/1 for Z and A*

GetMotorSteps

Description	Retrieves number of motor steps (?motorsteps).
C++	int LSX_GetMotorSteps (int lLSID, int *lX, int *lY, int *lZ, int *lA);
Parameters	***X, Y, Z, A***: Number of motor steps
Example	pTango->GetMotorSteps(1, &X, &Y, &Z, &A);

SetMotorSteps

Description	Set number of motor steps. Default 200 for 1,8° stepper motors (!motorsteps).
C++	int LSX_SetMotorSteps (int lLSID, int lX, int lY, int lZ, int lA);
Parameters	***X, Y, Z, A***: Motor steps X, Y, Z and A-Axis
Example	pTango->SetMotorCurrent(1, 200, 200, 400, 24); *// set X, Y to default 200, Z to 400 and A to 24 steps motor type (1.8°, 0.9° and 15° mot.)*

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

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

GetReduction	
Description	Retrieves motor current reduction factor (?reduction).
C++	int LSX_GetReduction (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA)
Parameters	***X, Y, Z, A***: Electrical motor current reduction 0.00 … 1.00 (= 0…100%)
Example	pTango->GetReduction(1, &X, &Y, &Z, &A);

SetReduction	
Description	Set reduction factor of motor current (!reduction).
C++	int LSX_SetReduction (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: Electrical motor current reduction 0.00 … 1.00 (= 0…100%)
Example	pTango->SetReduction(1, 0.1, 0.7, 0.5, 0.5); *// standby current X-Axis = 0.1*rated current, Y-Axis = 0.7*rated current,* *Z- and A-Axis = 0,5*rated current*

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

SetCurrentDelay	
Description	Sets the time delay, after which the motor current is reduced (!curdelay).
C++	int LSX_SetCurrentDelay (int lLSID, int lX, int lY, int lZ, int lA);
Parameters	***X, Y, Z, A***: 0…65000 [ms] (A delay of 0 disables the current reduction = default)
Example	pTango->SetCurrentDelay(1, 100, 300, 1000, 0);

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++	int LSX_GetSpeedPoti (int lLSID, BOOL *pbSpePoti);
Parameter:	The SpePoti flag shows, whether potentiometer is switched on (1) or off (0)
Example	pTango->GetSpeedPoti(1, &flag);

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++	int LSX_SetSpeedPoti (int lLSID, BOOL bSpeedPoti);
Parameters	***SpeedPoti*** = 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	pTango->SetSpeedPoti(1, TRUE); *// switch potentiometer mode on*

GetStopPolarity

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

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++	int LSX_SetStopPolarity (int lLSID, BOOL bHighActiv);
Parameters	***HighActiv***: TRUE→ stop input high active FALSE → stop input low active
Example	pTango->SetStopPolarity(1, FALSE); *// stop input is low active (e.g. normally open switch to ground)*

GetVel

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

SetVel

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

SetVelSingleAxis	
Description	Set velocity of a single axis (!vel x,y,z,a).
C++	int LSX_SetVelSingleAxis (int lLSID, int lAxis, double dVel);
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	pTango->SetVelSingleAxis(1, 2, 10.0); *// sets speed of Y-Axis to 10 [rev/s or mm/s]*

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++	int LSX_GetSecVel (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***pdX, pdY, pdZ, pdA***: Velocity values [mm/s]
Example	pTango->GetSecVel(1, &X, &Y, &Z, &A);

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++	int LSX_SetSecVel (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***dX, dY, dZ, dA***: >0 … max. speed [mm/s]
Example	pTango->SetSecVel(1, 20.0, 15.0, 0.5, 10);

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++	int LSX_SetSecVelSingleAxis (int lLSID, int lAxis, double dVel);
Parameters	***Axis***: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) ***Vel***: >0 … max. speed [mm/s]
Example	pTango->SetSecVelSingleAxis(1, 2, 10.0); *// sets secure speed of Y-Axis to 10 mm/s*

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++	int LSX_GetVelFac (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***dX, dY, dZ, dA***: Velocity factor, default = 1
Example	pTango->GetVelFac(1, &X, &Y, &Z, &A);

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++	int LSX_SetVelFac (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***dX, dY, dZ,d A***: Velocity reduction factor, within parameters 0.01 -- 1.00
Example	pTango->SetVelFac(1, 1, 1, 0.1, 0.1); *// reduces velocity of Z and A axes to 1/10 of nominal velocity*

GetAccel	
Description	Retrieves acceleration (?accel).
C++	int LSX_GetAccelFunc (double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***dX, dY, dZ, dA***: Acceleration values [m/s²]
Example	pTango->GetAccel(1, &X, &Y, &Z, &A);

SetAccel	
Description	Set acceleration (!accel).
C++	int LSX_SetAccel (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	**dX, dY, dZ, dA**: 0.01 - 20.00 [m/s²]
Example	pTango->SetAccel(1, 1.0, 1.5, 0, 0);

SetAccelSingleAxis	
Description	Set acceleration of a single axis (!accel).
C++	int LSX_SetAccelSingleAxis (int lLSID, int lAxis, double dAccel);
Parameters	***Axis***: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) ***Accel***: Acceleration 0.01 - 20.00 [m/s²]
Example	pTango->SetAccelSingleAxis(1, 3, 1,0); *// sets acceleration of Z-Axis to 1.0 m/s²*

GetAccelFunc	
Description	Retrieves acceleration function (?accelfunc).
C++	int LSX_GetAccelFunc (int lLSID, int *lX, int *lY, int *lZ, int *lR);
Parameters	***lX, lY, lZ, lR***: Acceleration function of the axes (pointer to variable) 0 = trapezoidal acceleration 1 = s-curve acceleration
Example	pTango->GetAccelFunc(1, &lX, &lY, &lZ, &lR);

SetAccelFunc	
Description	Sets acceleration function: 0 for trapezoidal, 1 for s-curve (!accelfunc).
C++	int LSX_SetAccelFunc (int lLSID, int lX, int lY, int lZ, int lR);
Parameters	***lX, lY, lZ, lR***: Acceleration function for the axes 0 = trapezoidal acceleration 1 = s-curve acceleration
Example	pTango->SetAccel(1, lX, lY, lZ, lR);

GetStopAccel

Description	Provides deceleration for error conditions (?stopaccel).
C++	int LSX_GetStopAccel (int lLSID, double *pdXD, double *pdYD, double *pdZD, double *pdAD);
Parameters	***XD, YD, ZD, AD***: Deceleration values [m/s²]
Example	pTango->GetStopAccel(1, &XD, &YD, &ZD, &AD);

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++	int LSX_SetStopAccel (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: Brake acceleration, within parameters 0.01 to 20 [m/s²]
Example	pTango->SetStopAccel(1, 1.5, 1.5, 1.5, 1.5);

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++	int LSX_GetBlSmoothAxis (int lLSID, int lAxis, int *plBlSmooth);
Parameters	lAxis 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) lBlSmooth Pointer to int for returning blsmooth mode of the specified axis
Example	pTango->LSX_ GetBlSmoothSingleAxis(1, 2, &lBlSmooth); // Get blsmooth of Y

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++	int LSX_SetBlSmoothSingleAxis (int lLSID, int lAxis, int lSmooth);
Parameters	lAxis 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) lBlSmooth New blsmoth mode for the axis
Example	pTango->LSX_ SetBlSmoothSingleAxis (1, 2, 0); // Set blsmooth of Y to 0

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++	int LSX_LStepSave (int lLSID);
Parameters	-
Example	pTango->LStepSave(1);

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

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++	int LSX_Calibrate (int lLSID);
Parameters	-
Example	pTango->Calibrate(1);

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++	int LSX_CalibrateEx (int lLSID, int lFlags);
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	pTango->CalibrateEx(1, 6); *// only calibrate Y- and Z-Axis (Bit 1 and 2 set = 2+4 = 6)*

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++	int LSX_RMeasure (int lLSID);
Parameters	-
Example	pTango->RMeasure(1);

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++	int LSX_RMeasureEx (int lLSID, int lFlags);
Parameters	***Flags***: Bit mask Bit 2 = 1 → calibrate Z-Axis Bit 2 = 0 → Do not calibrate Z-Axis ...
Example	pTango->RMeasureEx(1, 3); *// 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++	int LSX_GetDelay (int lLSID, int *plDelay);
Parameters	***Delay***: Delay [ms]
Example	pTango->GetDelay(1, &Delay);

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++	int LSX_SetDelay (int lLSID, int lDelay);
Parameters	***Delay***: 0 - 10000 [ms]
Example	pTango->SetDelay(1, 1000); *// 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++	int LSX_MoveAbs (int lLSID, double dX, double dY, double dZ, double dA, BOOL bWait);
Parameters	***X, Y, Z, A***: ± 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	pTango->MoveAbs(1, 10.0, 10.0, -10.0, 10.0, TRUE);

MoveAbsSingleAxis	
Description	Positions a single axis at the transferred position (!moa).
C++	int LSX_MoveAbsSingleAxis (int lLSID, int lAxis, double dValue, BOOL bWait);
Parameters	***Axis***: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) ***Value***: Position, command depends on measuring unit (dimension)
Example	pTango->MoveAbsSingleAxis(1, 2, 10.0); *// position Y-Axis absolutely at 10mm (dimension=2)*

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++	int LSX_MoveEx (int lLSID, double dX, double dY, double dZ, double dA, BOOL bRelative, BOOL bWait, int lAxisCount);
Parameters	***X, Y, Z, A***: Position vectors ***Relative***: When Relative = FALSE, values of X, Y, Z and A are interpreted as absolute coordinates when Relative = TRUE, they are interpreted as relative coordinates to current position ***Wait***: If Wait = TRUE is set, function doesn't return before reaching the target position, otherwise it 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	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*

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++	int LSX_MoveRel (int lLSID, double dX, double dY, double dZ, double dA, BOOL bWait);
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	pTango->MoveRel(1, 10.0, 10.0, -10.0, 10.0, TRUE);

MoveRelSingleAxis

Description	Move single axis relative (!mor).
C++	int LSX_MoveRelSingleAxis (int lLSID, int lAxis, double dValue, BOOL bWait);
Parameters	***Axis***: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) ***Value***: Distance, command depends on set measuring unit
Example	pTango->MoveRelSingleAxis(1, 3, 5,0); *// Z-Axis is moved by 5mm in positive direction*

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++	int LSX_MoveRelShort (int lLSID);
Parameters	-

Example	pTango->SetDistance(1, 1.0, 1.0, 0, 0);
	for (i = 0; i < 10; i++) pTango->MoveRelShort(1);
	// position X- and Y-Axis 10 times relatively by 1mm

GetDistance

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

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++	int LSX_SetDistance (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: Min-/max- travel range, values depend on measuring unit.
Example	pTango->SetDistance(1, 1, 2, 0, 0); *// 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*

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++	int LSX_Go (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: ± Travel range, command depends on measuring unit
Example	pTango->Go(1, 10.0, 10.0, -10.0, 10.0);

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++	int LSX_GoSingleAxis (int lLSID, int lAxis, double dValue);
Parameters	***Axis***: 1, 2, 3, 4 (corresponding to X, Y, Z, A axes) ***Value***: ± Travel range, command depends on measuring unit
Example	pTango->GoSingleAxis(1, 2, 12.34); //move Y to target position 12.34

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++	int LSX_GoEx (int lLSID, double dX, double dY, double dZ, double dA, int lAxisCount);
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	pTango->GoEx(1, 2.0, 3.0, 56.78, 67.89, 2); *// X and Y are moved relatively by 2 or 3 while Z and A will not move*

GetDigJoySpeed

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

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++	int LSX_SetDigJoySpeed (int lLSID, double dX, double dY, double dZ, double dA);
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	pTango->SetDigJoySpeed(1, 0, 10.0, 25.0, 0); *// Axes X and A - speed 0 and Joystick operation „OFF“,* *Axis Y - speed 10.0 r/sec and Joystick operation „ON“,* *Axis Z -speed 25.0 r/sec and Joystick operation „ON“*

StopAxes

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

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++	int LSX_StopAxesEx (int lLSID, int lFlags);
Parameters	***lFlags:*** 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.
C++	int LSX_WaitForAxisStop (int lLSID, int lAFlags, int lATimeoutValue, BOOL *pbATimeout);
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	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.

4.7. Joystick and Handwheel

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

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

GetJoystickDir	
Description	Retrieves axis direction for the Joystick and other HDI input devices (?joydir). Individual axes can be reversed or disabled.
C++	int LSX_GetJoystickDir (int lLSID, int *plXD, int *plYD, int *plZD, int *plAD);
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	pTango->GetJoystickDir(1, &XD, &YD, &ZD, &AD);

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++	int LSX_SetJoystickDir (int lLSID, int lXD, int lYD, int lZD, int lAD);
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	pTango->SetJoystickDir(1, 1, 1, -1, 0); *// X- and Y-Axis positive direction, Z-Axis negative direction, A-Axis blocked*

GetJoystick

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

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

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

GetHandWheel	
Description	Retrieves hand wheel status (?hw). Not supported by TANGO controllers! Use GetJoystick().
C++	int LSX_GetHandWheel (int lLSID, BOOL *pbHandWheelOn, BOOL *pbPositionCount, BOOL *pbEncoder);
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	pTango->GetHandWheel(1, &HandWheelOn, &PositionCount, &Encoder);

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

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

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

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++	int LSX_SetJoystickWindow (int lLSID, int lAValue);
Parameters	***AValue***: Analogue signal range (as digits) in which axes do not move. 0 ... 100
Example	pTango->SetJoystickWindow(1, 30);

GetHwFactor

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

SetHwFactor

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

GetHwFactorSingleAxis

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

SetHwFactorSingleAxis

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

GetHwFactorB

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

SetHwFactorB

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

GetHwFactorBSingleAxis

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

SetHwFactorBSingleAxis

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

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

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

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

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

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

ClearKeyLatch	
Description	Clear latched key state(s) of one specified (1-4) or all (0) keys. No bitmask. (!keyl)
C++	int LSX_ClearKeyLatch (int lLSID, int lKey);
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	pTango->ClearKeyLatch(1, 0); // Clear all

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++	int LSX_GetHdiSpeedIndex (int lLSID, int *plSi1, int *plSi2, int *plSi3, int *plSi4);
Parameters	Pointers to int for returning the speed index
Example	pTango->LSX_GetHdiSpeedIndex (1, &lSpeedIndex);

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++	int LSX_SetHdiSpeedIndex (int lLSID, int lSi1, int lSi2, int lSi3, int lSi4);
Parameters	New speed index for the HDI axes
Example	pTango->LSX_SetHdiSpeedIndex (1, 0, 0, 0, 0); // Set speed index to 0

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++	int LSX_GetHdiSpeedIndexSingleAxis (int lLSID, int lAxis, int *plSi);
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	pTango->LSX_ GetHdiSpeedIndexSingleAxis(1, 2, &lSpeedIndex); *// Get Y index*

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++	int LSX_SetHdiSpeedIndexSingleAxis (int lLSID, int lAxis, int lSi);
Parameters	**lAxis:** As number 1, 2, 3, 4 corresponding to X, Y, Z, A axes **lSi:** New speed index for the HDI axis
Example	pTango->LSX_ SetHdiSpeedIndexSingleAxis (1, 2, 0); *// Set speed index of Y to 0*

4.8. BPZ Console with Trackball and Joyspeed Keys

GetBPZ	
Description	Retrieves status of a custom-built control console with trackball (?bpz).
C++	int LSX_GetBPZ (int lLSID, int *plAValue);
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	pTango->GetBPZ(1, &AValue);

SetBPZ	
Description	Switches custom-built control console on / off (!bpz).
C++	int LSX_SetBPZ (int lLSID, int lAValue);
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	pTango->SetBPZ(1, 1);

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

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

GetBPZTrackballBackLash

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

SetBPZTrackballBackLash

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

GetBPZTrackballFactor

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

SetBPZTrackballFactor

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

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++	int LSX_GetAutoLimitAfterCalibRM (int lLSID, int *plFlags);
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	pTango->GetAutoLimitAfterCalibRM(1, &Flags);

SetAutoLimitAfterCalibRM	
Description	Prevents setting of internal software limits when calibrating or measuring travel range (!nosetlimit).
C++	int LSX_SetAutoLimitAfterCalibRM (int lLSID, int lFlags);
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	pTango->SetAutoLimitAfterCalibRM(1, Flags);

GetLimit	
Description	Provides soft travel range limits, as set by SetLimit or cal+rm (?lim).
C++	int LSX_GetLimit (int lLSID, int lAxis, double *pdMinRange, double *pdMaxRange);
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	pTango->GetLimit(1, &MinRange, &MaxRange);

SetLimit	
Description	Set soft travel range limits (!lim).
C++	int LSX_SetLimit (int lLSID, int lAxis, double dMinRange, double dMaxRange);
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	pTango->SetLimit(1, 1, -10.0, 20.0); *// assign X-Axis –10 as lower and 20 as upper travel range limits*

GetLimitControl	
Description	Retrieves, whether area control (limits) is enabled or ignored (?limctr).
C++	int LSX_GetLimitControl (int lLSID, int lAxis, BOOL *pbActive);
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	pTango->GetLimitControl(1, 3, &Active); *// Read limit control enable state of Z-Axis*

SetLimitControl	
Description	Switches area control on / off (!limctr).
C++	int LSX_SetLimitControl (int lLSID, int lAxis, BOOL bActive);
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	pTango->SetLimitControl(1, 2, TRUE); *// Enable area control of Y-Axis is active*

GetSwitchActive

Description	Provides, whether hardware limit switches are enabled (?swact).
C++	int LSX_GetSwitchActive (int lLSID, int *plXA, int *plYA, int *plZA, int *plAA);
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	pTango->GetSwitchActive(1, &XA, &YA, &ZA, &AA);

SetSwitchActive

Description	Switches limit switches on / off (!swact).
C++	int LSX_SetSwitchActive (int lLSID, int lXA, int lYA, int lZA, int lAA);
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	pTango->SetSwitchActive(1, 7, 1, 5, 0); *// X-Axis: All limit switches enabled, Y-Axis: Only Zero limit switch enabled,* *// Z-Axis: E0 and EE switches enabled (default,) A-Axis: All limit switches ignored*

GetSwitchPolarity

Description	Retrieves polarity of limit switches (?swpol).
C++	int LSX_GetSwitchPolarity (int lLSID, int *plXP, int *plYP, int *plZP, int *plAP);
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	pTango->GetSwitchPolarity(1, &XP, &YP, &ZP, &AP);

SetSwitchPolarity

Description	Sets polarity of limit switches (!swpol).
C++	int LSX_SetSwitchPolarity (int lLSID, int lXP, int lYP, int lZP, int lAP);
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	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*

GetSwitchType

Description	Retrieves type of limit switches (?swtyp).
C++	int LSX_GetSwitchType (int lLSID, int *plXP, int *plYP, int *plZP, int *plAP);
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 for PNP type limit switch (default).
Example	pTango->GetSwitchType(1, &XP, &YP, &ZP, &RP);

SetSwitchType

Description	Sets type of limit switches (!swtyp).
C++	int LSX_SetSwitchType (int lLSID, int lXP, int lYP, int lZP, int lAP);
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 for PNP type limit switch with pull down resistor (default).
Example	pTango->SetSwitchType(1, XP, YP, ZP, AP);

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 lLSID, 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: 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++	int LSX_GetAnalogInput (int lLSID, int lIndex, int *plValue);
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	pTango->GetAnalogInput(1, 0, &Input); // Read channel 0

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

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++	int LSX_SetLedBright (int lLSID, double dBright);
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	pTango->SetLedBright(1, -1); *// set led off* pTango->SetLedBright(1, 0); *// set led to lowest possible brightness* pTango->SetLedBright(1, 12.345); *// set led to 12.345% brightness* pTango->SetLedBright(1, 100); *// set led to max. brightness*

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

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

SetAuxDigitalOutput	
Description	Set the specified digital output pin of the AUX-I/O port (!adigout).
C++	int LSX_SetAuxDigitalOutput (int lLSID, int lIndex, BOOL bValue);
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	pTango->SetAuxDigitalOutput(1, 0, TRUE); *// set output 0 to high*

GetAuxDigitalInput

Description	Get state of the specified digital input of the AUX-I/O port (?adigin).
C++	int LSX_GetAuxDigitalInput (int lLSID, int lIndex, BOOL *bValue);
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	pTango->GetAuxDigitalInput(1, 3, &bState); *// get input 3 state*

GetDigitalInputs

Description	Retrieve signal level of all 24 “I/O1 extension” digital input pins (?digin).
C++	int LSX_GetDigitalInputs (int lLSID, int *plValue);
Parameters	***Value***: Pointer to Integer value, to which the status of all inputs is written (as bit mask). LSB = Digital input 0

Example	int inputs;
	pTango->GetDigitalInputs(1, &inputs);
	if (Inputs & 16) ... *// if input 4 is set ...*

GetDigitalInputsE

Description	Retrieve signal level of all 12 “IO2 / Multi-IO” digital inputs (?edigin).
C++	int LSX_GetDigitalInputsE (int lLSID, int *plValue);
Parameters	***Value***: Pointer on a 32-Bit Integer, which returns the inputs 16...31 in the bits 0...15

Example	int ext_inputs;
	pTango->GetDigitalInputsE(1, &ext_inputs);

SetDigitalOutput

Description	Set individual digital output pin of IO1 extension (!digout).
C++	int LSX_SetDigitalOutput (int lLSID, int lIndex, BOOL bValue);

Parameters	***Index***: 0 to 7
	Value: Set pin level to
	FALSE = low
	TRUE = high

Example	pTango->SetDigitalOutput(1, 2, TRUE); *// set output pin 2 to´1´*

SetDigitalOutputs

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

SetDigitalOutputE

Description	Set individual digital output pin of Multi I/O / IO2 port (!edigout).
C++	int LSX_SetDigitalOutputE (int lLSID, int lIndex, BOOL bValue);

Parameters	***Index***: 0 to 7
	Value: Set pin level to
	FALSE = low
	TRUE = high

Example	pTango->SetDigitalOutputE(1, 2, TRUE); *// set output pin 2 to´1´*

SetDigitalOutputsE

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

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	***Index***: 0 to 15 (individual I/O pin), 16 (all 16 port pins) ***High*** = TRUE → high active ***High*** = 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++	int LSX_GetTVRMode (int lLSID, int *plValue);
Parameters	***Value***: Pointer to a 32-Bit Integer, which returns the TVR mode

Example	int mode;
	pTango->GetTVRMode (1, &mode);

SetTVRMode	
Description	Set the TVR mode of the clock & direction IO (!tvr)
C++	int LSX_SetTVRMode (int lLSID, int lMode1, int lMode2, int lMode3, int lMode4);

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	pTango->SetTVRMode(1, 0, 0, 5, 0); *// only set tvr mode 5 for Z*

GetFactorTVR	
Description	Retrieve the TVR factor for TVR modes 1-4 (?tvrf)
C++	int LSX_GetFactorTVR (int lLSID, double *pdVal1, double *pdVal2, double *pdVal3, double *pdVal4);
Parameters	***Val1…3***: Pointers to double, which hold the returned TVR factors

Example	double factor[4];
	pTango->GetFactorTVR (1, &factor[0] , &factor[1] , &factor[2] , &factor[3]);

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

4.12. Encoder Settings

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

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

GetEncoderActive	
Description	Retrieves which encoder will be activated after calibration (?encmask). Please note: This function is corresponding to the „?encmask“ command! Not “?enc”.
C++	int LSX_GetEncoderActive (int lLSID, int *plFlags);
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	pTango->GetEncoderActive(1, &Flags);

SetEncoderActive

Description	Sets which encoder is activated after calibration (!encmask) Please note: This function is corresponding to „!encmask“ command, not “!enc”.
C++	int LSX_SetEncoderActive (int lLSID, int lFlags);
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	pTango->SetEncoderActive(1, 0); *// No encoder will be used* pTango->SetEncoderActive(1, 2); *// encoder of Y-Axis will be activated after calibration*

GetEncoderMask

Description	Retrieve status of encoders (?enc). Please note: This function is corresponding to „?enc“ command, not “?encmask”!
C++	LSX_GetEncoderMask (int lLSID, int *plFlags);
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	int EncMask;
	pTango->GetEncoderMask(1, &EncMask);
	if (EncMask & 2) ...
	// if encoder of Y-Axis connected + active ...

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++	int LSX_SetEncoderMask (int lLSID, int lValue);
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	pTango->SetEncoderMask(1, 0); *// deactivate all encoders* pTango->SetEncoderMask(1, 2); *// deactivate X and Z encoders, activate Y-Axis encoder*

GetEncoderSingleAxis

Description	Read the encoder settings of Encoder Type, Signal Period and Reference Signal of the specified axis (?enctype, ?encref, ?encperiod).
C++	int LSX_GetEncoderSingleAxis (int lLSID, int *plAxis, double *pdPeriod, int *plReference);
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	pTango->GetEncoderSingleAxis(1, 3, &lEncType, &dPeriod, &lReference); *// axis 3(Z)*

SetEncoderSingleAxis

Description	Set the Encoder Type, Signal Period and availability of Reference Signal of the specified axis (!enctype, !encref, !encperiod).
C++	int LSX_SetEncoderSingleAxis (int lLSID, int lAxis, double dPeriod, int lReference);
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	pTango->SetEncoderSingleAxis(1, 3, 2, 0.02, 0); *// axis 3(Z) = Type 2, 20µm, no ref*

GetEncoderPeriod

Description	Retrieves encoder signal period length (?encperiod). A NULL pointer can be used for not required axes.
C++	int LSX_GetEncoderPeriod (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Period length [mm]

Example	double X,Y,Z,A;
	pTango->GetEncoderPeriod(1, &X, &Y, &Z, &A);
	pTango->GetEncoderPeriod(1, &X, &Y, NULL, NULL); // Ignore Z+A axes

SetEncoderPeriod

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

GetEncoderPosition

Description	Retrieves position response type (?encpos).
C++	int LSX_GetEncoderPosition (int lLSID, BOOL *pbValue);
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	pTango->GetEncoderPosition(1, &Value);

SetEncoderPosition

Description	Select readout of encoder- or motor-related position values (!encpos).
C++	int LSX_SetEncoderPosition (int lLSID, BOOL bValue);
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	pTango->SetEncoderPosition(1, TRUE);

GetEncoderRefSignal

Description	Retrieves whether the encoder reference signal is used when calibrating (?encref).
C++	int LSX_GetEncoderRefSignal (int lLSID, int *plXR, int *plYR, int *plZR, int *plAR);
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	pTango->GetEncoderRefSignal(1, &X, &Y, &Z, &A);

SetEncoderRefSignal

Description	Use reference signal from encoder when calibrating (!encref).
C++	int LSX_SetEncoderRefSignal (int lLSID, int lXR, int lYR, int lZR, int lAR);
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	pTango->SetEncoderRefSignal(1, 1, 1, 0, 0); *// when calibrating, reference signals of encoders X and Y are evaluated*

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++	int LSX_GetRefSpeed (int lLSID, int *plSpeed);
Parameters	Pointer to int for returning the velocity value (in 1/100 revolutions/s, one for all axes)
Example	pTango->GetRefSpeed (1, &Speed);

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++	int LSX_SetRefSpeed (int lLSID, int lSpeed);
Parameters	Velocity in 1/100 motor revolutions/s, one velocity applies to all axes
Example	pTango->SetRefSpeed (1, 50); *// search the reference switch (or the reference mark) search velocity of all axes to 50/100 motor revolutions/s (=0.5 rev/s)*

4.13. Closed Loop Settings

GetController	
Description	Retrieve Closed Loop mode (?ctr).
C++	int LSX_GetController (int lLSID, int *plXC, int *plYC, int *plZC, int *plRC);
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	pTango->GetController(1, &X, &Y, &Z, &A);

SetController	
Description	Set Closed Loop mode (!ctr).
C++	int LSX_SetController (int lLSID, int lXC, int lYC, int lZC, int lAC);
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	pTango->SetController(1, 2, 2, 0, 0); *// Enable permanent closed loop for X and Y axes*

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

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++	int LSX_SetControllerCall (int lLSID, int lCtrCall);
Parameters	***CtrCall***: Controller call time [ms]
Example	pTango->SetControllerCall(1, 5); *// CtrCall = 5 means: Closed Loop controller is called every 5 milliseconds*

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++	int LSX_GetControllerFactor (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Closed Loop factors
Example	pTango->GetControllerFactor(1, &X, &Y, &Z, &A);

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++	int LSX_SetControllerFactor (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: Position difference amplification factor 1 – 64
Example	pTango->SetControllerFactor(1, 2, 2, 2, 0); *//Closed Loop amplification is set to 2 for X, Y and Z axes*

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

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

GetControllerSteps

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

SetControllerSteps

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

GetControllerTimeout

Description	Read the closed loop control timeout (?ctrt). The maximum wait for TWI timeout.
C++	int LSX_GetControllerTimeout (int lLSID, int *plACtrTimeout);
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	pTango->GetControllerTimeout(1, &ActrTimeout);

SetControllerTimeout

Description	Set the closed loop control timeout (!ctrt). The maximum wait for TWI timeout.
C++	int LSX_SetControllerTimeout (int lLSID, int lACtrTimeout);
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	pTango->SetControllerTimeout(1, 500); *// Abort after trying to settle in the target window for 500ms*

GetControllerTWDelaySingleAxis

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

SetControllerTWDelaySingleAxis

Description	Set controller delay (!ctrd). Time to remain in TWI until "reached" state is assigned. For individual axes.
C++	int LSX_SetControllerTWDelay (int lLSID, int lAxis, int lCtrTWDelay);
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	pTango->SetControllerTWDelaySingleAxis(1, 3, 0); *// controller delay in Z switched off, closed loop end position will be inaccurate*

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++	int LSX_GetControllerTWDelay (int lLSID, int *plCtrTWDelay);
Parameters	***CtrTWDelay***: Controller delay [ms]
Example	pTango->GetControllerTWDelay(1, &CtrTWDelay);

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++	int LSX_SetControllerTWDelay (int lLSID, int lCtrTWDelay);
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	pTango->SetControllerTWDelay(1, 0); *// controller delay switched off, closed loop end position will be inaccurate*

GetTargetWindow

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

SetTargetWindow

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

SetCtrFastMoveOff

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

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++	int LSX_SetCtrFastMoveOn (int lLSID);
Parameters	-
Example	pTango->SetCtrFastMoveOn(1);

GetCtrFastMove

Description	Not supported by TANGO. Retrieves setting of FastMove function (?ctrfm).
C++	int LSX_GetCtrFastMove (int lLSID, 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 lLSID, 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++	int LSX_ClearCtrFastMoveCounter (int lLSID);
Parameters	-
Example	pTango->ClearCtrFastMoveCounter(1);

4.14. Trigger Output

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

SetTrigger	
Description	Switch trigger on / off (!trig).
C++	int LSX_SetTrigger (int lLSID, BOOL bATrigger);
Parameters	***Atrigger*** = TRUE → switch trigger on = FALSE → switch trigger off
Example	pTango->SetTrigger(1, TRUE);

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

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

GetTriggerPar

Description	Retrieves trigger parameters (?triga / ?trigm / ?trigs / ?trigd).
C++	int LSX_GetTriggerPar (int lLSID, int *plAxis, int *plMode, int *plSignal, double *pdDistance);
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	pTango->GetTriggerPar(1, &Axis, &Mode, & Signal, &Distance);

SetTriggerPar

Description	Set trigger parameters (!triga / !trigm / !trigs / !trigd).
C++	int LSX_SetTriggerPar (int lLSID, int lAxis, int lMode, int lSignal, double dDistance);
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	pTango->SetTriggerPar(1, 1, 3, 2, 5.0);

GetTrigCount

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

SetTrigCount

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

GetTriggerAxis

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

SetTriggerAxis

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

GetTriggerSignalLength

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

SetTriggerSignalLength

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

GetTriggerDistance

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

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

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++	int LSX_GetTriggerCompensation (int lLSID, int *plTime);
Parameters	***Time***: -10000 ... +10000 [µs] in increments of 10µs (0,10,20,…)
Example	pTango->GetTriggerCompensation (1, &Time);

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++	int LSX_SetTriggerCompensation (int lLSID, int lTime);
Parameters	***Time***: -10000 ... +10000 [µs] in increments of 10µs (0,10,20,…)
Example	pTango->SetTriggerCompensation (1, 40); *// Trigger compensation looks 40µs ahead*

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

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

GetTriggerFrequency

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

SetTriggerFrequency

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

GetTriggerOutput

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

SetTriggerOutput

Description	Select trigger output setting (!trigo).
C++	int LSX_SetTriggerOutput (int lLSID, 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 2nd output related to the trigger output (?trigbdelay).
C++	int LSX_Get2ndTriggerDelay (int lLSID, double *pdValue);
Parameters	***Value***: 0.00 to 32500000 [µs], internal resolution is 1/132µs
Example	pTango->Get2ndTriggerDelay (1, &Value);

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

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

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

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

Set2ndTriggerFrequency

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

GetTriggerRange

Description	Read back the trigger range settings which were set by LSX_SetTriggerRange (?trigr).

C++	`int LSX_GetTriggerRange (int lLSID,`
	`int *plAxis,`
	`double *pdStartPos,`
	`double *pdEndPos,`
	`int *plNumberOfTriggerPulses);`

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	pTango->GetTriggerRange (1, &Axis, &StartPos,&EndPos,&NumberOfTriggerPulses);

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.

C++	`int LSX_SetTriggerRange (int lLSID,`
	`int lAxis,`
	`double dStartPos,`
	`double dEndPos,`
	`int lNumberOfTriggerPulses);`

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	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*

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

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++	int LSX_SetTriggerPositionList (int lLSID, int lIndex, double dPosition);
Parameters	***Index***: 1…MAX_ENTRIES ***Pos***: Position list entry for the selected trigger axis (depends on Dimension)
Example	pTango->SetTriggerPositionList (1, 1, 10.5); *// Set first pulse at 10.5mm* pTango->SetTriggerPositionList (1, 2, 17.5031); *// Set 2nd pulse at* 17.5031*mm* pTango->SetTriggerPositionList (1, 3, 51.8774); *// Set 3rd pulse at* 51.8774*mm*

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++	int LSX_GetTriggerPositionListIndex (int lLSID, int *plIndex);
Parameters	***Index***: 1...N
Example	pTango->GetTriggerPositionListIndex (1, &Index);

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++	int LSX_SetTriggerPositionListIndex (int lLSID, int lIndex);
Parameters	***Index***: 1...N
Example	pTango->SetTriggerPositionListIndex (1, 17); *// Set the next trigger position at list entry number 17 (to skip the trigger positions in between)*

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++	int LSX_GetTriggerPositionListEntries (int lLSID, int *plNumberOfEntries);
Parameters	***NumberOfEntries***: 0...N
Example	pTango->GetTriggerPositionListEntries (1, &NumberOfEntries);

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++	int LSX_SetTriggerPositionListEntries (int lLSID, int lNumberOfEntries);
Parameters	***NumberOfEntries***: 0, 1...N
Example	pTango->SetTriggerPositionListEntries (1, 5); *// Reduce the number of entries to 5* pTango->SetTriggerPositionListEntries (1, 0); *// Delete the entire trigger position list*

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++	int LSX_GetTriggerLevel (int lLSID, int *plLevel);
Parameters	***Level***: 0 = active low, 1 = active high trigger pulse
Example	pTango->GetTriggerLevel (1, &Level);

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++	int LSX_SetTriggerLevel (int lLSID, int lLevel);
Parameters	***Level***: 0 = active low, 1 = active high trigger pulse
Example	pTango->SetTriggerLevel (1, 0); *// Set trigger to active low (0)*

4.15. Snapshot Input

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

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

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

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

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

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

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

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++	int LSX_SetSnapshotFilter (int lLSID, int lTime);
Parameters	***Time***: Filter time, within 0-100 ms
Example	pTango->SetSnapshotFilter(1, 0); *// no filter, fast response (e.g. for TTL signals)*

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++	int LSX_GetSnapshotPar (int lLSID, BOOL *pbHigh, BOOL *pbAutoMode);
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	pTango->GetSnapshotPar(1, &High, &AutoMode);

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++	int LSX_SetSnapshotPar (int lLSID, BOOL bHigh, BOOL bAutoMode);
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	pTango->SetSnapshotPar(1, TRUE, FALSE);

GetSnapshotPos

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

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

SetSnapshotPosArray	
Description	Set, append or change entries of the position array (!snsa).
C++	int LSX_SetSnapshotPosArray (int lLSID, int lIndex, double dX, double dY, double dZ, double dA);
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	pTango->SetSnapshotPosArray(1, -1, 0.55, 2.4, 0.0, 0.0); *// Append a position array entry by software*

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

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++	int LSX_GetSnapshotIndex (int lLSID, int *plSnsIndex);
Parameters	***SnsIndex***: Current position of the index pointer within the position array
Example	pTango->GetSnapshotIndex(1, &SnsIndex);

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++	int LSX_SetSnapshotIndex (int lLSID, int lSnsIndex);
Parameters	***SnsIndex***: Required position of the index pointer within the PosArray, e.g. for SnapshotMode “Automatic”
Example	pTango->SetSnapshotIndex(1, 5); // Set pointer to Index 5

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

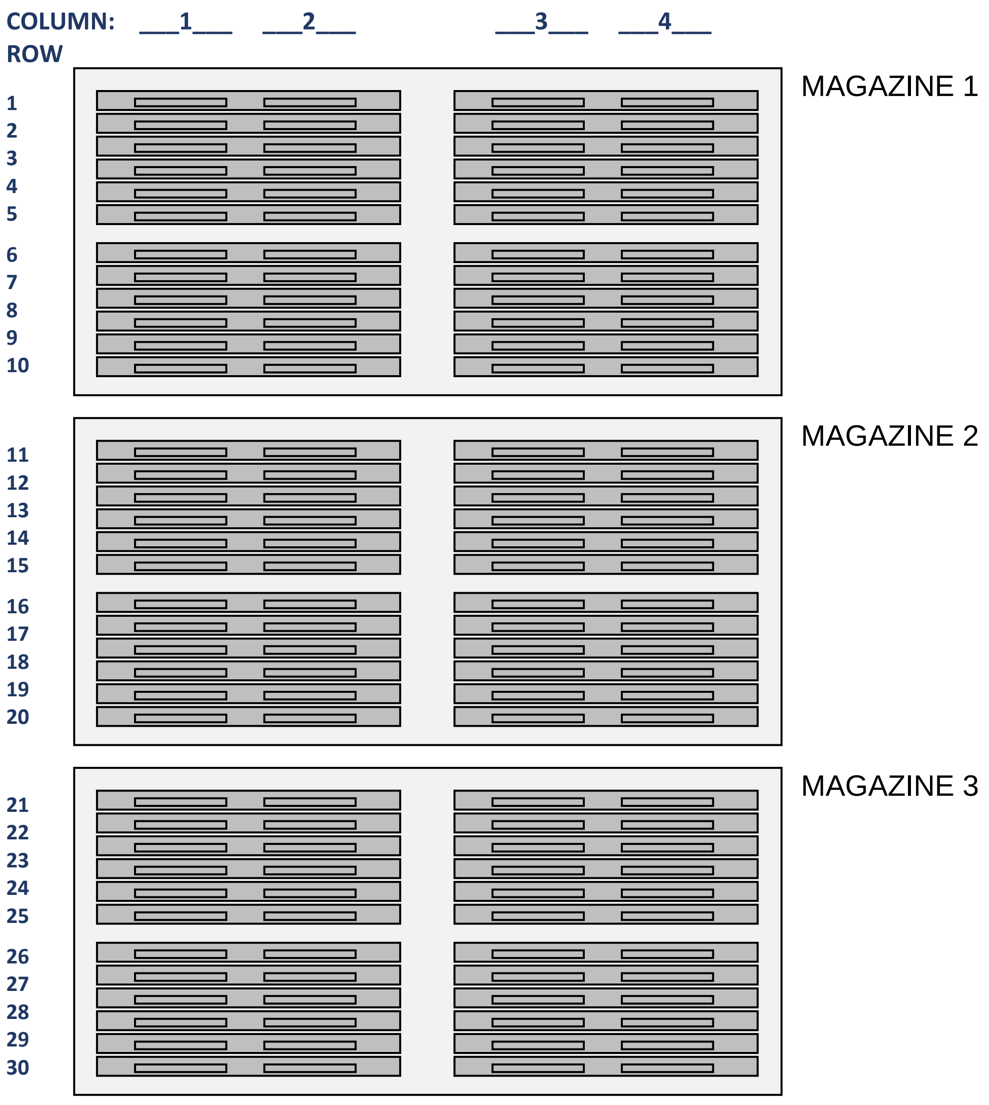
SlideExpress 2: Organization of slides in rows and columns

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:



Eject

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

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++	int LSX_Insert (int lLSID);
Parameters	-
Example	pTango->Insert(1);

SlideSeated

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

MagazinSeated

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

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++	int LSX_GetGripper (int lLSID, int *c1, int *r1, int *c2, int *r2);
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 owerwrite 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++	int LSX_SetGripper (int lLSID, int c1, int r1, int c2, int r2);
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++	int LSX_GetClipType (int lLSID, int *plClipType);
Parameters	plClipType → pointer to int returns clip type
Example	pTango->GetClipType (1, plClipType);

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

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++	int LSX_PutSlide (int lLSID, int col, int row);
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	pTango->PutSlide (1, 4, 20); // put slide to magazine 4 slot 20.
	pTango->PutSlide (1, 0, 0); // put slide back to where it was taken from.

Obsolete:

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

Obsolete:

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

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++	int LSX_Insert (int lLSID);
Parameters	-
Example	pTango->Insert(1);

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

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

GetGripper	
Description	Query gripper status information (?gripper). Returns status of gripper.
C++	int LSX_GetGripper (int lLSID, int *c1, int *s1, int *c2, int *s2);
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++	int LSX_SetGripper (int lLSID, int c1, int s1, int c2, int s2);
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++	int LSX_GetTray (int lLSID, int slot, int mode);
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++	int LSX_PutTray (int lLSID, int slot);
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++	int LSX_GetRFID (int lLSID, int slot, int bank, int *plRFID);
Parameters	slot → slot number [1..MAXSLOT] bank → bank number [0 to 64] plRFID → pointer to int returns data stored in RFID transponder device
Example	pTango->GetRFID (1, 1, 0, plRFID);

SetRFID

Description	Store RFID data into addressed tray (!rfid)
C++	int LSX_SetRFID (int lLSID, int slot, int bank, int rfdata);
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	pTango->SetRFID (1, 1, 0, rfdata);

GetNumberOfSlots

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

GetNumberOfMagazines

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

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++	int LSX_GetLoaderType (int lLSID, int *plLoaderType);
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	pTango->GetLoaderType (1, plLoaderType);

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++	int LSX_GetNumberOfRows (int lLSID, int *plRows);
Parameters	plRows → pointer to int returns number of available slots or magazine rows
Example	pTango->GetNumberOfRows (1, plRows);

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++	int LSX_GetNumberOfColumns (int lLSID, int *plCols);
Parameters	plCols → pointer to int returns number of magazine column (6 manual, 6 for loader system)
Example	pTango->GetNumberOfColumns (1, plCols);

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

GetTrayType

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

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++	int LSX_SetTrayType (int lLSID, int slot, int aTrayType);
Parameters	aTrayType → int data contains information of required tray type

Example	int aTrayType = 0x0100010a; //see customer specification requirements for explanation
	pTango->SetTrayType (1, 1, aTrayType);

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

Example	pTango->SetCabinLED (1, 1); // switch ON illumination
	pTango->SetCabinLED (1, 0); // switch OFF

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

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

Example	pTango->SetLabelLED (1, 1); // switch ON illumination
	pTango->SetLabelLED (1, 0); // switch OFF

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

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++	int LSX_Xpos3GetPosSingleAxis (int lLSID, int lAxis, double *plPos);
Parameters	lAxis → xPos axis 1, 2 or 3 lPos → variable to return the position to
Example	pTango->Xpos3GetPosSingleAxis(1, 2, &Pos); *// Read Position of the 2nd xPos axis*

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

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

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

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 134nitialized	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 chksum

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 desciption 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, GetBlSmoothSingleAxis, SetBlSmoothSingleAxis, 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)