TANGO-DLL

Dokumentation

MÄRZHÄUSER WETZLAR
GMBH u. Co. KG
Positioniersysteme

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

Inhaltsverzeichnis

1. Einführung

Die Tango-DLL (Programmierschnittstelle für die Steuerungen Tango DT und Tango PCI) soll Software-Entwicklern dabei helfen, Anwendungen zur Steuerung der 2/4- phasigen Schrittmotoren schnell und effektiv zu entwickeln, ohne sich mit hardware-naher Programmierung beschäftigen zu müssen. Die Tango-DLL unterstüzt sämtliche Befehle der Tango Steuerung.

1.1 Funktionsumfang

- Windows 32-bit DLL
- Unterstützt die Schrittmotorensteuerungen Tango DT und Tango PCI
- Ansteuerung über RS232, bzw. Virtual COM Port (PCI, USB)
- Unterstützt sämtliche Steuerungsbefehle
- Bis zu 4 Achsen

1.2 Systemanforderungen

Die Tango-DLL kann auf allen Windows-PCs ab Windows 98 verwendet werden.

1.3 Unterstütze Entwicklungsumgebungen

Tango-DLL wurde mit den folgenden Entwicklungs- und Laufzeitumgebungen getestet:

Microsoft Visual Basic
Microsoft Visual C++
National Instruments LabVIEW
Delphi 2007

Sie sollte kompatibel mit allen anderen Programmierumgebungen sein, die DLLs verwenden können.

(DLL = Dynamic Link Library, bezeichnet allgemein eine Dynamische Bibliothek. Eine Programmbibliothek bedeutet in der Programmierung eine Sammlung von Programmfunktionen für zusammengehörende Aufgaben. Bibliotheken sind im Unterschied zu Programmen keine eigenständig lauffähige Einheiten, sondern Hilfsmodule, die Programmen zur Verfügung gestellt werden.)

2. DLL-Schnittstelle

Hauptbestandteil der Tango-DLL ist die Datei Tango_DLL.dll. Diese Datei verwenden Sie bei der Entwicklung eigener Programme, um die Tango zu konfigurieren, Befehle zu senden, Ein-/ Ausgänge abzufragen etc..

2.1 Allgemeine Hinweise

Alle Funktionen sind mit einem 32-bit Integer als Rückgabewert deklariert. Eine 0 als Rückgabewert zeigt die fehlerfreie Ausführung der Funktion an, bei Fehlern (z.B. Timeouts) wird der entsprechende Fehlercode (siehe Fehlercodes) zurückgeliefert.

Die in der Dokumentation aufgeführten Beispiele verwenden ausschließlich „LSX_“-Befehle, in denen der erste Wert für die Tango-ID (LSID) steht. Diese ID wird gebraucht, um mehrere Steuerungen gleichzeitig anzusteuern. Da die „LSX_“-Befehle zurzeit nur eine Steuerung unterstützen, wird empfohlen, die „LS_“-Befehle zu verwenden. Hierbei entfällt in den Funktionsaufrufen der erste Wert für die Tango-ID. Es wird auch kein CreateLSID benötigt.

Beispiel:

„LS_“-Befehl:
pTango->MoveAbs(50.0, 50.0, 50.0, 10.0, true);

„LSX_“-Befehl:
pTango->MoveAbs(**1**, 50.0, 50.0, 50.0, 10.0, true);
// der erste Wert ist die LSID, welcher bei „LS_“-Befehlen entfällt

Bei Funktionen wie LSX_MoveAbs werden immer die Werte für 4 Achsen übergeben.
Handelt es sich um eine Steuerung mit 1-3 Achsen, werden die Werte für die nicht vorhandenen Achsen ignoriert, sie können auf 0 gesetzt werden.

2.2 Einbindung in Visual C++

Für Visual C++ wurde eine Kapselung der Tango_DLL.dll erstellt. Die Klasse CTango lädt die DLL und alle Zeiger auf Funktionsaufrufe dynamisch. Den Methoden des Tango-Objekts ist kein „LS_“ oder „LSX_“ vorangestellt.
(Beispiel: pTango->Calibrate() statt LS_Calibrate).
Von der Klasse CTango sollte nur eine Instanz erstellt werden, da mit der Tango-DLL momentan nicht mehrere Steuerungen gleichzeitig gesteuert werden können.
Die Dateinen Tango.h und Tango.cpp befinden sich auf der Cd im Verzeichnis Software\API\Examples\Visual_C\SourceCode.

Benötigte Dateien: Tango_DLL.dll, Tango.h und Tango.cpp

Visual C++-Beispiel für die Ansteuerung einer Tango:

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

pTango->ConnectSimple(1, „COM3“, 57600, true);
pTango->MoveAbs(30, 50, 70, 0, true);
pTango->Disconnect();
delete pTango;
```

2.3 Einbindung in Visual Basic

Um die Funktionen der Tango-DLL verwenden zu können, muss die Datei Tango.vb dem Projekt hinzugefügt werden. Die Datei Tango.vb befindet sich auf der Cd im Verzeichnis Software\API\Examples\Visual_Basic\SourceCode.

Benötigte Dateien: Tango_DLL.dll und Tango.vb

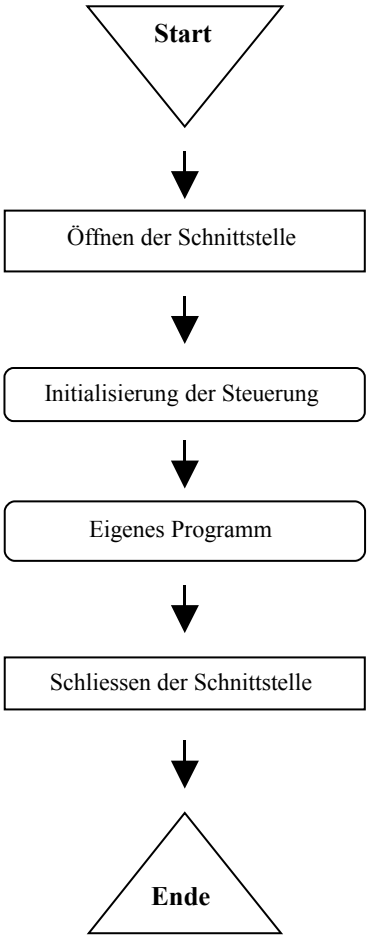
Visual Basic-Beispiel für die Ansteuerung einer Tango:

```
Dim Rueckgabewert As Integer
Dim Rueckgabewert2 As Integer
Dim Rueckgabewert3 As Integer

...
Rueckgabewert = LS_ConnectSimple(1, „COM3“, 57600, 1)
Rueckgabewert2 = LS_MoveAbs(30, 50, 70, 0, 1)
Rueckgabewert3 = LS_Disconnect
...
```

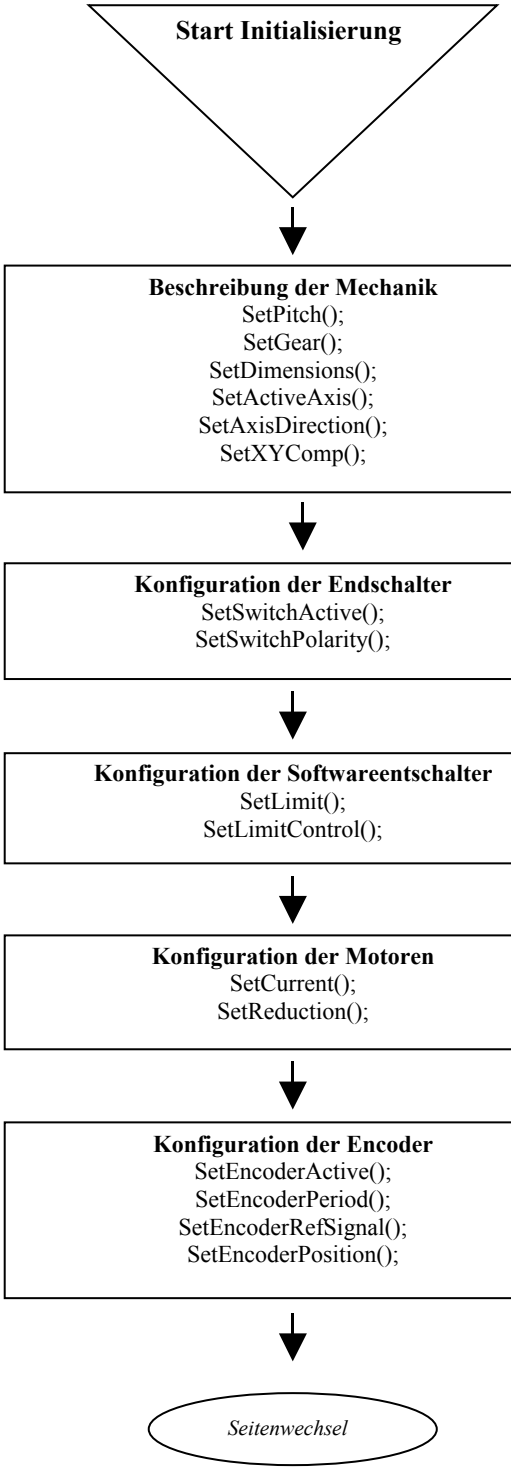
3. Hinweise zum Aufbau eigener Programme bei der Programmierung der Steuerungen über die DLL

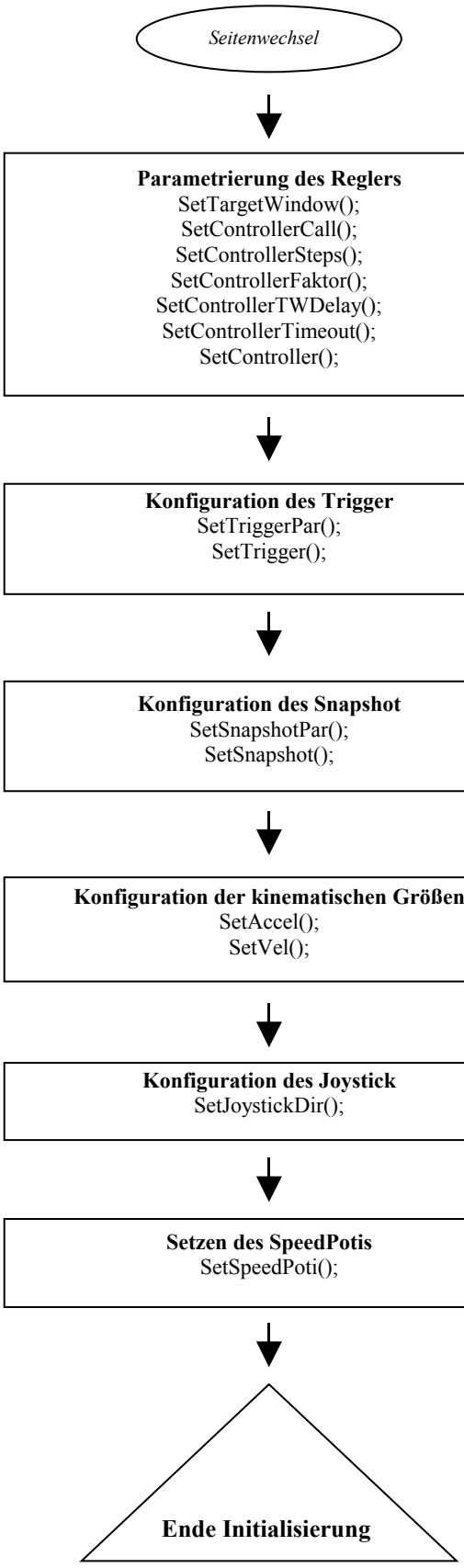
Die folgende Abbildung zeigt den Programmflussplan nach dem die Programme zur Steuerung von Schrittmotoren aufgebaut sein sollten. Die verwendeten Funktionen sind in der Tango-DLL Dokumentation aufgelistet und werden dort genauer beschrieben.



3.1 Initialisierung der Steuerung

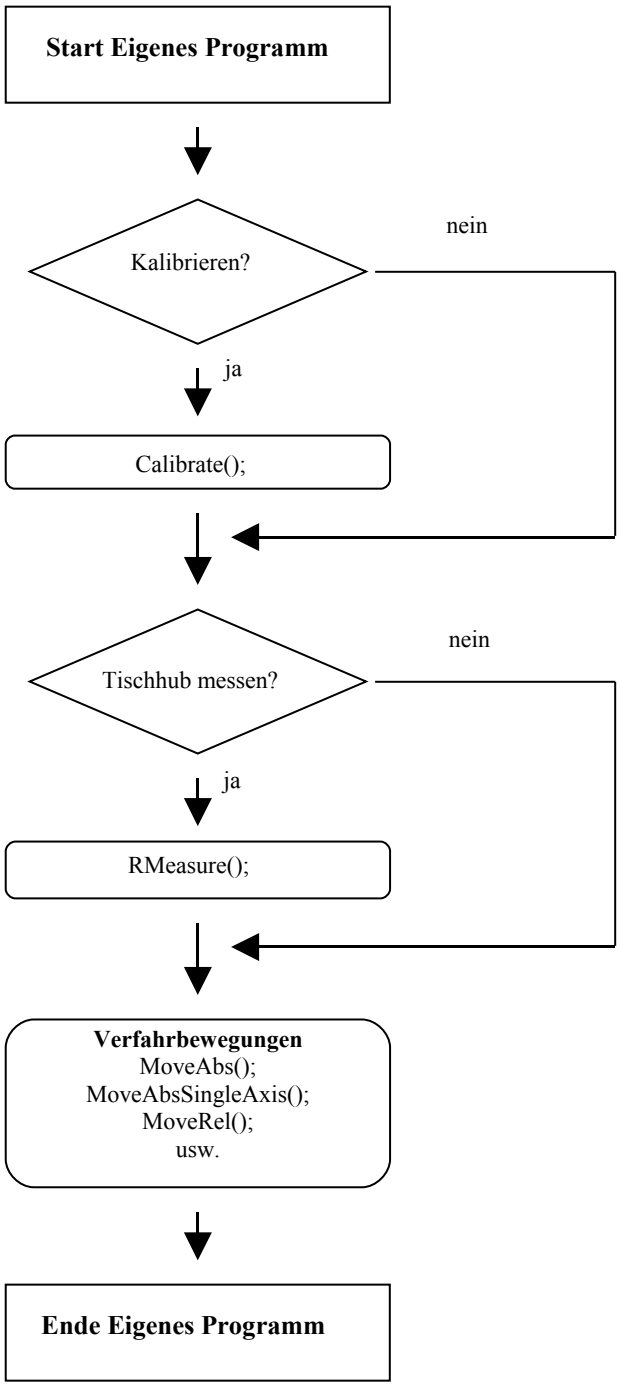
Vor dem Start des eigenen Programmteils sind bei der Initialisierung der verwendeten Tango die im folgenden beschriebenen Grundeinstellungen der Steuerung vorzunehmen um einen fehlerfreien Betrieb zu gewährleisten.




Seitenwechsel
Parametrierung des Reglers
SetTargetWindow();
SetControllerCall();
SetControllerSteps();
SetControllerFaktor();
SetControllerTWDelay();
SetControllerTimeout();
SetController();
Konfiguration des Trigger
SetTriggerPar();
SetTrigger();
Konfiguration des Snapshot
SetSnapshotPar();
SetSnapshot();
Konfiguration der kinematischen Größen
SetAccel();
SetVel();
Konfiguration des Joystick
SetJoystickDir();
Setzen des SpeedPotis
SetSpeedPoti();
Ende Initialisierung

3.2 Eigener Programmteil

Im eigenen Programmteil kann der Anwender die gewünschte Funktionalität der Steuerung programmieren. Dazu zählen das Ausführen von Verfahrbewegungen in Abhänigkeit der zustände von digitalen I/Os ebenso wie das Setzen von Triggersignalen in Abhängigkeit von der Position, usw..



4. Funktionen

4.1 Inhaltsverzeichnis der Befehle

DLL-Konfiguration/Schnittstelle:

Steuerungs-Info:

Statusabfragen:

Einstellungen:

Fahrbefehle und Positionsverwaltung:

Joystick und Handrad:

Bedienpult mit Trackball und Joyspeed-Tasten (kundenspez. Anwendung):

Endschalter (Hardware und Software):

Digitale und analoge Ein- und Ausgänge:

Geber-Einstellungen:

Reglereinstellungen:

Befehl	Kurzbeschreibung	Seite
ClearCtrFastMoveCounter	Anzahl ausgeführter FastMove-Funktionen auf 0 setzen	60
GetController	Regler-Modus auslesen	60
GetControllerCall	Liefert Regleraufrufzeit	61
GetControllerFactor	Einstellung des Reglerfaktors auslesen	61
GetControllerSteps	Regler-Schritte auslesen	62
GetControllerTimeout	Liefert die Einstellung vom Regler-Überwachungs-Timeout	62
GetControllerTWDelay	Reglerverzögerung auslesen	63
GetCtrFastMove	Liest, ob die FastMove-Funktion ein- oder ausgeschaltet ist	63
GetCtrFastMoveCounter	Anzahl ausgeführter FastMove-Funktionen auslesen	63
GetTargetWindow	Liest die Zielfenster aller Achsen	64
SetController	Regler-Modus einstellen	60
SetControllerCall	Regleraufrufzeit einstellen	61
SetControllerFactor	Reglerfaktor einstellen	61
SetControllerSteps	Regler-Schritte einstellen	62
SetControllerTimeout	Regler-Überwachungs-Timeout einstellen	62
SetControllerTWDelay	Reglerverzögerung setzen	63
SetCtrFastMoveOff	FastMove-Funktion ausschalten	64
SetCtrFastMoveOn	FastMove-Funktion einschalten	64
SetTargetWindow	Reglerzielfenster einstellen	64

Trigger-Ausgang:

Befehl	Kurzbeschreibung	Seite
GetTrigCount	Triggerzählerstand auslesen	65
GetTrigger	Triggereinstellung auslesen	65
GetTriggerPar	Trigger-Parameter auslesen	66
SetTrigCount	Triggerzählerstand setzen	65
SetTrigger	Trigger ein-/ausschalten	65
SetTriggerPar	Trigger-Parameter einstellen	66

Snapshot-Eingang:

Befehl	Kurzbeschreibung	Seite
GetSnapshot	Liefert den aktuellen Snapshot-Zustand	67
GetSnapshotCount	Snapshot-Zähler	67
GetSnapshotFilter	Eingangsfilter auslesen	67
GetSnapshotPar	Snapshot-Parameter auslesen	68
GetSnapshotPos	Snapshot-Position auslesen	68
GetSnapshotPosArray	Snapshot-Position aus Array auslesen	69
SetSnapshot	Snapshot ein-/ausschalten	67
SetSnapshotFilter	Eingangsfilter setzen	68
SetSnapshotPar	Snapshot-Parameter festlegen	68

4.2 DLL-Konfiguration/Schnittstelle

LSX_ConnectSimple	
Beschreibung:	Mit Tango verbinden. Ohne Verbindungsaufbau ist keine Verbindung möglich.
C++:	int LSX_ConnectSimple(int lLSID, int lAnInterfaceType, char *pcAComName, int lABaudRate, BOOL bAShowProt);
Parameter:	***AnInterfaceType***: Schnittstellentyp = 1 (immer 1 für RS232, PCI und USB) ***AComName***: Name der COMSchnittstelle, z.B. ´´COM2´´ ***ABaudRate***: z.B. 57600 Baud (nur wichtig bei RS232) ***AShowProt***: Bestimmt, ob das Schnittstellenprotokoll angezeigt werden soll
Beispiel:	pTango->ConnectSimple(1, 1, ´´COM2´´, 57600, true);

LSX_CreateLSID	
Beschreibung:	Erzeugt eine TangoIDNummer. Diese wird als zusätzlicher Parameter bei den Tango-DLL Befehlen verwendet, um aus mehreren angeschlossenen Tangos die Tango zu wählen, auf die sich der Befehl beziehen soll.
C++:	int LSX_CreateLSID(int *plLSID);
Parameter:	***LSID***: enthält nach Aufruf von CreateLSID eine neue TangoIDNummer, die dann für Connect, Verfahrbefehle und andere Befehle verwendet werden kann
Beispiel:	int Tango1, Tango2; pTango->CreateLSID(&Tango1); *// ID für erste Tango anlegen* pTango->CreateLSID(&Tango2); *// ID für zweite Tango anlegen*

LSX_Disconnect	
Beschreibung:	Verbindung mit Tango trennen. Nach Aufruf dieser Funktion können keine Befehle mehr zur Tango gesendet werden. Die Funktion sollte kurz vor Beendigung des Programms aufgerufen werden.
C++:	int LSX_Disconnect(int lLSID);
Parameter:	-
Beispiel:	pTango->Disconnect(1);

LSX_EnableCommandRetry	
Beschreibung:	Mit dieser Funktion kann das wiederholte Senden von Kommandos im Falle von Fehlern ein/ausgeschaltet werden (Standardmäßig ist dieses eingeschaltet).
C++:	int LSX_EnableCommandRetry (int lLSID, BOOL bAValue);
Parameter:	***AValue***: true → bei Fehlern wiederholt die Tango-DLL das Senden bestimmter Kommandos (insbesondere bei WaitForAxisStop) false → wiederholtes Senden abschalten
Beispiel:	pTango->EnableCommandRetry(1, false);

LSX_FlushBuffer	
Beschreibung:	KommunikationsEingabepuffer löschen. Kann in FehlerSituationen verwendet werden, um nicht mehr benötigte Rückmeldungungen aus dem Eingabepuffer zu entfernen.
C++:	int LSX_FlushBuffer (int lLSID, int lAValue);
Parameter:	***AValue***: wird momentan nicht verwendet, kann = 0 gesetzt werden
Beispiel:	pTango->FlushBuffer(1, 0);

LSX_FreeLSID	
Beschreibung:	Gibt eine erzeugte TangoIDNummer wieder frei. Diese wird als zusätzlicher Parameter bei den TangoDLL Befehlen verwendet, um aus mehreren angeschlossenen Tangos die Tango zu wählen, auf die sich der Befehl beziehen soll. FreeLSID sollte erst nach Disconnect aufgerufen werden.
C++:	int LSX_FreeLSID(int lLSID);
Parameter:	***LSID***: freizugebende TangoIDNummer; diese darf nach FreeLSID nicht mehr verwendet werden
Beispiel:	int Tango1; pTango->CreateLSID(&Tango1); pTango->ConnectSimple(Tango1, …); pTango->Disconnect(Tango1); pTango-> FreeLSID(Tango1);

LSX_SendString	
Beschreibung:	String an Tango senden.
C++:	int LSX_SendString (int lLSID, char *pcStr, char *pcRet, int lMaxLen, BOOL bReadLine, int lTimeOut);
Parameter:	***Str*** → nullterminierter String, der an die Steuerung gesendet werden soll. Der String muss mit einem cariage return (\r) beendet werden. ***Ret*** → Puffer, der die Rückmeldung der Tango enthält, falls ReadLine = true oder auch NULL, falls ReadLine = false; ***MaxLen*** → maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen ***ReadLine*** → true = Rückmeldung der Tango lesen false = nicht auf Rückmeldung warten ***TimeOut*** → maximale Wartezeit auf Rückmeldung [ms]
Beispiel:	pTango->SendString(1, ´´?version\r´´, pcVer, 256, true, 1000); *// Versionsnummer lesen, 1 Sekunde Timeout* pTango->SendString(1, ´´!baud 115200\r´´, NULL, 0, false, 0); *// maximale Baudrate für RS232 einstellen*

LSX_SendStringPosCmd	
Beschreibung:	Verfahrbefehl, welcher Rückmeldung erwartet, als String an Tango senden.
C++:	int LSX_SendStringPosCmd (int lLSID, char *pcStr, char *pcRet, int lMaxLen, BOOL bReadLine, int lTimeOut);
Parameter:	***Str*** → nullterminierter String, der an die Steuerung gesendet werden soll ***Ret*** → Puffer, der die Rückmeldung der Tango enthält, falls ReadLine = true oder auch NULL, falls ReadLine = false; ***MaxLen*** → maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen ***ReadLine*** → true = Rückmeldung der Tango lesen false = nicht auf Rückmeldung warten ***TimeOut*** → maximale Wartezeit auf Rückmeldung [ms]
Beispiel:	pTango->SendStringPosCmd(1, ´´!moa 1 2\r´´, pcRet , 256, true, 10000);

LSX_SetAbortFlag	
Beschreibung:	Flag setzen, damit die Kommunikation mit der Tango abgebrochen wird. Eine Funktion, die bei Aufruf von LSX_SetAbortFlag noch auf eine Rückmeldung der Steuerung wartet (z.B. Verfahrbefehle), kehrt dann mit einer Fehlermeldung zurück. Die Verwendung dieser Funktion ist insbesondere bei Programmen mit Botschaftsbehandlungsroutinen oder mehreren Threads sinnvoll, falls z.B. schnell eine Verfahrbewegung abgebrochen werden soll.
C++:	int LSX_SetAbortFlag (int lLSID);
Parameter:	-
Beispiel:	pTango->SetAbortFlag(1); pTango->StopAxes(1); *// bricht die Kommunikation mit der Tango ab und sendet das Kommando zum stoppen aller Achsen*

LSX_SetShowProt	
Beschreibung:	Schnittstellenprotokoll ein-/ausschalten.
C++:	int LSX_SetShowProt (int lLSID, BOOL bShowProt);
Parameter:	***ShowProt***: Gibt an, ob das Fenster ´´SchnittstellenProtokoll´´ gezeigt werden soll
Beispiel:	pTango->SetShowProt(1, true); *// SchnittstellenProtokoll für Tango1 zeigen, falls nicht bereits sichtbar*

4.3 Steuerungs-Info

LSX_GetSerialNr	
Beschreibung:	Liest die Seriennummer der Steuerung aus.
C++:	int LSX_GetSerialNr (int lLSID, char *pcSerialNr, int lMaxLen);
Parameter:	***SerialNr***: Zeiger auf einen Puffer, in dem die Seriennummer zurückgegeben wird ***MaxLen***: Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen
Beispiel:	pTango->GetSerialNr(1, pcSerialNr, 256);

LSX_GetVersionStr	
Beschreibung:	Liefert die aktuelle Versionsnummer der Firmware zurück.
C++:	int LSX_GetVersionStr (int lLSID, char *pcVers, int lMaxLen);
Parameter:	***Vers***: Zeiger auf einen Puffer, in dem der VersionsString zurückgegeben wird ***MaxLen***: Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen
Beispiel:	pTango->GetVersionStr(1, pcVers, 64); *// Versionsnummer auslesen*

LSX_GetVersionStrDet	
Beschreibung:	Liest die detaillierte Konfiguration der Tango aus.
C++:	int LSX_GetVersionStrDet (int lLSID, char *pcVersDet, int lMaxLen);
Parameter:	***VersDet:*** Zeiger auf einen Puffer, in dem der String zurückgegeben wird ***MaxLen:*** Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen
Beispiel:	pTango->GetVersionStrDet(1, pcVersDet, 16); *// detaillierte Konfiguration auslesen*

LSX_GetVersionStrInfo	
Beschreibung:	Liefert interne Versionsinformationen.
C++:	int LSX_GetVersionStrInfo (int lLSID, char *pcVersInfo, int lMaxLen);
Parameter:	***VersInfo:*** Zeiger auf einen Puffer ***MaxLen:*** maximale Anzahl von Zeichen, die in den Puffer kopiert werden
Beispiel:	pTango->GetVersionStrInfo(1, pcVersInfo, 16);

4.4 Statusabfragen

LSX_GetError	
Beschreibung:	Liefert die aktuelle Fehlernummer.
C++:	int LSX_GetError (int lLSID, int *plErrorCode);
Parameter:	***ErrorCode***: Fehlernummer
Beispiel:	pTango->GetError(1, &ErrorCode);

LSX_GetPos	
Beschreibung:	Abfrage der aktuellen Positionen aller Achsen.
C++:	int LSX_GetPos (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z, A***: Positionswerte
Beispiel:	pTango->GetPos(1, &X, &Y, &Z, &A);

LSX_GetPosEx	
Beschreibung:	Abfrage der aktuellen Geber bzw. Positionswerte aller Achsen. Für nicht vorhandene Achsen wird der Wert 0.0 zurückgeliefert.
C++:	int LSX_GetPosEx (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA, BOOL bEncoder);
Parameter:	***X, Y, Z, A***: Positionswerte ***Encoder*** = true → Geberwerte liefern, falls Geber angeschlossen = false → Positionswerte liefern
Beispiel:	pTango->GetPosEx(1, &X, &Y, &Z, &A, true);

LSX_GetPosSingleAxis	
Beschreibung:	Abfrage der aktuellen Position einer einzelnen Achse. Wenn die Achse nicht vorhanden ist, wird der Wert 0.0 zurückgeliefert.
C++:	int LSX_GetPosSingleAxis (int lLSID, int lAxis, double *pdPos);
Parameter:	***Axis***: Achse, deren Positionswert abgefragt werden soll, X, Y, Z und A nummeriert von 1 bis 4 ***Pos***: Positionswert
Beispiel:	pTango->GetPosSingleAxis(1, 2, &Pos); *// Position der Y-Achse auslesen*

LSX_GetStatus	
Beschreibung:	Liefert den aktuellen Zustand der Steuerung.
C++:	int LSX_GetStatus (int lLSID, char *pcStat, int lMaxLen);
Parameter:	***Stat***: Zeiger auf einen Puffer, in dem der Statusstring zurückgegeben wird ***MaxLen***: Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen
Beispiel:	pTango->GetStatus(1, &Stat, 16);

LSX_GetStatusAxis	
Beschreibung:	Liefert den aktuellen Zustand der einzelnen Achsen.
C++:	int LSX_GetStatusAxis (int lLSID, char *pcStatusAxisStr, int lMaxLen);
Parameter:	***StatusAxisStr***: Zeiger auf einen Puffer, in dem der Statusstring zurückgegeben wird ***MaxLen***: maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen z.B.: @ M -- J -- C -- S -- A -- D -- U T @ = Achse steht M = Achse ist in Bewegung (Motion) = Achse ist nicht freigegeben J = Joystick eingeschaltet C = Achse ist in Regelung A = Rückmeldung nach dem Kalibrieren E = Fehler beim Kalibrieren (Endschalter nicht korrekt freigefahren) D = Rückmeldung nach dem Tischhubmessen U = Einrichtbetrieb T = Timeout
Beispiel:	pTango->GetStatusAxis(1, &StatusAxisStr, 16);

LSX_GetStatusLimit	
Beschreibung:	Liefert den aktuellen Zustand der Softwaregrenzen jeder einzelnen Achse.
C++:	int LSX_GetStatusLimit (int lLSID, char *pcLimit, int lMaxLen);
Parameter:	***Limit***: Zeiger auf einen Puffer, in dem der Zustand der Achsen zurückgegeben wird. z.B.: AA A DD LL L L A = Achse wurde kalibriert D = Tischhub wurde gemessen L = SoftwareLimit wurde gesetzt = SoftwareGrenze wurde nicht verändert ***MaxLen***: maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen
Beispiel:	pTango->GetStatusLimit(1, &Limit, 32);

LSX_SetAutoStatus	
Beschreibung:	AutoStatus ein-/ausschalten. Hinweis: der AutoStatusModus sollte normalerweise nicht verändert werden, da die Tango-DLL bei Verfahrbefehlen etc. den richtigen Modus einstellt, eine Änderung auf 0 oder 2 könnte zu Fehlern führen.
C++:	int LSX_SetAutoStatus (int lLSID, int lValue);
Parameter:	***Value***: AutoStatusModus: 0 → es wird kein Status von der Steuerung gesendet 1 → es werden automatisch ´´Position erreicht´´ Meldungen von der Steuerung gesendet 2 → es werden automatisch ´´Position erreicht´´ und Status Meldungen von der Steuerung gesendet 3 → es gibt bei ´´Position erreicht´´ nur ein Carriage Return zurück
Beispiel:	pTango->SetAutoStatus(1, 1);

4.5 Einstellungen

LSX_GetAccel	
Beschreibung:	Beschleunigung abfragen.
C++:	int LSX_GetAccel (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z, A***: Beschleunigungswerte [m/s²]
Beispiel:	pTango->GetAccel(1, &X, &Y, &Z, &A);

LSX_SetAccel	
Beschreibung:	Beschleunigung einstellen.
C++:	int LSX_SetAccel (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	**X, Y, Z, A**: 0.01 - 20.00 [m/s²]
Beispiel:	pTango->SetAccel(1, 1.0, 1.5, 0, 0);

LSX_GetActiveAxes	
Beschreibung:	Liefert den Status der Achsenfreigabe.
C++:	int LSX_GetActiveAxes (int lLSID, int *plFlags);
Parameter:	***Flags***: 32BitInteger, welcher nach Aufruf der Funktion in den Bits 04 die BitMaske enthält Bit 0 = 1 → XAchse freigeschaltet Bit 2 = 0 → ZAchse nicht freigeschaltet
Beispiel:	pTango->GetActiveAxes(1, &Flags);

LSX_SetActiveAxes	
Beschreibung:	Achsenfreigabe einstellen.
C++:	int LSX_SetActiveAxes (int lLSID, int lFlags);
Parameter:	***Flags***: BitMaske Bit 0 = 1 → XAchse freigeschaltet Bit 2 = 0 → ZAchse nicht freigeschaltet
Beispiel:	pTango->SetActiveAxes(1, 3); *// X und YAchse freigeben (Bits 0 und 1 gesetzt),* *ZAchse nicht freigegeben (Bit 2 = 0)*

LSX_GetAxisDirection	
Beschreibung:	Drehrichtung der Achsen abfragen.
C++:	int LSX_GetAxisDirection (int lLSID, int *plXD, int *plYD, int *plZD, int *plAD);
Parameter:	***XD, YD, ZD, AD***: 32BitIntegers 0 → normale Drehrichtung 1 → umgekehrte Drehrichtung
Beispiel:	pTango->GetAxisDirection(1, &XD, &YD,&ZD,&AD);

LSX_SetAxisDirection	
Beschreibung:	Drehrichtung der Achsen einstellen.
C++:	int LSX_SetAxisDirection (int lLSID, int lXD, int lYD, int lZD, int lAD);
Parameter:	***XD, YD, ZD, AD***: 32BitIntegers 0 → normale Drehrichtung 1 → Drehrichtung umkehren
Beispiel:	pTango->SetAxisDirection(1, 1, 0, 0, 0); *// Drehrichtung der XAchse umkehren*

LSX_GetCalibBackSpeed	
Beschreibung:	Liest die Umdrehungsgeschwindigkeit, mit der die Achsen beim Kalibrieren aus dem Endschalter herausfahren. Die Geschwindigkeit entspricht dem ausgegebenen Wert * 0.01 U/s.
C++:	int LSX_GetCalibBackSpeed (int lLSID, int *plSpeed);
Parameter:	***Speed***: Geschwindigkeitswert
Beispiel:	pTango->GetCalibBackSpeed(1, &lSpeed);

LSX_SetCalibBackSpeed	
Beschreibung:	Setzt die Umdrehungsgeschwindigkeit, mit der die Achsen beim Kalibrieren aus dem Endschalter herausfahren. Die Geschwindigkeit entspricht dem angegebenen Wert * 0.01 U/s.
C++:	int LSX_SetCalibBackSpeed (int lLSID, int lSpeed);
Parameter:	***Speed***: Geschwindigkeitswert, Wertebereich 1 bis 100
Beispiel:	pTango->SetCalibBackSpeed(1, 10); *// die Endschalter werden bei der Kalibrierung nach dem Anfahren mit 0.1 U/s verlassen*

LSX_GetCalibOffset	
Beschreibung:	Nullpunkts-Verschiebung der Achsen abfragen.
C++:	int LSX_GetCalibOffset (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA)
Parameter:	***X, Y, Z, A***: Nullpunkts-Verschiebung, abhängig von der Dimension
Beispiel:	pTango->GetCalibOffset(1, &X, &Y, &Z, &A);

LSX_SetCalibOffset	
Beschreibung:	Nullpunkts-Verschiebung der Achsen setzen. Der Nullpunkt der Achsen wird um diesen Betrag vor den Endschalter verlegt.
C++:	int LSX_SetCalibOffset (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A***: typisch 0 – 5 [mm]
Beispiel:	pTango->SetCalibOffset(1, 1, 1, 1, 1); *// die Achsen X, Y, Z und A werden beim Kalibrieren jeweils 1 mm (bei Dimension 2 2 2 2) vom Nullendschalter in Richtung Tischmitte verfahren und dann die Position Null gesetzt (Softwaregrenze)*

LSX_GetCalibrateDir	
Beschreibung:	Kalibrier-Richtung abfragen.
C++:	int LSX_GetCalibrateDir (int lLSID, int *plXD, int *plYD, int *plZD, int *plAD);
Parameter:	***XD, YD, ZD, AD***: 32BitIntegers 0 → normale Kalibrier-Richtung 1 → umgekehrte Kalibrier-Richtung
Beispiel:	pTango->GetCalibrateDir(1, &XD, &YD,&ZD,&AD);

LSX_SetCalibrateDir	
Beschreibung:	Kalibrier-Richtung setzen.
C++:	int LSX_SetCalibrateDir (int lLSID, int lXD, int lYD, int lZD, int lAD);
Parameter:	***XD, YD, ZD, AD***: 32BitIntegers 0 → normale Kalibrier-Richtung 1 → umgekehrte Kalibrier-Richtung
Beispiel:	pTango->(1, 1, 1, 0, 0);

LSX_GetCurrentDelay	
Beschreibung:	Liefert die Zeitverzögerung, nach der die Stromabsenkung aktiviert wird.
C++:	int LSX_GetCurrentDelay (int lLSID, int *plX, int *plY, int *plZ, int *plA);
Parameter:	***X, Y, Z, A***: Zeitverzögerung [ms]
Beispiel:	pTango->GetCurrentDelay(1, &X, &Y,&Z,&A);

LSX_SetCurrentDelay	
Beschreibung:	Setzt die Zeitverzögerung, nach der die Stromabsenkung aktiviert wird.
C++:	int LSX_SetCurrentDelay (int lLSID, int lX, int lY, int lZ, int lA);
Parameter:	***X, Y, Z, A***: 010000 [ms]
Beispiel:	pTango->SetCurrentDelay(1, 100, 300, 1000, 0);

LSX_GetDimensions	
Beschreibung:	Liefert die aktuell eingestellte Maßeinheit der Achsen.
C++:	int LSX_GetDimensions (int lLSID, int *plXD, int *plYD, int *plZD, int *plAD);
Parameter:	***XD, YD, ZD, AD***: Maßeinheiten 0 → Microsteps 1 → µm 2 → mm (Voreinstellung) 3 → Grad 4 → Umdrehungen 5 → cm 6 → m 7 → Zoll 8→ mil (1/1000 Zoll)
Beispiel:	pTango->GetDimensions(1, &XD, &YD,&ZD,&AD);

LSX_SetDimensions	
Beschreibung:	Maßeinheiten der Achsen setzen.
C++:	int LSX_SetDimensions (int lLSID, int lXD, int lYD, int lZD, int lAD);
Parameter:	***XD, YD, ZD, AD***: Maßeinheiten 0 → Microsteps 1 → µm 2 → mm (Voreinstellung) 3 → Grad 4 → Umdrehungen 5 → cm 6 → m 7 → Zoll 8→ mil (1/1000 Zoll)
Beispiel:	pTango->SetDimensions(1, 3, 2, 2, 1); *// X-Achse in Grad, Y- und Z-Achse in mm und A-Achse in µm*

LSX_GetGear	
Beschreibung:	Getriebeübersetzung abfragen.
C++:	int LSX_GetGear (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z, A***: Getriebeübersetzungswerte
Beispiel:	pTango->GetGear(1, &X, &Y, &Z, &A);

LSX_SetGear	
Beschreibung:	Getriebeübersetzung einstellen.
C++:	int LSX_SetGear (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A:*** 0.01 - 1000
Beispiel:	pTango->SetGear(1, 4.0, 2.0, 1.0, 1.0); *// GetriebeÜbersetzungen ¼ bei Z, ½ bei Y und 1/1 bei Z und A werden programmiert*

LSX_GetMotorCurrent	
Beschreibung:	Motorströme abfragen.
C++:	int LSX_GetMotorCurrent (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z, A***: Motorströme [A]
Beispiel:	pTango->GetMotorCurrent(1, &X, &Y, &Z, &A);

LSX_SetMotorCurrent	
Beschreibung:	Motorströme einstellen.
C++:	int LSX_SetMotorCurrent (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A***: Motorstrom X, Y, Z und AAchse [A]
Beispiel:	pTango->SetMotorCurrent(1, 1.5, 1.5, 1.0, 1.0); *// Motorstrom X- und Y-Achse 1.5 Ampere; Z- und A-Achse 1.0 Ampere*

LSX_GetPitch	
Beschreibung:	Liefert die Spindelsteigungen.
C++:	int LSX_GetPitch (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z, A***: Spindelsteigungen [mm]
Beispiel:	pTango->GetPitch(1, &X, &Y, &Z, &A);

LSX_SetPitch	
Beschreibung:	Spindelsteigungen einstellen.
C++:	int LSX_SetPitch (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A***: 0.001 - 68 [mm]
Beispiel:	pTango->SetPitch(1, 4, 4, 4, 4); *// Spindelsteigungen aller Achsen auf 4 mm setzen*

LSX_GetPowerAmplifier	
Beschreibung:	Gibt an, ob die Endstufen ein- oder ausgeschaltet sind.
C++:	int LSX_GetPowerAmplifier (int lLSID, BOOL *pbAmplifier);
Parameter:	***Amplifier***: true → die Endstufen sind eingeschaltet false → die Endstufen sind ausgeschaltet
Beispiel:	pTango->GetPowerAmplifier(1, &Amplifier);

LSX_SetPowerAmplifier	
Beschreibung:	Endstufen ein-/ausschalten.
C++:	int LSX_SetPowerAmplifier (int lLSID, BOOL bAmplifier);
Parameter:	***Amplifier***: true → Endstufen einschalten false → Endstufen ausschalten
Beispiel:	pTango->SetPowerAmplifier(1, true); *// die Endstufen einschalten*

LSX_GetReduction	
Beschreibung:	Stromabsenkung abfragen.
C++:	int LSX_GetReduction (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA)
Parameter:	***X, Y, Z, A***: Stromabsenkung (Wertebereich von 0 bis 1)
Beispiel:	pTango->GetReduction(1, &X, &Y, &Z, &A);

LSX_SetReduction	
Beschreibung:	Stromabsenkungsfaktor einstellen.
C++:	int LSX_SetReduction (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A:*** 0 - 1.0
Beispiel:	pTango->SetReduction(1, 0.1, 0.7, 0.5, 0.5); *// Ruhestrom XAchse = 0.1*Nennstrom, YAchse = 0.7*Nennstrom,* *Z- und A-Achse = 0,5*Nennstrom*

LSX_GetRMOffset	
Beschreibung:	Verschiebung der Achsen-Endpositionen abfragen.
C++:	int LSX_GetRMOffset (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z, A***: Endpositions-Verschiebung, abhängig von der Maßeinheit
Beispiel:	pTango->GetRMOffset(1, &X, &Y, &Z, &A);

LSX_SetRMOffset	
Beschreibung:	Verschiebung der Achsen-Endpositionen setzen. Der Endpunkt der Achsen wird um diesen Betrag vor den Endschalter verlegt.
C++:	int LSX_SetRMOffset (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A***: typisch 0 – 5 [mm]
Beispiel:	pTango->SetRMOffset(1, 1, 1, 1, 1); *// die Endpositionen der Achsen werden jeweils um 1mm in Richtung Tischmitte versetzt (bei Dimension 2 2 2 2).*

LSX_GetSpeedPoti	
Beschreibung:	Gibt an, ob das Speedpotentiometer ein- oder ausgeschaltet ist.
C++:	int LSX_GetSpeedPoti (int lLSID, BOOL *pbSpePoti);
Parameter:	Das SpePoti-Flag gibt an, ob das Potentiometer ein- oder ausgeschaltet ist
Beispiel:	pTango->(1, &flag);

LSX_SetSpeedPoti	
Beschreibung:	Speedpotentiometer ein-/ausschalten.
C++:	int LSX_SetSpeedPoti (int lLSID, BOOL bSpeedPoti);
Parameter:	***SpeedPoti*** = false → die vorgegebene Geschwindigkeit (vel) wird als Verfahrgeschwindigkeit genutzt = true → die vorgegebene Geschwindigkeit (vel), wird in Abhängigkeit von der Stellung des Potentiometers, prozentual genutzt
Beispiel:	pTango->SetSpeedPoti(1, true); *// das Potentiometer ist eingeschaltet*

LSX_GetStopAccel	
Beschreibung:	Liefert die Bremsbeschleunigung für Fehlerzustände.
C++:	int LSX_GetStopAccel (int lLSID, double *pdXD, double *pdYD, double *pdZD, double *pdAD);
Parameter:	***XD, YD, ZD, AD***: Bremsbeschleunigungswerte [m/s²]
Beispiel:	pTango->GetStopAccel(1, &XD, &YD, &ZD, &AD);

LSX_SetStopAccel	
Beschreibung:	Wird ein Enschalter betätigt, wird diese Beschleunigung für das Abbremsen der Achse verwendet. Sollte die mit LSX_SetAccel gesetzte Beschleunigung größer sein, wird diese verwendet.
C++:	int LSX_SetStopAccel (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A***: Bremsbeschleunigung, Wertebereich 0.01 bis 20 [m/s²]
Beispiel:	pTango->SetStopAccel(1, 15.0, 15.0, 15.0, 15.0);

LSX_GetStopPolarity	
Beschreibung:	Abfrage der Polarität des Stopeingangs.
C++:	int LSX_GetStopPolarity (int lLSID, BOOL *pbHighActiv);
Parameter:	***HighActiv***: true → Stopeingang high-aktiv false → Stopeingang low-aktiv
Beispiel:	pTango->GetStopPolarity(1, &HighActiv);

LSX_SetStopPolarity	
Beschreibung:	Polarität am Stopeingang einstellen. Da der Stopeingang einen Pull Up nach 5V hat, muß man bei einem Schließer low-aktiv und bei einem Öffner high-aktiv einstellen.
C++:	int LSX_SetStopPolarity (int lLSID, BOOL bHighActiv);
Parameter:	***HighActiv***: true → Stopeingang high-aktiv false → Stopeingang low-aktiv
Beispiel:	pTango->SetStopPolarity(1, false); *// der Stopeingang ist low-aktiv*

LSX_GetVel	
Beschreibung:	Geschwindigkeiten aller Achsen abfragen.
C++:	int LSX_GetVel (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z, A***: Geschwindigkeitswerte [U/s]
Beispiel:	pTango->GetVel(1, &X, &Y, &Z, &A);

LSX_SetVel	
Beschreibung:	Geschwindigkeiten aller Achsen einstellen.
C++:	int LSX_SetVel (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A***: 0 - maximale Geschwindigkeit [U/s]
Beispiel:	pTango->SetVel(1, 1.0, 15.0, 0, 0);

LSX_GetVelFac	
Beschreibung:	Geschwindigkeitsuntersetzung abfragen.
C++:	int LSX_GetVelFac (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z, A***: Geschwindigkeitsuntersetzungswerte
Beispiel:	pTango->GetVelFac(1, &X, &Y, &Z, &A);

LSX_SetVelFac	
Beschreibung:	Geschwindigkeitsuntersetzung setzen.
C++:	int LSX_SetVelFac (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A***: Geschwindigkeitsuntersetzung, Wertebereich 0.01 -- 1.00
Beispiel:	pTango->SetVelFac(1, 0.1, 0.1, 0.1, 0.1); *// reduziert die Geschwindigkeiten aller Achsen auf 1/10 der eingestellten Geschwindigkeit*

LSX_LStepSave	
Beschreibung:	Aktuelle Konfiguration in Tango speichern (EEPROM).
C++:	int LSX_LStepSave (int lLSID);
Parameter:	-
Beispiel:	pTango->LStepSave(1);

LSX_SetAccelSingleAxis	
Beschreibung:	Beschleunigung einer einzelnen Achse einstellen.
C++:	int LSX_SetAccelSingleAxis (int lLSID, int lAxis, double dAccel);
Parameter:	***Axis***: X, Y, Z, A nummeriert von 1 bis 4 ***Accel***: Beschleunigung 0.01 - 20.00 [m/s²]
Beispiel:	pTango->SetAccelSingleAxis(1, 3, 1,0); *// setzt die Beschleunigung der Z-Achse auf 1.0 m/s²*

LSX_SetVelSingleAxis	
Beschreibung:	Geschwindigkeit einer einzelnen Achse einstellen.
C++:	int LSX_SetVelSingleAxis (int lLSID, int lAxis, double dVel);
Parameter:	***Axis***: X, Y, Z und A nummeriert von 1 bis 4 ***Vel***: 0 - maximale Geschwindigkeit [U/s]
Beispiel:	pTango->SetVelSingleAxis(1, 2, 10.0); *// setzt die Geschwindigkeit der Y-Achse auf 10 U/s*

LSX_SoftwareReset	
Beschreibung:	Die Software wird in den Startzustand versetzt.
C++:	int LSX_SoftwareReset (int lLSID);
Parameter:	-
Beispiel:	pTango->SoftwareReset(1);

4.6 Fahrbefehle und Positionsverwaltung

LSX_Calibrate	
Beschreibung:	Es werden alle freigegebenen Achsen kalibriert. Die Achsen fahren so lange in Richtung kleinerer Positionwerte gefahren, bis der Endschalter angefahren wird. Danach wird mit reduzierter Geschwindigkeit so lange in umgekehrter Richtung gefahren, bis der Endschalter nicht mehr betätigt ist. Dann wird der Nullpunkt gesetzt.
C++:	int LSX_Calibrate (int lLSID);
Parameter:	-
Beispiel:	pTango->Calibrate(1);

LSX_CalibrateEx	
Beschreibung:	Einzelne Achsen kalibrieren. Es werden nur die Achsen kalibriert, deren entsprechendes Bit in dem übergebenen IntegerWert gesetzt ist.
C++:	int LSX_CalibrateEx (int lLSID, int lFlags);
Parameter:	***Flags***: BitMaske Bit 2 = 1 → ZAchse kalibrieren Bit 2 = 0 → ZAchse nicht kalibrieren ...
Beispiel:	pTango->CalibrateEx(1, 6); *// nur Y und ZAchse kalibrieren (Bit 1 und 2 gesetzt)*

LSX_ClearPos	
Beschreibung:	Setzt die aktuelle Position und den internen Zähler auf 0. Diese Funktion wird für Endlosachsen gebraucht, da die Steuerung nur +-1000 Motorumdrehungen vom Wertebereich verarbeiten kann. Bei erkanntem Geber wird die Funktion für die jeweilige Achse nicht ausgeführt.
C++:	int LSX_ClearPos (int lLSID, int lFlags);
Parameter:	***Flags***: BitMaske Bit 0 = 1 → Position der XAchse wird genullt Bit 1 = 0 → für die YAchse wird die Funktion nicht ausgeführt
Beispiel:	pTango->ClearPos(1, 5); *// Positionen der X- und Z-Achsen werden genullt (Bit 0 und 2 gesetzt)*

LSX_GetDelay	
Beschreibung:	Abfrage der Verzögerungszeit.
C++:	int LSX_GetDelay (int lLSID, int *plDelay);
Parameter:	***Delay***: Verzögerung [ms]
Beispiel:	pTango->GetDelay(1, &Delay);

LSX_SetDelay	
Beschreibung:	Setzen der Verzögerungszeit. Die Steuerung wartet vor jeder Positionierung diese Verzögerungszeit ab.
C++:	int LSX_SetDelay (int lLSID, int lDelay);
Parameter:	***Delay***: 0 - 10000 [ms]
Beispiel:	pTango->SetDelay(1, 1000); *// 1 Sekunde Verzögerung*

LSX_GetDistance	
Beschreibung:	Abfrage der zuletzt benutzten Distanzwerte für LSX_MoveRelShort.
C++:	int LSX_GetDistance (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z, A***: die aktuellen Distanzen aller Achsen, abhängig von der jeweiligen Maßeinheit
Beispiel:	pTango->GetDistance(1, &X, &Y, &Z, &A);

LSX_SetDistance	
Beschreibung:	Setzen der Distanz. Die Parameter für den Befehl LSX_MoveRelShort werden gesetzt. Damit sind sehr schnelle gleichartige Positionierungen möglich.
C++:	int LSX_SetDistance (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A***: min/max-Verfahrbereich, Werte sind abhängig von der Maßeinheit
Beispiel:	pTango->SetDistance(1, 1, 2, 0, 0); *// Distanzen für die Achsen X und Y werden gesetzt,* *Z und A werden bei Aufruf der Funktion LS_MoveRelShort nicht bewegt*

LSX_MoveAbs	
Beschreibung:	Alle Achsen werden auf Absolut-Positionen gefahren. Die Achsen X, Y, Z und A werden auf die übergebenen Positionswerte positioniert.
C++:	int LSX_MoveAbs (int lLSID, double dX, double dY, double dZ, double dA, BOOL bWait);
Parameter:	***X, Y, Z, A***: + Verfahrbereich, Eingabe ist abhängig von der Maßeinheit ***Wait***: gibt an, ob die Funktion nachdem die Position erreicht wurde (= true) oder direkt zurückkehren soll (= false)
Beispiel:	pTango->MoveAbs(1, 10.0, 10.0, -10.0, 10.0, true);

LSX_MoveAbsSingleAxis	
Beschreibung:	Eine einzelne Achse absolut positionieren.
C++:	int LSX_MoveAbsSingleAxis (int lLSID, int lAxis, double dValue, BOOL bWait);
Parameter:	***Axis***: X, Y, Z und A nummeriert von 1 bis 4 ***Value***: Position, Eingabe ist abhängig von der eingestellten Maßeinheit
Beispiel:	pTango->MoveAbsSingleAxis(1, 2, 10.0); *// Y-Achse auf 10mm absolut positionieren*

LSX_MoveEx	
Beschreibung:	Erweiterter VerfahrBefehl. Die Funktion LSX_MoveEx kann relative und absolute Verfahrbefehle ausführen, synchron und asynchron. Die Anzahl der Achsen, die verfahren werden sollen, kann mit dem Parameter AxisCount bestimmt werden. Diese Funktion kann beispielsweise genutzt werden, um nur X und Y zu verfahren.
C++:	int LSX_MoveEx (int lLSID, double dX, double dY, double dZ, double dA, BOOL bRelative, BOOL bWait, int lAxisCount);
Parameter:	***X, Y, Z, A***: Positionsvektoren ***Relative***: bei Relative = false werden die Werte X, Y, Z und A als absolute Koordinaten interpretiert bei Relative = true als relative Koordinaten zur aktuellen Position ***Wait***: wird Wait = true gesetzt, kehrt die Funktion erst nach Erreichen der Zielposition zurück, ansonsten kehrt sie unmittelbar nach Senden des Befehls an die Tango zurück ***AxisCount***: Anzahl der Achsen, die verfahren werden sollen ist AxisCount = 1, wird nur X verfahren ist AxisCount = 2, werden X und Y verfahren ...
Beispiel:	pTango->MoveEx(1, 2.0, 3.0, 0, 0, true, true, 2); *// es werden X und Y um 2 bzw 3 relativ verfahren*

LSX_MoveRel	
Beschreibung:	Relativen Vektor fahren. Die Achsen X, Y, Z und A werden um die übergebenen Strecken verfahren und kommen alle gleichzeitig an Ihrem Ziel an.
C++:	int LSX_MoveRel (int lLSID, double dX, double dY, double dZ, double dA, BOOL bWait);
Parameter:	***X, Y, Z, A***: + Verfahrbereich, Eingabe ist abhängig von der Dimension ***Wait***: true = die Funktion wartet, bis die Position erreicht wird false = die Funktion wartet nicht
Beispiel:	pTango->MoveRel(1, 10.0, 10.0, -10.0, 10.0, true);

LSX_MoveRelShort	
Beschreibung:	Positionieren Relativ (short command). Dieser Befehl sollte verwendet werden, damit aufeinander folgende relative Verfahrbefehle (mit derselben Strecke) schneller angefahren werden. Die Strecke muß zuvor einmal mit LSX_SetDistance gesetzt worden sein.
C++:	int LSX_MoveRelShort (int lLSID);
Parameter:	-
Beispiel:	pTango->SetDistance(1, 1.0, 1.0, 0, 0); for (i = 0; i < 10; i++) pTango->MoveRelShort(1); *// 10 mal X und YAchse um 1 mm relativ positionieren*

LSX_MoveRelSingleAxis	
Beschreibung:	Einzelne Achse relativ verfahren.
C++:	int LSX_MoveRelSingleAxis (int lLSID, int lAxis, double dValue, BOOL bWait);
Parameter:	***Axis***: X, Y, Z und A nummeriert von 1 bis 4 ***Value***: Strecke, Eingabe ist abhängig von der eingestellten Maßeinheit
Beispiel:	pTango->MoveRelSingleAxis(1, 3, 5,0); *// ZAchse um 5mm in positiver Richtung verfahren*

LSX_RMeasure	
Beschreibung:	Die Endpositionen aller freigegebenen Achsen werden gesucht. Die Achsen fahren so lange in Richtung größerer Positionwerte, bis der Endschalter gefunden wird. Danach wird mit reduzierter Geschwindigkeit so lange in umgekehrter Richtung gefahren, bis der Endschalter nicht mehr betätigt ist. Dann wird der maximale mögliche Verfahrbereich gesetzt.
C++:	int LSX_RMeasure (int lLSID);
Parameter:	-
Beispiel:	pTango->RMeasure(1);

LSX_RMeasureEx	
Beschreibung:	Endposition der Achsen (max. Fahrweg) messen. Endposition der Achsen (max. Fahrweg) messen wird nur bei den Achsen durchgeführt, deren entsprechendes Bit in dem übergebenen IntegerWert gesetzt ist.
C++:	int LSX_RMeasureEx (int lLSID, int lFlags);
Parameter:	***Flags***: BitMaske Bit 2 = 1 → ZAchse kalibrieren Bit 2 = 0 → ZAchse nicht kalibrieren ...
Beispiel:	pTango->RMeasureEx(1, 2); *// nur Endposition der Y-Achse messen*

LSX_SetPos	
Beschreibung:	Positionswerte setzen.
C++:	int LSX_SetPos (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A***: min/maxVerfahrbereich, Eingabe ist abhängig von der Dimension
Beispiel:	pTango->SetPos(1, 10, 10, 0, 0);

LSX_StopAxes	
Beschreibung:	Abbruch. Es werden alle Verfahrbewegungen abgebrochen.
C++:	int LSX_StopAxes (int lLSID);
Parameter:	-
Beispiel:	pTango->StopAxes(1);

LSX_WaitForAxisStop	
Beschreibung:	Die Funktion kehrt zurück, sobald die in der BitMaske AFlags gewählten Achsen ihre Zielposition erreicht haben. LS_WaitForAxisStop verwendet '?statusaxis', um den Status der Achsen zu polen.
C++:	int LSX_WaitForAxisStop (int lLSID, int lAFlags, int lATimeoutValue, BOOL *pbATimeout);
Parameter:	***AFlags***: BitMaske Bit 0: XAchse Bit 1: YAchse Bit 2: ZAchse Bit 3: AAchse ***AtimeoutValue***: Timeout in Millisekunden ***WaitForAxisStop*** kehrt nach dieser Zeit mit ATimeout = true zurück, wenn die Achsen immer noch in Bewegung sind ***ATimeoutValue*** = 0 setzt den Timeout auf ´´unendlich´´ ***ATimeout Flag***: gibt an, ob ein Timeout aufgetreten ist
Beispiel:	pTango->WaitForAxisStop(1, 3, 0, flag); *// warten bis X- und YAchse gestoppt haben, kein Timeout* pTango->WaitForAxisStop(1, 7, 10000, flag); *// warten bis X-, Y- und ZAchse gestoppt haben, 10 Sekunden Timeout*

4.7 Joystick und Handrad

LSX_GetDigJoySpeed	
Beschreibung:	Auslesen der eingestellten Joystick-Geschwindigkeiten.
C++:	int LSX_GetDigJoySpeed (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z, A***: Geschwindigkeitswerte [U/s]
Beispiel:	pTango->GetDigJoySpeed(1, &X, &Y, &Z, &A);

LSX_SetDigJoySpeed	
Beschreibung:	Mit diesem Befehl können einzelne Achsen mit einer konstanten Geschwindigkeit verfahren werden. Will man nach dem Ausführen der Funktion wieder absolut oder relativ positionieren, muss man erst mit LSX_SetJoystick(0) den Joystick ausschalten und die Geschwindigkeit neu setzen.
C++:	int LSX_SetDigJoySpeed (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A***: Geschwindigkeit [U/s], Wertebereich: + max. Geschwindigkeit
Beispiel:	pTango->SetDigJoySpeed(1, 0, 10.0, 25.0, 0); *// Achsen X und A - Geschwindigkeit 0 und JoystickBetrieb ´´AUS´´,* *Achse Y - Geschwindigkeit 10.0 U/s und JoystickBetrieb ´´EIN´´,* *Achse Z - Geschwindigkeit 25.0 U/s und JoystickBetrieb ´´EIN´´*

LSX_GetHandWheel	
Beschreibung:	Liest den Zustand des Handrades ab.
C++:	int LSX_GetHandWheel (int lLSID, BOOL *pbHandWheelOn, BOOL *pbPositionCount, BOOL *pbEncoder);
Parameter:	***HandWheelOn***: true = Handrad ist eingeschaltet false = Handrad ist ausgeschaltet ***PositionCount***: true = Positionszählung ist eingeschaltet false = Positionszählung ist ausgeschaltet ***Encoder***: true = Geberwerte, wenn vorhanden
Beispiel:	pTango->GetHandWheel(1, &HandWheelOn, &PositionCount, &Encoder);

LSX_GetJoystick	
Beschreibung:	Abfrage des aktuellen Zustands vom AnalogJoystick.
C++:	int LSX_GetJoystick (int lLSID, BOOL *pbJoystickOn, BOOL *pbManual, BOOL *pbPositionCount, BOOL *pbEncoder);
Parameter:	***JoystickOn***: true = Joystick ist eingeschaltet ***Manual***: false = Joystickschalter steht auf Automatik true = Joystick ist manual über Schalter eingeschaltet ***PositionCount***: true = Positionszählung ist eingeschaltet ***Encoder***: true = Geberwerte, wenn vorhanden
Beispiel:	pTango->GetJoystick(1, &JoystickOn, &Manual, &PositionCount, &Encoder);

LSX_GetJoystickDir	
Beschreibung:	Liest Motordrehrichtung für Joystick
C++:	int LSX_GetJoystickDir (int lLSID, int *plXD, int *plYD, int *plZD, int *plAD);
Parameter:	***XD, YD, ZD, AD:*** 0 → Achse gesperrt 1 → positive Drehrichtung 1 → negative Drehrichtung 2 → mit Stromreduzierung 2 → mit Stromreduzierung
Beispiel:	pTango->GetJoystickDir(1, &XD, &YD, &ZD, &AD);

LSX_SetJoystickDir	
Beschreibung:	Setzt die Motordrehrichtung für den Joystick.
C++:	int LSX_SetJoystickDir (int lLSID, int lXD, int lYD, int lZD, int lAD);
Parameter:	***XD, YD, ZD, AD:*** 0 → Achse gesperrt 1 → positive Drehrichtung 1 → negative Drehrichtung 2 → mit Stromreduzierung 2 → mit Stromreduzierung
Beispiel:	pTango->SetJoystickDir(1, 1, 1, -1, 0); *// X und YAchse positive Drehrichtung, ZAchse negative Drehrichtung,* *AAchse gesperrt*

LSX_GetJoystickWindow	
Beschreibung:	JoystickFenster ablesen.
C++:	int LSX_GetJoystickWindow (int lLSID, int *plAValue);
Parameter:	***AValue***: der Analogbereich, in dem sich die Achsen nicht bewegen
Beispiel:	pTango->GetJoystickWindow(1, &AValue);

LSX_SetJoystickWindow	
Beschreibung:	JoystickFenster setzen.
C++:	int LSX_SetJoystickWindow (int lLSID, int lAValue);
Parameter:	***AValue***: 0100
Beispiel:	pTango->SetJoystickWindow(1, 30);

LSX_SetHandWheelOff	
Beschreibung:	Handrad ausschalten.
C++:	int LSX_SetHandWheelOff (int lLSID);
Parameter:	-
Beispiel:	pTango->SetHandWheelOff(1);

LSX_SetHandWheelOn	
Beschreibung:	Handrad einschalten.
C++:	int LSX_SetHandWheelOn (int lLSID, BOOL bPositionCount, BOOL bEncoder);
Parameter:	***PositionCount*** = true → Positionszählung einschalten = false → Positionszählung ausschalten ***Encoder*** = true → Geberwerte, wenn vorhanden
Beispiel:	pTango->SetHandWheelOn(1, true, true); *// Handrad mit Positionszählung (Geberwerte) einschalten*

LSX_SetJoystickOff	
Beschreibung:	AnalogJoystick ausschalten.
C++:	int LSX_SetJoystickOff (int lLSID);
Parameter:	-
Beispiel:	pTango->SetJoystickOff(1);

LSX_SetJoystickOn	
Beschreibung:	AnalogJoystick einschalten.
C++:	int LSX_SetJoystickOn (int lLSID, BOOL bPositionCount, BOOL bEncoder);
Parameter:	***PositionCount*** = true → Positionszählung einschalten = false → Positionszählung ausschalten ***Encoder*** = true → Geberwerte, wenn vorhanden
Beispiel:	pTango->SetJoystickOn(1, true, true); *// Joystick mit Positionszählung (Geberwerte) einschalten*

4.8 Bedienpult mit Trackball und Joyspeed-Tasten

LSX_GetBPZ	
Beschreibung:	Liest den Zustand des Zusatzbedienpults mit Trackball.
C++:	int LSX_GetBPZ (int lLSID, int *plAValue);
Parameter:	***AValue***: 0 → Bedienpult ist ´´AUS´´ 1 → Bedienpult aktiv, Trackball wird mit 0,1µ Schrittauflösung betrieben 2 → Bedienpult aktiv, Trackball wird mit Faktor betrieben.
Beispiel:	pTango->GetBPZ(1, &AValue);

LSX_SetBPZ	
Beschreibung:	Bedienpult ein-/ausschalten.
C++:	int LSX_SetBPZ (int lLSID, int lAValue);
Parameter:	***AValue***: 02 0 → Bedienpult ´´AUS´´ 1 → Bedienpult aktivieren und Trackball mit 0,1µ Schrittauflösung betreiben 2 → Bedienpult aktivieren und Trackball mit Faktor betreiben
Beispiel:	pTango->SetBPZ(1, 1);

LSX_GetBPZJoyspeed	
Beschreibung:	Liest Bedienpult JoystickGeschwindigkeit..
C++:	int LSX_GetBPZJoyspeed (int lLSID, int lAPar, double *pdAValue);
Parameter:	***APar***: 1, 2 oder 3 ***AValue***: maximale Geschwindigkeit [U/s]
Beispiel:	pTango->GetBPZJoyspeed(1, &AValue); *// auslesen der eingestellten Geschwindigkeit von Parameter 1*

LSX_SetBPZJoyspeed	
Beschreibung:	Bedienpult Joystick-Geschwindigkeit einstellen.
C++:	int LSX_SetBPZJoyspeed (int lLSID, int lAPar, double dAValue);
Parameter:	***APar***: 1, 2 oder 3 ***AValue***: +-maximale Geschwindigkeit [U/s]
Beispiel:	pTango->SetBPZJoyspeed(1, 1, 25); *// Parameter 1 mit Geschwindigkeit 25 beschreiben*

LSX_GetBPZTrackballBackLash	
Beschreibung:	Liest Bedienpult Trackball-Umkehrspiel.
C++:	int LSX_GetBPZTrackballBackLash (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z A***: Umkehrspiel [mm]
Beispiel:	pTango->GetBPZTrackballBackLash(1, &X, &Y, &Z, &A);

LSX_SetBPZTrackballBackLash	
Beschreibung:	Bedienpult TrackballUmkehrspiel einstellen.
C++:	int LSX_SetBPZTrackballBackLash (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A***: 0.001 bis 0.15 mm
Beispiel:	pTango->SetBPZTrackballBackLash(1, 0.01, 0.01, 0.01, 0.01);

LSX_GetBPZTrackballFactor	
Beschreibung:	Bedienpult TrackballFaktor auslesen.
C++:	int LSX_GetBPZTrackballFactor (int lLSID, double *pdAValue);
Parameter:	***AValue***: Trackball-Faktor z.B. AValue = 3 bedeutet: ein TrackballImpuls ergibt 3 MotorIncremente
Beispiel:	pTango->GetBPZTrackballFactor(1, &AValue);

LSX_SetBPZTrackballFactor	
Beschreibung:	Bedienpult TrackballFaktor einstellen.
C++:	int LSX_SetBPZTrackballFactor (int lLSID, double dAValue);
Parameter:	***AValue***: 0.01 - 100 AValue = 1 → Trackball-Faktor = 1, d.h. ein TrackballImpuls ergibt ein Motor-Increment
Beispiel:	pTango->SetBPZTrackballFactor(1, 1,0);

4.9 Endschalter (Hardware und Software)

LSX_GetAutoLimitAfterCalibRM	
Beschreibung:	Gibt an, ob beim Kalibrieren und Tischhubmessen die internen Software Limits gesetzt werden.
C++:	int LSX_GetAutoLimitAfterCalibRM (int lLSID, int *plFlags);
Parameter:	***Flags***: BitMaske Bit 0 = 1 → bei der XAchse werden keine Verfahrbereichsgrenzen gesetzt Bit 1 = 0 → für die YAchse werden SoftwareLimits gesetzt (calib/rm)
Beispiel:	pTango->GetAutoLimitAfterCalibRM(1, &Flags);

LSX_SetAutoLimitAfterCalibRM	
Beschreibung:	Verhindert, dass beim Kalibrieren und Tischhubmessen die internen SoftwareLimits gesetzt werden.
C++:	int LSX_SetAutoLimitAfterCalibRM (int lLSID, int lFlags);
Parameter:	***Flags***: BitMaske Bit 0 = 1 → Bei der XAchse werden keine Verfahrbereichsgrenzen gesetzt Bit 1 = 0 → Für die YAchse werden SoftwareLimits gesetzt (calib/rm)
Beispiel:	pTango->SetAutoLimitAfterCalibRM(1, Flags);

LSX_GetLimit	
Beschreibung:	Verfahrbereichsgrenzen lesen
C++:	int LSX_GetLimit (int lLSID, int lAxis, double *pdMinRange, double *pdMaxRange);
Parameter:	***Axis***: die Achse, welcher Verfahrbereichsgrenzen gelesen werden sollen (X, Y, Z, A nummeriert von 1 bis 4) ***MinRange***: untere Verfahrbereichsgrenze, abhängig von der Dimension ***MaxRange***: obere Verfahrbereichsgrenze, abhängig von der Dimension
Beispiel:	pTango->GetLimit(1, &MinRange, &MaxRange);

LSX_SetLimit	
Beschreibung:	Verfahrbereichsgrenzen einstellen.
C++:	int LSX_SetLimit (int lLSID, int lAxis, double dMinRange, double dMaxRange);
Parameter:	***Axis***: die Achse, von der Verfahrbereichsgrenzen gelesen werden sollen (X, Y, Z und A nummeriert von 1 bis 4) ***MinRange***: untere Verfahrbereichsgrenze, abhängig von der Dimension ***MaxRange***: obere Verfahrbereichsgrenze, abhängig von der Dimension
Beispiel:	pTango->SetLimit(1, 1, -10.0, 20.0); *// X-Achse -10 als untere und 20 als obere Verfahrbereichsgrenze zuweisen*

LSX_GetLimitControl	
Beschreibung:	Liest, ob die Bereichsüberwachung aktiv ist.
C++:	int LSX_GetLimitControl (int lLSID, int lAxis, BOOL *pbActive);
Parameter:	***Axis***: X, Y, Z und A nummeriert von 1 bis 4 ***Active***: true = Bereichsüberwachung der jeweiligen Achse ist aktiv false = Bereichsüberwachung der jeweiligen Achse ist deaktiviert
Beispiel:	pTango->GetLimitControl(1, 2, &Active);

LSX_SetLimitControl	
Beschreibung:	Bereichsüberwachung ein-/ausschalten.
C++:	int LSX_SetLimitControl (int lLSID, int lAxis, BOOL bActive);
Parameter:	***Axis***: X, Y, Z und A nummeriert von 1 bis 4 ***Active***: Bereichsüberwachung der jeweiligen Achse aktivieren
Beispiel:	pTango->SetLimitControl(1, 2, true); *// Bereichsüberwachung der Y-Achse ist aktiv*

LSX_GetSwitchActive	
Beschreibung:	Gibt an, ob die Endschalter eingeschaltet sind.
C++:	int LSX_GetSwitchActive (int lLSID, int *plXA, int *plYA, int *plZA, int *plAA);
Parameter:	Für jede Achse wird eine Bit-Maske geliefert: Bit 0 → NullEndschalter Bit 1 → ReferenzEndschalter Bit 2 → EndEndschalter Ist das Bit gesetzt, ist der jeweilige Schalter aktiv.
Beispiel:	pTango->GetSwitchActive(1, &XA, &YA, &ZA, &AA);

LSX_SetSwitchActive	
Beschreibung:	Endschalter ein-/ausschalten.
C++:	int LSX_SetSwitchActive (int lLSID, int lXA, int lYA, int lZA, int lAA);
Parameter:	Für jede Achse wird eine Bit-Maske geliefert: Bit 0 → NullEndschalter Bit 1 → ReferenzEndschalter Bit 2 → EndEndschalter Um den jeweiligen Schalter zu aktivieren, muss das Bit gesetzt werden.
Beispiel:	pTango->SetSwitchActive(1, 7, 1, 5, 0); *// alle Endschalter der XAchse ´´Ein´´; NullEndschalter der YAchse ´´Ein´´; E0 und EE der ZAchse ein; AAchse: alle Endschalter ´´Aus´´*

LSX_GetSwitches	
Beschreibung:	Liest den Zustand aller Endschalter.
C++:	int LSX_GetSwitches (int lLSID, int *plFlags);
Parameter:	***Flags***: Zeiger auf Integerwert, der den Zustand aller Endschalter als Bit-Maske enthält In der Bit-Maske ist der Zustand der Endschalter wie folgt verschlüsselt: Endschalter EE Ref. E0 Achse AZYX AZYX AZYX Bit 0000 0000 0000 z.B.: Flags = 0x003 → E0 von X und YAchse sind angefahren Flags = 0x200 → EE von YAchse ist angefahren
Beispiel:	pTango->GetSwitches(1, &Flags);

LSX_GetSwitchPolarity	
Beschreibung:	Liest Endschalterpolarität.
C++:	int LSX_GetSwitchPolarity (int lLSID, int *plXP, int *plYP, int *plZP, int *plAP);
Parameter:	Für jede Achse wird eine Bit-Maske geliefert: Bit 0 → NullEndschalter Bit 1 → ReferenzEndschalter Bit 2 → EndEndschalter Ist das Bit gesetzt reagiert der jeweilige Schalter auf die positive Flanke.
Beispiel:	pTango->GetSwitchPolarity(1, &XP, &YP, &ZP, &AP);

LSX_SetSwitchPolarity	
Beschreibung:	Setzt Endschalterpolarität.
C++:	int LSX_SetSwitchPolarity (int lLSID, int lXP, int lYP, int lZP, int lAP);
Parameter:	Für jede Achse wird eine Bit-Maske übergeben: Bit 0 → NullEndschalter Bit 1 → ReferenzEndschalter Bit 2 → EndEndschalter Reagiert der jeweilige Schalter auf die positive Flanke, muß das Bit gesetzt werden.
Beispiel:	pTango->SetSwitchPolarity(1, 7, 0, 0, 0); *// alle Endschalter der XAchse sind HighAktive,* *alle Endschalter der Y, Z- und A-Achse sind LowAktive*

4.10 Digitale und analoge Ein – und Ausgänge

LSX_GetAnalogInput	
Beschreibung:	Lesen des aktuellen A/D-Wandlungswerts eines Analogkanals.
C++:	int LSX_GetAnalogInput (int lLSID, int lIndex, int *plValue);
Parameter:	***Index***: 015 (Analogkanäle), 0...9 = HDI Buchse, Pins 1...10 10 = ANAIN0 des AUX-IO Steckers ***Value***: Zeiger auf Integerwert, der den aktuellen Zustand des Analogkanals angibt 0...5V analog = 0...1023
Beispiel:	pTango->GetAnalogInput(1, 0, &Eingang0);

LSX_GetDigitalInputs	
Beschreibung:	Alle Inputpins lesen.
C++:	int LSX_GetDigitalInputs (int lLSID, int *plValue);
Parameter:	***Value***: Zeiger auf Integerwert, der den Zustand aller Eingänge als Bit-Maske enthält
Beispiel:	int Eingaenge; pTango->GetDigitalInputs(1, &Eingange); if (Eingaenge & 16) ... *// wenn Inputpin 4 gesetzt ...*

LSX_GetDigitalInputsE	
Beschreibung:	Zusätzliche digitale Eingänge lesen (1631).
C++:	int LSX_GetDigitalInputsE (int lLSID, int *plValue);
Parameter:	***Value***: Zeiger auf einen 32BitInteger, welcher nach Aufruf der Funktion in den Bits 015 den Status der Eingänge 1631 enthält
Beispiel:	pTango->GetDigitalInputsE(1, i);

LSX_SetAnalogOutput	
Beschreibung:	Analogkanal setzen.
C++:	int LSX_SetAnalogOutput (int lLSID, int lIndex, int lValue);
Parameter:	***Index***: 01 (Analogkanäle) ***Value***: 0100 [%]
Beispiel:	pTango->SetAnalogOutput(1, 0, 100); *// Ausgang 0 auf Maximun setzen*

LSX_SetDigIO_Distance	
Beschreibung:	Funktion der digitalen Ein/Ausgänge. Aktivierung eines Ausgangs in Abhängigkeit der eingestellten Strecke vor/nach der Zielposition.
C++:	int LSX_SetDigIO_Distance (int lLSID, int lIndex, BOOL bFkt, double dDist, int lAxis);
Parameter:	***Index***: 0 bis 15 (Outputpin) ***Fkt*** = false → Aktivierung eines Ausgang in Abhängigkeit der eingestellten Strecke vor der Zielposition ***Fkt*** = true → Aktivierung eines Ausgang in Abhängigkeit der eingestellten Strecke nach der Startposition ***Dist***: Strecke, Eingabe ist abhängig von der eingestellten Dimension ***Axis***: X, Y, Z und A nummeriert von 1 bis 4
Beispiel:	pTango->SetDigIO_Distance(1, 7, false, 78.9, 3); *// Ausgang 7 wird 78.9mm vor erreichen der Zielposition (Z Achse) aktiviert*

LSX_SetDigIO_EmergencyStop	
Beschreibung:	Funktion der digitalen Ein/Ausgänge. Zuordnung des NotStopPins.
C++:	int LSX_SetDigIO_EmergencyStop (int lLSID, int lIndex);
Parameter:	***Index***: 0 bis 15 (Input/Output)
Beispiel:	pTango->SetDigIO_EmergencyStop(1, 15); *// NotStopPin 15*

LSX_SetDigIO_Off	
Beschreibung:	Funktion der digitalen Ein/Ausgänge ausschalten. (Keine Beeinflussung der Ein/Ausgänge).
C++:	int LSX_SetDigIO_Off (int lLSID, int lIndex);
Parameter:	Index: 0 bis 15 (Input/Output), 16 (alle 16 Portpins)
Beispiel:	pTango->SetDigIO_Off(1, 0); *// digitale Funktion Input/Outputpin 0 ´´Aus´´*

LSX_SetDigIO_Polarity	
Beschreibung:	Funktion der digitalen Ein/Ausgänge. Einstellung der Polarität.
C++:	int LSX_SetDigIO_Polarity (int lLSID, int lIndex, BOOL bHigh);
Parameter:	***Index***: 0 bis 15 (Input/Output), 16 (alle 16 Portpins) ***High*** = true → HighAktiv ***High*** = false → LowAktiv
Beispiel:	pTango->SetDigIO_Polarity(1, 3, true); *// Input/Outputpin 3 HighAktiv*

LSX_SetDigitalOutput	
Beschreibung:	Outputpin setzen.
C++:	int LSX_SetDigitalOutput (int lLSID, int lIndex, BOOL bValue);
Parameter:	***Index***: 015 ***Value***: Zustand auf ´´0´´ oder ´´1´´ setzen
Beispiel:	pTango->SetDigitalOutput(1, 0, true); *// Outputpin 0 auf ´´1´´ setzen*

LSX_SetDigitalOutputs	
Beschreibung:	Digitale Ausgänge 0-15 setzen.
C++:	int LSX_SetDigitalOutputs (int lLSID, int lValue);
Parameter:	***Value***: Bit-Maske, der Wert auf den die Ausgänge 015 gesetzt werden, wird durch die Bits 015 bestimmt
Beispiel:	pTango->SetDigitalOutputs(1, 3); *// Ausgänge 0 und 1 auf 1 setzen, die übrigen auf 0*

LSX_SetDigitalOutputsE	
Beschreibung:	Zusätzliche digitale Ausgänge 16-31 setzen.
C++:	int LSX_SetDigitalOutputsE (int lLSID, int lValue);
Parameter:	***Value***: Bit-Maske, der Wert auf den die Ausgänge 1631 gesetzt werden, wird durch die Bits 015 bestimmt
Beispiel:	pTango->SetDigitalOutputsE(1, 3); *// Ausgänge 16 und 17 auf 1 setzen, die übrigen auf 0*

4.11 Geber-Einstellungen

LSX_ClearEncoder	
Beschreibung:	GeberZähler auf null setzen.
C++:	int LSX_ClearEncoder (int lLSID, int lAxis);
Parameter:	***Axis***: X, Y, Z und A nummeriert von 1 bis 4
Beispiel:	pTango->ClearEncoder(1, 2); *// GeberZähler der YAchse auf Null setzen*

LSX_GetEncoder	
Beschreibung:	Liest alle Geberpositionen.
C++:	int LSX_GetEncoder (int lLSID, double *pdXP, double *pdYP, double *pdZP, double *pdAP);
Parameter:	***XP, YP, ZP, AP***: Zählerwerte, 4fach interpoliert
Beispiel:	pTango->GetEncoder(1, &XP, &YP, &ZP, &AP);

LSX_GetEncoderActive	
Beschreibung:	Liest, welche Geber nach der Kalibration aktiviert werden. Hinweis: Diese Funktion entspricht dem „?encmask“ Befehl.
C++:	int LSX_GetEncoderActive (int lLSID, int *plFlags);
Parameter:	***Flags***: Geber-Maske
Beispiel:	pTango->GetEncoderActive(1, &Flags);

LSX_SetEncoderActive	
Beschreibung:	Mit dieser Funktion kann ausgewählt werden, welche Geber nach der Kalibration aktiviert werden sollen. Hinweis: Diese Funktion entspricht dem „!encmask“ Befehl.
C++:	int LSX_SetEncoderActive (int lLSID, int lFlags);
Parameter:	***Value***: Geber-Maske
Beispiel:	pTango->SetEncoderActive(1, 0); *// alle Geber deaktivieren* pTango->SetEncoderActive(1, 2); *// Geber YAchse aktivieren*

LSX_GetEncoderMask	
Beschreibung:	Geberzustände auslesen. Hinweis: Diese Funktion entspricht dem „?enc“ Befehl.
C++:	LSX_GetEncoderMask (int lLSID, int *plFlags);
Parameter:	***Flags***: aktive Geber
Beispiel:	int EncMask; pTango->GetEncoderMask(1, &EncMask); if (EncMask & 2) ... *// wenn Geber YAchse angeschlossen + aktiv ...*

LSX_SetEncoderMask	
Beschreibung:	Geber manuell (de)aktivieren. Hinweis: Diese Funktion entspricht dem „!enc“ Befehl. Nicht im closed loop anwenden. Die Geber sollten immer durch den Calibrate-Befehl aktiviert werden.
C++:	int LSX_SetEncoderMask (int lLSID, int lValue);
Parameter:	***Value***: aktive Geber (Flags)
Beispiel:	pTango->SetEncoderMask(1, 0); *// alle Geber deaktivieren* pTango->SetEncoderMask (1, 2); *// Geber YAchse aktivieren*

LSX_GetEncoderPeriod	
Beschreibung:	Geberperiodenlängen auslesen.
C++:	int LSX_GetEncoderPeriod (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z, A***: Periodenlängen [mm]
Beispiel:	pTango->GetEncoderPeriod(1, &X, &Y, &Z, &A);

LSX_SetEncoderPeriod	
Beschreibung:	Geberperiodenlängen einstellen.
C++:	int LSX_SetEncoderPeriod (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A***: 0.0001 - Spindelsteigung * 0.8 (mm)
Beispiel:	pTango->SetEncoderPeriod(1, 0.1, 0.1, 0.1, 0.1); *// Geber-Periodenlänge aller Achsen ist 0.1mm*

LSX_GetEncoderPosition	
Beschreibung:	Einstellung der Geberwertanzeige lesen.
C++:	int LSX_GetEncoderPosition (int lLSID, BOOL *pbValue);
Parameter:	***Value***: true → bei der Positionsabfrage werden die Geberwerte der erkannten Geber angezeigt false → Geberpositionsanzeige ist ´´AUS´´
Beispiel:	pTango->GetEncoderPosition(1, &Value);

LSX_SetEncoderPosition	
Beschreibung:	Geberwertanzeige ein-/ausschalten.
C++:	int LSX_SetEncoderPosition (int lLSID, BOOL bValue);
Parameter:	***Value*** = true → bei der Positionsabfrage werden die Geberwerte der erkannten Geber angezeigt
Beispiel:	pTango->SetEncoderPosition(1, true);

LSX_GetEncoderRefSignal	
Beschreibung:	Liest, ob beim Kalibrieren das Referenzsignal vom Geber ausgewertet wird.
C++:	int LSX_GetEncoderRefSignal (int lLSID, int *plXR, int *plYR, int *plZR, int *plAR);
Parameter:	**X, Y, Z, A**: 1 → beim Kalibrieren wird das Referenzsignal ausgewertet 0 → das Referenzsignal wird nicht ausgewertet
Beispiel:	pTango->GetEncoderRefSignal(1, &X, &Y, &Z, &A);

LSX_SetEncoderRefSignal	
Beschreibung:	Beim Kalibrieren Referenzsignal von Geber auswerten.
C++:	int LSX_SetEncoderRefSignal (int lLSID, int lXR, int lYR, int lZR, int lAR);
Parameter:	***X, Y, Z, A***: 0 oder 1
Beispiel:	pTango->SetEncoderRefSignal(1, 1, 1, 0, 0); *// beim Kalibrieren wird das Referenzsignal der Geber X und Y ausgewertet*

4.12 Reglereinstellungen

LSX_ClearCtrFastMoveCounter	
Beschreibung:	Bei einer Reglerdifferenz, die größer als der Fangbereich ist, wird ein neuer Vektor gestartet und der dazu gehörige Counter um eins erhöht.
C++:	int LSX_ClearCtrFastMoveCounter (int lLSID);
Parameter:	-
Beispiel:	pTango->ClearCtrFastMoveCounter(1);

LSX_GetController	
Beschreibung:	ReglerModus auslesen.
C++:	int LSX_GetController (int lLSID, int *plXC, int *plYC, int *plZC, int *plRC);
Parameter:	***ReglerModus X, Y, Z, A***: 0 → Regler ´´AUS´´ 1 → Regler ´´AUS nach erreichen der Zielposition´´ 2 → Regler ´´Immer EIN´´ 3 → Regler ´´AUS nach erreichen der Zielposition´´ mit reduziertem Strom 4 → Regler ´´Immer EIN´´ mit reduziertem Strom
Beispiel:	pTango->GetController(1, &X, &Y, &Z, &A);

LSX_SetController	
Beschreibung:	ReglerModus einstellen.
C++:	int LSX_SetController (int lLSID, int lXC, int lYC, int lZC, int lAC);
Parameter:	ReglerModus ***X, Y, Z, A***: 0 → Regler ´´AUS´´ 1 → Regler ´´AUS nach erreichen der Zielposition´´ 2 → Regler ´´Immer EIN´´ 3 → Regler ´´AUS nach erreichen der Zielposition´´ mit reduziertem Strom 4 → Regler ´´Immer EIN´´ mit reduziertem Strom
Beispiel:	pTango->SetController(1, 1, 2, 0, 0);

LSX_GetControllerCall	
Beschreibung:	Liefert Regleraufrufzeit.
C++:	int LSX_GetControllerCall (int lLSID, int *plCtrCall);
Parameter:	***CtrCall***: Regleraufrufzeit [ms]
Beispiel:	pTango->GetControllerCall(1, &CtrCall);

LSX_SetControllerCall	
Beschreibung:	Regleraufrufzeit einstellen.
C++:	int LSX_SetControllerCall (int lLSID, int lCtrCall);
Parameter:	***CtrCall***: Regleraufrufzeit [ms]
Beispiel:	pTango->SetControllerCall(1, 10); *// nach dem Funktionsaufruf CtrCall = 10 bedeutet: Regleraufruf alle 10 ms*

LSX_GetControllerFactor	
Beschreibung:	ReglerFaktoren auslesen.
C++:	int LSX_GetControllerFactor (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z, A***: ReglerFaktoren
Beispiel:	pTango->GetControllerFactor(1, &X, &Y, &Z, &A);

LSX_SetControllerFactor	
Beschreibung:	Reglerfaktor einstellen.
C++:	int LSX_SetControllerFactor (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A***: 1 - 64
Beispiel:	pTango->SetControllerFactor(1, 1, 2, 3, 4);

LSX_GetControllerSteps	
Beschreibung:	Liefert die ReglerSchrittlänge.
C++:	int LSX_GetControllerSteps (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z, A***: ReglerSchrittlänge [mm]
Beispiel:	pTango->GetControllerSteps(1, &X, &Y, &Z, &A);

LSX_SetControllerSteps	
Beschreibung:	ReglerSchritte einstellen.
C++:	int LSX_SetControllerSteps (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A***: 1 - Spindelsteigung (Werte sind abhängig von der Dimension)
Beispiel:	pTango->SetControllerSteps(1, 4, 5, 7, 9);

LSX_GetControllerTimeout	
Beschreibung:	Liest ReglerTimeout.
C++:	Int LSX_GetControllerTimeout (int lLSID, int *plACtrTimeout);
Parameter:	***ACtrTimeout***: Timeout [ms], nach dem ein Verfahrbefehl mit Fehlermeldung (Fehlercode 4013) zurückkehrt, wenn die Steuerung eine Position nicht endgültig finden konnte.
Beispiel:	pTango->GetControllerTimeout(1, &ACtrTimeout);

LSX_SetControllerTimeout	
Beschreibung:	ReglerTimeout einstellen.
C++:	int LSX_SetControllerTimeout (int lLSID, int lACtrTimeout);
Parameter:	***ACtrTimeout***: Timeout [ms], nach dem ein Verfahrbefehl mit Fehlermeldung (Fehlercode 4013) zurückkehrt, wenn die Steuerung eine Position nicht endgültig finden konnte.
Beispiel:	pTango->SetControllerTimeout(1, 500);

LSX_GetControllerTWDelay	
Beschreibung:	Reglerverzögerung auslesen.
C++:	int LSX_GetControllerTWDelay (int lLSID, int *plCtrTWDelay);
Parameter:	***CtrTWDelay***: Reglerverzögerung [ms]
Beispiel:	pTango->GetControllerTWDelay(1, &CtrTWDelay);

LSX_SetControllerTWDelay	
Beschreibung:	Reglerverzögerung setzen.
C++:	int LSX_SetControllerTWDelay (int lLSID, int lCtrTWDelay);
Parameter:	***CtrTWDelay***: Reglerverzögerung 0 - 100 [ms]
Beispiel:	pTango->SetControllerTWDelay(1, 0); *// ReglerVerzögerung aus*

LSX_GetCtrFastMove	
Beschreibung:	Liest die Einstellung der FastMove-Funktion.
C++:	int LSX_GetCtrFastMove (int lLSID, BOOL *pbActive);
Parameter:	***Active***: true → FastMove-Funktion aktiv
Beispiel:	pTango->GetCtrFastMove(1, &Active);

LSX_GetCtrFastMoveCounter	
Beschreibung:	Bei einer Reglerdifferenz, die größer als der Fangbereich ist, wird ein neuer Vektor gestartet und der dazu gehörige Counter um eins erhöht. Die Funktion liefert Fast Move Counters
C++:	int LSX_GetCtrFastMoveCounter (int lLSID, int *plXC, int *plYC, int *plZC, int *plAC);
Parameter:	***XC, YC, ZC, AC***: Anzahl ausgeführter Fast Move Funktionen
Beispiel:	pTango->GetCtrFastMoveCounter(1, &XC, &YC,&ZC,&AC);

LSX_GetTargetWindow	
Beschreibung:	Liest die Zielfenster aller Achsen.
C++:	int LSX_GetTargetWindow (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z, A***: Zielfenster, abhängig von der Dimension
Beispiel:	pTango->GetTargetWindow(1, &X, &Y, &Z, &A);

LSX_SetTargetWindow	
Beschreibung:	Reglerzielfenster einstellen.
C++:	int LSX_SetTargetWindow (int lLSID, double dX, double dY, double dZ, double dA);
Parameter:	***X, Y, Z, A***: 1 - 25000 (Motorincremente) 0.1 - Spindelsteigung/2 (µm) 0.0001 - Spindelsteigung/2 (mm) (Werte sind abhängig von der Dimension)
Beispiel:	pTango->SetTargetWindow(1, 1.0, 0.002, 1.0, 1.0);

LSX_SetCtrFastMoveOff	
Beschreibung:	FastMove-Funktion deaktivieren.
C++:	int LSX_SetCtrFastMoveOff (int lLSID);
Parameter:	-
Beispiel:	pTango->SetCtrFastMoveOff(1);

LSX_SetCtrFastMoveOn	
Beschreibung:	FastMove-Funktion aktivieren, d.h. bei einer Reglerdifferenz, die größer als der Fangbereich ist, wird ein neuer Vektor gestartet.
C++:	int LSX_SetCtrFastMoveOn (int lLSID);
Parameter:	-
Beispiel:	pTango->SetCtrFastMoveOn(1);

4.13 Trigger-Ausgang

LSX_GetTrigCount	
Beschreibung:	Triggerzählerstand lesen.
C++:	int LSX_GetTrigCount (int lLSID, int *plValue);
Parameter:	***Value***: Anzahl der ausgeführten Trigger
Beispiel:	pTango->GetTrigCount(1, &Value);

LSX_SetTrigCount	
Beschreibung:	Triggerzählerstand setzen.
C++:	int LSX_SetTrigCount (int lLSID, int lValue);
Parameter:	***Value***: 0 bis 2147483647
Beispiel:	pTango->SetTrigCount(1, 0);

LSX_GetTrigger	
Beschreibung:	Liefert den aktuellen Triggerzustand.
C++:	int LSX_GetTrigger (int lLSID, BOOL *pbATrigger);
Parameter:	***ATrigger:*** true → Trigger ´´Ein´´ false → Trigger ´´Aus´´
Beispiel:	pTango->GetTrigger(1, &ATrigger);

LSX_SetTrigger	
Beschreibung:	Trigger ein-/ausschalten.
C++:	int LSX_SetTrigger (int lLSID, BOOL bATrigger);
Parameter:	***ATrigger*** = true → Trigger einschalten = false → Trigger ausschalten
Beispiel:	pTango->SetTrigger(1, true);

LSX_GetTriggerPar	
Beschreibung:	Liefert TriggerParameter.
C++:	int LSX_GetTriggerPar (int lLSID, int *plAxis, int *plMode, int *plSignal, double *pdDistance);
Parameter:	***Axis***: Achse (1..4) ***Mode***: Trigger-Modus (siehe Befehl !trigm) ***Signal***: Trigger-Signal (siehe Befehl !trigs) ***Distance***: Trigger-Distanz (siehe Befehl !trigd)
Beispiel:	pTango->GetTriggerPar(1, &Axis, &Mode, & Signal, &Distance);

LSX_SetTriggerPar	
Beschreibung:	TriggerParameter einstellen.
C++:	int LSX_SetTriggerPar (int lLSID, int lAxis, int lMode, int lSignal, double dDistance);
Parameter:	***Axis***: Achse (1..4) ***Mode***: Trigger-Modus (siehe Befehl !trigm) ***Signal***: Trigger-Signal (siehe Befehl !trigs) ***Distance***: Trigger-Distanz (siehe Befehl !trigd)
Beispiel:	pTango->SetTriggerPar(1, 1, 3, 2, 5.0);

4.14 Snapshot-Eingang

LSX_GetSnapshot	
Beschreibung:	Liefert den aktuellen SnapshotZustand.
C++:	int LSX_GetSnapshot (int lLSID, BOOL *pbASnapshot);
Parameter:	ASnapshot: true → Snapshot ´´Ein´´ false → Snapshot ´´Aus´´
Beispiel:	pTango->GetSnapshot(1, &ASnapshot);

LSX_SetSnapshot	
Beschreibung:	Snapshot ein-/ausschalten.
C++:	int LSX_SetSnapshot (int lLSID, BOOL bASnapshot);
Parameter:	***ASnapshot***: true → Snapshot ´´Ein´´ false → Snapshot ´´Aus´´
Beispiel:	pTango->SetSnapshot(1, true);

LSX_GetSnapshotCount	
Beschreibung:	SnapshotZähler.
C++:	int LSX_GetSnapshotCount (int lLSID, int *plSnsCount);
Parameter:	***SnsCount***: SnapshotZähler
Beispiel:	pTango->GetSnapshotCount(1, &SnsCount);

LSX_GetSnapshotFilter	
Beschreibung:	Eingangsfilter auslesen.
C++:	int LSX_GetSnapshotFilter (int lLSID, int *plTime);
Parameter:	***Time***: Filterzeit [ms]
Beispiel:	pTango->GetSnapshotFilter(1, &Time);

LSX_SetSnapshotFilter	
Beschreibung:	Eingangsfilter bei prellenden Schaltern setzen.
C++:	int LSX_SetSnapshotFilter (int lLSID, int lTime);
Parameter:	***Time***: Filterzeit, Wertebereich 0 - 100 [ms]
Beispiel:	pTango->SetSnapshotFilter(1, 0); *// kein Eingangsfilter*

LSX_GetSnapshotPar	
Beschreibung:	SnapshotParameter auslesen.
C++:	int LSX_GetSnapshotPar (int lLSID, BOOL *pbHigh, BOOL *pbAutoMode);
Parameter:	***High***: true → Snapshot ist HighAktiv false → Snapshot ist LowAktiv ***AutoMode***: true → Snapshot ´´Automatik´´. Die Position wird nach dem ersten Impuls automatisch angefahren.
Beispiel:	pTango->GetSnapshotPar(1, &High, &AutoMode);

LSX_SetSnapshotPar	
Beschreibung:	SnapshotParameter festlegen.
C++:	int LSX_SetSnapshotPar (int lLSID, BOOL bHigh, BOOL bAutoMode);
Parameter:	***High***: true → Snapshot ist HighAktiv false → Snapshot ist LowAktiv ***AutoMode***: true → Snapshot ´´Automatik´´. Die Position wird nach dem ersten Impuls automatisch angefahren.
Beispiel:	pTango->SetSnapshotPar(1, true, false);

LSX_GetSnapshotPos	
Beschreibung:	SnapshotPosition auslesen.
C++:	int LSX_GetSnapshotPos (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***X, Y, Z, A***: Positionswerte
Beispiel:	pTango->GetSnapshotPos(1, &X, &Y, &Z, &A);

LSX_GetSnapshotPosArray	
Beschreibung:	SnapshotPosition aus Array auslesen.
C++:	int LSX_GetSnapshotPosArray (int lLSID, int lIndex, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameter:	***Index***: Nummer der SnapshotPosition (1200) ***X, Y, Z, A***: Positionswerte
Beispiel:	pTango->GetSnapshotPosArray(1, 2, &X, &Y, &Z, &A);

5. Fehlercodes

0: Kein Fehler

5.1 TangoFehlermeldungen

1: Keine gültige Achsenbezeichnung
2: Keine ausführbare Funktion
3: Zu viele Zeichen in BefehlsString
4: Kein gültiger Befehl
5: Außerhalb des gültigen Zahlenbereiches
6: Falsche Anzahl der Parameter
7: Kein !oder ?
8: Kein TVR möglich, da Achse aktiv
9: Kein Ein oder Ausschalten der Achsen, da TVR aktiv
10: Funktion nicht konfiguriert
11: Kein MoveBefehl möglich, da JoystickHand
12: Endschalter betätigt
13: Funktion kann nicht ausgeführt werden, da Encoder erkannt (clear pos.)
14: Fehler beim Kalibrieren (Endschalter nicht korrekt freigefahren)
20: Treiberrelaise defekt (Sicherheitskreis K3/K4)
21: Es dürfen nur einzelne Vektoren verfahren werden (Einrichtbetrieb)
22: Es darf kein Kalibrieren, Tischhubmessen oder JoystickBetrieb durchgeführt werden (Tür offen oder Einrichtbetrieb)
23: SECURITY Error XAchse
24: SECURITY Error YAchse
25: SECURITY Error ZAchse
26: SECURITY Error AAchse
27: NOTSTOP
28: Fehler im Türschaltersicherheitskreis
29: Endstufen nicht eingeschaltet
30: GAL Sicherheitsfehler

5.2 DLLFehlermeldungen

4001: interner Fehler
4002: interner Fehler
4003: undefinierter Fehler
4004: Unbekannter Schnittstellentyp (kann bei Connect... auftreten)
4005: Fehler beim Initialisieren der Schnittstelle
4006: Keine Verbindung zur Steuerung (z.B. wenn SetPitch vor Connect aufgerufen wird)
4007: Timeout während Lesen von der Schnittstelle
4008: Fehler bei Befehlsübertragung an die Tango
4009: Befehl wurde abgebrochen (mit SetAbortFlag)
4010: Befehl wird von Tango nicht unterstützt
4011: JoystickHand eingeschaltet (kann bei SetJoystickOn/Off auftreten)
4012: Kein VerfahrBefehl möglich, da JoystickHand
4013: ReglerTimeout
4014:
4015: Endschalter in Verfahrrichtung betätigt
4016: Wiederholter Vektorstart!! (Regelung)
4017: Fehler beim Kalibrieren (Endschalter nicht korrekt freigefahren)