

TANGO-DLL

Documentation



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

Table of Contents

1.	Introduction	4
1.1.	Functional Range	4
1.2.	System Requirements	4
1.3.	Supported Development Environments	4
2.	DLL-Interface	5
2.1.	General Information	5
2.2.	Integration in Visual C++	6
2.3.	Integration in Visual Basic	6
2.4.	Integration in LabVIEW	7
3.	General Information of DLL Usage	10
3.1.	Initialization of Controller	11
3.2.	Own Program Section	13
3.3.	API State Diagram	14
4.	Functions	15
4.1.	Quick Reference	15
4.2.	DLL Configuration / Interface	22
4.3.	Controller Information	27
4.4.	Status Requests	29
4.5.	Settings	32
4.6.	Move Commands and Positioning Management	45
4.7.	Joystick and Handwheel	53
4.8.	Control Console with Trackball and Joyspeed Keys	59
4.9.	Limit Switches (Hardware and Software)	62
4.10.	Digital and Analog Inputs and Outputs	67
4.11.	Encoder Settings	72
4.12.	Closed Loop Settings	77
4.13.	Trigger Output	82
4.14.	Snapshot Input	84
5.	SlideExpress Interface	88
5.1.	Eject	88
5.2.	Insert	88
5.3.	SlideSeated	88
5.4.	MagazinSeated	89
5.5.	GetGripper	89
5.6.	SetGripper	89
5.7.	GetSlide	89
5.8.	PutSlide	90
5.9.	GetPrioHandlerPos	90
5.10.	SetPrioHandlerPos	90
6.	TrayExpress Interface	91
6.1.	Eject	91
6.2.	Insert	91
6.3.	SlideSeated	91
6.4.	MagazinSeated	91
6.5.	GetGripper	92
6.6.	SetGripper	92
6.7.	GetTray	92
6.8.	PutTray	92



6.9.	GetRFID	93
6.10.	SetRFID	93
6.11.	GetNumberOfSlots	93
6.12.	GetNumberOfMagazines	93
7.	Express Interface Extensions	94
7.1.	GetLoaderType	94
7.2.	GetNumberOfRows	94
7.3.	GetNumberOfColumns	94
7.4.	GetTraySN	94
7.5.	GetTrayType	95
7.6.	SetTrayType	95
7.7.	SetCabinLED	95
7.8.	GetCabinLED	95
7.9.	SetLabelLED	96
7.10.	GetLabelLED	96
8.	Error Codes	97
8.1.	Tango Error Messages	97
8.2.	DLL Error Messages	100
9.	Document Revision History	101

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, or Virtual COM Port (USB, PCI and PCI-E)
- Supports most controller commands directly
- Up to 4 axes per TANGO
- Up to 8 TANGO controllers

1.2. System Requirements

The Tango-DLL can be used on all Windows PCs from Windows XP to Windows 10. It requires the *Microsoft Visual C++ 2010 Redistributable Package*, which often is already installed on Windows PCs. If not, it can be downloaded from the www.microsoft.com website.

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 XP, Windows 7, Windows 8 and Windows 10 using following development tools:

- Microsoft Visual Studio 2010 languages Visual Basic, C# and C++
- National Instruments LabVIEW
- Embarcadero Delphi 2007 and Delphi XE
- Java
- Compatibility is assumed for all other programming environments which are able to use 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, send commands, retrieve the status of inputs and outputs, etc.

2.1. General Information

All functions are declared with a 32-bit integer return value. A return value of 0 (zero) indicates the error free execution of a function. For other integer values refer chapter **Error Codes**. DLL function

LSX_GetErrorString can be used to translate these error code into short english text.

The examples provided in this documentation exclusively use „LSX_“ commands in which the first value stands for the TANGO ID (LSID). This ID is used to address a several controllers simultaneously. As the "LSX_" commands currently only support one controller, we recommend using the "LS_" commands. With this, the first value of the Tango-ID is not needed in function calls, neither is a CreateLSID required.

Example

„LS_“-Command:

```
pTango->MoveAbs(50.0, 50.0, 50.0, 10.0, TRUE);
```

„LSX_“-Command:

```
pTango->MoveAbs(1, 50.0, 50.0, 50.0, 10.0, TRUE);
```

// the first value is the LSID, which is not needed with „LS_“ commands

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

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

2.2. Integration in Visual C++

An enclosure of Tango_DLL.dll has been created for Visual C++. The class CTango 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).

Only one instance should be created of the class CTango, as with Tango-DLL, momentarily, it is not possible to operate several controllers at the same time.

The required files for your C/C++ Application Tango.h and Tango.cpp can be found on the CD in the directory Software\API\Examples\Visual_C\SourceCode.

Required files:

- Tango_DLL.dll,
- Tango.h and
- Tango.cpp

Visual C++ example for controlling a Tango:

```
...
pTango = new CTango();
...

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

2.3. Integration in Visual Basic

In order 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 in directory Software\API\Examples\Visual_Basic\SourceCode.

Required files: Tango_DLL.dll and Tango.vb

Visual Basic example for controlling a Tango:

```
Dim return value As Integer
Dim return value2 As Integer
Dim return value3 As Integer

...
Return value = LS_ConnectSimple(1, „COM3“, 57600, 1)
Return value2 = LS_MoveAbs(30, 50, 70, 0, 1)
Return value3 = LS_Disconnect
```

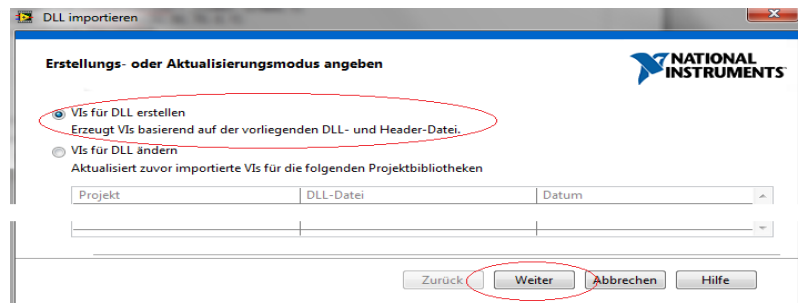
2.4. Integration in LabVIEW

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

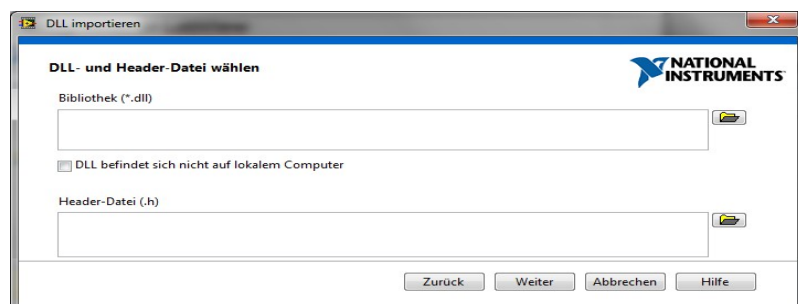
In order to use the functions of TANGO-DLL with LabVIEW, the TANGO-DLL has to be imported to LabVIEW.

Therefore follow the steps listed below:

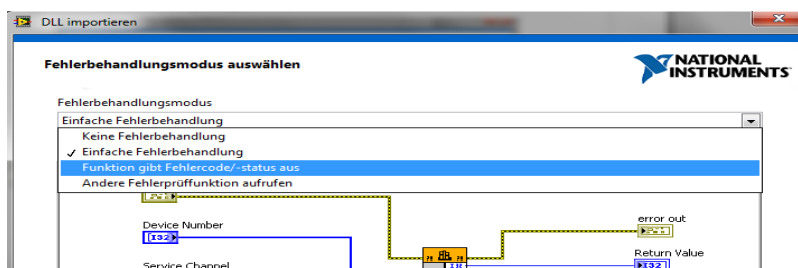
- 1) Start LabVIEW
- 2) In LabVIEW window: Tools → Import → DLL select the first radio button and press next.



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

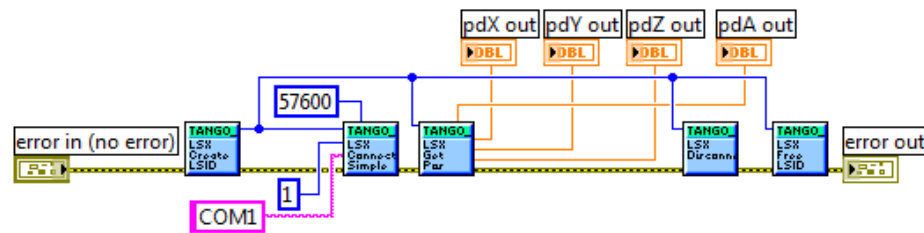


- 4) "Including Paths" in the next window need not to be configured.
- 5) In the next window the included functions of the TANGO_DLL.dll are listed and selectable. It is recommended to select all functions. You may notice, that only half of the functions included in TANGO_DLL.dll are found in the TANGOLSX_API.h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



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

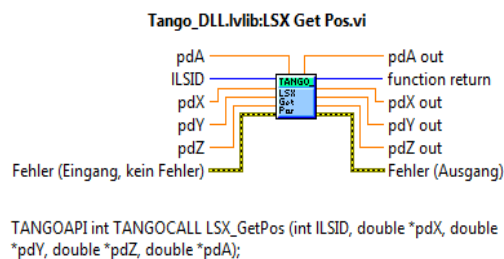
LabVIEW starting example for controlling a TANGO:



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

Remark:

“Get” functions defined in TANGO_DLL.dll often have pointer as parameters. These pointer 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.

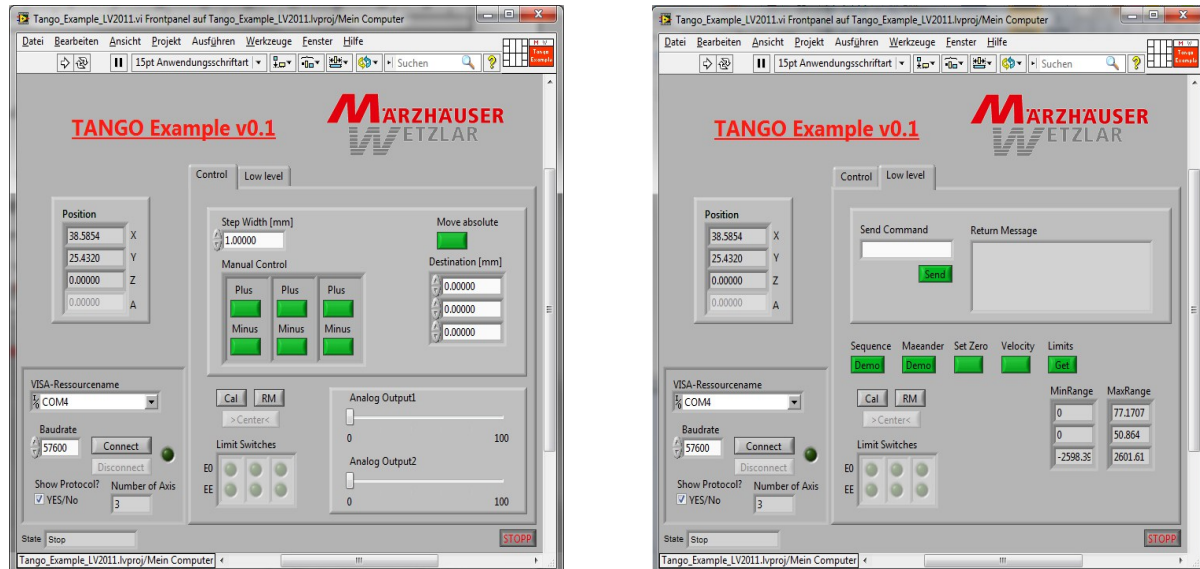


In all such "Get" VIs just connect only required output parameters. It is useless to connect input parameters because they will be ignored anyway and won't have any effect.

Program Example:

Required LabVIEW-Version: LabVIEW 2011 and newer

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



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

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

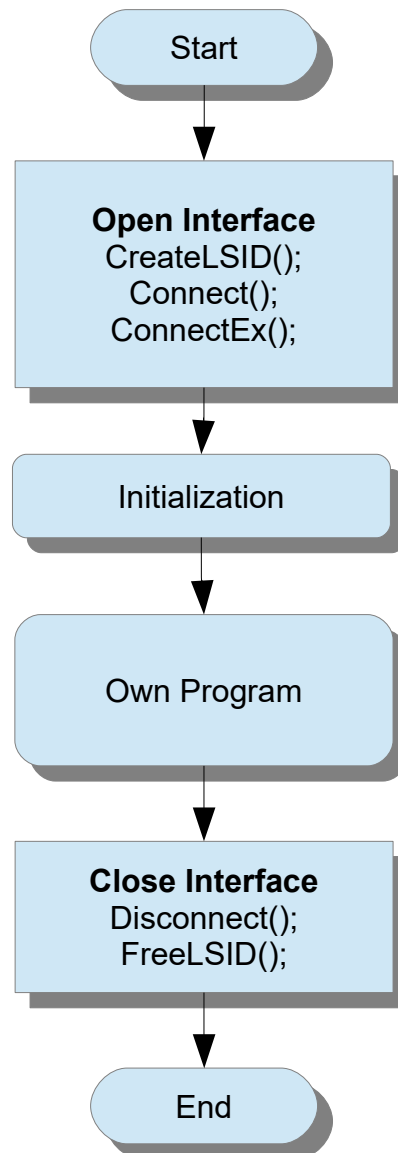
Functions included in TANGO example VI:

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

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 and PCI-E. All Tango application programs, independent of chosen and involved programming language, should follow this guideline.

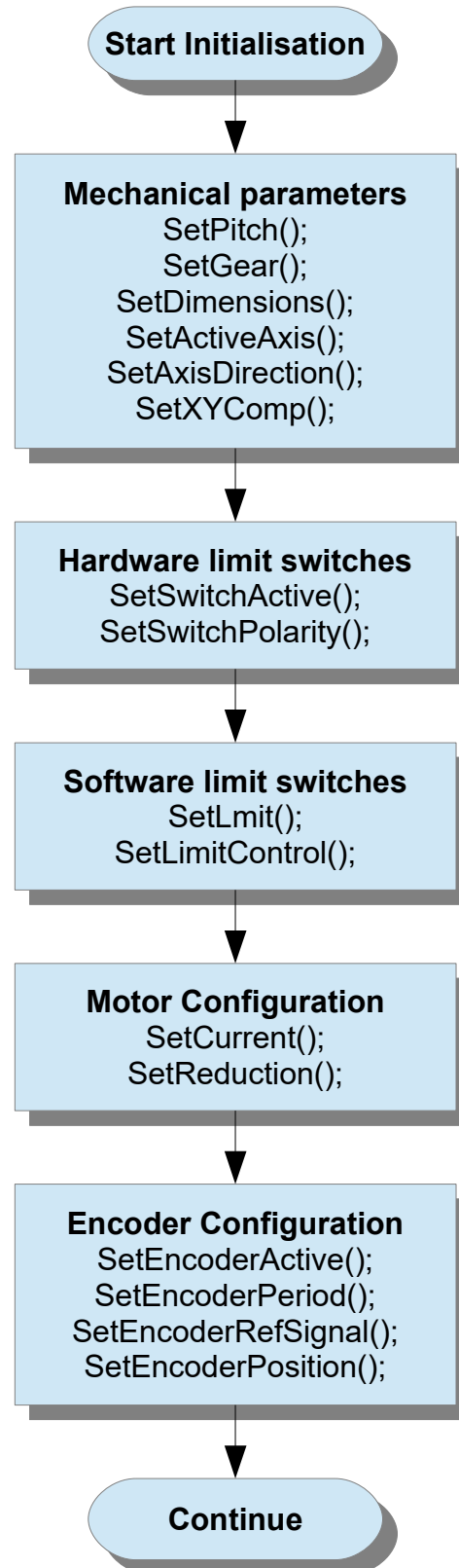
DLL functions are listed and described in detail in next chapters.

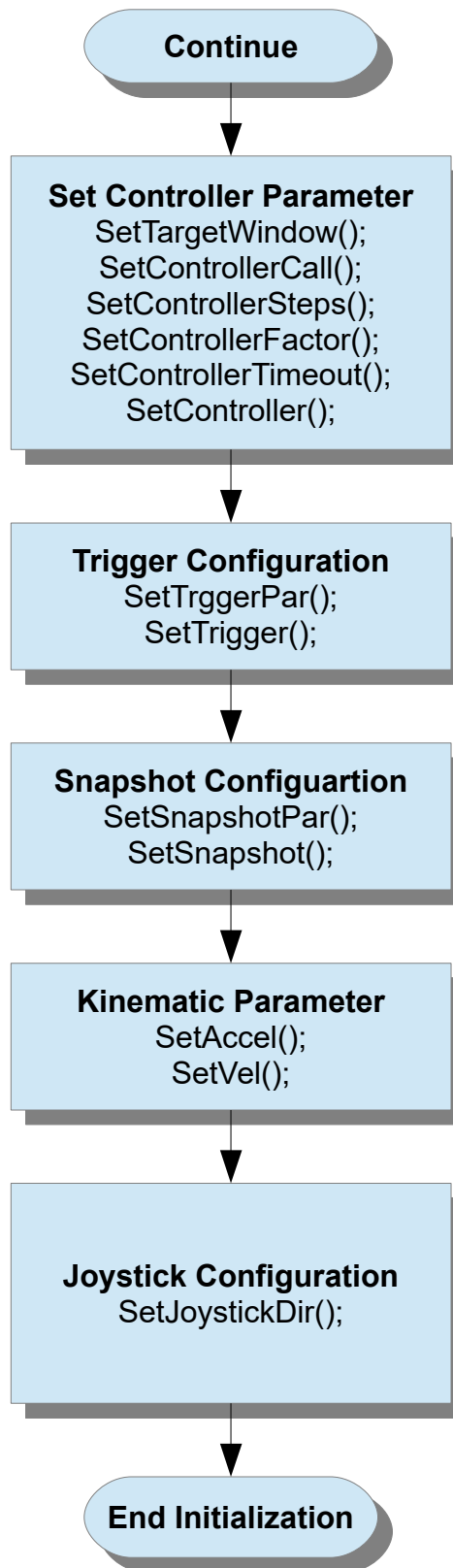


3.1. Initialization of Controller

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.

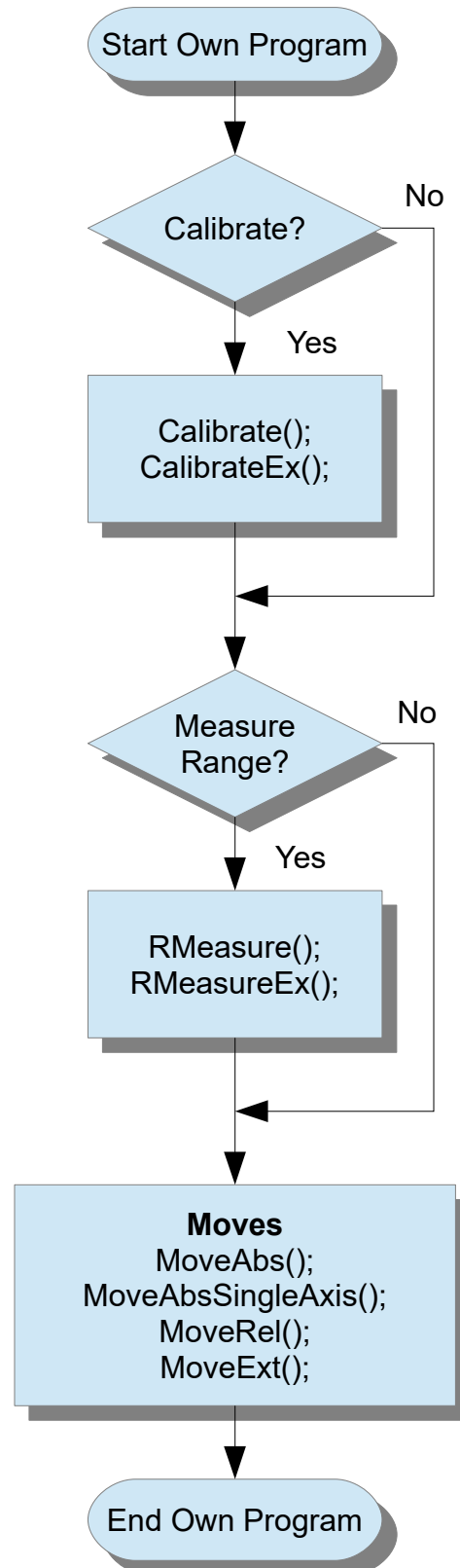
Note: Any mechanics may be damaged if wrong parameters are used. Please be careful to use correct stage data to prevent any damage. Follow below flow chart to transmit individual settings.





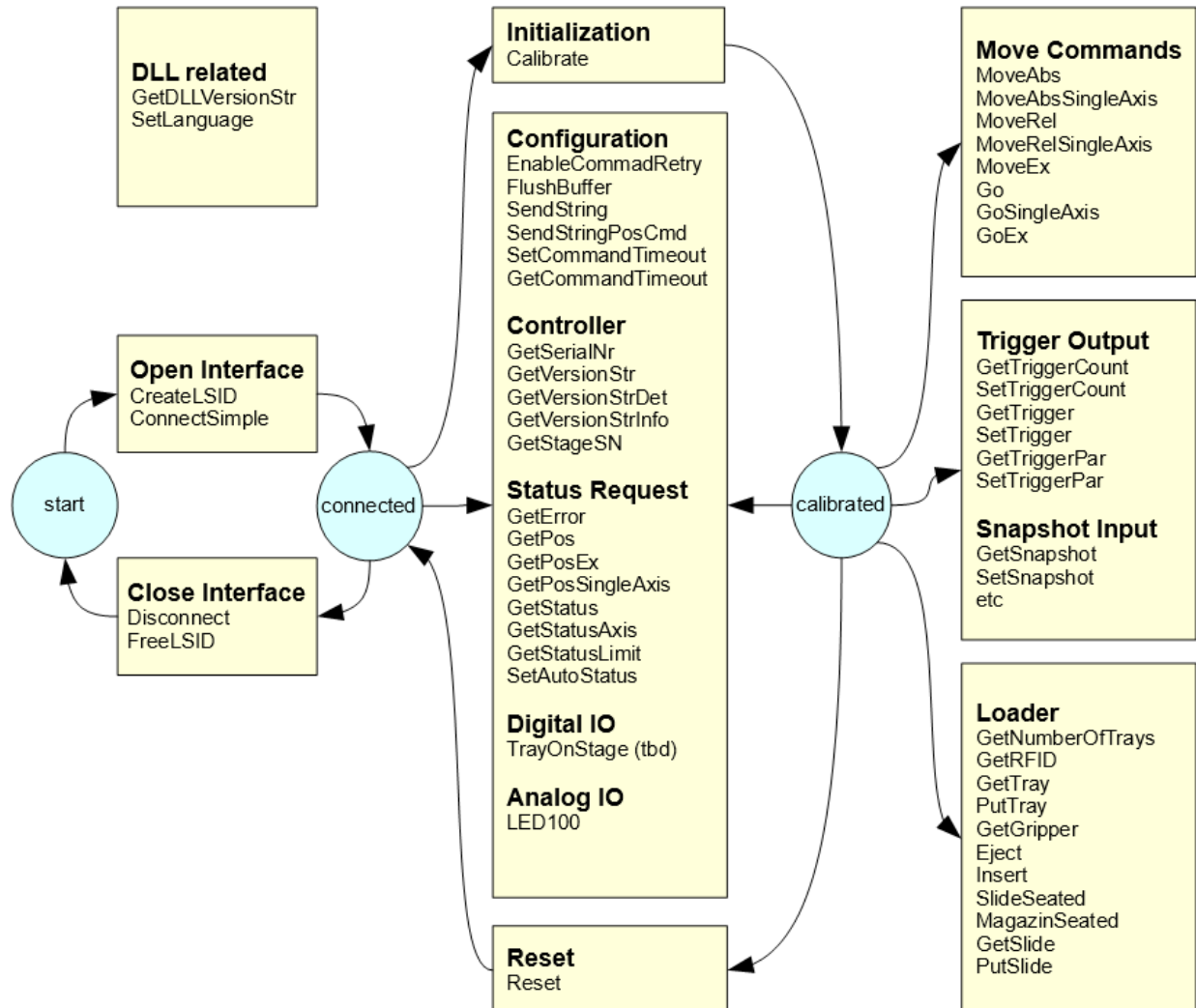
3.2. Own Program Section

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 a limit switch is taken as reference.



4. Functions

4.1. Quick Reference

DLL Configuration / Interface:

Command	Brief Description	Page
CreateLSID	Creates a Tango-ID number	22
ConnectSimple	Connect to Tango using default controller settings	22
Disconnect	Disconnects Tango Controller from DLL	22
EnableCommandRetry	This command enables switching on / off of repeated command sending in case of communication errors	23
FlushBuffer	Clears the receive buffer from possibly remaining data fragments	23
FreeLSID	Releases the previously created Tango ID-Number	23
SendString	Sends strings to Tango (enables using all commands as ASCII text)	24
SendStringPosCmd	Send an ASCII move command and wait for completion reply	24
SetAbortFlag	Set internal DLL flag to abort a (hanging) communication	25
SetShowProt	Switches communication monitoring on/off	25
GetCommandTimeout	read current DLL timeout for read, move and calibration	25
SetCommandTimeout	set DLL timeout for read, move an calibration	25
GetDLLVersion	read DLL version string	26
LoadConfig	Load configuration from ini file	26
Connect	Connect using data from LoadConfig ini file	28
SaveConfig	Save configuration to ini file	26
ReadControlPars	Read actual setup parameter from controller	
ConnectEx	Connect using data from ReadControlPars	
SetControlPars	Send setup parameter to controller	
SetLanguage	Set language of protocol window	26

Controller information:

Command	Brief Description	Page
GetSerialNr	Read out the Controller serial number	27
GetVersionStr	Provides current firmware version number	27
GetVersionStrDet	Reads detailed firmware version information	27
GetVersionStrInfo	Retrieves additional information to current version number	27
GetStageSN	Retrieves stage serial number (if available)	28

Status Requests:

Command	Brief Description	Page
GetError	Provides current error number	29
GetPos	Retrieves current position of all axes	29
GetPosEx	Retrieves values of current encoder- or motor-positions of all axes	29
GetPosSingleAxis	Retrieves current position of one axis	30
GetStatus	Provides current Controller status	30
GetStatusAxis	Provides current status of one axis	30
GetStatusLimit	Provides current status of software limits of all axes	31
SetAutoStatus	Switches Auto-Status reply on/off	31

Controller Settings:

Command	Brief Description	Page
GetAccel	Read actual acceleration	32
SetAccel	Set required acceleration	32
GetActiveAxes	Retrieve axes state	32
GetAccelFunction	Retrieve actual acceleration function	32
SetAccelFunction	Set acceleration function trapezoidal or sinusoidal	32
SetActiveAxes	Set axes state	33
GetAxisDirection	Retrieve axis direction	33
SetAxisDirection	Set axis direction	33
GetCalibBackSpeed	Retrieve calibration backward speed	33
SetCalibBackSpeed	Set calibration backward speed	34
GetCalibOffset	Retrieve calibration offset	34
SetCalibOffset	Set calibration offset	34
GetCalibrateDir	Retrieve calibration direction	34
SetCalibrateDir	Set calibration direction	35
GetCurrentDelay	Provides time delay for motor current reduction	35
SetCurrentDelay	Sets the time delay, after which the motor current is reduced	35
GetDimensions	Provides the applied measuring units of axes	35
SetDimensions	Set measuring units of axes	36
GetResolution	Get digits after decimal point	36
SetResolution	Set digits after decimal point	37
GetGear	Retrieves gear ratio	37
SetGear	Set gear ratio	37
GetMotorCurrent	Retrieves electrical motor current	38
SetMotorCurrent	Set electrical current of motor	38
GetMotorSteps	Retrieves number of motor steps	38
SetMotorSteps	Set number of motor steps	38
GetPitch	Read actual spindle pitch	38
SetPitch	Set required spindle pitch	39
GetPowerAmplifier	Retrieves actual state of power amplifier	39
SetPowerAmplifier	Set required state of power amplifier	39
GetReduction	Read actual current reduction	39
SetReduction	Set current reduction	40
GetRMOffset	Retrieve range measure offset	40
SetRMOffset	Set range measure offset	40
GetSpeedPoti	Retrieve speed potentiometer	41
SetSpeedPoti	Set speed potentiometer	41
GetStopAccel	Retrieve stop acceleration	41
SetStopAccel	Set stop acceleration	41
GetStopPolarity	Retrieve stop polarity	41
SetStopPolarity	Set stop polarity	42
GetVel	Retrieves actual max velocity	42
SetVel	Set required velocity	42
GetVelFac	Retrieves velocity factor	42
SetVelFac	Set velocity factor	43
LStepSave	save all actual parameter in controller	43
SetAccelSingleAxis	Set acceleration for a single axis	43
SetVelSingleAxis	Set velocity for a single axis	43
SoftwareReset	Reset and reboot the controller	43
IsVel	Read actual velocities at which the axis are currently travelling	44



IsVelSingleAxis	Read actual velocity of specified axis	44
-----------------	--	----

Move Commands and Position Management:

Command	Brief Description	Page
Calibrate	Calibrate enabled axes to the CAL limit switches	45
CalibrateEx	Calibrates single axes	45
ClearPos	Sets position values to zero	45
GetDelay	Provides delay of vector start	46
SetDelay	Causes delay of vector start	46
GetDistance	Provides distance started with MoveRelShort	46
SetDistance	Sets distance for MoveRelShort command	46
MoveAbs	Moves to absolute position of all axes	47
MoveAbsSingleAxis	Moves to absolute position of single axis	47
MoveEx	Extended move/move relative command with axis bit mask	48
MoveRel	Move by relative vector for all axes	48
MoveRelShort	Relative positioning (short command)	49
MoveRelSingleAxis	Move single axis relatively	49
RMeasure	Measure maximum travel range of all axes	49
RMeasureEx	Measure max. travel range of axes selected by the axis bit mask	50
SetPos	Set current position to the desired value	50
StopAxes	Stops all moving axes	50
WaitForAxisStop	Function returns as soon as all axes chosen in bit mask have reached their end position	51
Go	Move command designed to be used with mouse drag events	51
GoSingleAxis	Go for single axis	52
GoEx	Extended Go command	52

Joystick and Handwheel:

Command	Brief Description	Page
GetDigJoySpeed	Retrieves current digital joystick speed	53
SetDigJoySpeed	Start a move at constant speed (commanded digital joystick)	53
GetHandWheel	Retrieves handwheel status	53
GetJoystick	Retrieves analog joystick status	54
GetJoystickDir	Retrieves revolve direction of motor for joystick	54
SetJoystickDir	Sets analog joystick direction	55
GetJoystickWindow	Retrieves joystick window	55
SetJoystickWindow	Set analog joystick idle window	55
SetHandWheelOff	Switches handwheel off	55
SetHandWheelOn	Switches handwheel on	56
SetJoystickOff	Switches analog joystick off	56
SetJoystickOn	Switches analog joystick on	56
GetHwFactor	Retrieves handwheel factor	56
SetHwFactor	Set handwheel factor	57
GetHwFactorB	Retrieves second handwheel factor	57
SetHwFactorB	Set second handwheel factor	57
GetZwTravel	Retrieves z-wheel travel distances	57
SetZwTravel	Set z-wheel travel distances	57
GetKey	Retrieves key state	58
GetKeyLatch	Retrieves and clears latched key states	58
ClearKeyLatch	Clears latched key states	58

Control Console with Trackball and Joyspeed Keys (Customized Application):

Command	Brief Description	Page
GetBPZ	Retrieves status of control console	59
SetBPZ	Switches control console on / off	59
GetBPZJoyspeed	Retrieves control console joystick speed	59
SetBPZJoyspeed	Set control console joystick speed	60
GetBPZTrackballBackLash	Retrieves control console trackball backlash	60
SetBPZTrackballBackLash	Set control console trackball backlash	60
GetBPZTrackballFactor	Retrieves control console trackball factor	60
SetBPZTrackballFactor	Set control console trackball factor	61

Limit Switches (Hardware and Software):

Command	Brief Description	Page
GetAutoLimitAfterCalibRM	Provides, whether internal software limits are set when calibrating or measuring stage travel range	62
SetAutoLimitAfterCalibRM	Prevents setting internal software limits by calibration or range measure	62
GetLimit	Provides travel range limits of single axes	62
SetLimit	Sets travel range limits of single axes	63
GetLimitControl	Retrieves whether area control is switched on or off	63
SetLimitControl	Switches area control on / off	63
GetSwitchActive	Provides, whether limit switches are active	64
SetSwitchActive	Enable/disable limit switches	64
GetSwitches	Retrieves status of all limit switches	64
GetSwitchPolarity	Retrieves polarity of limit switches	65
SetSwitchPolarity	Sets polarity of limit switches	65
GetSwitchType	Retrieves status of pull up or pull down resistor array (NPN or PNP)	65
SetSwitchType	Set resistor pull-up or pull down to match NPN or PNP switches	66

Digital and Analog Inputs and Outputs:

Command	Brief Description	Page
GetAnalogInput	Retrieves current level of analogue input signals	67
GetDigitalInputs	Retrieve all digital input pin levels	67
GetDigitalInputsE	Retrieve additional digital inputs 16-31	67
SetAnalogOutput	Set analogue output voltage	67
SetDigIO_Distance	Activate an output, depending on set distance before or after reaching determined position	68
SetDigIO_EmergencyStop	Assign Emergency-Stop pin	68
SetDigIO_Off	Switch off digital I/O functionality	68
SetDigIO_Polarity	Set polarity	69
SetDigitalOutput	Set individual digital outputs of I/O1-Module	69
SetDigitalOutputs	Set digital outputs 0-7 of I/O1-Module	69
SetDigitalOutputsE	Set individual digital outputs of Multi-I/O-Module	69
SetAuxDigitalOutput	Set individual digital outputs of AUX-I/O connector	70
SetLedBright	Set the brightness of the LED100 illumination OFF/0-100%	71

Encoder Settings:

Command	Brief Description	Page
ClearEncoder	Set encoder position to zero	72
GetEncoder	Retrieves all encoder positions	72
GetEncoderActive	Retrieves which encoder is activated after calibration (<i>encmask</i>)	72
SetEncoderActive	Select encoder to be activated after calibration	73
GetEncoderMask	Retrieve status of encoders (" <i>enc</i> " command!)	73
SetEncoderMask	Activates / deactivates encoders	73
GetEncoderPeriod	Retrieves length of encoder signal period	74
SetEncoderPeriod	Set length of encoder period	75
GetEncoderPosition	Provides, whether encoder- or motor- position is displayed	75
SetEncoderPosition	Switches encoder value display on / off	75
GetEncoderRefSignal	Provides if reference signal from encoder shall be evaluated when calibrating	75
SetEncoderRefSignal	Evaluate encoder reference signal when calibrating.	76

Closed Loop Settings:

Command	Brief Description	Page
ClearCtrFastMoveCounter	Resets number of executed FastMove functions to 0	77
GetController	Retrieve controller mode	77
SetController	Set controller mode	77
GetControllerCall	Provides controller call interval	78
SetControllerCall	Set controller call time	78
GetControllerFactor	Retrieve setting of controller factor	78
SetControllerFactor	Set controller factor	78
GetControllerSteps	Retrieve controller steps	79
SetControllerSteps	Set controller steps	79
GetControllerTimeout	Retrieves setting of controller monitoring timeout	79
SetControllerTimeout	Set controller monitoring timeout	79
GetControllerTWDelay	Retrieve controller delay for target window	80
SetControllerTWDelay	Set controller delay	80
GetCtrFastMove	Retrieves whether FastMove function is switched on or off	80
GetCtrFastMoveCounter	Retrieves number of executed FastMove functions	80
GetTargetWindow	Retrieves target windows of all axes	81
SetTargetWindow	Set controller target windows	81
SetCtrFastMoveOff	Switch off FastMove function	81
SetCtrFastMoveOn	Switch on FastMove function	81

Trigger Output:

Command	Brief Description	Page
GetTrigCount	Retrieve trigger counter value	82
SetTrigCount	Set trigger counter value	82
GetTrigger	Retrieve trigger setting	82
SetTrigger	Switch trigger on / off	82
GetTriggerPar	Retrieve trigger parameters	83
SetTriggerPar	Set trigger parameters	83

Snapshot-Input:

Command	Brief Description	Page
GetSnapshot	Retrieve current on/off status of Snapshot	84
SetSnapshot	Switch Snapshot on / off	84
GetSnapshotMode	Retrieve Snapshot mode	84
SetSnapshotMode	Set Snapshot mode	84
GetSnapshotCount	Read Snapshot counter (number of PosArray entries)	84
SetSnapshotCount	Set Snapshot counter to less entries (truncate/discard the last entries)	85
GetSnapshotFilter	Retrieve input filter debounce delay	85
SetSnapshotFilter	Set input filter debounce delay	85
GetSnapshotPar	Retrieve Snapshot parameters (signal polarity and modes 0,1)	85
SetSnapshotPar	Set Snapshot parameters (signal polarity and modes 0,1)	86
GetSnapshotPos	Retrieve current Snapshot position	86
GetSnapshotPosArray	Retrieve a Snapshot position from the position array	86
SetSnapshotPosArray	Add or change a position of the position array	87
ClearSnapshotPosArray	Delete all position array entries	87
GetSnapshotIndex	Read Snapshot index (current pointer position in array (0...n-1))	87
SetSnapshotIndex	Set Snapshot index (current pointer position in array (0...n-1))	87

SlideExpress Interface:

Command	Brief Description	Page
Eject	Eject magazines	88
Insert	Magazines are inserted and tested if seated on which slides are present.	88
SlideSeated	Query if slide is present (seated) or not or unknown.	88
MagazinSeated	Query if magazine is present (seated) or not or unknown.	89
GetGripper	Set input filterQuery gripper status information. Returns status of gripper 1 and 2.	89
SetGripper	Set gripper status information. (possibly useful for slide sorting tasks)	89
GetSlide	Get slide(s) from addressed position in magazine or priority handler	89
PutSlide	Put slide(s) back to addressed position in magazine or priority handler	90
GetPrioHandlerPosition	Query actual priority handler position.	90
SetPrioHandlerPosition	Enables user to shift priority handler to required position. Handler is locked at destination or after 30s timeout	90

TrayExpress Interface:

Command	Brief Description	Page
Eject	Eject magazine	91
Insert	Magazine is inserted and tested if seated and which trays are present	91
GetGripper	Retrieve gripper status, e.g. which tray is gripped	92
SetGripper	Set gripper status information	92
GetTray	Get tray from addressed slot in magazine	92
PutTray	Put tray back to addressed slot in magazine	92
GetRFID	Retrieve RFID of addressed tray (if properly seated in magazine)	93
GetNumberOfSlots	Retrieve max available number of slots in magazine	93
GetNumberOfMagazines	Retrieve max available number of magazines	93

4.2. DLL Configuration / Interface

CreateLSID	
Description	This must always be the first command before establish a new connection. This commands the DLL to generate a unique ID to be used to establish a connection. Send this ID as 1 st parameter in all subsequent commands to address one single Tango out of multiple connected Tangos. DLL provides up to 8 ID's , e.g. is able to connect up to 8 Tango controller simultaneously.
C++	int LSX_CreateLSID(int *pLSID);
Parameters	LSID: Contains a new Tango ID-Number after calling CreateLSID, which 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	Connect to Tango. Hint: Use parameter ID given from command CreateLSID(). Without connection setup, connection is not possible.
C++	int LSX_ConnectSimple(int lLSID, int lAnInterfaceType, char *pcAComName, int lABaudRate, BOOL bAShowProt);
Parameters	AnInterfaceType: Interface type = 1 (always 1 for RS232, PCI and USB) Interface type = -1 (connects the DLL to the first USB or PCI TANGO found on the computer, without specifying a COM port) AComName: Name of COM-Interface, e.g. "COM2" ABaudRate: e.g. 57600 Baud (only important for RS232) AShowProt: Determines, if interface protocol shall be shown
Example	pTango->ConnectSimple(1, 1, "COM2", 57600, TRUE); pTango->ConnectSimple(1, -1, NULL, 57600, TRUE); // Autoconnect with the first found USB or PCI TANGO in the system

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);

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

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

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

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

SendStringPosCmd	
Description	Send move command to Tango as a string and wait for return message.
C++	<pre>int LSX_SendStringPosCmd (int ILSID, char *pcStr, char *pcRet, int lMaxLen, BOOL bReadLine, int lTimeout);</pre>
Parameters	<p>Str → Zero-terminated ASCII string, which is to be sent to the controller</p> <p>Ret → Buffer, containing return message from Tango, in case ReadLine = TRUE Or also ZERO (NULL), in case ReadLine = FALSE;</p> <p>MaxLen → Max. amount of characters allowed copied into buffer</p> <p>ReadLine → TRUE = read return message from Tango FALSE = don't wait for return message</p> <p>Timeout → Max. waiting period for return message [ms]</p>
Example	<pre>pTango->SendStringPosCmd(1, "!moa 1 2\r", pcRet, 256, TRUE, 10000);</pre>

SetAbortFlag

Description	<p>Set flag so that communication with Tango is cut off.</p> <p>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.</p>
C++	int LSX_SetAbortFlag (int ILSID);
Parameters	-
Example	<pre>pTango->SetAbortFlag(1); pTango->StopAxes(1); // closes communication with Tango and sends stop command for all axes</pre>

SetShowProt

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

GetCommandTimeout

Description	read current DLL timeout for read, move and calibration
C++	int LSX_GetCommandTimeout (int ILSID, int *toRead, int *toMove, int *toCal);
Parameters	toRead: DLL standard timeout to get answer from 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 ILSID, 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);

GetDLLVersionString

Description	get DLL version string
C++	int LSX_GetDLLVesionString (int ILSID, char *pcVers, int IMaxLen);
Parameters	<i>pcVers</i> → Buffer, containing return message from DLL <i>IMaxLen</i> → Max. amount of characters allowed copied into buffer
Example	pTango->GetDLLVesionString (ILSID, pcVers, IMaxLen);

LoadConfig

Description	Load configuration data from certain file
C++	int LSX_LoadConfig (int ILSID, char *pcFileName);
Parameters	<i>pcFileName</i> → file name to be used to read data from. Data must be simple ASCII text only.
Example	pTango-> LoadConfig (ILSID, pcFileName);

Connect

Description	Connect using previously loaded configuration data
C++	int LSX_Connect (int ILSID);
Parameters	
Example	pTango-> LoadConfig (ILSID);

SaveConfig

Description	Save configuration data to certain file
C++	int LSX_SaveConfig (int ILSID, char *pcFileName);
Parameters	<i>pcFileName</i> → file name to be used to write data to. Data is simple ASCII text only.
Example	pTango-> SaveConfig (ILSID, pcFileName);

SetLanguage

Description	Set language of protocol window
C++	int LSX_SaveConfig (int ILSID, char *pcPLN);
Parameters	<i>pcPLN</i> → 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 (ILSID, pcPLN);

4.3. Controller Information

GetSerialNr	
Description	Reads out the Tango serial number.
C++	int LSX_GetSerialNr (int ILSID, char *pcSerialNr, int IMaxLen);
Parameters	<p>SerialNr: Pointer to a buffer, in which the serial number will be returned</p> <p>MaxLen: Max. amount of digits allowed to be copied into buffer</p> <p>Example value 090103001 = 09 = YY, 01 = WW, 03 = 3Axes max., 001 Index</p>
Example	pTango->GetSerialNr(1, pcSerialNr, 256);

GetVersionStr	
Description	Returns current firmware version number (?ver).
C++	int LSX_GetVersionStr (int ILSID, char *pcVers, int IMaxLen);
Parameters	<p>Vers: Pointer to a character buffer, in which the version number will be returned</p> <p>MaxLen: Max. amount of characters allowed to be copied into buffer</p>
Example	<p>pTango->GetVersionStr(1, pcVers, 64);</p> <p>// retrieve version number</p>

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

GetVersionStrInfo	
Description	Provides optional internal information on the controller version (?iver).
C++	int LSX_GetVersionStrInfo (int ILSID, char *pcVersInfo, int IMaxLen);
Parameters	<p>VersInfo: Pointer to a buffer</p> <p>MaxLen: Max. amount of characters to be copied into buffer</p>
Example	pTango->GetVersionStrInfo(1, pcVersInfo, 16);

GetStageSN	
Description	Provides optional internal information on the stage serial number (?stagesn).
C++	int LSX_GetStageSN (int lLSID, char *pcSN, int lMaxLen);
Parameters	<i>pcSN</i> : Pointer to a buffer <i>MaxLen</i> : Max. amount of characters to be copied into buffer
Example	pTango->GetVersionStrInfo(1, pcSN, 16);

4.4. Status Requests

GetError	
Description	Provides current error number.
C++	int LSX_GetError (int ILSID, int *plErrorCode);
Parameters	<i>ErrorCode</i> : Error number
Example	pTango->GetError(1, &ErrorCode);

GetPos	
Description	Retrieves current position of all axes.
C++	int LSX_GetPos (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	<i>X, Y, Z, A</i> : Positions
Example	pTango->GetPos(1, &X, &Y, &Z, &A);

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

GetPosSingleAxis

Description	Retrieves current position of a single axis. If axis is not available, 0.0 is returned as a value.
C++	<code>int LSX_GetPosSingleAxis (int ILSID, int IAxis, double *pdPos);</code>
Parameters	Axis: Axis of which the position parameters shall be retrieved from, X, Y, Z and A, numbered from 1 to 4 Pos: Positions
Example	<code>pTango->GetPosSingleAxis(1, 2, &Pos);</code> <i>// retrieves position of Y-Axis</i>

GetStatus

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

GetStatusAxis

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

GetStatusLimit

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

SetAutoStatus

Description	<p>Switches Auto-Status on/off.</p> <p>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.</p>
C++	<code>int LSX_SetAutoStatus (int ILSID, int IValue);</code>
Parameters	<p>Value: AutoStatus mode:</p> <p>0 → Controller sends no status</p> <p>1 → Controller automatically sends "Position reached" messages</p> <p>2 → Controller automatically sends "Position reached" and status messages</p> <p>3 → There is only one carriage return sent for "Position reached"</p>
Example	<code>pTango->SetAutoStatus(1, 1);</code>

4.5. Settings

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

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

GetActiveAxes	
Description	Provides the axis enable states.
C++	<code>int LSX_GetActiveAxes (int ILSID, int *plFlags);</code>
Parameters	<i>Flags</i> : 32-Bit Integer. After calling this function the axis bitmask is returned in Bits 0-4 Bit 0 = 1 → X-Axis cleared Bit 2 = 0 → Z-Axis not cleared
Example	<code>pTango->GetActiveAxes(1, &Flags);</code>

GetAccelFunc	
Description	Retrieves acceleration function.
C++	<code>int LSX_GetAccelFunc (int ILSID, int *IX, int *IY, int *IZ, int *IR);</code>
Parameters	<i>IX, IY, IZ, IR</i> : Acceleration function 0 indicate trapezoidal 1 indicate sinusoidal
Example	<code>pTango->GetAccel(1, &IX, &IY, &IZ, &IR);</code>

SetAccelFunc	
Description	Sets acceleration function (0 for trapezoidal, 1 for sinusoidal).
C++	<code>int LSX_SetAccelFunc (int ILSID, int IX, int IY, int IZ, int IR);</code>
Parameters	<i>IX, IY, IZ, IR</i> : Acceleration function 0 indicate trapezoidal 1 indicate sinusoidal
Example	<code>pTango->SetAccel(1, IX, IY, IZ, IR);</code>

SetActiveAxes	
Description	Enable or disable axes.
C++	<code>int LSX_SetActiveAxes (int ILSID, int IFlags);</code>
Parameters	Flags: Bit mask, bits 0 to 4 represent axes X to A Bit 0 = 1 → X-Axis disabled Bit 2 = 0 → Z-Axis enabled
Example	<pre>pTango->SetActiveAxes(1, 3); // X- and Y-Axis cleared (Bits 0 and 1 set), // Z-Axis not cleared (Bit 2 = 0)</pre>

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

SetAxisDirection	
Description	Set axis directions.
C++	<code>int LSX_SetAxisDirection (int ILSID, int lXD, int lYD, int lZD, int lAD);</code>
Parameters	XD, YD, ZD, AD: 4 32-Bit Integers 0 → normal motor turning direction 1 → reverse reversed motor turning direction
Example	<pre>pTango->SetAxisDirection(1, 1, 0, 0, 0); // reverse direction of X-Axis</pre>

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

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

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

GetCalibrateDir	
Description	Retrieves calibrating direction.
C++	int LSX_GetCalibrateDir (int ILSID, int *plXD, int *plYD, int *plZD, int *plAD);
Parameters	<i>XD, YD, ZD, AD</i> : 32-Bit Integer 0 → normal calibration direction 1 → reversed calibration direction
Example	pTango->GetCalibrateDir(1, &XD, &YD,&ZD,&AD);

SetCalibrateDir	
Description	Set calibrating direction.
C++	int LSX_SetCalibrateDir (int ILSID, int IXd, int IYd, int IZd, int IAd);
Parameters	<i>XD, YD, ZD, AD</i> : 32-Bit Integer 0 → normal calibration direction 1 → reverse calibration direction
Example	pTango->(1, 1, 1, 0, 0);

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

SetCurrentDelay	
Description	Sets the time delay, after which the motor current is reduced.
C++	int LSX_SetCurrentDelay (int ILSID, int IX, int IY, int IZ, int IA);
Parameters	<i>X, Y, Z, A</i> : 010000 [ms] (A delay of 0 disables the current reduction)
Example	pTango->SetCurrentDelay(1, 100, 300, 1000, 0);

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

SetDimensions	
Description	Set measuring units of axes.
C++	int LSX_SetDimensions (int ILSID, int IXD, int IYD, int IZD, int IAD);
Parameters	<p><i>XD, YD, ZD, AD</i>: Dimension units</p> <p>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</p>
Example	<p>pTango->SetDimensions(1, 3, 2, 2, 1);</p> <p><i>// X-Axis in degree, Y- and Z-Axis in mm and A-Axis in μm</i></p>

GetResolution	
Description	<p>This command is helpful working with default dimension 2 or 9.</p> <p>Provides the applied number of digits after the decimal point.</p> <p>The Tango preset is 4 digits (1/10μm) resolution for preset dimensions 2 or 9.</p> <p>Data from Tango to DLL is transmitted with this resolution.</p>
C++	int LSX_GetDimensions (int ILSID, int *plValue);
Parameters	<p><i>Value</i>: Resolution units</p> <p>3 → 1 μm resolution 4 → 0.1 μm resolution (Preset) 5 → 10 nm resolution 6 → 1 nm resolution</p>
Example	pTango->GetResolution(1, &resolution);

SetResolution	
Description	Sets the required number of digits after the decimal point. The Tango preset is 4 digits (1/10µm) resolution for preset dimensions 2 or 9. You may specify 5 (10nm) or 6 (1nm) digits to get higher resolution from Tango.
C++	int LSX_SetDimensions (int ILSID, int IValue);
Parameters	<i>Value</i> : Resolution units 3 → 1 µm resolution 4 → 0.1 µm resolution (Preset) 5 → 10 nm resolution 6 → 1 nm resolution
Example	pTango->SetResolution(1, 5); <i>// set 5 digits after the decimal point for all axes</i>

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

SetGear	
Description	Set gear ratio.
C++	int LSX_SetGear (int ILSID, double dX, double dY, double dZ, double dA);
Parameters	<i>X, Y, Z, A</i> : 0.01 - 1000
Example	pTango->SetGear(1, 4.0, 2.0, 1.0, 1.0); <i>// programs gear ratios 1/4 for X, 1/2 for Y and 1/1 for Z and A</i>

GetMotorCurrent

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

SetMotorCurrent

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

GetMotorSteps

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

SetMotorSteps

Description	Set number of motor steps. (default 200 for 1,8° stepper motors)
C++	<code>int LSX_SetMotorSteps (int ILSID, int lX, int lY, int lZ, int lA);</code>
Parameters	<i>X, Y, Z, A</i> : Motor steps X, Y, Z and A-Axis
Example	<code>pTango->SetMotorCurrent(1, 200, 200, 200, 20);</code> <i>// set X, Y, Z to default and A axis to 20 for special motor</i>

GetPitch

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

SetPitch	
Description	Set spindle pitch.
C++	int LSX_SetPitch (int ILSID, double dX, double dY, double dZ, double dA);
Parameters	<i>X, Y, Z, A:</i> 0.001 - 68 [mm]
Example	pTango->SetPitch(1, 4, 4, 4, 4); <i>// Set spindle pitch of all axes to 4mm</i>

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

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

GetReduction	
Description	Retrieves motor current reduction factor.
C++	int LSX_GetReduction (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA)
Parameters	<i>X, Y, Z, A:</i> Electrical motor current reduction (Within parameters from 0 to 1)
Example	pTango->GetReduction(1, &X, &Y, &Z, &A);

SetReduction	
Description	Set reduction factor of motor current.
C++	int LSX_SetReduction (int ILSID, double dX, double dY, double dZ, double dA);
Parameters	<i>X, Y, Z, A:</i> 0 - 1.0
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

GetRMOffset	
Description	Retrieves axis position offsets to RM limit switch.
C++	int LSX_GetRMOffset (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	<i>X, Y, Z, A:</i> 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. The axis stops this amount before the hardware RM endswitch.
C++	int LSX_SetRMOffset (int ILSID, double dX, double dY, double dZ, double dA);
Parameters	<i>X, Y, Z, A:</i> 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

GetSpeedPoti	
Description	Shows, whether the speed potentiometer functionality is switched on or off.
C++	int LSX_GetSpeedPoti (int ILSID, BOOL *pbSpePoti);
Parameter:	The SpePoti flag shows, whether potentiometer is switched on or off
Example	pTango->(1, &flag);

SetSpeedPoti	
Description	Switches Speed Potentiometer functionality on or off.
C++	<code>int LSX_SetSpeedPoti (int ILSID, BOOL bSpeedPoti);</code>
Parameters	<i>SpeedPoti</i> = FALSE → pre-set speed (vel) is used as movement speed = TRUE → pre-set speed (vel) can be reduced depending on the speed-potentiometer deflection
Example	<code>pTango->SetSpeedPoti(1, TRUE);</code> <i>// potentiometer is switched on</i>

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

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

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

SetStopPolarity

Description	Set polarity for active stop input signal. 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 ILSID, 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.
C++	int LSX_GetVel (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	pdX, pdY, pdZ, pdA: Velocity values [r/sec]
Example	pTango->GetVel(1, &X, &Y, &Z, &A);

SetVel

Description	Set velocity of all axes.
C++	int LSX_SetVel (int ILSID, double dX, double dY, double dZ, double dA);
Parameters	X, Y, Z, A: >0 – max. speed [r/sec]
Example	pTango->SetVel(1, 20.0, 15.0, 0.5, 10);

GetVelFac

Description	Retrieves velocity reduction factor of all axes.
C++	int LSX_GetVelFac (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	X, Y, Z, A: Velocity factor
Example	pTango->GetVelFac(1, &X, &Y, &Z, &A);

SetVelFac

Description	Set velocity reduction factor.
C++	<code>int LSX_SetVelFac (int ILSID, double dX, double dY, double dZ, double dA);</code>
Parameters	<i>X, Y, Z, A</i> : Velocity reduction factor, within parameters 0.01 -- 1.00
Example	<code>pTango->SetVelFac(1, 1, 1, 0.1, 0.1);</code> <i>// reduces velocity of Z and A axes to 1/10 of nominal velocity</i>

LStepSave

Description	Save current configuration in Tango (EEPROM).
C++	<code>int LSX_LStepSave (int ILSID);</code>
Parameters	-
Example	<code>pTango->LStepSave(1);</code>

SetAccelSingleAxis

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

SetVelSingleAxis

Description	Set velocity of a single axis.
C++	<code>int LSX_SetVelSingleAxis (int ILSID, int IAxis, double dVel);</code>
Parameters	<i>Axis</i> : X, Y, Z, A numbered from 1 to 4 <i>Vel</i> : >0 – max. speed [r/sec]
Example	<code>pTango->SetVelSingleAxis(1, 2, 10.0);</code> <i>// sets speed of Y-Axis to 10 r/sec</i>

SoftwareReset

Description	Software is reset to starting condition (reboot).
C++	<code>int LSX_SoftwareReset (int ILSID);</code>
Parameters	-
Example	<code>pTango->SoftwareReset(1);</code>

IsVel

Description	Read the actual velocities at which the axes are currently travelling. Unlike '?vel' or '?speed' this instruction returns the currently travelled (true) speed of the axes, even when controlled by a HDI device.
C++	int LSX_IsVel (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	<i>pdX, pdY, pd Z, pdA</i> : 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.
C++	int LSX_IsVelSingleAxis (int ILSID, int lAxis, double *pdVel);
Parameters	<i>lAxis</i> : X, Y, Z, A numbered from 1 to 4 <i>pdVel</i> : actual axis velocity in [mm/s]
Example	pTango->IsVel(1, 2, &vel); //returns actual velocity of y axis

4.6. Move Commands and Positioning Management

Calibrate	
Description	<p>All enabled axes will be calibrated.</p> <p>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.</p>
C++	int LSX_Calibrate (int ILSID);
Parameters	-
Example	pTango->Calibrate(1);

CalibrateEx	
Description	<p>Calibrates single axes.</p> <p>Only calibrates axes with corresponding Bit set in transferred Integer value.</p>
C++	int LSX_CalibrateEx (int ILSID, int IFlags);
Parameters	<p>Flags: Bit mask</p> <p>Bit 0=X, Bit 1=Y, Bit 2=Z, Bit 3=A</p> <p>If Bit 2 = 1 → calibrate Z-Axis</p> <p>If Bit 2 = 0 → do not calibrate Z-Axis</p>
Example	<p>pTango->CalibrateEx(1, 6);</p> <p><i>// only calibrate Y- and Z-Axis (Bit 1 and 2 set)</i></p>

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

GetDelay

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

SetDelay

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

GetDistance

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

SetDistance

Description	Set 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 ILSID, double dX, double dY, double dZ, double dA);
Parameters	<i>X, Y, Z, A</i> : Min-/max- travel range, values depend on measuring unit.
Example	pTango->SetDistance(1, 1, 2, 0, 0); <i>// sets distances for axes X to 1mm and Y to 2mm (if dimension=2), Z and A are not moved when calling function LSX MoveRelShort</i>

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

MoveAbsSingleAxis	
Description	Positions a single axis at the transferred position.
C++	int LSX_MoveAbsSingleAxis (int ILSID, int lAxis, double dValue, BOOL bWait);
Parameters	Axis: X, Y, Z and A, numbered from 1 to 4 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	<p>Extended move command.</p> <p>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.</p>
C++	<pre>int LSX_MoveEx (int ILSID, double dX, double dY, double dZ, double dA, BOOL bRelative, BOOL bWait, int lAxisCount);</pre>
Parameters	<p>X, Y, Z, A: Position vectors</p> <p>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</p> <p>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.</p> <p>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 ...</p>
Example	<pre>pTango->MoveEx(1, 2.0, 3.0, 0, 0, TRUE, TRUE, 2); // X and Y are moved relatively by 2 or 3, function call returns when positions are reached</pre>

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

MoveRelShort

Description	Relative positioning (short command). This command may be used to execute several fast equal distance relative moves. Distances have to be pre-set once with LSX_SetDistance.
C++	int LSX_MoveRelShort (int ILSID);
Parameters	-
Example	pTango->SetDistance(1, 1.0, 1.0, 0, 0); for (i = 0; i < 10; i++) pTango->MoveRelShort(1); <i>// position X- and Y-Axis 10 times relatively by 1mm</i>

MoveRelSingleAxis

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

RMeasure

Description	Travels to maximum position of all enabled axes. 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 ILSID);
Parameters	-
Example	pTango->RMeasure(1);

RMeasureEx	
Description	Measure maximum position of axes (max. travel range). Moves the stage towards the RM limit switch only for the axes whose corresponding axis bit mask is set.
C++	int LSX_RMeasureEx (int ILSID, int IFlags);
Parameters	<i>Flags</i> : Bit mask Bit 2 = 1 → calibrate Z-Axis Bit 2 = 0 → Do not calibrate Z-Axis ...
Example	pTango->RMeasureEx(1, 2); <i>// only measure maximum position of Y-Axis</i>

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

StopAxes	
Description	Abort. Stops all moving axes.
C++	int LSX_StopAxes (int ILSID);
Parameters	-
Example	pTango->StopAxes(1);

WaitForAxisStop

Description	<p>Function returns as soon as the axes selected by the bit mask “lAFlags” have reached their target positions or the timeout is exceeded.</p> <p>LSX WaitForAxisStop uses '?statusaxis', to poll axis status.</p>
C++	<pre>int LSX_WaitForAxisStop (int ILSID, int lAFlags, int lATimeoutValue, BOOL *pbATimeout);</pre>
Parameters	<p>AFlags: Bit mask</p> <p>Bit 0: X-Axis</p> <p>Bit 1: Y-Axis</p> <p>Bit 2: Z-Axis</p> <p>Bit 3: A-Axis</p> <p>ATimeoutValue: Timeout in milliseconds</p> <p>WaitForAxisStop returns latest after this period of time</p> <p>pbATimeout is set to “TRUE”, if axes are still in motion.</p> <p>Setting lATimeoutValue = 0 disables the Timeout (wait infinite)</p> <p>pbATimeout Flag: Shows whether a Timeout has occurred</p>
Example	<pre>pTango->WaitForAxisStop(1, 3, 0, flag); // wait until X- and Y-Axes have stopped, no Timeout pTango->WaitForAxisStop(1, 7, 10000, flag); // wait until X-, Y- and Z-Axis has stopped, 10 sec. Timeout</pre>

Go

Description	<p>All axes are moved to given absolute positions.</p> <p>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.</p>
C++	<pre>int LSX_Go (int ILSID, double dX, double dY, double dZ, double dA);</pre>
Parameters	<p>X, Y, Z, A: ± Travel range, command depends on measuring unit</p>
Example	<pre>pTango->Go(1, 10.0, 10.0, -10.0, 10.0);</pre>

GoSingleAxis	
Description	<p>One axes is moved to given absolute position.</p> <p>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.</p>
C++	int LSX_GoSingleAxis (int ILSID, int IAxis, double dYValue);
Parameters	X, Y, Z, A: \pm Travel range, command depends on measuring unit
Example	pTango->Go(1, 2, 12.34); //move Y to target position 12.34

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

4.7. Joystick and Handwheel

GetDigJoySpeed	
Description	Retrieves current travel speed (initiated by SetDigJoySpeed digital Joystick command).
C++	<pre>int LSX_GetDigJoySpeed (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</pre>
Parameters	<i>X, Y, Z, A</i> : Speed values [r/sec]
Example	pTango->GetDigJoySpeed(1, &X, &Y, &Z, &A);

SetDigJoySpeed	
Description	<p>This command moves axes at a constant speed.</p> <p>To stop the axes, a speed of 0 has to be set. Else the constant velocity is maintained until approaching a limit switch.</p>
C++	<pre>int LSX_SetDigJoySpeed (int ILSID, double dX, double dY, double dZ, double dA);</pre>
Parameters	<i>X, Y, Z, A</i> : Speed [r/sec], within parameter range: + max. speed
Example	<pre>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"</pre>

GetHandWheel	
Description	Retrieves hand wheel status.
C++	<pre>int LSX_GetHandWheel (int ILSID, BOOL *pbHandWheelOn, BOOL *pbPositionCount, BOOL *pbEncoder);</pre>
Parameters	<p>HandWheelOn: TRUE = hand wheel switched on FALSE = hand wheel switched off</p> <p>PositionCount: TRUE = position count switched on FALSE = position count switched off</p> <p>Encoder: TRUE = encoder values, if available</p>
Example	pTango->GetHandWheel(1, &HandWheelOn, &PositionCount, &Encoder);

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

GetJoystickDir	
Description	Retrieves axis direction for the analog Joystick and other HDI input devices.
C++	<pre>int LSX_GetJoystickDir (int ILSID, int *plXD, int *plYD, int *plZD, int *plAD);</pre>
Parameters	<p><i>XD, YD, ZD, AD:</i></p> <p>0 → Axis disabled for Joystick (deflection ignored)</p> <p>1 → positive axis direction, current reduction disabled</p> <p>-1 → negative axis direction, current reduction disabled</p> <p>2 → positive axis direction with current reduction (default)</p> <p>-2 → negative axis direction with current reduction</p>
Example	pTango->GetJoystickDir(1, &XD, &YD, &ZD, &AD);

SetJoystickDir

Description	Sets axis direction for Joystick and other HDI input devices.
C++	int LSX_SetJoystickDir (int ILSID, int IXD, int IYD, int IZD, int IAD);
Parameters	<p><i>XD, YD, ZD, AD:</i></p> <p>0 → Axis disabled for Joystick (deflection ignored)</p> <p>1 → positive axis direction, current reduction disabled</p> <p>-1 → negative axis direction, current reduction disabled</p> <p>2 → positive axis direction with current reduction (default)</p> <p>-2 → negative axis direction with current reduction</p>
Example	<pre>pTango->SetJoystickDir(1, 1, 1, -1, 0); // X- and Y-Axis positive direction, Z-Axis negative direction, A-Axis blocked</pre>

GetJoystickWindow

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

SetJoystickWindow

Description	Set Joystick idle window. A value in digits which configures an angle where a 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.
C++	int LSX_SetJoystickWindow (int ILSID, int IAValue);
Parameters	<p><i>AValue:</i> Analogue signal range (as digits) in which axes do not move.</p> <p>0 ... 100</p>
Example	pTango->SetJoystickWindow(1, 30);

SetHandWheelOff

Description	Switch hand wheel off.
C++	int LSX_SetHandWheelOff (int ILSID);
Parameters	-
Example	pTango->SetHandWheelOff(1);

SetHandWheelOn	
Description	Switch hand wheel on.
C++	int LSX_SetHandWheelOn (int ILSID, 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); <i>// switch on hand wheel with position count (encoder values)</i>

SetJoystickOff	
Description	Switch analogue Joystick off.
C++	int LSX_SetJoystickOff (int ILSID);
Parameters	-
Example	pTango->SetJoystickOff(1);

SetJoystickOn	
Description	Switch analogue Joystick on.
C++	int LSX_SetJoystickOn (int ILSID, 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); <i>// switch on joystick with position count (encoder values)</i>

GetHwFactor	
Description	Read hand wheel factor of all axes, in [mm per knob rotation]
C++	int LSX_GetHwFactor (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	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]
C++	int LSX_SetHwFactor (int ILSID, double dX, double dY, double dZ, double dA)
Parameters	Double values
Example	pTango->SetHwFactor(1, dX, dY, dZ, dA);

GetHwFactorB

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

SetHwFactorB

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

GetZwTravel

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

GetKey

Description	Get HDI device key states
C++	<code>int LSX_GetKey (int ILSID, BOOL *pbKey1, BOOL *pbKey2, BOOL *pbKey3, BOOL *pbKey4);</code>
Parameters	Pointers to BOOL, TRUE=Key pressed
Example	<code>pTango-> GetKey(1, &bKey[0], &bKey[1], &bKey[2], &bKey[3]);</code>

GetKeyLatch

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

ClearKeyLatch

Description	Clear latched key state(s)
C++	<code>int LSX_ClearKeyLatch (int ILSID, int IKey);</code>
Parameters	IKey: 0 = clear latched keystate of all 4 keys 1 = clear latched keystate of key 1 only 2 = clear latched keystate of key 2 only 3 = clear latched keystate of key 3 only 4 = clear latched keystate of key 4 only
Example	<code>pTango-> ClearKeyLatch(1, 0); // Clear all</code>

4.8. Control Console with Trackball and Joyspeed Keys

GetBPZ	
Description	Retrieves status of a custom-built control console with trackball.
C++	int LSX_GetBPZ (int ILSID, int *pAValue);
Parameters	<i>AValue</i> : 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.
C++	int LSX_SetBPZ (int ILSID, int lAValue);
Parameters	<i>AValue</i> : 0...2 0 → control console is "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.
C++	int LSX_GetBPZJoyspeed (int ILSID, int lAPar, double *pdAValue);
Parameters	<i>APar</i> : 1, 2 or 3 (console keys for speed selection: slow, medium, fast) <i>AValue</i> : max. speed [r/sec]
Example	pTango->GetBPZJoyspeed(1, &AValue); <i>// retrieve set speed of key 1 (slow)</i>

SetBPZJoyspeed

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

GetBPZTrackballBackLash

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

SetBPZTrackballBackLash

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

GetBPZTrackballFactor

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

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

4.9. Limit Switches (Hardware and Software)

GetAutoLimitAfterCalibRM	
Description	Provides, whether internal software limits are set when calibrating (cal) or measuring stage travel range (rm).
C++	int LSX_GetAutoLimitAfterCalibRM (int ILSID, 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.
C++	int LSX_SetAutoLimitAfterCalibRM (int ILSID, 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.
C++	int LSX_GetLimit (int ILSID, int lAxis, double *pdMinRange, double *pdMaxRange);
Parameters	Axis: Axis from which travel range limits are to be retrieved (X, Y, Z, A numbered from 1=X to 4=A) 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.
C++	int LSX_SetLimit (int ILSID, int lAxis, double dMinRange, double dMaxRange);
Parameters	<p>Axis: Axis from which travel range limits are to be retrieved (X, Y, Z, A numbered from 1=X to 4=A)</p> <p>MinRange: lower travel range limit, unit depends on dimension</p> <p>MaxRange: upper travel range limit, unit depends on dimension</p>
Example	<pre>pTango->SetLimit(1, 1, -10.0, 20.0); // assign X-Axis -10 as lower and 20 as upper travel range limits</pre>

GetLimitControl	
Description	Retrieves, whether area control (limits) is switched on or off.
C++	int LSX_GetLimitControl (int ILSID, int lAxis, BOOL *pbActive);
Parameters	<p>Axis: X, Y, Z and A, numbered from 1=X to 4=A</p> <p>Active: TRUE = area control of corresponding axis is active FALSE = area control of corresponding axis is deactivated</p>
Example	pTango->GetLimitControl(1, 2, &Active);

SetLimitControl	
Description	Switches area control on / off.
C++	int LSX_SetLimitControl (int ILSID, int lAxis, BOOL bActive);
Parameters	<p>Axis: X, Y, Z and A, numbered from 1=X to 4=A</p> <p>Active: TRUE = activate area control of corresponding axis FALSE = disable area control of corresponding axis</p>
Example	pTango->SetLimitControl(1, 2, TRUE); // Area control of Y-Axis is active

GetSwitchActive

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

SetSwitchActive

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

GetSwitches

Description	Retrieves actuation status of all limit switches.									
C++	int LSX_GetSwitches (int ILSID, int *plFlags);									
Parameters	<p>Flags: Pointer on Integer Value, which includes status of all limit switches as bit mask</p> <p>In bit mask, status of limit switches is encoded as follows:</p> <table><tr><td>Limit switch</td><td>EE (rm)Ref.</td><td>E0 (cal)</td></tr><tr><td>Axis</td><td>AZYX</td><td>AZYX</td></tr><tr><td>Bit</td><td>0000</td><td>0000</td></tr></table> <p>E.g.:</p> <p>Flags = 0x003 → E0 of X- and Y-Axis are actuated</p> <p>Flags = 0x200 → EE of Y-Axis is actuated</p>	Limit switch	EE (rm)Ref.	E0 (cal)	Axis	AZYX	AZYX	Bit	0000	0000
Limit switch	EE (rm)Ref.	E0 (cal)								
Axis	AZYX	AZYX								
Bit	0000	0000								
Example	pTango->GetSwitches(1, &Flags);									

GetSwitchPolarity

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

SetSwitchPolarity

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

GetSwitchType

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

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

4.10. Digital and Analog Inputs and Outputs

GetAnalogInput	
Description	Retrieves current A/D conversion result of an analogue channel.
C++	<code>int LSX_GetAnalogInput (int ILSID, int IIndex, int *pIValue);</code>
Parameters	<p>Index: 0...15 (analog channel), 0...9 = HDI connector, pins 1...10 10 = ANAIN0 of AUX-IO connector</p> <p>Value: Pointer to Integer value, to which the channel's A/D conversion result is written. 0...5V analog = 0...1023</p>
Example	<code>pTango->GetAnalogInput(1, 0, &Input); // Read channel 0</code>

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

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

SetAnalogOutput	
Description	Set analogue output signals.
C++	<code>int LSX_SetAnalogOutput (int ILSID, int IIndex, int IValue);</code>
Parameters	<p>Index: 0,1 (analogue circuits)</p> <p>Value: 0...100 [%]</p>
Example	<pre>pTango->SetAnalogOutput(1, 0, 100); // set analogue output 0 to max. voltage (10V)</pre>

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 ILSID, int lIndex, BOOL bFkt, double dDist, int lAxis);
Parameters	<i>Index</i> : 0 to 15 (output pin) <i>Fkt</i> = FALSE → activation of an output depending on set distance before reaching determined position <i>Fkt</i> = TRUE → activation of an output depending on set distance after start position <i>Dist</i> : Distance, depends on selected dimension (unit) <i>Axis</i> : X, Y, Z and A, numbered from 1 to 4
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 ILSID, int lIndex);
Parameters	<i>Index</i> : 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 ILSID, 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	Set polarity of digital inputs / outputs.
C++	int LSX_SetDigIO_Polarity (int ILSID, int IIndex, 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); <i>// input pin / output pin 3 high active</i>

SetDigitalOutput	
Description	Set individual digital output pin.
C++	int LSX_SetDigitalOutput (int ILSID, int IIndex, BOOL bValue);
Parameters	Index: 0 to 15 Value: Set pin level to FALSE = low TRUE = high
Example	pTango->SetDigitalOutput(1, 0, TRUE); <i>// set output pin 0 to '1'</i>

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

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

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

SetLedBright	
Description	<p>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.</p> <p>The SetLedBright function also controls the TAKT_OUT digital pin in order to entirely switch of LED100 with the LED-DR1 driver.</p>
C++	int LSX_SetLedBright (int ILSID, double dBright);
Parameters	<p><i>dBright</i>: Brightness of the LED100</p> <p>-1 = OFF A negative value <0 switches the LED entirely off (digital pin)</p> <p>0 ... 100 Brightness in %, up to 3 fractional digits supported</p>
Example	<pre>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</pre>

4.11. Encoder Settings

ClearEncoder	
Description	Reset encoder positions to zero.
C++	int LSX_ClearEncoder (int ILSID, int lAxis);
Parameters	<i>Axis</i> : X, Y, Z and A, numbered from 1 to 4
Example	pTango->ClearEncoder(1, 2); // reset encoder counter of Y-Axis to zero

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

GetEncoderActive	
Description	Retrieves which encoder will be activated after calibration. Please note: This function is corresponding to the „?encmask“ command!
C++	int LSX_GetEncoderActive (int ILSID, int *plFlags);
Parameters	<i>Flags</i> : 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	Retrieves which encoder is activated after calibration Please note: This function is corresponding to „!encmask“ command.
C++	int LSX_SetEncoderActive (int ILSID, int IFlags);
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); <i>// No encoder will be used</i> pTango->SetEncoderActive(1, 2); <i>// encoder of Y-Axis will be activated after calibration</i>

GetEncoderMask	
Description	Retrieve status of encoders. Please note: This function is corresponding to „?enc“ command.
C++	LSX_GetEncoderMask (int ILSID, 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) ... <i>// if encoder of Y-Axis connected + active ...</i>

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

GetEncoderPeriod	
Description	Retrieves encoder signal period length.
C++	<pre>int LSX_GetEncoderPeriod (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</pre>
Parameters	<i>X, Y, Z, A</i> : Period length [mm]
Example	<pre>pTango->GetEncoderPeriod(1, &X, &Y, &Z, &A);</pre>

SetEncoderPeriod

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

GetEncoderPosition

Description	Retrieves position response type.
C++	int LSX_GetEncoderPosition (int ILSID, BOOL *pbValue);
Parameters	<i>Value</i> : TRUE → axis position values will be read from the encoder, if activated. Else the position will be taken from the motor position. FALSE → Position will be taken from the motor position.
Example	pTango->GetEncoderPosition(1, &Value);

SetEncoderPosition

Description	Switches encoder value display on / off.
C++	int LSX_SetEncoderPosition (int ILSID, BOOL bValue);
Parameters	<i>Value</i> : TRUE → axis position values will be read from the encoder, if activated. Else the position will be taken from the motor position. FALSE → Position will be taken from the motor position.
Example	pTango->SetEncoderPosition(1, TRUE);

GetEncoderRefSignal

Description	Retrieves whether the encoder reference signal is evaluated when calibrating.
C++	int LSX_GetEncoderRefSignal (int ILSID, 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	Evaluate reference signal from encoder when calibrating.
C++	int LSX_SetEncoderRefSignal (int lLSID, int lXR, int lYR, int lZR, int lAR);
Parameters	<i>XR, YR, ZR, AR</i> : 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); <i>// when calibrating, reference signals of encoders X and Y are evaluated</i>

4.12. Closed Loop Settings

ClearCtrFastMoveCounter	
Description	If position difference is larger than lock-in range, a new vector will be started and corresponding counter will be increased by one.
C++	int LSX_ClearCtrFastMoveCounter (int ILSID);
Parameters	-
Example	pTango->ClearCtrFastMoveCounter(1);

GetController	
Description	Retrieve Closed Loop mode.
C++	int LSX_GetController (int ILSID, int *pIXC, int *pIYC, int *pIZC, int *pIRC);
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.
C++	int LSX_SetController (int ILSID, int IXC, int IYC, int IZC, int IAC);
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	Provides Closed Loop interval time.
C++	int LSX_GetControllerCall (int ILSID, int *pICtrCall);
Parameter:	<i>ICtrCall</i> : Controller call time [ms]
Example	pTango->GetControllerCall(1, &ICtrCall);

SetControllerCall

Description	Set Closed Loop interval time.
C++	int LSX_SetControllerCall (int ILSID, int IICtrCall);
Parameters	<i>ICtrCall</i> : Controller call time [ms]
Example	pTango->SetControllerCall(1, 5); // ICtrCall = 5 means: Closed Loop controller is called every 5 milliseconds

GetControllerFactor

Description	Retrieve Closed Loop controller factors.
C++	int LSX_GetControllerFactor (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	<i>X, Y, Z, A</i> : Closed Loop factors
Example	pTango->GetControllerFactor(1, &X, &Y, &Z, &A);

SetControllerFactor

Description	Set Closed Loop controller factor.
C++	int LSX_SetControllerFactor (int ILSID, double dX, double dY, double dZ, double dA);
Parameters	<i>X, Y, Z, A</i> : 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

GetControllerSteps

Description	Retrieves length of controller steps.
C++	<pre>int LSX_GetControllerSteps (int ILSID, double *pdX, double *pdY, double *pdZ, double *pdA);</pre>
Parameters	<i>X, Y, Z, A</i> : Length of controller steps [mm]
Example	<code>pTango->GetControllerSteps(1, &X, &Y, &Z, &A);</code>

SetControllerSteps

Description	Set controller steps.
C++	<pre>int LSX_SetControllerSteps (int ILSID, double dX, double dY, double dZ, double dA);</pre>
Parameters	<i>X, Y, Z, A</i> : 1 - spindle pitch (values depend on dimension)
Example	<code>pTango->SetControllerSteps(1, 4, 5, 7, 9);</code>

GetControllerTimeout

Description	Retrieves controller timeout.
C++	<pre>Int LSX_GetControllerTimeout (int ILSID, int *pLACtrTimeout);</pre>
Parameters	<i>ACtrTimeout</i> : Timeout [ms], If the Closed Loop controller is unable to settle in the target window for this time, the move is aborted (move function calls return with error code 4013).
Example	<code>pTango->GetControllerTimeout(1, &ACtrTimeout);</code>

SetControllerTimeout

Description	Set controller timeout.
C++	<pre>int LSX_SetControllerTimeout (int ILSID, int lACtrTimeout);</pre>
Parameters	<i>ACtrTimeout</i> : 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	<pre>pTango->SetControllerTimeout(1, 500); // Abort after trying to settle in the target window for 500ms</pre>

GetControllerTWDelay

Description	Retrieve controller delay.
C++	<code>int LSX_GetControllerTWDelay (int ILSID, int *plCtrTWDelay);</code>
Parameters	CtrTWDelay: Controller delay [ms]
Example	<code>pTango->GetControllerTWDelay(1, &CtrTWDelay);</code>

SetControllerTWDelay

Description	Set controller delay.
C++	<code>int LSX_SetControllerTWDelay (int ILSID, int lCtrTWDelay);</code>
Parameters	CtrTWDelay: Controller delay 0 - 250 ms Time for which the axis has to remain in the target window. Moves are delayed by at least this time.
Example	<code>pTango->SetControllerTWDelay(1, 0);</code> <i>// controller delay switched off, closed loop end position will be inaccurate</i>

GetCtrFastMove

Description	Retrieves setting of FastMove function.
C++	<code>int LSX_GetCtrFastMove (int ILSID, BOOL *pbActive);</code>
Parameters	Active: TRUE → FastMove function active
Example	<code>pTango->GetCtrFastMove(1, &Active);</code>

GetCtrFastMoveCounter

Description	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.
C++	<code>int LSX_GetCtrFastMoveCounter (int ILSID, int *plXC, int *plYC, int *plZC, int *plAC);</code>
Parameters	XC, YC, ZC, AC: Number of carried out Fast Move functions
Example	<code>pTango->GetCtrFastMoveCounter(1, &XC, &YC,&ZC,&AC);</code>

GetTargetWindow

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

SetTargetWindow

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

SetCtrFastMoveOff

Description	FastMove function deactivated.
C++	<pre>int LSX_SetCtrFastMoveOff (int ILSID);</pre>
Parameters	-
Example	<code>pTango->SetCtrFastMoveOff(1);</code>

SetCtrFastMoveOn

Description	Activate FastMove function , meaning a new vector is started if controller position difference is larger than the lock-in range.
C++	<pre>int LSX_SetCtrFastMoveOn (int ILSID);</pre>
Parameters	-
Example	<code>pTango->SetCtrFastMoveOn(1);</code>

4.13. Trigger Output

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

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

GetTrigger	
Description	Retrieve trigger setting.
C++	int LSX_GetTrigger (int ILSID, BOOL *pbATrigger);
Parameters	<i>ATrigger</i> : TRUE → trigger is "On" FALSE → trigger is "Off"
Example	pTango->GetTrigger(1, &ATrigger);

SetTrigger	
Description	Switch trigger on / off.
C++	int LSX_SetTrigger (int ILSID, BOOL bATrigger);
Parameters	<i>ATrigger</i> = TRUE → switch trigger on = FALSE → switch trigger off
Example	pTango->SetTrigger(1, TRUE);

GetTriggerPar

Description	Retrieves trigger parameters.
C++	<pre>int LSX_GetTriggerPar (int ILSID, int *plAxis, int *plMode, int *plSignal, double *pdDistance);</pre>
Parameters	<p>Axis: Axis 1...4</p> <p>Mode: Trigger mode (see command !trigm)</p> <p>Signal: Trigger signal (see command !trigs)</p> <p>Distance: Trigger distance (see command !trigd)</p>
Example	pTango->GetTriggerPar(1, &Axis, &Mode, &Signal, &Distance);

SetTriggerPar

Description	Set trigger parameters.
C++	<pre>int LSX_SetTriggerPar (int ILSID, int lAxis, int lMode, int lSignal, double dDistance);</pre>
Parameters	<p>Axis: Axis 1...4</p> <p>Mode: Trigger mode (see command !trigm)</p> <p>Signal: Trigger signal (see command !trigs)</p> <p>Distance: Trigger distance (see command !trigd)</p>
Example	pTango->SetTriggerPar(1, 1, 3, 2, 5.0);

4.14. Snapshot Input

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

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

GetSnapshotMode	
Description	Provides the current Snapshot mode.
C++	<code>int LSX_GetSnapshotMode (int ILSID, int*plMode);</code>
Parameters	<i>Mode</i> : 0-11 (refer to snsm documentation in TANGO Instruction Set)
Example	<code>pTango->GetSnapshotMode(1, &Mode);</code>

SetSnapshotMode	
Description	Sets the Snapshot mode (functionality).
C++	<code>int LSX_SetSnapshotMode (int ILSID, int lMode);</code>
Parameters	<i>Mode</i> : 0-11 (refer to snsm documentation in TANGO Instruction Set)
Example	<code>pTango->SetSnapshotMode(1, 0); // Set mode to 0 = capture positions @ HDI F2 key</code>

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

SetSnapshotCount

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

GetSnapshotFilter

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

SetSnapshotFilter

Description	Set input filter when switches chatter.
C++	<code>int LSX_SetSnapshotFilter (int ILSID, int lTime);</code>
Parameters	<i>Time</i> : Filter time, within 0-100 ms
Example	<code>pTango->SetSnapshotFilter(1, 0); // no filter, fast response (e.g. for TTL signals)</code>

GetSnapshotPar

Description	Retrieve Snapshot parameters.
C++	<code>int LSX_GetSnapshotPar (int ILSID, BOOL *pbHigh, BOOL *pbAutoMode);</code>
Parameters	<p><i>High</i>: TRUE → snapshot is high active FALSE → snapshot is low active</p> <p><i>AutoMode</i>: TRUE → snapshot "Automatic": Position is automatically moved to after first snapshot pulse (corresponds to SnapshotMode 1) FALSE → snapshot capture mode (corresponds to SnapshotMode 0)</p>
Example	<code>pTango->GetSnapshotPar(1, &High, &AutoMode);</code>

SetSnapshotPar

Description	Set Snapshot parameters (polarity and mode 0 or 1). The AutoMode might interfere with a previously set SnapshotMode, if that was set to a mode higher than 1).
C++	<code>int LSX_SetSnapshotPar (int ILSID, BOOL bHigh, BOOL bAutoMode);</code>
Parameters	High: TRUE → snapshot is high active FALSE → snapshot is low active AutoMode: TRUE → snapshot "Automatic": Position is automatically moved to after first snapshot pulse (corresponds to SnapshotMode 1) FALSE → snapshot capture mode (corresponds to SnapshotMode 0)
Example	<code>pTango->SetSnapshotPar(1, TRUE, FALSE);</code>

GetSnapshotPos

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

GetSnapshotPosArray

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

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

ClearSnapshotPosArray	
Description	Deletes the entire position array (clear all entries).
C++	<pre>int LSX_ClearSnapshotPosArray (int ILSID,);</pre>
Parameters	-
Example	<pre>pTango->ClearSnapshotPosArray(1); // Delete the entire PosArray</pre>

GetSnapshotIndex	
Description	Retrieve the current Snapshot index, 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++	<pre>int LSX_GetSnapshotIndex (int ILSID, int *plSnsIndex);</pre>
Parameters	SnsIndex: Current position of the index pointer within the position array
Example	<pre>pTango->GetSnapshotIndex(1, &SnsIndex);</pre>

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

5. SlideExpress Interface

This chapter describes additional DLL functions usable with SlideExpress. From application point of view there are only few differences between previous top loader and new front loader system.

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

5.1. Eject

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

5.2. Insert

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

5.3. SlideSeated

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

5.4. MagazinSeated

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

5.5. GetGripper

Description	Query gripper status information. Returns status of gripper 1 and 2.
C++	<code>int LSX_GetGripper (int ILSID, int *c1, int *r1, int *c2, int *r2);</code>
Parameters	c1 → column number [-1, 0, 1..MAXCOL] of slide 1 in gripper r1 → row number [-1, 0, 1..MAXROW] of slide 1 in gripper c2 → column number [-1, 0, 1..MAXCOL] of slide 2 in gripper r2 → row number [-1, 0, 1..MAXROW] of slide 2 in gripper
Example	<code>pTango-> GetGripper (1, &c1, &r1, &c2, &r2); //check status of gripper 1 and 2</code> 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)

5.6. SetGripper

Description	Set gripper status information. (possibly useful for slide sorting tasks)
C++	<code>int LSX_SetGripper (int ILSID, int c1, int r1, int c2, int r2);</code>
Parameters	c1 → column number [-1, 0, 1..MAXCOL] of slide 1 in gripper r1 → row number [-1, 0, 1..MAXROW] of slide 1 in gripper c2 → column number [-1, 0, 1..MAXCOL] of slide 2 in gripper r2 → row number [-1, 0, 1..MAXROW] of slide 2 in gripper
Example	<code>pTango->SetGripper (1, 0, 0, 0, 0); //set gripper to "empty"</code>

5.7. GetSlide

Description	Get slide(s) from addressed position in magazine or priority handler.
C++	<code>int LSX_GetSlide (int ILSID, int col, int row, int mode);</code>
Parameters	col → column number [1..MAXCOL] row → row number [1..MAXROW] (obsolete: or [0] for priority handler) mode → (0 = inspection, 1 = oiler, 2 = bar code reader)
Example	<code>pTango-> GetSlide (1, 1, 1, 0);</code>

5.8. PutSlide

Description	Put slide(s) back to addressed position in magazine or priority handler.
C++	<code>int LSX_PutSlide (int ILSID, int col, int row);</code>
Parameters	col → column [1..MAXCOL] row → slot number [1..MAXROW] (obsolete: or [0] for priority handler) If both parameters are 0 the DLL transmits !putslide without arguments. In this case Tango uses known gripper information to put slides back (if any).
Example	<code>pTango->PutSlide (1, 4, 50); //put slide to magazine 4 slot 50.</code>

Obsolete:

5.9. GetPrioHandlerPos

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

Obsolete:

5.10. SetPrioHandlerPos

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

6. TrayExpress Interface

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

6.1. Eject

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

6.2. Insert

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

6.3. SlideSeated

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

6.4. MagazinSeated

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

6.5. GetGripper

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

6.6. SetGripper

Description	Set gripper status information. (possibly useful for tray sorting tasks)
C++	<code>int LSX_SetGripper (int ILSID, int c1, int s1, int c2, int s2);</code>
Parameters	c1 → magazine number [-1, 0, 1..4] of slide in gripper s1 → slot number [-1, 0, 1..50] of slide in gripper c2 → dummy for compatibility with slide express s2 → dummy for compatibility with slide express
Example	pTango->SetGripper (1, 0, 0, 0, 0); //set gripper to "empty"

6.7. GetTray

Description	Get tray from addressed position in magazine
C++	<code>int LSX_GetTray (int ILSID, int slot, int mode);</code>
Parameters	slot → slot number [1..24] mode → (0 = inspection, 1 = oiler, 2 = bar code reader)
Example	pTango->GetTray (1, 1, 0);

6.8. PutTray

Description	Put tray back to addressed position in magazine
C++	<code>int LSX_PutTray (int ILSID, int slot);</code>
Parameters	slot → slot number [1..24]
Example	pTango->PutSlide (1, 10); //put tray to magazine slot 10.

6.9. GetRFID

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

6.10. SetRFID

Description	Set RFID stores data into addressed magazine slot if tray is properly seated
C++	<code>int LSX_SetRFID (int ILSID, int slot, int bank, int rfddata);</code>
Parameters	slot → slot number [1..MAXSLOT] bank → bank number [2 to 64] (bank 0 and 1 are not writeable) rfddata → int contains customer data to be coded into RFID transponder device
Example	<code>pTango-> SetTray (1, 1, 0);</code>

6.11. GetNumberOfSlots

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

6.12. GetNumberOfMagazines

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

7. Express Interface Extensions

Following commands are superset of SlideExpress and TrayExpress commands and expand commands of previous 2 chapters.

7.1. GetLoaderType

Description	Get loader type Response depends on system configuration.
C++	<code>int LSX_GetLoaderType (int ILSID, int *plLoaderType);</code>
Parameters	plLoaderType → pointer to int returns loader type 0 => SlideExpress 1 => Manual System (customer special) 2 => Loader System (customer special) 3 => Loader System (slave response of 2 nd Tango)
Example	<code>pTango->GetLoaderType (1, plLoaderType);</code>

7.2. GetNumberOfRows

Description	Get number of magazine rows, e.g. max. number of slots to insert trays Response is number of magazine rows.
C++	<code>int LSX_GetNumberOfRows (int ILSID, int *plRows);</code>
Parameters	plRows → pointer to int returns number of magazine rows (1 for manual, 35 for loader system)
Example	<code>pTango->GetNumberOfRows (1, plRows);</code>

7.3. GetNumberOfColumns

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

7.4. GetTraySN

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

7.5. GetTrayType

Description	GetTrayType returns tray type of addressed tray. (Data is read from RFID transponder.)
C++	<code>int LSX_GetTrayType (int ILSID, int slot, int *plTrayType);</code>
Parameters	plTrayType → pointer to int returns tray type (user coded data)
Example	<code>pTango->GetTrayType (1, 1, plTrayType);</code>

7.6. SetTrayType

Description	SetTrayType stores tray type data into RFID transponder of addressed slot / tray.
C++	<code>int LSX_SetTrayType (int ILSID, int slot, int aTrayType);</code>
Parameters	aTrayType → int data contains information of required tray type
Example	<code>int aTrayType = 0x0100010a; //see customer specification requirements for explanation pTango->SetTrayType (1, 1, aTrayType);</code>

7.7. SetCabinLED

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

7.8. GetCabinLED

Description	GetCabinLED returns actual state of loader illumination
C++	<code>int LSX_GetCabinLED (int ILSID, int *plState);</code>
Parameters	plState → pointer to int returns illumination state
Example	<code>pTango->GetCabinLED (1,plState);</code>

7.9. SetLabelLED

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

7.10. GetLabelLED

Description	GetLabelLED returns actual state of label illumination
C++	<code>int LSX_GetLabelLED (int lLSID, int *plState);</code>
Parameters	plState → pointer to int returns illumination state
Example	pTango->GetLabelLED (1,plState);

8. Error Codes

8.1. Tango Error Messages

1	no valid axis name
2	no executable instruction
3	command line too long
4	unknown keyword
5	number outside range
6	wrong number of arguments
7	missing '!' or '?'
8	TVR not possible while axis is moving
9	switching axis on/off not possible while TVR is active
10	function not configured
11	automatic move not possible while jogging manual
12	limit switch activated
13	function not executable because encoder detected
14	error during calibration (limit switch not released)
27	STOP input active
29	amplifier off
50	one argument only expected
51	argument is not a number
52	keyword BEGIN or EOF missing
53	unexpected geo type
58	unexpected sequence
59	alpha and beta must not be equal
70	wrong CPLD data
71	ETS error
72	parameter is write protected
73	internal error
74	closed loop switched off due to parameter change
75	axis correction disabled or not enabled
76	io extension error
77	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
83	overload on AUX I/O +5V supply
84	overload on encoder +5V supply
85	overload on AUX I/O +24V supply
86	low brake output voltage

Following errors are generated from SlideExpress or TrayExpress only.

Error messages for SlideExpress and TrayExpress

100	hardware missing (IO1)	
101	magazine not correct seated	
102	magazine slot is empty	
103	magazine slot is occupied	
104	sensor reports get failure	(during pull from magazine)
105	sensor reports put failure	(during insert in magazine)
106	sensor overmodulation	
107	magazine unknown	
108	ejector timeout	(magnet or other part)
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	timeout open door	
117	timeout close door	
118	no priority handler available	
119	gripper is not empty	
120	gripper contains unknown clip or slide(s)	
121	system not yet initialised	
122	clip not correct seated in 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	crash detection	

RFID error messages

These error messages may be generated in case an RFID device is connected to the Tango controller.

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

Piezo Z stage error messages

These error messages may be generated in case a Piezo Z stage device is connected to the Tango controller.

140	Piezo connect
141	Piezo timeout
142	Piezo address

Loader subsystem error messages

These error messages may be generated from Tango communication to any loader subsystem.

- 150 HL connect
- 151 HL timeout
- 152 HL cal X
- 153 HL cal Y
- 154 HL cal Z
- 155 HL insert
- 156 HL eject
- 157 HL get tray
- 158 HL put tray
- 159 HL protocol
- 160 HL hall tray detection
- 161 clamp electronic
- 162 no tray on stage but RFID read
- 163 escape position Z
- 164 HL tray alignment
- 165 tray on stage but no RFID

8.2. DLL Error Messages

0	no error
4001	internal error
4002	internal error
4003	function call with wrong LSID value or maximum of 8 open connections reached
4004	Unknown interface type (may appear with Connect...)
4005	Error while initializing interface
4006	No connection with controller (e.g. if SetPitch is called before Connect)
4007	Timeout while reading from interface
4008	Error during command transmission to Tango controller
4009	Command aborted (with SetAbortFlag)
4010	Command is not supported by Tango controller
4011	Manual Joystick mode switched on (may appear with SetJoystickOn/Off)
4012	No move command possible, because manual joystick enabled
4013	Closed Loop Controller Timeout (could not settle within target window)
4015	Limit switch activated in travel direction
4016	Repeated vector start!! (Closed Loop controller)
4017	Error while calibrating (Limit switch not correctly released)
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	-
4110	Function not configured
4111	-
4112	Limit switch active
4113	Function not executable, because encoder detected

9. 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)