Programmierschnittstelle

LSTEP-API

LANG GMBH & CO. KG
Dillstraße 4
D-35625 Hüttenberg
Tel. +49 6403 7009-0
Telefax +49 6403 7009-40

Inhaltsverzeichnis

1 Einführung

Die LSTEP-API (Programmierschnittstelle für die LSTEP Feinpositioniersysteme) soll Software-Entwicklern dabei helfen, Anwendungen mit Steuerungen der LSTEP-Familie schnell und effektiv zu entwickeln, ohne sich mit hardwarenaher Programmierung beschäftigen zu müssen. Es bietet Zugriff auf den kompletten Befehlssatz der LSTEP-Positioniersysteme. Es stehen zwei DLLs zur Verfügung: Die LSTEP4X.DLL und die LSTEP64.DLL.

1.1 Funktionsumfang

- Windows 32-bit DLL
- Windows 64-bit DLL
- Unterstützung der Motorsteuerungen LSTEP xx, LSTEP xx/2, LSTEP-PC, ECO-STEP, LSTEP-44, LSTEP-PCI, LSTEP-PCI express und LSTEP express
- Ansteuerung über RS232, USB, Ethernet, ISA, PCI (DPRAM) oder PCIexpress. Die Ansteuerung der Schnittstelle ist abhängig vom Steuerungstypen.
- Automatische Erkennung der angeschlossenen Steuerung
- Konfiguration der Steuerung
- Ausführung aller von der Steuerung unterstützten Befehle
- Bis zu 4 Achsen
- Multithreading-fähig

1.2 Systemanforderungen

Mit der LSTEP-API können auf Intel-PCs unter MS Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8, Windows 8.1 Anwendungen entwickelt werden.

1.3 Unterstützte Entwicklungsumgebungen

Die LSTEP-API wurden mit den folgenden Entwicklungs- und Laufzeitumgebungen getestet:

Borland/Inprise Delphi 3-7, Embarcadero Delphi XE2 und neuer

Microsoft Visual C++ 2013

Microsoft Visual C# 2010

National Instruments LabVIEW 32 und 64 Bit (veraltet)

Sie sollten kompatibel mit allen anderen Programmier-Umgebungen sein, die DLLs verwenden können.

(DLL = Dynamic Link Library; Eine DLL ist ein ausführbares Modul, das Code und Ressourcen enthält, die von anderen Anwendungen oder DLLs verwendet werden.)

2 DLL-Schnittstelle

2.1 LSTEP-API

Hauptbestandteil des LSTEP-APIs ist die Datei LSTEP4X.DLL bzw. LSTEP64.DLL. Diese DLLs verwenden Sie bei der Entwicklung eigener Programme, um eine oder mehrere LSTEPs zu konfigurieren, Befehle zu senden, Positionswerte, Ein-/Ausgänge abzufragen etc.

2.2 Allgemeine Hinweise

Die DLL LSTEP4X.DLL bzw. LSTEP64.DLL implementiert die Befehle der LSTEP-API. Alle Funktionen sind mit einem 32-Bit Integer als Rückgabewert deklariert. Eine Null als Rückgabewert zeigt die fehlerfreie Ausführung der Funktion an, bei Fehlern (z. B. Timeouts) wird der entsprechende Fehlercode (siehe Abschnitt 6 Fehlercodes) zurückgeliefert.

Als erster Parameter wird bei sämtlichen Funktionen der API ein 32-bit Integer-Wert übergeben (zwischen 1 und 32), welcher die Nummer der LSTEP angibt, an die der Befehl gesendet werden soll.

Die Funktion LSX_CreateLSID muss verwendet werden, um einen solchen ID-Wert zu erzeugen. Mit einem Aufruf von LSX_FreeLSID wird ein ID-Wert wieder freigegeben. (Siehe Delphi-Beispiel)

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.

2.3 Einbindung in Delphi

Allen Funktionsnamen der LSTEP-API ist ein „LSX_“ vorangestellt (siehe LStep4x.pas). Um die Funktionen der LSTEP-API verwenden zu können, muss LSTEP4X.pas bzw. LStep64.pas in der uses-Klausel der entsprechenden Unit stehen, und in einem der eingestellten Suchpfade liegen.

Delphi-Beispiel für die parallele Ansteuerung von 2 LSTEP-Steuerungen

benötigte Dateien: LSTEP4X.DLL und LSTEP4X.pas oder LSTEP64.DLL und LSTEP64.pas

```
uses … LSTEP4X, …
...
var LStep1, LStep2: Integer;
…
begin
LSX_CreateLSID(LStep1);
LSX_CreateLSID(LStep2);
```

```
LSX_ConnectSimple(LStep1, 1, 'COM1', 9600, True);
LSX_ConnectSimple(LStep2, 1, 'COM2', 9600, True);

LSX_MoveAbs(LStep1, 10.0, 20.0, 30.0, 0.0, True);
LSX_MoveAbs(LStep2, 5.0, 10.0, 0.0, 0.0, True);

LSX_Disconnect(LStep1);
LSX_Disconnect(LStep2);

LSX_FreeLSID(LStep1);
LSX_FreeLSID(LStep2);

end;
```

2.4 Einbindung in Visual C++

Für Visual C++ wurde eine Kapselung der LSTEP4X.DLL bzw. LSTEP64.DLL erstellt. Die Klasse CLStep4X bzw. CLStep64 lädt die DLL und alle Zeiger auf Funktionsaufrufe dynamisch. Den Methoden des LSTEP Objekts ist kein „LSX_“ vorangestellt.
(Beispiel: LSX.Calibrate() statt LSX_Calibrate)

Die Funktionen LSX_CreateLSID und LSX_FreeLSID müssen Sie in C++ zur Verwendung der LSTEP4X.DLL bzw. LSTEP64.DLL nicht aufrufen, da die Wrapper-Klasse CLStep4X bzw. CLStep64 den Integer-Wert, welcher die Nummer der LSTEP angibt, selbst verwaltet. Die Methoden von CLStep4X besitzen also keinen zusätzlichen Parameter für die Nummer der LSTEP.

Visual C++-Beispiel für die parallele Ansteuerung von 2 LSTEP-Steuerungen
Benötigte Dateien: LSTEP4X.DLL, LSTEP4X.h und LSTEP4X.cpp oder LSTEP64.DLL, LSTEP64.h und LSTEP64.cpp

```
...
CLStep4X* LS1,* LS2;
…
LS1 = new CLStep4X;
LS2 = new CLStep4X;

LS1->ConnectSimple(1, "COM1", 9600, true);
LS2->ConnectSimple(1, "COM2", 9600, true);

LS1->MoveAbs(10.0, 20.0, 30.0, 0.0, true);
LS2->MoveAbs(5.0, 10.0, 0.0, 0.0, true);
```

```
LS1->Disconnect();
delete LS1;
LS2->Disconnect();
delete LS2;
```

2.5 Einbindung in LabVIEW

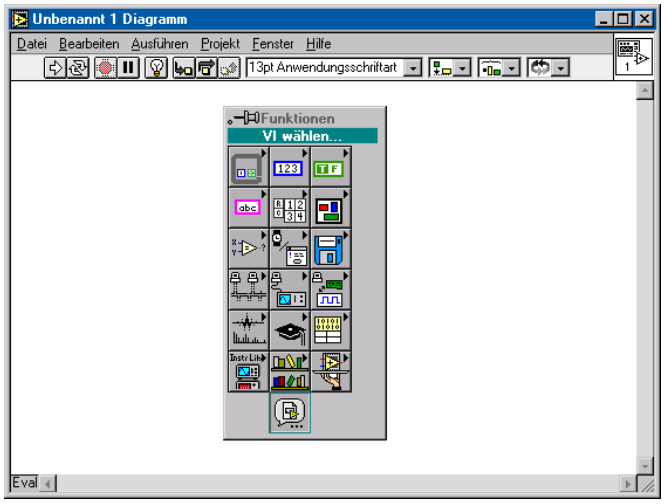
NI LabVIEW ist eine auf der graphischen Programmiersprache G basierende Entwicklungsumgebung. Sie ermöglicht eine vereinfachte und schnelle Programmierung mit graphischen Symbolen. Die Erstellung komplilierter 32-bit bzw. 64-bit Programme ist möglich, sodass die nötige Ausführungsgeschwindigkeit für Steuerungs-, Test- und Meßanwendungen gegeben ist.

Alle LabVIEW-Programme (sogenannte VIs, Virtual Instruments) besitzen ein Frontpanel und ein Blockdiagramm, und können wiederum als Unterprogramm (SubVI) in andere Programme eingebunden werden.

Für die Einbindung der LSTEP-API (LSTEP4X.DLL bzw. LSTEP64.DLL) wurden VI-Bibliotheken (LSTEP4X.LLB bzw. LSTEP64.LLB) erstellt, die eine Sammlung an VIs enthalten. Diese einzelnen VIs (z.B. LS4 ConnectSimple.vi) kapseln die entsprechenden LSTEP-API-Funktionen. Die LSTEP4X.DLL wird mittels ‚Call Library Function' (Aufruf ext. Bibliotheken) verwendet.

2.5.1 Vorgehensweise zur Verwendung eines LSTEP4-VI's

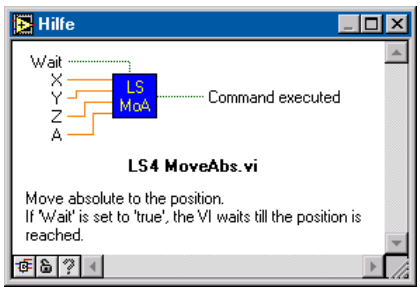
1. Neues VI erstellen
2. Zu Blockdiagramm-Fenster wechseln (Strg+E)
3. Auf Diagramm klicken (rechte Maustaste)



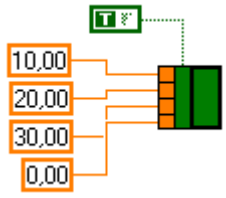
4. VI wählen...

5. Im Dateidialog die mitgelieferte VI-Bibliothek LSTEP4X.LLB öffnen, daraus dann
   den gewünschten Befehl wählen (z.B. LS4 MoveAbs.vi)
6. VI im Diagramm plazieren

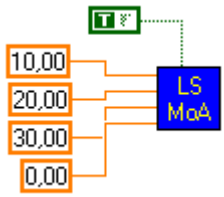
Die Tastenkombination Strg+H öffnet ein Hilfe-Fenster, dieses zeigt Informationen über das VI an, auf dem sich momentan der Mauszeiger befindet



Die Übergabe von Parametern an SubVIs erfolgt über Anschlüsse, die ‚verkabelt' werden müssen. Damit diese Anschlüsse im Diagramm gezeigt werden, klicken Sie mit der rechten Maustaste auf das VI und wählen ‚Anzeigen/Anschlüsse'. Anschließend können Sie den Anschlüssen Werte/Quellen zuweisen. Einer von mehreren möglichen Wegen: Klicken Sie mit der rechten Maustaste auf den gewünschten Anschluss, dann auf den Menüpunkt ‚Konstante erzeugen'.



In diesem Beispiel wird ein absoluter Verfahrbefehl (X 10mm, Y 20mm, Z 30mm) ausgeführt.



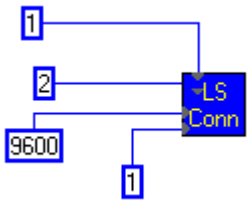
Die Belegung der Anschlüsse der VIs kann der Dokumentation der entsprechenden API-Funktion entnommen werden, dort findet sich die graphische Darstellung, welche auch im Hilfe-Fenster von LabVIEW erscheint. Die Parameter entsprechen

weitgehend denen der DLL-Funktion: Lediglich bei Funktionen, denen Bitmasken als Parameter übergeben werden, sind einige Unterschiede vorhanden (z.B. LS4 SetActiveAxes.vi)

Die LSTEP4 VIs verfügen über einen Anschluss namens ‚Command executed'. Ist dieser boolesche Wert ‚true', wurde der Befehl erfolgreich ausgeführt. Falls ein Fehler aufgetreten ist, wird der Wert auf ‚false' gesetzt.

Bevor Verfahrbefehle ausgeführt, Positionswerte ausgelesen werden können etc. muss die Verbindung zur LSTEP geöffnet werden. Dies ist am einfachsten über das VI ‚LS4 ConnectSimple.vi' möglich. Es initialisiert die Schnittstelle und erkennt die angeschlossene LSTEP.

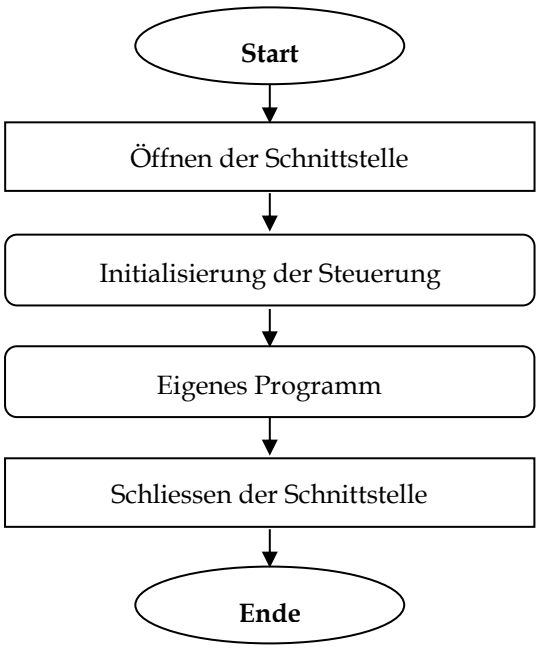
Beispiel für RS232 (COM2 und 9600 Baud):



3 Aufbau eigener Programme mit der API

Die folgende Abbildung zeigt den Programmflussplan nach dem die Programme zum Steuern der Positioniersysteme aufgebaut sein sollten. Die verwendeten Funktionen sind in der Beschreibung der LSTEP-API aufgelistet und werden dort genauer beschrieben.

Die LSTEP-Steuerungen werden vorkonfiguriert ausgeliefert. Diese Konfiguration deckt allerdings nicht jeden Anwendungsfall ab und muss an die eigene Anwendung angepasst werden. Diese Anpassung erfolgt nach dem Öffnen der Schnittstelle. Anschliessend kann unter Verwendung der LSTEP-API ein beliebiges Anwenderprogramm geschrieben werden. Nach Beendigung des Anwendungsprogramms ist die Schnittstelle wieder zu schließen.



3.1 Öffnen der Schnittstelle

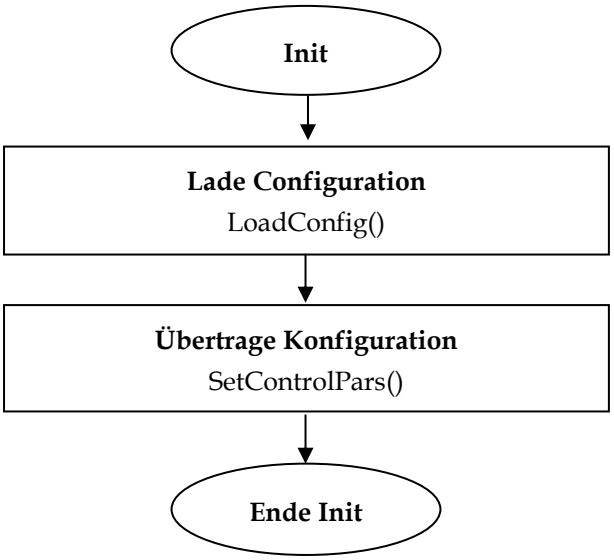
Das Öffnen der Schnittstelle erfolgt über eine der Funktionen Connect, ConnectEx oder ConnectSimple, wobei die Funktion ConnectSimple zum Verbindungsaufbau empfohlen wird. Welche Schnittstelle zur Verbindung mit einer Steuerung geöffnet werden muss ist abhängig vom Steuerungstyp und der Einstellung der Schnittstelle. Die Standardeinstellung der Schnittstelle kann in der Dokumentation der jeweiligen Steuerung nachgelesen werden. Des Weiteren kann die folgende Tabelle zurate gezogen werden:

Steuerungsserie	Steuerungsname	Befehlssatz	Unterstützter ConnectSimple Interface-Type (Default)						
			1	2	3	4	5	6	11
	MCL	Register	x						
LSTEP Serie	LSTEP-xx	Register	x						
	LSTEP-PC	Register			x				
LSTEP 2000 Serie	LSTEP-PCI 32Bit-Windows	Ipreter Register	x			x			
	LSTEP-PCI 64Bit-Windows	Ipreter Register						x	
	LSTEP-xx/2	Ipreter Register	x						
	LSTEP-44	Ipreter	x						
	ECO-STEP	Ipreter Register	x						
	ECO-Mot	Ipreter Register	x						
	ECO-Drive	Ipreter Register	x						
LSTEP express Serie	LSTEP-PCI express	Ipreter					x		x
	LSTEP express	Ipreter					x		x

3.2 Initialisierung der Steuerung

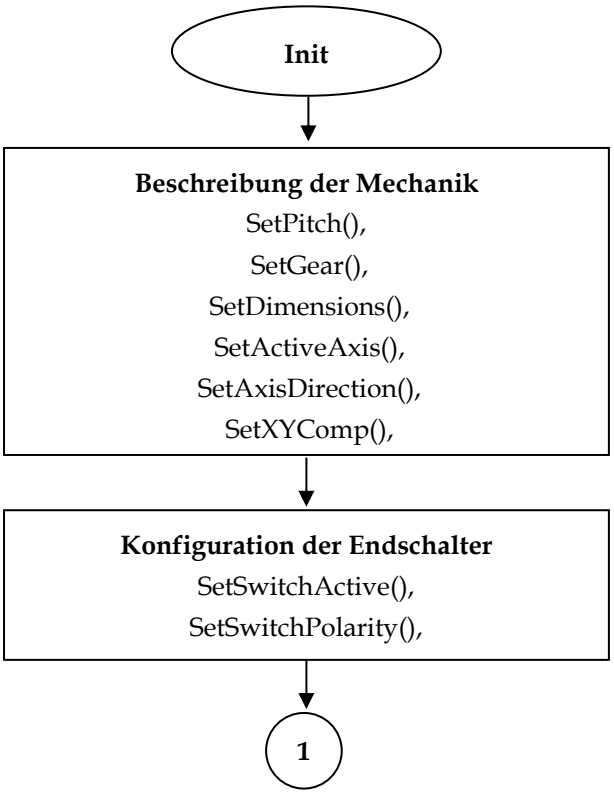
3.2.1 Allgemein

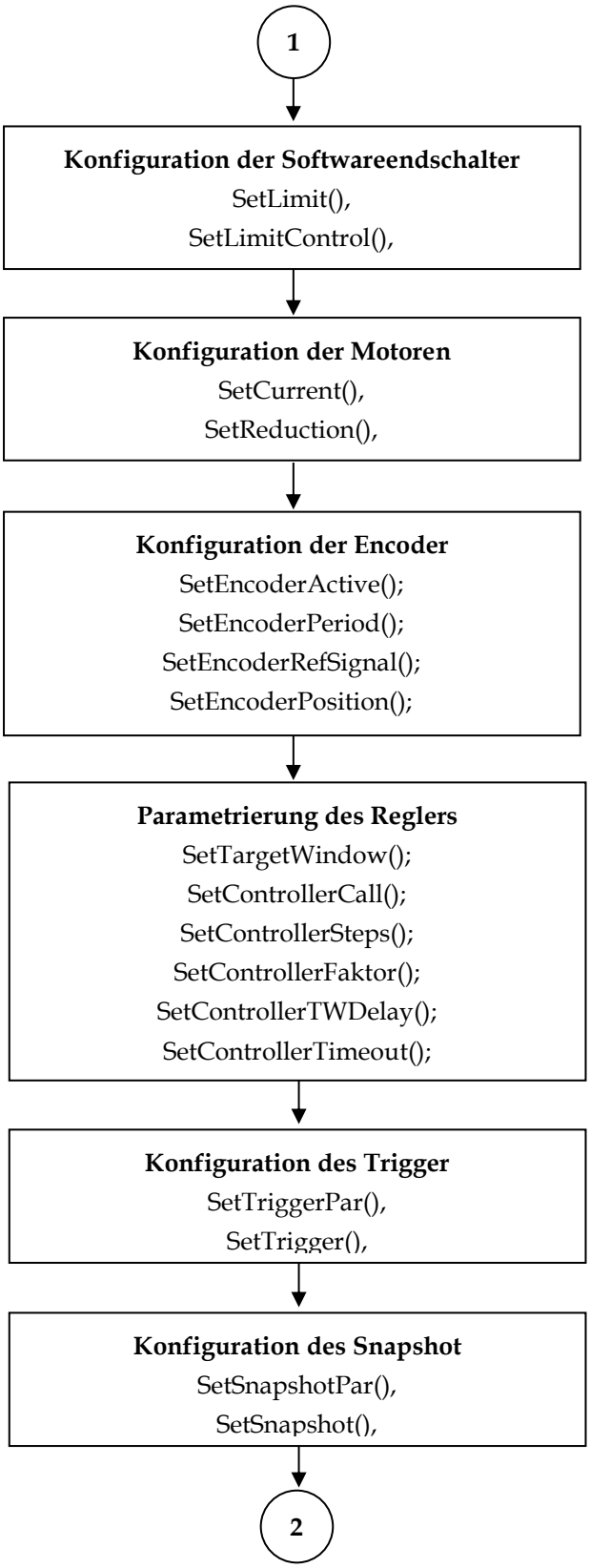
Vor dem Start des eigenen Programmteils sollte die LSTEP-Steuerung konfiguriert werden. Hierzu kann z.B. das Inbetriebnahme-Programm WIN-Commander verwendet werden, aus dem eine Konfigurationsdatei erstellt werden kann. Diese Datei kann anschließend von der API geladen und an die Steuerung gesendet werden.

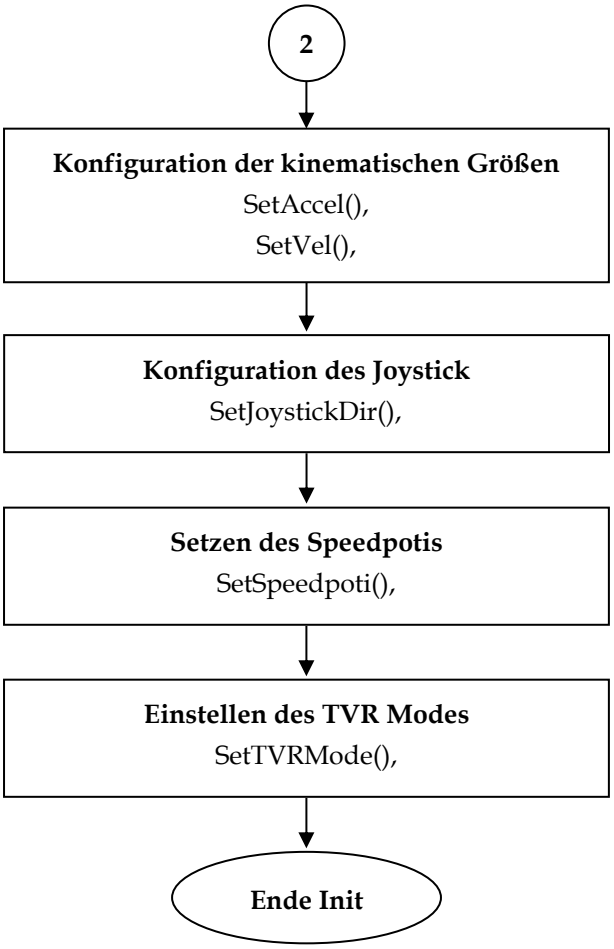


Sollen keine grundlegenden Änderungen an den Steuerungseinstellungen von der API vorgenommen werden, können die Einstellungen in der Steuerung gespeichert werden. Hierzu kann ebenfalls das Inbetriebnahme-Programm WIN-Commander verwendet werden.

Neben diesen Möglichkeiten, bietet die API Funktionen zur Konfiguration der Steuerung an, die im nachfolgenden Flussdiagramm für eine LSTEP-Steuerung gezeigt werden.


Init
Beschreibung der Mechanik
SetPitch(),
SetGear(),
SetDimensions(),
SetActiveAxis(),
SetAxisDirection(),
SetXYComp(),
Konfiguration der Endschalter
SetSwitchActive(),
SetSwitchPolarity(),
1


1
Konfiguration der Softwareendschalter
SetLimit(),
SetLimitControl(),
Konfiguration der Motoren
SetCurrent(),
SetReduction(),
Konfiguration der Encoder
SetEncoderActive();
SetEncoderPeriod();
SetEncoderRefSignal();
SetEncoderPosition();
Parametrierung des Reglers
SetTargetWindow();
SetControllerCall();
SetControllerSteps();
SetControllerFaktor();
SetControllerTWDelay();
SetControllerTimeout();
Konfiguration des Trigger
SetTriggerPar(),
SetTrigger(),
Konfiguration des Snapshot
SetSnapshotPar(),
SetSnapshot(),
2



3.2.2 LSTEP 2000 Familie

Zur Konfiguration der Steuerungen der LSTEP 2000 Familie steht eine Vielzahl an API-Funktionen bereit. Weiterhin kann über den WIN-Commander 4 und dem Menüpunkt „INI-Datei speichern unter…“ unter dem Menü Hauptmenü → Optionen eine Konfigurationsdatei erzeugt werden. Diese Konfigurationsdatei enthält neben der Steuerungskonfiguration zusätzlich die Konfiguration des WinCommander 4. Soll nur die Steuerungskonfiguration gespeichert werden ist dies über „Einstellungen speichern“ unter dem Menü Hauptmenü → Steuerung möglich.

Die jeweilige Konfigurationsdatei kann über den API-Befehl LoadConfig gelesen und mit dem Befehl SetControlPars an die Steuerung gesendet werden. Anschließend kann mit dem API-Befehl LStepSave die Konfiguration in der Steuerung gespeichert werden, so dass sie nach dem nächsten Einschalten der Steuerung sofort geladen wird. Das Speichern der Einstellungen in der Steuerung kann weiterhin aus dem WIN-Commander heraus erfolgen, wodurch sich ein manuelles Laden der Datei zum Programmstart erübrigt.

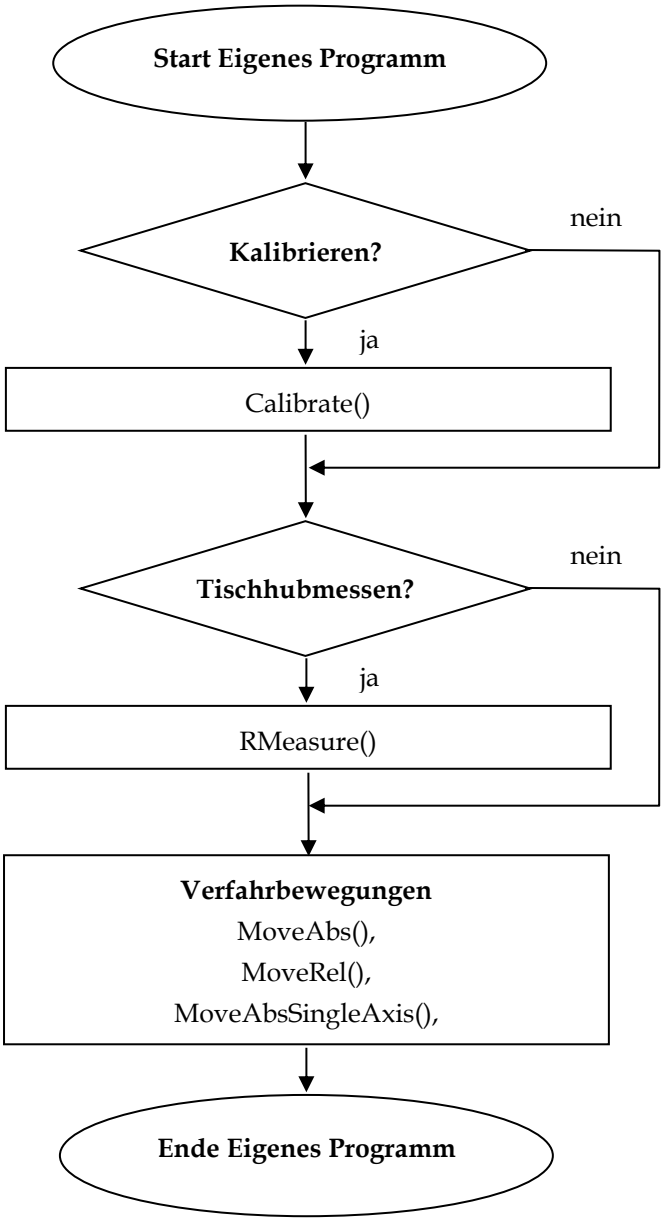
3.2.3 LSTEP express / LSTEP-PCI express Steuerungen

Die LSTEP express hat wesentlich mehr Einstellungsmöglichkeiten als die Steuerungen der LSTEP 2000 Familie, sodass nicht alle Elemente als gesonderte API-Funktion zur Verfügung stehen. Daher bietet es sich an die Konfiguration der LSTEP express

mit dem WIN-Commander 5 durchzuführen. Diese Konfiguration kann in eine Datei gespeichert werden indem man im WIN-Commander 5 den Menüpunkt „Konfiguration exportieren“ unter Hauptmenü → Steuerung verwendet. Nach dem Export kann diese Datei mit dem API-Befehl LoadConfig gelesen und mit dem Befehl SetControlPars gesendet werden. Mit dem API-Befehl LStepSave wird die Konfiguration in die Steuerung gespeichert, so dass sie nach dem nächsten Einschalten der Steuerung sofort geladen wird. Das Speichern der Einstellungen in der Steuerung kann auch aus dem WIN-Commander heraus erfolgen, was ein manuelles Laden der Datei zum Programmstart erübrigt.

3.3 Eigener Programmteil

Im eigenen Programmteil kann der Anwender die gewünschte Funktionalität der Steuerung programmieren. Dazu zählen das Ausführen von Positionierbewegungen in Abhängigkeit der Zustände von digitalen I/O‘s ebenso wie das Setzen von Triggersignalen in Abhängigkeit von der Position usw.



4 Funktionen

4.1 Kurzbeschreibung der API-Befehle

Aufbau der Tabelle:

Die nachfolgenden Tabellen enthalten Kurzbescheibungen zu den einzelnen API-Funktionen. Hierzu ist in der Spalte „Befehl“ der Funktionsname gelistet. In der nächsten Spalte wird diese Funktion kurz beschrieben. Die Spalte mit der Kennzeichnung „X“ zeigt an, welche Steuerung diesen API-Befehl unterstützt. Diese Kennzeichnung baut sich wie folgt auf:

Werte der Spalte „X“: 1 = LSTEP; 2 = LSTEP express; 3 = beide Steuerungen

In der letzten Spalte ist eine Verlinkung bzw. die Seitenzahl zur Detailbeschreibung der Funktion zu finden.

Achtung!

In dieser Kurzbeschreibung sind nur die DLL-Aufrufe beschrieben.

Alle Befehle, die nicht als API-Funktion vorhanden sind, müssen mit dem DLL-Aufruf „SendString“ gesendet werden. Welche Funktionen zur Verfügung stehen sowie deren Beschreibung ist in der entsprechenden Dokumentation der Steuerung nachzuschlagen.

4.1.1 API-Konfiguration / Schnittstellen-Konfiguration

Befehl	Kurzbeschreibung	X	Seite
CreateLSID	Erzeugt eine ID Nummer	3	39
FreeLSID	Gibt die erzeugte ID Nummer wieder frei	3	40
Connect	Mit der LSTEP verbinden	3	41
ConnectEx ConnectExW	Mit der LSTEP verbinden	3	41
ConnectSimple ConnectSimpleW	Mit der LSTEP verbinden (Empfohlen)	3	42
Disconnect	Verbindung zur LSTEP trennen	3	43
SetExtValue	Erweiterungen einschalten	3	44
SetProcessMessagesProc	Ermöglicht das Ersetzen der internen Message-Dispatching Prozedur der LSTEP-API	3	45
SetOsziCallBackFct	Übergibt eine globale Callback-Funktion für asynchrone Ereignisse an die API. (Siehe Kapitel 5.1)	2	46
SetExtCallBackFct	Übergibt eine Callback-Funktion für asynchrone Ereignisse an die API. (Siehe 5.2)	2	47
SetLanguage SetLanguageW	Sprachumschaltung der LSTEP-API (Protokoll/Meldungen)	3	48

Befehl	Kurzbeschreibung	X	Seite
LSX_SetSysSampleRate	System-Ausgabefrequenz der Steuerung auslesen	2	63
LSX_GetBaud	Auslesen der Baudrate	2	64
LSX_SetBaud	Einstellen der Baudrate	2	64

4.1.2 Steuerungs- und API-Informationen

Befehl	Kurzbeschreibung	X	Seite
GetAPIVersion GetAPIVersionW	Liefert Versionsnummer der LSTEP-API	3	65
GetSerialNr GetSerialNrW	Seriennummer der Steuerung auslesen	1	66
GetVersionStr GetVersionStrW	Liefert die aktuelle Versionsnummer der Firmware zurück	3	67
GetVersionStrDet GetVersionStrDetW	Konfiguration der Firmware auslesen	1	68
GetVersionStrInfo GetVersionStrInfoW	Ergänzung zur aktuellen Versionsnummer auslesen	3	69

4.1.3 Statusabfragen

Befehl	Kurzbeschreibung	X	Seite
GetError	Liefert die aktuelle Fehlernummer	3	69
LSX_Quit	Quittiert aufgetretene Fehler	2	70
GetSecurityErr	Liest alle Zustände und Ergebnisse der GAL-Sicherheits-überwachung (nur bei LS44-Steuerungen)	1	70
GetSecurityStatus	Liest den aktuellen Zustand der Sicherheitsüberwachung	1	72
GetStatus GetStatusW	Liefert den aktuellen Zustand der Steuerung	3	73
GetStatusAxis GetStatusAxisW	Liefert den aktuellen Zustand der einzelnen Achsen	3	74
SetAutoStatus	Schaltet den AutoStatus ein/aus.	3	75
GetStatusLimit GetStatusLimitW	Liefert den aktuellen Zustand der Software-Grenzen jeder einzel-nen Achse	3	76
LSX_GetSysstat, LSX_Get-SysStatus	Auslesen des aktuellen Systemstatuses der Achsen	2	76
LSX_GetStopStatus	Auslesen des Stopp-Eingang-Zustandes	2	78
LSX_GetPowerAmplifier-Status	Auslesen der Endstufen-Zustände inkl. Ein- und Ausschaltpro-zesse	2	78
LSX_GetPTemp	Auslesen der Endstufen-Temperatursensoren	2	79

4.1.4 Parameterhandling

Befehl	Kurzbeschreibung	X	Seite
SetControlPars	Überträgt die mit LSX_LoadConfig geladenen Parameter an die LSTEP	3	80
LStepSave	Aktuelle Konfiguration in die LSTEP speichern (EEPROM)	3	80
SoftwareReset	Führt einen Neustart der Steuerung durch	3	81

4.1.5 Achs- und Motorkonfiguration

Befehl	Kurzbeschreibung	X	Seite
LSX_ValidConfig	Aktivierung aller Parameter mit ValidConfig-Abhänigkeit	2	82
LSX_ValidPar	Parameterübergabe an die Steuerung	2	82
ConfigMaxAxes	Stellt Anzahl der verwendeten Achsen ein	3	83
GetDimensions	Fragt die Dimensionen der Achsen ab	3	83
SetDimensions	Stellt die Dimensionen der Achsen ein	3	84
GetActiveAxes	Liefert die freigegebenen Achsen	3	84
SetActiveAxes	Setzt die freigegebenen Achsen	3	85
LSX_GetAxisMap	Zuordnung der Hardwareachsen zu den Achsen des User-Interface auslesen	2	86
LSX_SetAxisMap	Zuordnen der Hardwareachsen zu den Achsen des User-Interface	2	86
GetAxisDirection	Fragt die Drehrichtungs-Umkehr ab	3	86
SetAxisDirection	Setzt die Drehrichtungs-Umkehr	3	87
GetGear	Liest den Getriebefaktor	1	88
SetGear	Setzt den Getriebefaktor	1	88
GetGearDenominator	Fragt den Nenner der Getriebeübersetzung ab	2	89
SetGearDenominator	Stellt den Nenner der Getriebeübersetzung ein	2	89
GetGearNumerator	Fragt den Zähler der Getriebeübersetzung ab	2	90
SetGearNumerator	Stellt den Zähler der Getriebeübersetzung ein	2	90
GetMotorCurrent	Fragt den Motorstrom ab	3	91
SetMotorCurrent	Stellt den Motorstrom ein	3	91
LSX_GetMotorpeakCurrent	Fragt den maximalen Motorstrom ab	2	92
LSX_SetMotorpeakCurrent	Stellt den maximalen Motorstrom ein	2	92
LSX_GetMotortempSensor	Fragt die Werte der Motortemperatursensoren ab	2	92
LSX_SetMotortempSensor	Stellt die Motortemperatursensoren ein	2	93
LSX_GetMotortempSensor-min	Fragt den minimal erlaubten Motortemperaturwert ab	2	93

Befehl	Kurzbeschreibung	X	Seite
LSX_SetMotorIITCheck	Status der Motor-I²t- Überwachung einstellen	2	120
LSX_GetMotorpeakCurrentTime	Motorspitzenstromdauer zur I²T-Überwachung auslesen	2	121
LSX_SetMotorpeakCurrentTime	Motorspitzenstromdauer zur I²T-Überwachung einstellen	2	121
LSX_GetMotorAutoKommCurrent	Motorstrom zum Autokommutieren im Servobetrieb auslesen	2	122
LSX_SetMotorAutoKommCurrent	Motorstrom zum Autokommutieren im Servobetrieb einstellen	2	122

4.1.6 Kinematik

Befehl	Kurzbeschreibung	X	Seite
GetAccel	Fragt die Beschleunigung ab	3	123
SetAccel	Stellt die Beschleunigung ein	3	123
SetAccelSingleAxis	Stellt die Beschleunigung einer Achse ein	3	125
GetAccelJerk	Fragt den Ruck für die Beschleunigung ab	2	125
SetAccelJerk	Stellt den Ruck für die Beschleunigung ein	2	126
GetDeceleration	Fragt die Verzögerung ab	2	126
SetDeceleration	Stellt die Verzögerung ein	2	127
SetDecelSingleAxis	Stellt die Verzögerung einer Achse ein	2	127
GetDecelJerk	Fragt den Ruck während der Verzögerung ab	2	128
SetDecelJerk	Stellt den Ruck während der Verzögerung ein	2	128
GetVel	Fragt die Geschwindigkeit aller Achsen ab	3	129
SetVel	Stellt die Geschwindigkeit aller Achsen ein	3	129
SetVelSingleAxis	Stellt die Geschwindigkeit für eine Achse ein	3	130
GetVelFac	Fragt die Geschwindigkeitsuntersetzung aller Achsen ab	1	130
SetVelFac	Stellt die Geschwindigkeitsuntersetzung aller Achsen ein	1	131
SetVelFacSingleAxis	Stellt die Geschwindigkeitsuntersetzung einer Achse ein	1	131
GetVLevel	Liefert die ausgeblendete Geschwindigkeit	1	132
SetVLevel	Setzt die ausgeblendeten Geschwindigkeiten, bei denen Resonanzen auftreten	1	133
GetSpeedPoti	Gibt an, ob Speed-Poti Ein oder Aus ist.	1	134
SetSpeedPoti	Setzt das Speed-Poti Ein oder Aus	1	134
GetStopAccel	Liefert die Bremsbeschleunigung bei aktivem Stopp	1	135
SetStopAccel	Setzt die Bremsbeschleunigung bei aktivem Stopp	1	135

Befehl	Kurzbeschreibung	X	Seite
GetStopDecel	Liest den Wert ein, mit dem die Achse im Falle eines Stoppsignals bremsen soll	2	136
SetStopDecel	Stellt den Wert ein, mit dem die Achse im Falle eines Stoppsignals bremsen soll	2	136
GetStopDecelJerk	Fragt den Ruck während Verzögerung des Systems im Falle eines Notstoppsignals ab	2	137
SetStopDecelJerk	Stellt den Ruck während Verzögerung des Systems im Falle eines Notstoppsignals ein	2	137
LSX_GetRoomAccelJerk	Ruck beim Beschleunigen im Raum auslesen	2	138
LSX_SetRoomAccelJerk	Ruck beim Beschleunigen im Raum einstellen	2	138
LSX_GetRoomDecelJerk	Ruck beim Verzögern im Raum auslesen	2	139
LSX_SetRoomDecelJerk	Ruck beim Verzögern im Raum einstellen	2	139
LSX_GetRoomStopJerk	Ruck bei einem Notstopp-Ereignis im Raum auslesen	2	140
LSX_SetRoomStopJerk	Ruck bei einem Notstopp-Ereignis im Raum einstellen	2	140
LSX_GetRoomAccel	Beschleunigung im Raum auslesen	2	141
LSX_SetRoomAccel	Beschleunigung im Raumeinstellen	2	141
LSX_GetRoomDecel	Verzögerung im Raum auslesen	2	142
LSX_SetRoomDecel	Verzögerung im Raum einstellen	2	142
LSX_GetRoomStopDecel	Verzögerung bei einem Notstopp-Ereignis auslesen	2	143
LSX_SetRoomStopDecel	Verzögerung bei einem Notstopp-Ereignis einstellen	2	143
LSX_GetRoomVel	Geschwindigkeit im Raum auslösen	2	144
LSX_SetRoomVel	Geschwindigkeit im Raum einstellen	2	144
LSX_GetSpeedEnable	Freigabe Geschwindigkeits- bzw. Drehzahlvorwahl auslesen	2	145
LSX_SetSpeedEnable	Freigabe Geschwindigkeits- bzw. Drehzahlvorwahl einstellen	2	145
LSX_GetSpeed	Geschwindigkeits- bzw. Drehzahlvorwahl auslesen	2	146
LSX_SetSpeed	Geschwindigkeits- bzw. Drehzahlvorwahl einstellen	2	146

4.1.7 Endschalter und Softwarelimits

Befehl	Kurzbeschreibung	X	Seite
GetLimitControl	Liest, ob die Bereichsüberwachung eingeschaltet ist	3	147
SetLimitControl	Stellt die Bereichsüberwachung ein	3	147
GetLimitControlMode	Liefert den Modus für die Überwachung der Softwarelimits	1	148
SetLimitControlMode	Setzt den Modus für die Überwachung der Softwarelimits	1	148
GetAutoLimitAfterCalibRM	Gibt an, ob beim Kalibrieren und Tischhubmessen die interne Software-Limits gesetzt werden	3	149

Befehl	Kurzbeschreibung	X	Seite
SetAutoLimitAfterCalibRM	Verhindert, dass beim Kalibrieren und Tischhubmessen die internen Software-Limits gesetzt werden	3	150
GetLimit	Liefert Verfahrbereichsgrenzen	3	150
SetLimit	Stellt Verfahrbereichsgrenzen ein	3	151
GetSwitchActive	Gibt an, ob die Endschalter eingeschaltet sind	3	152
SetSwitchActive	Setzt den Status, ob die Endschalter Ein oder Aus geschaltet sind	3	152
GetSwitchPolarity	Liest die Endschalterpolarität	3	153
SetSwitchPolarity	Stellt die Endschalterpolarität ein	3	154
GetSwChange	Zeigt die Einstellungen der Endschalter	2	154
SetSwChange	Setzt den Wert, ob die Endschalter getauscht werden sollen	2	155
GetSwitches	Liest den Zustand aller Endschalter	3	156
LSX_GetStopSwitchOffDelay	Liest Verzögerung der Endstufenabschaltung bei aktivem Stopp-Eingang aus	2	156
LSX_SetStopSwitchOffDelay	Zeigt Verzögerung der Endstufenabschaltung bei aktivem Stopp-Eingang an	2	157
LSX_GetMonitoringVelFilter	Auslesen der Zeitkonstante für Istwertfilter der Geschwindigkeitsüberwachungen	2	157
LSX_SetMonitoringVelFilter	Setzen der Zeitkonstante für Istwertfilter der Geschwindigkeitsüberwachungen	2	157
LSX_GetHaltSignalVel	Auslesen des Geschwindigkeitslevel für Stillstandssignal	2	158
LSX_SetHaltSignalVel	Setzen des Geschwindigkeitslevel für Stillstandssignal	2	158
LSX_GetThresholdSignalVel	Auslesen des Geschwindigkeits-Schwellwerts	2	159
LSX_SetThresholdSignalVel	Setzen des Geschwindigkeits-Schwellwerts	2	159
LSX_GetCSOffset	Koordinatensystem-Offset auslesen	2	160
LSX_SetCSOffset	Koordinatensystem-Offset einstellen	2	160
LSX_GetCalibRMOffsetSWAct	Status der Endschalterauswertung während Kalibrier- und Tischhub-Offset auslesen	2	161
LSX_SetCalibRMOffsetSWAct	Status der Endschalterauswertung während Kalibrier- und Tischhub-Offset einstellen	2	161

4.1.8 Referenzfahrten

Befehl	Kurzbeschreibung	X	Seite
GetCalibRMAccel	Fragt die Beschleunigung während des Kalibriervorgangs ab	2	162
SetCalibRMAccel	Stellt die Beschleunigung während des Kalibriervorgangs ein	2	162
GetCalibRMJerk	Fragt den Ruck während des Kalibriervorgangs ab	2	163
SetCalibRMJerk	Stellt den Ruck während des Kalibriervorgangs ein	2	163

Befehl	Kurzbeschreibung	X	Seite
GetCalibBackSpeed	Liefert die Geschwindigkeit, mit der aus den Endschaltern gefahren wird	1	164
SetCalibBackSpeed	Stellt die Verfahrgeschwindigkeiten für das Herausfahren aus den Endschaltern während des Kalibriervorgangs ein	1	164
GetCalibRMBackSpeed	Liefert die Geschwindigkeit, mit der aus den Endschaltern gefahren wird	2	165
SetCalibRMBackSpeed	Stellt die Verfahrgeschwindigkeiten für das Herausfahren aus den Endschaltern während des Kalibriervorgangs ein	2	165
GetCalibRMVel	Fragt die Verfahrgeschwindigkeit währen des Kalibriervorgangs ab	2	166
SetCalibRMVel	Stellt die Verfahrgeschwdigkeiten während des Kalibriervorgangs ein	2	166
GetRefSpeed	Fragt die Geschwindigkeit ab, mit der beim Kalibrieren die Referenzmarke gesucht wird	1	167
SetRefSpeed	Setzt die Geschwindigkeit, mit der beim Kalibrieren die Referenzmarke gesucht wird	1	167
GetCalibOffset	Fragt den Kalibrier-Offset ab	3	168
SetCalibOffset	Stellt den Kalibrier-Offset ein	3	168
GetRMOffset	Fragt den eingestellten RM-Offset ab	3	169
SetRMOffset	Stellt den RM-Offset ein	3	169
GetCalibrateDir	Liefert, ob die Vorzeichen-Umkehr beim Kalibrieren eingestellt ist	3	170
SetCalibrateDir	Setzt die Vorzeichen-Umkehr für das Kalibrieren	3	170
Calibrate	Startet das Kalibrieren für alle aktiven Achsen	3	171
CalibrateEx	Startet das Kalibrieren für bestimmte Achsen	3	171
RMeasure	Startet das Tischhubmessen für alle aktiven Achsen	3	172
RMeasureEx	Startet das Tischhubmessen für bestimmte Achsen	3	172

4.1.9 Fahrbefehle und Positionsverwaltung

Befehl	Kurzbeschreibung	X	Seite
GetPos	Fragt die aktuellen Positionen aller Achsen ab	3	173
GetPosEx	Fragt die aktuellen Geber- bzw. Positionswerte aller Achsen ab	3	173
GetPosSingleAxis	Fragt die aktuelle Position einer Achse ab	3	174
SetPos	Setzt die Position aller Achsen	3	174
ClearPos	Setzt die Positionswerte auf Null (für endlos Drehachsen)	1	175
GetDelay	Liefert die Verzögerung des Vektorstarts	1	175
SetDelay	Setzt die Verzögerung des Vektorstarts	1	176
GetDistance	Liefert die Strecke, die mit MoveRelShort gestartet wird	3	176

Befehl	Kurzbeschreibung	X	Seite
LSX_GetMixedMoveAxis-Mode	Gemischte Verfahrbewegung - Achsen Modus auslesen	2	194
LSX_SetMixedMoveAxis-Mode	Gemischte Verfahrbewegung - Achsen Modus einstellen	2	194
LSX_MixedMoveAbs	Gemischte Verfahrbewegung - Position absolut	2	195
LSX_MixedMoveRel	Gemischte Verfahrbewegung - Position relativ	2	195
LSX_GetMixedMovePos	Gemischte Verfahrbewegung - Position auslesen	2	197
LSX_SetMixedMovePos	Gemischte Verfahrbewegung - Position einstellen	2	197
LSX_GetMixedMo-veAmpl	Gemischte Verfahrbewegung - Amplitude / Skalierungfaktor aus-lesen	2	198
LSX_SetMixedMoveAmpl	Gemischte Verfahrbewegung - Amplitude / Skalierungfaktor ein-stellen	2	198
LSX_GetIndexTableDivi-der	Teilungsfaktor für Teilkopfbetrieb auslesen	2	199
LSX_SetIndexTableDivi-der	Teilungsfaktor für Teilkopfbetrieb einstellen	2	199
LSX_MoveIndexTable	Absolutpositionierung auf übergebene Teilkopfposition	2	200
LSX_GetMoveSeqStatus-Pos	Bewegungsablauf auslesen	2	201
LSX_SetMoveSeqStatus-Pos	Bewegungsablauf auswählen	2	201
LSX_GetMoveSeqStatus	Status des Bewegungsablaufs auslesen	2	203
LSX_SetMoveSeqVar	Variablen des ausgewählten Bewegungsablaufs einstellen	2	204

4.1.10 Tabellenbefehle

Befehl	Kurzbeschreibung	X	Seite
LSX_GetTablePos	Tabellenpositionen auslesen	2	205
LSX_ MoveTablePosRel	Relativpositionierung mit Werten aus Tabellenposition	2	206

4.1.11 Joystick und Handrad

Befehl	Kurzbeschreibung	X	Seite
GetHandWheel	Liest den Zustand des Handrads	1	208
SetHandWheelOff	Deaktiviert das Handrad	1	209
SetHandWheelOn	Aktiviert das Handrad	1	209
GetDigJoySpeed	Liest die Geschwindigkeit des digitalen Joysticks	1	210
SetDigJoySpeed	Setzt die Geschwindigkeit des digitalen Joysticks	1	210

Befehl	Kurzbeschreibung	X	Seite
SetDigJoyOff	Schaltet digitalen Joystick aus	1	211
GetJoystick	Liest den Zustand des Analog-Joysticks	3	211
SetJoystickOff	Deaktiviert den Analog-Joystick	3	212
SetJoystickOn	Aktiviert den Analog-Joystick	3	212
GetJoystickFilter	Gibt an, ob die Filterung im Joystick-Betrieb aktiv ist	1	213
SetJoystickFilter	Aktiviert bzw. deaktiviert die Filterung im Joystick-Betrieb	1	213
GetJoystickWindow	Lesen des Joystick-Fenster	3	214
SetJoystickWindow	Setzen des Joystick-Fenster	3	214
GetJoyChangeAxis	Liest die Joystickachszuordnung	1	215
JoyChangeAxis	Setzt die Joystickachszuordnung	1	215
GetJoystickAxes	Zeigt für welche Achsen der Joystick eingeschaltet ist	2	216
SetJoystickAxes	Schaltet den Joystick für die angegebene Achsen ein	2	217
GetJoystickDir	Liest die eingestellte Richtung des Joysticks ein	3	218
SetJoystickDir	Setzt die Richtung des Joysticks	3	219
LSX_GetJoyVel	Fragt die maximalen Verfahrgeschwindigkeiten im Joystickbetrieb ab	2	220
LSX_SetJoyVel	Stellt die maximalen Verfahrgeschwindigkeiten im Joystickbetrieb ein	2	220
LSX_GetJoyRedcur	Liest aus, ob die Stromreduzierung der Achsen aktiv ist	2	221
LSX_SetJoyRedcur	Aktiviert/ Deaktiviert die Stromreduzierung der Achsen	2	221
LSX_GetJoytoAxis	Liest die Zuordnung der Achsen zu den Joystick-Eingängen aus	2	222
LSX_SetJoytoAxis	Ordnet den Joystick-Eingängen Achsen zu	2	222
LSX_GetManModePreselection	Liest die ausgewählte manuelle Betriebsart aus	2	223
LSX_SetManModePreselection	Setzt die manuelle Betriebsart	2	223
LSX_GetManModeLinktoAxis	Liest die Kopplung einer Achse aus	2	224
LSX_SetManModeLinktoAxis	Koppelt eine Achse an eine Andere	2	225
LSX_GetTipp	Liest aus, ob der Tippbetrieb aktiv ist	2	225
LSX_SetTipp	Aktiviert/ Deaktiviert den Tippbetrieb	2	226
LSX_GetTippEnable	Liest aus welche Achsen für den Tippbetrieb freigegeben sind	2	226
LSX_SetTippEnable	Gibt Achsen für den Tippbetrieb frei/Sperrt Achsen.	2	226
LSX_GetTippRedCur	Liest aus bei welchen Achsen die Stromreduzierung im Tippbetrieb aktiv ist	2	227

Befehl	Kurzbeschreibung	X	Seite
LSX_SetTippRedCur	Setzt die Stromreduzierung bei Achsen im Tippbetrieb aktiv	2	227
LSX_GetTippVel	Liest die Verfahrgeschwindigkeit im Tippbetrieb aus	2	227
LSX_SetTippVel	Setzt die Verfahrgeschwindigkeit im Tippbetrieb	2	228
LSX_GetTippDir	Liest die Verfahrrichtung im Tippbetrieb aus	2	228
LSX_SetTippDir	Setzt die Verfahrrichtung im Tippbetrieb	2	228
LSX_GetTippOutPass	Filterzeitkonstante für Tippbetrieb auslesen	2	229
LSX_SetTippOutPass	Filterzeitkonstante für Tippbetrieb einstellen	2	229
LSX_GetTrackBall	Trackballbetrieb auslesen	2	230
LSX_SetTrackBall	Ein- und Ausschalten des Trackballbetriebs	2	230
LSX_GetTrackBallEnable	Trackballbetrieb Achsen-Freigabe auslesen	2	231
LSX_SetTrackBallEnable	Trackballbetrieb Achsen-Freigabe einstellen	2	231
LSX_GetTrackBallRedCur	Auslesen der Stromreduzierung im Trackballbetrieb	2	232
LSX_SetTrackBallRedCur	Ein- und Ausschalten der Stromreduzierung im Trackballbetrieb	2	232
LSX_GetTrackBallVel	Maximale Geschwindigkeit im Trackballbetrieb auslesen	2	233
LSX_SetTrackBallVel	Maximale Geschwindigkeit im Trackballbetrieb einstellen	2	233
LSX_GetTrackBallOut-Pass	Filterzeitkonstante für Trackballbetrieb auslesen	2	234
LSX_SetTrackBallDir	Motordrehrichtungen im Trackballbetrieb einstellen	2	235
LSX_GetTrackBallToAxis	Zuordnung der Trackballachsen zu den Maschinenachsen auslesen	2	236
LSX_SetTrackBallToAxis	Zuordnung der Trackballachsen zu den Maschinenachsen einstellen	2	236

4.1.12 Bedienpult mit Trackball und Joyspeed-Tasten

Befehl	Kurzbeschreibung	X	Seite
GetBPZ	Liest den Zustand des Bedienpults	1	237
SetBPZ	Aktiviert bzw. deaktiviert das Bedienpult	1	237
GetBPZJoyspeed	Liest den Joystick-Speed des Bedienpults	1	239
SetBPZJoyspeed	Setzt den Joystick-Speed des Bedienpults	1	239
GetBPZTrackballBacklash	Liest das Trackball-Umkehrspiel des Bedienpults	1	240
SetBPZTrackballBacklash	Setzt das Trackball-Umkehrspiel des Bedienpults	1	240
LSX_GetBPZTrackballFactor	Liest das Trackball-Umkehrspiel des Bedienpults	1	241
LSX_SetBPZTrackballFactor	Setzt das Trackball-Umkehrspiel des Bedienpults	1	241

LSX_GetJoyOutPass	Liest die Filterzeitkonstante der Joystick-Funktion aus	2	242
LSX_SetJoyOutPass	Setzt die Filterzeikonstante der Joystick-Funktion	2	242

4.1.13 Digitale und analoge Ein- und Ausgänge

Befehl	Kurzbeschreibung	X	Seite
GetAnalogInput	Liest den aktuellen Zustands eines Analogkanals	3	244
GetAnalogInputs2	Liest die aktuellen Zustände der Analogkanäle PT100, MV und V24	1	244
SetAnalogOutput	Setzt einen Analogkanal	3	245
GetDigitalInputs	Liest die digitalen Eingänge (0-15)	3	245
GetDigitalInputsE	Liest die zusätzlichen digitale Eingänge (16-31)	1	246
SetDigitalOutput	Setzt einen digitalen Ausgang	3	246
SetDigitalOutputs	Setzt die digitalen Ausgänge (0-15)	3	247
SetDigitalOutputsE	Setzt die zusätzlichen digitalen Ausgänge (16-31)	1	247
SetDigIO_Distance	Setzt die Aktivierung eines Ausgangs in Abhängigkeit der eingestellten Strecke vor/nach der Zielposition	1	247
SetDigIO_EmergencyStop	Setzt die Zuordnung der digitalen Ein-/Ausgänge zum Not-Stopp-Pin	1	249
SetDigIO_Off	Stellt die Funktion der digitalen Ein-/Ausgänge aus	1	250
SetDigIO_Polarity	Stellt die Polarität zu den Funktionen der digitalen Ausgänge ein	1	250
LSX_GetDigOutLinktoSignal	Liest, ob dem Ausgang ein Signal zugeordnet ist.	2	251
LSX_SetDigitalOutLinktoSignal	Ordnet einem Ausgang ein Signal zu.	2	252
LSX_GetInvertDigOutSignal	Liest aus, ob ein Signal invertiert werden soll.	2	253
LSX_SetInvertDigOutSignal	Invertiert ein Signal	2	253
LSX_GetDigInStatus	Auslesen der digitalen Eingänge 0 - 15 (24 V)	2	254
LSX_SetDigInStatus	Automatisches Senden der digitalen Eingänge 0 - 15 (24 V)	2	255
LSX_GetDigInExtStatus	Auslesen der digitalen Eingänge 16 – 32 (24 V)	2	256
LSX_SetDigInExtStatus	Automatisches Senden der digitalen Eingänge 16 – 32 (24 V)	2	257
LSX_GetTTLDigIn	Digitale Eingänge (TTL) einlesen	2	258
LSX_GetTTLDigOut	Digitale Eingänge (TTL) einstellen	2	258
LSX_SetTTLDigOut	Digitale Ausgänge (TTL) setzen	2	258
LSX_GetMotorBrakeDigOut	Motorbremsenausgang (im Modus "Digitaler Ausgang") auslesen	2	259
LSX_SetMotorBrakeDigOut	Motorbremsenausgang (im Modus "Digitaler Ausgang") setzen	2	259

Befehl	Kurzbeschreibung	X	Seite
LSX_GetMotorBrakeOut	Zustand der Motorbremsenausgänge einlesen	2	260
LSX_GetTVR	Auslesen des Status des Takt-V/R – Betrieb	2	260
LSX_Set	Ein- und Ausschalten des Takt-V/R – Betrieb	2	260
LSX_GetTVRToAxis	Takt-Vor/Rück-Achsen-Zuordnung auslesen	2	261
LSX_SetTVRToAxis	Takt-Vor/Rück-Achsen-Zuordnung einstellen	2	261

4.1.14 Takt-Vor/Rück In

Befehl	Kurzbeschreibung	X	Seite
GetFactorTVR	Liest den Takt Vor / Rück Faktor	3	262
SetFactorTVR	Setzt den Takt Vor / Rück Faktor	3	262
GetTVRMode	Liest den Takt Vor / Rück Modus ein	1	263
SetTVRMode	Stellt den Takt Vor / Rück Modus ein	1	264
SetTVRInPulse	Setzt ein Takt-Vor/Rück Impuls über die Schnittstelle ab	1	265

4.1.15 Takt-Vor/Rück Ausgänge für weitere Achsen

Befehl	Kurzbeschreibung	X	Seite
GetAccelTVRO	Liest alle eingestellten Beschleunigungen des TVRO	1	266
SetAccelTVRO	Setzt die Beschleunigungen des TVRO	1	266
SetAccelSingleAxisTVRO	Setzt die einzelnen Beschleunigungen des TVRO	1	267
GetVelTVRO	Liest alle eingestellten Geschwindigkeiten des TVRO	1	267
SetVelTVRO	Setzt alle Geschwindigkeiten des TVRO	1	268
SetVelSingleAxisTVRO	Setzt die Geschwindigkeit einer einzelnen TVRO-Achse	1	268
GetPosTVRO	Liefert TVRO Positionswerte, in Abhängigkeit von Dimension	1	269
SetPosTVRO	Setzt die TVRO Position	1	269
GetStatusTVRO GetStatusTVROW	Liefert den aktuellen Status der Achsen	1	270
GetTVROutMode	Liest den eingestellten TVRO Modus aus	1	271
SetTVROutMode	Stellt den TVRO Modus ein	1	271
GetTVROutPitch	Liest die TVRO Spindelsteigung	1	272
SetTVROutPitch	Setzt die TVRO Spindelsteigung	1	272
GetTVROutResolution	Liefert die Auflösung der anzusteuernden TVRO-Endstufe	1	273
SetTVROutResolution	Setzt die Auflösung der anzusteuernden TVRO-Endstufe	1	273
MoveAbsTVRO	Fährt eine Absolutposition mit allen TVRO-Achsen an	1	274
MoveAbsTVROSingleAxis	Fährt eine Absolutposition mit einer TVRO-Achse an	1	275
MoveRelTVRO	Fährt einen relativen Vector mit allen TVRO-Achse	1	275
MoveRelTVROSingleAxis	Fährt einen relativen Vector mit einer TVRO-Achse	1	276

4.1.16 Gebereinstellungen

Befehl	Kurzbeschreibung	X	Seite
ClearEncoder	Setzt die Geberposition auf Null	1	277
GetEncoder	Liest alle Geberpositionen	1	277
GetEncoderActive	Liest welche Geber nach dem Kalibrieren aktiv werden	1	278
SetEncoderActive	Setzt welche Geber nach dem Kalibrieren aktiviert werden sollen	1	278
GetEncoderMask	Liest die Geberzustände aus	3	279
SetEncoderMask	Aktiviert bzw. deaktivert die Geber	1	280
GetEncoderPeriod	Liest die eingestellte Geberperiodenlängen aus	3	280
SetEncoderPeriod	Stellt die Geberperiodenlängen ein	3	281
GetEncoderPosition	Liefert die eingestellte Positionsquelle	3	282
SetEncoderPosition	Stellt die Positionsquelle ein	3	282
GetEncoderRefSignal	Liest aus ob beim Kalibrieren das Referenzsignal von Geber ausgewertet werden soll	3	283
SetEncoderRefSignal	Stellt ein ob beim Kalibrieren das Referenzsignal des Gebers ausgewertet werden soll	3	283
LSX_GetEncDir	Fragt die Lagegeber Zählrichtung ab	2	284
LSX_SetEncDir	Stellt die Lagegeberzählrichtung ein	2	284
LSX_GetEncPolePairs	Fragt die Anzahl der Lagegeberpolpaare pro Umdrehung ab	2	284
LSX_SetEncPolePairs	Stellt die Anzahl der Lagegeberpolpaare pro Umdrehung ein	2	285
LSX_GetEnctoAxis	Fragt die Zuordnung der Gebereingänge ab	2	285
LSX_SetEnctoAxis	Stellt die Zuordnung der Gebereingänge ein	2	286
LSX_GetEncType	Fragt den Type des Lagegebers ab	2	286
LSX_SetEncType	Stellt den Typ des Lagegebers ein	2	287
LSX_GetKommMode	Zeigt den Kommutierungsmodus im Servobetrieb an	2	288
LSX_SetKommMode	Stellt den Kommutierungsmodus im Servobetrieb ein	2	288
LSX_GetKommEncDir	Zeigt die Kommutierungsgeber Zählrichtung an	2	289
LSX_SetKommEncDir	Stellt die Kommutierungsgeber Zählrichtung ein	2	289
LSX_GetKom-mEncPolePairs	Zeigt die Kommutierungsgeberpolpaare pro Umdrehung an	2	289
LSX_SetKom-mEncPolePairs	Stellt die Kommutierungsgeberpolpaare pro Umdrehung ein	2	290
LSX_GetKommEnctoAxis	Zeigt die Zuordnungen der Gebereingänge an	2	290
LSX_SetKommEnctoAxis	Stellt die Zuordnungen der Gebereingänge ein	2	291
LSX_GetKommEncType	Zeigt den Kommutierungsgebertyp an	2	291
LSX_SetKommEncType	Stellt den Kommutierungsgebertyp ein	2	292

4.1.17 Reglereinstellungen

Befehl	Kurzbeschreibung	X	Seite
GetController	Liest den eingestellten Regler-Modus aus	1	293
SetController	Stellt den Regler-Modus ein	1	294
GetControllerCall	Liest die Einstellung vom Regleraufruf aus	1	294
SetControllerCall	Stellt den Regleraufruf ein	1	295
GetControllerFactor	Liest die Einstellung vom Reglerfaktor aus	1	295
SetControllerFactor	Stellt den Reglerfaktor ein	1	296
GetControllerSteps	Liest die Regler-Schritte aus	1	296
SetControllerSteps	Stellt die Regler-Schritte ein	1	297
GetControllerTimeout	Liefert die Einstellung des Regler-Überwachungs-Timeouts	1	297
SetControllerTimeout	Setzt den Regler-Überwachungs-Timeout	1	298
GetControllerTWDelay	Liest die Reglerverzögerung aus	1	298
SetControllerTWDelay	Stellt die Reglerverzögerung ein	1	299
GetCtrFastMove	Liest die Einstellung der Fast Move Funktion ein	1	299
SetCtrFastMoveOff	Stellt die Fast Move Funktion aus	1	300
SetCtrFastMoveOn	Stellt die Fast Move Funktion ein	1	300
GetCtrFastMoveCounter	Liest die Anzahl ausgeführter FastMove Funktionen aus	1	301
ClearCtrFastMoveCounter	Setzt die Anzahl ausgeführter FastMove Funktionen auf Null	1	301
GetTargetWindow	Liefert das Reglerzielfenster	3	302
SetTargetWindow	Stellt das Reglerzielfenster ein	3	302
LSX_GetDeviationRange	Fragt den maximal zulässigen Schleppfehler ab	2	302
LSX_SetDeviationRange	Stellt den maximal zulässigen Schleppfehler ein	2	303
LSX_GetDeviationTime	Fragt die Zeit bis die Schleppfehlerüberwachung angesprochen wird ab	2	303
LSX_SetDeviationTime	Stellt die Zeit bisw die Shleppfehlerüberwachung angesprochenwird ein	2	303
LSX_GetDeviationCheck	Fragt den Zustand der Schleppfehlerüberwachung ab	2	304
LSX_SetDeviationCheck	Schaltet die Schleppfehlerüberwachung ein/aus	2	304
LSX_GetDeviationValue	Fragt den Schleppfehler ab	2	304
LSX_GetPoscon	Fragt die Lagereglereistellung ab	2	306
LSX_SetPoscon	Stellt den Lageregler ein	2	306
LSX_GetPosConKP	P-Anteil des Lagereglers auslesen	2	307
LSX_SetPosConKP	P-Anteil des Lagereglers einstellen	2	307
LSX_GetPosConAdaptiveKP	P-Anteil des adaptiven Lagereglers auslesen	2	308

Befehl	Kurzbeschreibung	X	Seite
LSX_SetPosConAdaptiveKP	P-Anteil des adaptiven Lagereglers einstellen	2	308
LSX_GetPosConKI	I-Anteil des Lagereglers auslesen	2	309
LSX_SetPosConKI	I-Anteil des Lagereglers einstellen	2	309
LSX_GetPosConAdaptiveKI	I-Anteil des adaptiven Lagereglers auslesen	2	310
LSX_SetPosConAdaptiveKI	I-Anteil des adaptiven Lagereglers einstellen	2	310
LSX_GetPosConOutPass	Zeitkonstante des Lageregler-Ausgangsfilters auslesen	2	311
LSX_SetPosConOutPass	Zeitkonstante des Lageregler-Ausgangsfilters einstellen	2	311
LSX_GetPosConAdaptiveOutPass	Zeitkonstante des adaptiven Lageregler-Ausgangsfilters auslesen	2	312
LSX_SetPosConAdaptiveOutPass	Zeitkonstante des adaptiven Lageregler-Ausgangsfilters einstellen	2	312
LSX_GetPosConNominalSpeed	Nenngeschwindigkeit auslesen	2	313
LSX_SetPosConNominalSpeed	Nenngeschwindigkeit einstellen	2	313
LSX_GetPosConAdaptiveSpeed	Adaptive Nenngeschwindigkeit auslesen	2	314
LSX_SetPosConAdaptiveSpeed	Adaptive Nenngeschwindigkeit einstellen	2	314
LSX_GetPosConEnable	Achsenfreigabe für Lageregler auslesen	2	315
LSX_SetPosConEnable	Achsenfreigabe für Lageregler einstellen	2	315
LSX_GetSpeedConKP	P-Anteil des Drehzahlreglers auslesen	2	316
LSX_SetSpeedConKP	P-Anteil des Drehzahlreglers einstellen	2	316
LSX_GetSpeedConKI	I-Anteil des Drehzahlreglers auslesen	2	317
LSX_SetSpeedConKI	I-Anteil des Drehzahlreglers einstellen	2	317
LSX_GetSpeedConKD	D-Anteil des Drehzahlreglers auslesen	2	318
LSX_SetSpeedConKD	D-Anteil des Drehzahlreglers einstellen	2	318
LSX_GetSpeedConOutPass	Zeitkonstante des Drehzahlregler-Ausgangsfilters auslesen	2	319
LSX_SetSpeedConOutPass	Zeitkonstante des Drehzahlregler-Ausgangsfilters einstellen	2	319
LSX_GetActSpeedFilterConst	Zeitkonstante des Drehzahlistwert-Filters auslesen	2	320
LSX_SetActSpeedFilterConst	Zeitkonstante des Drehzahlistwert-Filters einstellen	2	320
LSX_GetSpeedFeedForward	Drehzahl- bzw. Geschwindigkeitsvorsteuerung auslesen	2	321
LSX_SetSpeedFeedForward	Drehzahl- bzw. Geschwindigkeitsvorsteuerung einstellen	2	321
LSX_GetActAccelFilterConst	Zeitkonstante des Beschleunigungsistwert-Filters auslesen	2	322
LSX_SetActAccelFilterConst	Zeitkonstante des Beschleunigungsistwert-Filters einstellen	2	322
LSX_GetAccelFeedForward	Beschleunigungsvorsteuerung auslesen	2	323

Befehl	Kurzbeschreibung	X	Seite
LSX_SetAccelFeedForward	Beschleunigungsvorsteuerungeinstellen	2	323
LSX_GetAccelFeedForward-OutPass	Zeitkonstante des Beschleunigungsvorsteuerung-Filters auslesen	2	324
LSX_SetAccelFeedForward-OutPass	Zeitkonstante des Beschleunigungsvorsteuerung-Filters einstellen	2	324
LSX_GetSpeedConTN	Nachstellzeit des Drehzahlreglers auslesen	2	325

4.1.18 Trigger-Ausgang

Befehl	Kurzbeschreibung	X	Seite
GetTrigger	Liest die Einstellung vom Trigger	3	326
SetTrigger	Schaltet den Trigger ein bzw. aus	3	326
GetTriggerCount	Liest den Triggerzählerstand	3	327
SetTriggerCount	Setzt den Triggerzählerstand	3	327
GetTriggerPar	Liest die Trigger-Parameter aus	3	327
SetTriggerPar	Setzt die Trigger-Parameter	3	328
LSX_GetTriggerDim	Liest die Einheit für die Parameterübergabe aus	2	329
LSX_SetTriggerDim	Stellt die Einheit für die Parameterübergabe ein	2	329
LSX_GetTriggerSource	Liest die Triggerquelle aus	2	330
LSX_SetTriggerSource	Stellt eine Triggerquelle ein	2	330
LSX_GetTriggerHysterese	Liest die Triggerhysterese aus	2	331
LSX_SetTriggerHysterese	Stellt die Triggerhysterese ein	2	331
LSX_GetTriggerPLength	Liest die Triggersignal-Periodenlänge aus	2	332
LSX_SetTriggerPLength	Stellt die Triggersignal-Periodenlänge ein	2	332
LSX_GetTriggerPuls-Count	Liest die Trigger Pulsanzahl im Burst Modus aus	2	332
LSX_SetTriggerPulsCount	Stellt die Trigger Pulsanzahl im Burst-Modus ein	2	333
LSX_GetTrigger_Two	Liest die Einstellung vom zweiten Trigger	2	333
LSX_SetTrigger_Two	Schaltet den zweiten Trigger ein bzw. aus	2	333
LSX_GetTrig-ger_TwoCount	Liest den Triggerzählerstand des zweiten Triggers aus	2	334
LSX_SetTrig-ger_TwoCount	Setzt den Triggerzählerstand des zweiten Triggers	2	334
LSX_GetTrigger_TwoPar	Liest die Trigger-Parameter vom zweiten Trigger aus	2	335
LSX_SetTrigger_TwoPar	Setzt die Trigger-Parameter des zweiten Triggers	2	336
LSX_GetTrigger_TwoDim	Liest die Einheit für die Parameterübergabe des zweiten Triggers aus	2	336

Befehl	Kurzbeschreibung	X	Seite
LSX_SetTrigger_TwoDim	Stellt die Einheit für die Parameterübergabe des zweiten Triggers ein	2	337
LSX_GetTrigger_Two-Source	Liest die Triggerquelle des zweiten Triggers aus	2	337
LSX_SetTrigger_Two-Source	Stellt eine Triggerquelle des zweiten Triggers ein	2	338
LSX_GetTrig-ger_TwoHysterese	Liest die Triggerhysterese des zweiten Triggers aus	2	338
LSX_SetTrigger_TwoHys-terese	Stellt die Triggerhysterese des zweiten Triggers ein	2	339
LSX_GetTrig-ger_TwoPLength	Liest die Triggersignal-Periodenlänge des zweiten Triggers aus	2	339
LSX_SetTrig-ger_TwoPLength	Stellt die Triggersignal-Periodenlänge des zweiten Triggers ein	2	340
LSX_GetTrigger_TwoPul-sCount	Liest die Trigger Pulsanzahl des zweiten Triggers im Burst Modus aus	2	340
LSX_SetTrigger_TwoPuls-Count	Stellt die Trigger Pulsanzahl des zweiten Triggers im Burst-Modus ein	2	340

4.1.19 Snapshot-Eingang

Befehl	Kurzbeschreibung	X	Seite
GetSnapshot	Liest die Einstellung vom Snapshot ein	3	341
SetSnapshot	Schaltet den Snapshot ein bzw. aus	3	341
GetSnapshotFilter	Liest den Eingangsfilter aus	3	342
SetSnapshotFilter	Setzt den Eingangsfilter	3	342
GetSnapshotPar	Liest die Snapshot-Parameter aus	3	342
SetSnapshotPar	Setzt die Snapshot-Parameter	3	343
LSX_GetSnapshotSource	Liest die aktuell parametrierte Lagewertequelle aus	2	343
LSX_SetSnapshotSource	Setzt die Lagewertequelle	2	344
GetSnapshotCount	Liest den Snapshot-Zähler aus	3	345
GetSnapshotPos	Liest die Snapshot-Position aus	3	345
GetSnapshotPosArray	Liest das Snapshot-Positions Array aus	3	346

4.2 Detaillierte Funktionsbeschreibung

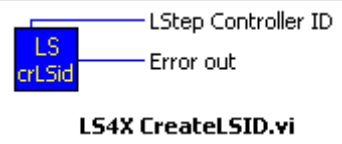
Die nachfolgenden detailierten Funktions-Beschreibungen enthalten einen beschreibenden Text der Funktion, eine Beschreibung der Funktions-Parameter, einen Beispielaufruf der Funktion sowie die Prototypen für die Programmiersprachen Delpi und C++. Weiterhin ist in der Zeile Sonstiges vermerkt, welche Steuerung die Funktion unterstützt, welcher Befehl zur Steuerung gesendet wird und ob eine Aktivierung des Befehls nötig ist.

Das Feld Aktivierung betrifft derzeit nur die LSTEP express Steuerungsserie.

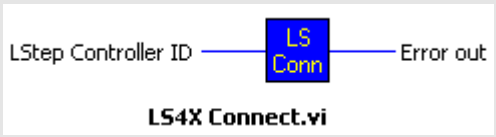
Ein Tabellenausschnitt für den Bereich Sonstiges ist nachfolgend zur besseren Erklärung gelistet:

Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	(Kompatibel mit X, siehe Abschnitt 4.1)	(Verwendeter Befehl aus dem Befehlssatz der Steuerung)	(Aktivierung über die ganannten „SendString" Befehle möglich)

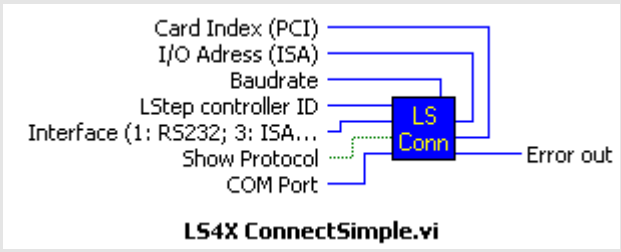
4.2.1 API-Konfiguration / Schnittstellen-Konfiguration

LSX_CreateLSID			
Beschreibung:	Erzeugt eine LSTEP-ID-Nummer. Diese wird als zusätzlicher Parameter bei den LSTEP-API- Befehlen verwendet, um aus mehreren angeschlossenen LSTEP-Steuerungen die LSTEP zu wählen, auf die sich der Befehl beziehen soll.		
Delphi:	function LSX_CreateLSID(var LSID: Integer): Integer;		
C++:	-		
LabView:			
Parameter:	LSID	Enthält nach Aufruf von CreateLSID eine neue LSTEP-ID-Nummer, die dann für Connect-, Verfahrbefehle und andere Befehle verwendet werden kann	
Beispiel:	var LStep1: Integer; ... LSX_CreateLSID(&LStep1);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

<table>
<tr><th colspan="4">LSX_FreeLSID</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Gibt eine erzeugte LSTEP-ID-Nummer wieder frei. Diese wird als zusätzlicher Parameter bei den LSTEP-API- Befehlen verwendet, um aus mehreren angeschlossenen LSTEP-Steuerungen die LSTEP zu wählen, auf die sich der Befehl beziehen soll. FreeLSID sollte erst nach Disconnect aufgerufen werden.</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_FreeLSID(LSID: Integer): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">-</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td>Parameter:</td><td>LSID</td><td colspan="2">Freizugebende LSTEP-ID-Nummer. Diese darf nach FreeLSID nicht mehr verwendet werden.</td></tr>
<tr><td>Beispiel:</td><td colspan="3">var LStep1: Integer;
...
LSX_CreateLSID(&LStep1);
LSX_ConnectSimple(LStep1, ...);
...
LSX_Disconnect(LStep1);
LSX_FreeLSID(LStep1);</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>3</td><td>-</td><td>-</td></tr>
</table>

LSX_Connect			
Beschreibung:	Diese Funktion baut eine Verbindung zu einer LSTEP-Steuerung auf. Dazu werden die Schnittstellen-Parameter verwendet, welche mittels LSX_LoadConfig aus der INI-Datei geladen wurden. (Eine der Funktionen LSX_Connect, LSX_ConnectSimple oder LSX_ConnectEx muss zur Initialisierung der Schnittstelle aufgerufen werden, damit die Kommunikation mit der LSTEP möglich ist.)		
Delphi:	function LSX_Connect(LSID: Integer): Integer;		
C++:	int Connect();		
LabView:			
Parameter:	-		
Beispiel:	LSX.LoadConfig("C:\LStepTest\LStep.INI"); LSX.Connect();		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

LSX_ConnectEx, LSX_ConnectExW			
Beschreibung:	Diese Funktion baut eine Verbindung zu einer LSTEP-Steuerung auf. Übergeben wird ein Zeiger auf eine Datenstruktur, die die Schnittstellenparameter enthält. In dem Record werden außerdem Informationen über die erkannte Steuerung zurückgeliefert (Versionsnummer...) (Eine der Funktionen LSX_Connect, LSX_ConnectSimple oder LSX_ConnectEx muss zur Initialisierung der Schnittstelle aufgerufen werden, damit die Kommunikation mit der LSTEP möglich ist.)		
Delphi:	function LSX_ConnectEx(LSID: Integer; var AControlInitPar: TLS_ControlInitPar): Integer; function LSX_ConnectExW(LSID: Integer; var AControlInitPar: TLS_ControlInitParW): Integer;		
C++:	int ConnectEx (TLS_ControlInitPar *pAControlInitPar); int ConnectExW (TLS_ControlInitParW *pAControlInitPar);		
LabView:	-		
Parameter:	AControlInitPar	Zeiger auf einen Record des Typs TLS_ControlInitPar bzw. TLS_ControlInitParW	
Beispiel:	LSX.ConnectEx(&ControlInitPar1);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

<table>
<tr><th colspan="4">LSX_ConnectSimple, LSX_ConnectSimpleW</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Diese Funktion baut eine Verbindung zu einer LSTEP-Steuerung auf. Die Einstellungen der Schnittstelle werden als Parameter übergeben.
(Eine der Funktionen LSX_Connect, LSX_ConnectSimple oder LSX_ConnectEx muss zur Initialisierung der Schnittstelle aufgerufen werden, damit die Kommunikation mit der LSTEP möglich ist.)</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_ConnectSimple(LSID: Integer; AnInterfaceType: Integer; AComName: PAnsiChar; ABaudRate: Integer; AShowProt: LongBool): Integer;
function LSX_ConnectSimpleW(LSID: Integer; AnInterfaceType: Integer; AComName: PWideChar; ABaudRate: Integer; AShowProt: LongBool): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int ConnectSimple (int lAnInterfaceType, char *pcAComName, int lABR, BOOL AShowProt);
int ConnectSimpleW (int lAnInterfaceType, TCHAR *pcAComName, int lABR, BOOL AShowProt);</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td rowspan="4">Parameter:</td><td>AnInterfaceType</td><td colspan="2">Schnittstellentyp
1 = RS232
2 = ArcNet
3 = DPRAM / ISA-Bus
4 = DPRAM / PCI-Bus
5 = RS232 mit RTS/CTS und erweitertem Protokoll
11= RS232 mit RTS/CTS</td></tr>
<tr><td>AComName</td><td colspan="2">Name der COM-Schnittstelle, z. B. ‘COM2‘, bei ArcNet oder DPRAM auf NULL setzen</td></tr>
<tr><td>ABR</td><td colspan="2">Bedeutung ist abhängig vom Schnittstellentyp
RS232 = Baudrate, z. B. 9600
ArcNet = 0 für Koax, 1 für Twisted Pair
DPRAM/ISA-Bus = Basis-I/O-Adresse, z.B. 0x0340
DPRAM/PCI-Bus = 0 für erste Karte, 1 für zweite Karte</td></tr>
<tr><td>AShowProt</td><td colspan="2">Schnittstellenprotokoll angezeigen
True = Anzeigen
False = Ausblenden</td></tr>
<tr><td>Beispiel:</td><td colspan="3">LSX.ConnectSimple(1, “COM2“, 9600, true); // RS232, 9600 Baud
LSX_ConnectSimple(4, nil, 0, true); //LSTEP PCI Karte 0;</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>3</td><td>-</td><td>-</td></tr>
</table>

<table>
<tr><th colspan="4">LSX_Disconnect</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Trennt die Verbindung zur LSTEP-Steuerung. Nach Aufruf dieser Funktion können keine Befehle mehr zur LSTEP gesendet werden. Die Funktion sollte kurz vor Beendigung des Programms aufgerufen werden.</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_Disconnect(LSID: Integer): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int Disconnect ();</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td>Parameter:</td><td colspan="3">-</td></tr>
<tr><td>Beispiel:</td><td colspan="3">LSX.Disconnect();</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>3</td><td>-</td><td>-</td></tr>
</table>

<table>
<tr><th colspan="4">LSX_SetExtValue</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Schaltet Erweiterungen des API ein, teilweise handelt es sich dabei um experimentelle Modi zu Debugging-Zwecken</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_SetExtValue(LSID: Integer; AName, AValue: Integer): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int SetExtValue (int lAName, int lAValue);</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td rowspan="6">Parameter:</td><td>AName</td><td colspan="2">Nummer der erweiterten Funktion</td></tr>
<tr><td>AValue</td><td colspan="2">Parameter</td></tr>
<tr><td>AName=2
(IFSleepTime)</td><td colspan="2">Stellt das Polling-Intervall für das DPRAM der LSTEP-PCI ein
AValue: Zeit-Intervall in [ms], Standard ist 10</td></tr>
<tr><td>AName=3
(ProtMoveOnly)</td><td colspan="2">Schaltet Filter für Log-Datei ein, durch welches nur Moves & Fehler protokolliert werden
AValue=1: Filter an
AValue=0: Filter aus</td></tr>
<tr><td>AName=4
(Max_LogLn)</td><td colspan="2">Begrenzt die Länge der Log-Datei, ältere Log-Datei wird in .old umbenannt
AValue=Maximale Zeilenzahl</td></tr>
<tr><td>AName=5
(ThreadPriority)</td><td colspan="2">Ändert die Priority der Threads der LSTEP-API. Nach Connect werden die Threads immer auf normale Priorität gesetzt, mit SetExtValue(5, ...) kann dies im Nachhinein geändert werden.
AValue:Windows-API-Konstante für Thread-Priorität wie THREAD_PRIORITY_ABOVE_NORMAL</td></tr>
<tr><td>Beispiel:</td><td colspan="3">LSX.SetExtValue(3, 1); // Filter für Move-Befehle an
LSX.SetExtValue(4, 10000); // maximale Länge der Log-Datei = 10000 Zeilen
LSX.SetExtValue(5, THREAD_PRIORITY_HIGHEST);</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>3</td><td>-</td><td>-</td></tr>
</table>

LSX_SetProcessMessagesProc			
Beschreibung:	Ermöglicht das Ersetzen der internen Message-Dispatching Prozedur der LSTEP-API. Die LSTEP-API verarbeitet, während des Wartens auf Rückmeldungen der LStep im Main-Thread, Messages. Wenn Sie das Message-Dispatching abschalten wollen oder durch eigenen Code ersetzen wollen, können Sie SetProcessMessagesProc zum Setzen einer Callback-Prozedur verwenden.		
Delphi:	function LSX_SetProcessMessagesProc(LSID: Integer; Proc: Pointer): Integer;		
C++:	int SetProcessMessagesProc(void* pProc);		
LabView:			
Parameter:	pProc	Zeiger auf Ersatzfunktion Dies muss ein Zeiger auf eine stdcall-Prozedur ohne Parameter sein: void MyProcessMessages ()	
Beispiel:	LSX.SetProcessMessagesProc(&MyProcessMessages);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

<table>
<tr><th colspan="5">LSX_SetOsziCallBackFct</th></tr>
<tr><td>Beschreibung:</td><td colspan="4">Vermittelt der API die Adresse einer CallBack-Funktion, die aufgerufen wird, wenn ein asynchroner Event auftritt. Die gesetzte CallBack-Funktion ist global und unterscheidet nicht zwischen den LSTEPs. Soll eine steuerungsspezifische CallBack-Funktion aufgerufen werden, ist die Funkion LSX_SetExtCallBackFct zu verwenden. Solange über die Funkion LSX_SetExtCallBackFct keine steuerungsbezogene CallBack-Funktion zugewiesen wurde, wird die mit LSX_SetOsziCallBackFct zugewiesene Funktion verwendet.
Diese Funktionalität ist abhängig vom Schnittstellentyp und der Steuerung. Eine Liste der unterstützten Schnittstellentypen und Steuerungen ist unter „5.4 Unterstützte Steuerungen bzw. Schnittstellentypen“ zu finden.
Die Beschreibung zur Verwendung der CallBack-Funktion der LSTEP-API sowie deren Aufbau ist unter „5 CallBack-Funktionen der LSTEP-API“ beschrieben.</td></tr>
<tr><td>Delphi:</td><td colspan="4">LSX_SetOsziCallBackFct(LSID: Integer; Fct: Pointer): Integer;</td></tr>
<tr><td>C++:</td><td colspan="4">int SetOsziCallBackFct (void* pFct)</td></tr>
<tr><td>LabView:</td><td colspan="4">-</td></tr>
<tr><td>Parameter:</td><td>pFct</td><td colspan="3">Zeiger auf die CallBack-Funktion. Für die Funktionsdeklaration siehe „5 CallBack-Funktionen der LSTEP-API“.</td></tr>
<tr><td>Beispiel:</td><td colspan="4">void CALLBACK OsziCallBackFct(char *pcData, int lMaxLen, int lChannelID)
{
…
}
…
LSX.SetOsziCallBackFct(OsziCallBackFct);
//Erweitertes Protokoll aktivieren:
LSX.SendString("!errorchannel 2\r", 0, 0, false, 1000);</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td colspan="2">“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>2</td><td colspan="2">-</td><td>-</td></tr>
</table>

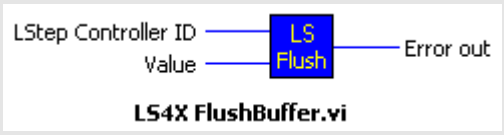
<table>
<tr><th colspan="4">LSX_SetExtCallBackFct</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Vermittelt der API die Adresse einer CallBack-Funktion, die aufgerufen wird wenn ein asynchroner Event auftritt. Die gesetzte CallBack-Funktion bezieht sich auf die LSTEP Instanzen und bietet die Möglichkeit ein Objekt z. B. zur Steuerungserkennung zu übergeben. Solange über die Funkion LSX_SetExtCallBack-Fct keine steuerungsbezogene CallBack-Funktion zugewiesen wurde, wird die mit LSX_SetOsziCallBackFct zugewiesene Funktion verwendet.
Diese Funktionalität ist abhängig vom Schnittstellentyp und der Steuerung. Eine Liste der unterstützten Schnittstellentypen und Steuerungen ist unter „5.4 Unterstützte Steuerungen bzw. Schnittstellentypen“ zu finden.
Die Beschreibung zur Verwendung der CallBack-Funktion der LSTEP-API sowie deren Aufbau ist unter „5 CallBack-Funktionen der LSTEP-API“ beschrieben.</td></tr>
<tr><td>Delphi:</td><td colspan="3">LSX_SetExtCallBackFct(LSID: Integer; Fct: Pointer; pprivate: Pointer): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int SetExtCallBackFct (void* pFct, void* ppprivate)</td></tr>
<tr><td>LabView:</td><td colspan="3">-</td></tr>
<tr><td rowspan="2">Parameter:</td><td>pFct</td><td colspan="2">Zeiger auf die CallBack-Funktion. Für die Funktionsdeklaration siehe „5 CallBack-Funktionen der LSTEP-API“.</td></tr>
<tr><td>pprivate</td><td colspan="2">Zeiger auf Objekt, welches beim Aufruf der Callback-Funktion mit übergeben wird.</td></tr>
<tr><td>Beispiel:</td><td colspan="3">int CALLBACK LSTEPExtCallBackFct(char *pcData, int lMaxLen, int lChannelID, void* pObject)
{
…
}
…
LSX.SetExtCallBackFct(LSTEPExtCallBackFct);
//Erweitertes Protokoll aktivieren:
LSX.SendString("!errorchannel 2\r", 0, 0, false, 1000);</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>2</td><td>-</td><td>-</td></tr>
</table>

LSX_SetLanguage, LSX_SetLanguageW			
Beschreibung:	Sprachumschaltung der LSTEP-API (Protokoll/Meldungen)		
Delphi:	function LSX_SetLanguage(LSID: Integer; PLN: PAnsiChar): Integer; function LSX_SetLanguageW(LSID: Integer; PLN: PWideChar): Integer;		
C++:	int SetLanguage (char *pcPLN); int SetLanguageW (TCHAR *pcPLN);		
LabView:			
Parameter:	PLN	Sprache (Kürzel, z.B. „DEU„ oder „ENG„) Die entsprechende ANSI- oder UNICODE-Textdatei (LSTEP4deu.txt oder LSTEP4eng.txt) muss im Verzeichnis des Programms liegen	
Beispiel:	LSX.SetLanguage('ENG');		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

LSX_TranslateErrMsg			
Beschreibung:	Übersetzt Fehlermeldung, die über eine CallBack-Funktion vermittelt wurde.		
Delphi:	LSX_TranslateErrMsg(LSID: Integer; MsgIn: PWideChar; MsgOut: PWideChar; MaxLen: Integer): Integer;		
C++:	int TranslateErrMsg (TCHAR *pcMsgIn, TCHAR *pcMsgOut, int lMaxLen)		
LabView:	-		
Parameter:	MsgIn	Fehlermeldung, die der CallBack-Funktion vermittelt wurde, umgewandelt in UNICODE, nullterminiert.	
	pcMsgOut	Puffer, der die Übersetzung der Fehlermeldung enthält.	
	MaxLen	Maximale Anzahl an Zeichen, die in den Puffer kopiert werden dürfen.	

Beispiel:	TCHAR InputString[255];
	TCHAR TranslatedString [255];
	
	// Konvertiere ASCII-Zeichenkette nach UNICODE
	wcscpy_s(InputString, CString(pcData, lMaxLen));
	LSX.TranslateErrMsg(InputString, TranslatedString, 255);
	
	… // Verarbeite TranslatedString		

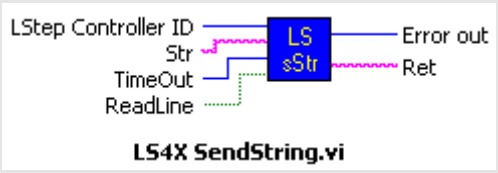
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	-	-

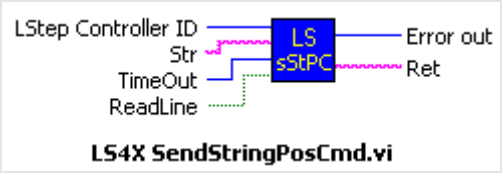
LSX_FlushBuffer			
Beschreibung:	Löscht den Kommunikations-Eingabepuffer bei der RS-232 bzw. PCI Schnittstelle. Dies kann in Fehler-Situationen verwendet werden, um nicht mehr benötigte Rückmeldungungen aus dem Eingabepuffer zu entfernen.		
Delphi:	function LSX_FlushBuffer(LSID: Integer; AValue: Integer): Integer;		
C++:	int FlushBuffer (int lAValue);		
LabView:			
Parameter:	AValue	Derzeit nicht verwendet, kann auf Null gesetzt werden.	
Beispiel:	LSX.FlushBuffer(0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

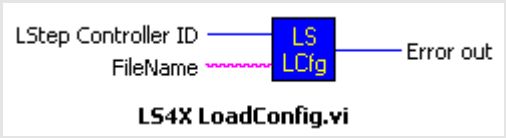
LSX_SetAbortFlag			
Beschreibung:	Flag setzen, damit die Kommunikation mit der LSTEP abgebrochen wird. Eine Funktion, die bei Aufruf von LSX_SetAbortFlag noch auf eine Rückmeldung der Steuerung warten (z.B. Verfahrbefehle), kehrt dann mit einer Fehlermeldung zurück. Die Verwendung dieser Funktion ist insbesondere bei Programmen mit Botschaftsbehandlungsroutinen oder mehreren Threads sinnvoll, falls z. B. schnell eine Verfahrbewegung abgebrochen werden soll.		
Delphi:	function LSX_SetAbortFlag(LSID: Integer): Integer;		
C++:	int SetAbortFlag ();		
LabView:			
Parameter:	-		
Beispiel:	LSX.SetAbortFlag(); LSX.StopAxes(); (Kommunikation mit der LSTEP abbrechen und das Kommando zum Stoppen aller Achsen senden)		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

LSX_EnableCommandRetry			
Beschreibung:	Mit dieser Funktion kann das wiederholte Senden von Kommandos im Falle von Fehlern ein-/ausgeschaltet werden (Standardmäßig ist dieses eingeschaltet)		
Delphi:	function LSX_EnableCommandRetry(LSID: Integer; AValue: LongBool): Integer;		
C++:	int EnableCommandRetry (BOOL bAValue);		
LabView:			
Parameter:	AValue	Wiederholtes Senden einschalten bzw. ausschalten True = Wiederholtes Senden bei Fehlern einschalten False = Wiederholtes Senden bei Fehlern ausschalten	
Beispiel:	LSX.EnableCommandRetry(false) ;		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

LSX_SetCommandTimeout		
Beschreibung:	Setzt die Timeoutzeiten für das Warten auf Rückmeldungen bei allgemeinen API-Aufrufen, beim Positionieren und beim Kalibrieren. Der Standardwert für die Timeoutzeit beträgt 1000 ms.	
Delphi:	LSX_SetCommandTimeout(LSID: Integer; AtoRead, AtoMove, AtoCalibrate: Integer): Integer;	
C++:	int SetCommandTimeout (int lAtoRead, int lAtoMove, int lAtoCalibrate) ;	
LabView:		
Parameter:	AtoRead	Timeoutzeit für das Warten auf allgemeine Rückmeldung eines API-Aufrufs in ms
	AtoMove	Timeoutzeit für Positionieren in ms
	AtoCalibrate	Timeoutzeit für Kalibrieren in ms
Beispiel:	LSX. SetCommandTimeout (int lAtoRead, int lAtoMove, int lAtoCalibrate) ;	
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

<table>
<tr><th colspan="4">LSX_SendString, LSX_SendStringW</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Sendet einen String an eine LSTEP-Steuerung. Mit dieser Funktion können Befehle an eine Steuerung gesendet werden, für die es keine API-Funktion gibt. Dabei ist auf den Befehlsaufbau und die Abschlusszeichen zu achten. Siehe hierzu die jeweilige Steuerungs-Dokumentation.
Achtung: Der Befehl „!configmaxaxis“ darf nicht über diese Funktion an die Steuerung gesendet werden, sondern muss durch den entsprechenden API Befehl gesendet werden. Dieser lautet „LSX_ConfigMaxAxis“ und ist im Kapitel „Achs- und Motorkonfiguration“ zu finden.</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_SendString(LSID: Integer; Str, Ret: PAnsiChar; MaxLen: Integer; ReadLine: LongBool; TimeOut: Integer): Integer;
function LSX_SendString(LSID: Integer; Str, Ret: PWideChar; MaxLen: Integer; ReadLine: LongBool; TimeOut: Integer): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int SendString (char *pcStr,char *pcRet,int lMaxLen,BOOL ReadLine,int lTimeOut);
int SendStringW (TCHAR *pcStr,TCHAR *pcRet,int lMaxLen,BOOL ReadLine,int lTimeOut);</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td rowspan="5">Parameter:</td><td>Str</td><td colspan="2">Nullterminierter String, der an die Steuerung gesendet werden soll.</td></tr>
<tr><td>Ret</td><td colspan="2">Puffer, der die Rückmeldung der LSTEP enthält, falls ReadLine = True.</td></tr>
<tr><td>MaxLen</td><td colspan="2">Maximale Anzahl von Zeichen, die in den Puffer für die Rückmeldung kopiert werden dürfen.</td></tr>
<tr><td>ReadLine</td><td colspan="2">Warte auf Rückmeldung von der LSTEP-Steuerung.
True = Rückmeldung erwartet
False = Keine Rückmeldung</td></tr>
<tr><td>TimeOut</td><td colspan="2">Maximale Wartezeit auf ein Rückmeldung in ms.</td></tr>
<tr><td>Beispiel:</td><td colspan="3">LSX.SendString(“?ver\r“, pcLStepVer, 256, true, 1000);
// Versionsnummer lesen, Timeout 1s</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>3</td><td>-</td><td>-</td></tr>
</table>

<table>
<tr><th colspan="4">LSX_SendStringPosCmd, LSX_SendStringPosCmdW</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Sendet einen String als Verfahrbefehl an die Steuerung, welcher die Achsmaske (siehe LSX_GetStatusAxis, LSX_GetStatusAxisW) als Rückmeldung von der Steuerung erwarten kann.</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_SendStringPosCmd(LSID: Integer; Str, Ret: PAnsiChar; MaxLen: Integer; ReadLine: LongBool; TimeOut: Integer): Integer;
function LSX_SendStringPosCmdW(LSID: Integer; Str, Ret: PWideChar; MaxLen: Integer; ReadLine: LongBool; TimeOut: Integer): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int SendStringPosCmd (char *pcStr, char *pcRet, int lMaxLen, BOOL bReadLine, int lTimeOut);
int SendStringPosCmdW(TCHAR*pcStrTCHAR *pcRet, int lMaxLen, BOOL bReadLine, int lTimeOut);</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td rowspan="6">Parameter:</td><td>Str</td><td colspan="2">Nullterminierter String, der an die Steuerung gesendet werden soll.</td></tr>
<tr><td>Ret</td><td colspan="2">Puffer, der die Rückmeldung der LSTEP enthält, falls ReadLine = True</td></tr>
<tr><td>MaxLen</td><td colspan="2">Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen.</td></tr>
<tr><td>ReadLine</td><td colspan="2">Rückmeldung der LSTEP lesen:</td></tr>
<tr><td>TimeOut</td><td colspan="2">Maximale Wartezeit auf Rückmeldung [ms]</td></tr>
<tr><td></td></tr>
<tr><td>Beispiel:</td><td colspan="3">LSX.SendStringPosCmd(“!moa 1 2\r“, pcLStepVer, 256, true, 100000);</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>3</td><td>-</td><td>-</td></tr>
</table>

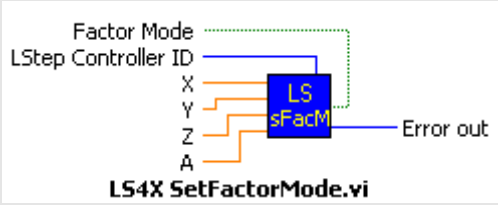
<table>
<tr><th colspan="4">LSX_LoadConfig, LSX_LoadConfigW</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Lädt LSTEP-Konfiguration (Schnittstelle, Achseinstellungen, Regler) aus einer INI-Datei. Die geladene Konfiguration wird in den Funktionen LSX_Connect und LSX_SetControlPars verwendet.
Das Format der INI-Datei bei LSTEP-Steuerungen ist mit der WIN-Commander-INI-Datei kompatibel, d.h. die Einstellungen können aus dem WIN-Commander (Wincom4.ini) übernommen werden. Dies gilt nicht für WIN-Commander 5 und die Steuerungen der LSTEP express Serie
Achtung!
Für die LSTEP-PCI express oder LSTEP express kann man mit dem WIN-Commander 5 im Menü Steuerung / Konfiguration exportieren die Einstellungen speichern und diese mit LoadConfig laden. Mit LSX_SetControlPars wird diese Konfiguration zur Steuerung übertragen.</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_LoadConfig(LSID: Integer; FileName: PAnsiChar): Integer;
function LSX_LoadConfigW(LSID: Integer; FileName: PWideChar): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int LoadConfig (char *pcFileName);
int LoadConfigW (TCHAR *pcFileName);</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td>Parameter:</td><td>FileName</td><td colspan="2">Dateiname der INI-Datei als nullterminierter String</td></tr>
<tr><td>Beispiel:</td><td colspan="3">LSX.LoadConfig(“C:\LStepTest\LStep.INI“);</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>3</td><td>-</td><td>-</td></tr>
</table>

LSX_SaveConfig, LSX_SaveConfigW			
Beschreibung:	Speichert die LSTEP-Konfiguration (Schnittstelle, Achseinstellungen, Regler) in einer INI-Datei. Das Format der INI-Datei ist mit der WIN-Commander 4-INI-Datei kompatibel. (nicht für LSTEP express)		
Delphi:	function LSX_SaveConfig(LSID: Integer; FileName: PAnsiChar): Integer; function LSX_SaveConfig(LSID: Integer; FileName: PWideChar): Integer;		
C++:	int SaveConfig (char *pcFileName); int SaveConfigW (TCHAR *pcFileName);		
LabView:			
Parameter:	FileName	Dateiname der INI-Datei als nullterminierter String	
Beispiel:	LSX.SaveConfig("C:\LStepTest\LStep.INI");		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	-	-

<table>
<tr><th colspan="4">LSX_SetFactorMode</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Positionswert-Umrechnung für ‚krumme' Spindelsteigungen

Erst nach SetFactorMode ist SetPitch mit der tatsächlichen, physikalischen Spindelsteigung aufzurufen.

Alle Verfahrbefehle verwenden nach dem Aufruf von SetFactorMode und SetPitch einen Faktor zur Umrechnung, damit die LSTEP korrekt positioniert. Der resultierende Vektor ergibt sich wie folgt:

Gesendeter Positionsvektor = Gesendeter Positionsvektor * Spindelsteigung LSTEP / physikalische Spindelsteigung</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_SetFactorMode(LSID: Integer; AFactorMode: LongBool; X, Y, Z, A: Double): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int SetFactorMode (BOOL bAFactorMode, double dX, double dY, double dZ, double dA);</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td rowspan="2">Parameter:</td><td>AFactorMode</td><td colspan="2">Dieser Parameter aktiviert eine API-interne Umrechnung der Positionswerte/Spindelsteigung, um so bei 'krummen' Spindelsteigungen Rundungsfehler zu vermeiden</td></tr>
<tr><td>X, Y, Z, A</td><td colspan="2">Spindelsteigungswerte, die an die LSTEP übertragen werden (möglichst Werte wie 1.0 oder 4.0, sodass ein Mikrostep einem nichtperiodischen Dezimalbruch entspricht)</td></tr>
<tr><td>Beispiel:</td><td colspan="3">LSX.SetFactorMode(true, 1, 1, 1, 0);
LSX.SetPitch(1.234, 1.234, 2.345, 0);
LSX.MoveAbs(1.234, 2.468, 2.345, 0, true);</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>1</td><td>-</td><td>-</td></tr>
</table>

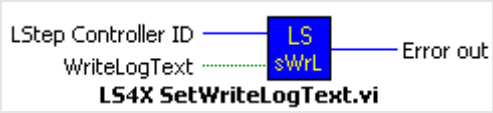
LSX_Initialize, LSX_InitializeW			
Beschreibung	Funktion zum Einstellen des Pfades, in dem die Einstellungen für das Protokollfenster gespeichert werden. Unter diesem Pfad werden auch die Log-Dateien abgelegt, falls für sie kein Pfad mit SetWriteLogTextFN eingestellt wurde.		
Delphi	function LSX_Initialize(LSID: Integer; Workpath: PAnsiChar): Integer; function LSX_InitializeW(LSID: Integer; Workpath: PWideChar): Integer;		
C++	int Initialize(char *pcWorkpath); int InitializeW(TCHAR *pcWorkpath);		
LabView:	-		
Parameter	Workpath – nullterminierter String		
Beispiel	LSX.Initialize("C:\Users\All Users\MyProg\LS1");		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

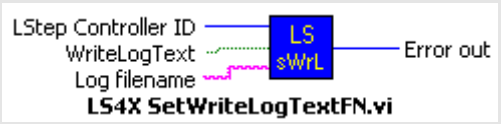
LSX_SetShowCmdList			
Beschreibung:	LSTEP-API Befehlsliste anzeigen oder verstecken		
Delphi:	function LSX_SetShowCmdList(LSID: Integer; ShowCmdList: LongBool): Integer;		
C++:	int SetShowCmdList (BOOL bShowCmdList);		
LabView:	-		
Parameter:	ShowProt	Gibt an, ob das Fenster „LSTEP-API Befehlsliste" gezeigt werden soll	
Beispiel:	LSX.SetShowCmdList(true);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

LSX_SetShowProt			
Beschreibung:	Schnittstellen-Protokoll anzeigen oder verstecken		
Delphi:	function LSX_SetShowProt(LSID: Integer; ShowProt: LongBool): Integer;		
C++:	int SetShowProt (BOOL ShowProt);		
LabView:			
Parameter:	ShowProt	Gibt an, ob das Fenster „Schnittstellen-Protokoll“ gezeigt werden soll	
Beispiel:	LSX.SetShowProt(true);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

LSX_EnableGlobalLogging			
Beschreibung:	Deaktiviert jegliche Logging-Funktionalität der API		
Delphi:	function LSX_EnableGlobalLogging(LSID: Integer; Enable: LongBool): Integer;		
C++:	int EnableGlobalLogging (BOOL Enable);		
LabView:	Nicht unterstützt		
Parameter:	Enable	Sollen alle Logging-Funktionen aktiviert sein?	
Beispiel:	LSX.EnableGlobalLogging(false); //deaktiviert das Logging vollständig		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

LSX_EnableGuiLogging			
Beschreibung:	Deaktiviert nur das Logging in das Protkollfenster der API Das Deaktivieren sorgt dafür, dass die CPU-Last der API deutlich sinkt.		
Delphi:	function LSX_EnableGuiLogging(LSID: Integer; Enable: LongBool): Integer;		
C++:	int EnableGuiLogging (BOOL Enable);		
LabView:	Nicht unterstützt		
Parameter:	Enable	Soll das Logging in das Protokollfenster der API aktiviert sein?	
Beispiel:	LSX.EnableGuiLogging(false); //deaktiviert das Logging des Protokollfensters		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

LSX_SetWriteLogText			
Beschreibung:	Schreiben der Protokoll-Datei LSTEP4.log ein-/ausschalten (Standardmäßig ist das Schreiben in LSTEP4.log ausgeschaltet)		
Delphi:	function LSX_SetWriteLogText(LSID: Integer; AWriteLogText: LongBool): Integer;		
C++:	int SetWriteLogText (BOOL AWriteLogText);		
LabView:			
Parameter:	-		
Beispiel:	LSX.SetWriteLogText (true);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

LSX_SetWriteLogTextFN, LSX_SetWriteLogTextFNW			
Beschreibung:	Schreiben des Schnittstellen-Protokolls in eine bestimmte Datei ein-/ausschalten (Standardmäßig ist das Schreiben ausgeschaltet)		
Delphi:	function LSX_SetWriteLogTextFN(LSID: Integer; AWriteLogText: LongBool; ALogFN: PAnsiChar): Integer; function LSX_SetWriteLogTextFNW(LSID: Integer; AWriteLogText: LongBool; ALogFN: PWideChar): Integer;		
C++:	int SetWriteLogTextFN (BOOL bAWriteLogText, char *pcALogFN); int SetWriteLogTextFNW (BOOL bAWriteLogText, TCHAR *pcALogFN);		
LabView:			
Parameter:	AWriteLogText		Wenn True dann schreibe Protokolldatei
	ALogFN		Dateiname der Protokolldatei
Beispiel:	LSX.SetWriteLogTextFN(true, „C:\Temp\prot.txt");		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

LSX_GetEqepConfig			
Beschreibung:	Fragt die eingestellte Funktion der Pins 16-19 vom 50 poligen Mulitfunktionsport (MFP) ab.		
Delphi:	function LSX_GetEqepConfig (LSID: Integer; var GetWert: Integer): Integer;		
C++:	int GetEqepConfig (int *plGetWert);		
LabView:	Nicht unterstützt		
Parameter:	Wert:	0 = TTL-Eingänge	
		1 = Tipp-Betrieb	
		2 = Trackball-Betrieb	
		3 = Handrad-Betrieb (noch nicht implementiert)	
		4 = TVR-Eingänge	
		5 = QEP-Gebersysteme	
		6 = Achsenfreigabe im Joystickbetrieb	
Beispiel:	LSX.GetEqepConfig(&GetWert); //Pins als TTL-Eingänge		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?eqepconfig	-

LSX_SetEqepConfig			
Beschreibung:	Konfiguriert die Funktion der Pins 20-25 vom 50 poligen Mulitfunktionsport (MFP).		
Delphi:	function LSX_SetEqepConfig(LSID: Integer; Wert: Integer): Integer;		
C++:	int SetEqepConfig (int *plWert);		
LabView:	Nicht unterstützt		
Parameter:	Wert:	0 = TTL-Eingänge	
		1 = Tipp-Betrieb	
		2 = Trackball-Betrieb	
		3 = Handrad-Betrieb (noch nicht implementiert)	
		4 = TVR-Eingänge	
		5 = QEP-Gebersysteme	
		6 = Achsenfreigabe im Joystickbetrieb	
Beispiel:	LSX.SetEqepConfig(0); //Pins als TTL-Eingänge		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!eqepconfig	!validconfig

LSX_GetTTLOutConfig			
Beschreibung:	Fragt die eingestellte Funktion der Pins 16-19 vom 50 poligen Mulitfunktionsport (MFP) ab.		
Delphi:	function LSX_GetTTLOutConfig(LSID: Integer; TTLOut: Integer): Integer;		
C++:	int GetTTLOutConfig (int *plTTLOut);		
LabView:	Nicht unterstützt		
Parameter:	TTLOut	0 = TTL-Ausgänge	
		1 = TVR-Out-Betrieb	
Beispiel:	LSX.GetTTLOutConfig(&TTLOut);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?ttloutconfig	-

LSX_SetTTLOutConfig			
Beschreibung:	Stellt die Funktion der Pins 16-19 vom 50 poligen Mulitfunktionsport (MFP) ein.		
Delphi:	function LSX_SetTTLOutConfig(LSID: Integer; TTLOut: Integer): Integer;		
C++:	int SetTTLOutConfig (int lTTLOut);		
LabView:	Nicht unterstützt		
Parameter:	TTLOut	0 = TTL output	
		1 = TVR Out operation	
Beispiel:	LSX.SetTTLOutConfig(1);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!ttloutconfig	!validconfig

LSX_GetStopMode			
Beschreibung:	Fragt den eingestellten Modus des aktiven Stopp-Eingangs ab.		
Delphi:	function LSX_GetStopMode (LSID: Integer; var Mode: Integer): Integer;		
C++:	int GetStopMode (int *plMode);		
LabView:	Nicht unterstützt		
Parameter:	TTLOut	0 = Stoppen der Bewegungen wird ausgelöst	
		1 = Stoppen der Bewegungen und Abschalten der Endstufen wird ausgelöst	
Beispiel:	LSX.GetStopMode (&Mode);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?stopmode	-

LSX_SetStopMode			
Beschreibung:	Stellt den Modus des aktiven Stopp-Eingangs ein.		
Delphi:	function LSX_SetStopMode (LSID: Integer;Mode: Integer): Integer;		
C++:	int SetStopMode (int lMode);		
LabView:	Nicht unterstützt		
Parameter:	Mode	0 = Stoppen der Bewegungen wird ausgelöst	
		1 = Stoppen der Bewegungen und Abschalten der Endstufen wird ausgelöst	
Beispiel:	LSX.SetStopMode (1);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!stopmode	!validconfig

LSX_GetConfigurated			
Beschreibung:	Fragt die Konfiguriert Kennung ab.Parametersatz der einzelnen Achsen wird in Steuerung als gültig erklärt. Achsen ohne Konfiguriert-Kennung können nicht betrieben werden.		
Delphi:	function LSX_GetConfigurated(LSID: Integer; var X,Y,Z,A: LongBool): Integer;		
C++:	int GetConfigurated (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	False = Die Konfiguriert Kennung nicht setzen	
		True = Die Konfiguriert Kennung setzen	
Beispiel:	LSX.GetConfigurated (&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?configurated	-

LSX_SetConfigurated			
Beschreibung:	Stellt die Konfiguriert Kennung ein. Parametersatz der einzelnen Achsen wird in Steuerung als gültig erklärt. Achsen ohne Konfiguriert-Kennung können nicht betrieben werden.		
Delphi:	function LSX_SetConfigurated(LSID: Integer; X,Y,Z,A: LongBool): Integer;		
C++:	int SetConfigurated (BOOL bX,BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	False = Die Konfiguriert Kennung nicht setzen	
		True = Die Konfiguriert Kennung setzen	
Beispiel:	LSX.SetConfigurated (True,True,False,False);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!configurated	!validconfig

LSX_GetSysSampleRate			
Beschreibung:	Befehl zum Auslesen der System-Ausgabefrequenz bzw. Abtastzeit		
Delphi:	function LSX_GetSysSampleRate (LSID: Integer; val rate : integer): Integer;		
C++:	int GetSysSampleRate (int *plrate);		
LabView:	Nicht unterstützt		
Parameter:	rate	Samplerate: 0: 40,0 µs / 25,0 kHz 1: 50,0 µs / 20,0 kHz 2: 62,5 µs / 16,0 kHz 3: 66,6… µs / 15,0 kHz 4: 75,0 µs / 13,3… kHz 5: 80,0 µs / 12,5 kHz	
Beispiel:	LSX.GetSysSampleRate (&Rate);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?syssamplerate	-

LSX_SetSysSampleRate			
Beschreibung:	Befehl zum Einstellen der System-Ausgabefrequenz bzw. Abtastzeit		
Delphi:	function LSX_SetSysSampleRate (LSID: Integer; rate : integer): Integer;		
C++:	int SetSysSampleRate (int lrate);		
LabView:	Nicht unterstützt		
Parameter:	rate	Samplerate: 0: 40,0 µs / 25,0 kHz 1: 50,0 µs / 20,0 kHz 2: 62,5 µs / 16,0 kHz 3: 66,6… µs / 15,0 kHz 4: 75,0 µs / 13,3… kHz 5: 80,0 µs / 12,5 kHz	
Beispiel:	LSX.SetSysSampleRate (0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!syssamplerate	!save und Steuerungs-Neustart

LSX_GetBaud				
Beschreibung:	Befehl zum Einstellen der Baudrate, Maximalwert ist abhängig von der verwendeten Hardware-Schnittstelle:			
Delphi:	function LSX_GetBaud (LSID: Integer; val rate : integer): Integer;			
C++:	int GetBaud (int *plrate);			
LabView:	Nicht unterstützt			
Parameter:	rate	Mögliche Werte: 9600, 19200, 38400, 57600, 115200, 230400, 460800 USB: 460.800 baud Ethernet: 115.200 baud RS232: 115.200 baud PCIexpress: 115.200 baud		
Beispiel:	LSX.GetBaud (&Rate);			
Sonstiges:	Kompatibilität		"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2		?baud	-

LSX_SetBaud				
Beschreibung:	Befehl zum Einstellen der Baudrate, Maximalwert ist abhängig von der verwendeten Hardware-Schnittstelle:			
Delphi:	function LSX_SetBaud (LSID: Integer; rate : integer): Integer;			
C++:	int SetBaud (int lrate);			
LabView:	Nicht unterstützt			
Parameter:	rate	Mögliche Werte: 9600, 19200, 38400, 57600, 115200, 230400, 460800 USB: 460.800 baud Ethernet: 115.200 baud RS232: 115.200 baud PCIexpress: 115.200 baud		
Beispiel:	LSX.SetBaud (115200);			
Sonstiges:	Kompatibilität		"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2		!baud	sofort

4.2.2 Steuerungs- und API-Informationen

LSX_GetAPIVersion, LSX_GetAPIVersionW			
Beschreibung:	Liefert die Versionsnummer der LSTEP-API.		
Delphi:	LSX_GetAPIVersion(LSID: Integer; APIVer: PAnsiChar; MaxLen: Integer): Integer; LSX_GetAPIVersionW(LSID: Integer; APIVer: PWideChar; MaxLen: Integer): Integer;		
C++:	int GetAPIVersion (char *pcAPIVer, int lMaxLen) int GetVersionStrDetW (TCHAR *pcVersDet, int lMaxLen)		
LabView:			
Parameter:	APIVer	Puffer, der die Versionsnummer von LSTEP-API enthält	
	MaxLen	Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen. Die Versionsnummer kann eine Länge von ca. 20 Zeichen erreichen.	
Beispiel:	LSX.GetAPIVersion(pcVers, 100);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

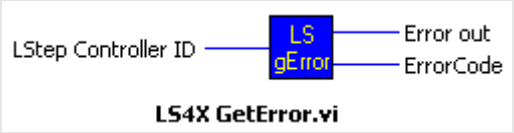
LSX_GetSerialNr, LSX_GetSerialNrW			
Beschreibung:	Seriennummer der Steuerung auslesen. (nicht für LSTEP express)		
Delphi:	function LSX_GetSerialNr(LSID: Integer; SerialNr: PAnsiChar; MaxLen: Integer): Integer; function LSX_GetSerialNrW(LSID: Integer; SerialNr: PWideChar; MaxLen: Integer): Integer;		
C++:	int GetSerialNr (char *pcSerialNr,int lMaxLen); int GetSerialNr W(TCHAR *pcSerialNr,int lMaxLen);		
LabView:			
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:	LSX.GetSerialNr(pcSerialNr, 256);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?readsn	-

LSX_GetVersionStr, LSX_GetVersionStrW			
Beschreibung:	Liefert die aktuelle Versionsnummer der Firmware zurück. Für zusätzliche Information sollte GetVersionStrDet und GetVersionStrInfo ebenfalls abgefragt werden.		
Delphi:	function LSX_GetVersionStr(LSID: Integer; Vers: PAnsiChar; MaxLen: Integer): Integer; function LSX_GetVersionStrW(LSID: Integer; Vers: PWideChar; MaxLen: Integer): Integer;		
C++:	int GetVersionStr (char *pcVers,int lMaxLen);		
LabView:			
Parameter:	Vers	Zeiger auf einen Puffer, in dem der Versions-String zurückgegeben wird	
	MaxLen	Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen	
Beispiel:	LSX.GetVersionStr(pcVers, 64);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?ver	-

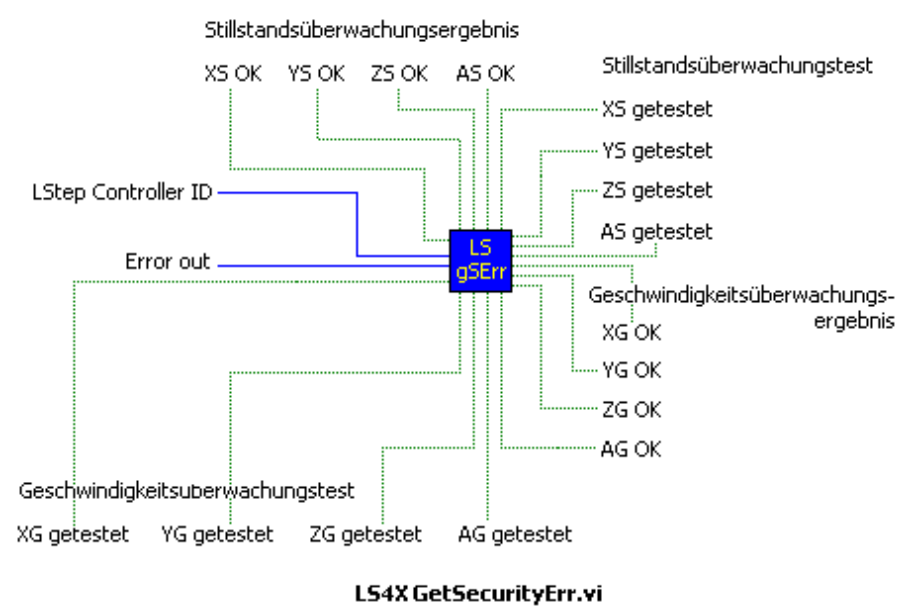
LSX_GetVersionStrDet, LSX_GetVersionStrDetW			
Beschreibung:	Funktion zum Auslesen der Firmware-Konfiguration. Für zusätzliche Information sollte GetVersionStr und GetVersionStrInfo ebenfalls abgefragt werden. (nicht für LSTEP express)		
Delphi:	function LSX_GetVersionStrDet(LSID: Integer; VersDet: PAnsiChar; MaxLen: Integer): Integer; function LSX_GetVersionStrDetW(LSID: Integer; VersDet: PWideChar; MaxLen: Integer): Integer;		
C++:	int GetVersionStrDet (char *pcVersDet, int lMaxLen); int GetVersionStrDetW (TCHAR *pcVersDet, int lMaxLen);		
LabView:			
Parameter:	VersDet	Zeiger auf einen Puffer, in dem der detaillierteVersions-String zurückgegeben wird	
	MaxLen	Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen	
Beispiel:	LSX.GetVersionStrDet(pcVersDet, 64);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?det	-

LSX_GetVersionStrInfo, LSX_GetVersionStrInfoW			
Beschreibung:	Liefert detaillierte Informationen zur Versionsnummer. Für zusätzliche Information sollte GetVersionStr und GetVersionStrDet ebenfalls abgefragt werden.		
Delphi:	function LSX_ GetVersionStrInfo (LSID: Integer; VersInfo: PAnsiChar; MaxLen: Integer): Integer; function LSX_ GetVersionStrInfoW (LSID: Integer; VersInfo: PWideChar; MaxLen: Integer): Integer;		
C++:	int GetVersionStrInfo (char *pcVersInfo, int lMaxLen); int GetVersionStrInfoW (TCHAR *pcVersInfo, int lMaxLen);		
LabView:			
Parameter:	VersInfo	Zeiger auf einen Puffer, in dem die detaillierte Versionsnummer gespeichert wird. z. B.: T04.35.02-0004	
	MaxLen	Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen	
Beispiel:	LSX.GetVersionStrInfo (pcVersInfo, 64);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?iver	-

4.2.3 Statusabfragen

LSX_GetError			
Beschreibung:	Diese Funktion fragt die aktuelle Fehlernummer aus der Steuerung ab.		
Delphi:	function LSX_GetError(LSID: Integer; var ErrorCode: Integer): Integer;		
C++:	int GetError(int *plErrorCode);		
LabView:			
Parameter:	ErrorCode	Fehlernummer	
Beispiel:	LSX.GetError(&ErrCode);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!err	-

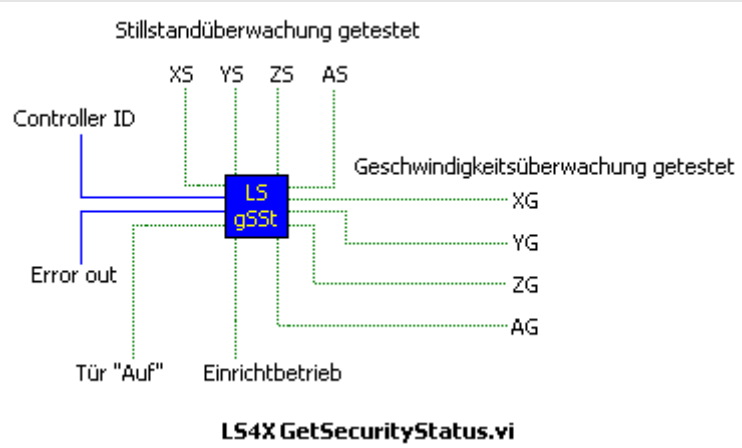
LSX_Quit			
Beschreibung:	Diese Funktion bestätigt aufgetretene Fehler und setzt diese somit zurück.		
Delphi:	function LSX_Quit(LSID: Integer): Integer;		
C++:	int Quit();		
LabView:	Nicht unterstützt		
Beispiel:	LSX.Quit();		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!quit	-

LSX_GetSecurityErr	
Beschreibung:	Liest alle Zustände und Ergebnisse der GAL-Sicherheitsüberwachung. (nur bei LS44-Steuerungen)
Delphi:	function LSX_GetSecurityErr(LSID: Integer; var Value: LongWord): Integer;
C++:	int GetSecurityErr(LongWord *pValue);
LabView:	

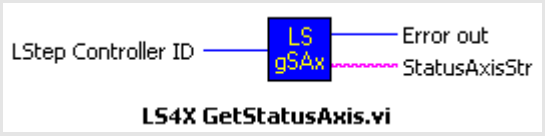
<table>
<tr><td>Parameter:</td><td>Value</td><td colspan="3">32-Bit LongWord ohne Vorzeichen, welcher nach Aufruf der Funktion in den Bits 0-15 die Bit-Maske enthält.

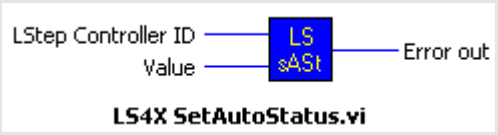
Wert 0 bei Überwachungsergebnis = nicht OK
Wert 1 bei Überwachungsergebnis = OK
Wert 0 bei Überwachungstest = nicht getestet
Wert 1 bei Überwachungstest = getestet

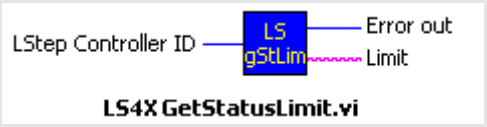
Bit 0 = X-Achse Stillstandüberwachungsergebnis
Bit 1 = Y-Achse Stillstandüberwachungsergebnis
Bit 2 = Z-Achse Stillstandüberwachungsergebnis
Bit 3 = A-Achse Stillstandüberwachungsergebnis
Bit 5 = X-Achse Stillstandüberwachungstest
Bit 6 = Y-Achse Stillstandüberwachungstest
Bit 7 = Z-Achse Stillstandüberwachungstest
Bit 8 = A-Achse Stillstandüberwachungstest
Bit 9 = X-Achse Geschwindigkeitsüberwachungsergebnis
Bit 10 = Y-Achse Geschwindigkeitsüberwachungsergebnis
Bit 11 = Z-Achse Geschwindigkeitsüberwachungsergebnis
Bit 12 = A-Achse Geschwindigkeitsüberwachungsergebnis
Bit 13 = X-Achse Geschwindigkeitsüberwachungstest
Bit 14 = Y-Achse Geschwindigkeitsüberwachungstest
Bit 15 = Z-Achse Geschwindigkeitsüberwachungstest</td></tr>
<tr><td>Beispiel:</td><td colspan="4">LSX.GetSecurityErr(&Value);</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td colspan="2">Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>1</td><td>?securityerror</td><td colspan="2">-</td></tr>
</table>

<table>
<tr><th colspan="4">LSX_GetSecurityStatus</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Liefert den aktuellen Zustand der Sicherheitsüberwachung.
(nur bei LS44-Steuerungen)</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_GetSecurityStatus(LSID: Integer; var Value: LongWord): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int GetSecurityStatus(LongWord *pValue);</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td>Parameter:</td><td>Value</td><td colspan="2">32-Bit LongWord ohne Vorzeichen, welcher nach Aufruf der Funktion in den Bits 0-15 die Bit-Maske enthält.
Bit 0-3 = interne Merker
Bit 4 = X-Achse Stillstandüberwachung getestet
Bit 5 = Y-Achse Stillstandüberwachung getestet
Bit 6 = Z-Achse Stillstandüberwachung getestet
Bit 7 = A-Achse Stillstandüberwachung getestet
Bit 8 = X-Achse Geschwindigkeitsüberwachung getestet
Bit 9 = Y-Achse Geschwindigkeitsüberwachung getestet
Bit 10 = Z-Achse Geschwindigkeitsüberwachung getestet
Bit 11 = A-Achse Geschwindigkeitsüberwachung getestet
Bit 14 = Zustand Einrichtbetrieb (Einrichtbetrieb = 1)
Bit 15 = Zustand Tür (Tür „Auf“ = 1)</td></tr>
<tr><td>Beispiel:</td><td colspan="3">LSX.GetSecurityStatus(&Value);</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>1</td><td>?securitystatus</td><td>-</td></tr>
</table>

LSX_GetStatus, LSX_GetStatusW			
Beschreibung:	Liefert den aktuellen Zustand der Steuerung.		
Delphi:	function LSX_GetStatus(LSID: Integer; Stat: PAnsiChar; MaxLen: Integer): Integer; function LSX_GetStatusW(LSID: Integer; Stat: PWideChar; MaxLen: Integer): Integer;		
C++:	int GetStatus(char *pcStat,int lMaxLen); int GetStatusW(TCHAR *pcStat,int lMaxLen);		
LabView:			
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:	LSX.GetStatus(pcStat, 256);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?status	-

LSX_GetStatusAxis, LSX_GetStatusAxisW			
Beschreibung:	liefert den aktuellen Zustand der einzelnen Achsen		
Delphi:	function LSX_GetStatusAxis(LSID: Integer; StatusAxisStr: PAnsiChar; MaxLen: Integer): Integer; function LSX_GetStatusAxisW(LSID: Integer; StatusAxisStr: PWideChar; MaxLen: Integer): Integer;		
C++:	int GetStatusAxis(char *pcStatusAxisStr,int lMaxLen); int GetStatusAxisW(TCHAR *pcStatusAxisStr,int lMaxLen);		
LabView:			
Parameter:	StatusAxisStr	Zeiger auf einen Puffer, in dem der Statusstring zurückgegeben wird. Der Statusstring enthält für jede Achse ein Zeichen und wird mit einem . abgeschlossen. - = Achse ist nicht freigegeben @ = Achse steht und ist bereit M = Achse ist in Bewegung (Motion) J = Joystick eingeschaltet C = Achse ist in Regelung * S = Achse steht in Endschalter A = Achse wurde kalibriert und ist bereit E = Fehler beim Kalibrieren (Endschalter nicht korrekt freigefahren) * F = Achse ist im Fehlerstatus D = Rückmeldung nach dem Tischhubmessen U = Einrichtbetrieb * T = Timeout * (* Nur LSTEP 2000 Serie)	
Parameter:	MaxLen	Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen	
Beispiel:	LSX.GetStatusAxis(pcStatAxis, 256);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?statusaxis	-

LSX_SetAutoStatus			
Beschreibung:	Diese Funktion schaltet den Autostatus ein bzw. aus. **Hinweis:** Der AutoStatus-Modus sollte normalerweise nicht verändert werden, da die LSTEP-API bei Verfahrbefehlen etc. den richtigen Modus einstellt, eine Änderung auf 0 oder 2 könnte zu Fehlern führen.		
Delphi:	function LSX_SetAutoStatus(LSID: Integer; Value: Integer): Integer;		
C++:	int SetAutoStatus(int lValue);		
LabView:			
Parameter:	Value	AutoStatus-Modus 0 = Es wird kein Status von der Steuerung gesendet. 1 = Es werden automatisch „Positionerreicht“- Meldungen von der Steuerung gesendet. 2 = Es werden automatisch „Positionerreicht“- und Status - Meldungen von der Steuerung gesendet. * 3 = Es gibt bei „Positionerreicht“ nur ein Carriage Return zurück. * (* Nur LSTEP 2000 Serie)	
Beispiel:	LSX.SetAutoStatus(3);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	3 (siehe Value)	!autostatus	-

LSX_GetStatusLimit, LSX_GetStatusLimitW			
Beschreibung:	Liefert den aktuellen Zustand der Software-Grenzen jeder einzelnen Achse		
Delphi:	function LSX_GetStatusLimit(LSID: Integer; Limit: PAnsiChar; MaxLen: Integer): Integer; function LSX_GetStatusLimitW(LSID: Integer; Limit: PWideChar; MaxLen: Integer): Integer;		
C++:	int GetStatusLimit(char *pcLimit, int lMaxLen); int GetStatusLimitW (TCHAR *pcLimit, int lMaxLen);		
LabView:			
Parameter:	pc 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 = Software-Limit wurde gesetzt - = Software-Grenze wurde nicht verändert	
	MaxLen	Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen	
Beispiel:	LSX.GetStatusLimit(pc Limit, 64);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?statuslimit	-

LSX_GetSysstat, LSX_GetSysStatus			
Beschreibung:	Liefern den aktuellen Zustand der Achsen		
Delphi:	function LSX_GetSysStat(LSID: Integer; var X,Y,Z,A: Integer): Integer; function LSX_GetSysStatus(LSID: Integer; Status :PWideChar; MaxLen: Integer): Integer		
C++:	int GetSysStat (int *plX, int *plY, int *plZ, int *plA); int GetSysStatus(TCHAR *pcStatus, int MaxLen);		
LabView:	Nicht unterstützt		
Parameter:	Sysstat	SysStatus	Beschreibung
	0	Just turned On	Zustand nach Einschalten der Steuerung
	1	Not Configured	Achse ist nicht konfiguriert
	2	Configured	Achse ist konfiguriert

	3	Ready for Komm	Achse ist berreit zur Autokommentierung
	4	Autokommutating	Achse wird autokommutiert
	5	Ready for Power	Achse ist bereit zum Einschalten der Endstufe
	6	Positioncontrol	Achse in Lagereglung bzw. Schrittmotorbetrieb
	7	Speedcontrol	Achse in Drehzahlreglung
	8	Manual Mode	Achse ist im manuellen Modus (z.B Joystick)
	9	Calibrating	Achse wird kalibriert
	10	Error-Power-Off	Fehler Treiberspannung, Zwischenkreisspannung, Motortemeratur, Endstufentemperatur, Autokommutierung oder Endstufenhardware
	12	Error-Stop-On	Fehler durch Lagereglerabweichung (Schleppfehler) oder I2T-Überwachung
	13	System Error	Systemfehler → Hersteller kontaktieren
	15	Changing	Systemstatuswechsel aktiv
	Anmerkung:		Status 11 und 14 nicht implementiert
Beispiele:	LSX.GetSysStat(&X,&Y,&Z,&A); LSX.GetSysStatus(&Status);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?sysstat//?sysstatus	-

LSX_GetStopStatus			
Beschreibung:	Auslesen des Stopp-Eingang-Zustands unter Berücksichtigung der eingestellten Stopp-Polarität		
Delphi:	function LSX_GetStopStatus(LSID: Integer; StopStatus: Boolean): Integer;		
C++:	int GetStopStatus (BOOL *pbStopStatus);		
LabView:	Nicht unterstützt		
Parameter:	Zustand des Stopp- Eingangs: False = Stopp inaktiv True = Stopp aktiv		
Beispiel:	LSX.GetStopStatus(&StopStatus);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?stopstatus	-

LSX_GetPowerAmplifierStatus			
Beschreibung:	Befehl zum Auslesen der aktuellen Endstufen-Zustände inlusive Ein- und Ausschaltprozessen (mit Berücksichtigung der Motorbremsenausgänge).		
Delphi:	function LSX_GetPowerAmplifierStatus (LSID: Integer; var X, Y, Z, R: Integer): Integer;		
C++:	int GetPowerAmplifierStatus (int *pbX, int *pbY, int *pbZ, int *pbR);		
LabView:	Nicht unterstützt		
Parameter:	Zustand des Eingangs: 000 = Endstufe ist aus, kein Prozess aktiv 010 = Endstufe ist aus und Ausschaltprozess (mit Berücksichtigung der Bremsenausgänge) aktiv 100 = Endstufe ist aus und Einschaltprozess (mit Berücksichtigung der Bremsenausgänge) aktiv 001 = Endstufe ist an, kein Prozess aktiv 011 = Endstufe ist an und Ausschaltprozess (mit Berücksichtigung der Bremsenausgänge) aktiv 101 = Endstufe ist an und Einschaltprozess (mit Berücksichtigung der Bremsenausgänge) aktiv		
Beispiel:	LSX.GetPowerAmplifierStatus(&X, &Y, &Z, &R);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?pastatus	-

LSX_GetPTemp			
Beschreibung:	Befehl zum Auslesen der Endstufentemperaturen		
Delphi:	function LSX_GetPTemp (LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int GetPTemp (double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Temperatur für jede Achse in [°C]	
Beispiel:	LSX.GetPTemp (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?ptemp	-

4.2.4 Parameterhandling

LSX_SetControlPars			
Beschreibung:	Überträgt die mit LSX_LoadConfig geladenen Parameter an die LSTEP.		
Delphi:	function LSX_SetControlPars(LSID: Integer): Integer;		
C++:	int SetControlPars ();		
LabView:			
Parameter:	-		
Beispiel:	LSX.SetControlPars();		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	-	-

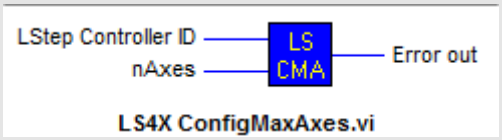
LSX_LStepSave			
Beschreibung	Diese Funktion löst das Speichern der aktuellen Konfiguration im nicht flüchtigen Speicher der LSTEP-Steuerung aus. Dabei wird auf die Rückmeldung der Steuerung gewartet, die anzeigt, dass das Speichern beendet wurde.		
Delphi	function LSX_LStepSave(LSID: Integer): Integer;		
C++	int LStepSave();		
LabView:			
Parameter	-		
Beispiel	LSX.LStepSave() ;		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!save	-

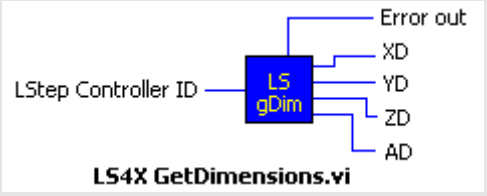
<table>
<tr><th colspan="4">LSX_SoftwareReset</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Die Steuerung wird neugestartet. Dabei lädt die Steuerung die zuletzt gespeicherten Daten.</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_SoftwareReset(LSID: Integer): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int SoftwareReset();</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td>Parameter:</td><td colspan="3">-</td></tr>
<tr><td>Beispiel:</td><td colspan="3">LSX.SoftwareReset();</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>3</td><td>!reset</td><td>-</td></tr>
</table>

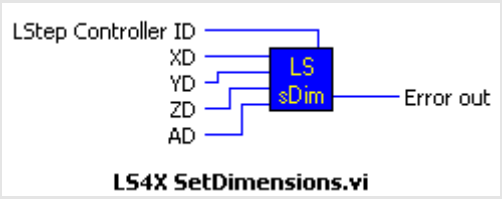
4.2.5 Achs- und Motorkonfiguration

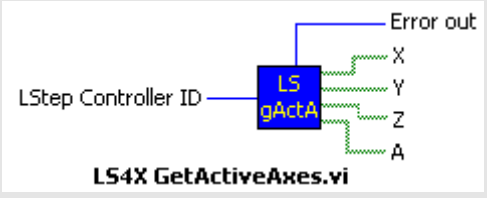
LSX_ValidConfig			
Beschreibung:	Aktiviert die ValidConfig abhänigen Parameter auf der jeweils angegebenen Achse. Position wird genullt, neue Kalibrierung notwendig.		
Delphi:	function LSX_ValidConfig(LSID: Integer; AxisInt: Integer): Integer;		
C++:	int ValidConfig (int lAxisint);		
LabView:	Nicht unterstützt		
Parameter:	AxisInt	0 = Alle Achsen nutzen 1 = x-Achse nutzen 2 = y-Achse nutzen 3 = z-Achse nutzen 4 = A-Achse nutzen	
Beispiel:	LSX.ValidConfig(0); // Alle Achsen nutzen		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!validconfig	-

LSX_ValidPar			
Beschreibung:	Überträgt geänderte Parameter Achsenweise an die Steuerung. Keine neue Kalibrierung notwendig		
Delphi:	function LSX_ValidPar(LSID: Integer; AxisInt: Integer): Integer;		
C++:	int ValidPar(int lAxisInt);		
LabView:	Nicht unterstützt		
Parameter:	AxisInt	0 = Alle Achsen übertragen 1 = x-Achse übertragen 2 = y-Achse übertragen 3 = z-Achse übertragen 4 = a-Achse übertragen	
Beispiel:	LSX.ValidPar(0); // Alle Achsen übertragen		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!validpar	-

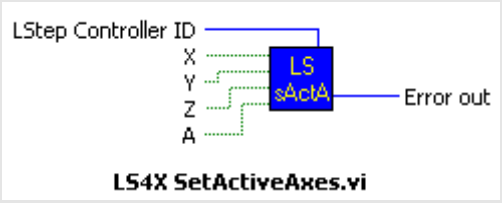
LSX_ConfigMaxAxes			
Beschreibung:	Konfiguriert die Anzahl der Achsen.		
Delphi:	function LSX_ConfigMaxAxes(LSID: Integer; nAxes: Integer): Integer;		
C++:	int ConfigMaxAxes(int nAxes);		
LabView:			
Parameter:	nAxes	Anzahl Achsen	
Beispiel:	LSX.ConfigMaxAxes(2); // Steuerung hat zwei Achsen		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!configmaxaxis	-

LSX_GetDimensions		
Beschreibung:	Abfrage der eingestellten Dimensionen der Achsen.	
Delphi:	function LSX_GetDimensions(LSID: Integer; var XD, YD, ZD, AD: Integer): Integer;	
C++:	int GetDimensions(int *plXD, int *plYD, int *plZD, int *plAD);	
LabView:		
Parameter:	XD, YD, ZD, AD	Dimensionswerte 0 = Microsteps 1 = µm 2 = Millimeter 3 = Grad 4 = Umdrehungen
Beispiel:	LSX.GetDimensions(&XD, &YD, &ZD, &AD);	
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?dim	-

LSX_SetDimensions			
Beschreibung:	Setzen der Dimensionen der Achsen		
Delphi:	function LSX_SetDimensions(LSID: Integer; XD, YD, ZD, AD: Integer): Integer;		
C++:	int SetDimensions(int lXD,int lYD,int lZD,int lAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Dimensionswerte 0 = Microsteps 1 = µm 2 = Millimeter 3 = Grad 4 = Umdrehungen	
Beispiel:	LSX.SetDimensions(3, 2, 2); // X-Achse in Grad; Y und Z in mm		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!dim	-

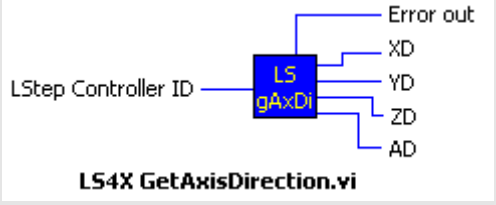
LSX_GetActiveAxes	
Beschreibung:	Liefert die freigegebenen Achsen.
Delphi:	function LSX_GetActiveAxes(LSID: Integer; var Flags: Integer): Integer;
C++:	int GetActiveAxes(int *plFlags);
LabView:	

Parameter:	Flags	32-Bit-Integer, welcher nach Aufruf der Funktion in den Bits 0-4 eine Bit-Maske enthält. Bit 0 = X-Achse Bit 1 = Y-Achse … Wert 0 = Achse deaktiviert Wert 1 = Achse aktiviert	
Beispiel:	LSX.GetActiveAxes(&Flags);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?axis	-

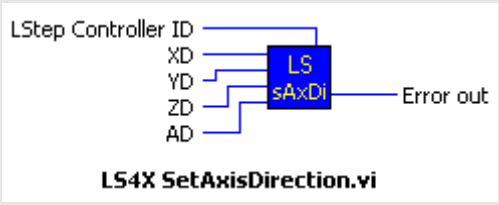
LSX_SetActiveAxes			
Beschreibung:	Setzt die Freigabe der Achsen.		
Delphi:	function LSX_SetActiveAxes(LSID: Integer; Flags: Integer): Integer;		
C++:	int SetActiveAxes(int Flags);		
LabView:			
Parameter:	Flags	Bit-Maske zur Freigabe der Achsen Bit 0 = X-Achse Bit 1 = Y-Achse … Wert 0 = Achse deaktiviert Wert 1 = Achse aktiviert	
Beispiel:	LSX.SetActiveAxes(3); /* X- und Y-Achse freigeben (Bits 0 u. 1 gesetzt), Z-Achse nicht freigegeben (Bit 2 = 0) */		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!axis	-

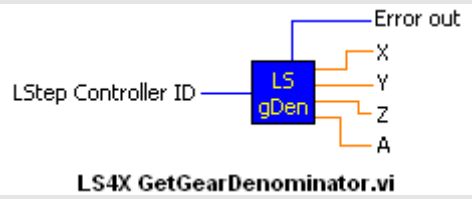
LSX_GetAxisMap			
Beschreibung	Befehl zum Auslesen der Zuordnung der Hardwareachsen zu den Achsen des User-Interface		
Delphi	function LSX_GetAxisMap (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++	int GetAxisMap (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter	X, Y, Z, A	Nummerierung der Achsen: 1, 2, 3 oder 4	
Beispiel	LSX.GetAxisMap(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?axismap	-

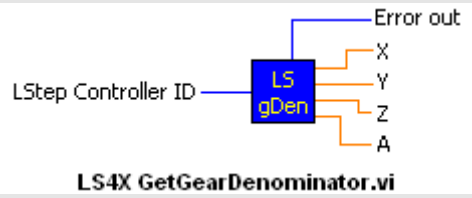
LSX_SetAxisMap			
Beschreibung	Befehl zum Zuordnen der Hardwareachsen zu den Achsen des User-Interface		
Delphi	function LSX_SetAxisMap (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++	int SetAxisMap (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter	X, Y, Z, A	Nummerierung der Achsen: 1, 2, 3 oder 4	
Beispiel	LSX.SetAxisMap(2, 1, 4, 3);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!axismap	ValidConfig

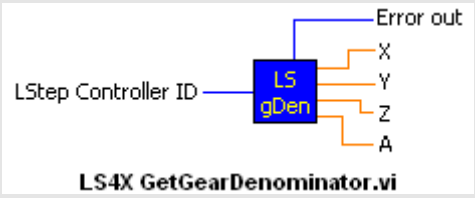
LSX_GetAxisDirection	
Beschreibung	Funktion zum Abfragen der Drehrichtungs-Umkehr.
Delphi	function LSX_GetAxisDirection(LSID: Integer; var XD, YD, ZD, AD: Integer): Integer;
C++	int GetAxisDirection(int *plXD, int *plYD, int *plZD, int *plAD);
LabView:	

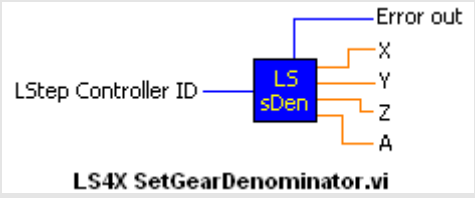
Parameter	XD, YD, ZD, AD	32-Bit-Integer mit Angabe der Drehrichtung. 0 = Normale Drehrichtung 1 = Drehrichtungs-Umkehr	
Beispiel	LSX.GetAxisDirection(&XD, &YD, &ZD, &AD);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?axisdir	-

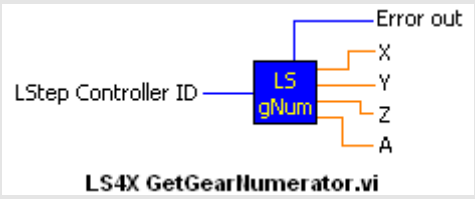
LSX_SetAxisDirection			
Beschreibung	Funktion zum Setzen der Drehrichtungs-Umkehr.		
Delphi	function LSX_SetAxisDirection(LSID: Integer; XD, YD, ZD, AD: Integer): Integer;		
C++	int SetAxisDirection (int lXD, int lYD, int lZD, int lAD);		
LabView:			
Parameter	XD, YD, ZD, AD	32-Bit-Integer mit Angabe der Drehrichtung. 0 = Normale Drehrichtung 1 = Drehrichtungs-Umkehr	
Beispiel	LSX.SetAxisDirection(1, 0, 0, 0); // Drehrichtung der X-Achse umkehren		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!axisdir	ValidConfig

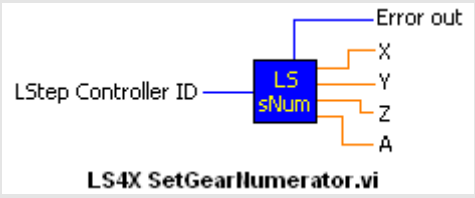
LSX_GetGear			
Beschreibung:	Funktion zum Abfragen der Getriebeübersetzung. (nicht für LSTEP express)		
Delphi:	function LSX_GetGear(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetGear (double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	X, Y, Z, A	Ausgelesene Getriebeübersetzung (Motorseitig/Antriebsseitig)	
Beispiel:	LSX.GetGear(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?gear	-

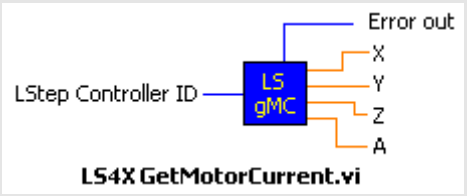
LSX_SetGear			
Beschreibung:	Funktion zum Setzen der Getriebeübersetzung. (nicht für LSTEP express)		
Delphi:	function LSX_SetGear(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int SetGear (double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	X, Y, Z, A	Zu setzende Getriebeübersezung (Motorseitig/Antriebsseitig)	
Beispiel:	LSX.SetGear (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!gear	-

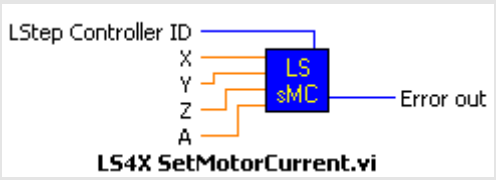
LSX_GetGearDenominator			
Beschreibung:	Nenner der Getriebeübersetzung abfragen. (nur für LSTEP express)		
Delphi:	function LSX_GetGearDenominator(LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetGearDenominator(int *plX, int *plY, int *plZ, int *plA);		
LabView:			
Parameter:	X, Y, Z, A	Nenner der Getriebeübersetzung (Motorseitig)	
Beispiel:	LSX.GetGearDenominator(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?geardenominator	-

LSX_SetGearDenominator			
Beschreibung:	Nenner der Getriebeübersetzung einstellen. (nur für LSTEP express)		
Delphi:	function LSX_SetGearDenominator(LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetGearDenominator(int lX, int lY, int lZ, int lA);		
LabView:			
Parameter:	X, Y, Z, A	Wert des Nenners der Getriebeübersetzung (Motorseitig)	
Beispiel:	LSX.SetGearDenominator(2, 1, 1, 1); // Getriebeübersetzung 2/γ bei x		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!geardenominator	ValidConfig

LSX_GetGearNumerator			
Beschreibung:	Zähler der Getriebeübersetzung abfragen. (nur für LSTEP express)		
Delphi:	function LSX_GetGearNumerator(LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetGearNumerator(int *plX, int *plY, int *plZ, int *plA);		
LabView:			
Parameter:	X, Y, Z, A	Zähler der Getriebeübersetzung (Abtriebsseitig)	
Beispiel:	LSX.GetGearNumerator(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?gearnumerator	-

LSX_SetGearNumerator			
Beschreibung	Zähler der Getriebeübersetzung einstellen. (nur für LSTEP express)		
Delphi	function LSX_SetGearNumerator(LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++	int SetGearNumerator(int lX, int lY, int lZ, int lA);		
LabView:			
Parameter	X, Y, Z, A	Wert des Zählers der Getriebeübersetzung (Abtriebsseitig)	
Beispiel	LSX.SetGearNumerator(4, 1, 1, 1);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!gearnumerator	ValidConfig

LSX_GetMotorCurrent			
Beschreibung:	Funktion zum Abfragen des eingestellten Motorstroms.		
Delphi:	function LSX_GetMotorCurrent(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetMotorCurrent(double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	X, Y, Z, A	Motorstrom in A	
Beispiel:	LSX.GetMotorCurrent(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	LSTEP 2000: ?cur LSTEP express: ?motorcurrent	-

LSX_SetMotorCurrent			
Beschreibung:	Funktion zum Setzen des Motorstroms.		
Delphi:	function LSX_SetMotorCurrent(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetMotorCurrent (double dX,double dY,double dZ,double dA);		
LabView:			
Parameter:	X, Y, Z, A	Motorstrom in A	
Beispiel:	LSX.SetMotorCurrent(1.5, 1.5, 1.0, 1.0); // Motorstrom X u. Y 1.5 Ampere; Z u. A 1.0 Ampere		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	LSTEP2000: !cur LSTEP express: !motorcurrent	ValidPar / ValidConfig

LSX_GetMotorpeakCurrent			
Beschreibung:	Funktion zum Abfragen des eingestellten Motorspitzenstroms.		
Delphi:	function LSX_GetMotorpeakCurrent(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetMotorpeakCurrent(double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Motorstrom in A	
Beispiel:	LSX.GetMotorpeakCurrent(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorpeakcurrent	-

LSX_SetMotorpeakCurrent			
Beschreibung:	Funktion zum Setzen des Motorspitzenstroms.		
Delphi:	function LSX_SetMotorpeakCurrent(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetMotorpeakCurrent (double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Motorstrom in A	
Beispiel:	LSX.SetMotorpeakCurrent(1.5, 1.5, 1.0, 1.0); // Motorspitzenstrom X u. Y 1.5 Ampere; Z u. A 1.0 Ampere		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorpeakcurrent	ValidPar / ValidConfig

LSX_GetMotortempSensor			
Beschreibung:	Funktion zur Zustandsabfrage des Motortemperatursensors.		
Delphi:	function LSX_GetMotortempSensor(LSID: Integer; var X,Y,Z,R: LongBool): Integer;		
C++:	int GetMotortempSensor (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Motortemperatursensorzustand	
Beispiel:	LSX.GetMotortempSensor(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motortempsensor	-

LSX_SetMotortempSensor			
Beschreibung:	Funktion zum aktivieren/deaktivieren des Motortemperatursensors.		
Delphi:	function LSX_SetMotortempSensor(LSID: Integer; X,Y,Z,R : longBool): Integer;		
C++:	int SetMotortempSensor (BOOL bX, BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Motortemperatursensorerzustand	
Beispiel:	LSX.SetMotortempSensor(True,True,False,False); // Motortemperatursensoren X u. Y werden aktiviert; Z u. A werden deaktiviert		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motortempsensor	ValidPar / ValidConfig

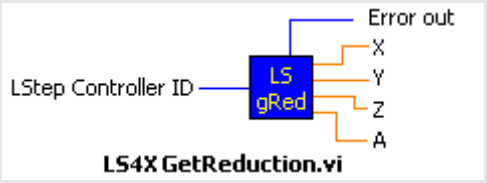
LSX_GetMotortempSensormin			
Beschreibung:	Funktion zur Abfrage des Minimal akkzeptierbaren Wertes für die Motortemperatur.		
Delphi:	function LSX_GetMotortempSensormin(LSID: Integer; var X,Y,Z,A:Integer): Integer;		
C++:	int GetMotortempSensormin (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Minimaler Widertandswert der Motortemperatursensoren	
Beispiel:	LSX.GetMotortempSensormin(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motortempsensor-minimum	-

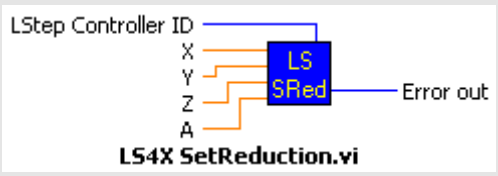
LSX_SetMotortempSensormin			
Beschreibung:	Funktion zum setzen eines minimal akkzeptierbaren Wertes für den Widerstand des Motortemperatursensors.		
Delphi:	function LSX_SetMotortempSensormin(LSID: Integer; X,Y,Z,A : Integer): Integer;		
C++:	int SetMotortempSensormin (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Minimaler Widerstandswert der Motortemperatursensoren	
Beispiel:	LSX.SetMotortempSensormin(100,200,300,400); // Motortemperatursensor minimaler Widerstandswert für X ist 100 Ω.		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motortempsensor-minimum	ValidPar / ValidConfig

LSX_GetMotortempSensormax			
Beschreibung:	Funktion zur Abfrage des Maximal akkzeptierbaren Wertes für die Motortemperatur.		
Delphi:	function LSX_GetMotortempSensormax(LSID: Integer; var X, Y, Z, A:Integer): Integer;		
C++:	int GetMotortempSensormax (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Maximaler Widertandswert der Motortemperatursensoren	
Beispiel:	LSX.GetMotortempSensormax(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motortempsensor-maximum	-

LSX_SetMotortempSensormax			
Beschreibung:	Funktion zum setzen eines maximal akkzeptierbaren Wertes für den Widerstand des Motortemperatursensors.		
Delphi:	function LSX_SetMotortempSensormax (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetMotortempSensormax (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Maximaler Widerstandswert der Motortemperatursensoren	
Beispiel:	LSX.SetMotortempSensormax(100,200,300,400); // Motortemperatursensor maximaler Widerstandswert für X ist 100 Ω.		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motortempsen-sormaximum	ValidPar / ValidConfig

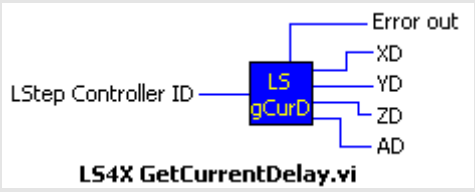
LSX_GetMotortempSensorValue			
Beschreibung:	Funktion zur Abfrage des Widerstandwertes vom Motortemperatursensor.		
Delphi:	function LSX_GetMotortempSensorValue(LSID: Integer; var X, Y, Z, A:Integer): Integer;		
C++:	int GetMotortempSensorValue (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Widertandswert des Motortemperatursensores	
Beispiel:	LSX.GetMotortempSensorValue(& X,&Y,& Z,& A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motortempsensor-value	-

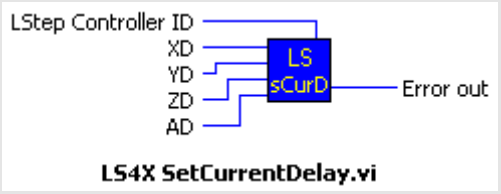
LSX_GetReduction			
Beschreibung:	Abfragen der eingestellten Stromabsenkung.		
Delphi:	function LSX_GetReduction(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetReduction(double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	X, Y, Z, A	Eingestellte Stromabsenkung in %	
Beispiel:	LSX.GetReduction(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?reduction	-

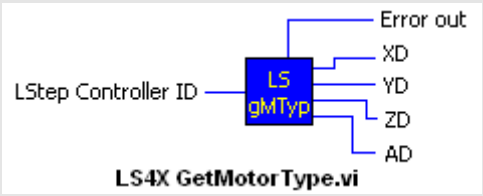
LSX_SetReduction			
Beschreibung:	Einstellen des Verhältnisses, um welches der Motornennstrom reduziert wird, wenn die Stromabsenkung aktiv ist.		
Delphi:	function LSX_SetReduction(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetReduction(double dX, double dY, double dZ, double dA);		
LabView:			
Parameter:	X, Y, Z, A	Wert der Stromabsenkung. LSTEP Serie = 0.0 – 1.0 LSTEP express Serie = 0 – 100 in %	

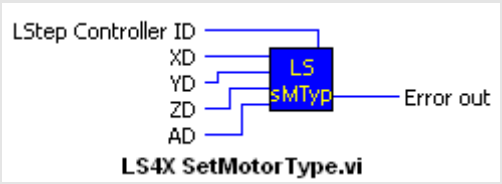
Beispiel:	LSX.SetReduction(0.1, 0.7, 0.5, 0.5); //LSTEP 2000 Familie
	/* Ruhestrom X-Achse = 0.1*Nennstrom; Y-Achse = 0.7*Nennstrom ... */		

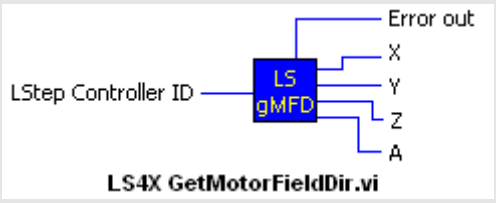
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!reduction	-

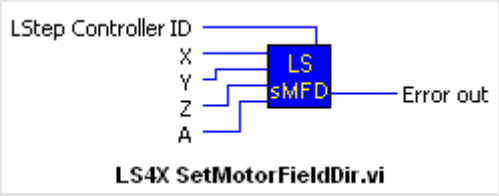
LSX_GetCurrentDelay			
Beschreibung:	Gibt die Zeitverzögerung bis zum Aktivieren der Stromabsenkung an.		
Delphi:	function LSX_GetCurrentDelay(LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetCurrentDelay(int *plX, int *plY, int *plZ, int *plA);		
LabView:			
Parameter:	X, Y, Z, A:	Zeitverzögerung in ms	
Beispiel:	LSX.SetCurrentDelay(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?curdelay	-

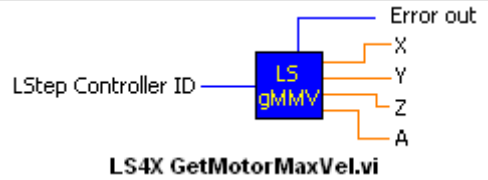
LSX_SetCurrentDelay			
Beschreibung:	Setzt die Zeitverzögerung für das Aktivieren der Stromabsenkung.		
Delphi:	function LSX_SetCurrentDelay(LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetCurrentDelay(int lX, int lY, int lZ, int lA);		
LabView:			
Parameter:	X, Y, Z, A	Zeitverzögerung in ms	
Beispiel:	LSX.SetCurrentDelay(100, 300, 1000, 0) ;		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!curdelay	-

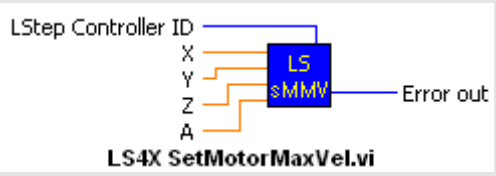
LSX_GetMotorType			
Beschreibung:	Funktion zum Auslesen des eingestellten Motortyps. (nur für LSTEP express)		
Delphi:	function LSX_GetMotorType(LSID: Integer; var X: Integer; var Y: Integer; var Z: Integer; var A: Integer): Integer;		
C++:	Int GetMotorType(int *plX, int *plY, int *plZ, int *plA)		
LabView:			
Parameter:	X, Y, Z, A	Motortyp 0 = rotativer 2-Phasen Schrittmotor 1 = linearer 2-Phasen Schrittmotor 2 = rotativer 3-Phasen Schrittmotor 3 = linearer 3-Phasen Schrittmotor 4 = rotativer 2-Phasen Servomotor 5 = linearer 2-Phasen Servomotor 6 = rotativer 3-Phasen Servomotor 7 = linearer 3-Phasen Servomotor	
Beispiel:	LSX.GetMotorType(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motortype	-

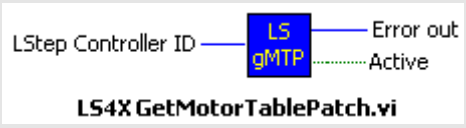
<table>
<tr><th colspan="4">LSX_SetMotorType</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Funktion zum Einstellen des Motortyps.
(nur für LSTEP express)</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_SetMotorType(LSID: Integer; X: Integer; var Y: Integer; var Z: Integer; var A: Integer): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int SetMotorType(int lX, int lY, int lZ, int lA);</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td>Parameter:</td><td>X, Y, Z, A</td><td colspan="2">Motortyp
0 = rotativer 2-Phasen Schrittmotor
1 = linearer 2-Phasen Schrittmotor
2 = rotativer 3-Phasen Schrittmotor
3 = linearer 3-Phasen Schrittmotor
4 = rotativer 2-Phasen Servomotor
5 = linearer 2-Phasen Servomotor
6 = rotativer 3-Phasen Servomotor
7 = linearer 3-Phasen Servomotor</td></tr>
<tr><td>Beispiel:</td><td colspan="3">LSX.SetMotorType(5, 4, 4, 4);</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>2</td><td>!motortype</td><td>ValidConfig</td></tr>
</table>

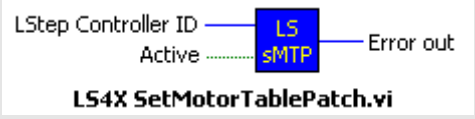
LSX_GetMotorFieldDir			
Beschreibung:	Funktion zum Abfragen des Felddrehsinns des Motors. (nur für LSTEP express)		
Delphi:	function LSX_GetMotorFieldDir(LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetMotorFieldDir(int *plX, int *plY, int *plZ, int *plA);		
LabView:			
Parameter:	X, Y, Z, A	Aktuell eingestellte Drehrichtung 0 = normaler Felddrehsinn 1 = umgekehrter Felddrehsinn	
Beispiel:	LSX.GetMotorFieldDir(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorfielddir	-

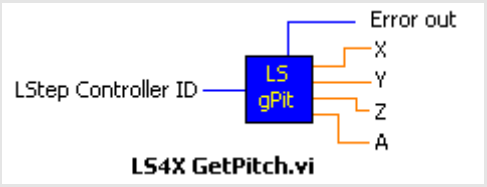
LSX_SetMotorFieldDir			
Beschreibung:	Funktion zum Setzen des Felddrehsinns des Motors. (nur für LSTEP express)		
Delphi:	function LSX_SetMotorFieldDir(LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetMotorFieldDir(int lX, int lY, int lZ, int lA);		
LabView:			
Parameter:	X, Y, Z, A	Aktuell einzustellende Drehrichtung 0 = normaler Felddrehsinn 1 = Felddrehsinn umkehren	
Beispiel:	LSX.SetMotorFieldDir(1, 0, 0, 0); // Drehrichtung der X-Achse umkehren		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorfielddir	ValidConfig

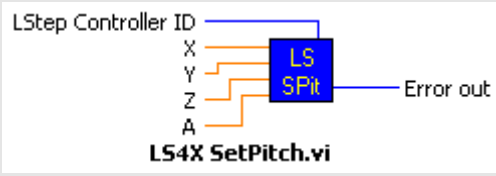
LSX_GetMotorMaxVel			
Beschreibung:	Auslesen der maximalen Motor-Drehzahl bzw. Geschwindigkeit. (nur für LSTEP express)		
Delphi:	function LSX_GetMotorMaxVel(LSID: Integer; var XD, YD, ZD, AD: Double): Integer;		
C++:	int GetMotorMaxVel(double *pdXD, double *pdYD, double *pdZD, double *pdAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Motor-Drehzahl bzw. Geschwindigkeit. Bei rotativem Motor in U/min Bei linearem Motor in mm/s	
Beispiel:	LSX.GetMotorMaxVel(&XD, &YD, &ZD, &AD);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motormaxvel	-

LSX_SetMotorMaxVel			
Beschreibung:	Einstellen der maximalen Motor-Drehzahl bzw. Geschwindigkeit. (nur für LSTEP express)		
Delphi:	function LSX_SetMotorMaxVel(LSID: Integer; XD, YD, ZD, AD: Double): Integer;		
C++:	int SetMotorStandForce(double dXD, double dYD, double dZD, double dAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Motor-Drehzahl bzw. Geschwindigkeit. Bei rotativem Motor in U/min Bei linearem Motor in mm/s	
Beispiel:	LSX.GetMotorMaxVel(1000, 1000, 500, 250);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motormaxvel	ValidConfig

LSX_GetMotorTablePatch			
Beschreibung:	Gibt an, ob die Korrekturtabelle aktiviert ist. (nicht für LSTEP express)		
Delphi:	function LSX_GetMotorTablePatch (LSID: Integer; var bActive: LongBool): Integer;		
C++:	int GetMotorTablePatch (BOOL *pbActive);		
LabView:			
Parameter:	bActive	Status der Korrekturtabelle True = Tabelle ist aktiviert False = Tabelle ist deaktiviert	
Beispiel:	LSX.GetMotorTablePatch(&Active);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?motorpatch	-

LSX_SetMotorTablePatch			
Beschreibung:	Die Korrekturtabelle wird aktiviert. Die Korrekturtabelle wurde für einen Sondermotor durch Messung ermittelt. Korrekturtabellen können auf Kundenwunsch ermittelt werden. (nicht für LSTEP express)		
Delphi:	function LSX_SetMotorTablePatch (LSID: Integer; bActive: LongBool): Integer;		
C++:	int SetMotorTablePatch (BOOL bActive);		
LabView:			
Parameter:	bActive	Aktivierung der Korrekturtabelle True = Tabelle ist aktiviert False = Tabelle ist deaktiviert	
Beispiel:	LSX.SetMotorTablePatch(True);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!motorpatch	-

LSX_GetPitch			
Beschreibung:	Liefert den eingestellten Wert der Spindelsteigung.		
Delphi:	function LSX_GetPitch(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetPitch(double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	X, Y, Z, A	Spindelsteigungen in mm/Umdrehung	
Beispiel:	LSX.GetPitch(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?pitch	-

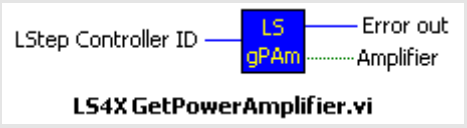
LSX_SetPitch			
Beschreibung:	Setzt die Spindelsteigung der Achse.		
Delphi:	function LSX_SetPitch(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetPitch(double dX, double dY, double dZ, double dA);		
LabView:			
Parameter:	X, Y, Z, A	Spindelsteigungen in mm/Umdrehung	
Beispiel:	LSX.SetPitch(4, 4, 4, 4); // Spindelsteigungen aller Achsen auf 4 mm setzen		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!pitch	ValidConfig

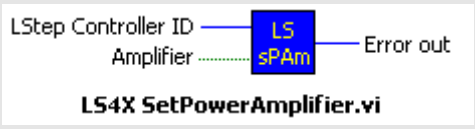
LSX_GetXYAxisComp			
Beschreibung	Abfrage XY-Achsüberlagerung (nicht für LSTEP express)		
Delphi	function LSX_GetXYAxisComp(LSID: Integer; var Value: Integer): Integer;		
C++	int GetXYAxisComp (int *plValue);		
LabView:			
Parameter	Value	Modus der Achsüberlagerung (siehe LSTEP-Dokumentation)	
Beispiel	LSX.SetXYAxisComp(&mode) ;		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?xycomp	-

LSX_SetXYAxisComp			
Beschreibung	XY-Achsüberlagerung aktivieren (nicht für LSTEP express)		
Delphi	function LSX_SetXYAxisComp(LSID: Integer; Value: Integer): Integer;		
C++	int SetXYAxisComp(int lValue);		
LabView:			
Parameter	Value	Modus der Achsüberlagerung (siehe LSTEP-Dokumentation)	
Beispiel	LSX.SetXYAxisComp(1) ;		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!xycomp	-

LSX_GetStopPolarity			
Beschreibung:	Auslesen der Polarität des Stopp-Eingangs.		
Delphi:	function LSX_GetStopPolarity(LSID: Integer; var bHighActiv: LongBool): Integer;		
C++:	int GetStopPolarity(BOOL *pbHighActiv);		
LabView:			
Parameter:	bHighActiv	Wert der eingestellten Polarität True = Stopp-Eingang high-aktiv False = Stopp-Eingang low-aktiv	
Beispiel:	LSX.GetStopPolarity(&HighActiv);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?stoppol	-

LSX_SetStopPolarity			
Beschreibung:	Funktion zum Einstellen der Polarität des Stopp-Eingangs.		
Delphi:	function LSX_SetStopPolarity(LSID: Integer; bHighActiv: LongBool): Integer;		
C++:	int SetStopPolarity(BOOL bHighActiv);		
LabView:			
Parameter:	bHighActiv	Wert der eingestellten Polarität True = Stopp-Eingang high-aktiv False = Stopp-Eingang low-aktiv	
Beispiel:	LSX.SetStopPolarity(False); //Der Stopeingang ist lowaktiv.		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!stoppol	-

LSX_GetPowerAmplifier			
Beschreibung:	Gibt an, ob die Endstufen ein- oder ausgeschaltet sind. (nur bei LS44-Steuerung und bei LSTEP express)		
Delphi:	function LSX_GetPowerAmplifier(LSID: Integer; var bAmplifier: LongBool): Integer;		
C++:	int GetPowerAmplifier (BOOL *pbAmplifier);		
LabView:			
Parameter:	bAmplifier	Status der Endstufen True = Alle Endstufen sind eingeschaltet False = Alle Endstufen sind ausgeschaltet	
Beispiel:	LSX.GetPowerAmplifier(&Amplifier);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3 (siehe Beschreibung)	?pa	-

LSX_SetPowerAmplifier			
Beschreibung:	Schaltet die Endstufen der Steuerung Ein bzw. Aus. (nur bei LS44-Steuerung und bei LSTEP express)		
Delphi:	function LSX_SetPowerAmplifier (LSID: Integer; bAmplifier: LongBool): Integer;		
C++:	int SetPowerAmplifier(BOOL bAmplifier);		
LabView:			
Parameter:	bAmplifier	Beabsichtigter Zustand der Endstufen True = Alle Endstufen sind eingeschaltet False = Alle Endstufen sind ausgeschaltet	
Beispiel:	LSX.SetPowerAmplifier(True); // Die Endstufen einschalten		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3 (siehe Beschreibung)	!pa	-

LSX_GetModulo			
Beschreibung:	Fragt an ob die Achsen sich im Modulo-Betrieb befinden.		
Delphi:	function LSX_GetModulo(LSID: Integer; var X,Y,Z,A: LongBool): Integer;		
C++:	int GetModulo (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	True = Achse im Modulo Betrieb False = Achse im normalen Betrieb	
Beispiel:	LSX.GetModulo(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?modulo	-

LSX_SetModulo			
Beschreibung:	Schaltet die Modulo Betriebsart für die Achsen ein		
Delphi:	function LSX_SetModulo(LSID: Integer; X,Y,Z,A: LongBool): Integer;		
C++:	int SetModulo (BOOL bX, BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	True = Achse auf Modulo Betrieb schalten False = Achse auf normalen Betrieb schalten	
Beispiel:	LSX.SetModulo(True,True,True,True); // Die Achsen auf Modulo-Betrieb schalten.		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!modulo	ValidConfig

LSX_GetModulomode			
Beschreibung:	Fragt an in welchen Modulomodus die Achsen sich befinden.		
Delphi:	function LSX_GetModuloMode(LSID: Integer; var X,Y,Z,A: Integer): Integer;		
C++:	int GetModuloMode (int*plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y, Z,A	0 = Move wird ohne Optimierung gefahren 1 = Zielposition wird immer in positiver Drehrichtung angefahren. 2 = Zielposition wird immer in negativer Drehrichtung angefahren. 3 = Es wir immer die kürzeste Verbindung zwischen Start und Zielposition gefahren.	
Beispiel:	LSX.GetModulomode(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?modulomode	-

LSX_SetModulomode			
Beschreibung:	Setzt den Modulomodus für die Achsen.		
Delphi:	function LSX_SetModuloMode(LSID: Integer; X,Y,Z,A: Integer): Integer;		
C++:	int SetModulomode (int lX, int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z, A	0 = Move wird ohne Optimierung gefahren 1 = Zielposition wird immer in positiver Drehrichtung angefahren. 2 = Zielposition wird immer in negativer Drehrichtung angefahren. 3 = Es wir immer die kürzeste Verbindung zwischen Start und Zielposition gefahren.	
Beispiel:	LSX.SetModulomode(0,0,0,0); //Alle Achsen ohne Optimierung.		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!modulomode	-

LSX_GetMotorPoleScale			
Beschreibung:	Fragt die Polteilung bei Linearmotoren an.		
Delphi:	function LSX_GetMotorPoleScale (LSID: Integer; X,Y,Z,A: double): Integer;		
C++:	Int GetMotorPoleScale(int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Polteilung in mm	
Beispiel:	LSX.GetMotorPoleScale(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorpolescale	-

LSX_SetMotorPoleScale			
Beschreibung:	Einstellen der Polteilung von Linearmotoren		
Delphi:	function LSX_SetMotorPoleScale(LSID: Integer; X,Y,Z,A : double) : Integer;		
C++:	Int SetMotorPoleScale (int lLSID, double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Polteilung in mm	
Beispiel:	LSX.SetMotorPoleScale(10,10,10,10);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorpolescale	-

LSX_GetMotorPolePairs			
Beschreibung:	Fragt die Polpaarzahl bei rotativen Motoren ab.		
Delphi:	function LSX_GetMotorPolePairs (LSID: Integer; X,Y,Z,A: integer): Integer;		
C++:	Int GetMotorPolePairs(int lLSID, int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Polpaaranzahl	
Beispiel:	LSX.GetMotorPolePairs(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorpolpairs	-

LSX_SetMotorPolePairs			
Beschreibung:	Stellt die Polpaarzahl bei rotativen Motoren ein.		
Delphi:	function LSX_SetMotorPolePairs (LSID: Integer; X,Y,Z,A : integer): Integer;		
C++:	Int SetMotorPolePairs (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Polpaarzahl	
Beispiel:	LSX.SetPolePairs(10,10,10,10);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorpolepairs	-

LSX_GetMotorPolePairRes			
Beschreibung:	Fragt die Schrittauflösung der Polpaare und/oder der Polteilung ab.		
Delphi:	function LSX_GetMotorPolePairRes (LSID: Integer; X,Y,Z,A: integer): Integer;		
C++:	Int GetMotorPolePairRes(int lLSID, int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Schrittauflösung in mm	
Beispiel:	LSX.GetMotorPolePairRes(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorpolepairres	-

LSX_SetMotorPolePairRes			
Beschreibung:	Stellt die Schrittauflösung der Polpaare und/oder der Poleteilung ein		
Delphi:	function LSX_SetMotorPolePairRes(LSID: Integer; X,Y,Z,A : Integer): Integer;		
C++:	Int SetMotorPolePairRes (int lLSID, int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Schrittauflösung in mm	
Beispiel:	LSX.SetMotorPolePairRes(50,50,50,50);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorpolepairres	ValidConfig

LSX_GetMotorBrake			
Beschreibung:	Liest den Modus der Motorbremsenausgänge aus.		
Delphi:	function LSX_GetMotorBrake (LSID: Integer; X,Y,Z,A: integer): Integer;		
C++:	Int GetMotorBrake(int lLSID, int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Modus der Motorbremsen: 0: Ausgang schaltet nach GND 1: Ausgang schaltet in Abhängigkeit der Endstufe 2: Ausgang schaltet nach 24V bzw. 12V bei PC-Variante ohne 24V Versorgung (I/O) 3: Verwendung als digitaler Ausgang, siehe Befehl	
Beispiel:	LSX.GetMotorBrake(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorbrake	-

LSX_SetMotorBrake			
Beschreibung:	Stellt den Modus der Motorbremsenausgänge ein.		
Delphi:	function LSX_SetMotorBrake(LSID: Integer; X,Y,Z,A : Integer): Integer;		
C++:	Int SetBrake (int lLSID, int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Modus der Motorbremsen: 0: Ausgang schaltet nach GND 1: Ausgang schaltet in Abhängigkeit der Endstufe 2: Ausgang schaltet nach 24V bzw. 12V bei PC-Variante ohne 24V Versorgung (I/O) 3: Verwendung als digitaler Ausgang, siehe Befehl	
Beispiel:	LSX.SetMotorBrake(0,0,0,0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorbrake	ValidConfig

LSX_GetMotorBrakeSwitchOnDelay			
Beschreibung:	Befehl zum Auslesen der Verzögerung zwischen dem Einschalten der Endstufe und des Motorbremsenausgangs (Modus "MotorBrake 1"). Bei negativen Werten wird die Endstufe verzögert eingeschaltet. Bei positiven Werten wird der Ausgang verzögert eingeschaltet. Ausgänge können bspw. zum Schalten von Motorbremsen oder Türverriegelungen genutzt werden.		
Delphi:	function LSX_GetMotorBrakeSwitchOnDelay (LSID: Integer; X,Y,Z,A: integer): Integer;		
C++:	Int GetMotorBrakeSwitchOnDelay (int lLSID, int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Verzögerung in ms, Bereich: -5000 ms bis 5000 ms	
Beispiel:	LSX.GetMotorBrakeSwitchOnDelay (&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorbrakeswitchondelay	-

LSX_SetMotorBrakeSwitchOnDelay			
Beschreibung:	Befehl zum Einstellen der Verzögerung zwischen dem Einschalten der Endstufe und des Motorbremsenausgangs (Modus "MotorBrake 1"). Bei negativen Werten wird die Endstufe verzögert eingeschaltet. Bei positiven Werten wird der Ausgang verzögert eingeschaltet. Ausgänge können bspw. zum Schalten von Motorbremsen oder Türverriegelungen genutzt werden.		
Delphi:	function LSX_SetMotorBrakeSwitchOnDelay (LSID: Integer; X,Y,Z,A : Integer): Integer;		
C++:	Int SetBrakeSwitchOnDelay (int lLSID, int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Verzögerung in ms, Bereich: -5000 ms bis 5000 ms	
Beispiel:	LSX.SetMotorBrakeSwitchOnDelay (0,0,0,0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorbrakeswitchondelay	ValidConfig

LSX_GetMotorBrakeSwitchOffDelay			
Beschreibung:	Befehl zum Auslesen der Verzögerung zwischen dem Ausschalten der Endstufe und des Motorbremsenausgangs (Modus "MotorBrake 1"). Bei negativen Werten wird die Endstufe verzögert ausgeschaltet. Bei positiven Werten wird der Ausgang verzögert ausgeschaltet. Ausgänge können bspw. zum Schalten von Motorbremsen oder Türverriegelungen genutzt werden.		
Delphi:	function LSX_GetMotorBrakeSwitchOffDelay (LSID: Integer; X,Y,Z,A: integer): Integer;		
C++:	Int GetMotorBrakeSwitchOffDelay (int lLSID, int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Verzögerung in ms, Bereich: -5000 ms bis 5000 ms	
Beispiel:	LSX.GetMotorBrakeSwitchOffDelay (&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorbrakeswitchoffdelay	-

LSX_SetMotorBrakeSwitchOffDelay			
Beschreibung:	Befehl zum Einstellen der Verzögerung zwischen dem Ausschalten der Endstufe und des Motorbremsenausgangs (Modus "MotorBrake 1"). Bei negativen Werten wird die Endstufe verzögert ausgeschaltet. Bei positiven Werten wird der Ausgang verzögert ausgeschaltet. Ausgänge können bspw. zum Schalten von Motorbremsen oder Türverriegelungen genutzt werden.		
Delphi:	function LSX_SetMotorBrakeSwitchOffDelay (LSID: Integer; X,Y,Z,A : Integer): Integer;		
C++:	Int SetMotorBrakeSwitchOffDelay (int lLSID, int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Verzögerung in ms, Bereich: -5000 ms bis 5000 ms	
Beispiel:	LSX.SetMotorBrakeSwitchOffDelay (0,0,0,0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorbrakeswitchoffdelay	ValidConfig

LSX_GetMotorAutoKommDelay			
Beschreibung:	Befehl zum Auslesen der Verzögerung zwischen dem Einschalten der Endstufe und des Startens der Autokommutierung.		
Delphi:	function LSX_GetMotorAutoKommDelay (LSID: Integer; X,Y,Z,A: integer): Integer;		
C++:	Int GetMotorAutoKommDelay (int lLSID, int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Verzögerung in ms, Bereich: 0 bis 5000 ms	
Beispiel:	LSX.GetMotorAutoKommDelay (&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorautokommdelay	-

LSX_SetMotorAutoKommDelay			
Beschreibung:	Befehl zum Einstellen der Verzögerung zwischen dem Einschalten der Endstufe und des Startens der Autokommutierung.		
Delphi:	function LSX_SetMotorAutoKommDelay (LSID: Integer; X,Y,Z,A : Integer): Integer;		
C++:	Int SetMotorAutoKommDelay (int lLSID, int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Verzögerung in ms, Bereich: 0 bis 5000 ms	
Beispiel:	LSX.SetMotorAutoKommDelay (5,5,5,5);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorautokommdelay	ValidConfig

LSX_GetMotorAutoKommSpeedScale			
Beschreibung:	Befehl zum Auslesen der Geschwindigkeit beim Autokommutieren. Dient zum Anpassen des Bewegungsvorgangs beim Autokommutiern an die Motorkraft bzw. das Motordrehmoment.		
Delphi:	function LSX_GetMotorAutoKommSpeedScale (LSID: Integer; X,Y,Z,A: double): integer;		
C++:	Int GetMotorAutoKommSpeedScale (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Skalierungsfaktor, Bereich: 0.01 bis 100	
Beispiel:	LSX.GetMotorAutoKommSpeedScale (&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorautokommspeedscale	-

LSX_SetMotorAutoKommSpeedScale			
Beschreibung:	Befehl zum Einstellen der Geschwindigkeit beim Autokommutieren. Dient zum Anpassen des Bewegungsvorgangs beim Autokommutiern an die Motorkraft bzw. das Motordrehmoment.		
Delphi:	function LSX_SetMotorAutoKommSpeedScale (LSID: Integer; X,Y,Z,A : double): Integer;		
C++:	Int SetMotorAutoKommSpeedScale (int lLSID, double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Skalierungsfaktor, Bereich: 0.01 bis 100	
Beispiel:	LSX.SetMotorAutoKommSpeedScale (10.25, 10.25, 10.25, 10.25);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorautokommspeedscale	ValidConfig

LSX_GetMotorForceConstant			
Beschreibung:	Befehl zum Auslesen der Kraftkonstante eines Linearmotors in [N/A]		
Delphi:	function LSX_GetMotorForceConstant (LSID: Integer; X,Y,Z,A: double): Integer;		
C++:	Int GetMotorForceConstant (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Kraftkonstante, Bereich: 0 bis 500 N/A	
Beispiel:	LSX.GetMotorForceConstant (&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorforceconstant	-

LSX_SetMotorForceConstant			
Beschreibung:	Befehl zum Einstellen der Kraftkonstante eines Linearmotors in [N/A]		
Delphi:	function LSX_SetMotorForceConstant (LSID: Integer; X,Y,Z,A : double): Integer;		
C++:	Int SetMotorForceConstant (int lLSID, double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Kraftkonstante, Bereich: 0 bis 500 N/A	
Beispiel:	LSX.SetMotorForceConstant (2.1, 2.1, 2.1, 2.1);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorforceconstant	ValidPar / ValidConfig

LSX_GetMotorLoad			
Beschreibung:	Befehl zum Auslesen der eingestellten Motorlast in [kg] von Achsen mit Linearmotoren. Parameter dient zur Servoregler-Auslegung, die gesamte zu beschleunigende Masse (Achse, Anbauten, Beladung, …) ist zu übergeben.		
Delphi:	function LSX_GetMotorLoad (LSID: Integer; X,Y,Z,A: double): Integer;		
C++:	Int GetMotorLoad (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte Motorlast, Bereich: 0 bis 100 kg	
Beispiel:	LSX.GetMotorLoad (&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorload	-

LSX_SetMotorLoad			
Beschreibung:	Befehl zum Einstellen der Motorlast in [kg] von Achsen mit Linearmotoren. Parameter dient zur Servoregler-Auslegung, die gesamte zu beschleunigende Masse (Achse, Anbauten, Beladung, …) ist zu übergeben.		
Delphi:	function LSX_SetMotorLoad (LSID: Integer; X,Y,Z,A : double): Integer;		
C++:	Int SetMotorLoad (int lLSID, double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellende Motorlast, Bereich: 0 bis 100 kg	
Beispiel:	LSX.SetMotorLoad (5.1, 2.4, 3.0, 4.2);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorload	ValidPar / ValidConfig

LSX_GetMotorMomentConstant			
Beschreibung:	Befehl zum Auslesen der eingestellten Motorlast in [kg] von Achsen mit Linearmotoren. Parameter dient zur Servoregler-Auslegung, die gesamte zu beschleunigende Masse (Achse, Anbauten, Beladung, …) ist zu übergeben.		
Delphi:	function LSX_GetMotorMomentConstant (LSID: Integer; X,Y,Z,A: double): Integer;		
C++:	Int GetMotorMomentConstant (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte Drehmomentkonstante, Bereich: 0 bis 50 Nm/A	
Beispiel:	LSX.GetMotorMomentConstant (&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motormomentconstant	-

LSX_SetMotorMomentConstant			
Beschreibung:	Befehl zum Einstellen der Motorlast in [kg] von Achsen mit Linearmotoren. Parameter dient zur Servoregler-Auslegung, die gesamte zu beschleunigende Masse (Achse, Anbauten, Beladung, …) ist zu übergeben.		
Delphi:	function LSX_SetMotorMomentConstant (LSID: Integer; X,Y,Z,A : double): Integer;		
C++:	Int SetMotorMomentConstant (int lLSID, double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellende Drehmomentkonstante, Bereich: 0 bis 50 Nm/A	
Beispiel:	LSX.SetMotorMomentConstant (0.75, 0.75, 0.75, 0.75);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motormomentconstant	ValidPar / ValidConfig

LSX_GetMotorMomentOfInertia			
Beschreibung:	Befehl zum Auslesen des eingestellten Massenträgheitsmomentes in [kgcm²] an der Motorwelle. Parameter dient zur Servoregler-Auslegung, die Summe aller Drehmomente (Motorrotorträgheitsmoment, (rückgerechnetes) Trägheitsmoment der angeschlossenen Mechanik, …) ist zu übergeben.		
Delphi:	function LSX_GetMotorMomentOfInertia (LSID: Integer; X, Y, Z, A: double): Integer;		
C++:	Int GetMotorMomentOfInertia (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestelltes Trägheitsmoment, Bereich: 0 bis 100000 kgcm²	
Beispiel:	LSX.GetMotorMomentOfInertia (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motormomentofinertia	-

LSX_SetMotorMomentOfInertia			
Beschreibung:	Befehl zum Einstellen des Massenträgheitsmomentes in [kgcm²] an der Motorwelle. Parameter dient zur Servoregler-Auslegung, die Summe aller Drehmomente (Motorrotorträgheitsmoment, (rückgerechnetes) Trägheitsmoment der angeschlossenen Mechanik, …) ist zu übergeben.		
Delphi:	function LSX_SetMotorMomentOfInertia (LSID: Integer; X,Y,Z,A : double): Integer;		
C++:	Int SetMotorMomentOfInertia (int lLSID, double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellende Trägheitsmoment, Bereich: 0 bis 100000 kgcm²	
Beispiel:	LSX.SetMotorMomentOfInertia (0.25, 0.25, 0.25, 0.25);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motormomentofinertia	ValidPar / ValidConfig

LSX_GetMotorIITCheck			
Beschreibung:	Befehl zum Auslesen des Status der Motor-I²t-Überwachung		
Delphi:	function LSX_GetMotorIITCheck (LSID: Integer; X, Y, Z, A: LongBool): Integer;		
C++:	Int GetMotorIITCheck (int lLSID, bool *pbX, bool *pbY, bool *pbZ, bool *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellender Status: True / False	
Beispiel:	LSX.GetMotorIITCheck (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorisquaretcheck	-

LSX_SetMotorIITCheck			
Beschreibung:	Befehl zum Ein- und Ausschalten der Motor-I²t-Überwachung		
Delphi:	function LSX_SetMotorIITCheck (LSID: Integer; X, Y, Z, A: LongBool): Integer;		
C++:	Int SetMotorIITCheck (int lLSID, bool bX, bool bY, bool bZ, bool bA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellender Status: True / False	
Beispiel:	LSX.SetMotorIITCheck (False, True, False, False);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorisquaretcheck	ValidPar / ValidConfig

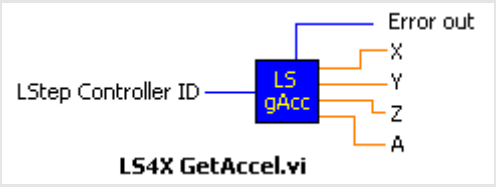
LSX_GetMotorpeakCurrentTime			
Beschreibung:	Befehl zum Auslesen der eingestellten maximalen Motorspitzenstromdauer in [ms]		
Delphi:	function LSX_GetMotorpeakCurrentTime (LSID: Integer; X, Y, Z, A: LongBool): Integer;		
C++:	Int GetMotorpeakCurrentTime (int lLSID, bool *pbX, bool *pbY, bool *pbZ, bool *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte maximale Spitzenstromdauer, Bereich: 0 bis 10000ms	
Beispiel:	LSX.GetMotorpeakCurrentTime (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorpeakcurrenttime	-

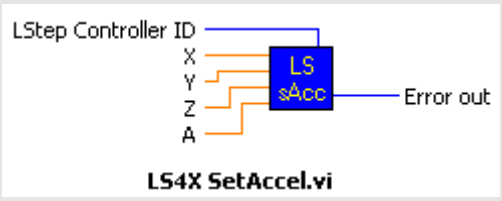
LSX_SetMotorpeakCurrentTime			
Beschreibung:	Befehl zum Einstellen der maximalen Motorspitzenstromdauer in [ms]		
Delphi:	function LSX_SetMotorpeakCurrentTime (LSID: Integer; X, Y, Z, A: LongBool): Integer;		
C++:	Int SetMotorpeakCurrentTime (int lLSID, bool bX, bool bY, bool bZ, bool bA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellende maximale Spitzenstromdauer, Bereich: 0 bis 10000ms	
Beispiel:	LSX.SetMotorpeakCurrentTime (3000, 3000, 2000, 2000);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorpeakcurrenttime	ValidPar / ValidConfig

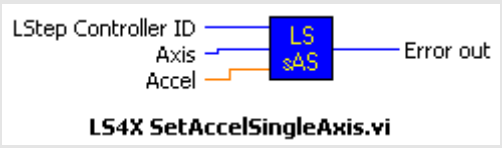
LSX_GetMotorAutoKommCurrent			
Beschreibung:	Befehl zum Auslesen des Motorstromes zum Autokommutieren (nur Servobetrieb), muss kleiner gleich maximalem Gerätestrom (siehe maxcur) und Motornennstrom sein.		
Delphi:	function LSX_GetMotorAutoKommCurrent (LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	Int GetMotorAutoKommCurrent (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Strom zum Autokommutieren	
Beispiel:	LSX.GetMotorAutoKommCurrent (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorautokommcurrent	-

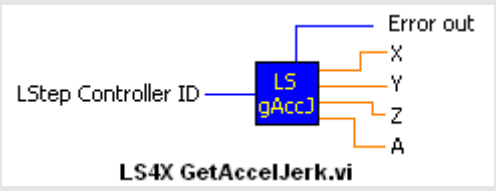
LSX_SetMotorAutoKommCurrent			
Beschreibung:	Befehl zum Einstellen des Motorstromes zum Autokommutieren (nur Servobetrieb), muss kleiner gleich maximalem Gerätestrom (siehe maxcur) und Motornennstrom sein.		
Delphi:	function LSX_SetMotorAutoKommCurrent (LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	Int SetMotorAutoKommCurrent (int lLSID, double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Strom zum Autokommutieren	
Beispiel:	LSX.SetMotorAutoKommCurrent (1.5, 1.5, 1.5, 1.5);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorautokommcurrent	ValidPar / ValidConfig

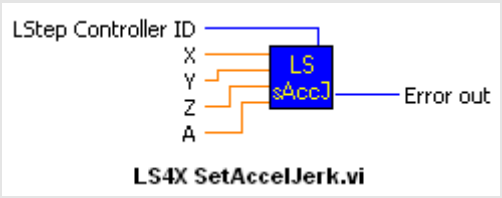
4.2.6 Kinematik

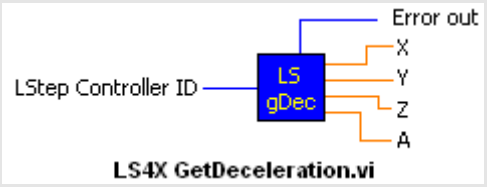
LSX_GetAccel			
Beschreibung:	Funktion zum Abfragen der Beschleunigung für Positioniervorgänge.		
Delphi:	function LSX_GetAccel(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetAccel(double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	X, Y, Z, A	Ausgelesene Beschleunigungswerte LSTEP 2000 Serie = m/s² LSTEP express Serie = Eingestellte Dimension/s²	
Beispiel:	LSX.GetAccel(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?accel	-

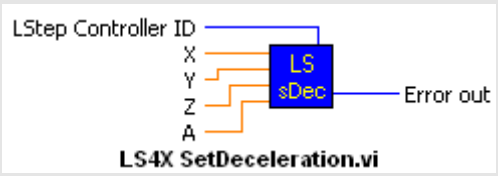
LSX_SetAccel			
Beschreibung:	Funktion zum Setzen der Beschleunigung für Positioniervorgänge.		
Delphi:	function LSX_SetAccel(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetAccel(double dX, double dY, double dZ, double dA);		
LabView:			
Parameter:	X, Y, Z, A	Beschleunigungswerte der Achsen. LSTEP 2000 Serie = m/s², begrenzt auf 20 m/s² LSTEP express Serie = Eingestellte Dimension/s², Maximalwert begrenzt durch Steuerung	
Beispiel:	LSX.SetAccel(1.0, 1.5, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!accel	-

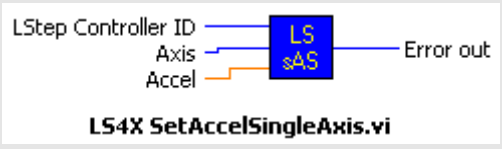
LSX_SetAccelSingleAxis			
Beschreibung:	Funktion zum Einstellen der Beschleunigung einer Achse für Positioniervorgänge.		
Delphi:	function LSX_SetAccelSingleAxis(LSID: Integer; Axis: Integer; Accel: Double): Integer;		
C++:	int SetAccelSingleAxis(int lAxis,double dAccel);		
LabView:			
Parameter:	Axis	Achse deren Beschleunigung gesetzt werden soll X = 1 Y = 2 …	
	Accel	Einzustellende Beschleunigung LSTEP 2000 Serie = m/s², begrenzt auf 20 m/s² LSTEP express Serie = Eingestellte Dimension/s², Maximalwert begrenzt durch Steuerung	
Beispiel:	LSX.SetAccelSingleAxis(4, 1.0); // Beschleunigung A-Achse 1.0 m/s²		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!accel	-

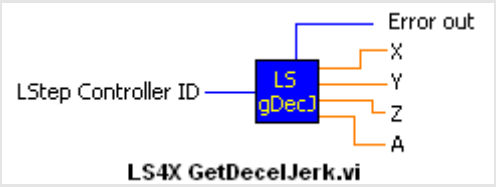
LSX_GetAccelJerk			
Beschreibung:	Funktion zum Abfragen des verwendeten Rucks während der Beschleunigungsphase bei Positioniervorgängen. (nur für LSTEP express)		
Delphi:	function LSX_GetAccelJerk(LSID: Integer; var XD, YD, ZD, AD: Double): Integer;		
C++:	int GetAccelJerk(double *pdXD, double *pdYD, double *pdZD, double *pdAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Ruckwerte in der eingestellten Dimension/s³	
Beispiel:	LSX.GetAccelJerk(&XD, &YD, &ZD, &AD);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?acceljerk	-

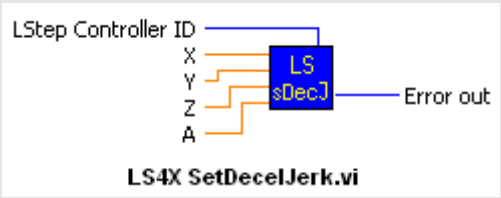
LSX_SetAccelJerk			
Beschreibung:	Funktion zum Setzen des verwendeten Rucks während der Beschleunigungsphase bei Positioniervorgängen. (nur für LSTEP express)		
Delphi:	function LSX_SetAccelJerk(LSID: Integer; XD, YD, ZD, AD: Double): Integer;		
C++:	int SetAccelJerk(double dXD, double dYD, double dZD, double dAD);		
LabView:			
Parameter:	X, Y, Z und A	Ruck in der eingestellten Dimension/s³	
Beispiel:	LSX.SetAccelJerk(100.0, 100.5, 200, 300);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!acceljerk	-

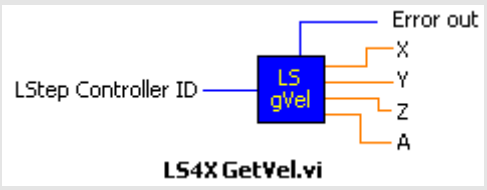
LSX_GetDeceleration			
Beschreibung:	Abfragen der verwendeten Verzögerung für Bewegungen. (nur für LSTEP express)		
Delphi:	function LSX_GetDeceleration(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetDeceleration(double *pdXD, double *pdYD, double *pdZD, double *pdAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Verzögerungswerte in der eingestellten Dimension/s²	
Beispiel:	LSX.GetDeceleration(&XD, &YD, &ZD, &AD);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?decel	-

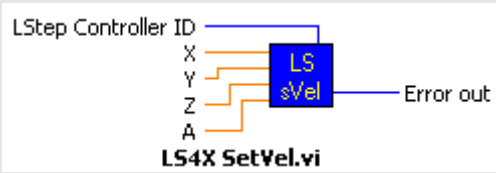
LSX_SetDeceleration			
Beschreibung:	Setzen der Verzögerung die bei Bewegungen verwendet werden soll. (nur für LSTEP express)		
Delphi:	function LSX_SetDeceleration(LSID: Integer; XD, YD, ZD, AD: Double): Integer;		
C++:	int SetDeceleration(double dXD, double dYD, double dZD, double dAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Verzögerungswerte in der eingestellten Dimension/s²	
Beispiel:	LSX.SetDeceleration(1.0, 1.5, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!decel	-

LSX_SetDecelSingleAxis			
Beschreibung:	Setzt die verwendete Verzögerung bei einer Bewegung für eine einzelne Achse. (nur für LSTEP express)		
Delphi:	function LSX_SetDecelSingleAxis(LSID: Integer; Axis: Integer; Decel: Double): Integer;		
C++:	int SetDecelSingleAxis(int lAxis,double dDecel);		
LabView:			
Parameter:	Axis	Achse deren Verzögerung gesetzt werden soll X = 1 Y = 2 …	
	Decel	Verzögerung in der eingestellten Dimension/s²	
Beispiel:	LSX.SetDecelSingleAxis(1, 1000.0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!decel	-

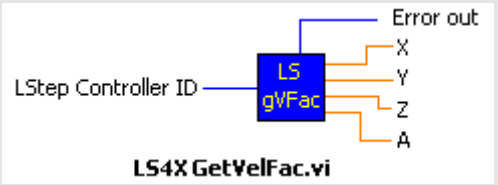
LSX_GetDecelJerk			
Beschreibung:	Abfragen des Rucks, der während der Verzögerungsphase einer Bewegung verwendet werden soll. (nur für LSTEP express)		
Delphi:	function LSX_GetDecelJerk(LSID: Integer; var XD, YD, ZD, AD: Double): Integer;		
C++:	int GetDecelJerk(double *pdXD, double *pdYD, double *pdZD, double *pdAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Ruckwerte in der eingestellten Dimenstion/s³	
Beispiel:	LSX.GetDecelJerk(&XD, &YD, &ZD, &AD);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?deceljerk	-

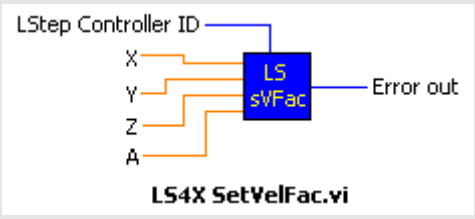
LSX_SetDecelJerk			
Beschreibung:	Setzen des Rucks, der während der Verzögerungsphase einer Bewegung verwendet werden soll. (nur für LSTEP express)		
Delphi:	function LSX_SetDecelJerk(LSID: Integer; XD, YD, ZD, AD: Double): Integer;		
C++:	int SetDecellJerk(double dXD, double dYD, double dZD, double dAD);		
LabView:			
Parameter:	X, Y, Z, A	Ruck in der eingestellten Dimension/s³	
Beispiel:	LSX.SetDecelJerk(1.0, 1.5, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!deceljerk	-

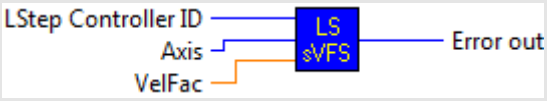
LSX_GetVel			
Beschreibung:	Abfragen der eingestellten Geschwindigkeit die für Positioniervorgänge verwendet wird.		
Delphi:	function LSX_GetVel(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetVel(double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	X, Y, Z, A	Ausgelesene Geschwindigkeitswerte LSTEP 2000 Serie = m/s LSTEP express Serie = Eingestellte Dimension/s	
Beispiel:	LSX.GetVel(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?vel	-

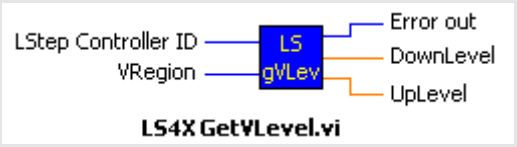
LSX_SetVel			
Beschreibung:	Einstellen der Geschwindigkeit die bei Positioniervorgängen verwendet wird.		
Delphi:	function LSX_SetVel(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetVel(double dX, double dY, double dZ, double dA);		
LabView:			
Parameter:	X, Y, Z, A:	Einzustellende Geschwindigkeitswerte LSTEP 2000 Serie = m/s LSTEP express Serie = Eingestellte Dimension/s	
Beispiel:	LSX.SetVel(1.0, 15.0, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!vel	-

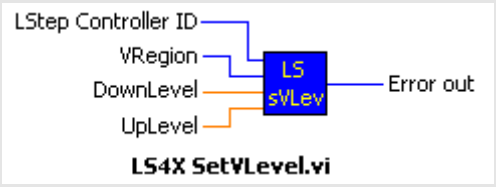
LSX_SetVelSingleAxis			
Beschreibung:	Funktion zum Einstellen der Geschwindigkeit für eine einzelne Achse.		
Delphi:	function LSX_SetVelSingleAxis(LSID: Integer; Axis: Integer; Vel: Double): Integer;		
C++:	int SetVelSingleAxis(int lAxis,double dVel);		
LabView:			
Parameter:	Axis	Achse deren Geschwindigkeit gesetzt werden soll X = 1 Y = 2 …	
	Vel	Einzustellender Geschwindigkeitswert LSTEP 2000 Serie = m/s LSTEP express Serie = Eingestellte Dimension/s	
Beispiel:	LSX.SetVelSingleAxis(1, 10.0) // Geschwindigkeit X-Achse 10 U/s		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!vel	-

LSX_GetVelFac			
Beschreibung:	Funktion zum Abfragen der Geschwindigkeitsuntersetzung. (nicht für LSTEP express)		
Delphi:	function LSX_GetVelFac(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetVelFac(double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	X, Y, Z, A:	Eingestellter Faktor für die Geschwindigkeitsuntersetzung.	
Beispiel:	LSX.GetVelFac(X, Y, Z, A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?velfac	-

LSX_SetVelFac			
Beschreibung:	Funktion zum Setzen der Geschwindigkeitsuntersetzung. (nicht für LSTEP express)		
Delphi:	function LSX_SetVelFac(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetVelFac(double dX, double dY, double dZ, double dA);		
LabView:			
Parameter:	X, Y, Z, A:	Einzustellender Faktor für die Geschwindigkeitsuntersetzung. Die resultierende Geschwindigkeit ist v=Vel*VelFac.	
Beispiel:	LSX.SetVelFac(1, 1, 0.1, 0.1);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!velfac	-

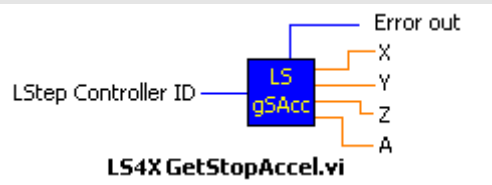
LSX_SetVelFacSingleAxis			
Beschreibung:	Funktion zum Setzen der Geschwindigkeitsuntersetzung einer Achse. (nicht für LSTEP express)		
Delphi:	function LSX_SetVelFacSingleAxis(LSID: Integer; Axis: Integer; Value: Double): Integer;		
C++:	int SetVelFacSingleAxis(int lAxis, double dValue);		
LabView:			
Parameter:	Axis	Achse deren Geschwindigkeitsuntersetzung gesetzt werden soll X = 1 Y = 2 …	
	VelFac	Einzustellender Faktor für die Geschwindigkeitsuntersetzung Die resultierende Geschwindigkeit ist v=Vel*VelFac.	
Beispiel:	LSX.SetVelFacSingleAxis(1, 0.1)		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!velfac	-

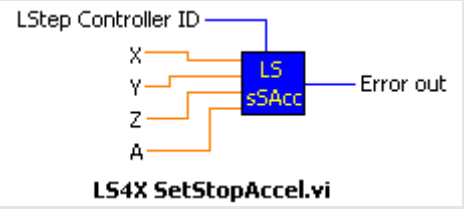
LSX_GetVLevel			
Beschreibung:	Liefert die Geschwindigkeitsgrenzen von dem angegebenen Geschwindigkeitsbereich. (nicht für LSTEP express)		
Delphi:	function LSX_GetVLevel(LSID: Integer; lVRegion: Integer; var dDownLevel, dUpLevel: Double): Integer;		
C++:	int GetVLevel(int lVRegion, double *pdDownLevel, double *pdUpLevel);		
LabView:			
Parameter:	lVRegion	Wertebereich 1-4. 1 – Erster/Unterer Geschwindigkeitsbereich 2 – Zweiter/Mittlerer Geschwindigkeitsbereich 3 – Dritter/Oberer Geschwindigkeitsbereich 4 – Bis zu dieser Geschwindigkeitsgrenze wird die Korrekturtabelle genutzt.	
	dDownLevel	Untere Grenze des Bereichs (bei lVRegion = 4 Geschwindigkeitsgrenze) in U/s	
	dUpLevel	Obere Grenze des Bereichs (bei lVRegion = 4 hat keine Bedeutung) in U/s	
Beispiel:	LSX.GetVLevel(2, &DownLevel, &UpLevel); // DownLevel = Untere Grenze des zweiten Geschwindigkeitsbereich, UpLevel = Obere Grenze des zweiten Geschwindigkeitsbereich.		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?vlevel	-

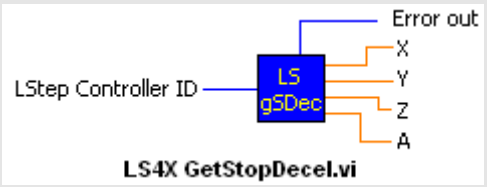
LSX_SetVLevel			
Beschreibung:	Geschwindigkeitsbereich ausklammern, in denen das System zu Resonanzen neigt. (nicht für LSTEP express)		
Delphi:	function LSX_SetVLevel(LSID: Integer; lVRegion: Integer; dDownLevel, dUppLevel: Double): Integer;		
C++:	int SetVLevel(int lVRegion, double dDownLevel, double dUppLevel);		
LabView:			
Parameter:	lVRegion	Wertebereich 1-4. 1 – Erster/Unterer Geschwindigkeitsbereich 2 – Zweiter/Mittlerer Geschwindigkeitsbereich 3 – Dritter/Oberer Geschwindigkeitsbereich 4 – Bis zu dieser Geschwindigkeitsgrenze wird die Korrekturtabelle genutzt.	
	dDownLevel	Untere Grenze des Bereichs (bei lVRegion = 4 Geschwindigkeitsgrenze) in U/s	
	dUppLevel	Obere Grenze des Bereichs (bei lVRegion = 4 hat keine Bedeutung) in U/s	
Beispiel:	LSX.SetVLevel(4, 10.0, 0.0); //Die Korrekturtabelle wirkt bis zu einer Geschwindigkeit von 10 Umdrehungen/s.		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!vlevel	-

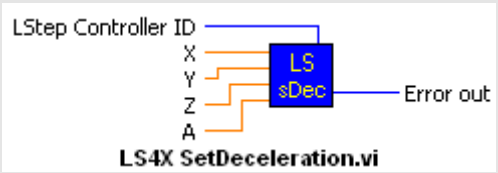
LSX_GetSpeedPoti			
Beschreibung:	Fragt ab, ob das Potentiometer ein- oder ausgeschaltet ist.		
Delphi:	function LSX_GetSpeedPoti(LSID: Integer; var SpePoti: LongBool): Integer;		
C++:	int GetSpeedPoti(BOOL *pbSpePoti);		
LabView:			
Parameter:	SpePoti	Status des Potentiometers True = Alle Endstufen sind eingeschaltet False = Alle Endstufen sind ausgeschaltet	
Beispiel:	LSX.GetSpeedPoti(&flag);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?pot	-

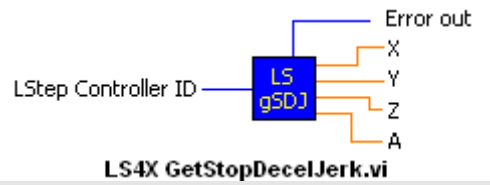
LSX_SetSpeedPoti			
Beschreibung:	Ein- bzw. Ausschalten des Potentiometers. Ist dieses eingeschaltet wird die Verfahrgeschwindigkeit in Abhängigkeit der Stellung des Potentiometers prozentual genutzt. Ist es aus wird die zuvor festgelegte Geschwindigkeit verwendet. Im manuellen Joystickbetrieb ist die Auswertung des Potentiometers immer aktiv.		
Delphi:	function LSX_SetSpeedPoti(LSID: Integer; SpeedPoti: LongBool): Integer;		
C++:	int SetSpeedPoti(BOOL SpeedPoti);		
LabView:			
Parameter:	SpePoti	Status des Potentiometers True = Alle Endstufen sind eingeschaltet False = Alle Endstufen sind ausgeschaltet	
Beispiel:	LSX.SetSpeedPoti(true);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!pot	-

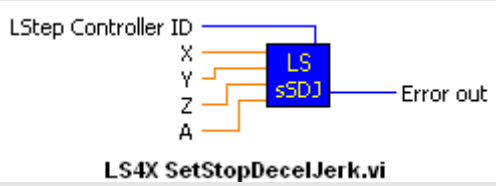
LSX_GetStopAccel			
Beschreibung:	Liefert die Bremsbeschleunigung, wenn der Stopp-Eingang aktiv ist. (nicht für LSTEP express)		
Delphi:	function LSX_GetStopAccel(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetStopAccel(double *pdXD, double *pdYD, double *pdZD, double *pdAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Verzögerung in m/s²	
Beispiel:	LSX.GetStopAccel(&XD, &YD, &ZD, &AD);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?stopaccel	-

LSX_SetStopAccel			
Beschreibung:	Funktion zum Einstellen der Bremsbeschleunigung, wenn der Stopp-Eingang aktiv ist. Dieser Wert gilt nur für den Vektorbetrieb, nicht für: Joystick, Kalibrieren und Hubmessen. Weiterhin wird er nicht in Steuerungen der LSTEP 2000 Familie mit einem Save gespeichert. (nicht für LSTEP express)		
Delphi:	function LSX_SetStopAccel(LSID: Integer; XD, YD, ZD, AD: Double): Integer;		
C++:	int SetStopAccel(double dXD, double dYD, double dZD, double dAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Verzögerung in m/s²	
Beispiel:	LSX.SetStopAccel(1.0, 1.5, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!stopaccel	-

LSX_GetStopDecel			
Beschreibung:	Abfragen der Bremsbeschleunigung bei aktivem Stopp-Eingang. (nur für LSTEP express)		
Delphi:	function LSX_GetStopDecel(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetStopDecel(double *pdXD, double *pdYD, double *pdZD, double *pdAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Verzögerungswerte in der eingestellten Dimension/s²	
Beispiel:	LSX.GetStopDecel(&XD, &YD, &ZD, &AD);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?stopdecel	-

LSX_SetStopDecel			
Beschreibung:	Funktion zum Einstellen der Bremsbeschleunigung mit der die Achsen im Falle eines Stoppsignals bremsen sollen. Damit dieser Wert aktiv verwendet wird, muss er größer sein als der mit LSX_SetDecel gesetzte Wert. Ist er kleiner wird ein Stopp mit der Bremsbeschleunigung von LSX_SetDecel ausgeführt. (nur für LSTEP express)		
Delphi:	function LSX_SetStopDecel(LSID: Integer; XD, YD, ZD, AD: Double): Integer;		
C++:	int SetStopDecel(double dXD, double dYD, double dZD, double dAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Verzögerungswerte in der eingestellten Dimension/s²	
Beispiel:	LSX.SetStopDecel(1.0, 1.5, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!stopdecel	-

LSX_GetStopDecelJerk			
Beschreibung:	Funktion zum Abfragen des Rucks, der verwendet wird um die Verzögerung im Falle eines Notstoppsignals auf- bzw. abzubauen. (nur für LSTEP express)		
Delphi:	function LSX_GetStopDecelJerk(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetStopDecelJerk(double *pdXD, double *pdYD, double *pdZD, double *pdAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Ruckwerte in der eingestellten Dimension/s³	
Beispiel:	LSX.GetStopDecelJerk(&XD, &YD, &ZD, &AD);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?stopdeceljerk	-

LSX_SetStopDecelJerk			
Beschreibung:	Funktion zum Setzen des Rucks, der verwendet wird um die Verzögerung im falle eines Notstoppsignals auf bzw. abzubauen. Damit dieser Wert aktiv verwendet wird, muss er größer sein als der mit LSX_SetDecelJerk gesetzte Wert. Ist er kleiner wird ein Stopp mit dem Bremsruck von LSX_SetDecelJerk ausgeführt. (nur für LSTEP express)		
Delphi:	function LSX_SetStopDecelJerk(LSID: Integer; XD, YD, ZD, AD: Double): Integer;		
C++:	int SetStopDecelJerk(double dXD, double dYD, double dZD, double dAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Ruckwerte in der eingestellten Dimension/s³	
Beispiel:	LSX.SetStopDecelJerk(1000, 1500, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!stopdeceljerk	-

LSX_GetRoomAccelJerk			
Beschreibung:	Befehl zum Auslesen des Rucks während der Beschleunigung im Raum: Bei schwingungskritischer Maschinenmechanik ist eine Ruckbegrenzung zu empfehlen. Die Werte sollten nur so klein wie erforderlich zur Schwingungsunterdrückung und so groß wie möglich eingestellt werden, da diese Parameter erheblichen Einfluss auf die Positionierzeiten haben. Eingabewerte sind abhängig von der Dimension.		
Delphi:	function LSX_GetRoomAccelJerk (LSID: Integer; var value: Double): Integer;		
C++:	int GetRoomAccelJerk (double *pdvalue);		
LabView:	Nicht unterstützt		
Parameter:	value	Ruck während Beschleunigung	
Beispiel:	LSX.GetRoomAccelJerk (&value);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?roomacceljerk	-

LSX_SetRoomAccelJerk			
Beschreibung:	Befehl zum Einstellen des Rucks während der Beschleunigung im Raum: Bei schwingungskritischer Maschinenmechanik ist eine Ruckbegrenzung zu empfehlen. Die Werte sollten nur so klein wie erforderlich zur Schwingungsunterdrückung und so groß wie möglich eingestellt werden, da diese Parameter erheblichen Einfluss auf die Positionierzeiten haben. Eingabewerte sind abhängig von der Dimension.		
Delphi:	function LSX_SetRoomAccelJerk (LSID: Integer; value: Double): Integer;		
C++:	int SetRoomAccelJerk (double dvalue);		
LabView:	Nicht unterstützt		
Parameter:	value	Ruck während Beschleunigung	
Beispiel:	LSX.SetRoomAccelJerk (4500);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!roomacceljerk	Sofort

LSX_GetRoomDecelJerk			
Beschreibung:	Befehl zum Auslesen des Rucks während der Verzögerung im Raum: Bei schwingungskritischer Maschinenmechanik ist eine Ruckbegrenzung zu empfehlen. Die Werte sollten nur so klein wie erforderlich zur Schwingungsunterdrückung und so groß wie möglich eingestellt werden, da diese Parameter erheblichen Einfluss auf die Positionierzeiten haben. Eingabewerte sind abhängig von der Dimension.		
Delphi:	function LSX_GetRoomDecelJerk (LSID: Integer; var value: Double): Integer;		
C++:	int GetRoomDecelJerk (double *pdvalue);		
LabView:	Nicht unterstützt		
Parameter:	value	Ruck während Beschleunigung	
Beispiel:	LSX.GetRoomDecelJerk (&value);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?roomdeceljerk	-

LSX_SetRoomDecelJerk			
Beschreibung:	Befehl zum Einstellen des Rucks während der Beschleunigung im Raum: Bei schwingungskritischer Maschinenmechanik ist eine Ruckbegrenzung zu empfehlen. Die Werte sollten nur so klein wie erforderlich zur Schwingungsunterdrückung und so groß wie möglich eingestellt werden, da diese Parameter erheblichen Einfluss auf die Positionierzeiten haben. Eingabewerte sind abhängig von der Dimension.		
Delphi:	function LSX_SetDecelJerk (LSID: Integer; value: Double): Integer;		
C++:	int SetRoomDecelJerk (double dvalue);		
LabView:	Nicht unterstützt		
Parameter:	value	Ruck während Beschleunigung	
Beispiel:	LSX.SetRoomDecelJerk (4800);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!roomdeceljerk	Sofort

LSX_GetRoomStopJerk			
Beschreibung:	Befehl zum Auslesen des Rucks während des Bremsvorgangs nach Auftreten eines Notstopp-Ereignisses: Bei schwingungskritischer Maschinenmechanik ist eine Ruckbegrenzung zu empfehlen. Die Werte sollten so klein wie erforderlich zur Schwingungsunterdrückung und so groß wie möglich eingestellt werden, da dieser Parameter erheblichen Einfluss auf die Dauer des Bremsvorgangs haben kann. Eingabewerte sind abhängig von der Dimension. Unterschreitet "!stopdeceljerk" "!deceljerk" wird Rampe mit "!deceljerk" berechnet.		
Delphi:	function LSX_GetRoomStopJerk (LSID: Integer; var value: Double): Integer;		
C++:	int GetRoomStopJerk (double *pdvalue);		
LabView:	Nicht unterstützt		
Parameter:	value	Ruck während Verzögerung nach Auftreten eines Notstop-Ereignisses	
Beispiel:	LSX.GetRoomStopJerk (&value);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?roomstopjerk	-

LSX_SetRoomStopJerk			
Beschreibung:	Befehl zum Einstellen des Rucks während des Bremsvorgangs nach Auftreten eines Notstopp-Ereignisses: Bei schwingungskritischer Maschinenmechanik ist eine Ruckbegrenzung zu empfehlen. Die Werte sollten so klein wie erforderlich zur Schwingungsunterdrückung und so groß wie möglich eingestellt werden, da dieser Parameter erheblichen Einfluss auf die Dauer des Bremsvorgangs haben kann. Eingabewerte sind abhängig von der Dimension. Unterschreitet "!stopdeceljerk" "!deceljerk" wird Rampe mit "!deceljerk" berechnet.		
Delphi:	function LSX_SetStopJerk (LSID: Integer; value: Double): Integer;		
C++:	int SetRoomStopJerk (double dvalue);		
LabView:	Nicht unterstützt		
Parameter:	value	Ruck während Verzögerung nach Auftreten eines Notstop-Ereignisses	
Beispiel:	LSX.SetRoomStopJerk (5800);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!roomstopjerk	Sofort

LSX_GetRoomAccel			
Beschreibung:	Befehle zum Auslesen der Beschleunigung im Raum		
Delphi:	function LSX_GetRoomAccel (LSID: Integer; var value: Double): Integer;		
C++:	int GetRoomAccel (double *pdvalue);		
LabView:	Nicht unterstützt		
Parameter:	value	Beschleunigung im Raum	
Beispiel:	LSX.GetRoomAccel (&value);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?roomaccel	-

LSX_SetRoomAccel			
Beschreibung:	Befehle zum Einstellen der Beschleunigung im Raum		
Delphi:	function LSX_SetRoomAccel (LSID: Integer; value: Double): Integer;		
C++:	int SetRoomAccel (double dvalue);		
LabView:	Nicht unterstützt		
Parameter:	Value	Beschleunigung im Raum	
Beispiel:	LSX.SetRoomAccel (9500);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!roomaccel	Sofort

LSX_GetRoomDecel			
Beschreibung:	Befehle zum Auslesen der Verzögerung im Raum		
Delphi:	function LSX_GetRoomDecel (LSID: Integer; var value: Double): Integer;		
C++:	int GetRoomDecel (double *pdvalue);		
LabView:	Nicht unterstützt		
Parameter:	Value	Verzögerung im Raum	
Beispiel:	LSX.GetRoomDecel (&value);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?roomdecel	-

LSX_SetRoomDecel			
Beschreibung:	Befehle zum Einstellen der Verzögerung im Raum		
Delphi:	function LSX_SetRoomDecel (LSID: Integer; value: Double): Integer;		
C++:	int SetRoomDecel (double dvalue);		
LabView:	Nicht unterstützt		
Parameter:	Value	Verzögerung im Raum	
Beispiel:	LSX.SetRoomDecel (2900);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!roomdecel	Sofort

LSX_GetRoomStopDecel			
Beschreibung:	Befehl zum Auslesen der Verzögerung im Raum nach Auftreten eines Notstopp-Ereignisses: Unterschreiten !stopdecel bzw. !roomstopdecel die Werte von !decel bzw. roomdecel wird Bremsrampe mit deren Werten berechnet.		
Delphi:	function LSX_GetRoomStopDecel (LSID: Integer; var value: Double): Integer;		
C++:	int GetRoomStopDecel (double *pdvalue);		
LabView:	Nicht unterstützt		
Parameter:	Value	Verzögerung im Raum	
Beispiel:	LSX.GetRoomStopDecel (&value);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?roomstopdecel	-

LSX_SetRoomStopDecel			
Beschreibung:	Befehl zum Einstellen der Verzögerung im Raum nach Auftreten eines Notstopp-Ereignisses: Unterschreiten !stopdecel bzw. !roomstopdecel die Werte von !decel bzw. roomdecel wird Bremsrampe mit deren Werten berechnet.		
Delphi:	function LSX_SetRoomStopDecel (LSID: Integer; value: Double): Integer;		
C++:	int SetRoomStopDecel (double dvalue);		
LabView:	Nicht unterstützt		
Parameter:	Value	Verzögerung im Raum	
Beispiel:	LSX.SetRoomStopDecel (2900);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!roomstopdecel	Sofort

LSX_GetRoomVel			
Beschreibung:	Befehl zum Auslesen der Verzögerung im Raum nach Auftreten eines Notstopp-Ereignisses: Unterschreiten !stopdecel bzw. !roomstopdecel die Werte von !decel bzw. roomdecel wird Bremsrampe mit deren Werten berechnet.		
Delphi:	function LSX_GetRoomVel (LSID: Integer; var value: Double): Integer;		
C++:	int GetRoomVel (double *pdvalue);		
LabView:	Nicht unterstützt		
Parameter:	Value	Verzögerung im Raum	
Beispiel:	LSX.GetRoomVel (&value);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?roomvel	-

LSX_SetRoomVel			
Beschreibung:	Befehl zum Einstellen der Verzögerung im Raum nach Auftreten eines Notstopp-Ereignisses: Unterschreiten !stopdecel bzw. !roomstopdecel die Werte von !decel bzw. roomdecel wird Bremsrampe mit deren Werten berechnet.		
Delphi:	function LSX_SetRoomVel (LSID: Integer; value: Double): Integer;		
C++:	int SetRoomVel (double dvalue);		
LabView:	Nicht unterstützt		
Parameter:	Value	Verzögerung im Raum	
Beispiel:	LSX.SetRoomVel (3200);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!roomvel	Sofort

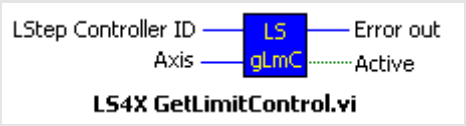
LSX_GetSpeedEnable			
Beschreibung:	Befehl zum Auslesen der Freigabe der Geschwindigkeits- bzw. Drehzahlvorwahl.		
Delphi:	function LSX_GetSpeedEnable (LSID: Integer; var X, Y, Z, A: LongBool): Integer;		
C++:	int GetSpeedEnable (BOOL *plX, BOOL *plY, BOOL *plZ, BOOL *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Status der Achsen	
Beispiel:	LSX.GetSpeedEnable (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?senable	-

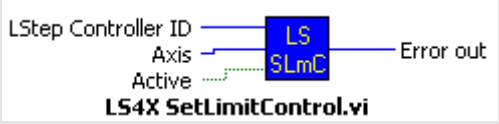
LSX_SetSpeedEnable			
Beschreibung:	Befehl zum Einstellen der Freigabe der Geschwindigkeits- bzw. Drehzahlvorwahl.		
Delphi:	function LSX_SetSpeedEnable (LSID: Integer; X, Y, Z, A: LongBool): Integer;		
C++:	int SetSpeedEnable (BOOL lX, BOOL lY, BOOL lZ, BOOL lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Status der Achsen	
Beispiel:	LSX.SetSpeedEnable (1, 1, 1, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!senable	Sofort

LSX_GetSpeed			
Beschreibung:	Befehl zum Auslesen der Soll-Geschwindigkeit bzw. Drehzahl. Wert wird ohne Rampe aktiv. Der Wert wird als Hexadezimalzahl ausgelesen. (20 Bit signed)		
Delphi:	function LSX_GetSpeed (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetSpeed (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Geschwindigkeit/Drehzahl	
Beispiel:	LSX.GetSpeed (&value);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?s	-

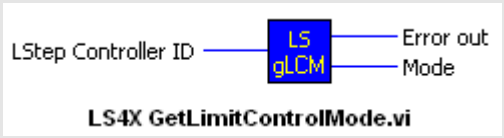
LSX_SetSpeed			
Beschreibung:	Befehl zum Einstellen der Soll-Geschwindigkeit bzw. Drehzahl. Wert wird ohne Rampe aktiv. Der Wert kann entweder als Dezimalzahl oder Hexadezimalzahl übergeben werden. (20 Bit signed)		
Delphi:	function LSX_SetSpeed (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetSpeed (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Geschwindigkeit/Drehzahl	
Beispiel:	LSX.SetSpeed (300, $12C, -300, $FFED4);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!s	SpeedEnable

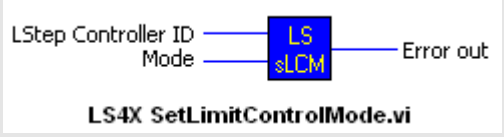
4.2.7 Endschalter und Softwarelimits

LSX_GetLimitControl			
Beschreibung:	Diese Funktion liest aus der Steuerung aus, ob die Bereichsüberwachung ein- oder ausgeschaltet ist.		
Delphi:	function LSX_GetLimitControl(LSID: Integer; Axis: Integer; var Active: LongBool): Integer;		
C++:	int GetLimitControl(int lAxis, BOOL *pbActive);		
LabView:			
Parameter:	Axis	Achse, von der die Bereichsüberwachung gelesen werden sollen 1 = X-Achse 2 = Y-Achse …	
	Active	Eingestellter Wert der Bereichsüberwachung True = Bereichsüberwachung ist aktiv False = Bereichsüberwachung ist nicht aktiv	
Beispiel:	LSX.GetLimitControl(2, &Active); // Activ = False heißt: Bereichsüberwachung von Achse y ist deaktiviert.		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?limctr	-

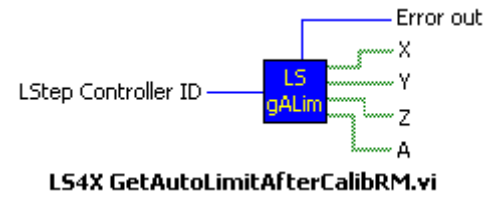
LSX_SetLimitControl	
Beschreibung:	Diese Funktion schaltet die Bereichsüberwachung der Steuerung ein bzw. aus.
Delphi:	function LSX_SetLimitControl(LSID: Integer; Axis: Integer; Active: LongBool): Integer;
C++:	int SetLimitControl(int lAxis,BOOL Active);
LabView:	

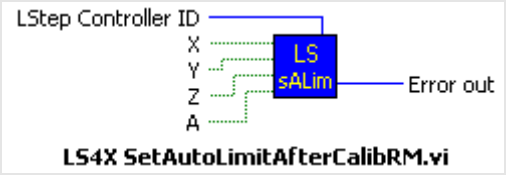
Parameter:	Axis	Achse, bei der die Bereichsüberwachung geschrieben werden soll 1 = X-Achse 2 = Y-Achse …	
	Active	Einzustellender Wert der Bereichsüberwachung True = Bereichsüberwachung ist aktiv False = Bereichsüberwachung ist nicht aktiv	
Beispiel:	LSX.SetLimitControl(2, true); // Bereichsüberwachung von Achse y aktiv		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!limctr	-

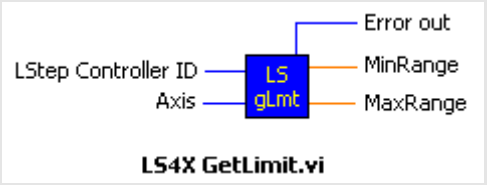
LSX_GetLimitControlMode			
Beschreibung:	Liefert den Modus für die Überwachung der Softwarelimits. (nicht für LSTEP express)		
Delphi:	function LSX_GetLimitControlMode (LSID: Integer; var Mode: Integer): Integer;		
C++:	int GetLimitControlMode(int *plMode);		
LabView:			
Parameter:	Mode	Eingestellter Modus 0 = Moves, die außerhalb des Verfahrbereiches liegen, werden nur bis zu den Grenzen des Verfahrbereiches ausgeführt. 1 = Moves, die außerhalb des Verfahrbereiches liegen, werden nicht ausgeführt.	
Beispiel:	LSX.GetLimitControlMode(&lMode);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?limmode	-

LSX_SetLimitControlMode			
Beschreibung:	Setzt den Modus für die Überwachung der Softwarelimits. (nicht für LSTEP express)		
Delphi:	function LSX_SetLimitControlMode (LSID: Integer; Mode: Integer): Integer;		
C++:	int SetLimitControlMode (int lMode);		
LabView:			

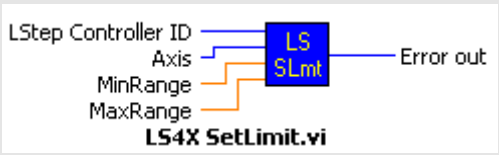
Parameter:	Mode	Einzustellender Modus 0 = Moves, die außerhalb des Verfahrbereiches liegen, werden nur bis zu den Grenzen des Verfahrbereiches ausgeführt. 1 = Moves, die außerhalb des Verfahrbereiches liegen, werden nicht ausgeführt.	
Beispiel:	LSX.SetLimitControlMode(1);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!limmode	-

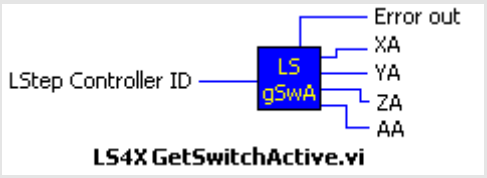
LSX_GetAutoLimitAfterCalibRM			
Beschreibung:	Gibt an, ob beim Kalibrieren und Tischhubmessen die internen Software-Limits gesetzt werden.		
Delphi:	function LSX_GetAutoLimitAfterCalibRM(LSID: Integer; var lFlags: Integer): Integer;		
C++:	int GetAutoLimitAfterCalibRM(int *plFlags);		
LabView:			
Parameter:	lFlags	32-Bit-Integer, welcher nach Aufruf der Funktion in den Bits 0-4 eine Bit-Maske enthält. Bit 0 = X-Achse Bit 1 = Y-Achse … Wert 0 = Automatische Limits werden verwendet Wert 1 = Es werden keine automatischen Limits gesetzt	
Beispiel:	LSX.SetAutoLimitAfterCalibRM(&lFlags);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?nosetlimit	-

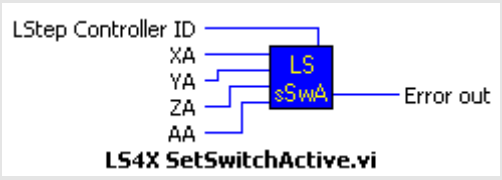
LSX_SetAutoLimitAfterCalibRM			
Beschreibung:	Verhindert, dass beim Kalibrieren und Tischhubmessen die internen Software-Limits gesetzt werden.		
Delphi:	function LSX_SetAutoLimitAfterCalibRM(LSID: Integer; lFlags: Integer): Integer;		
C++:	int SetAutoLimitAfterCalibRM(int lFlags);		
LabView:			
Parameter:	lFlags	32-Bit-Integer, welcher nach Aufruf der Funktion in den Bits 0-4 eine Bit-Maske enthält. Bit 0 = X-Achse Bit 1 = Y-Achse … Wert 0 = Automatische Limits werden verwendet Wert 1 = Es werden keine automatischen Limits gesetzt	
Beispiel:	LSX.SetAutoLimitAfterCalibRM(lFlags);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!nosetlimit	-

LSX_GetLimit			
Beschreibung:	Liest die Verfahrbereichsgrenzen der Achsen aus. Bei Steuerungen der LSTEP express Serie wird bei ungültigen Bereichsgrenzen der Fehlercode 4032 zurückgegeben.		
Delphi:	function LSX_GetLimit(LSID: Integer; Axis: Integer; var MinRange, MaxRange: Double): Integer;		
C++:	int GetLimit(int lAxis, double *pdMinRange, double *pdMaxRange);		
LabView:			

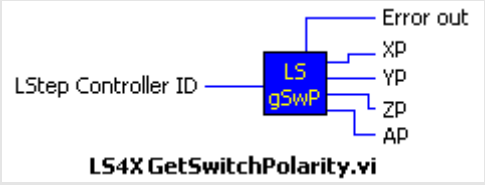
Parameter:	Axis	Achse, von der die Verfahrbereichsgrenzen gelesen werden sollen 1 = X-Achse 2 = Y-Achse …	
	MinRange	Untere Verfahrbereichsgrenze in der eingestellten Dimension der Achse	
	MaxRange	Obere Verfahrbereichsgrenze in der eingestellten Dimension der Achse	
Beispiel:	LSX.GetLimit(1, &MinRange, &MaxRange);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?lim	-

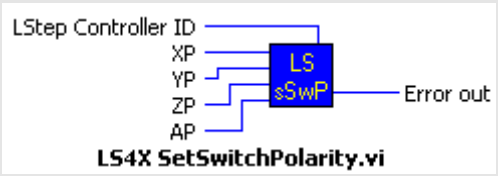
LSX_SetLimit			
Beschreibung:	Setzt die Verfahrbereichsgrenzen einer Achse.		
Delphi:	function LSX_SetLimit(LSID: Integer; Axis: Integer; MinRange, MaxRange: Double): Integer;		
C++:	int SetLimit(int lAxis,double dMinRange,double dMaxRange);		
LabView:			
Parameter:	Axis	Achse, von der die Verfahrbereichsgrenzen gelesen werden sollen 1 = X-Achse 2 = Y-Achse …	
	MinRange	Untere Verfahrbereichsgrenze in der eingestellten Dimension der Achse	
	MaxRange	Obere Verfahrbereichsgrenze in der eingestellten Dimension der Achse	
Beispiel:	LSX.SetLimit(1, -10.0, 20.0); //Achse X –10 als untere und 20 als obere Verfahrbereichsgrenze zuweisen		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!lim	-

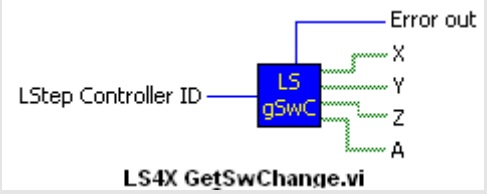
LSX_GetSwitchActive			
Beschreibung:	Diese Funktion liest ein, welche Endschalter zur Überwachung konfiguriert wurden.		
Delphi:	function LSX_GetSwitchActive(LSID: Integer; var XA, YA, ZA, AA: Integer): Integer;		
C++:	int GetSwitchActive(int *plXA, int *plYA, int *plZA, int *plAA);		
LabView:			
Parameter:	XA, YA, ZA, AA	Bitmaske über die Konfiguration der Endschalter Bit 0 = Konfiguration Null-Endschalter Bit 1 = Konfiguration Referenz-Endschalter (bei LSTEP express immer 0) Bit 2 = Konfiguration End-Endschalter	
Beispiel:	LSX.GetSwitchActive(&XA, &YA, &ZA, &AA);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?swact	-

LSX_SetSwitchActive		
Beschreibung:	Aktiviert die Endschalterüberwachung.	
Delphi:	function LSX_SetSwitchActive(LSID: Integer; XA, YA, ZA, AA: Integer): Integer;	
C++:	int SetSwitchActive(int lXA,int lYA,int lZA,int lAA);	
LabView:		

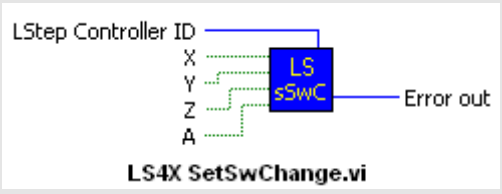
Parameter:	XA, YA, ZA, AA	Bitmaske über die Konfiguration der Endschalter Bit 0 = Konfiguration Null-Endschalter Bit 1 = Konfiguration Referenz-Endschalter (bei LSTEP express immer 0) Bit 2 = Konfiguration End-Endschalter	
Beispiel:	LSX.SetSwitchActive(7, 1, 5, 0); // Alle Endschalter der X-Achse Ein; Null-Endschalter der Y-Achse Ein; E0 und EE der Z-Achse ein; A-Achse: Alle Endschalter Aus		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!swact	-

LSX_GetSwitchPolarity			
Beschreibung:	Funktion zum Einlesen der eingestellten Endschalterpolarität.		
Delphi:	function LSX_GetSwitchPolarity(LSID: Integer; var XP, YP, ZP, AP: Integer): Integer;		
C++:	int GetSwitchPolarity(int *plXP, int *plYP, int *plZP, int *plAP);		
LabView:			
Parameter:	XP, YP, ZP, AP	Bitmaske über die konfigurierte Polarität der Endschalter Bit 0 = Polarität Null-Endschalter Bit 1 = Polarität Referenz-Endschalter (bei LSTEP express immer 0) Bit 2 = Polarität End-Endschalter Wert 0 = Reagiert auf negative Flanke des Endschalters Wert 1 = Reagiert auf positive Flanke des Endschalters	
Beispiel:	LSX.GetSwitchPolarity(&XP, &YP, &ZP, &AP);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?swpol	-

LSX_SetSwitchPolarity			
Beschreibung:	Funktion zum Einstellen der eingestellten Endschalterpolarität.		
Delphi:	function LSX_SetSwitchPolarity(LSID: Integer; XP, YP, ZP, AP: Integer): Integer;		
C++:	int SetSwitchPolarity(int lXP,int lYP,int lZP,int lAA);		
LabView:			
Parameter:	XP, YP, ZP, AP	Bitmaske über die konfigurierte Polarität der Endschalter Bit 0 = Polarität Null-Endschalter Bit 1 = Polarität Referenz-Endschalter (bei LSTEP express keine Wirkung) Bit 2 = Polarität End-Endschalter Wert 0 = Reagiert auf negative Flanke des Endschalters Wert 1 = Reagiert auf positive Flanke des Endschalters	
Beispiel:	LSX.SetSwitchPolarity(7, 0, 0, 0); //Alle Endschalter der X-Achse high-aktiv, alle Endschalter der Y-Achse low-aktiv...		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!swpol	ValidConfig

LSX_GetSwChange	
Beschreibung:	Funktion zum Einlesen der Einstellung Endschalter tauschen. (nur für LSTEP express)
Delphi:	function LSX_GetSwChange(LSID: Integer; var Flags: Integer): Integer;
C++:	int GetSwChange(int *plFlags);
LabView:	

Parameter:	Flags	32-Bit-Integer, welcher nach Aufruf der Funktion in den Bits 0-4 eine Bit-Maske enthält. Bit 0 = X-Achse Bit 1 = Y-Achse … Wert 0 = Endschalter nicht tauschen Wert 1 = Endschalter tauschen	
Beispiel:	LSX.GetSwChange(&Flags);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?swchange	-

LSX_SetSwChange			
Beschreibung:	Funktion zum Setzen der Einstellung Endschalter tauschen. (nur für LSTEP express)		
Delphi:	function LSX_SetSwChange(LSID: Integer; Flags: Integer): Integer;		
C++:	int SetSwChange(int Flags);		
LabView:			
Parameter:	Flags	32-Bit-Integer, mit einer Bit-Maske in den Bits 0-4 Bit 0 = X-Achse Bit 1 = Y-Achse … Wert 0 = Endschalter nicht tauschen Wert 1 = Endschalter tauschen	
Beispiel:	LSX.SetSwChange(3); /* X- und Y-Achse – Endschalter wechseln (Bits 0 u. 1 gesetzt) Z- und A-Achse – kein Endschalterwechsel (Bit 2 = 0) */		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!swchange	ValidConfig

LSX_GetSwitches			
Beschreibung:	Diese Funktion liest den Zustand aller Endschalter ein.		
Delphi:	function LSX_GetSwitches(LSID: Integer; var Flags: Integer): Integer;		
C++:	int GetSwitches(int *plFlags);		
LabView:			
Parameter:	Value	Zeiger auf einen Integerwert, der nach dem Funktionsaufruf den aktuellen Zustand aller Endschalter als Bitmaske enthält. In der Bitmaske 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 Y-Achse sind angefahren Flags = 0x200 → EE von Y-Achse ist angefahren	
Beispiel:	LSX.GetSwitches(&Flags);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?readsw	-

LSX_GetStopSwitchOffDelay			
Beschreibung:	Fragt die Verzögerung zwischen dem Aktivieren des Stopp-Eingangs und Ausschalten der Endstufen und der Motorbremsenausgänge ab (nur in stopmode 1)		
Delphi:	function LSX_GetStopSwitchOffDelay (LSID: Integer; var delay: Integer): Integer;		
C++:	int GetStopSwitchOffDelay(int *pldelay);		
LabView:	Nicht unterstützt		
Parameter:	delay	Eingestellte Verzögerung, Wertebereich: 0 bis 5000ms	
Beispiel:	LSX.GetStopSwitchOffDelay(&delay);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?stopswitchoffdelay	-

LSX_SetStopSwitchOffDelay			
Beschreibung:	Zeigt die Verzögerung zwischen dem Aktivieren des Stopp-Eingangs und Ausschalten der Endstufen und der Motorbremsenausgänge an (nur in stopmode 1)		
Delphi:	function LSX_SetStopSwitchOffDelay (LSID: Integer; delay: Integer): Integer;		
C++:	int SetStopSwitchOffDelay(int ldelay);		
LabView:	Nicht unterstützt		
Parameter:	delay	Einzustellende Verzögerung, Wertebereich: 0 bis 5000ms	
Beispiel:	LSX.SetStopSwitchOffDelay(100);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!stopswitchoffdelay	LSX_ValidConfig

LSX_GetMonitoringVelFilter			
Beschreibung:	Befehl zum Auslesen der Zeitkonstante für Istwertfilter der Geschwindigkeitsüberwachungen (siehe nachfolgende Befehle "HaltSignalVel" und "ThresholdSignalVel")		
Delphi:	function LSX_GetMonitoringVelFilter (LSID: Integer; var X, Y, Z, R: Integer): Integer;		
C++:	int GetMonitoringVelFilter (int *plX, int *plY, int *plZ, int *plR);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, R	16 x System-Ausgabefrequenz 0 - 100 ms	
Beispiel:	LSX.GetMonitoringVelFilter (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?monitoringvelfilter	-

LSX_SetMonitoringVelFilter	
Beschreibung:	Befehl zum Setzen der Zeitkonstante für Istwertfilter der Geschwindigkeitsüberwachungen (siehe nachfolgende Befehle "HaltSignalVel" und "ThresholdSignalVel")
Delphi:	function LSX_SetMonitoringVelFilter (LSID: Integer; X, Y, Z, R: Integer): Integer;
C++:	int SetMonitoringVelFilter (int lX, int lY, int lZ, int lR);

LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, R	16 x System-Ausgabefrequenz 0 - 100 ms	
Beispiel:	LSX.SetMonitoringVelFilter (20, 50, 30, 10);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!monitoringvelfilter	LSX_ValidConfig, LSX_ValidPar

LSX_GetHaltSignalVel			
Beschreibung:	Befehl zum Auslesen des Geschwindigkeitslevel für Stillstandssignal. Das Stillstandsignal wird bei Überschreiten einer der achsspezifischen Geschwindigkeitslevel ausgeschaltet. Ebenso wird das Signal bei einer per Befehl oder durch Handbetrieb angestoßenen Bewegung unabhängig von dem aktuellen Geschwindigkeitswert ausgeschaltet.		
Delphi:	function LSX_GetHaltSignalVel (LSID: Integer; var XD, YD, ZD, RD: Double): Integer;		
C++:	int GetHaltSignalVel (double *pdXD, double *pdYD, double *pdZD, double *pdRD);		
LabView:	Nicht unterstützt		
Parameter:	XD, YD, ZD, RD	Eingestellte Geschwindigkeit: 0 bis Maximum mit bis zu 4 Nachkommastellen in der eingestellten Dimension	
Beispiel:	LSX.GetHaltSignalVel (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?haltsignalvel	-

LSX_SetHaltSignalVel			
Beschreibung:	Befehl zum Setzen des Geschwindigkeitslevel für Stillstandssignal. Das Stillstandsignal wird bei Überschreiten einer der achsspezifischen Geschwindigkeitslevel ausgeschaltet. Ebenso wird das Signal bei einer per Befehl oder durch Handbetrieb angestoßenen Bewegung unabhängig von dem aktuellen Geschwindigkeitswert ausgeschaltet.		
Delphi:	function LSX_SetHaltSignalVel (LSID: Integer; X, Y, Z, R: Integer): Integer;		
C++:	int SetHaltSignalVel (double dXD, double dYD, double dZD, double dRD);		
LabView:	Nicht unterstützt		
Parameter:	XD, YD, ZD, RD	Einzustellende Geschwindigkeit: 0 bis Maximum mit bis zu 4 Nachkommastellen in der eingestellten Dimension	
Beispiel:	LSX.SetHaltSignalVel (2, 4, 3, 1);		

Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!haltsignalvel	LSX_ValidPar

LSX_GetThresholdSignalVel			
Beschreibung:	Befehl zum Auslesen des Wertes für Geschwindigkeits-Schwellwertsignal. Das Geschwindigkeits-Schwellwertignal wird lediglich bei Überschreiten des Schwellwertes ausgeschaltet.		
Delphi:	function LSX_GetThresholdSignalVel (LSID: Integer; var XD, YD, ZD, RD: Double): Integer;		
C++:	int GetThresholdSignalVel (double *pdXD, double *pdYD, double *pdZD, double *pdRD);		
LabView:	Nicht unterstützt		
Parameter:	XD, YD, ZD, RD	Eingestellter Geschwindigkeitsschwellwert: 0 bis Maximum mit bis zu 4 Nachkommastellen in der eingestellten Dimension	
Beispiel:	LSX.GetThresholdSignalVel (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?thresholdsignalvel	-

LSX_SetThresholdSignalVel			
Beschreibung:	Befehl zum Einstellen des Wertes für Geschwindigkeits-Schwellwertsignal. Das Geschwindigkeits-Schwellwertignal wird lediglich bei Überschreiten des Schwellwertes ausgeschaltet.		
Delphi:	function LSX_SetThresholdSignalVel (LSID: Integer; X, Y, Z, R: Integer): Integer;		
C++:	int SetThresholdSignalVel (double dXD, double dYD, double dZD, double dRD);		
LabView:	Nicht unterstützt		
Parameter:	XD, YD, ZD, RD	Einzustellender Geschwindigkeitsschwellwert: 0 bis Maximum mit bis zu 4 Nachkommastellen in der eingestellten Dimension	
Beispiel:	LSX.SetThresholdSignalVel (2, 4, 3, 1);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!thresholdsignalvel	LSX_ValidPar

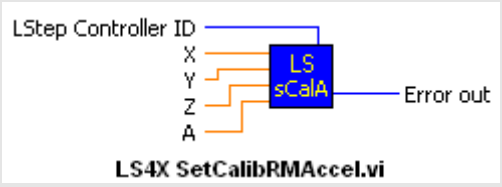
LSX_GetCSOffset			
Beschreibung:	Befehl zum Auslesen des Koordinatensystem-Offsets (Positionswert nach Kalibriervorgang)		
Delphi:	function LSX_GetCSOffset (LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetCSOffset (double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Länge bzw. Winkel des Koordinatensystem-Offsets	
Beispiel:	LSX.GetCSOffset (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?csoffset	-

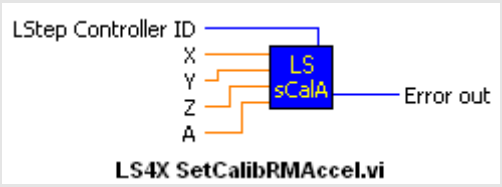
LSX_SetCSOffset			
Beschreibung:	Befehl zum Einstellen des Koordinatensystem-Offsets (Positionswert nach Kalibriervorgang)		
Delphi:	function LSX_SetCSOffset (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetCSOffset (double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Länge bzw. Winkel des Koordinatensystem-Offsets	
Beispiel:	LSX.SetCSOffset (5, 10, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!csoffset	Nach Kalibrieren

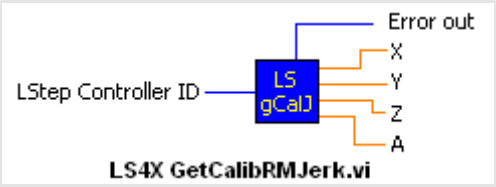
LSX_GetCalibRMOffsetSWAct			
Beschreibung:	Befehl zum Auslesen des Status der Endschalterauswertung während Offsetfahren beim Kalibrieren und Tischhubmessen. Bei abgeschalteter Endschalterauswertung, während Offset-Fahren kann über die Endschalter hinaus ein Offset gefahren werden.		
Delphi:	function LSX_GetCalibRMOffsetSWAct (LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetCalibRMOffsetSWAct (double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Aktuelle Einstellung: An/Aus	
Beispiel:	LSX.GetCalibRMOffsetSWAct (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?calibrmoffsetswact	-

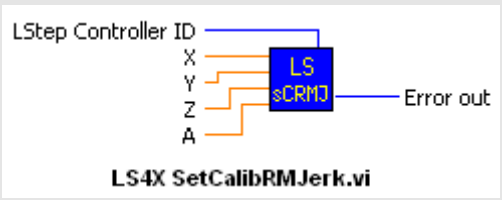
LSX_SetCalibRMOffsetSWAct			
Beschreibung:	Befehl zum Ein- und Ausschalten der Endschalterauswertung während Offsetfahren beim Kalibrieren und Tischhubmessen. Bei abgeschalteter Endschalterauswertung, während Offset-Fahren kann über die Endschalter hinaus ein Offset gefahren werden.		
Delphi:	function LSX_SetCalibRMOffsetSWAct (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetCalibRMOffsetSWAct (double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Aktuelle Einstellung: An/Aus	
Beispiel:	LSX.SetCalibRMOffsetSWAct (False, False, True, True);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!calibrmoffsetswact	Sofort

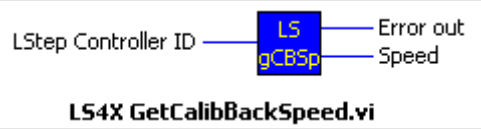
4.2.8 Referenzfahrten

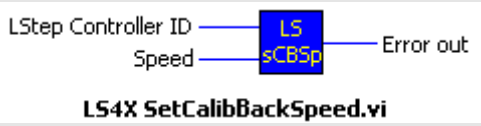
LSX_GetCalibRMAccel			
Beschreibung:	Fragt die Beschleunigung ab, die für den Kalibriervorgang verwendet werden soll. (nur für LSTEP express)		
Delphi:	function LSX_GetCalibRMAccel(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetCalibRMAccel(double *pdXD, double *pdYD, double *pdZD, double *pdAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Beschleunigungswerte in der eingestellten Dimension/s²	
Beispiel:	LSX.GetCalibRMAccel(&XD, &YD, &ZD, &AD);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?calibrmaccel	-

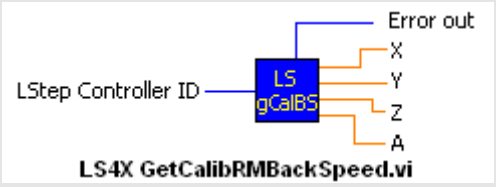
LSX_SetCalibRMAccel			
Beschreibung:	Stellt Beschleunigung für den Kalibriervorgang ein. (nur für LSTEP express)		
Delphi:	function LSX_SetCalibRMAccel(LSID: Integer; XD, YD, ZD, AD: Double): Integer;		
C++:	int SetCalibRMAccel(double dXD, double dYD, double dZD, double dAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Beschleunigungswerte in der eingestellten Dimension/s²	
Beispiel:	LSX.SetCalibRMAccel (1.0, 1.5, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!calibrmaccel	-

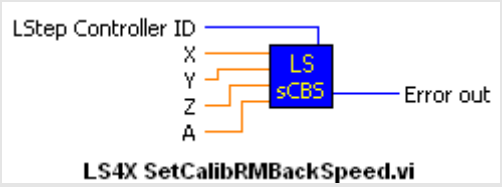
LSX_GetCalibRMJerk			
Beschreibung:	Abfragen des Rucks der während des Kalibriervorgangs verwendet werden soll. (nur für LSTEP express)		
Delphi:	function LSX_GetCalibRMJerk(LSID: Integer; var XD, YD, ZD, AD: Double): Integer;		
C++:	int GetCalibRMJerk(double *pdXD, double *pdYD, double *pdZD, double *pdAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Ruckwerte in der eingestellten Dimension/s³	
Beispiel:	LSX.GetCalibRMJerk(&XD, &YD, &ZD, &AD);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?calibrmjerk	-

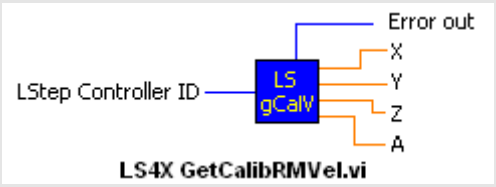
LSX_SetCalibRMJerk			
Beschreibung:	Setzen des Rucks, welcher während des Kalibriervorgangs verwendet werden soll. (nur für LSTEP express)		
Delphi:	function LSX_SetCalibRMJerk(LSID: Integer; XD, YD, ZD, AD: Double): Integer;		
C++:	int SetCalibRMJerk(double dXD, double dYD, double dZD, double dAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Ruckwerte in der eingestellten Dimension/s³	
Beispiel:	LSX.SetCalibRMJerk(1.0, 1.5, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!calibrmjerk	-

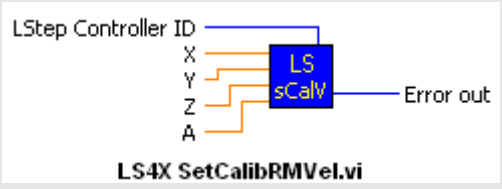
LSX_GetCalibBackSpeed			
Beschreibung:	Liest Verfahrgeschwindigkeiten für das Herausfahren aus den Endschaltern während des Kalibriervorgangs aus. (nur für LSTEP-2000 Familie. Für LSTEP express soll LS_GetCalibRMBackSpeed benutzt werden)		
Delphi:	function LSX_GetCalibBackSpeed(LSID: Integer; var Speed: Integer): Integer;		
C++:	Int GetCalibBackSpeed(int *plSpeed)		
LabView:			
Parameter:	Speed	Geschwindigkeit, entspricht dem gelesenen Wert * 0.01 U/s.	
Beispiel:	LSX.GetCalibBackSpeed (&Speed);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?calbspeed	-

LSX_SetCalibBackSpeed			
Beschreibung:	Stellt Verfahrgeschwindigkeiten für das Herausfahren aus den Endschaltern während des Kalibriervorgangs ein. (nur für LSTEP-2000 Familie. Für LSTEP express soll LS_SetCalibRMBackSpeed benutzt werden)		
Delphi:	function LSX_SetCalibBackSpeed(LSID: Integer; Speed: Integer): Integer;		
C++:	Int SetCalibBackSpeed(int lSpeed)		
LabView:			
Parameter:	Speed	Geschwindigkeit, entspricht dem gelesenen Wert * 0.01 U/s.	
Beispiel:	LSX.SetCalibBackSpeed (10);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!calbspeed	-

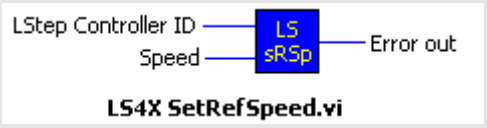
LSX_GetCalibRMBackSpeed			
Beschreibung:	Abfragen der Verfahrgeschwindigkeiten für das Herausfahren aus den Endschaltern die beim Kalibriervorgang bzw. Tischhubmessen verwendet werden sollen. (nur für LSTEP express)		
Delphi:	function LSX_GetCalibRMBackSpeed(LSID: Integer; var XD, YD, ZD, AD: Double): Integer;		
C++:	int GetCalibRMBackSpeed(double *pdXD, double *pdYD, double *pdZD, double *pdAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Geschwindigkeitswerte in der eingestellten Dimension/s	
Beispiel:	LSX.GetCalibRMBackSpeed(&XD, &YD, &ZD, &AD);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?calibrmbspeed	-

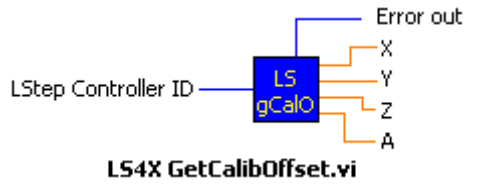
LSX_SetCalibRMBackSpeed			
Beschreibung:	Setzen der Verfahrgeschwindigkeiten die für das Herausfahren aus den Endschaltern während des Kalibriervorgangs bzw. des Tischhubmessens verwendet werden sollen. (nur für LSTEP express)		
Delphi:	function LSX_SetCalibRMBackSpeed(LSID: Integer; XD, YD, ZD, AD: Double): Integer;		
C++:	int SetCalibRMBackSpeed(double dXD, double dYD, double dZD, double dAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Geschwindigkeitswerte in der eingestellten Dimension/s	
Beispiel:	LSX.SeCalibRMBackSpeed(1.0, 15.0, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?calibrmbspeed	-

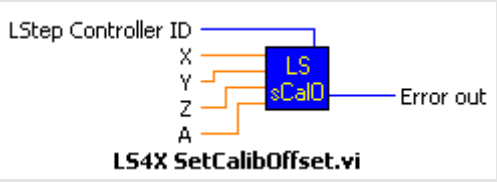
LSX_GetCalibRMVel			
Beschreibung:	Abfragen der Verfahrgeschwindigkeit die während des Kalibriervorgangs verwendet werden soll. (nur für LSTEP express)		
Delphi:	function LSX_GetCalibRMVel(LSID: Integer; var XD, YD, ZD, AD: Double): Integer;		
C++:	int GetCalibRMVel(double *pdXD, double *pdYD, double *pdZD, double *pdAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Geschwindigkeitswerte in der eingestellten Dimension/s	
Beispiel:	LSX.GetCalibRMVel(&XD, &YD, &ZD, &AD);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?calibrmvel	-

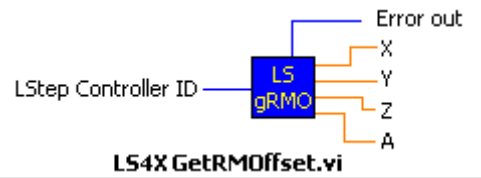
LSX_SetCalibRMVel			
Beschreibung:	Setzen der Verfahrgeschwindigkeiten, welche während des Kalibriervorgangs verwendet werden sollen. (nur für LSTEP express)		
Delphi:	function LSX_SetCalibRMVel (LSID: Integer; XD, YD, ZD, AD: Double): Integer;		
C++:	int SetCalibRMVel(double dXD, double dYD, double dZD, double dAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Geschwindigkeitswerte in der eingestellten Dimension/s	
Beispiel:	LSX.SeCalibRMVel(1.0, 15.0, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!calibrmvel	-

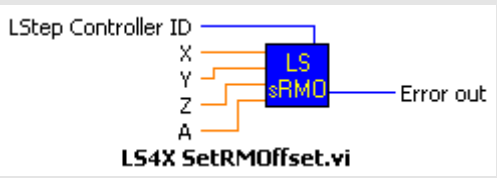
LSX_GetRefSpeed			
Beschreibung:	Liest die Umdrehungsgeschwindigkeit ein, mit der die Achsen beim Suchen nach der Referenzmarke gefahren werden. Die Geschwindigkeit entspricht dem ausgegebenen Wert * 0.01 U/s. (nicht für LSTEP express)		
Delphi:	function LSX_GetRefSpeed(LSID: Integer; var lSpeed: Integer): Integer;		
C++:	int GetRefSpeed(int *plSpeed);		
LabView:			
Parameter:	lSpeed	Geschwindigkeitswert	
Beispiel:	LSX.GetRefSpeed(&lSpeed);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?calrefspeed	-

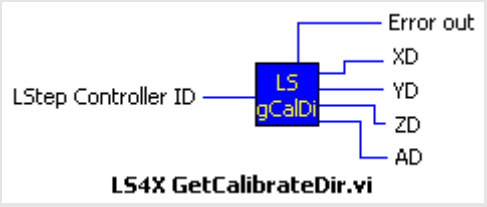
LSX_SetRefSpeed			
Beschreibung:	Setzt die Umdrehungsgeschwindigkeit, mit der die Achsen beim Suchen nach der Referenzmarke gefahren werden. Die Geschwindigkeit entspricht dem angegebenen Wert * 0.01 U/s. (nicht für LSTEP express)		
Delphi:	function LSX_SetRefSpeed(LSID: Integer; lSpeed: Integer): Integer;		
C++:	int SetRefSpeed(int lSpeed);		
LabView:			
Parameter:	lSpeed	Geschwindigkeit	
Beispiel:	LSX.SetRefSpeed(10); //Beim Suchen nach der Referenzmarke wird mit 0.1 U/s gefahren.		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!calrefspeed	-

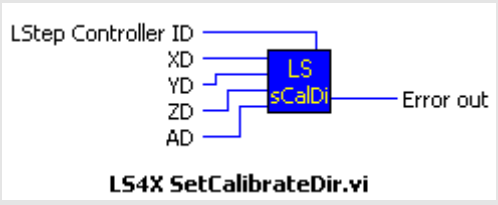
LSX_GetCalibOffset			
Beschreibung:	Funktion zum Abfragen eines Kalibrier-Offsets, der beim Kalibrieren nach dem Freifahren des Entschalters gefahren wird.		
Delphi:	function LSX_GetCalibOffset(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetCalibOffset (double *pdX, double *pdY, double *pdZ, double *pdR);		
LabView:			
Parameter:	X, Y, Z, A	Kalibrier-Offset in der eingestellten Dimension der Achse	
Beispiel:	LSX.GetCalibOffset(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?caliboffset	-

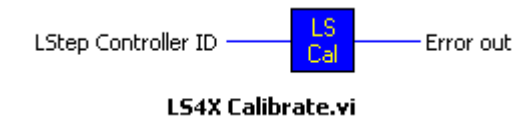
LSX_SetCalibOffset			
Beschreibung:	Funktion zum Setzen eines Kalibrier-Offsets, der beim Kalibrieren nach dem Freifahren des Entschalters gefahren wird.		
Delphi:	function LSX_SetCalibOffset(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetCalibOffset (double dX,double dY,double dZ,double dA);		
LabView:			
Parameter:	X, Y, Z, A	Kalibrier-Offset in der eingestellten Dimension der Achse.	
Beispiel:	LSX.SetCalibOffset(1, 1, 1, 1);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!caliboffset	-

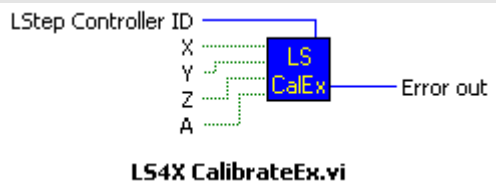
LSX_GetRMOffset			
Beschreibung:	Eingestellten Offset für das nächste Tischhubmessen abfragen.		
Delphi:	function LSX_GetRMOffset(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetRMOffset(double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	X, Y, Z, A	Offset beim Tischhubmessen in der eingestellten Dimension der Achse	
Beispiel:	LSX.GetRMOffset(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?rmoffset	-

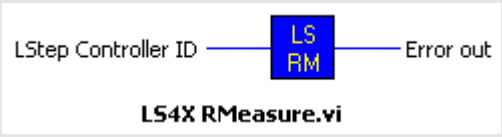
LSX_SetRMOffset			
Beschreibung:	Eingestellten Offset für das nächste Tischhubmessen einstellen.		
Delphi:	function LSX_SetRMOffset(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetRMOffset (double dX, double dY,double dZ,double dA);		
LabView:			
Parameter:	X, Y, Z, A	Offset beim Tischhubmessen in der eingestellten Dimension der Achse	
Beispiel:	LSX.SetRMOffset(1.0, 1.0, 1.0, 1.0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!rmoffset	-

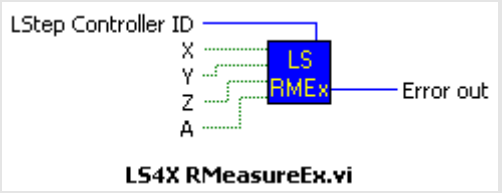
LSX_GetCalibrateDir			
Beschreibung:	Abfrage ob die Vorzeichen-Umkehr beim Kalibrieren aktiv ist.		
Delphi:	function LSX_GetCalibrateDir(LSID: Integer; var XD, YD, ZD, AD: Integer): Integer;		
C++:	int GetCalibrateDir (int *plXD, int *plYD, int *plZD, int *plAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	32-Bit-Integer mit Angabe der Vorzeichen-Umkehr. 0 = keine Vorzeichen-Umkehr 1 = Vorzeichen-Umkehr	
Beispiel:	LSX.GetCalibrateDir(&XD, &YD, &ZD, &AD);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?caldir	-

LSX_SetCalibrateDir			
Beschreibung:	Aktivieren der Vorzeichen-Umkehr beim Kalibrieren		
Delphi:	function LSX_SetCalibrateDir(LSID: Integer; XD, YD, ZD, AD: Integer): Integer;		
C++:	int SetCalibrateDir(int lXD, int lYD, int lZD, int lAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	32-Bit-Integer mit Angabe der Vorzeichen-Umkehr. 0 = keine Vorzeichen-Umkehr 1 = Vorzeichen-Umkehr	
Beispiel:	LSX.SetCalibrateDir(1, 1, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!caldir	-

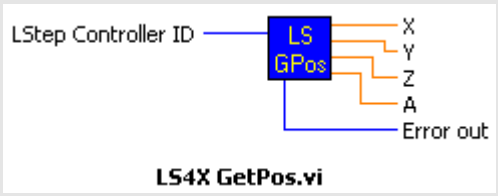
LSX_Calibrate			
Beschreibung:	Diese Funktion startet die Kalibrierroutine. Dadurch werden alle freigegebenen Achsen in Richtung kleinerer Positionswerte verfahren. Die Verfahrbewegung wird unterbrochen sobald die Endschalter angefahren wurden. Der Positionswert wird auf 0 gesetzt.		
Delphi:	function LSX_Calibrate(LSID: Integer): Integer;		
C++:	int Calibrate();		
LabView:			
Parameter:	-		
Beispiel:	LSX.Calibrate();		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!cal	-

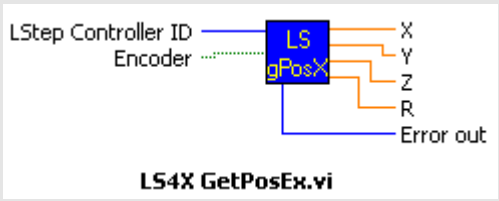
LSX_CalibrateEx			
Beschreibung:	Funktion zum Starten der Kalibrierroutine einer einzelnen Achse.		
Delphi:	function LSX_CalibrateEx(LSID: Integer; Flags: Integer): Integer;		
C++:	int CalibrateEx(int lFlags);		
LabView:			
Parameter:	Flags	Bit-Maske Bit 0 = X-Achse Bit 1 = Y-Achse … Wert 0 = Achse nicht kalibrieren Wert 1 = Achse kalibrieren	
Beispiel:	LSX.CalibrateEx(6); // Nur Y- und Z-Achse kalibrieren		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!cal	-

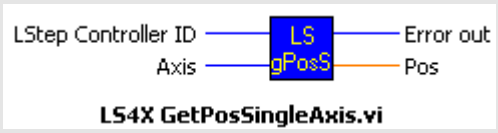
LSX_RMeasure			
Beschreibung:	Startet den Vorgang des Tischhubmessens. Dabei werden alle freigegebenen Achsen in Richtung des größeren Positionswertes verfahren. Die Verfahrbewegung wird unterbrochen sobald die Endschalter angefahren wurden.		
Delphi:	function LSX_RMeasure(LSID: Integer): Integer;		
C++:	int RMeasure();		
LabView:			
Parameter:	-		
Beispiel:	LSX.RMeasure();		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!rm	-

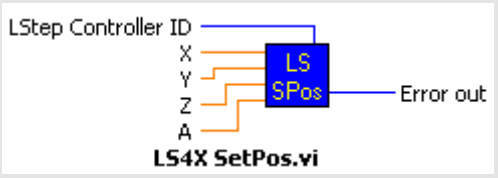
LSX_RMeasureEx			
Beschreibung:	Startet den Vorgang des Tischhubmessens. Dabei werden nur die angegebenen Achsen in Richtung des größeren Positionswertes verfahren. Die Verfahrbewegung wird unterbrochen sobald die Endschalter angefahren wurden.		
Delphi:	function LSX_RMeasureEx(LSID: Integer; Flags: Integer): Integer;		
C++:	int RMeasureEx (int lFlags);		
LabView:			
Parameter:	Flags	Bit-Maske Bit 0 = X-Achse Bit 1 = Y-Achse … Wert 0 = Achse nicht kalibrieren Wert 1 = Achse kalibrieren	
Beispiel:	LSX.RMeasureEx(2); // Tischhub messen (nur Y-Achse)		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!rm	-

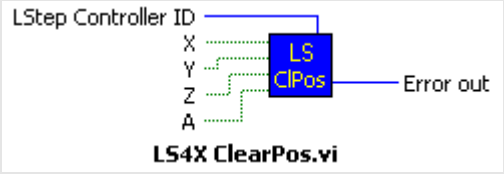
4.2.9 Fahrbefehle und Positionsverwaltung

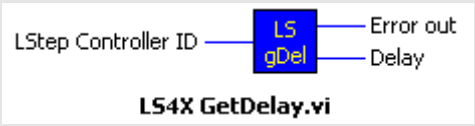
LSX_GetPos			
Beschreibung:	Abfrage der aktuellen Geber- bzw. Positionswerte aller Achsen. Für nicht vorhandene Achsen wird der Wert 0.0 zurückgeliefert.		
Delphi:	function LSX_GetPos(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetPos(double *pdX,double *pdY,double *pdZ,double *pdA);		
LabView:			
Parameter:	X, Y, Z, A	Positionswerte	
Beispiel:	LSX.GetPos(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?pos	-

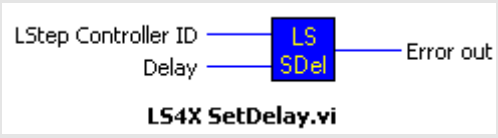
LSX_GetPosEx			
Beschreibung:	Funktion zum Abfragen der aktuellen Geber- bzw. Positionswerte aller Achsen. Für nicht vorhandene Achsen wird der Wert 0.0 zurückgeliefert. Die Quelle der Positionsdaten kann über den Parameter Encoder verändert werden.		
Delphi:	function LSX_GetPosEx(LSID: Integer; var X, Y, Z, R: Double; Encoder: LongBool): Integer;		
C++:	int GetPosEx(double *pdX,double *pdY,double *pdZ,double *pdA,BOOL Encoder);		
LabView:			
Parameter:	X, Y, Z, A	Positionswerte in der eingestellten Einheit	
	Encoder	Encoderwerte auslesen True = Geberwerte liefern falls Geber angeschlossen False = Positionswerte liefern	
Beispiel:	LSX.GetPosEx(&X, &Y, &Z, &A, true);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!enc ?pos	-

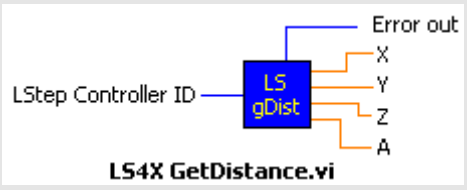
LSX_GetPosSingleAxis			
Beschreibung:	Abfrage der aktuellen Position einer Achse. Für nicht vorhandene Achsen wird der Wert 0.0 zurückgeliefert.		
Delphi:	function LSX_GetPosSingleAxis(LSID: Integer; Axis: Integer; var Pos: Double): Integer;		
C++:	int GetPosSingleAxis(int lAxis,double *pdPos);		
LabView:			
Parameter:	Axis	Achse deren Positionswert abgefragt werden soll X = 1 Y = 2 …	
	Pos	Positionswert	
Beispiel:	LSX.GetPosSingleAxis(2, &YPos); // Position Y-Achse auslesen		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?pos	-

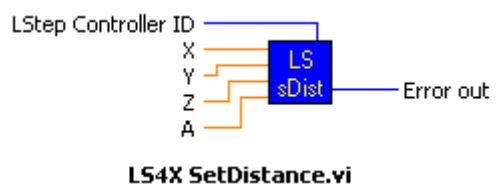
LSX_SetPos			
Beschreibung:	Funktion zum Setzen eines neuen Positionswerts. Dabei wird die aktuelle Position auf die übergebene gesetzt. Der Null-Punkt des Systems verschiebt sich entsprechend.		
Delphi:	function LSX_SetPos(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetPos(double dX, double dY, double dZ, double dA);		
LabView:			
Parameter:	X, Y, Z, A	Positionswerte in der eingestellten Einheit der Achse	
Beispiel:	LSX.SetPos(10, 10, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!pos	-

LSX_ClearPos			
Beschreibung:	Setzt die Position auf Null, auch den internen Zähler. Diese Funktion wird für Endlosachsen gebraucht, da die Steuerung nur einen Wertebereich von +1000 Motorumdrehungen verarbeiten kann. Bei erkanntem Geber wird die Funktion für die jeweilige Achse nicht ausgeführt. (nicht für LSTEP express, siehe Modulo-Betrieb)		
Delphi:	function LSX_ClearPos(LSID: Integer; lFlags: Integer): Integer;		
C++:	int ClearPos (int lFlags);		
LabView:			
Parameter:	lFlags	Bit-Maske Bit 0 = X-Achse Bit 1 = Y-Achse … Wert 0 = Position wird nicht genullt Wert 1 = Position wird genullt	
Beispiel:	LSX.ClearPos(5); //Positionen der x- und z- Achsen werden genullt.		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!clearpos	-

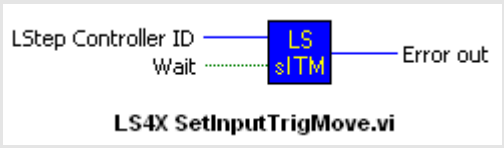
LSX_GetDelay			
Beschreibung:	Liest die Verzögerung des Vektorstarts. (nicht für LSTEP express)		
Delphi:	function LSX_GetDelay(LSID: Integer; var Delay: Integer): Integer;		
C++:	int GetDelay (int *plDelay);		
LabView:			
Parameter:	Delay	Verzögerung in ms	
Beispiel:	LSX.GetDelay(&Delay);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?delay	-

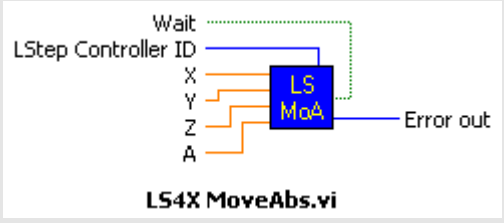
LSX_SetDelay			
Beschreibung:	Durch den Befehl Delay kann eine Verzögerung des Vektorstarts erzeugt werden. (nicht für LSTEP express)		
Delphi:	function LSX_SetDelay(LSID: Integer; Delay: Integer): Integer;		
C++:	int SetDelay(int lDelay);		
LabView:			
Parameter:	Delay	Verzögerung in ms	
Beispiel:	LSX.SetDelay(1000); // 1s Verzögerung		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!delay	-

LSX_GetDistance			
Beschreibung:	Liefert die Strecke für LSX_MoveRelShort		
Delphi:	function LSX_GetDistance(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetDistance(double *pdX, double *pdY, double *pdZ, double *pdR);		
LabView:			
Parameter:	X, Y, Z, A	Aktuelle Strecken aller Achsen, abhängig von den eingestellten Dimensionen	
Beispiel:	LSX.GetDistance(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?distance	-

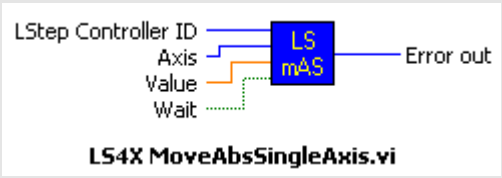
LSX_SetDistance			
Beschreibung:	Strecke setzen für LSX_MoveRelShort		
Delphi:	function LSX_SetDistance(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetDistance(double dX,double dY,double dZ,double dA);		
LabView:			
Parameter:	X, Y, Z, A	Strecke die gefahren werden soll, abhängig von der eingestellten Dimension der Achse	
Beispiel:	LSX.SetDistance(1, 2, 0, 0); /* Strecken für die Achsen X und Y werden gesetzt, Z und A werden bei Aufruf der Funktion LS_MoveRelShort nicht bewegt. */		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!distance	-

LSX_GetInputTrigMove			
Beschreibung:	Liefert die Konfiguration vom Pin1 auf dem MFP. (nicht für LSTEP express)		
Delphi:	function LSX_GetInputTrigMove (LSID: Integer; var Mode: Integer): Integer;		
C++:	int GetInputTrigMove (int *plMode);		
LabView:			
Parameter:	lMode	Eingestellter Modus 0 = Funktion nicht aktiv 1 = Absolut positionieren bei positiver Flanke 2 = Absolut positionieren bei negativer Flanke 3 = Relativ positionieren bei positiver Flanke 4 = Relativ positionieren bei negativer Flanke	
Beispiel:	LSX.GetInputTrigMove(&lMode);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?itm	-

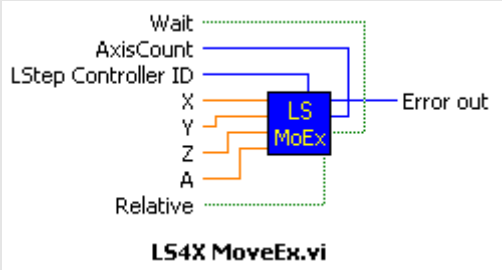
LSX_SetInputTrigMove			
Beschreibung:	Konfiguriert den Pin 1 auf dem MFP, so dass man mit einem externen Signal einen Move starten kann. Verfahren wird der Wert, der über die Funktion LSX_SetDistance gesetzt wurde. (nicht für LSTEP express)		
Delphi:	function LSX_SetInputTrigMove(LSID: Integer; Mode: Integer; Wait: LongBool): Integer;		
C++:	int SetInputTrigMove(int lMode, BOOL bWait);		
LabView:			
Parameter:	lMode	Einzustellender Modus 0 = Funktion nicht aktiv 1 = Absolut positionieren bei positiver Flanke 2 = Absolut positionieren bei negativer Flanke 3 = Relativ positionieren bei positiver Flanke 4 = Relativ positionieren bei negativer Flanke	
Parameter:	bWait	Warte auf einen Move 1 = Es wird so lange gewartet, bis nach einem externen Signal ein Move ausgeführt wird, danach wird der Modus auf 0 gesetzt. 0 = Es wird nicht auf einen Move gewartet. bWait wird nicht ausgewertet, wenn lMode = 0.	
Beispiel:	LSX.SetInputTrigMove(3, False);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!itm	-

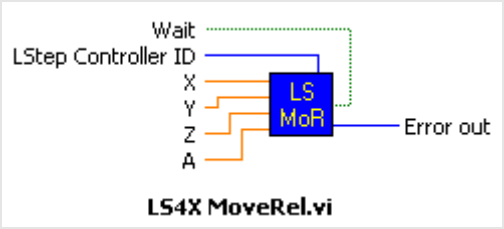
LSX_MoveAbs	
Beschreibung:	Anfahren einer Absolutposition von der aktuellen Position. Die Bewegung erfolgt linearinterpoliert.
Delphi:	function LSX_MoveAbs(LSID: Integer; X, Y, Z, A: Double; Wait: LongBool): Integer;
C++:	int MoveAbs (double dX, double dY, double dZ, double dA, BOOL Wait);
LabView:	

Parameter:	X, Y, Z, A	Absolutposition in der eingestellten Dimension der Achse	
	Wait	Warte auf das Ende der Bewegung True = Warte False = Warte nicht	
Beispiel:	LSX.MoveAbs(10.0, 10.0, 10.0, 10.0, true);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!moa	-

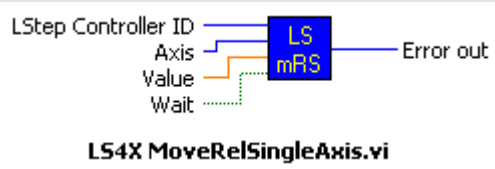
LSX_MoveAbsSingleAxis		
Beschreibung:	Einzelne Achse von der aktuellen Position absolut positionieren	
Delphi:	function LSX_MoveAbsSingleAxis(LSID: Integer; Axis: Integer; Value: Double; Wait: LongBool): Integer;	
C++:	int MoveAbsSingleAxis (int lAxis,double dValue,BOOL Wait);	
LabView:		
Parameter:	Axis	Achse die verfahren werden soll X = 1 Y = 2 …
	Value	Absolutposition in der eingestellten Dimension der Achse
	Wait	Warte auf das Ende der Bewegung True = Warte False = Warte nicht
Beispiel:	LSX.MoveAbsSingleAxis(2, 10.0); // Y-Achse auf 10mm absolut positionieren	

Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!moa	-

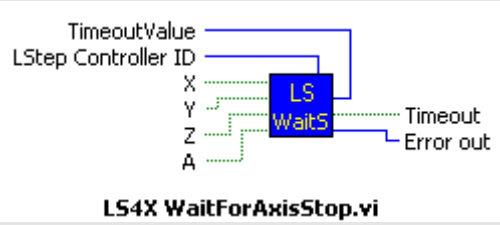
<table>
<tr><th colspan="4">LSX_MoveEx</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Die Funktion LSX_MoveEx ist ein erweiterter Verfahr-Befehl. Sie kann relative und absolute Verfahrbefehle ausführen, synchron und asynchron. Die Anzahl der Achsen, die verfahren werden sollen, kann mit dem Parameter AxisCount festgelegt werden, siehe die Beschreibung zum Parameter AxisCount.</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_MoveEx(LSID: Integer; X, Y, Z, A: Double; Relative, Wait: LongBool; AxisCount: Integer): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int MoveEx(double dX, double dY, double dZ, double dA, BOOL bRelative, BOOL bWait, int lAxisCount);</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td rowspan="4">Parameter:</td><td>X, Y, Z, A</td><td colspan="2">Positions-Vektor in der eingestellten Einheit der Achse</td></tr>
<tr><td>Relative</td><td colspan="2">Angabe ob Positions-Vektor relativ gefahren werden soll
True = Vektor wird relativ zur aktuellen Position gefahren
False = Vektor wird absolut zur aktuellen Position gefahren</td></tr>
<tr><td>Wait</td><td colspan="2">Warte auf Ende der Verfahrbewegung
True = Die Funktion kehrt erst nach Erreichen der Zielposition zurück.
False = Die Funktion kehrt unmittelbar nach Senden des Befehls zurück.</td></tr>
<tr><td>AxisCount</td><td colspan="2">Anzahl der Achsen, die verfahren werden sollen
1 = nur X-Achse
2 = X- und Y-Achse
3 = X-, Y- und Z-Achse
…</td></tr>
<tr><td>Beispiel:</td><td colspan="3">LS_MoveEx(2.0, 3.0, 0, 0, true, true, 2) ; // Es werden X und Y um 2 bzw 3 relativ verfahren</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>3</td><td>!moa / !mor</td><td>-</td></tr>
</table>

LSX_MoveRel			
Beschreibung:	Funktion zum Fahren eines relativen Vektors von der aktuellen Position.		
Delphi:	function LSX_MoveRel(LSID: Integer; X, Y, Z, A: Double; Wait: LongBool): Integer;		
C++:	int MoveRel(double dX, double dY, double dZ, double dA, BOOL Wait);		
LabView:			
Parameter:	X, Y, Z, A	Relative Positionsangabe in der eingestellten Dimension der Achse	
	Wait	Warte auf das Ende der Bewegung True = Warte False = Warte nicht	
Beispiel:	LSX.MoveRel(10.0, 10.0, 10.0, 10.0, true);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!mor	-

LSX_MoveRelShort			
Beschreibung:	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.		
Delphi:	function LSX_MoveRelShort(LSID: Integer): Integer;		
C++:	int MoveRelShort();		
LabView:			
Parameter:	-		
Beispiel:	LSX.SetDistance(1.0, 1.0, 0, 0); for (i = 0; i < 10; i++) LSX.MoveRelShort(); // 10mal X- und Y-Achse um 1 mm relativ positionieren		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!m	-

LSX_MoveRelSingleAxis			
Beschreibung:	Einzelne Achse von der aktuellen Position relativ verfahren.		
Delphi:	function LSX_MoveRelSingleAxis(LSID: Integer; Axis: Integer; Value: Double; Wait: LongBool): Integer;		
C++:	int MoveRelSingleAxis(int lAxis,double dValue,BOOL Wait);		
LabView:			
Parameter:	Axis	Achse die verfahren werden soll X = 1 Y = 2 …	
	Value	Relativposition in der eingestellten Dimension der Achse	
	Wait	Warte auf das Ende der Bewegung True = Warte False = Warte nicht	
Beispiel:	LSX.MoveRelSingleAxis(3, 5.0); // Z-Achse um 5mm in positiver Richtung verfahren		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!mor	-

LSX_StopAxes			
Beschreibung	Funktion zum Abbruch aller Verfahrbewegungen.		
Delphi	function LSX_StopAxes(LSID: Integer): Integer;		
C++	int StopAxes ();		
LabView			
Parameter	-		
Beispiel	LSX.StopAxes();		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!a	-

LSX_WaitForAxisStop			
Beschreibung:	Die Funktion kehrt zurück, sobald die in der Bit-Maske AFlags gewählten Achsen ihre Zielposition erreicht haben. LSX_WaitForAxisStop verwendet ‚?statusaxis', um den Status der Achsen zu pollen.		
Delphi:	function LSX_WaitForAxisStop(LSID: Integer; AFlags: Integer; ATimeoutValue: Integer; var ATimeout: LongBool): Integer;		
C++:	int WaitForAxisStop(int lAFlags, int lATimeoutValue, BOOL *pbATimeout);		
LabView:			
Parameter:	AFlags	Bit-Maske Bit 0 = X-Achse Bit 1 = Y-Achse … Wert 0 = Achse nicht kalibrieren Wert 1 = Achse Kalibrieren	
	AtimeoutValue	Timeout in Millisekunden. Der Wert 0 deaktiviert die Funktion des Timeout.	
	ATimeout	Das Atimeout Flag gibt an, ob ein Timeout aufgetreten ist. Dies ist der Fall, wenn nach Ablauf der Zeit in ATimeOutValue immernoch die Bewegung der angegebenen Achsen aktiv ist. True = Timeout aufgetreten, Bewegung noch aktiv False = Timeout nicht aufgetreten, Bewegung beendet	

Beispiel:	LSX.WaitForAxisStop(3, 0, flag);
	// Warten bis X und Y-Achse gestoppt haben, kein Timeout
	LSX.WaitForAxisStop(7, 10000, flag);
	// Warten bis X und Y-Achse gestoppt haben, 10 Sekunden Timeout		

Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?statusaxis	-

LSX_GetPosWindowRange			
Beschreibung:	Fragt die eingestellten Zielfenster der Achsen ab. Zielposition erreicht wird aktiv, wenn sich die Achse für die Dauer der Zielfensterzeit im Zielfenster befindet.		
Delphi:	function LSX_GetPosWindowRange(LSID: Integer,var X,Y,Z,A: Double) : Integer;		
C++:	int GetPosWindowRange (double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Zielfenster der Achse	
Beispiel:	LSX.GetPosWindowRange(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?poswindowrange	-

LSX_SetPosWindowRange			
Beschreibung:	Stellt die Zielfenster der Achsen ein. Zielposition erreicht wird aktiv, wenn sich die Achse für die Dauer der Zielfensterzeit im Zielfenster befindet.		
Delphi:	function LSX_SetPosWindowRange(LSID: Integer; X,Y,Z,A : double): Integer;		
C++:	int SetPosWindowRange (double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Zielfenster der Achse	
Beispiel:	LSX.SetPosWindowRange(0.01,0.02,0.03,0.04);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!poswindowrange	-

LSX_GetPosWindowTime			
Beschreibung:	Befehl zum Abfragen der Zielfensterzeit. Zielposition-Erreicht-Meldung wird aktiv, wenn die Achse sich für die Dauer der Zielfensterzeit im Zielfenster befindet.		
Delphi:	function LSX_GetPosWindowTime(LSID: Integer;var X,Y,Z,A: Integer): Integer;		
C++:	int GetPosWindowTime (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Zielfensterzeit	
Beispiel:	LSX.GetPosWindowTime(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?poswindowtime	-

LSX_SetPosWindowTime			
Beschreibung:	Befehl zum Abfragen der Zielfensterzeit. Zielposition-Erreicht-Meldung wird aktiv, wenn die Achse sich für die Dauer der Zielfensterzeit im Zielfenster befindet.		
Delphi:	function LSX_SetPosWindowTime(LSID: Integer; X,Y,Z,A : integer): Integer;		
C++:	int SetPosWindowTime(int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Zielfensterzeit	
Beispiel:	LSX.SetPosWindowTime(100,100,100,100);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!poswindowtime	-

LSX_GetPosWindowTimeout			
Beschreibung:	Liest den Timeout aller Achsen aus. Der Timeout stellt sich ein, wenn das Einregeln (nach Ende der Verfahrbewegung) in das Zielfenster mit einer Fehlermeldung abgebrochen wird. Es muss größer gleich der Zielfensterzeit gewählt werden.		
Delphi:	function LSX_GetPosWindowTimeout(LSID: Integer;var X,Y,Z,A: Integer): Integer;		
C++:	int GetPosWindowTimeout (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Eingestelltes Timeout der jeweiligen Achsen.	
Beispiel:	LSX.GetPosWindowTimeout(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?poswindowtimeout	-

LSX_SetPosWindowTimeout			
Beschreibung:	Stellt den Timeout aller Achsen ein. Der Timeout stellt sich ein, wenn das Einregeln (nach Ende der Verfahrbewegung) in das Zielfenster mit einer Fehlermeldung abgebrochen wird. Es muss größer gleich der Zielfensterzeit gewählt werden.		
Delphi:	function LSX_SetPosWindowTimeout(LSID: Integer; X,Y,Z,A : Integer): Integer;		
C++:	int SetPosWindowTimeout(int lX, int lY, int lZ, int lA) ;		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Eingestelltes Timeout der jeweiligen Achsen.	
Beispiel:	LSX.SetPosWindowRange(800,800,800,800);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!poswindowtimeout	-

LSX_GetPosWindowCheck			
Beschreibung:	Befehl zur Abrage ob die Zielfensterüberwachung aktiv ist.		
Delphi:	function LSX_GetPosWindowCheck (LSID: Integer;var X,Y,Z,A: LongBool):Integer;		
C++:	int GetPosWindowCheck(BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Zustand der Zielfensterüberwachung	
Beispiel:	LSX.GetPosWindowCheck(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?poswindowcheck	-

LSX_SetPosWindowCheck			
Beschreibung:	Befehl zum Ein- und Ausschalten der Zielfensterüberwachung.		
Delphi:	function LSX_SetPosWindowCheck (LSID: Integer;X,Y,Z,A: LongBool):Integer;		
C++:	int SetPosWindowCheck (BOOL bX, BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Zustand der Zielfensterüberwachung	
Beispiel:	LSX.SetPosWindowCheck(True,True,True,True);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!poswindowcheck	-

LSX_SetHalt			
Beschreibung:	Befehl zum Anhalten von Verfahrbewegungen. Es wird mit Verzögerungen der Bewegungen angehalten.		
Delphi:	function LSX_SetHalt (LSID: Integer;X,Y,Z,R: LongBool):Integer;		
C++:	int SetHalt (BOOL bX, BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Achsenmaske (siehe Befehl „statusaxis") mit einem '@' für Achsen die nach einem Halt angehalten sind. Ist keine Bewegung aktiv erfolgt keine Rückmeldung.	
Beispiel:	LSX.SetHalt(True,True,True,True);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!halt	-

LSX_GetPosMode			
Beschreibung:	Befehl zum Auslesen des Status für Sollposition-Handling bei Regler und Endstufe im ungeregelten Schrittmotorbetrieb		
Delphi:	function LSX_GetPosMode (LSID: Integer; var X, Y, Z, A: LongBool): Integer;		
C++:	int GetPosMode (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellter Modus: An/Aus	
Beispiel:	LSX.GetPosMode (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?posmode	-

LSX_SetPosMode			
Beschreibung:	Befehl zum Einstellen des Status für Sollposition-Handling bei Regler und Endstufe im ungeregelten Schrittmotorbetrieb		
Delphi:	function LSX_SetPosMode (LSID: Integer; X, Y, Z, A: LongBool): Integer;		
C++:	int SetPosMode (BOOL bX, BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellter Modus: An/Aus	
Beispiel:	LSX.SetPosMode (False, True, True, False);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!posmode	Sofort

LSX_MoveAbsV, LSX_MoveAbsVW			
Beschreibung:	Anfahren einer Absolutposition. Es wird eine Linearinterpolation unter Berücksichtigung der achsspezifischen Grenzwerte für Geschwindigkeit, Beschleunigung und Ruck durchgeführt. Eingabewerte sind abhängig von der Dimension.		
Delphi:	function LSX_MoveAbsV(LSID: Integer; XD, YD, ZD, AD: Double; var Feedback: PAnsiChar; MaxLen: Integer): Integer; LSX_MoveAbsVW(LSID: Integer; XD, YD, ZD, AD: Double; var Feedback: PWideChar; MaxLen: Integer): Integer;		
C++:	int MoveAbsV (double dX, double dY, double dZ, double dA, char *pcFeedback, int lMaxLen); int MoveAbsVW (double dX, double dY, double dZ, double dA, TCHAR *pcFeedback, int lMaxLen);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Verfahrensbereich jeder Achse	
Rückmeldung:	Feedback	Für jede Achse nach Erreichen: @	
Beispiel:	LSX.MoveAbsV (2.5, 4, 8.3, 10, pcFeedback,256);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!moav	Sofort

LSX_MoveRelV, LSX_MoveRelVW		
Beschreibung:	Anfahren einer Position relativ zur Startposition. Es wird eine Linearinterpolation unter Berücksichtigung der achsspezifischen Grenzwerte für Geschwindigkeit, Beschleunigung und Ruck durchgeführt. Eingabewerte sind abhängig von der Dimension.	
Delphi:	LSX_MoveRelV(LSID: Integer; XD, YD, ZD, AD: Double; var Feedback: PAnsiChar; MaxLen: Integer): Integer; LSX_MoveRelVW(LSID: Integer; XD, YD, ZD, AD: Double; var Feedback: PWideChar; MaxLen: Integer): Integer;	
C++:	int MoveRelV (double dX, double dY, double dZ, double dA, char *pcFeedback, int lMaxLen); int MoveRelVW (double dX, double dY, double dZ, double dA, TCHAR *pcFeedback, int lMaxLen);	
LabView:	Nicht unterstützt	
Parameter:	X, Y, Z, A	Verfahrensbereich jeder Achse
Rückmeldung:	Feedback	Für jede Achse nach Erreichen: @
Beispiel:	LSX.MoveRelV (2.5, 4, 8.3, 10, pcFeedback, 256);	

Sonstiges:	Kompatibili-tät	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	!morv	Sofort

LSX_GetAutoKomm			
Beschreibung:	Befehl zum Auslesen des Status der Autokommutierung für Servo-Achsen mit inkrementellen Messsystemen. Beim Autokommutieren wird die Lage der Motorwicklungen zum Messsystem mit Hilfe von Verfahrbewegungen ermittelt, um einen Servobetrieb zu ermöglichen. Hierbei findet zusätzlich ein Vergleich der rechnerischen und gemessenen Messsystemauslenkung statt. Für eine erfolgreiche Autokommutierung darf die Abweichung zwischen der rechnerischen und gemessenen Messsystemauslenkung maximal ± 30% (siehe Befehl "GetAutoKommResult") betragen.		
Delphi:	function LSX_GetAutoKomm (LSID: Integer; var X, Y, Z, A: LongBool): Integer;		
C++:	int GetAutoKomm (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Status: 0: Autokommutierung nicht durchgeführt 1: Autokommutierung durchgeführt	
Beispiel:	LSX.GetAutoKomm (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?autokomm	-

LSX_SetAutoKomm, LSX_SetAutokommW		
Beschreibung:	Befehl zum manuellen Anstoßen der Autokommutierung für Servo-Achsen mit inkrementellen Messsystemen. Beim Autokommutieren wird die Lage der Motorwicklungen zum Messsystem mit Hilfe von Verfahrbewegungen ermittelt, um einen Servobetrieb zu ermöglichen. Hierbei findet zusätzlich ein Vergleich der rechnerischen und gemessenen Messsystemauslenkung statt. Für eine erfolgreiche Autokommutierung darf die Abweichung zwischen der rechnerischen und gemessenen Messsystemauslenkung maximal ± 30% (siehe Befehl "GetAutoKommResult") betragen.	
Delphi:	function LSX_SetAutoKomm(LSID: Integer; var Feedback: PAnsiChar; MaxLen: Integer): Integer; function LSX_SetAutoKommW(LSID: Integer; var Feedback: PWideChar; MaxLen: Integer): Integer;	
C++:	int SetAutoKomm (char *pcFeedback, int lMaxLen); int SetAutoKommW (TCHAR *pcFeedback, int lMaxLen);	
LabView:	Nicht unterstützt	
Rückmeldung:	Feedback	Für jede vorhandene Achse ein '@' nach erfolgreicher Autokommutierung
Beispiel:	LSX.SetAutoKomm(pcFeedback,256);	

Sonstiges:	Kompatibili-tät	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	!autokomm	Sofort

LSX_GetAutoKommResult			
Beschreibung:	Befehl zum Auslesen des Autokommutierungs-Ergebnisses. Verhältnis zwischen der gemessenen Messsystemauslenkung und rechnerischen Auslenkung beim Autokommutieren in [%], Wertebereich ist ± 999.99 %. Optimales Ergebnis entspricht 100%. Zur Aktivierung des Servobetriebs muss der Wert zwischen 70% und 130% liegen.		
Delphi:	function LSX_GetAutoKommResult (LSID: Integer; var X1, X2, Y1, Y2, Z1, Z2, A1, A2: Double): Integer;		
C++:	int GetAutoKommResult (double *pdX1, double *pdX2, double *pdY2, double *pdY2, double *pdZ1, double *pdZ2, double *pdA1, double *pdA2);		
LabView:	Nicht unterstützt		
Parameter:	X1, X2, Y1, Y2, Z1, Z2, A1, A2	Verhältnis zwischen der gemessenen Messsystemauslenkung und rechnerischen Auslenkung (2 Werte pro Achse)	
Beispiel:	LSX. GetAutoKommResult (&X1, &X2, &Y1, &Y2, &Z1, &Z2, &A1, &A2);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!autokommresult	-

LSX_GetMixedMoveAxisMode			
Beschreibung:	Liest den Modus der Achse für eine gemischte Verfahrbewegung aus. Der Zusammenhang der Modi, der zugehörigen Einheiten und dahinterstehenden Funktionen kann in Tabelle „Gemischte Verfahrbewegung – Zusammenhang der Modi“ nachgeschlagen werden		
Delphi:	function LSX_GetMixedMoveAxisMode (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetMixedMoveAxisMode (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Modus der Achsen: 0: Lineare Bewegung 1: Sinus Bewegung 2: Cosinus Bewegung	
Beispiel:	LSX.GetMixedMoveAxisMode (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	?mmaxism	-

LSX_SetMixedMoveAxisMode			
Beschreibung:	Stellt den Modus der Achse für eine gemischte Verfahrbewegung ein. Der Zusammenhang der Modi, der zugehörigen Einheiten und dahinterstehenden Funktionen kann in Tabelle „Gemischte Verfahrbewegung – Zusammenhang der Modi“ nachgeschlagen werden		
Delphi:	function LSX_SetMixedMoveAxisMode (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetMixedMoveAxisMode (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Modus der Achsen: 0: Lineare Bewegung 1: Sinus Bewegung 2: Cosinus Bewegung	
Beispiel:	LSX.SetMixedMoveAxisMode (2, 1, 0, 0);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	!mmaxism	MixedMoveAbs / MixedMoveRel

LSX_MixedMoveAbs, LSX_MixedMoveAbsW			
Beschreibung:	Anfahren einer Absolutposition. Es wird eine Bewegung unter Berücksichtigung der achsspezifischen Grenzwerte für Geschwindigkeit, Beschleunigung und Ruck durchgeführt. Eingabewerte sind abhängig vom eingestellten Modus der gemischten Verfahrbewegung (siehe Tabelle „Gemischte Verfahrbewegung - Zusammenhang der Modi“).		
Delphi:	function LSX_MixedMoveAbs(LSID: Integer; X: Double; Y: Double; Z: Double; A: Double; var Feedback: PAnsiChar; MaxLen: Integer): Integer; function LSX_MixedMoveAbsW(LSID: Integer; X: Double; Y: Double; Z: Double; A: Double; var Feedback: PWideChar; MaxLen: Integer): Integer;		
C++:	int MixedMoveAbs (double dX, double dY, double dZ, double dA, char *pcFeedback, int lMaxLen); int MixedMoveAbsW (double dX, double dY, double dZ, double dA, TCHAR *pcFeedback, int lMaxLen);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Verfahrensbereich jeder Achse	
Rückmeldung:	Feedback	Für jede Achse nach Erreichen: @	
Beispiel:	LSX.MixedMoveAbs (2.5, 4, 8.3, 10, pcFeedback,256);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	!mmoa	Sofort

LSX_MixedMoveRel, LSX_MixedMoveRelW		
Beschreibung:	Positionierung relativ zur Startposition. Es wird eine Bewegung unter Berücksichtigung der achsspezifischen Grenzwerte für Geschwindigkeit, Beschleunigung und Ruck durchgeführt. Eingabewerte sind abhängig vom eingestellten Modus der gemischten Verfahrbewegung (siehe Tabelle „Gemischte Verfahrbewegung - Zusammenhang der Modi“).	
Delphi:	function LSX_MixedMoveRel(LSID: Integer; X: Double; Y: Double; Z: Double; A: Double; var Feedback: PAnsiChar; MaxLen: Integer): Integer; function LSX_MixedMoveRelW(LSID: Integer; X: Double; Y: Double; Z: Double; A: Double; var Feedback: PWideChar; MaxLen: Integer): Integer;	
C++:	int MixedMoveRel (double dX, double dY, double dZ, double dA, char *pcFeedback, int lMaxLen); int MixedMoveRelW (double dX, double dY, double dZ, double dA, TCHAR *pcFeedback, int lMaxLen);	
LabView:	Nicht unterstützt	
Parameter:	X, Y, Z, A	Verfahrensbereich jeder Achse Rückgabe nach Erreichen: @

Rückmeldung:	Feedback	Für jede Achse nach Erreichen: @	
Beispiel:	LSX.MixedMoveRel (2.5, 4, 8.3, 10, pcFeedback,256);		
Sonstiges:	Kompatibili-tät	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!mmor	Sofort

LSX_GetMixedMovePos			
Beschreibung:	Befehl zum Auslesen von Positionswerten der gemischten Verfahrbewegung. Ist der Modus 0 eingestellt, wird die Position nicht gesetzt bzw. 0 ausgelesen.		
Delphi:	function LSX_GetMixedMovePos (LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetMixedMovePos (double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Positionswerte der Achsen	
Beispiel:	LSX.GetMixedMovePos (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?mmpos	-

LSX_SetMixedMovePos			
Beschreibung:	Befehl zum Einstellen von Positionswerten der gemischten Verfahrbewegung. Ist der Modus 0 eingestellt, wird die Position nicht gesetzt bzw. 0 ausgelesen.		
Delphi:	function LSX_SetMixedMovePos (LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetMixedMovePos (double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Positionswerte der Achsen	
Beispiel:	LSX.SetMixedMovePos (360, 200, 300, 200);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!mmpos	Sofort

LSX_GetMixedMoveAmpl			
Beschreibung:	Befehl zum Auslesen der Amplitude bzw. des Skalierungsfaktors bei gemischten Verfahrbewegungen. Die Auswirkung der Amplitude kann in Tabelle "Gemischte Verfahrbewegung - Zusammenhang der Modi" nachgeschlagen werden.		
Delphi:	function LSX_GetMixedMoveAmpl (LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetMixedMoveAmpl (double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Amplitude / Skalierungsfaktor	
Beispiel:	LSX.GetMixedMoveAmpl (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?mmampl	-

LSX_SetMixedMoveAmpl			
Beschreibung:	Befehl zum Einstellen der Amplitude bzw. des Skalierungsfaktors bei gemischten Verfahrbewegungen. Die Auswirkung der Amplitude kann in Tabelle "Gemischte Verfahrbewegung - Zusammenhang der Modi" nachgeschlagen werden.		
Delphi:	function LSX_SetMixedMoveAmpl (LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetMixedMoveAmpl (double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Amplitude / Skalierungsfaktor	
Beispiel:	LSX.SetMixedMoveAmpl (10, 300, -9, 100);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!mmampl	MixedMoveAbs / MixedMoveRel

LSX_GetIndexTableDivider			
Beschreibung:	Befehl zum Auslesen des Teilers einer Rundachse für Teilkopfbetrieb. Dieser Befehl lässt sich nur mit den Dimensionen 'Microsteps', 'Grad' und 'Umdrehungen' verwenden.		
Delphi:	function LSX_GetIndexTableDivider (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetIndexTableDivider (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Teiler der Rundachsen	
Beispiel:	LSX.GetIndexTableDivider (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?itd	-

LSX_SetIndexTableDivider			
Beschreibung:	Befehl zum Festlegen des Teilers einer Rundachse für Teilkopfbetrieb. Dieser Befehl lässt sich nur mit den Dimensionen 'Microsteps', 'Grad' und 'Umdrehungen' verwenden.		
Delphi:	function LSX_SetIndexTableDivider (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetIndexTableDivider (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Teiler der Rundachsen	
Beispiel:	LSX.SetIndexTableDivider (50, 10, 5, 20);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!itd	Sofort

LSX_MoveIndexTable, LSX_MoveIndexTableW			
Beschreibung:	Befehl zum Anfahren einer Position im Teilkopfbetrieb. Der übergebene Wert stellt den Multiplikator für die Berechnung der Zielposition dar. Dieser Befehl lässt sich nur mit den Dimensionen 'Microsteps', 'Grad' und 'Umdrehungen' verwenden.		
Delphi:	function LSX_MoveIndexTable(LSID: Integer; X: Integer; Y: Integer; Z: Integer; A: Integer; var Feedback: PAnsiChar; MaxLen: Integer): Integer; function LSX_MoveIndexTableW(LSID: Integer; X: Integer; Y: Integer; Z: Integer; A: Integer; var Feedback: PWideChar; MaxLen: Integer): Integer;		
C++:	int MoveIndexTable (int lX, int lY, int lZ, int lA, char *pcFeedback, int lMaxLen); int MoveIndexTableW (int lX, int lY, int lZ, int lA, TCHAR *pcFeedback, int lMaxLen);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Positionswerte der Achsen	
Rückmeldung:	Feedback	Für jede vorhandene Achse nach Erreichen: @	
Beispiel:	LSX.MoveIndexTable (5, 4, 2, 8, pcFeedback,256);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!mita	Sofort

LSX_GetMoveSeqStatusPos			
Beschreibung:	Befehl zum Auslesen des Status eines konfigurierbaren Bewegungsablauf (Derzeit ist nur Ablauf Nr. 1 in Steuerung integriert, weitere Abläufe auf Anfrage). Ausgewählter Ablauf kann mit 'MoveSeqStatus 1' gestartet werden.		
Delphi:	function LSX_GetMoveSeqStatusPos (LSID: Integer; var status: LongBool): Integer;		
C++:	int GetMoveSeqStatusPos (BOOL *pbstatus);		
LabView:	Nicht unterstützt		
Parameter:	status	Status des Bewegungsablaufs: 0: Kein Bewegungsablauf gewählt 1: Bewegungsablauf 1 (n-fache Relativ-Bewegung auf Tabellenposition) gewählt	
Beispiel:	LSX.GetMoveSeqStatusPos (&status);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?mssp	-

LSX_SetMoveSeqStatusPos			
Beschreibung:	Befehl zur Auswahl eines konfigurierbaren Bewegungsablauf (Derzeit ist nur Ablauf Nr. 1 in Steuerung integriert, weitere Abläufe auf Anfrage). Ausgewählter Ablauf kann mit 'MoveSeqStatus 1' gestartet werden.		
Delphi:	function LSX_SetMoveSeqStatusPos (LSID: Integer; status: LongBool): Integer;		
C++:	int SetMoveSeqStatusPos (BOOL bstatus);		
LabView:	Nicht unterstützt		
Parameter:	status	Status des Bewegungsablaufs: 0: Kein Bewegungsablauf gewählt 1: Bewegungsablauf 1 (n-fache Relativ-Bewegung auf Tabellenposition) gewählt	
Beispiel:	LSX.SetMoveSeqStatusPos (1);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!mssp	Sofort

Aktuell implementierter Bewegungsablauf 1:

```
vardef
  int a = 0
  int b = 0
  int c = 0
  int d = 0
  int e = 0
endvardef
prog
  autostatus 0
  loop a
    set b = c
    for b to d step e
      mtpr b
      waitforaxisstop
      next
  endloop
  autostatus 1
  sendstatusaxis
end prog
```

LSX_GetMoveSeqStatus			
Beschreibung:	Befehl zum Auslesen des Status des ausgewählten Bewegungsablaufs (siehe Befehl 'MoveSeqStatus'). Ablauf kann gestartet, gestoppt oder in Einzelschritten abgearbeitet werden.		
Delphi:	function LSX_GetMoveSeqStatus (LSID: Integer; var status, line: Integer): Integer;		
C++:	int GetMoveSeqStatus (int *plstatus, int *plline);		
LabView:	Nicht unterstützt		
Parameter:	Status, Line	Ablaufstatus und nächste auszuführende Programmzeile: 0: Ablauf inaktiv 1: Ablauf aktiv Aktuelle Programmzeile	
Beispiel:	LSX.GetMoveSeqStatus (&status);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?mss	-

LSX_SetMoveSeqStatus, LSX_SetMoveSeqStatusW			
Beschreibung:	Befehl zum Steuern des ausgewählten Bewegungsablaufs (siehe Befehl 'MoveSeqStatus'). Ablauf kann gestartet, gestoppt oder in Einzelschritten abgearbeitet werden.		
Delphi:	function LSX_SetMoveSeqStatus(LSID: Integer; status: Integer; var Feedback: PAnsiChar; MaxLen: Integer): Integer; function LSX_SetMoveSeqStatusW(LSID: Integer; status: Integer; var Feedback: PWideChar; MaxLen: Integer): Integer;		
C++:	int SetMoveSeqStatus (int lstatus, char *pcFeedback, int lMaxLen); int SetMoveSeqStatusW (int lstatus, TCHAR *pcFeedback, int lMaxLen);		
LabView:	Nicht unterstützt		
Parameter:	status	Status des Bewegungsablaufs: 0: Ablauf stoppen 1: Ablauf starten 2: Einzelschritt ausführen	
Beispiel:	LSX.SetMoveSeqStatus (1, pcFeedback,256);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!mss	Sofort

LSX_GetMoveSeqVar			
Beschreibung:	Befehl zum Auslesen von Variablenwerten, die für Bewegungsabläufe verwendet werden, um ihren Laufeigenschaften zu beeinflussen. Wird vor Beginn des Programms kein Wert gesetzt, werden Standardwerte verwendet. Diese rufen bei Standardprogrammen keine Bewegung hervor.		
Delphi:	function LSX_GetMoveSeqVar (LSID: Integer; var a, var b, var c, var d, var e: Double): Integer;		
C++:	int GetMoveSeqVar (double *pda, double *pdb, double *pdc, double *pdd, double *pde);		
LabView:	Nicht unterstützt		
Parameter:	a, b, c, d, e	Werte aller Variablen (a bis e)	
Beispiel:	LSX.GetMoveSeqVar (&a, &b, &c, &d, &e);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?msv	-

LSX_SetMoveSeqVar			
Beschreibung:	Befehl zum Einstellen von Variablenwerten, um Bewegungsabläufe in ihren Laufeigenschaften beeinflussen zu können. Wird vor Beginn des Programms kein Wert gesetzt, werden Standardwerte verwendet. Diese rufen bei Standardprogrammen keine Bewegung hervor.		
Delphi:	function LSX_SetMoveSeqVar (LSID: Integer; a, b, c, d, e: Double): Integer;		
C++:	int SetMoveSeqVar (double da, double db, double dc, double dd, double de);		
LabView:	Nicht unterstützt		
Parameter:	a, b, c, d, e	Werte aller Variablen (a bis e)	
Beispiel:	LSX.SetMoveSeqVar (5, 0, 2, 4, 1);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!msv	MoveSeqStatus

4.2.10 Tabellenbefehle

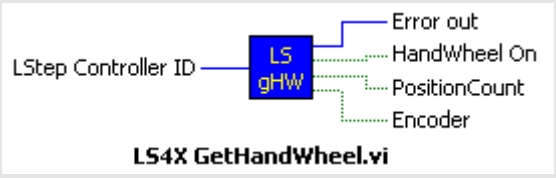
LSX_GetTablePos			
Beschreibung:	Befehl zum Auslesen von Tabellenpositionen. Es können bis zu 1000 Werte für jede Achse gespeichert werden. Das Anfahren der Tabellenpositionen erfolgt mit den Befehlen 'MoveTablePosAbs' und 'MoveTablePosRel'		
Delphi:	function LSX_GetTablePos (LSID: Integer; var position: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetTablePos (int lposition; double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	Position, X, Y, Z, A	Tabellenposition und zugehörige Positionswerte	
Beispiel:	LSX.GetTablePos (5, &X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?tpos	-

LSX_SetTablePos			
Beschreibung:	Befehl zum Einstellen von Tabellenpositionen. Es können bis zu 1000 Werte für jede Achse gespeichert werden. Das Anfahren der Tabellenpositionen erfolgt mit den Befehlen 'MoveTablePosAbs' und 'MoveTablePosRel'		
Delphi:	function LSX_SetTablePos (LSID: Integer; var position: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetTablePos (int lposition, double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	Position, X, Y, Z, A	Tabellenposition und zugehörige Positionswerte	
Beispiel:	LSX.SetTablePos (5, 10, 7.5, 210, 25.7);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!tpos	Sofort

LSX_MoveTablePosAbs, LSX_MoveTablePosAbsW			
Beschreibung:	Fährt die in der übergebenen Tabellenposition gespeicherten Positionswerte der einzelnen Achsen absolut an		
Delphi:	function LSX_MoveTablePosAbs(LSID: Integer; X: Integer; Y: Integer; Z: Integer; A: Integer; var Feedback: PAnsiChar; MaxLen: Integer): Integer; function LSX_MoveTablePosAbsW(LSID: Integer; X: Integer; Y: Integer; Z: Integer; A: Integer; var Feedback: PWideChar; MaxLen: Integer): Integer;		
C++:	int MoveTablePosAbs (int lX, int lY, int lZ, int lA, char *pcFeedback, int lMaxLen); int MoveTablePosAbsW (int lX, int lY, int lZ, int lA, TCHAR *pcFeedback, int lMaxLen);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Verfahrensbereich jeder Achse	
Rückmeldung:	Feedback	Für jede Achse nach Erreichen: @	
Beispiel:	LSX.MoveTablePosAbs (5, 2, 7, 81, pcFeedback,256);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!mtpa	Sofort

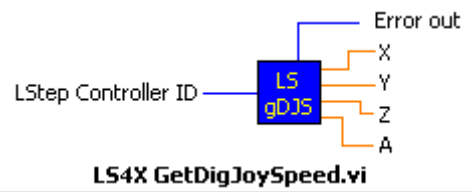
LSX_ MoveTablePosRel			
Beschreibung:	Verfährt die Achse(n) um die in der übergebenen Tabellenposition gespeicherten Werte relativ zur Startposition		
Delphi:	function LSX_MoveTablePosRel(LSID: Integer; X: Integer; Y: Integer; Z: Integer; A: Integer; var Feedback: PAnsiChar; MaxLen: Integer): Integer; function LSX_MoveTablePosRelW(LSID: Integer; X: Integer; Y: Integer; Z: Integer; A: Integer; var Feedback: PWideChar; MaxLen: Integer): Integer;		
C++:	int MoveTablePosRel (int lX, int lY, int lZ, int lA, char *pcFeedback, int lMaxLen); int MoveTablePosRelW (int lX, int lY, int lZ, int lA, TCHAR *pcFeedback, int lMaxLen);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Verfahrensbereich jeder Achse	
Rückmeldung:	Feedback	Für jede Achse nach Erreichen: @	
Beispiel:	LSX.MoveTablePosRel (5, 2, 7, 81, pcFeedback,256);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!mtpr	Sofort

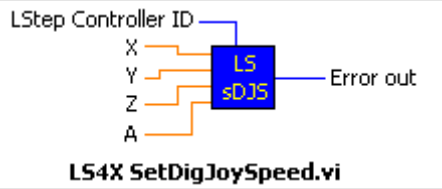
4.2.11 Joystick und Handrad

LSX_GetHandWheel			
Beschreibung:	Funktion zum Auslesen des Handradzustandes. (nicht für LSTEP express)		
Delphi:	function LSX_GetHandWheel(LSID: Integer; var PositionCount, Encoder: Long-Bool): Integer;		
C++:	int GetHandWheel(BOOL *pbHandWheelOn, BOOL *pbPositionCount, BOOL *pbEncoder);		
LabView:			
Parameter:	HWOn	Handrad aktiv True = Handrad ist eingeschaltet False = Handrad ist ausgeschaltet	
	PosCount	Positionszählung aktiv True = Positionszählung ist eingeschaltet True = Positionszählung ist ausgeschaltet	
	Encoder	Encoderwerte werden für Positionszählung verwendet wenn vorhanden True = Geberwerte werden verwendet False = Geberwerte werden nicht verwendet	
Beispiel:	LSX.GetHandWheel(&HWOn, &PosCount, &Encoder);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?hw	-

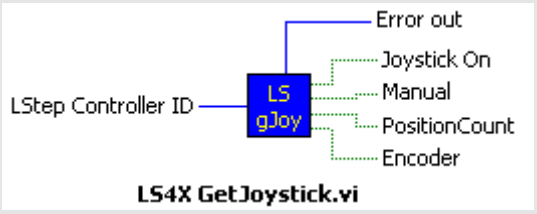
LSX_SetHandWheelOff			
Beschreibung:	Funktion schaltet das Handrad aus (nicht für LSTEP express)		
Delphi:	function LSX_SetHandWheelOff(LSID: Integer): Integer;		
C++:	int SetHandWheelOff();		
LabView:			
Parameter:	-		
Beispiel:	LSX.SetHandWheelOff();		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!hw	-

LSX_SetHandWheelOn			
Beschreibung:	Funktion schaltet das Handrad ein. (nicht für LSTEP express)		
Delphi:	function LSX_SetHandWheelOn(LSID: Integer; PositionCount, Encoder: Long-Bool): Integer;		
C++:	int SetHandWheelOn(BOOL fPositionCount,BOOL fEncoder);		
LabView:			
Parameter:	PositionCount	Schaltet die Positionszählung ein bzw aus True = Ein False = Aus	
	Encoder	Geberwerte werden falls vorhanden für die Positionszählung verwendet True = Geberwerte verwenden False = Keine Geberwerte verwenden	
Beispiel:	LSX.SetHandWheelOn(true, true); // Handrad Ein mit Positionszählung (Geberwerte)		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!hw	-

LSX_GetDigJoySpeed			
Beschreibung:	Auslesen der eingestellten Geschwindigkeit für das Fahren mit einer konstanten Geschwindigkeit (nicht für LSTEP express)		
Delphi:	function LSX_GetDigJoySpeed(LSID: Integer; var dX, dY, dZ, dA: Double): Integer;		
C++:	int GetDigJoySpeed(double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	dX, dY, dZ, dA	Geschwindigkeitswerte in U/s	
Beispiel:	LSX.GetDigJoySpeed(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?speed	-

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, kann der digitale Joystick mit der Funktion LSX_SetDigJoyOff deaktiviert werden. (nicht für LSTEP express)		
Delphi:	function LSX_SetDigJoySpeed(LSID: Integer; dX, dY, dZ, dA: Double): Integer;		
C++:	int SetDigJoySpeed (double dX, double dY, double dZ, double dA);		
LabView:			
Parameter:	dX, dY, dZ, dA	Geschwindigkeit in U/s	
Beispiel:	LSX.SetDigJoySpeed(0, 10.0, 25.0, 0); //Achsen X und A – Geschwindigkeit 0 und Joystick-Betrieb „Aus", Achse Y – Geschwindigkeit 10.0 U/s und Joystick-Betrieb „Ein", Achse Z – Geschwindigkeit 25.0 U/s und Joystick-Betrieb „Ein".		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!speed	-

LSX_SetDigJoyOff			
Beschreibung:	Schaltet digitalen Joystick aus. (nicht für LSTEP express)		
Delphi:	function LSX_SetDigJoyOff(LSID: Integer): Integer;		
C++:	int SetDigJoyOff() ;		
LabView:			
Parameter:	-		
Beispiel:	LSX.SetDigJoyOff() ;		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!speed	-

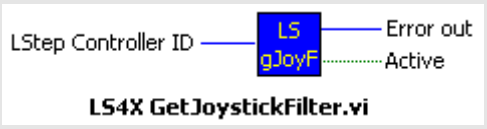
LSX_GetJoystick		
Beschreibung:	Abfrage des aktuellen Zustands vom Analog-Joystick.	
Delphi:	function LSX_GetJoystick(LSID: Integer; var JoystickOn, Manual, PositionCount, Encoder: LongBool): Integer;	
C++:	int GetJoystick(BOOL *pbJoystickOn, BOOL *pbManual, BOOL *pbPositionCount, BOOL *pbEncoder);	
LabView:		
Parameter:	JoyOn	Joystick Aktiv True = Joystick ist eingeschaltet False = Joystick ist ausgeschaltet
	Manual	Aktivierungsart des manuellen Modus (nicht für LSTEP express) True = Joystick ist manuell über Schalter eingeschaltet False = Joystickschalter steht auf Automatik
	PosCount	Positionszählung aktiv (nicht für LSTEP express) True = Positionszählung ist eingeschaltet False = Positionszählung ist ausgeschaltet
	Enc	Encoderwerte werden für Positionszählung verwendet wenn vorhanden (nicht für LSTEP express) True = Geberwerte werden verwendet False = Geberwerte werden nicht verwendet

Beispiel:	LSX.GetJoystick(&JoyOn, &Manual, &PosCount, &Enc);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?joy	-

LSX_SetJoystickOff			
Beschreibung:	Funktion schaltet den Analog-Joystick aus		
Delphi:	function LSX_SetJoystickOff(LSID: Integer): Integer;		
C++:	int SetJoystickOff();		
LabView:			
Parameter:	-		
Beispiel:	LSX.SetJoystickOff();		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!joy	-

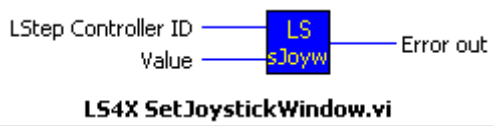
LSX_SetJoystickOn		
Beschreibung:	Funktion schaltet den Analog-Joystick ein.	
Delphi:	function LSX_SetJoystickOn(LSID: Integer; PositionCount, Encoder: LongBool): Integer;	
C++:	int SetJoystickOn(BOOL PositionCount,BOOL Encoder);	
LabView:		
Parameter:	PositionCount	Schaltet die Positionszählung ein bzw. aus True = Ein False = Aus
	Encoder	Geberwerte werden falls vorhanden für die Positionszählung verwendet (nicht für LSTEP express) True = Geberwerte verwenden False = Keine Geberwerte verwenden
Beispiel:	LSX.SetJoystickOn(true, true); // Joystick Ein mit Positionszählung (Geberwerte)	

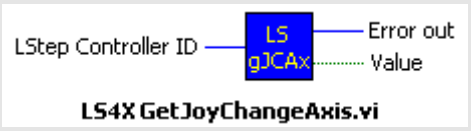
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!joy	-

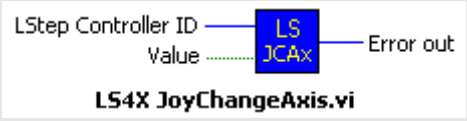
LSX_GetJoystickFilter			
Beschreibung:	Gibt an, ob die Filterung und Hysterese im Joystick-Betrieb aktiviert ist. (nicht für LSTEP express)		
Delphi:	function LSX_GetJoystickFilter(LSID: Integer; var bActive: LongBool): Integer;		
C++:	int GetJoystickFilter(BOOL *pbActive);		
LabView:			
Parameter:	bActive	Aktuell eingestellter Wert True = Filterung aktiviert False = Filterung deaktiviert	
Beispiel:	LSX.GetJoystickFilter(&Active);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?joyfilter	-

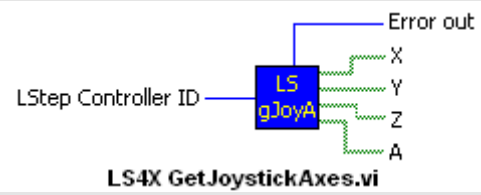
LSX_SetJoystickFilter			
Beschreibung:	Aktivierung/Deaktivierung der Filterung und Hysterese im Joystick-Betrieb. (nicht für LSTEP express)		
Delphi:	function LSX_SetJoystickFilter(LSID: Integer; bActive: LongBool): Integer;		
C++:	int SetJoystickFilter(BOOL bActive);		
LabView:			
Parameter:	bActive	Aktivierung des Filters True = Filter aktivieren False = Filter deaktivieren	
Beispiel:	LSX.SetJoystickFilter(True);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!joyfilter	-

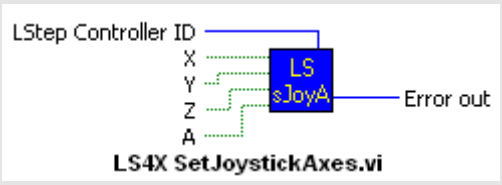
LSX_GetJoystickWindow			
Beschreibung	Funktion zum Auslesen des Joystick-Fensters. Mit dem Joystickfenster wird ein Analogbereich definiert in dem die Achsen sich nicht bewegen.		
Delphi	function LSX_GetJoystickWindow(LSID: Integer; var AValue: Integer): Integer;		
C++	int GetJoystickWindow(int *plAValue);		
LabView:			
Parameter	AValue	Analogbereich , in dem sich die Achsen nicht bewegen.	
Beispiel	LSX.GetJoystickWindow(&AValue) ;		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?joywindow	-

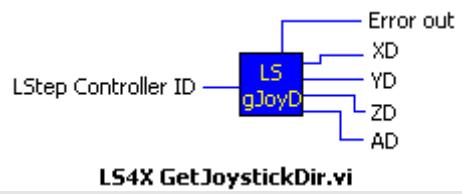
LSX_SetJoystickWindow			
Beschreibung	Funktion zum Setzen des Joystick-Fensters. Mit dem Joystickfenster wird ein Analogbereich definiert in dem die Achsen sich nicht bewegen.		
Delphi	function LSX_SetJoystickWindow(LSID: Integer; AValue: Integer): Integer;		
C++	int SetJoystickWindow(int lAValue);		
LabView:			
Parameter	AValue	Analogbereich , in dem sich die Achsen nicht bewegen.	
Beispiel	LSX.SetJoystickWindow(20) ;		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!joywindow	SetJoystickOn

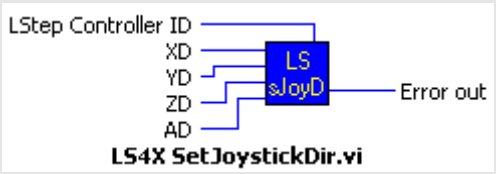
LSX_GetJoyChangeAxis			
Beschreibung:	Liest Joystick-Achszuordnung aus (nicht für LSTEP express)		
Delphi:	LSX_GetJoyChangeAxis(LSID: Integer; var Value: LongBool): Integer;		
C++:	int GetJoyChangeAxis(BOOL *pbValue);		
LabView:			
Parameter:	Value	Achszuordnung True = Konventionelle Joystickauswertung False = X- und Y-Achszuordnung ist getauscht	
Beispiel:	LSX.GetJoyChangeAxis(&Value);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?joychangeaxis	-

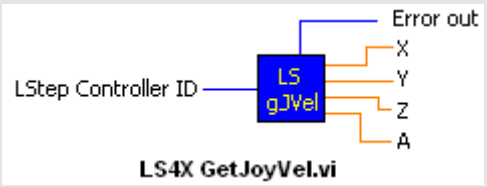
LSX_JoyChangeAxis			
Beschreibung:	Setzt Joystick-Achszuordnung. (nicht für LSTEP express)		
Delphi:	LSX_JoyChangeAxis(LSID: Integer; Value: LongBool): Integer;		
C++:	int JoyChangeAxis(BOOL bValue);		
LabView:			
Parameter:	Value	Achszuordnung True = Konventionelle Joystickauswertung False = X- und Y-Achszuordnung ist getauscht	
Beispiel:	LSX.JoyChangeAxis(true);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!joychangeaxis	-

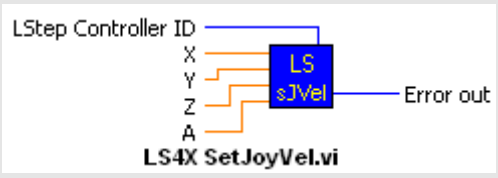
LSX_GetJoystickAxes			
Beschreibung:	Zeigt, für welche Achsen der Joystick aktiv ist, wenn der Joystickbetrieb eingeschaltet wird. (nur für LSTEP express)		
Delphi:	function LSX_GetJoystickAxes(LSID: Integer; var Flags: Integer): Integer;		
C++:	int GetJoystickAxes(int *plFlags);		
LabView:			
Parameter:	Flags	Integer, welcher nach Aufruf der Funktion in den Bits 0-4 die Bit-Maske enthält. Bit 0 = X-Achse Bit 1 = Y-Achse … Wert 0 = Joystick bleibt ausgeschaltet Wert 1 = Joystick wird eingeschaltet	
Beispiel:	LSX.GetJoystickAxes(&Flags);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?joyenable	-

<table>
<tr><th colspan="4">LSX_SetJoystickAxes</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Erlaubt den Joystick-Betrieb für die angegebenen Achsen.
(nur für LSTEP express)</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_SetJoystickAxes(LSID: Integer; Flags: Integer): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int SetJoystickAxes(int Flags);</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td>Parameter:</td><td>Flags</td><td colspan="2">Bit-Maske mit den Joystickachsen die beim Einschalten des Joysticks aktiviert werden sollen.
Bit 0 = X-Achse
Bit 1 = Y-Achse
…
Wert 0 = Joystick bleibt ausgeschaltet
Wert 1 = Joystick wird eingeschaltet</td></tr>
<tr><td>Beispiel:</td><td colspan="3">LSX.SetJoystickAxes(3);
/* X- und Y-Achse – Joystick eingeschaltet (Bits 0 u. 1 gesetzt), Z- und A-Achse – Joystick „Aus“ (Bit 2 = 0) */</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>2</td><td>!joyenable</td><td>SetJoystickOn</td></tr>
</table>

LSX_GetJoystickDir			
Beschreibung:	Liest Motordrehrichtung für Joystick aus.		
Delphi:	function LSX_GetJoystickDir(LSID: Integer; var XD, YD, ZD, AD: Integer): Integer;		
C++:	int GetJoystickDir(int *plXD, int *plYD, int *plZD, int *plRD);		
LabView:			
Parameter:	X, Y, Z, A	Drehrichtung des Motors **LSTEP 2000 Serie:** 0 = Achse gesperrt 1 = Positive Drehrichtung -1 = Negative Drehrichtung 2 = Positive Drehrichtung mit Stromreduzierung -2 = Negative Drehrichtung mit Stromreduzierung **LSTEP express Serie:** 0 = Normale Drehrichtung 1 = Drehrichtung umkehren	
Beispiel:	LSX.GetJoystickDir(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?joydir	-

<table>
<tr><th colspan="4">LSX_SetJoystickDir</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Drehrichtung des Joystick setzen.</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_SetJoystickDir(LSID: Integer; XD, YD, ZD, AD: Integer): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int SetJoystickDir(int lXD,int lYD,int lZD,int lAD);</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td>Parameter:</td><td>X, Y, Z, A</td><td colspan="2">Drehrichtung des Motors
LSTEP 2000 Serie:
0 = Achse gesperrt
1 = Positive Drehrichtung
-1 = Negative Drehrichtung
2 = Positive Drehrichtung mit Stromreduzierung
-2 = Negative Drehrichtung mit Stromreduzierung
LSTEP express Serie:
0 = Normale Drehrichtung
1 = Drehrichtung umkehren</td></tr>
<tr><td>Beispiel:</td><td colspan="3">LSX.SetJoystickDir(1, 1, -1, 0);
/* X- und Y-Achse positive Drehrichtung; Z-Achse negative Drehrichtung; A-Achse gesperrt */</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>3</td><td>!joydir</td><td>SetJoystickOn</td></tr>
</table>

LSX_GetJoyVel			
Beschreibung:	Die maximalen Verfahrgeschwindigkeiten im Joystickbetrieb abfragen. (nur für LSTEP express)		
Delphi:	function LSX_GetJoyVel(LSID: Integer; var XD, YD, ZD, AD: Double): Integer;		
C++:	int GetJoyVel(double *pdXD, double *pdYD, double *pdZD, double *pdAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Geschwindigkeitswerte in der eingestellten Dimension/s	
Beispiel:	LSX.GetJoyVel(&XD, &YD, &ZD, &AD);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?joyvel	-

LSX_SetJoyVel			
Beschreibung:	Die maximalen Verfahrgeschwindigkeiten im Joystickbetrieb einstellen (nur für LSTEP express)		
Delphi:	function LSX_SetJoyVel(LSID: Integer; XD, YD, ZD, AD: Double): Integer;		
C++:	int SetJoyVel(double dXD, double dYD, double dZD, double dAD);		
LabView:			
Parameter:	XD, YD, ZD, AD	Geschwindigkeitswerte in der eingestellten Dimension/s	
Beispiel:	LSX.SetJoyVel(1.0, 15.0, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!joyvel	SetJoystickOn

LSX_GetJoyRedcur			
Beschreibung:	Abfragen, ob die Stromreduzierung im Joystickbetrieb aktiv ist. Bei aktiver Stromreduzierung wird bei Joystick in Ruhestellung Motorstrom auf den Befehl „reduction“ eingestellten Wert abgesenkt.		
Delphi:	function LSX_GetJoyRedCur (LSID: Integer;var X,Y,Z,A: LongBool):Integer;		
C++:	int GetJoyRedCur (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Zustand der Stromreduzierung	
Beispiel:	LSX.GetJoyRedCur(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	?joyredcur	-

LSX_SetJoyRedcur			
Beschreibung:	Befehl zum Ein- und Ausschalten der Stromreduzierung im Joystickbetrieb. Bei aktiver Stromreduzierung wird bei Joystick in Ruhestellung Motorstrom auf den Befehl „reduction“ eingestellten Wert abgesenkt.		
Delphi:	function LSX_SetJoyRedCur (LSID: Integer;X,Y,Z,R: LongBool):Integer; function LSX_SetJoyRedCur (LSID: Integer; Axis:Integer; Reduce:Boolean): Integer;		
C++:	int SetJoyRedCur (BOOL bX, BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Zustand der Stromreduzierung	
Beispiel:	LSX.SetJoyRedCur(True,True,True,True);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	!joyredcur	SetJoystickOn

LSX_GetJoytoAxis			
Beschreibung:	Liest die Zuordnung der Joystick-Eingänge 1 bis 4 zu den machanischen achsen X, Y, Z und A aus.		
Delphi:	function LSX_GetJoytoAxis(LSID: Integer; var X,Y,Z,A: Integer): Integer;		
C++:	int GetJoytoAxis (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Zugeordneter Joystick-Eingang	
Beispiel:	LSX.GetJoytoAxis(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?joytoaxis	-

LSX_SetJoytoAxis			
Beschreibung:	Ordnet die Joystick-Eingänge 1 bis 4 zu den machanischen Achsen X, Y, Z und A zu.		
Delphi:	function LSX_SetJoytoAxis(LSID: Integer; X,Y,Z,A: Integer): Integer;		
C++:	int SetJoytoAxis (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X,Z,Y,A	Zugeordneter Joystick-Eingang	
Beispiel:	LSX.SetJoytoAxis (1,2,3,4);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!joytoaxis	SetJoystickOn

LSX_GetManModePreselection			
Beschreibung	Faktor für den Trackball des Bedienpults auslesen.		
Delphi	function LSX_GetManModePreselection(LSID: Integer;var MMode: Integer): Integer;		
C++	int GetManModePreselection (int *plMMode);		
LabView:	-		
Parameter	0 = Kein manueller Mode		
	1 = Joystickbetrieb		
	2 = Tippbetrieb		
	3 = Trackball		
Beispiel	LSX.GetManmodePreselection(&MMode) ;		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?manmodepreselection	-

LSX_SetManModePreselection			
Beschreibung	Befehl zum Auswählen der gewünschten manuellen Betriebsart beim Einschalten mittels digitalen Eingang.		
Delphi	function LSX_SetManModePreSelection(LSID:Integer;MMode: Integer): Integer;		
C++	int SetManModePreselection (int lMMode);		
LabView:	-		
Parameter	0 = Kein manueller Mode		
	1 = Joystickbetrieb		
	2 = Tippbetrieb		
	3 = Trackball		
Beispiel	LSX.SetManModePreselection(0) ;		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	! manmodepreselection	-

LSX_GetManModeLinktoAxis			
Beschreibung	Fragt ab, welche Achse an die ausgewählte Achse gekoppelt ist. Die Gekoppelte Achse wird im manuellen Betrieb mit den Parametern (für manuelle Bedienung) die Achse, an die gekoppelt wurde verfahren. Mehrfachkopplungen sind nicht zulässig und werden automatisch bei neuer Zuordnung aufgehoben. Erreicht eine der beteiligten Achsen ihre Verfahrbewegungsgrenze werden beide gestoppt. Eine Kopplung von Servoachsen mit Schrittmotorachsen und umgekehrt ist nicht implementiert.		

Delphi	function LSX_GetManModeLinktoAxis (LSID: Integer;Axis: Integer;var Link2: AnsiChar): Integer;
	function LSX_GetManModeLinktoAxisX(LSID: Integer;var Link2: AnsiChar): Integer;
	function LSX_GetManModeLinktoAxisY(LSID: Integer;var Link2: AnsiChar): Integer;
	function LSX_GetManModeLinktoAxisZ(LSID: Integer;var Link2: AnsiChar): Integer;
	function LSX_GetManModeLinktoAxisA(LSID: Integer;var Link2: AnsiChar): Integer;		

C++	int GetManModeLinktoAxis (int lAxis, char *pcLink2); int GetManModeLinktoAxisX (char *pcLink2); int GetManModeLinktoAxisY (char *pcLink2); int GetManModeLinktoAxisZ (char *pcLink2); int GetManModeLinktoAxisA (char *pcLink2);		
LabView:	-		
Parameter	Achse, die gekoppelt werden soll:	Achse, an die gekoppelt werden soll:	
	1=X, 2=Y, 3=Z, 4=A	X Y Z A	
Beispiel	LSX.GetManmodeLinktoAxis(&Link2) ;		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?manmodelinktoaxis	-

LSX_SetManModeLinktoAxis			
Beschreibung	Koppelt den manuellen Betrieb einer Achse an eine andere Achse. Die Gekoppelte Achse wird im manuellen Betrieb mit den Parametern (für manuelle Bedienung) die Achse, an die gekoppelt wurde verfahren. Mehrfachkopplungen sind nicht zulässig und werden automatisch bei neuer Zuordnung aufgehoben. Erreicht eine der beteiligten Achsen ihre Verfahrbewegungsgrenze werden beide gestoppt. Eine Kopplung von Servoachsen mit Schrittmotorachsen und umgekehrt ist nicht implementiert.		
Delphi	function LSX_SetManModeLinktoAxis(LSID: Integer; Link1,Link2: Char): Integer;		
C++	int SetManModeLinktoAxis (char cLink1, char cLink2);		
LabView:	-		
Parameter	Achse, die gekoppelt werden soll:	Achse, an die gekoppelt werden soll:	
	x,y,z,a	x,y,z,a	
Beispiel	LSX.SetManModeLinktoAxis (x,y) ;		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	! manmodelinktoaxis	-

LSX_GetTipp			
Beschreibung:	Abfragen, ob der Tippbetrieb aktiv ist.		
Delphi:	function LSX_GetTipp (LSID: Integer;var Tipp: LongBool): Integer;		
C++:	int GetTipp (BOOL *pbTipp);		
LabView:	Nicht unterstützt		
Parameter:	Zustand des Tippbetriebes		
Beispiel:	LSX.GetTipp(&Tipp);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?tipp	-

LSX_SetTipp			
Beschreibung:	Aktivieren/ Deaktivieren des Tippbetriebes.		
Delphi:	function LSX_SetTipp (LSID: Integer;Tipp: LongBool): Integer;		
C++:	int SetTipp (BOOL bTipp);		
LabView:	Nicht unterstützt		
Parameter:	Zustand des Tippbetriebes		
Beispiel:	LSX.SetTipp(True);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!tipp	-

LSX_GetTippEnable			
Beschreibung:	Auslesen welche Achsen für den Tippbetrieb freigegeben sind		
Delphi:	function LSX_GetTippEnable (LSID: Integer;var X,Y, Z,A: LongBool): Integer;		
C++:	int GetTippEnable (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A		Achsenfreigabe
Beispiel:	LSX.GetTippEnable(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?tippenable	-

LSX_SetTippEnable			
Beschreibung:	Achsen für den Tippbetrieb freigeben/sperren.		
Delphi:	function LSX_SetTippEnable (LSID: Integer;X,Y,Z,A: LongBool): Integer;		
C++:	int SetTipp (BOOL bTipp);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A		Achsenfreigabe
Beispiel:	LSX.SetTippEnable(True,True,True,True);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!tippenable	SetTipp

LSX_GetTippRedCur			
Beschreibung:	Auslesen, ob die Stromreduzierung im Tippbetrieb für die Achsen aktiv ist. Bei aktivier Stromreduzierung und alle Achsen im Stillstand wird der motorstrom auf den mit dem Befehl „reduction“ eingestellten Wert abgesenkt.		
Delphi:	function LSX_GetTippRedCur (LSID: Integer; var X,Y, Z,A: LongBool): Integer;		
C++:	int GetTippRedCur (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Stromreduzierung Aktiviert/Deaktiviert	
Beispiel:	LSX.GetTippRedCur(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	?tippredcur	-

LSX_SetTippRedCur			
Beschreibung:	Ein- und Ausschalten der Stromreduzierung im Tippbetrieb. Bei aktiver Stromreduzierung und allen Achsen im Stillstand wird der Motorstrom auf den mit dem Befehl „reduction“ eingestellten Wert abgesenkt.		
Delphi:	function LSX_SetTippRedCur (LSID: Integer;X,Y,Z,A: LongBool): Integer;		
C++:	int SetTippRedCur (BOOL bX, BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Stromreduzierung Aktiviert/Deaktivert	
Beispiel:	LSX.SetTippRedCur(True,True,True,True);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	!tippredcur	SetTipp

LSX_GetTippVel			
Beschreibung:	Liest die Verfahrgeschwindigkeit im Tippbetrieb aus.		
Delphi:	function LSX_GetTippVel(LSID: Integer;var X,Y, Z,A: Double): Integer;		
C++:	int GetTippVel (double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Verfahrgeschwindigkeit	
Beispiel:	LSX.GetTippVel(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	?tippvel	-

LSX_SetTippVel			
Beschreibung:	Stellt die Verfahrgeschwindigkeit im Tippbetrieb ein.		
Delphi:	function LSX_SetTippVel(LSID: Integer;X,Y,Z,A: Double): Integer;		
C++:	int SetTippVel (double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A		Verfahrgeschwindigkeit
Beispiel:	LSX.SetTippVel(10,10 ,5.5 ,10);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!tippvel	SetTipp

LSX_GetTippDir			
Beschreibung:	Liest die Verfahrrichtung der Tipp-Eingänge aus.		
Delphi:	function LSX_GetTippDir (LSID: Integer;var X,Y,Z,A: LongBool): Integer;		
C++:	int GetTippDir (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A		Verfahrrichtung
Beispiel:	LSX.GetTippDir(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?tippdir	-

LSX_SetTippDir			
Beschreibung:	Stellt die Verfahrrichtung im Tippbetrieb ein.		
Delphi:	function LSX_SetTippDir (LSID: Integer;X,Y,Z,A: LongBool): Integer;		
C++:	int SetTippDir (BOOL bX, BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A		Verfahrrichtung
Beispiel:	LSX.SetTippDir(True,True,True ,False);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!tippdir	SetTipp

LSX_GetTippOutPass			
Beschreibung:	Befehl zum Auslesen der Filterzeitkonstante im Tippbetrieb. Der Filter verhindert ruckartige Veränderungen der Geschwindigkeit (Rampenfunktion) bei Aktivierung der Tipp-Eingänge.		
Delphi:	function LSX_GetTippOutPass (LSID: Integer;var X,Y,Z,A: Integer): Integer;		
C++:	int GetTippOutPass (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Filterzeitkonstante	
Beispiel:	LSX.GetTippOutPass (&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?tippoutpass	-

LSX_SetTippOutPass			
Beschreibung:	Befehl zum Einstellen der Filterzeitkonstante im Tippbetrieb. Der Filter verhindert ruckartige Veränderungen der Geschwindigkeit (Rampenfunktion) bei Aktivierung der Tipp-Eingänge.		
Delphi:	function LSX_SetTippOutPass (LSID: Integer;X,Y,Z,A: Integer): Integer;		
C++:	int SetTippOutPass (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Filterzeitkonstante	
Beispiel:	LSX.SetTippOutPass (20,20,10 ,10);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!tippoutpass	SetTipp

LSX_GetTrackBall			
Beschreibung:	Befehl zum Auslesen des Status des Trackballbetriebs		
Delphi:	function LSX_GetTrackBall (LSID: Integer; var status: LongBool): Integer;		
C++:	int GetTrackBall (BOOL *pbstatus);		
LabView:	Nicht unterstützt		
Parameter:	status	Einstellung: An/Aus	
Beispiel:	LSX.GetTrackBall (&status);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?tb	-

LSX_SetTrackBall			
Beschreibung:	Befehl zum Ein- und Ausschalten des Trackballbetriebs		
Delphi:	function LSX_SetTrackBall (LSID: Integer; status: LongBool): Integer;		
C++:	int SetTrackBall (BOOL bstatus);		
LabView:	Nicht unterstützt		
Parameter:	status	Einstellung: An/Aus	
Beispiel:	LSX.SetTrackBall (True);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!tb	Sofort

LSX_GetTrackBallEnable			
Beschreibung:	Befehl zum Auslesen des Status des Trackballbetriebs		
Delphi:	function LSX_GetTrackBallEnable (LSID: Integer; var X, Y, Z, A: LongBool): Integer;		
C++:	int GetTrackBallEnable (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einstellung pro Achse: An/Aus	
Beispiel:	LSX.GetTrackBallEnable (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?tbenable	-

LSX_SetTrackBallEnable			
Beschreibung:	Befehl zum Ein- und Ausschalten des Trackballbetriebs		
Delphi:	function LSX_SetTrackBallEnable (LSID: Integer; X, Y, Z, A: LongBool): Integer;		
C++:	int SetTrackBallEnable (BOOL bX, BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einstellung pro Achse: An/Aus	
Beispiel:	LSX.SetTrackBallEnable (True, True, True, True);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!tbenable	TrackBall

LSX_GetTrackBallRedCur			
Beschreibung:	Befehl zum Einlesen des Status der Stromreduzierung im Trackballbetrieb. Bei aktiver Stromreduzierung und allen Achsen im Stillstand wird der Motorstrom auf den mit Befehl "reduction" eingestellten Wert abgesenkt.		
Delphi:	function LSX_GetTrackBallRedCur (LSID: Integer; var X, Y, Z, A: LongBool): Integer;		
C++:	int GetTrackBallRedCur (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einstellung pro Achse: An/Aus	
Beispiel:	LSX.GetTrackBallRedCur (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?tbredcur	-

LSX_SetTrackBallRedCur			
Beschreibung:	Befehl zum Ein- und Ausschalten der Stromreduzierung im Trackballbetrieb. Bei aktiver Stromreduzierung und allen Achsen im Stillstand wird der Motorstrom auf den mit Befehl "reduction" eingestellten Wert abgesenkt.		
Delphi:	function LSX_SetTrackBallRedCur (LSID: Integer; X, Y, Z, A: LongBool): Integer;		
C++:	int SetTrackBallRedCur (BOOL bX, BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einstellung pro Achse: An/Aus	
Beispiel:	LSX.SetTrackBallRedCur (True, True, True, True);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!tbredcur	TrackBall

LSX_GetTrackBallVel			
Beschreibung:	Befehl zum Auslesen der maximalen Verfahrgeschwindigkeiten im Trackballbetrieb.		
Delphi:	function LSX_GetTrackBallVel (LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetTrackBallVel (double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Geschwindigkeit pro Achse	
Beispiel:	LSX.GetTrackBallVel (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?tbvel	-

LSX_SetTrackBallVel			
Beschreibung:	Befehl zum Einstellen der maximalen Verfahrgeschwindigkeiten im Trackballbetrieb.		
Delphi:	function LSX_SetTrackBallVel (LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetTrackBallVel (double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Geschwindigkeit pro Achse	
Beispiel:	LSX.SetTrackBallVel (10, 10, 10, 10);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!tbvel	TrackBall

LSX_GetTrackBallOutPass			
Beschreibung:	Befehl zum Auslesen der Filterzeitkonstante im Trackballbetrieb. Der Filter verhindert ruckartige Veränderungen der Geschwindigkeit (Rampenfunktion) bei Drehen des Trackballs.		
Delphi:	function LSX_GetTrackBallOutPass (LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetTrackBallOutPass (double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Filterzeitkonstante pro Achse, Bereich: 0 bis 500 000 µs	
Beispiel:	LSX.GetTrackBallOutPass (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?tboutpass	-

LSX_SetTrackBallOutPass			
Beschreibung:	Befehl zum Einstellen der Filterzeitkonstante im Trackballbetrieb. Der Filter verhindert ruckartige Veränderungen der Geschwindigkeit (Rampenfunktion) bei Drehen des Trackballs.		
Delphi:	function LSX_SetTrackBallOutPass (LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetTrackBallOutPass (double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Filterzeitkonstante pro Achse, Bereich: 0 bis 500 000 µs	
Beispiel:	LSX.SetTrackBallOutPass (20, 20, 20, 20);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!tboutpass	TrackBall

LSX_GetTrackBallDir			
Beschreibung:	Befehl zum Auslesen der Verfahrrichtung zur Drehrichtung des Trackballs.		
Delphi:	function LSX_GetTrackBallDir (LSID: Integer; var X, Y, Z, A: LongBool): Integer;		
C++:	int GetTrackBallDir (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einstellung pro Achse: An/Aus	
Beispiel:	LSX.GetTrackBallDir (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?tbdir	-

LSX_SetTrackBallDir			
Beschreibung:	Befehl zum Einstellen der Verfahrrichtung zur Drehrichtung des Trackballs.		
Delphi:	function LSX_SetTrackBallDir (LSID: Integer; X, Y, Z, A: LongBool): Integer;		
C++:	int SetTrackBallDir (BOOL bX, BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einstellung pro Achse: An/Aus	
Beispiel:	LSX.SetTrackBallDir (True, True, True¸ True);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!tbdir	TrackBall

LSX_GetTrackBallToAxis			
Beschreibung:	Befehl zum Auslesen der Zuordnung der Trackballachsen (horizontal und vertikal) zu den mechanischen Achsen X, Y, Z und A.		
Delphi:	function LSX_GetTrackBallToAxis (LSID: Integer;var X,Y,Z,A: Integer): Integer;		
C++:	int GetTrackBallToAxis (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Zuordnung: 0: Keine Achszuordnung 1: Horizontale Trackballachse 2. Vertikale Trackballachse	
Beispiel:	LSX.GetTrackBallToAxis (&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?tbtoaxis	-

LSX_SetTrackBallToAxis			
Beschreibung:	Befehl zum Zuordnen der Trackballachsen (horizontal und vertikal) zu den mechanischen Achsen X, Y, Z und A.		
Delphi:	function LSX_SetTrackBallToAxis (LSID: Integer;X,Y,Z,A: Integer): Integer;		
C++:	int SetTrackBallToAxis (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Zuordnung: 0: Keine Achszuordnung 1: Horizontale Trackballachse 2. Vertikale Trackballachse	
Beispiel:	LSX.SetTippOutPass (2,1,0,0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!tbtoaxis	Trackball

4.2.12 Bedienpult mit Trackball und Joyspeed-Tasten

LSX_GetBPZ			
Beschreibung:	Liest den Zustand des Zusatzbedienpults mit Trackball ein. (nicht für LSTEP express)		
Delphi:	function LSX_GetBPZ(LSID: Integer; var AValue: Integer): Integer;		
C++:	int GetBPZ(int *plAValue);		
LabView:	-		
Parameter:	AValue	Aktivität des Bedienpults 0 = Bedienpult ist „Aus“. 1 = Bedienpult aktiv und der Trackball wird mit 0,1µ Schrittauflösung betrieben 2 = Bedienpult aktiv, Trackball wird mit Faktor betrieben.	
Beispiel:	LSX.GetBPZ(&AValue);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	1	?bpz	-

LSX_SetBPZ		
Beschreibung	Setzt den Zustand des Zusatzbedienpults mit Trackball (nicht für LSTEP express)	
Delphi	function LSX_SetBPZ(LSID: Integer; AValue: Integer): Integer;	
C++	int SetBPZ(int lAValue);	
LabView:	-	
Parameter	AValue	Aktivität des Bedienpults 0 = Bedienpult ist „Aus“. 1 = Bedienpult aktiv und der Trackball wird mit 0,1µ Schrittauflösung betrieben 2 = Bedienpult aktiv, Trackball wird mit Faktor betrieben.
Beispiel	LSX.SetBPZ(1);	

Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	1	!bpz	-

LSX_GetBPZJoyspeed			
Beschreibung:	Bedienpult Geschwindigkeit des Joysticks abfragen (nicht für LSTEP express)		
Delphi:	function LSX_GetBPZJoyspeed(LSID: Integer; APar: Integer; var AValue: Double): Integer;		
C++:	int GetBPZJoyspeed(int lAPar, double *pdAValue);		
LabView:	-		
Parameter:	APar	Parameter 1 = X-Achse 2 = Y-Achse 3 = Z-Achse	
	AValue	Maximale Geschwindigkeit in U/s	
Beispiel:	GetBPZJoyspeed(1, &AValue); // Auslesen der eingestellten Geschwindigkeit von Parameter 1.		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?joyspeed	-

LSX_SetBPZJoyspeed			
Beschreibung	Bedienpult Geschwindigkeit des Joysticks einstellen (nicht für LSTEP express)		
Delphi	function LSX_SetBPZJoyspeed(LSID: Integer; APar: Integer; AValue: Double): Integer;		
C++	int SetBPZJoyspeed(int lAPar, double dAValue);		
LabView:	-		
Parameter:	APar	Parameter 1 = X-Achse 2 = Y-Achse 3 = Z-Achse	
	AValue	Maximale Geschwindigkeit in U/s	
Beispiel	SetBPZJoyspeed(1, 25) // Parameter 1 mit Geschwindigkeit 25 beschreiben		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!joyspeed	-

LSX_GetBPZTrackballBacklash			
Beschreibung	Funktion zum Auslesen des eingestellten Trackball-Umkehrspiels beim Bedienpult. (nicht für LSTEP express)		
Delphi	function LSX_GetBPZTrackballBackLash(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++	int GetBPZTrackballBackLash(double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:	-		
Parameter	X, Y, Z, A	Umkehrspiel in mm	
Beispiel	LSX.GetBPZTrackballBackLash(&X, &Y, &Z, &R);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?bpzbl	-

LSX_SetBPZTrackballBacklash			
Beschreibung	Funktion zum Setzen des Trackball-Umkehrspiels beim Bedienpult. (nicht für LSTEP express)		
Delphi	function LSX_SetBPZTrackballBackLash(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++	int SetBPZTrackballBackLash (double dX, double dY, double dZ, double dA);		
LabView:	-		
Parameter	X, Y, Z, A	Einzustellendes Umkehrspiel	
Beispiel	LSX.SetBPZTrackballBackLash(0.01, 0.01, 0.01, 0.01);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!bpzbl	-

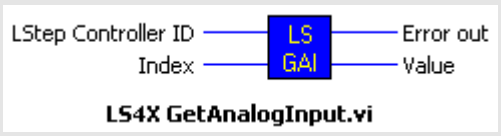
LSX_GetBPZTrackballFactor			
Beschreibung	Liest den Trackballfaktor aus. (nicht für die LSTEP express)		
Delphi	function LSX_GetBPZTrackballFactor(LSID: Integer; AValue: Double): Integer;		
C++	int GetBPZTrackballFactor(double *pdAValue);		
LabView:	-		
Parameter	AValue	Trackballfaktor in Motorinkrementen/Trackballimpulse	
Beispiel	LSX.GetBPZTrackballFactor(&AValue) ;		
Sontiges	Kompatabilität	"SendString" Befehl	Aktivierung (LSTEP express series)
	1	?bpztf	-

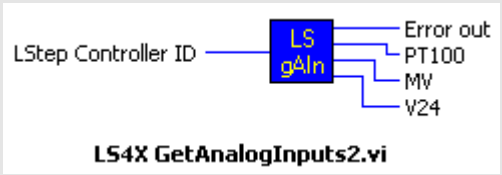
LSX_SetBPZTrackballFactor			
Beschreibung	Funktion zum setzen des Trackballfaktors auf dem Schaltbrett. (nicht für die LSTEP express)		
Delphi	function LSX_SetBPZTrackballFactor(LSID: Integer; AValue: Double): Integer;		
C++	int SetBPZTrackballFactor(double dAValue);		
LabView	-		
Parameter	AValue	Trackballfaktor in Motroinkrementen/Trackballimpuls Bsp. Faktor = 1, bedeutet, dass ein Trackbakkimpuls für ein Motor-inkrement zählt.	
Beispiel	LSX.SetBPZTrackballFactor(1.0) ;		
Sonstiges	Kompatabilität	"SendString" Befehl	Aktivierung (LSTEP express series)
	1	!bpztf	-

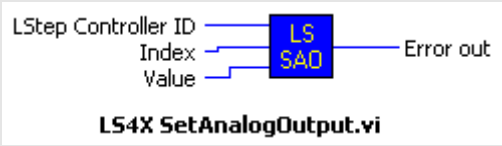
LSX_GetJoyOutPass			
Beschreibung	Befehl zum Auslesen der Filterzeitkonstante der Joystick-Funktion. Filter verhindert ruckartige Veränderungen der Geschwindigkeit (Rampenfunktion).		
Delphi	function LSX_GetJoyOutPass (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++	int GetJoyOutPass (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter	X, Y, Z, A	Filterzeitkonstante, Bereich: 0 – 500 000 µs	
Beispiel	LSX.GetJoyOutPass (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	? joyoutpass	-

LSX_SetJoyOutPass			
Beschreibung	Befehl zum Einstellen der Filterzeitkonstante der Joystick-Funktion. Filter verhindert ruckartige Veränderungen der Geschwindigkeit (Rampenfunktion).		
Delphi	function LSX_SetJoyOutPass (LSID: Integer; X, Y, Z, A): Integer;		
C++	int SetJoyOutPass (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter	X, Y, Z, A	Filterzeitkonstante, Bereich: 0 – 500 000 µs	
Beispiel	LSX.SetJoyOutPass (20, 20, 20, 10);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!joyoutpass	-

4.2.13 Digitale und analoge Ein- und Ausgänge

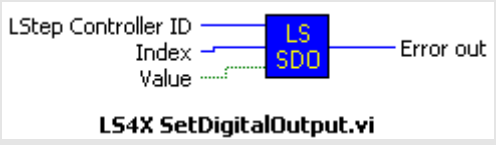
LSX_GetAnalogInput			
Beschreibung:	Funktion zum Lesen des aktuellen Werts eines Analogkanals.		
Delphi:	function LSX_GetAnalogInput(LSID: Integer; Index: Integer; var Value: Integer): Integer;		
C++:	int GetAnalogInput(int lIndex,int *plValue);		
LabView:			
Parameter:	Index	Auszulesender Analogkanal	
	Value	Zeiger auf Integerwert, in den der aktuelle Zustand des Analogkanals kopiert wird	
Beispiel:	LSX.GetAnalogInput(0, &Eingang0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?anain	-

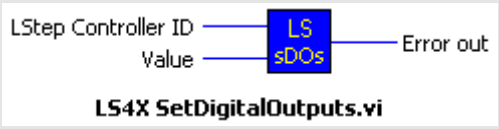
LSX_GetAnalogInputs2			
Beschreibung:	Lesen der aktuellen Zustände der Analogkanäle (Channel 6, 7, 8). (nur bei der LSTEP-PCI, LSTEP-PC)		
Delphi:	function LSX_GetAnalogInputs2(LSID: Integer; var PT100, MV, V24: Integer): Integer;		
C++:	int GetAnalogInputs2 (int *plPT100, int *plMV, int *plV24);		
LabView:			
Parameter:	PT100, MV, V24	Zeiger auf Integerwert, in den GetAnalogInputs2 den aktuellen Zustand des Analogkanals schreiben soll	
Beispiel:	LSX.GetAnalogInputs2(&PT100, &MV, &V24);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	-	-

LSX_SetAnalogOutput			
Beschreibung:	Funktion zum Setzen eines Analogausgangs.		
Delphi:	function LSX_SetAnalogOutput(LSID: Integer; Index: Integer; Value: Integer): Integer;		
C++:	int SetAnalogOutput(int lIndex,int lValue);		
LabView:			
Parameter:	Index	Nummer des Analogkanals	
	Value	Aussteuerung des Analogausganges in %	
Beispiel:	LSX.SetAnalogOutput(0, 100); // Ausgang 0 auf Maximum setzen		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!anaout	-

LSX_GetDigitalInputs		
Beschreibung:	Funktion zum Einlesen der digitalen Eingänge 0 bis 15.	
Delphi:	function LSX_GetDigitalInputs(LSID: Integer; var Value: Integer): Integer;	
C++:	int GetDigitalInputs(int *plValue);	
LabView:		
Parameter:	Value	Zeiger auf einen Integerwert, der den Zustand der digitalen Eingänge als Bitmaske enthält. Bit 0 = Eingang 0 Bit 1 = Eingang 1 … Wert 0 = Am Eingang liegt eine logische 0 an Wert 1 = Am Eingang liegt eine logische 1 an
Beispiel:	int Eingange; LSX.GetDigitalInputs(&Eingange); if (Eingange & 16) ... // Wenn Inputpin 4 gesetzt	
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?digin	-

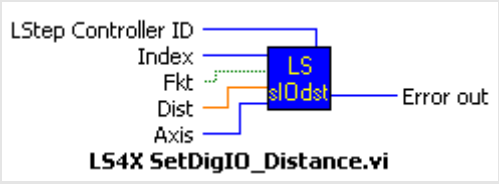
LSX_GetDigitalInputsE				
Beschreibung:	Zusätzliche digitale Eingänge 16 bis 31 einlesen.			
Delphi:	function LSX_GetDigitalInputsE(LSID: Integer; var Value: Integer): Integer;			
C++:	int GetDigitalInputsE(int *plValue);			
LabView:	Nicht unterstützt			
Parameter:	Value	Zeiger auf einen Integerwert, der den Zustand der digitalen Eingänge als Bitmaske enthält. Bit 0 = Eingang 16 Bit 1 = Eingang 17 … Wert 0 = Am Eingang liegt eine logische 0 an Wert 1 = Am Eingang liegt eine logische 1 an		
Beispiel:	LSX.GetDigitalInputsE(&i);			
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)	
	3	?diginext (LStepex-press)/?edigin	-	

LSX_SetDigitalOutput				
Beschreibung:	Funktion zum Setzen eines digitalen Ausgangs.			
Delphi:	function LSX_SetDigitalOutput(LSID: Integer; Index: Integer; Value: LongBool): Integer;			
C++:	int SetDigitalOutput(int lIndex,BOOL Value);			
LabView:				
Parameter:	Index	Nummer des digitalen Ausgangs 0 = Ausgang 0 1 = Ausgang 1 …		
	Value	Zustand auf „0" oder „1" setzen True = Ausgang auf 1 setzen False = Ausgang auf 0 setzen		
Beispiel:	LSX.SetDigitalOutput(0, true); // Outputpin 0 auf „1" setzen			
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)	
	3	!digout	-	

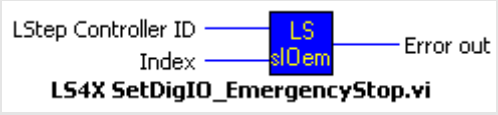
LSX_SetDigitalOutputs			
Beschreibung:	Funktion zum gleichzeitigen Setzen der digitalen Ausgänge 0 bis 15.		
Delphi:	function LSX_SetDigitalOutputs(LSID: Integer; Value: Integer): Integer;		
C++:	int SetDigitalOutputs(int lValue);		
LabView:			
Parameter:	Value	Bitmaske zum Setzen der Ausgänge Bit 0 = Ausgang 0 Bit 1 = Ausgang 1 … Wert 0 = Ausgang auf 0 setzen Wert 1 = Ausgang auf 1 setzen	
Beispiel:	LSX.SetDigitalOutputs($03); // Ausgänge 0 und 1 auf 1 setzen, die übrigen auf 0		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!digout	-

LSX_SetDigitalOutputsE			
Beschreibung:	Funktion zum gleichzeitigen Setzen der zusätzlichen digitalen Ausgänge 16 bis 31.		
Delphi:	function LSX_SetDigitalOutputsE(LSID: Integer; Value: Integer): Integer;		
C++:	int SetDigitalOutputsE(int lValue);		
LabView:	Nicht unterstützt		
Parameter:	Value	Bitmaske zum Setzen der Ausgänge Bit 0 = Ausgang 16 Bit 1 = Ausgang 17 … Wert 0 = Ausgang auf 0 setzen Wert 1 = Ausgang auf 1 setzen	
Beispiel:	LSX.SetDigitalOutputsE($03); // Ausgänge 16 und 17 auf 1 setzen, die übrigen auf 0		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!digoutext (LStepex-press)/!edigout	-

LSX_SetDigIO_Distance

Beschreibung:	Funktion zur Aktivierung eines Ausgangs in Abhängigkeit der eingestellten Strecke vor/nach der Zielposition. (nicht für LSTEP express)
Delphi:	function LSX_SetDigIO_Distance(LSID: Integer; Index: Integer; Fkt: LongBool; Dist: Double; Axis: Integer): Integer;
C++:	int SetDigIO_Distance(int lIndex,BOOL Fkt,double dDist,int lAxis);
LabView:	

Parameter:	Index	Nummer des digitalen Ausgangs	
	Fkt	Aktivierungsart False = Aktivierung eines Ausgangs in Abhängigkeit der eingestellten Strecke vor der Zielposition. True = Aktivierung eines Ausgangs in Abhängigkeit der eingestellten Strecke nach der Startposition.	
	Dist	Strecke in der eingestellten Dimension	
	Axis	Achse, zu der die Funktion zugeordnet werden soll 1 = X-Achse 2 = Y-Achse …	
Beispiel:	LSX.SetDigIO_Distance(7, false, 78.9, 3); /* Ausgang 7 wird 78.9mm vor erreichen der Zielposition (Z- Achse) aktiviert. */		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!digfkt / !edigfkt	-

LSX_SetDigIO_EmergencyStop			
Beschreibung:	Funktion zur Zuordnung eines digitalen Eingangs als Not-Stopp-Pin (nicht für LSTEP express)		
Delphi:	function LSX_SetDigIO_EmergencyStop(LSID: Integer; Index: Integer): Integer;		
C++:	int SetDigIO_EmergencyStop(int lIndex);		
LabView:			
Parameter:	Index	Nummer des digitalen Eingangs der die Funktion erhalten soll 0 = Eingang 0 1 = Eingang 1 …	
Beispiel:	LSX.SetDigIOEmergencyStop(15); // Not-Stopp-Pin 15		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!digfkt / !edigfkt	-

LSX_SetDigIO_Off			
Beschreibung:	Funktion der digitalen Ein-/Ausgänge deaktivieren. (Keine Beeinflussung der Ein-/Ausgänge) (nicht für LSTEP express)		
Delphi:	function LSX_SetDigIO_Off(LSID: Integer; Index: Integer): Integer;		
C++:	int SetDigIO_Off(int lIndex);		
LabView:			
Parameter:	Index	Nummer des digitalen Eingangs dessen Funktionszuweisung deaktiviert werden soll. 0 = Eingang 0 1 = Eingang 1 …	
Beispiel:	LSX.SetDigIO_Off(0); // dig. Fkt. Input-/Outputpin 0 Aus		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!digfkt / !edigfkt	-

LSX_SetDigIO_Polarity			
Beschreibung:	Einstellung der Polarität für die verschiedenen Funktionen der digitalen Ein- bzw. Ausgänge (nicht für LSTEP express)		
Delphi:	function LSX_SetDigIO_Polarity(LSID: Integer; Index: Integer; High: LongBool): Integer;		
C++:	int SetDigIO_Polarity(int lIndex,BOOL High);		
LabView:			
Parameter:	Index	Nummer des digitalen Eingangs dessen Polarität geändert werden soll 0 = Eingang 0 1 = Eingang 1 …	
	High	Einstellung der Polarität True = High-aktiv False = Low-aktiv	
Beispiel:	LSX.SetDigIO_Polarity(3, True); // Input-/Outputpin 3 high-aktiv		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!digfkt / !edigfkt	-

LSX_GetDigOutLinktoSignal			
Beschreibung:	Liest aus welches Signal zu einem der 16 bzw. 32 digitalen Ausgänge (24V) zugeordnet ist. Ist einem Ausgang ein Signal zugeordnet, dann kann dieser nicht mehr mit „LSX_SetDigitalOutputs“ beschrieben werden. Die Signale 1 bis 6 werden mit einer Abtastrate von ca. 1 ms abgefragt und ausgegeben und können nur einem Ausgang zugeordnet werden. Die Signale 10 bis 13, 100 bis 115 und 200 bis 205 werden mit einer Rate von 8 ms abgefragt und ausgegeben und können mehreren Ausgängen zugeordnet werden.		
Delphi:	function LSX_GetDigOutLinktoSignal (LSID: Integer; Output: Integer; var toSignal: Integer): Integer;		
C++:	int GetDigOutLinktoSignal (int lOutput, int *pltoSignal);		
LabView:	Nicht unterstützt		
Parameter:	Ausgang	Signal	
	0 bis 15 bzw. 31	0 → Signalzuordnung aufheben	
		1 → Stillstandsmeldung	
		2 → Stopp-Aktiv-Meldung	
		3 → Zielfenstermeldung	
		4 → Mindestens eine Endstufe eingeschaltet	
		5 → Manueller Modus aktiv	
		6 → Geschwindigkeitsschwellwert unterschritten	
		10 → Manueller Modus der X-Achse aktiv	
		11 → Manueller Modus der Y-Achse aktiv	
		12 → Manueller Modus der Z-Achse aktiv	
		13 → Manueller Modus der A-Achse aktiv	
		100 bis 131 → Digitaler Eingänge 0…31 (24V)	
		200 bis 205 → Digitaler Eingang 0…5 (TTL_Pegel)	
Beispiel:	LSX.GetDigitalOutLinktoSignal(Output,&toSignal);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	?digoutlinktosignal	-

LSX_SetDigitalOutLinktoSignal			
Beschreibung:	Ordnet einen der 16 bzw. 32 digitalen Ausgänge (24V) ein Signal zu. Ist einem Ausgang ein Signal zugeordnet, dann kann dieser nicht mehr mit „LSX_SetDigitalOutputs“ beschrieben werden. Die Signale 1 bis 6 werden mit einer Abtastrate von ca. 1 ms abgefragt und ausgegeben und können nur einem Ausgang zugeordnet werden. Die Signale 10 bis 13, 100 bis 115 und 200 bis 205 werden mit einer Rate von 8 ms abgefragt und ausgegeben und können mehreren Ausgängen zugeordnet werden.		
Delphi:	function LSX_SetDigOutLinktoSignal (LSID : Integer;Output: Integer; toSignal: Integer): Integer;		
C++:	int SetDigOutLinktoSignal (int lOutput, int ltoSignal);		
LabView:	Nicht unterstützt		
Parameter:	Ausgang	Signal	
	0 bis 15 bzw. 31	0 → Signalzuordnung aufheben	
		1 → Stillstandsmeldung	
		2 → Stopp-Aktiv-Meldung	
		3 → Zielfenstermeldung	
		4 → Mindestens eine Endstufe eingeschaltet	
		5 → Manueller Modus aktiv	
		6 → Geschwindigkeitsschwellwert unterschritten	
		10 → Manueller Modus der X-Achse aktiv	
		11 → Manueller Modus der Y-Achse aktiv	
		12 → Manueller Modus der Z-Achse aktiv	
		13 → Manueller Modus der A-Achse aktiv	
		100 bis 131 → Digitaler Eingänge 0…31 (24V)	
		200 bis 205 → Digitaler Eingang 0…5 (TTL_Pegel)	
Beispiel:	LSX.SetDigitalOutLinktoSignal(5,1);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	!digoutlinktosignal	-

LSX_GetInvertDigOutSignal			
Beschreibung:	Liest aus, ob ein digitales Signal invertiert werden soll. Die Signale können über den Befehl „LSX_SetDigitalOutLinktoSignal“ an einen digitalen Ausgang geleitet werden.		
Delphi:	function LSX_GetInvertDigOutSignal (LSID: Integer;Output: Integer; var invert: LongBool): Integer;		
C++:	int GetInvertDigOutSignal (int lOutput, BOOL *pbinvert);		
LabView:	Nicht unterstützt		
Parameter:	Signal		
	False = Keine Invertierung True = Invertierung		
Beispiel:	LSX.GetInvertDigOutSignal(Output,&invert);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	?invertdigoutsingal	-

LSX_SetInvertDigOutSignal			
Beschreibung:	Befehl zum invertieren von digitalen Signalen. Die Signale können über den Befehl „LSX_SetDigitalOutLinktoSignal“ an einen digitalen Ausgang geleitet werden.		
Delphi:	function LSX_SetDigOutLinktoSignal (LSID : Integer;Output: Integer; toSignal: Integer): Integer;		
C++:	int SetDigOutLinktoSignal (int lOutput, int ltoSignal);		
LabView:	Nicht unterstützt		
Parameter:	Signal		
	False = Keine Invertierung True = Invertierung		
Beispiel:	LSX.SetInvertDigOutSignal(5,True);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	!invertdigoutsignal	-

<table>
<tr><th colspan="5">LSX_GetDigInStatus</th></tr>
<tr><td>Beschreibung:</td><td colspan="4">Digitale Eingänge 0 bis 15
Befehle zum Auslesen des Status des automatischen Sendens von 16 digitalen Eingängen mit 24 V-Pegel bzw. 6 Eingängen mit TTL-Pegel bei Änderung derselben. Die Prüfung der Eingänge auf Änderung erfolgt zyklisch im Abstand von ca. 8 ms. Nach Aktivierung des automatischen Sendens wird Zustand der Eingänge einmalig direkt ausgegeben.</td></tr>
<tr><td>Delphi:</td><td colspan="4">function LSX_GetDigInStatus (LSID: Integer; status: Integer): Integer;</td></tr>
<tr><td>C++:</td><td colspan="4">int GetDigInStatus (int *plstatus);</td></tr>
<tr><td>LabView:</td><td colspan="4">Nicht unterstützt</td></tr>
<tr><td>Parameter:</td><td>status</td><td colspan="3">0: Inaktiv
1: Aktiv, bitweise Übertragung (16 Bit)
2: Aktiv, Übertragung im Hexadezimal-Format mit 4 Stellen, Low- und Highbyte sind getauscht
3: Aktiv, bitweise Übertragung (16 Bit), Erweitertes Protokoll Kanal 4
4: Aktiv, Übertragung im Hexadezimal-Format mit 4 Stellen, Low- und Highbyte sind getauscht, Erweitertes Protokoll Kanal 4</td></tr>
<tr><td>Beispiel:</td><td colspan="4">LSX.GetDigInStatus (&status);</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td colspan="2">Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>2</td><td>?diginstatus</td><td colspan="2">-</td></tr>
</table>

LSX_SetDigInStatus, LSX_SetDigInStatusW			
Beschreibung:	Digitale Eingänge 0 bis 15 Befehle zum Aktivieren des automatischen Sendens von 16 digitalen Eingängen mit 24 V-Pegel bzw. 6 Eingängen mit TTL-Pegel bei Änderung derselben. Die Prüfung der Eingänge auf Änderung erfolgt zyklisch im Abstand von ca. 8 ms. Nach Aktivierung des automatischen Sendens wird der Zustand der Eingänge einmalig direkt ausgegeben.		
Delphi:	function LSX_SetDigInStatus (LSID : Integer; status: Integer; var Feedback: PAnsiChar; MaxLen: Integer): Integer; function LSX_SetDigInStatusW (LSID : Integer; status: Integer; var Feedback: PWideChar; MaxLen: Integer): Integer;		
C++:	int SetDigInStatus (int lstatus, char *pcFeedback, int lMaxLen); int SetDigInStatusW (int lstatus, TCHAR *pcFeedback, int lMaxLen);		
LabView:	Nicht unterstützt		
Parameter:	status	0: Inaktiv 1: Aktiv, bitweise Übertragung (16 Bit) 2: Aktiv, Übertragung im Hexadezimal-Format mit 4 Stellen, Low- und Highbyte sind getauscht 3: Aktiv, bitweise Übertragung (16 Bit), Erweitertes Protokoll Kanal 4 4: Aktiv, Übertragung im Hexadezimal-Format mit 4 Stellen, Low- und Highbyte sind getauscht, Erweitertes Protokoll Kanal 4	
Rückmeldung	Feedback	Bitweise Übertragung: !0 xxxxxxxxxxxxxxxx (diginstatus) !1 xxxxxxxxxxxxxxxx (diginextstatus) !2 xxxxxxxxxxxxxxxx (ttldiginstatus) Hexadezimal-Format: !0 xxxx (diginstatus / Low- und Highbyte sind getauscht) !1 xxxx (diginextstatus / Low- und Highbyte sind getauscht) !2 xxxx (ttldiginstatus / Low- und Highbyte sind getauscht)	
Beispiel:	LSX.SetDigInStatus (status pcFeedback, 256);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!diginstatus	Sofort

<table>
<tr><th colspan="5">LSX_GetDigInExtStatus</th></tr>
<tr><td>Beschreibung:</td><td colspan="4">Digitale Eingänge 16 bis 31
Befehle zum Auslesen des Status des automatischen Sendens von 16 digitalen Eingängen mit 24 V-Pegel bzw. 6 Eingängen mit TTL-Pegel bei Änderung derselben. Die Prüfung der Eingänge auf Änderung erfolgt zyklisch im Abstand von ca. 8 ms. Nach Aktivierung des automatischen Sendens wird Zustand der Eingänge einmalig direkt ausgegeben.</td></tr>
<tr><td>Delphi:</td><td colspan="4">function LSX_GetDigInExtStatus (LSID: Integer; status: Integer): Integer;</td></tr>
<tr><td>C++:</td><td colspan="4">int GetDigInExtStatus (int *plstatus);</td></tr>
<tr><td>LabView:</td><td colspan="4">Nicht unterstützt</td></tr>
<tr><td>Parameter:</td><td>status</td><td colspan="3">0: Inaktiv
1: Aktiv, bitweise Übertragung (16 Bit)
2: Aktiv, Übertragung im Hexadezimal-Format mit 4 Stellen, Low- und Highbyte sind getauscht
3: Aktiv, bitweise Übertragung (16 Bit), Erweitertes Protokoll Kanal 4
4: Aktiv, Übertragung im Hexadezimal-Format mit 4 Stellen, Low- und Highbyte sind getauscht, Erweitertes Protokoll Kanal 4</td></tr>
<tr><td>Beispiel:</td><td colspan="4">LSX.GetDigInExtStatus (&status);</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td colspan="2">Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>2</td><td>?diginextstatus</td><td colspan="2">-</td></tr>
</table>

LSX_SetDigInExtStatus, LSX_SetDigInExtStatusW			
Beschreibung:	Digitale Eingänge 16 bis 31 Befehle zum Aktivieren des automatischen Sendens von 16 digitalen Eingängen mit 24 V-Pegel bzw. 6 Eingängen mit TTL-Pegel bei Änderung derselben. Die Prüfung der Eingänge auf Änderung erfolgt zyklisch im Abstand von ca. 8 ms. Nach Aktivierung des automatischen Sendens wird Zustand der Eingänge einmalig direkt ausgegeben.		
Delphi:	function LSX_SetDigInExtStatus(LSID: Integer; status: Integer; var Feedback: PAnsiChar; MaxLen: Integer): Integer; function LSX_SetDigInExtStatusW(LSID: Integer; status: Integer; var Feedback: PWideChar; MaxLen: Integer): Integer;		
C++:	int SetDigInExtStatus (int lstatus, char *pcFeedback, int lMaxLen); int SetDigInExtStatusW (int lstatus, TCHAR *pcFeedback, int lMaxLen);		
LabView:	Nicht unterstützt		
Parameter:	status	0: Inaktiv 1: Aktiv, bitweise Übertragung (16 Bit) 2: Aktiv, Übertragung im Hexadezimal-Format mit 4 Stellen, Low- und Highbyte sind getauscht 3: Aktiv, bitweise Übertragung (16 Bit), Erweitertes Protokoll Kanal 4 4: Aktiv, Übertragung im Hexadezimal-Format mit 4 Stellen, Low- und Highbyte sind getauscht, Erweitertes Protokoll Kanal 4	
Rückmeldung	Feedback	Bitweise Übertragung: !0 xxxxxxxxxxxxxxxx (diginstatus) !1 xxxxxxxxxxxxxxxx (diginextstatus) !2 xxxxxxxxxxxxxxxx (ttldiginstatus) Hexadezimal-Format: !0 xxxx (diginstatus / Low- und Highbyte sind getauscht) !1 xxxx (diginextstatus / Low- und Highbyte sind getauscht) !2 xxxx (ttldiginstatus / Low- und Highbyte sind getauscht)	
Beispiel:	LSX.SetDigInExtStatus (status, pcFeedback,256);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!diginextstatus	Sofort

LSX_GetTTLDigIn			
Beschreibung:	Befehl zum Einlesen der 6 digitalen Eingänge mit TTL-Pegel.		
Delphi:	function LSX_GetTTLDigIn (LSID: Integer; status: Integer): Integer;		
C++:	int GetTTLDigIn (int *plstatus);		
LabView:	Nicht unterstützt		
Parameter:	status	Zustand der digitalen Eingänge	
Beispiel:	LSX.GetTTLDigIn (&status);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?ttldigin	-

LSX_GetTTLDigOut			
Beschreibung:	Befehl zum Auslesen der 4 digitalen TTL-Ausgänge. Funktion muss durch Konfiguration der Pins 16 bis 19 des 50-poligen Multi-Funktions-Ports (MFP) als TTL-Ausgänge freigeschaltet werden (siehe Befehl "ttloutconfig").		
Delphi:	function LSX_GetTTLDigOut (LSID: Integer; status: Integer): Integer;		
C++:	int GetTTLDigOut (int *plstatus);		
LabView:	Nicht unterstützt		
Parameter:	status	Zustand der digitalen Ausgänge	
Beispiel:	LSX.GetTTLDigOut (&status);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?ttldigout	-

LSX_SetTTLDigOut			
Beschreibung:	Befehl zum Setzen und Löschen der 4 digitalen TTL-Ausgänge. Funktion muss durch Konfiguration der Pins 16 bis 19 des 50-poligen Multi-Funktions-Ports (MFP) als TTL-Ausgänge freigeschaltet werden (siehe Befehl "ttloutconfig").		
Delphi:	function LSX_SetTTLDigOut (LSID: Integer; status: Integer): Integer;		
C++:	int SetTTLDigOut (int *plstatus);		
LabView:	Nicht unterstützt		
Parameter:	status	Zustand der digitalen Ausgänge	
Beispiel:	LSX.SetTTLDigOut (1110);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!ttldigout	Sofort

LSX_GetMotorBrakeDigOut			
Beschreibung:	Befehl zum Auslesen der Verfahrrichtung zur Drehrichtung des Trackballs.		
Delphi:	function LSX_GetMotorBrakeDigOut (LSID: Integer; var X, Y, Z, A: LongBool): Integer;		
C++:	int GetMotorBrakeDigOut (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einstellung pro Achse: HIGH/LOW	
Beispiel:	LSX.GetMotorBrakeDigOut (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorbrakedigout	-

LSX_SetMotorBrakeDigOut			
Beschreibung:	Befehl zum Einstellen der Verfahrrichtung zur Drehrichtung des Trackballs.		
Delphi:	function LSX_SetMotorBrakeDigOut (LSID: Integer; X, Y, Z, A: LongBool): Integer;		
C++:	int SetMotorBrakeDigOut (BOOL bX, BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einstellung pro Achse: HIGH/LOW	
Beispiel:	LSX.SetMotorBrakeDigOut (1, 1, 1, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!motorbrakedigout	Sofort

LSX_GetMotorBrakeOut			
Beschreibung:	Befehl zum Einlesen des Zustands der Motorbremsenausgänge		
Delphi:	function LSX_GetMotorBrakeOut (LSID: Integer; var X, Y, Z, A: LongBool): Integer;		
C++:	int GetMotorBrakeOut (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einstellung pro Achse: HIGH/LOW	
Beispiel:	LSX.GetMotorBrakeOut (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?motorbrakeout	-

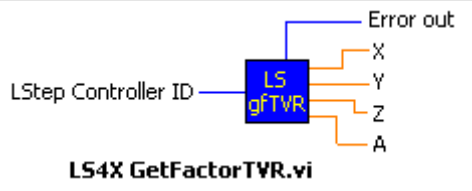
LSX_GetTVR			
Beschreibung:	Befehl zum Auslesen des Status von TVR-IN		
Delphi:	function LSX_GetTVR (LSID: Integer; var X, Y, Z, A: LongBool): Integer;		
C++:	int GetTVR (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einstellung pro Achse: An/Aus	
Beispiel:	LSX.GetTVR (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?tvr	-

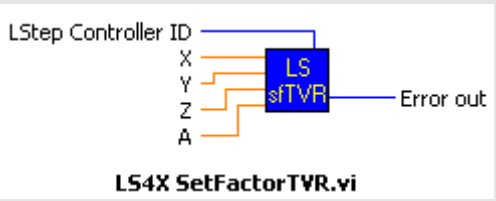
LSX_SetTVR			
Beschreibung:	Befehl zum Ein- und Ausschalten von TVR-IN		
Delphi:	function LSX_SetTVR (LSID: Integer; X, Y, Z, A: LongBool): Integer;		
C++:	int SetTVR (BOOL bX, BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einstellung pro Achse: An/Aus	
Beispiel:	LSX.SetTVR (1, 1, 1, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!tvr	Sofort

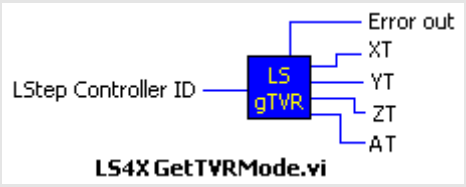
LSX_GetTVRToAxis			
Beschreibung:	Befehl zum Auslesen der Zuordnung der Trackballachsen (horizontal und vertikal) zu den mechanischen Achsen X, Y, Z und A.		
Delphi:	function LSX_GetTVRToAxis (LSID: Integer;var X,Y,Z,A: Integer): Integer;		
C++:	int GetTVRToAxis (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Zuordnung: 0: Keine Achszuordnung 1: QEP 1 2. QEP 2	
Beispiel:	LSX.GetTVRToAxis (&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?tvrtoaxis	-

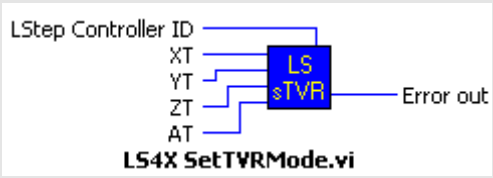
LSX_SetTVRToAxis			
Beschreibung:	Befehl zum Zuordnen der Trackballachsen (horizontal und vertikal) zu den mechanischen Achsen X, Y, Z und A.		
Delphi:	function LSX_SetTVRToAxis (LSID: Integer;X,Y,Z,A: Integer): Integer;		
C++:	int SetTVRToAxis (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	Zuordnung: 0: Keine Achszuordnung 1: QEP 1 2. QEP 2	
Beispiel:	LSX.SetTVRToAxis (2,1,0 ,0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!tvrtoaxis	TVR

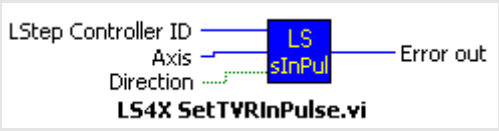
4.2.14 Takt-Vor/Rück In

LSX_GetFactorTVR			
Beschreibung:	Funktion zum Auslesen des Takt Vor / Rück Faktors.		
Delphi:	function LSX_GetFactorTVR(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetFactorTVR(double *pdX, double *pdY, double *pdZ, double *pdR);		
LabView:			
Parameter:	X, Y, Z, A	Eingestellter Takt Vor / Rück Faktor in Motorinkremente/Takt	
Beispiel:	LSX.GetFactorTVR(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?tvrf	-

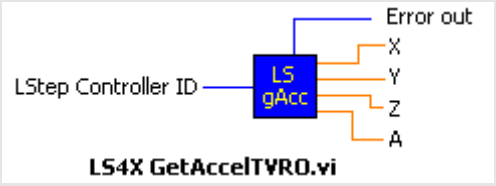
LSX_SetFactorTVR			
Beschreibung:	Funktion zum Setzen des Takt Vor / Rück Faktors.		
Delphi:	function LSX_SetFactorTVR(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetFactorTVR(double dX,double dY,double dZ,double dA);		
LabView:			
Parameter:	X, Y, Z, A	Einzustellender Takt Vor / Rück Faktor in Motorinkremente/Takt	
Beispiel:	LSX.SetFactorTVR(2.0, 2.0, 0, 0); /* Bei Achse X und Y soll Takt Vor/Rück mit dem Faktor 2 arbeiten */		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!tvrf	tvr

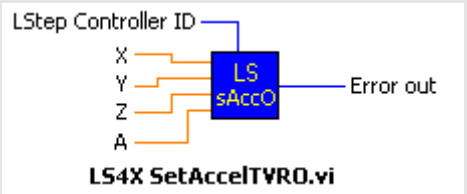
LSX_GetTVRMode			
Beschreibung:	Liest den eingestellten Takt Vor / Rück Modus aus.		
Delphi:	function LSX_GetTVRMode(LSID: Integer; var XT, YT, ZT, AT: Integer): Integer;		
C++:	int GetTVRMode(int *plXT, int *plYT, int *plZT, int *plAT);		
LabView:			
Parameter:	XT, YT, ZT, AT	Eingestellter Takt Vor/Rück Modus 0 = Takt Vor/Rück ist „Aus“ 1 = Normale Takt Vor/Rück Bearbeitung 2 = Takt Vor/Rück Bearbeitung arbeitet mit einem Faktor 3 = Takt Vor/Rück Bearbeitung benötigt externe Freigabe über Triggerout Pin (MFP). 4 = Kombination aus 2 & 3.	
Beispiel:	LSX.GetTVRMode(&XT, &YT, &ZT, &AT);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	1	?tvr	-

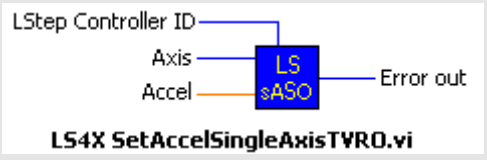
<table>
<tr><th colspan="4">LSX_SetTVRMode</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Funktion zum Einstellen des Takt Vor / Rück Modus.</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_SetTVRMode(LSID: Integer; XT, YT, ZT, AT: Integer): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int SetTVRMode(int lXT, int lYT,int lZT,int lAT);</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td>Parameter:</td><td>XT, YT, ZT, AT</td><td colspan="2">Einzustellender Takt Vor/Rück Modus
0 = Takt Vor/Rück ist „Aus“
1 = Normale Takt Vor/Rück Bearbeitung
2 = Takt Vor/Rück Bearbeitung arbeitet mit einem Faktor
3 = Takt Vor/Rück Bearbeitung benötigt externe Freigabe über Triggerout Pin (MFP).
4 = Kombination aus 2 & 3.</td></tr>
<tr><td>Beispiel:</td><td colspan="3">LSX.SetTVRMode(1, 1, 0, 0); // TVR X- und Y-Achse Ein</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>1</td><td>!tvr</td><td>-</td></tr>
</table>

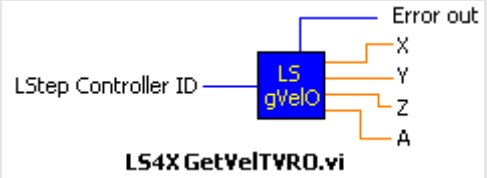
<table>
<tr><th colspan="4">LSX_SetTVRInPulse</th></tr>
<tr><td>Beschreibung:</td><td colspan="3">Diese Funktion kann verwendet werden um die Takt Vor / Rück Funktion per Software zu steuern.
(nicht für LSTEP express)</td></tr>
<tr><td>Delphi:</td><td colspan="3">function LSX_SetTVRInPulse(LSID: Integer; Axis: Integer; Direction: Boolean): Integer;</td></tr>
<tr><td>C++:</td><td colspan="3">int SetTVRInPulse(int Axis, BOOL Direction);</td></tr>
<tr><td>LabView:</td><td colspan="3"> 
</td></tr>
<tr><td rowspan="2">Parameter:</td><td>Axis</td><td colspan="2">Achse an die der Softwarepuls gesendet werden sollen
1 = X-Achse
2 = Y-Achse
…</td></tr>
<tr><td>Direction</td><td colspan="2">Richtungssignal
True = Vorwärts
False = Rückwärts</td></tr>
<tr><td>Beispiel:</td><td colspan="3">LSX.SetTVRInPulse (2, true); // 1 Takt vorwärts bei der y-Achse.</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td>Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td>1</td><td>!px / !nx
!py / !ny
…</td><td>-</td></tr>
</table>

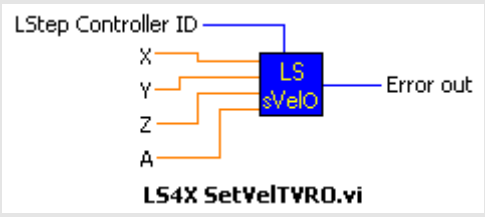
4.2.15 Takt-Vor/Rück Ausgänge für weitere Achsen

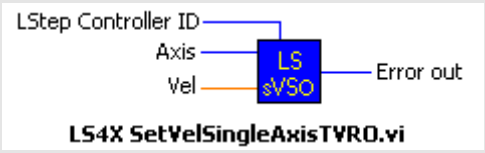
LSX_GetAccelTVRO			
Beschreibung:	Liest die eingestellten Beschleunigungen für die weiteren Takt Vor/Rück Ausgangs Achsen aus. (nicht für LSTEP express)		
Delphi:	function LSX_GetAccelTVRO(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetAccelTVRO(double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	X, Y, Z, A	Beschleunigungswerte in U/s²	
Beispiel:	LSX.GetAccelTVRO(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?tvroa	-

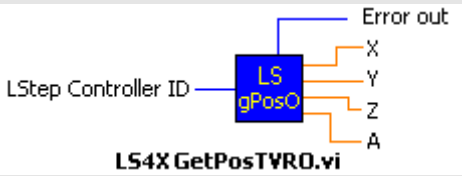
LSX_SetAccelTVRO			
Beschreibung:	Beschleunigung für die weiteren Takt Vor/Rück Ausgangs Achsen einstellen. (nicht für LSTEP express)		
Delphi:	function LSX_SetAccelTVRO(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetAccelTVRO(double dX, double dY, double dZ, double dA);		
LabView:			
Parameter:	X, Y, Z, A	Beschleunigungswerte in U/s²	
Beispiel:	LSX.SetAccelTVRO(1.0, 1.5, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!tvroa	-

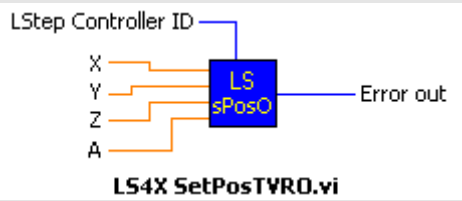
LSX_SetAccelSingleAxisTVRO			
Beschreibung:	Funktion zum Einstellen der Beschleunigung für eine einzelne TVRO Achse. (nicht für LSTEP express)		
Delphi:	function LSX_SetAccelSingleAxisTVRO(LSID: Integer; Axis: Integer; Accel: Double): Integer;		
C++:	int SetAccelSingleAxisTVRO(int lAxis, double dAccel);		
LabView:			
Parameter:	Axis	Achse deren Beschleunigung gesetzt werden soll TVRO X = 1 TVRO Y = 2 …	
	Accel	Einzustellende Beschleunigung in U/s²	
Beispiel:	LSX.SetAccelSingleAxis(2, 50.0); // Die Z-Achse soll wird mit 50 U/s² beschleunigt		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!tvroa	-

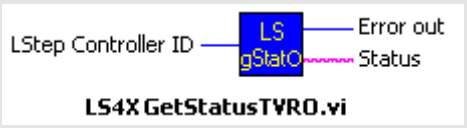
LSX_GetVelTVRO			
Beschreibung:	Liest die eingestellten Geschwindigkeiten für die TVRO Achsen aus der Steuerung aus. (nicht für LSTEP express)		
Delphi:	function LSX_GetVelTVRO(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetVelTVRO(double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	X, Y, Z, A	Geschwindigkeitswerte in U/s	
Beispiel:	LSX.GetVelTVRO(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?tvrov	-

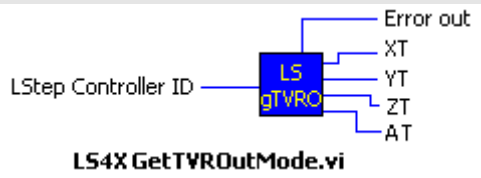
LSX_SetVelTVRO			
Beschreibung:	Funktion zum Einstellen der Geschwindigkeit für die TVRO Achsen. (nicht für LSTEP express)		
Delphi:	function LSX_SetVelTVRO(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetVelTVRO(double dX, double dY, double dZ, double dA);		
LabView:			
Parameter:	X, Y, Z, A	Geschwindigkeiten in U/s	
Beispiel:	LSX.SetVelTVRO(1.0, 1.5, 0, 0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!tvrov	-

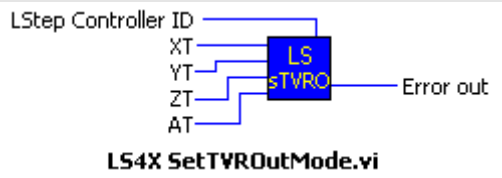
LSX_SetVelSingleAxisTVRO			
Beschreibung:	Funktion zum Einstellen der Geschwindigkeit für eine einzelne TVRO Achse. (nicht für LSTEP express)		
Delphi:	function LSX_SetVelSingleAxisTVRO(LSID: Integer; Axis: Integer; Vel: Double): Integer;		
C++:	int SetVelSingleAxisTVRO(int lAxis, double dVel);		
LabView:			
Parameter:	Axis	Achse deren Geschwindigkeit gesetzt werden soll TVRO X = 1 TVRO Y = 2 …	
	Vel	Einzustellender Geschwindigkeitswert in U/s	
Beispiel:	LSX.SetVelSingleAxis(1, 10.0); // Die X-Achse soll mit einer max. Geschwindigkeit von 10 U/s betrieben werden		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!tvrov	-

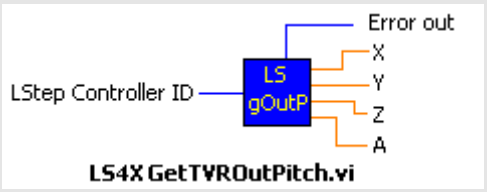
LSX_GetPosTVRO			
Beschreibung:	Funktion zum Auslesen der Position der zusätzlichen TVRO Achsen (nicht für LSTEP express)		
Delphi:	function LSX_GetPosTVRO(LSID: Integer; var dX, dY, dZ, dA: Double): Integer;		
C++:	int GetPosTVRO (double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	dX, dY, dZ, dA	Positionswert in abhängig von der eingestellten Dimension	
Beispiel:	LSX.GetPosTVRO(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?tvropos	-

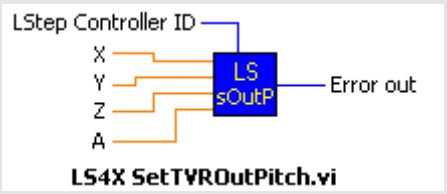
LSX_SetPosTVRO			
Beschreibung:	Position der zusätzlichen TVRO Achsen setzen. (nicht für LSTEP express)		
Delphi:	function LSX_SetPosTVRO(LSID: Integer; dX, dY, dZ, dA: Double): Integer;		
C++:	int SetPosTVRO(double dX, double dY, double dZ, double dA);		
LabView:			
Parameter:	dX, dY, dZ, dA	Positionswert, in Abhängigkeit von der eingestellten Dimension	
Beispiel:	LSX.SetPosTVRO(10.0, 5.0, 0.0, 0.0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!tvropos	-

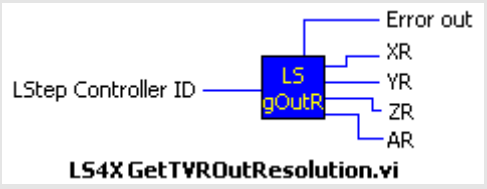
LSX_GetStatusTVRO, LSX_GetStatusTVROW			
Beschreibung:	Liefert den aktuellen Achsstatus der zusätzlichen TVRO Achsen (nicht für LSTEP express)		
Delphi:	function LSX_GetStatusTVRO(LSID: Integer; pcStat: PAnsiChar; MaxLen: Integer): Integer; function LSX_GetStatusTVRO(LSID: Integer; pcStat: PWideChar; MaxLen: Integer): Integer;		
C++:	int GetStatusTVRO(char *pcStat, int lMaxLen); int GetStatusTVROW(TCHAR *pcStat, int lMaxLen);		
LabView:			
Parameter:	pcStat	Zeiger auf einen Puffer, in dem der Statusstring zurückgegeben wird. Die Achsmaske enthält eines der folgenden Zeichen: @ = Achse steht M = Achse ist in Bewegung (Motion) - = Achse ist nicht freigegeben	
	MaxLen	Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen	
Beispiel:	LSX.GetStatusTVRO(pcStat, 256); // Die zusätzliche Z-Achse um 5mm in positiver Richtung verfahren		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?tvrostatus	-

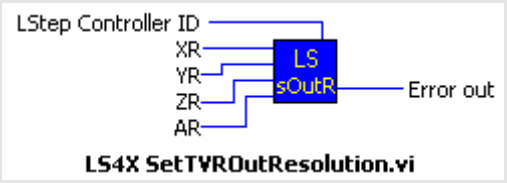
LSX_GetTVROutMode			
Beschreibung:	Funktion um den eingestellten Modus vom Takt Vor/Rück Ausgang zu lesen. (nicht für LSTEP express)		
Delphi:	function LSX_GetTVROutMode(LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetTVROutMode(int *plXT, int *plYT, int *plZT, int *plAT);		
LabView:			
Parameter:	X, Y, Z, A	Eingestellter TVRO Modus 0 = Takt Vor/Rück ist ausgeschaltet 1 = Takt Vor/Rück ist eingeschaltet	
Beispiel:	LSX.GetTVROutMode(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?tvrout	-

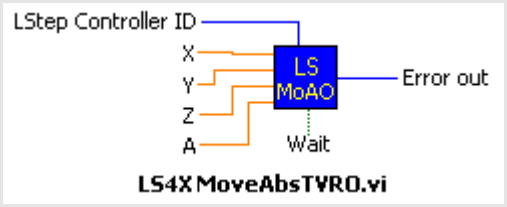
LSX_SetTVROutMode			
Beschreibung:	Funktion um den eingestellten Modus des Takt Vor/Rück Ausgangs zu setzen. (nicht für LSTEP express)		
Delphi:	function LSX_SetTVROutMode(LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetTVROutMode(int lXT, int lYT, int lZT, int lAT);		
LabView:			
Parameter:	X, Y, Z, A	Einzustellender TVRO Modus 0 = Takt Vor/Rück ist ausgeschaltet 1 = Takt Vor/Rück ist eingeschaltet	
Beispiel:	LSX.SetTVROutMode(1, 0, 1, 0); //Bei Achsen x und z soll Takt Vor/Rück aktiviert werden, bei y und a – deaktiviert.		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!tvrout	-

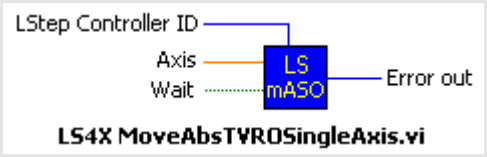
LSX_GetTVROutPitch			
Beschreibung:	Liest die Spindelsteigungen für die zusätzlichen TVRO Achsen ein. (nicht für LSTEP express)		
Delphi:	function LSX_GetTVROutPitch (LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetTVROutPitch (double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	X, Y, Z A	Spindelsteigungen in mm/U	
Beispiel:	LSX.GetTVROutPitch(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?tvropitch	-

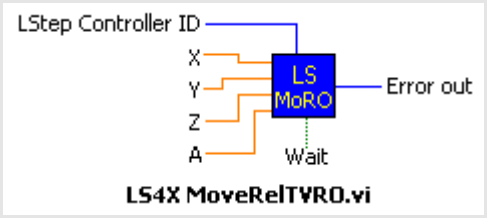
LSX_SetTVROutPitch			
Beschreibung:	Setzt die Spindelsteigungen für die zusätzlichen Achsen (nicht für LSTEP express)		
Delphi:	function LSX_SetTVROutPitch(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetTVROutPitch(double dX, double dY, double dZ, double dA);		
LabView:			
Parameter:	X, Y, Z, A	Spindelsteigungen in mm/U	
Beispiel:	LSX.SetTVROutPitch(1.0, 4.0, 1.0, 1.0); /* Spindelsteigung für y-Achse beträgt 4 mm. Für x-, z- und a-Achsen werden Spindeln mit 1mm Steigung eingesetzt*/		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!tvropitch	-

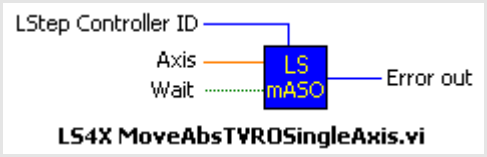
LSX_GetTVROutResolution			
Beschreibung:	Liest die in der LSTEP eingestellte Auflösung für die anzusteuernde Endstufe aus. (nicht für LSTEP express)		
Delphi:	function LSX_GetTVROutResolution(LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetTVROutResolution(int *plX, int *plY, int *plZ, int *plA);		
LabView:			
Parameter:	X, Y, Z, A	Eingestellte Auflösung in Impulsen/Umdrehung	
Beispiel:	LSX.GetTVROutResolution(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?tvrores	-

LSX_SetTVROutResolution			
Beschreibung:	Schreibt die Auflösung in die LSTEP-Steuerung mit der die externe Endstufe angesteuert werden soll. (nicht für LSTEP express)		
Delphi:	function LSX_SetTVROutResolution (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetTVROutResolution(int lX, int lY, int lZ, int lA);		
LabView:			
Parameter:	X, Y, Z, A	Einzustellende Auflösung in Impulse/Umdrehung	
Beispiel:	LSX.SetTVROutResolution(1000, 1000, 0, 0); /* Bei Achse X und Y wird eine Auflösung von 1000 Impulsen pro Umdrehung eingestellt*/		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!tvrores	-

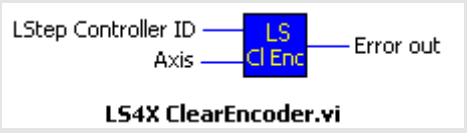
LSX_MoveAbsTVRO			
Beschreibung:	Absolutposition mit den TVRO Achsen von der aktuellen Position aus anfahren. (nicht für LSTEP express)		
Delphi:	function LSX_MoveAbsTVRO(LSID: Integer; X, Y, Z, A: Double; Wait: LongBool): Integer;		
C++:	int MoveAbsTVRO(double dX, double dY, double dZ, double dA, BOOL bWait);		
LabView:			
Parameter:	X, Y, Z, A	Absolute Positionsangabe in der eingestellen Dimension der Achse	
	Wait	Gibt an, ob die Funktion nachdem die Position erreicht wurde oder direkt zurückkehren soll True = Warte auf Erreichen der Position False = Warte nicht auf das Erreichen der Position	
Beispiel:	LSX.MoveAbsTVRO(10.0, 10.0, 10.0, 10.0, true);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!tvromoa	-

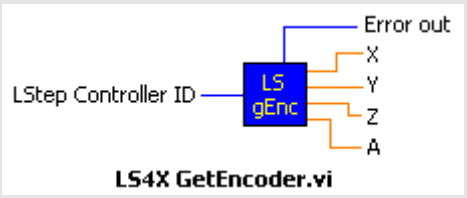
LSX_MoveAbsTVROSingleAxis			
Beschreibung:	Einzelne TVRO Achse absolut positionieren (nicht für LSTEP express)		
Delphi:	function LSX_MoveAbsTVROSingleAxis(LSID: Integer; Axis: Integer; Value: Double; Wait: LongBool): Integer;		
C++:	int MoveAbsTVROSingleAxis(int lAxis, double dValue, BOOL bWait);		
LabView:			
Parameter:	Axis	Achse die verfahren werden soll 1 = X-Achse 2 = Y-Achse …	
	Value	Position (Eingabe ist abhängig von der eingestellten Dimension)	
Beispiel:	LSX.MoveAbsTVROSingleAxis(2, 10.0); //Die zusätzliche Y-Achse auf 10mm absolut positionieren		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!tvromoa	-

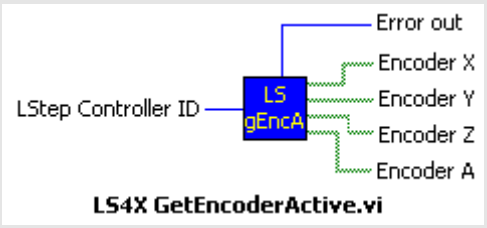
LSX_MoveRelTVRO			
Beschreibung:	Funktion zum Fahren eines relativen Vektors von der aktuellen Position. (nicht für LSTEP express)		
Delphi:	function LSX_MoveRelTVRO(LSID: Integer; X, Y, Z, A: Double; Wait: LongBool): Integer;		
C++:	int MoveRelTVRO(double dX, double dY, double dZ, double dR, BOOL bWait);		
LabView:			
Parameter:	X, Y, Z, A	Relative Positionsangabe in der eingestellten Dimension der Achse	
	Wait	Warte auf das Ende der Bewegung True = Warte False = Warte nicht	
Beispiel:	LSX.MoveRelTVRO(10.0, 10.0, 10.0, 10.0, true);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)

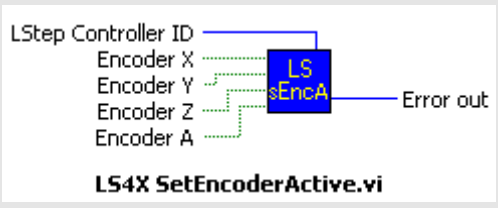
	1	!tvromor	-
LSX_MoveRelTVROSingleAxis			
Beschreibung:	Funktion zum relativ Verfahren einer einzelnen TVRO Achse. (nicht für LSTEP express)		
Delphi:	function LSX_MoveRelTVROSingleAxis(LSID: Integer; Axis: Integer; Value: Double; Wait: LongBool): Integer;		
C++:	int MoveRelTVROSingleAxis(int lAxis,double dValue,BOOL Wait);		
LabView:			
Parameter:	Axis	Achse die Verfahren werden soll TVRO X = 1 TVRO Y = 2 …	
	Value	Relativposition in der eingestellten Dimension der Achse	
	Wait	Warte auf das Ende der Bewegung True = Warte False = Warte nicht	
Beispiel:	LSX.MoveRelTVROSingleAxis(3, 5.0); // Die zusätzliche Z-Achse um 5mm in positiver Richtung verfahren		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!tvromor	-

4.2.16 Geber-Einstellungen

LSX_ClearEncoder			
Beschreibung:	Funktion zum Setzen des Geber-Zähler auf den Wert Null. (nicht für LSTEP express)		
Delphi:	function LSX_ClearEncoder(LSID: Integer; lAxis: Integer): Integer;		
C++:	int ClearEncoder(int lAxis);		
LabView:			
Parameter:	lAxis	Achse dessen Geberwerte auf Null gesetzt werden sollen X = 1 Y = 2 …	
Beispiel:	LSX.ClearEncoder (2); //Geber-Zähler der y-Achse auf null setzen		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!clearhwcount	-

LSX_GetEncoder			
Beschreibung:	Diese Funktion liest alle Geber-Positionen aus der Steuerung ein. (nicht für LSTEP express)		
Delphi:	function LSX_GetEncoder(LSID: Integer; XP, YP, ZP, AP: Double): Integer;		
C++:	int GetEncoder(double *pdXP, double *pdYP, double *pdZP, double *pdAP);		
LabView:			
Parameter:	XP, YP, ZP, AP	Anzahl der Encoderinkremente, 4-fach interpoliert	
Beispiel:	LSX.GetEncoder(&XP, &YP, &ZP, &AP);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?hwcount	-

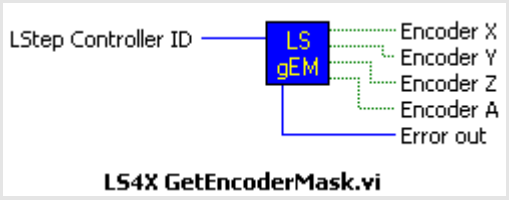
LSX_GetEncoderActive			
Beschreibung:	Liest welche Geber nach dem Kalibrieren aktiviert werden. (nicht für LSTEP express)		
Delphi:	function LSX_GetEncoderActive(LSID: Integer; var Flags: Integer): Integer;		
C++:	int GetEncoderActive(int *plFlags);		
LabView:			
Parameter:	Flags	32-Bit-Integer, mit einer Bit-Maske in den Bits 0-4 Bit 0 = X-Achse Bit 1 = Y-Achse … Wert 0 = Encoder soll aktivert werden Wert 1 = Encoder soll nicht aktiviert werden	
Beispiel:	LSX.GetEncoderActive(&Flags);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?encmask	-

LSX_SetEncoderActive	
Beschreibung:	Mit dieser Funktion kann ausgewählt werden, welche Geber nach dem Kalibrieren aktiviert werden sollen.
Delphi:	function LSX_SetEncoderActive(LSID: Integer; Flags: Integer): Integer;
C++:	int SetEncoderActive(int lFlags);
LabView:	

Parameter:	Flags	32-Bit-Integer, mit einer Bit-Maske in den Bits 0-4 Bit 0 = X-Achse Bit 1 = Y-Achse … Wert 0 = Encoder soll nicht aktiviert werden nach dem Kalibrieren Wert 1 = Encoder soll aktivert werden nach dem Kalibrieren	

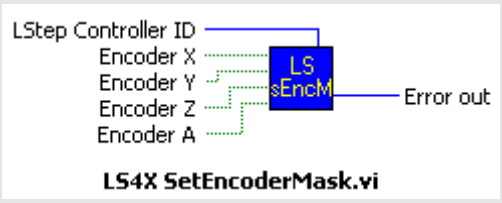
Beispiel:	LSX.SetEncoderActive(0);
	// Alle Geber deaktivieren
	LSX.SetEncoderMask(2);
	// Geber Y-Achse aktivieren		

Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?encmask	-

LSX_GetEncoderMask			
Beschreibung:	Funktion zum Auslesen der Geberaktivierung		
Delphi:	function LSX_GetEncoderMask(LSID: Integer; var Flags: Integer): Integer;		
C++:	int GetEncoderMask(int *plFlags);		
LabView:			
Parameter:	Flags	32-Bit-Integer, mit einer Bit-Maske in den Bits 0-4 Bit 0 = X-Achse Bit 1 = Y-Achse … Wert 0 = Encoder wurde nicht erkannt bzw. ist nich aktiv Wert 1 = Encoder wurde erkannt bzw. ist aktiv	

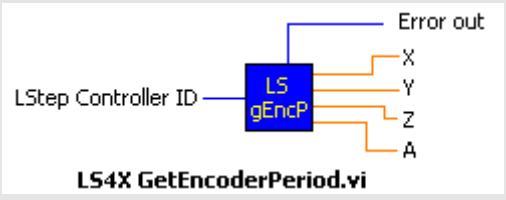
Beispiel:	int EncMask;
	LSX.GetEncoderMask(&EncMask);
	if (EncMask & 2) ...
	// Wenn Geber Y-Achse angeschlossen+aktiv		

Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?enc	-

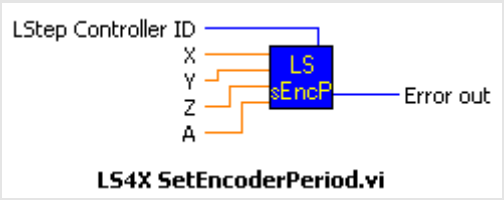
LSX_SetEncoderMask			
Beschreibung:	Funktion zum Geber (de-)aktivieren. (nicht für LSTEP express)		
Delphi:	function LSX_SetEncoderMask(LSID: Integer; Value: Integer): Integer;		
C++:	int SetEncoderMask(int lValue);		
LabView:			
Parameter:	Value	32-Bit-Integer, mit einer Bit-Maske in den Bits 0-4 Bit 0 = X-Achse Bit 1 = Y-Achse … Wert 0 = Encoder deaktivieren Wert 1 = Encoder aktivieren	

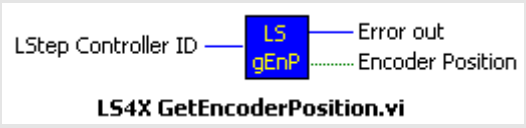
Beispiel:	LSX.SetEncoderMask(0);
	// Alle Geber deaktivieren
	LSX.SetEncoderMask(2);
	// Geber Y-Achse aktivieren		

Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!enc	-

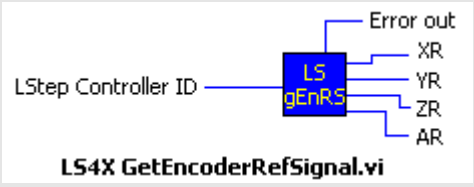
LSX_GetEncoderPeriod	
Beschreibung:	Funktion zum Auslesen der Geberperiodenlängen.
Delphi:	function LSX_GetEncoderPeriod(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetEncoderPeriod(double *pdX, double *pdY, double *pdZ, double *pdR);
LabView:	

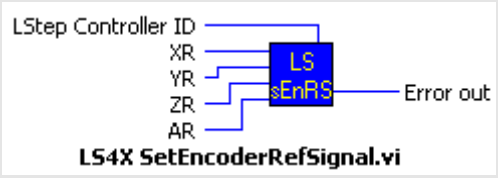
Parameter:	X, Y, Z, A	Eingestellte Periodenlängen LSTEP 2000 Serie = m/s LSTEP express Serie = Eingestellte Dimension/s	
Beispiel:	LSX.GetEncoderPeriod(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?encperiod	-

LSX_SetEncoderPeriod			
Beschreibung:	Funktion zum Einstellen der Geberperiodenlängen.		
Delphi:	function LSX_SetEncoderPeriod(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetEncoderPeriod(double dX,double dY,double dZ,double dA);		
LabView:			
Parameter:	X, Y, Z, A	Einzustellende Periodenlängen LSTEP 2000 Serie = m/s LSTEP express Serie = Eingestellte Dimension/s	
Beispiel:	LSX.SetEncoderPeriod(0.1, 0.1, 0.1, 0.1); // Geberperiodenlänge aller Achsen ist 0.1mm		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!encperiod	ValidConfig

LSX_GetEncoderPosition			
Beschreibung:	Einstellung der Quelldaten der Positionsanzeige lesen.		
Delphi:	function LSX_GetEncoderPosition(LSID: Integer; Value: LongBool): Integer;		
C++:	int GetEncoderPosition(BOOL *pbValue);		
LabView:			
Parameter:	Value	Quelle der Positionsanzeige True = Geberwerte bei Positionsabfrage verwenden False = Keine Geberwerte bei Positionsabfrage verwenden	
Beispiel:	LSX.GetEncoderPosition(&Value);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?encpos	-

LSX_SetEncoderPosition			
Beschreibung:	Einstellung der Quelldaten der Positionsanzeige setzen.		
Delphi:	function LSX_SetEncoderPosition(LSID: Integer; Value: LongBool): Integer;		
C++:	int SetEncoderPosition(BOOL fValue);		
LabView:			
Parameter:	Value	Quelle der Positionsanzeige True = Geberwerte bei Positionsabfrage verwenden False = Keine Geberwerte bei Positionsabfrage verwenden	
Beispiel:	LSX.SetEncoderPosition(true);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!encpos	-

LSX_GetEncoderRefSignal			
Beschreibung:	Liest, ob beim Kalibrieren das Referenzsignal des Gebers ausgewertet wird		
Delphi:	function LSX_GetEncoderRefSignal(LSID: Integer; var XR, YR, ZR, AR: Integer): Integer;		
C++:	int GetEncoderRefSignal(int *plXR, int *plYR, int *plZR, int *plAR);		
LabView:			
Parameter:	X, Y, Z, A	Eingestellter Wert 1 = Beim Kalibrieren wird das Referenzsignal ausgewertet 0 = Das Referenzsignal wird nicht ausgewertet	
Beispiel:	LSX.GetEncoderRefSignal(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?encref	-

LSX_SetEncoderRefSignal			
Beschreibung:	Funktion um Auswertung des Referenzsignals eines Gebers zu aktivieren.		
Delphi:	function LSX_SetEncoderRefSignal(LSID: Integer; XR, YR, ZR, AR: Integer): Integer;		
C++:	int SetEncoderRefSignal(int lXR,int lYR,int lZR,int lAR);		
LabView:			
Parameter:	X, Y, Z, A	Eingestelter Wert 1 = Beim Kalibrieren wird das Referenzsignal ausgewertet 0 = Das Referenzsignal wird nicht ausgewertet	
Beispiel:	LSX.SetEncoderRefSignal(1, 1, 0, 0); /* Beim Kalibrieren wird das Referenzsignal der Geber x und y ausgewertet. */		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!encref	-

LSX_GetEncDir			
Beschreibung:	Fragt die Lagegeber-Zählrichtung ab.		
Delphi:	function LSX_GetEncDir(LSID: Integer; var X,Y,Z,A LongBool): Integer;		
C++:	int GetEncDir (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestelter Wert True = Drehrichtungswechsel False = Kein Drehrichtungswechsel	
Beispiel:	LSX.GetEncDir(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?encdir	-

LSX_SetEncDir			
Beschreibung:	Stellt die Lagegeber-Zählrichtung ein.		
Delphi:	function LSX_SetEncDir(LSX: Integer; X,Y,Z,A: LongBool): Integer;		
C++:	int SetEncDir (BOOL bX, BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestelter Wert True = Drehrichtungswechsel False = Kein Drehrichtungswechsel	
Beispiel:	LSX.SetEncDir(True,True,False,False); /* Drehrichtungswechsel bei erster und zweiter Achse. */		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!encdir	-

LSX_GetEncPolePairs			
Beschreibung:	Fragt die Lagegeberpolpaare pro Umdrehung ab.		
Delphi:	function LSX_GetEncPolePairs(LSID:Integer; var X,Y,Z,A: Integer): Integer;		
C++:	int GetEncPolePairs (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Lagegeberpolpaare pro Umdrehung	
Beispiel:	LSX.GetEncPolePairs(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?encpolepairs	-

LSX_SetEncPolePairs			
Beschreibung:	Stellt die Lagegeberpolpaare pro Umdrehung ein.		
Delphi:	function LSX_SetEncPolePairs(LSID:Integer;X,Y,Z,A: Integer): Integer;		
C++:	int SetEncPolePairs (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Lagegeberpolpaare pro Umdrehung	
Beispiel:	LSX.SetEncPolePairs(50,50,50,50);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!encpolepairs	-

LSX_GetEnctoAxis		
Beschreibung:	Fragt die Zuordnung der Gebereingänge zu den Lagegeberauswertungen der Achsen ab (Lageregler 1).	
Delphi:	function LSX_GetEnctoAxis(LSID:Integer; var X,Y,Z,A: Integer): Integer;	
C++:	int GetEnctoAxis (int *plX, int *plY, int *plZ, int *plA);	
LabView:	Nicht unterstützt	
Parameter:	X, Y, Z, A	Eingestelter Wert 0 = kein Gebereingang 1 = QEP1 (TTL-Pegel, MFP) 2 = QEP2 (TTL-Pegel, MFP) 3 = QEP3 (TTL- bzw. RS422- Pegel, optional) 4 = QEP4 (TTL- bzw. RS422- Pegel, optional) 5 = QEP5 (TTL- bzw. RS422- Pegel, optional) 6 = QEP6 (TTL- bzw. RS422- Pegel, optional) 7 = ENC1 (Sinus-/Cosinus-Signale, optional) 8 = ENC2 (Sinus-/Cosinus-Signale, optional) 9 = ENC3 (Sinus-/Cosinus-Signale, optional) 10 = ENC4 (Sinus-/Cosinus-Signale, optional) 11 = ENC5 (Sinus-/Cosinus-Signale, optional) 12 = ENC6 (Sinus-/Cosinus-Signale, optional) 13 = ENC7 (Sinus-/Cosinus-Signale, optional) 14 = ENC8 (Sinus-/Cosinus-Signale, optional)
Beispiel:	LSX.GetEnctoAxis(&X,&Y,&Z,&A);	
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?enctoaxis	-

LSX_SetEnctoAxis			
Beschreibung:	Stellt die Zuordnung der Gebereingänge zu den Lagegeberauswertungen der Achsen ein (Lageregler 1).		
Delphi:	function LSX_SetEnctoAxis(LSID:Integer;X,Y,Z,A: Integer): Integer;		
C++:	int SetEnctoAxis (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestelter Wert 0 = kein Gebereingang 1 = QEP1 (TTL-Pegel, MFP) 2 = QEP2 (TTL-Pegel, MFP) 3 = QEP3 (TTL- bzw. RS422- Pegel, optional) 4 = QEP4 (TTL- bzw. RS422- Pegel, optional) 5 = QEP5 (TTL- bzw. RS422- Pegel, optional) 6 = QEP6 (TTL- bzw. RS422- Pegel, optional) 7 = ENC1 (Sinus-/Cosinus-Signale, optional) 8 = ENC2 (Sinus-/Cosinus-Signale, optional) 9 = ENC3 (Sinus-/Cosinus-Signale, optional) 10 = ENC4 (Sinus-/Cosinus-Signale, optional) 11 = ENC5 (Sinus-/Cosinus-Signale, optional) 12 = ENC6 (Sinus-/Cosinus-Signale, optional) 13 = ENC7 (Sinus-/Cosinus-Signale, optional) 14 = ENC8(Sinus-/ Cosinus-Signale, optional)	
Beispiel:	LSX.SetEnctoAxis(0,0,0,0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!enctoaxis	-

LSX_GetEncType	
Beschreibung:	Fragt die Lagegebertyp ab.
Delphi:	function LSX_GetEncType(LSID:Integer; var X,Y,Z,A: Integer): Integer;
C++:	int GetEncType (int *plX, int *plY, int *plZ, int *plA);
LabView:	Nicht unterstützt

Parameter:	X, Y, Z, A	Eingestelter Wert: 0 = kein Lagegeber 1 = motorseitiges rotatives 1Vss-Gebersystem 2 = motorseitiges rotatives 11µA-Gebersystem 3 = motorseitiges rotatives MR- Gebersystem 4 = motorseitiges rotatives TTL- Gebersystem 5 = lineares 1Vss- Gebersystem 6 = lineares 11µA- Gebersystem 7 = lineares MR- Gebersystem 8 = lineares TTL- Gebersystem 9 = abtriebsseitiges rotatives 1Vss- Gebersystem 10 = abtriebsseitiges rotatives 11µA- Gebersystem 11 = abtriebsseitiges rotatives MR- Gebersystem 12 = abtriebsseitiges rotatives TTL- Gebersystem
Beispiel:	LSX.GetEncType (&X,&Y,&Z,&A);	
Sonstiges:	Kompatibilität	"SendString" Befehl / Aktivierung (LSTEP express Serie)
	2	?enctype / -

LSX_SetEncType		
Beschreibung:	Stellt die Lagegebertyp ein.	
Delphi:	function LSX_SetEncType(LSID:Integer;X,Y,Z,A: Integer): Integer;	
C++:	int SetEncType (int lX, int lY, int lZ, int lA) ;	
LabView:	Nicht unterstützt	
Parameter:	X, Y, Z, A	Eingestelter Wert: 0 = kein Lagegeber 1 = motorseitiges rotatives 1Vss-Gebersystem 2 = motorseitiges rotatives 11µA-Gebersystem 3 = motorseitiges rotatives MR- Gebersystem 4 = motorseitiges rotatives TTL- Gebersystem 5 = lineares 1Vss- Gebersystem 6 = lineares 11µA- Gebersystem 7 = lineares MR- Gebersystem 8 = lineares TTL- Gebersystem 9 = abtriebsseitiges rotatives 1Vss- Gebersystem 10 = abtriebsseitiges rotatives 11µA- Gebersystem 11 = abtriebsseitiges rotatives MR- Gebersystem 12 = abtriebsseitiges rotatives TTL- Gebersystem
Beispiel:	LSX.SetEnctype(0,1,2,3);	
Sonstiges:	Kompatibilität	"SendString" Befehl / Aktivierung (LSTEP express Serie)
	2	!enctype / -

LSX_GetKommMode			
Beschreibung:	Fragt den Kommutierungsmodus im Servobetrieb ab.		
Delphi:	function LSX_GetKommMode(LSID:Integer; var X,Y,Z,A:LongBool): Integer;		
C++:	int GetEncType (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestelter Wert: False = Seperaten Gebereingang zur Kommutierung verwenden True = Lagegeberwerte zur Kommutierung verwenden	
Beispiel:	LSX.GetKommMode (&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?kommmode	-

LSX_SetKommMode			
Beschreibung:	Stellt den Kommutierungsmodus im Servobetrieb ein.		
Delphi:	function LSX_SetKommMode (LSID:Integer;X,Y,Z,A:LongBool): Integer;		
C++:	int SetKommMode (int lX, int lY, int lZ, int lA) ;		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestelter Wert: False = Seperaten Gebereingang zur Kommutierung verwenden True = Lagegeberwerte zur Kommutierung verwenden	
Beispiel:	LSX.SetKommMode (False,False,False,False);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!kommmode	-

LSX_GetKommEncDir			
Beschreibung:	Fragt die Kommutierungsgeber Zählrichtung ab.		
Delphi:	function LSX_GetKommEncDir (LSID:Integer; var X,Y,Z,A:LongBool): Integer;		
C++:	int GetKommEncDir (BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestelter Wert: False = Kein Drehrichtungswechsel True = Drehrichtungswechsel	
Beispiel:	LSX.GetKommEncDir(&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?kommencdir	-

LSX_SetKommEncDir			
Beschreibung:	Stellt die Kommutierungsgeber Zählrichtung ein.		
Delphi:	function LSX_SetKommEncDir (LSID:Integer;X,Y,Z,A:LongBool): Integer;		
C++:	int SetKommEncDir (BOOL bX, BOOL bY,BOOL bZ,BOOL bA) ;		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestelter Wert: False = Kein Drehrichtungswechsel True = Drehrichtungswechsel	
Beispiel:	LSX.KommEncDir(True,True,True,True);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!kommencdir	-

LSX_GetKommEncPolePairs			
Beschreibung:	Fragt die Kommutierungsgeber Signalperiode ab		
Delphi:	function LSX_GetKommEncPolePairs (LSID:Integer; var X,Y,Z,A: Integer): Integer;		
C++:	int GetKommEncPolePairs (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Kommutierungsgeberpolpaare pro Umdrehung	
Beispiel:	LSX.GetKommEncPolePairs (&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?kommencpolepairs	-

LSX_SetKommEncPolePairs			
Beschreibung:	Stellt die Kommutierungsgeber Signalperiode ein.		
Delphi:	function LSX_SetKommEncPolePairs (LSID:Integer;X,Y,Z,A: Integer): Integer;		
C++:	int SetKommEncPolePairs (int lX, int lY, int lZ, int lA) ;		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestelter Wert: Kommutierungsgeberpolpaare pro Umdrehung	
Beispiel:	LSX.SetKommEncPolePairs (50,10,2500,4000);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!kommencpolepairs	-

LSX_GetKommEnctoAxis			
Beschreibung:	Fragt die Zuordnung der Gebereingänge zu den Kommutierungsgeberauswertungen der Achsen ab.		
Delphi:	function LSX_GetKommEnctoAxis (LSID:Integer; var X,Y,Z,A: Integer): Integer;		
C++:	int GetKommEnctoAxis (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestelter Wert: 0 = kein Gebereingang 1 = QEP1 (TTL-Pegel, MFP) 2 = QEP2 (TTL-Pegel, MFP) 3 = QEP3 (TTL- bzw. RS422- Pegel, optional) 4 = QEP4 (TTL- bzw. RS422- Pegel, optional) 5 = QEP5 (TTL- bzw. RS422- Pegel, optional) 6 = QEP6 (TTL- bzw. RS422- Pegel, optional) 7 = ENC1 (Sinus- /Cosinus-Signale, optional) 8 = ENC2 (Sinus- /Cosinus-Signale, optional) 9 = ENC3 (Sinus- /Cosinus-Signale, optional) 10 = ENC4 (Sinus- /Cosinus-Signale, optional) 11 = ENC5 (Sinus- /Cosinus-Signale, optional) 12 = ENC6 (Sinus- /Cosinus-Signale, optional) 13 = ENC7 (Sinus- /Cosinus-Signale, optional) 14 = ENC8 (Sinus- /Cosinus-Signale, optional)	
Beispiel:	LSX.GetKommEnctoAxis (&X,&Y,&Z,&A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?kommenctoaxis	-

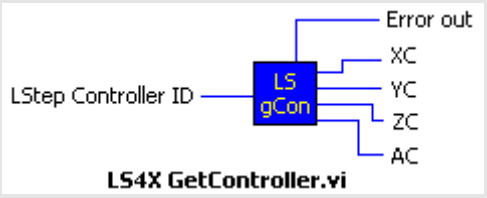
LSX_SetKommEnctoAxis			
Beschreibung:	Ordnet die Gebereingänge zu den Kommutierungsgeberauswertungen der Achsen zu.		
Delphi:	function LSX_SetKommEnctoAxis (LSID:Integer;X,Y,Z,A: Integer): Integer;		
C++:	int SetKommEnctoAxis (int lX, int lY, int lZ, int lA) ;		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestelter Wert: 0 = kein Gebereingang 1 = QEP1 (TTL-Pegel, MFP) 2 = QEP2 (TTL-Pegel, MFP) 3 = QEP3 (TTL- bzw. RS422- Pegel, optional) 4 = QEP4 (TTL- bzw. RS422- Pegel, optional) 5 = QEP5 (TTL- bzw. RS422- Pegel, optional) 6 = QEP6 (TTL- bzw. RS422- Pegel, optional) 7 = ENC1 (Sinus- /Cosinus-Signale, optional) 8 = ENC2 (Sinus- /Cosinus-Signale, optional) 9 = ENC3 (Sinus- /Cosinus-Signale, optional) 10 = ENC4 (Sinus- /Cosinus-Signale, optional) 11 = ENC5 (Sinus- /Cosinus-Signale, optional) 12 = ENC6 (Sinus- /Cosinus-Signale, optional) 13 = ENC7 (Sinus- /Cosinus-Signale, optional) 14 = ENC8 (Sinus- /Cosinus-Signale, optional)	
Beispiel:	LSX.SetKommEnctoAxis (0,1,2,3);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!kommenctoaxis	-

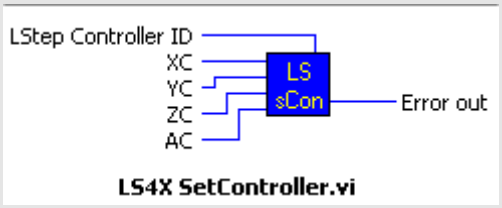
LSX_GetKommEncType	
Beschreibung:	Fragt den Kommutierungsgebertyp ab
Delphi:	function LSX_GetKommEncType(LSID:Integer; var X,Y,Z,A: Integer): Integer;
C++:	int GetKommEncType (int *plX, int *plY, int *plZ, int *plA);
LabView:	Nicht unterstützt

Parameter:	X, Y, Z, A	Eingestelter Wert: 0 = kein Lagegeber 1 = motorseitiges rotatives 1Vss-Gebersystem 2 = motorseitiges rotatives 11µA-Gebersystem 3 = motorseitiges rotatives MR- Gebersystem 4 = motorseitiges rotatives TTL- Gebersystem 5 = lineares 1Vss- Gebersystem 6 = lineares 11µA- Gebersystem 7 = lineares MR- Gebersystem 8 = lineares TTL- Gebersystem 9 = abtriebsseitiges rotatives 1Vss- Gebersystem 10 = abtriebsseitiges rotatives 11µA- Gebersystem 11 = abtriebsseitiges rotatives MR- Gebersystem 12 = abtriebsseitiges rotatives TTL- Gebersystem
Beispiel:	LSX.GetKommEncType(&X,&Y,&Z,&A);	
Sonstiges:	Kompatibilität	"SendString" Befehl / Aktivierung (LSTEP express Serie)
	2	?kommenctype / -

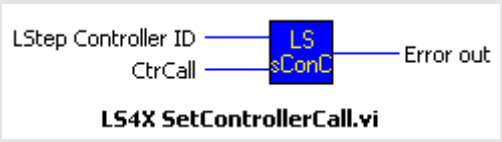
LSX_SetKommEncType		
Beschreibung:	Stellt den Kommutierungsgebertyp ein.	
Delphi:	function LSX_SetKommEncType(LSID:Integer;X,Y,Z,A: Integer): Integer;	
C++:	int SetKommEncType (int lX, int lY, int lZ, int lA) ;	
LabView:	Nicht unterstützt	
Parameter:	X, Y, Z, A	Eingestelter Wert: 0 = kein Lagegeber 1 = motorseitiges rotatives 1Vss-Gebersystem 2 = motorseitiges rotatives 11µA-Gebersystem 3 = motorseitiges rotatives MR- Gebersystem 4 = motorseitiges rotatives TTL- Gebersystem 5 = lineares 1Vss- Gebersystem 6 = lineares 11µA- Gebersystem 7 = lineares MR- Gebersystem 8 = lineares TTL- Gebersystem 9 = abtriebsseitiges rotatives 1Vss- Gebersystem 10 = abtriebsseitiges rotatives 11µA- Gebersystem 11 = abtriebsseitiges rotatives MR- Gebersystem 12 = abtriebsseitiges rotatives TTL- Gebersystem
Beispiel:	LSX.SetKommEnctype(0,1,2,3);	
Sonstiges:	Kompatibilität	"SendString" Befehl / Aktivierung (LSTEP express Serie)
	2	!kommenctype / -

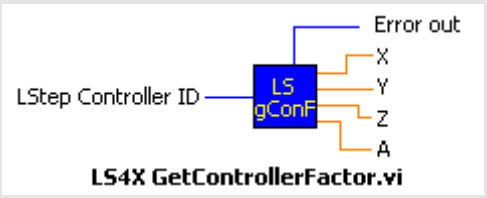
4.2.17 Reglereinstellungen

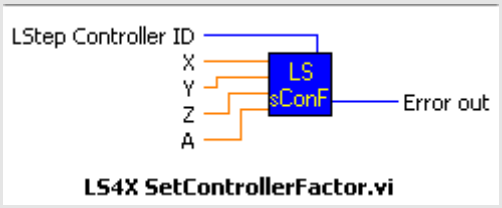
LSX_GetController			
Beschreibung:	Funktion zum Auslesen des Regler-Modus. (nicht für LSTEP express)		
Delphi:	function LSX_GetController(LSID: Integer; var XC, YC, ZC, AC: Integer): Integer;		
C++:	int GetController(int *plXC, int *plYC, int *plZC, int *plAC);		
LabView:			
Parameter:	X, Y, Z, A	Eingestellter Regler-Modus 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:	LSX.GetController(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	1	?ctr	-

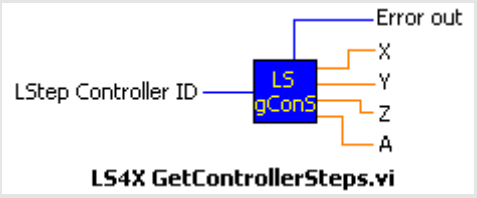
LSX_SetController			
Beschreibung:	Funktion zum Einstellen des Regler-Modus. (nicht für LSTEP express)		
Delphi:	function LSX_SetController(LSID: Integer; XC, YC, ZC, AC: Integer): Integer;		
C++:	int SetController(int lXC,int lYC,int lZC,int lAC);		
LabView:			
Parameter:	X, Y, Z, A	Einzustellender Regler-Modus 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:	LSX.SetController(1, 2, 0, 0);		
Sonstiges:	Kompatibilität 1	“SendString” Befehl !ctr	Aktivierung (LSTEP express Serie) -

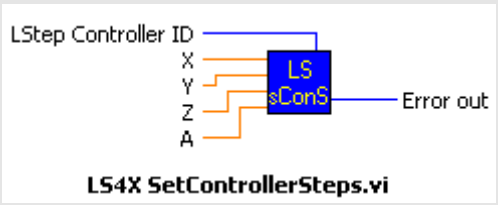
LSX_GetControllerCall			
Beschreibung:	Die Funktion liefert die eingestellte Regleraufrufzeit (nicht für LSTEP express)		
Delphi:	function LSX_GetControllerCall(LSID: Integer; var CtrCall: Integer): Integer;		
C++:	int GetControllerCall(int *plCtrCall);		
LabView:			
Parameter:	CtrCall	Regleraufrufzeit in ms	
Beispiel:	LSX.GetControllerCall(&CtrCall); // CtrCall = 10 bedeutet: Regleraufruf alle 10 ms		
Sonstiges:	Kompatibilität 1	“SendString” Befehl ?ctrc	Aktivierung (LSTEP express Serie) -

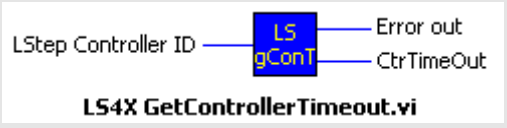
LSX_SetControllerCall			
Beschreibung:	Funktion zum Setzen der Regleraufrufzeit (nicht für LSTEP express)		
Delphi:	function LSX_SetControllerCall(LSID: Integer; CtrCall: Integer): Integer;		
C++:	int SetControllerCall(int lCtrCall);		
LabView:			
Parameter:	CtrCall	Regleraufrufzeit in ms	
Beispiel:	LSX.SetControllerCall(10);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!ctrc	-

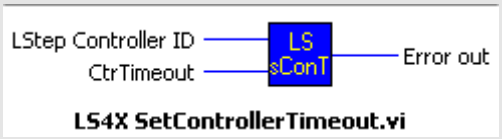
LSX_GetControllerFactor			
Beschreibung:	Funktion zum Auslesen der Regler-Faktoren. Siehe Dokumentation der LSTEP 2000 Serie. (nicht für LSTEP express)		
Delphi:	function LSX_GetControllerFactor(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetControllerFactor(double *pdX, double *pdY, double *pdZ, double *pdR);		
LabView:			
Parameter:	X, Y, Z, A	Eingestellte Regler-Faktoren	
Beispiel:	LSX.GetControllerFactor(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?ctrf	-

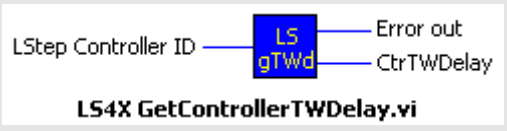
LSX_SetControllerFactor			
Beschreibung:	Funktion zum Einstellen der Regler-Faktoren. Siehe Dokumentation der LSTEP 2000 Serie. (nicht für LSTEP express)		
Delphi:	function LSX_SetControllerFactor(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetControllerFactor(double dX,double dY,double dZ,double dA);		
LabView:			
Parameter:	X, Y, Z, A	Einzustellende Regler-Faktoren	
Beispiel:	LSX.SetControllerFactor(X, Y, Z, A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!ctrf	-

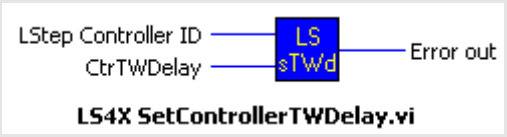
LSX_GetControllerSteps			
Beschreibung:	Diese Funktion liefert die eingestellte Regler-Schrittlänge. (nicht für LSTEP express)		
Delphi:	function LSX_GetControllerSteps(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetControllerSteps(double *pdX, double *pdY, double *pdZ, double *pdR);		
LabView:			
Parameter:	X, Y, Z, A	Reglerschrittlänge in der eingestellten Dimension der Achse	
Beispiel:	LSX.GetControllerSteps(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?ctrs	-

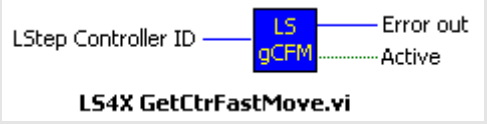
LSX_SetControllerSteps			
Beschreibung:	Funktion zum Einstellen der Regler-Schrittlänge. (nicht für LSTEP express)		
Delphi:	function LSX_SetControllerSteps(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetControllerSteps(double dX,double dY,double dZ,double dA);		
LabView:			
Parameter:	X, Y, Z, A	Reglerschrittlänge in der eingestellten Dimension der Achse	
Beispiel:	LSX.SetControllerSteps(4, 5, 7, 9);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!ctrs	-

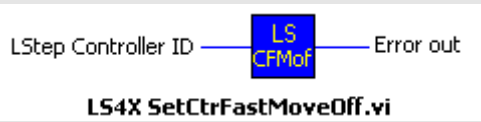
LSX_GetControllerTimeout			
Beschreibung:	Funktion zum Auslesen des Regler-Timeouts, nach dem ein Verfahrbefehl mit Fehlermeldung (Fehlercode 4013) zurückkehrt, wenn die Steuerung eine Position nicht endgültig finden konnte. (nicht für LSTEP express)		
Delphi:	function LSX_GetControllerTimeout(LSID: Integer; var ACtrTimeout: Integer): Integer;		
C++:	int GetControllerTimeout(int *plACtrTimeout);		
LabView:			
Parameter:	ACtrTimeout	Eingestellter Timeout in ms	
Beispiel:	LSX.GetControllerTimeout(&ACtrTimeout);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?ctrt	-

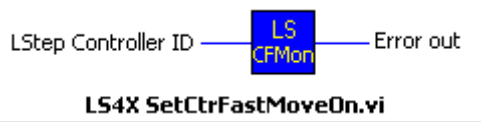
LSX_SetControllerTimeout			
Beschreibung:	Funktion zum Setzen des Regler-Timeouts, nach dem ein Verfahrbefehl mit Fehlermeldung (Fehlercode 4013) zurückkehrt, wenn die Steuerung eine Position nicht endgültig finden konnte. (nicht für LSTEP express)		
Delphi:	function LSX_SetControllerTimeout(LSID: Integer; ACtrTimeout: Integer): Integer;		
C++:	int SetControllerTimeout(int ACtrTimeout);		
LabView:			
Parameter:	ACtrTimeout	Einzustellender Timeout in ms	
Beispiel:	LSX.SetControllerTimeout(500);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!ctrt	-

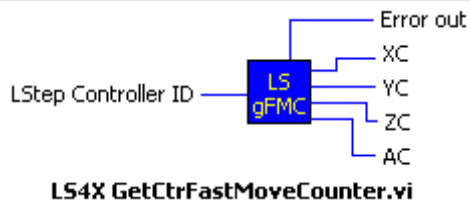
LSX_GetControllerTWDelay			
Beschreibung:	Eingestellte Regler-Verzögerung auslesen. (nicht für LSTEP express)		
Delphi:	function LSX_GetControllerTWDelay(LSID: Integer; var CtrTWDelay: Integer): Integer;		
C++:	int GetControllerTWDelay(int *plCtrTWDelay);		
LabView:			
Parameter:	CtrTWDelay	Regler-Verzögerung in ms	
Beispiel:	LSX.GetControllerTWDelay(&CtrTWDelay);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?ctrd	-

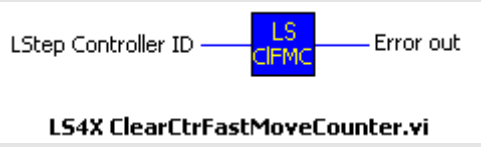
LSX_SetControllerTWDelay			
Beschreibung:	Funktion zum Setzen der Regler-Verzögerung. (nicht für LSTEP express)		
Delphi:	function LSX_SetControllerTWDelay(LSID: Integer; CtrTWDelay: Integer): Integer;		
C++:	int SetControllerTWDelay(int lCtrTWDelay);		
LabView:			
Parameter:	CtrTWDelay	Einzustellende Regler-Verzögerung in ms	
Beispiel:	LSX.SetControllerTWDelay(0); // Regler-Verzögerung Aus		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!ctrd	-

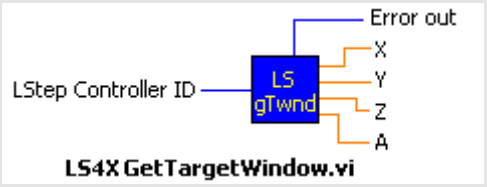
LSX_GetCtrFastMove			
Beschreibung:	Liest die Einstellung von der Fast Move Funktion ein. (nicht für LSTEP express)		
Delphi:	function LSX_GetCtrFastMoveOff(LSID: Integer; var bActive: LongBool): Integer;		
C++:	int GetCtrFastMove(BOOL *pbActive);		
LabView:			
Parameter:	bActive	Eingestellter Wert True = Fast Move Funktion aktiv True = Fast Move Funktion deaktiviert	
Beispiel:	LSX.GetCtrFastMoveOff(&bActive);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?ctrfm	-

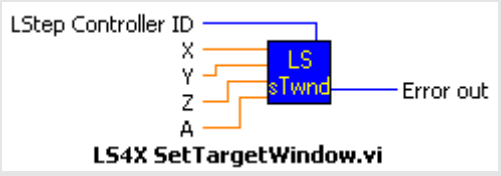
LSX_SetCtrFastMoveOff			
Beschreibung:	Funktion zum Deaktivieren der Fast Move Funktion. (nicht für LSTEP express)		
Delphi:	function LSX_SetCtrFastMoveOff(LSID: Integer): Integer;		
C++:	int SetCtrFastMoveOff();		
LabView:			
Parameter:	-		
Beispiel:	LSX.SetCtrFastMoveOff();		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!ctrfm	-

LSX_SetCtrFastMoveOn			
Beschreibung:	Fast Move Funktion aktivieren, d. h. bei einer Reglerdifferenz, die größer als der Fangbereich ist, wird ein neuer Vektor gestartet. (nicht für LSTEP express)		
Delphi:	function LSX_SetCtrFastMoveOn(LSID: Integer): Integer;		
C++:	int SetCtrFastMoveOn();		
LabView:			
Parameter:	-		
Beispiel:	LSX.SetCtrFastMoveOn();		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!ctrfm	-

LSX_GetCtrFastMoveCounter			
Beschreibung:	Diese Funktion liefert den Wert des Fast Move Counters. Bei einer Reglerdifferenz, die größer als der Fangbereich ist, wird ein neuer Vektor gestartet und der dazu gehörige Counter um Eins erhöht. (nicht für LSTEP express)		
Delphi:	function LSX_GetCtrFastMoveCounter(LSID: Integer; var XC, YC, ZC, AC: Integer): Integer;		
C++:	int GetCtrFastMoveCounter(int *plXC, int *plYC, int *plZC, int *plAC);		
LabView:			
Parameter:	XC, YC, ZC, AC	Anzahl der ausgeführter Fast Move Funktionen	
Beispiel:	LSX.SetCtrFastMoveCounter(&XC, &YC, &ZC, &AC);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	?ctrfmc	-

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. Die Function setzt Fast Move Counters aller Achsen auf Null. (nicht für LSTEP express)		
Delphi:	function LSX_ClearCtrFastMoveCounter(LSID: Integer): Integer;		
C++:	int ClearCtrFastMoveCounter();		
LabView:			
Parameter:			
Beispiel:	LSX.ClearCtrFastMoveCounter;		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	1	!ctrfmc	-

LSX_GetTargetWindow			
Beschreibung:	Liest die Zielfenster aller Achsen aus.		
Delphi:	function LSX_GetTargetWindow(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetTargetWindow(double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	X, Y, Z, A	Zielfenster abhängig von der eingestellten Dimension der Achse	
Beispiel:	LSX.GetTargetWindow(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?twi	-

LSX_SetTargetWindow			
Beschreibung:	Funktion zum Setzen des Zielfensters.		
Delphi:	function LSX_SetTargetWindow(LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetTargetWindow(double dX,double dY,double dZ,double dA);		
LabView:			
Parameter:	X, Y, Z, A	Einzustellendes Zielfenster in der Dimension der Achse	
Beispiel:	LSX.SetTargetWindow(1.0, 0.002, 1.0, 1.0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!twi	ValidPar / ValidConfig

LSX_GetDeviationRange		
Beschreibung:	Befehl zum Abfragen des maximal zulässigen Schleppfehlers (Lagereglerabweichung)	
Delphi:	function LSX_GetDeviationRange(LSID: Integer;var X,Y,Z,A: Double): Integer;	
C++:	int GetDeviationRange (double *pdX, double *pdY, double *pdZ, double *pdA);	
LabView:	Nicht unterstützt	
Parameter:	X, Y, Z, A	Eingestellter zulässiger Schleppfehler

Beispiel:	LSX.GetDeviationRange(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?deviationrange	-

LSX_SetDeviationRange			
Beschreibung:	Befehl zum Einstellen des maximal zulässigen Schleppfehlers (Lagereglerabweichung)		
Delphi:	function LSX_SetDeviationRange(LSID: Integer;X,Y,Z,A : Double): Integer;		
C++:	int SetDeviationRange (double dX, double dY, double dZ, double dA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellter zulässiger Schleppfehler	
Beispiel:	LSX.SetDeviationRange(1.0, 0.002, 1.0, 1.0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!deviationrange	ValidPar / ValidConfig

LSX_GetDeviationTime			
Beschreibung:	Befehl zum Abfragen der eingestellten Zeit, nach welcher bei Überschreitung des maximal zulässigen Schleppfehlers die Schleppfehlerüberwachung angesprochen wird.		
Delphi:	function LSX_GetDeviationTime(LSID: Integer;var X,Y,Z,A: Integer): Integer;		
C++:	int GetDeviationTime (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte Zeit zur Schleppfehlerüberwachung	
Beispiel:	LSX.GetDeviationTime(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?deviationtime	-

LSX_SetDeviationTime			
Beschreibung:	Befehl zum Einstellen des maximal zulässigen Schleppfehlers (Lagereglerabweichung)		
Delphi:	function LSX_SetDeviationTime(LSID: Integer; X,Y,Z,A : Integer): Integer;		
C++:	int SetDeviationTime(int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		

Parameter:	X, Y, Z, A	Eingestellte Zeit zur Schleppfehlerüberwachung	
Beispiel:	LSX.SetDeviationTime(1.0, 0.002, 1.0, 1.0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!deviationtime	ValidPar / ValidConfig

LSX_GetDeviationCheck			
Beschreibung:	Befehl zum Abfragen, ob die Schleppfehlerüberwachung eingeschaltet ist.		
Delphi:	function LSX_GetDeviationCheck (LSID: Integer;var X,Y,Z,A: LongBool):Integer;		
C++:	int GetDeviationCheck(BOOL *pbX, BOOL *pbY, BOOL *pbZ, BOOL *pbA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Zustand der Schleppfehlerüberwachung	
Beispiel:	LSX.GetDeviationCheck(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?deviationcheck	-

LSX_SetDeviationCheck			
Beschreibung:	Befehl zum Ein- und Ausschalten der Schleppfehlerüberwachung.		
Delphi:	function LSX_SetDeviationCheck (LSID: Integer;X,Y,Z,A: LongBool):Integer;		
C++:	int SetDeviationCheck (BOOL bX, BOOL bY, BOOL bZ, BOOL bA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Zustand der Schleppfehlerüberwachung	
Beispiel:	LSX.SetDeviationCheck(True,True,True,True);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!deviationcheck	ValidPar / ValidConfig

LSX_GetDeviationValue	
Beschreibung:	Fragt den Schleppfehler der Achsen ab.
Delphi:	function LSX_GetDeviationValue (LSID: Integer; var X,Y,Z,A: Double):Integer;
C++:	int GetDeviationValue(Double *pdX, Double *pdY, Double *pdZ, Double *pdA);
LabView:	Nicht unterstützt

Parameter:	X, Y, Z, A	Schleppfehlerwert	
Beispiel:	LSX.GetDeviationValue(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?deviationvalue	-

LSX_GetPoscon			
Beschreibung:	Fragt den Zustand der Lageregler in Abhänigkeit der achsenspezifischen Einstellungen ab		
Delphi:	function LSX_GetPoscon(LSID: Integer;var Enable: Boolean): Integer;		
C++:	int GetPoscon (BOOL *pbEnable);		
LabView:	Nicht unterstützt		
Parameter:	Enable	Zustand der Lagereglereinstellung	
Beispiel:	LSX.GetPoscon(&Enable);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?poscon	-

LSX_SetPoscon			
Beschreibung:	Befehl zum Ein- und Ausschalten der Lageregler in Abhänigkeit der achsenspezifischen Einstellungen		
Delphi:	function LSX_SetPoscon(LSID: Integer; Enable: Boolean): Integer;		
C++:	int SetPoscon (BOOL bEnable);		
LabView:	Nicht unterstützt		
Parameter:	Enable	Zustand der Lageregler	
Beispiel:	LSX.SetPoscon(True);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!poscon	ValidPar / ValidConfig

LSX_GetPosConKP			
Beschreibung:	Befehl zum Auslesen des Lageregler-Proportionalanteils in % für Servo und Schrittmotor		
Delphi:	function LSX_GetPosConKP (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetPosConKP (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellter Lageregler-Propotionalanteil, Bereich: 0 bis 5000%	
Beispiel:	LSX.GetPosConKP (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?posconkp	-

LSX_SetPosConKP			
Beschreibung:	Befehl zum Einstellen des Lageregler-Proportionalanteils in % für Servo und Schrittmotor		
Delphi:	function LSX_SetPosConKP (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetPosConKP (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellender Lageregler-Propotionalanteil, Bereich: 0 bis 5000%	
Beispiel:	LSX.SetPosConKP (40, 20, 80, 80);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!posconkp	ValidPar / ValidConfig

LSX_GetPosConAdaptiveKP			
Beschreibung:	Befehl zum Auslesen des adaptiven Lageregler-Proportionalanteils in % nur für Schrittmotor		
Delphi:	function LSX_GetPosConAdaptiveKP (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetPosConAdaptiveKP (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellter Lageregler-Propotionalanteil, Bereich: 0 bis 5000%	
Beispiel:	LSX.GetPosConAdaptiveKP (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?posconadaptivekp	-

LSX_SetPosConAdaptiveKP			
Beschreibung:	Befehl zum Einstellen des adaptiven Lageregler-Proportionalanteils in % nur für Schrittmotor		
Delphi:	function LSX_SetPosConAdaptiveKP (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetPosConAdaptiveKP (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellender Lageregler-Propotionalanteil, Bereich: 0 bis 5000%	
Beispiel:	LSX.SetPosConAdaptiveKP (40, 20, 80, 80);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!posconkp	ValidPar / ValidConfig

LSX_GetPosConKI			
Beschreibung:	Befehl zum Auslesen des Lageregler-Integralanteils in % für Servo und Schrittmotor		
Delphi:	function LSX_GetPosConKI (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetPosConKI (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellter Lageregler-Integralanteil, Bereich: 0 bis 5000%	
Beispiel:	LSX.GetPosConKI (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?posconki	-

LSX_SetPosConKI			
Beschreibung:	Befehl zum Einstellen des Lageregler-Integralanteils in % für Servo und Schrittmotor		
Delphi:	function LSX_SetPosConKI (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetPosConKI (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellender Lageregler-Integralanteil, Bereich: 0 bis 5000%	
Beispiel:	LSX.SetPosConKI (75, 60, 90, 90);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!posconki	ValidPar / ValidConfig

LSX_GetPosConAdaptiveKI			
Beschreibung:	Befehl zum Auslesen des adaptiven Lageregler-Integralanteils in % nur für Schrittmotor		
Delphi:	function LSX_GetPosConAdaptiveKI (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetPosConAdaptiveKI (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellter Lageregler-Integralanteil, Bereich: 0 bis 5000%	
Beispiel:	LSX.GetPosConAdaptiveKI (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?posconadaptiveki	-

LSX_SetPosConAdaptiveKI			
Beschreibung:	Befehl zum Einstellen des adaptiven Lageregler-Integralanteils in % nur für Schrittmotor		
Delphi:	function LSX_SetPosConAdaptiveKI (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetPosConAdaptiveKI (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellender Lageregler-Integralanteil, Bereich: 0 bis 5000%	
Beispiel:	LSX.SetPosConAdaptiveKI (40, 20, 80, 80);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!posconki	ValidPar / ValidConfig

LSX_GetPosConOutPass			
Beschreibung:	Befehl zum Auslesen der eingestellten Lagereglerausgangsfilter-Zeitkonstante für Servo und Schrittmotor		
Delphi:	function LSX_GetPosConOutPass (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetPosConOutPass (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte Lagereglerausgangsfilter-Zeitkonstante, Bereich: 0 bis 100000 µs	
Beispiel:	LSX.GetPosConOutPass (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?posconoutpass	-

LSX_SetPosConOutPass			
Beschreibung:	Befehl zum Einstellen der eingestellten Lagereglerausgangsfilter-Zeitkonstante für Servo und Schrittmotor		
Delphi:	function LSX_SetPosConOutPass (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetPosConOutPass (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellender Lagereglerausgangsfilter-Zeitkonstante, Bereich: 0 bis 100000 µs	
Beispiel:	LSX.SetPosConOutPass (200, 500, 200, 500);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!posconoutpass	ValidPar / ValidConfig

LSX_GetPosConAdaptiveOutPass			
Beschreibung:	Befehl zum Auslesen der eingestellten Lagereglerausgangsfilter-Zeitkonstante nur für Schrittmotor		
Delphi:	function LSX_GetPosConAdaptiveOutPass (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetPosConAdaptiveOutPass (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte Lagereglerausgangsfilter-Zeitkonstante, Bereich: 0 bis 100000 µs	
Beispiel:	LSX.GetPosConAdaptiveOutPass (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?posconadaptiveoutpass	-

LSX_SetPosConAdaptiveOutPass			
Beschreibung:	Befehl zum Einstellen der eingestellten Lagereglerausgangsfilter-Zeitkonstante nur für Schrittmotor		
Delphi:	function LSX_SetPosConAdaptiveOutPass (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetPosConAdaptiveOutPass (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellender Lagereglerausgangsfilter-Zeitkonstante, Bereich: 0 bis 100000 µs	
Beispiel:	LSX.SetPosConAdaptiveOutPass (200, 500, 200, 500);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!posconadaptiveoutpass	ValidPar / ValidConfig

LSX_GetPosConNominalSpeed			
Beschreibung:	Befehl zum Auslesen der Nominal-Geschwindigkeit/Drehzahl. Einheit ist abhängig von gewählter Dimension. PosCon…- Regelparamter (z.B. PosConKP) beziehen sich auf PosConNominalSpeed.		
Delphi:	function LSX_GetPosConNominalSpeed (LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetPosConNominalSpeed (double *plX, double *plY, double *plZ, double *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte Nominal-Geschwindigkeit/Drehzahl	
Beispiel:	LSX.GetPosConNominalSpeed (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?posconnominalspeed	-

LSX_SetPosConNominalSpeed			
Beschreibung:	Befehl zum Einstellen der Nominal-Geschwindigkeit/Drehzahl. Einheit ist abhängig von gewählter Dimension. PosCon…- Regelparamter (z.B. PosConKP) beziehen sich auf PosConNominalSpeed.		
Delphi:	function LSX_SetPosConNominalSpeed (LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetPosConNominalSpeed (double lX, double lY, double lZ, double lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellende Nominal-Geschwindigkeit/Drehzahl	
Beispiel:	LSX.SetPosConNominalSpeed (2.5, 2.5, 2.5, 2.5);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!posconnominalspeed	ValidPar / ValidConfig

LSX_GetPosConAdaptiveSpeed			
Beschreibung:	Befehl zum Auslesen der Adaptiv-Geschwindigkeit/Drehzahl. Einheit ist abhängig von gewählter Dimension. PosConAdaptive…- Regelparamter (z.B. PosConAdaptiveKP) beziehen sich auf PosConAdaptiveSpeed.		
Delphi:	function LSX_GetPosConAdaptiveSpeed (LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetPosConAdaptiveSpeed (double *plX, double *plY, double *plZ, double *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte Adaptiv-Geschwindigkeit/Drehzahl	
Beispiel:	LSX.GetPosConAdaptiveSpeed (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?posconadaptivespeed	-

LSX_SetPosConAdaptiveSpeed			
Beschreibung:	Befehl zum Einstellen der Adaptiv-Geschwindigkeit/Drehzahl. Einheit ist abhängig von gewählter Dimension. PosConAdaptive…- Regelparamter (z.B. PosConAdaptiveKP) beziehen sich auf PosConAdaptiveSpeed.		
Delphi:	function LSX_SetPosConAdaptiveSpeed (LSID: Integer; X, Y, Z, A: Double): Integer;		
C++:	int SetPosConAdaptiveSpeed (double lX, double lY, double lZ, double lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellende Adaptiv-Geschwindigkeit/Drehzahl	
Beispiel:	LSX.SetPosConAdaptiveSpeed (2.5, 2.5, 2.5, 2.5);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!posconadaptivespeed	ValidPar / ValidConfig

LSX_GetPosConEnable			
Beschreibung:	Befehl zum Auslesen der achsspezifischen Auswahl der Lageregler-Betriebsart		
Delphi:	function LSX_GetPosConEnable (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetPosConEnable (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte Lageregler-Betriebsart 0: Lageregler disabled 1: Lageregler enabled (Servo und Schrittmotor) 2: adaptiver Lageregler enabled (nur Schrittmotor)	
Beispiel:	LSX.GetPosConEnable (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?posconenable	-

LSX_SetPosConEnable			
Beschreibung:	Befehl zum Einstellen der achsspezifischen Auswahl der Lageregler-Betriebsart		
Delphi:	function LSX_SetPosConEnable (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetPosConEnable (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellende Lageregler-Betriebsart 0: Lageregler disabled 1: Lageregler enabled (Servo und Schrittmotor) 2: adaptiver Lageregler enabled (nur Schrittmotor)	
Beispiel:	LSX.SetPosConEnable (0, 2, 1, 1);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!posconenable	ValidPar / ValidConfig

LSX_GetSpeedConKP			
Beschreibung:	Befehl zum Auslesen des Drehzahlregler-Proportionalanteils in %		
Delphi:	function LSX_GetSpeedConKP (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetSpeedConKP (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellter Lageregler-Propotionalanteil, Bereich: 0 bis 5000%	
Beispiel:	LSX.GetSpeedConKP (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?speedconkp	-

LSX_SetSpeedConKP			
Beschreibung:	Befehl zum Einstellen des Drehzahlregler-Proportionalanteils in %		
Delphi:	function LSX_SetSpeedConKP (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetSpeedConKP (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellender Lageregler-Propotionalanteil, Bereich: 0 bis 5000%	
Beispiel:	LSX.SetSpeedConKP (40, 20, 80, 80);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!speedconkp	ValidPar / ValidConfig

LSX_GetSpeedConKI			
Beschreibung:	Befehl zum Auslesen des Drehzahlregler-Integralanteils in %		
Delphi:	function LSX_GetSpeedConKI (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetSpeedConKI (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellter Lageregler-Integralanteil, Bereich: 0 bis 5000%	
Beispiel:	LSX.GetSpeedConKI (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?speedconki	-

LSX_SetSpeedConKI			
Beschreibung:	Befehl zum Einstellen des Drehzahlregler-Integralanteils in %		
Delphi:	function LSX_SetSpeedConKI (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetSpeedConKI (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellender Lageregler-Integralanteil, Bereich: 0 bis 5000%	
Beispiel:	LSX.SetSpeedConKI (75, 60, 90, 90);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!speedconki	ValidPar / ValidConfig

LSX_GetSpeedConKD			
Beschreibung:	Befehl zum Auslesen des Drehzahlregler-Differenzialanteils in %		
Delphi:	function LSX_GetSpeedConKD (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetSpeedConKD (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellter Lageregler-Differenzialanteil, Bereich: 0 bis 5000%	
Beispiel:	LSX.GetSpeedConKD (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?speedconkd	-

LSX_SetSpeedConKD			
Beschreibung:	Befehl zum Einstellen des Drehzahlregler-Differenzialanteils in %		
Delphi:	function LSX_SetSpeedConKD (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetSpeedConKD (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellender Lageregler- Differenzialanteil, Bereich: 0 bis 5000%	
Beispiel:	LSX.SetSpeedConKD (75, 60, 90, 90);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!speedconkd	ValidPar / ValidConfig

LSX_GetSpeedConOutPass			
Beschreibung:	Befehl zum Auslesen der Drehzahlreglerausgang-Filterzeitkonstante		
Delphi:	function LSX_GetSpeedConOutPass (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetSpeedConOutPass (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte Lagereglerausgangsfilter-Zeitkonstante, Bereich: 0 bis 100000 µs	
Beispiel:	LSX.GetSpeedConOutPass (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?speedconoutpass	-

LSX_SetSpeedConOutPass			
Beschreibung:	Befehl zum Einstellen der Drehzahlreglerausgang-Filterzeitkonstante		
Delphi:	function LSX_SetSpeedConOutPass (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetSpeedConOutPass (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellender Lagereglerausgangsfilter-Zeitkonstante, Bereich: 0 bis 100000 µs	
Beispiel:	LSX.SetSpeedConOutPass (200, 500, 200, 500);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!speedconoutpass	ValidPar / ValidConfig

LSX_GetActSpeedFilterConst			
Beschreibung:	Befehl zum Auslesen der Drehzahlistwert-Filterzeitkonstante		
Delphi:	function LSX_GetActSpeedFilterConst (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetActSpeedFilterConst (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte Lagereglerausgangsfilter-Zeitkonstante, Bereich: 0 bis 100000 µs	
Beispiel:	LSX.GetActSpeedFilterConst (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?actspeedfilterconst	-

LSX_SetActSpeedFilterConst			
Beschreibung:	Befehl zum Einstellen der Drehzahlistwert-Filterzeitkonstante		
Delphi:	function LSX_SetActSpeedFilterConst (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetActSpeedFilterConst (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellender Lagereglerausgangsfilter-Zeitkonstante, Bereich: 0 bis 100000 µs	
Beispiel:	LSX.SetActSpeedFilterConst (200, 500, 200, 500);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!actspeedfilterconst	ValidPar / ValidConfig

LSX_GetSpeedFeedForward			
Beschreibung:	Befehl zum Auslesen der Drehzahl- bzw. Geschwindigkeitsvorsteuerung in %		
Delphi:	function LSX_GetSpeedFeedForward (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetSpeedFeedForward (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte Drehzahl- bzw. Geschwindigkeitsvorsteuerung, Bereich: 0 bis 400%	
Beispiel:	LSX.GetSpeedFeedForward (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?speedfeedforward	-

LSX_SetSpeedFeedForward			
Beschreibung:	Befehl zum Einstellen der Drehzahl- bzw. Geschwindigkeitsvorsteuerung in %		
Delphi:	function LSX_SetSpeedFeedForward (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetSpeedFeedForward (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte Drehzahl- bzw. Geschwindigkeitsvorsteuerung, Bereich: 0 bis 400%	
Beispiel:	LSX.SetSpeedFeedForward (75, 60, 90, 90);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!speedfeedforward	ValidPar / ValidConfig

LSX_GetActAccelFilterConst			
Beschreibung:	Befehl zum Auslesen der Drehzahlistwert-Filterzeitkonstante		
Delphi:	function LSX_GetActAccelFilterConst (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetActAccelFilterConst (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte Beschleunigungsistwert-Filterzeitkonstante, Bereich: 0 bis 100 ms	
Beispiel:	LSX.GetActAccelFilterConst (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?actaccelfilterconst	-

LSX_SetActAccelFilterConst			
Beschreibung:	Befehl zum Einstellen der Drehzahlistwert-Filterzeitkonstante		
Delphi:	function LSX_SetActAccelFilterConst (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetActAccelFilterConst (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellende Beschleunigungsistwert-Filterzeitkonstante, Bereich: 0 bis 100 ms	
Beispiel:	LSX.SetActAccelFilterConst (20, 5, 20, 5);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!actaccelfilterconst	ValidPar / ValidConfig

LSX_GetAccelFeedForward			
Beschreibung:	Befehl zum Auslesen der Beschleunigungsvorsteuerung in %		
Delphi:	function LSX_GetAccelFeedForward (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetAccelFeedForward (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte Beschleunigungsvorsteuerung, Bereich: 0 bis 400%	
Beispiel:	LSX.GetAccelFeedForward (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?accelfeedforward	-

LSX_SetAccelFeedForward			
Beschreibung:	Befehl zum Einstellen der Beschleunigungsvorsteuerung in %		
Delphi:	function LSX_SetAccelFeedForward (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetAccelFeedForward (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellende Beschleunigungsvorsteuerung, Bereich: 0 bis 400%	
Beispiel:	LSX.SetAccelFeedForward (50, 80, 70, 70);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!accelfeedforward	ValidPar / ValidConfig

LSX_GetAccelFeedForwardOutPass			
Beschreibung:	Befehl zum Auslesen der Beschleunigungsvorsteuerung-Filterzeitkonstante		
Delphi:	function LSX_GetAccelFeedForwardOutPass (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetAccelFeedForwardOutPass (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Eingestellte Beschleunigungsvorsteuerung-Filterzeitkonstante, Bereich: 0 bis 100000 µs	
Beispiel:	LSX.GetAccelFeedForwardOutPass (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?accelfeedforwardoutpass	-

LSX_SetAccelFeedForwardOutPass			
Beschreibung:	Befehl zum Einstellen der Beschleunigungsvorsteuerung-Filterzeitkonstante		
Delphi:	function LSX_SetAccelFeedForwardOutPass (LSID: Integer; X, Y, Z, A: Integer): Integer;		
C++:	int SetAccelFeedForwardOutPass (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Einzustellender Beschleunigungsvorsteuerung-Filterzeitkonstante, Bereich: 0 bis 100000 µs	
Beispiel:	LSX.SetAccelFeedForwardOutPass (200, 500, 200, 500);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!accelfeedforwardoutpass	ValidPar / ValidConfig

LSX_GetSpeedConTN			
Beschreibung:	Befehl zum Auslesen der wirksamen Nachstellzeit TN des Drehzahlreglers in [ms]		
Delphi:	function LSX_GetSpeedConTN (LSID: Integer; var X, Y, Z, A: Integer): Integer;		
C++:	int GetSpeedConTN (int *plX, int *plY, int *plZ, int *plA);		
LabView:	Nicht unterstützt		
Parameter:	X, Y, Z, A	Wirksame Nachstellzeit	
Beispiel:	LSX.GetSpeedConTN (&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?speedcontn	-

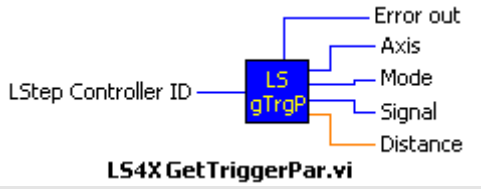
4.2.18 Trigger-Ausgang

LSX_GetTrigger			
Beschreibung:	Diese Funktion liefert den aktuellen Zustand der Trigger-Funktion.		
Delphi:	function LSX_GetTrigger(LSID: Integer; var ATrigger: Integer): Integer; function LSX_GetTrigger(LSID: Integer; var ATrigger: LongBool): Integer; (veraltet)		
C++:	int GetTrigger(int *plATrigger);		
LabView:	Nicht unterstützt		
Parameter:	ATrig-ger	Zustand der Trigger-Funktion 2 = Trigger „Ein“ (Triggerpunktberechnung mit Positionswerten und Geschwindigkeit) 1 = Trigger „Ein“ (Triggerpunktberechnung mit Positionswerten) 0 = Trigger „Aus“	
Beispiel:	LSX.GetTrigger(&ATrigger);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	?trig	-

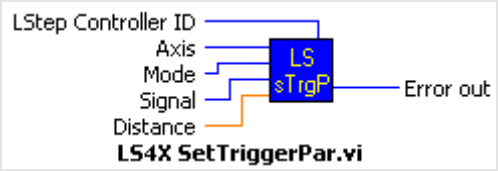
LSX_SetTrigger			
Beschreibung:	Funktion zum Ein- / Ausschalten der Trigger-Funktion.		
Delphi:	function LSX_SetTrigger(LSID: Integer; ATrigger: Integer): Integer; function LSX_SetTrigger(LSID: Integer; ATrigger: LongBool): Integer; (veraltet)		
C++:	int SetTrigger(int lATrigger);		
LabView:	Nicht unterstützt		
Parameter:	ATrig-ger	Zustand der Trigger-Funktion 2 = Trigger „Ein“ (Triggerpunktberechnung mit Positionswerten und Geschwindigkeit) 1 = Trigger „Ein“ (Triggerpunktberechnung mit Positionswerten) 0 = Trigger „Aus“	
Beispiel:	LSX.SetTrigger(1);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	!trig	-

LSX_GetTriggerCount			
Beschreibung:	Funktion zum Auslesen des Triggerzählerstandes.		
Delphi:	function LSX_GetTriggerCount(LSID: Integer; var Value: Cardinal): Integer; function LSX_GetTrigCount(LSID: Integer; var Value: Integer): Integer; (veraltet)		
C++:	int GetTriggerCount (unsigned long *plValue); int GetTrigCount (int *plValue);		
LabView:	Nicht unterstützt		
Parameter:	Value	Anzahl der ausgeführten Triggerevents	
Beispiel:	LSX.GetTriggerCount(&Value);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?trigcount	-

LSX_SetTriggerCount			
Beschreibung:	Funktion zum Setzen des Triggerzählerstands.		
Delphi:	function LSX_SetTriggerCount(LSID: Integer; Value: Cardinal): Integer; function LSX_SetTrigCount(LSID: Integer; Value: Integer): Integer; (veraltet)		
C++:	int SetTriggerCount (unsigned long lValue); int SetTrigCount (int lValue);		
LabView:	Nicht unterstützt		
Parameter:	Value	Gewollter Wert des Triggerzählers	
Beispiel:	LSX.SetTriggerCount(0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!trigcount	-

LSX_GetTriggerPar	
Beschreibung:	Liefert die eingestellten Parameter der Trigger-Funktion.
Delphi:	function LSX_GetTriggerPar(LSID: Integer; var Axis, Mode, Signal: Integer; var Distance: Double): Integer;
C++:	int GetTriggerPar(int *plAxis, int *plMode, int *plSignal, double *pdDistance);
LabView:	

Parameter:	Axis	Achszuweisung der Trigger-Funktion X = 1 Y = 2 …	
	Mode	Trigger-Modus (siehe Anleitung der Steuerung unter dem Befehl trigm)	
	Signal	Trigger-Signallänge in µs	
	Distance	Trigger-Distanz in der eingestellten Einheit der Achse	
Beispiel:	LSX.GetTriggerPar(&Axis, & Mode, & Signal, & Distance);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?triga / ?trigm ?trigs / ?trigd	-

LSX_SetTriggerPar			
Beschreibung:	Setzt die Trigger-Parameter für die Trigger-Funktion		
Delphi:	function LSX_SetTriggerPar(LSID: Integer; Axis, Mode, Signal: Integer; Distance: Double): Integer;		
C++:	int SetTriggerPar(int lAxis, int lMode, int lSignal, double dDistance);		
LabView:			
Parameter:	Axis	Achszuweisung der Trigger-Funktion X = 1 Y = 2 …	
	Mode	Trigger-Modus (siehe Anleitung der Steuerung unter dem Befehl trigm)	
	Signal	Trigger-Signallänge in µs	
	Distance	Trigger-Distanz in der eingestellten Einheit der Achse	
Beispiel:	LSX.SetTriggerPar(1, 3, 2, 5.0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!triga / !trigm !trigs / !trigd	SetTrigger

LSX_GetTriggerDim			
Beschreibung:	Fragt die ausgewählte Einheit für die Parameterübergabe zur Triggerdistanz und Triggerhysterese ab.		
Delphi:	function LSX_GetTriggerDim(LSID: Integer;var Dimension: Integer): Integer;		
C++:	int GetTriggerDim (int *plDimension);		
LabView:	Nicht unterstützt		
Parameter:	Dimension	0 → Parameter werden in der Anwendereinheit übergeben (siehe Befehl „Dim“) 1 → Parameter werden unabänig von der Anwendereinheit in Microsteps übergeben	
Beispiel:	LSX.GetTriggerDim (&Dimension);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	?trigdim	-

LSX_SetTriggerDim			
Beschreibung:	Stellt die Einheit für die Parameterübergabe zur Triggerdistanz und Triggerhysterese um.		
Delphi:	function LSX_SetTriggerDim(LSID: Integer; Dimension: Integer): Integer;		
C++:	int SetTriggerDim (int lDimension);		
LabView:	Nicht unterstützt		
Parameter:	Dimension	0 → Parameter werden in der Anwendereinheit übergeben (siehe Befehl „Dim“) 1 → Parameter werden unabänig von der Anwendereinheit in Microsteps übergeben	
Beispiel:	LSX.SetTriggerDim(0);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	!trigdim	-

LSX_GetTriggerSource			
Beschreibung:	Fragt die ausgewählte Triggerquelle ab. Es kann auf die Sollposition oder auf den Geberwert der gewählten Achse getriggert werden. Beim Triggern auf den Geberwert ist die Parametrierung einer geeigneten Triggerhysterese erforderlich.		
Delphi:	function LSX_GetTriggerSource (LSID: Integer; var Source: integer): Integer;		
C++:	int GetTriggerSource (int *plSource);		
LabView:	Nicht unterstützt		
Parameter:	Dimension	0 → Triggern auf die Sollposition 1 → Triggern auf die Geberwerte	
Beispiel:	LSX.GetTriggerSource (&Source);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?trigsource	-

LSX_SetTriggerSource			
Beschreibung:	Wählt die Triggerquelle aus. Es kann auf die Sollposition oder auf den Geberwert der gewählten Achse getriggert werden. Beim Triggern auf den Geberwert ist die Parametrierung einer geeigneten Triggerhysterese erforderlich.		
Delphi:	function LSX_SetTriggerSource (LSID: Integer; Source: Integer): Integer;		
C++:	int SetTriggerSource (int lSource);		
LabView:	Nicht unterstützt		
Parameter:	Dimension	0 → Triggern auf die Sollposition 1 → Triggern auf die Geberwerte	
Beispiel:	LSX.SetTriggerSource(0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!trigsource	SetTrigger

LSX_GetTriggerHysterese			
Beschreibung:	Fragt dieTriggerhysterese ab. Beim Triggern auf Geberwerte ist eine Hysterese um die Triggerposition erforderlich. Dadurch wird die ungewollte Ausgabe von Triggersignalen aufgrund von Störungen (z.B. Auslenkung der Achse durch externe Kräfte/ Drehmimente über die Triggerposition oder Rauschen auf dem Gebersignal,…) vermieden. Nach Ausgabe eines Triggerimpulses kann ein erneuter Impuls erst nach Zurücklegung der Triggerhysterese ausgegeben werden, die Triggerhysterese muss daher kleiner als die halbe Triggerdistanz sein.		
Delphi:	function LSX_GetTriggerHysterese (LSID: Integer; var Hysterese: Double): Integer;		
C++:	int GetTriggerHysterese (double *pdHysterese);		
LabView:	Nicht unterstützt		
Parameter:	Hysterese	0 bis Maximum mit bis zu 8 Nachkommastellen in der eingestellten Dimension	
Beispiel:	LSX.GetTriggerHysterese (&Hysterese);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?trigh	-

LSX_SetTriggerHysterese			
Beschreibung:	Wählt dieTriggerhysterese aus. Beim Triggern auf Geberwerte ist eine Hysterese um die Triggerposition erforderlich. Dadurch wird die ungewollte Ausgabe von Triggersignalen aufgrund von Störungen (z.B. Auslenkung der Achse durch externe Kräfte/ Drehmimente über die Triggerposition oder Rauschen auf dem Gebersignal,…) vermieden. Nach Ausgabe eines Triggerimpulses kann ein erneuter Impuls erst nach Zurücklegung der Triggerhysterese ausgegeben werden, die Triggerhysterese muss daher kleiner als die halbe Triggerdistanz sein.		
Delphi:	function LSX_SetTriggerHysterese (LSID: Integer; Hysterese: Double): Integer;		
C++:	int SetTriggerHysterese (double dHysterese);		
LabView:	Nicht unterstützt		
Parameter:	Hysterese	0 bis Maximum mit bis zu 8 Nachkommastellen in der eingestellten Dimension	
Beispiel:	LSX.SetTriggerHysterese(0.0022);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!trigh	SetTrigger

LSX_GetTriggerPLength			
Beschreibung:	Fragt die eingestellte Triggersignal-Periodenlänge ab.		
Delphi:	function LSX_GetTriggerPLength (LSID: Integer; var Length: Integer): Integer;		
C++:	int GetTriggerPLength (int *plLength);		
LabView:	Nicht unterstützt		
Parameter:	Length	Triggersignal-Periodenlänge	
Beispiel:	LSX.GetTriggerPLength (&Length);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?trigp	-

LSX_SetTriggerPLength			
Beschreibung:	Stellt die Triggersignal-Periodenlänge ein.		
Delphi:	function LSX_SetTriggerPLength (LSID: Integer; Length: Integer): Integer;		
C++:	int SetTriggerPLength (int lLength);		
LabView:	Nicht unterstützt		
Parameter:	Length	Triggersignal-Periodenlänge	
Beispiel:	LSX.SetTriggerPLength(0.0022);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!trigp	SetTrigger

LSX_GetTriggerPulsCount			
Beschreibung:	Fragt die auszugebende Pulsanzahl im Burst-Modus ab.		
Delphi:	function LSX_GetTriggerPulsCount (LSID: Integer;var Count : Integer): integer;		
C++:	int GetTriggerPulsCount (int *plCount);		
LabView:	Nicht unterstützt		
Parameter:	Count	Trigger-Puls Anzahl	
Beispiel:	LSX.GetTriggerPulsCount (&Count);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?trigc	-

LSX_SetTriggerPulsCount			
Beschreibung:	Stellt die auszugebende Pulsanzahl im Burst-Modus ein.		
Delphi:	function LSX_SetTriggerPulsCount (LSID: Integer; Count: Integer):Integer;		
C++:	int SetTriggerPulsCount (int lCount);		
LabView:	Nicht unterstützt		
Parameter:	Count	Trigger-Puls Anzahl	
Beispiel:	LSX.SetTriggerPulsCount(0.0022);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!trigc	SetTrigger

LSX_GetTrigger_Two			
Beschreibung:	Diese Funktion liefert den aktuellen Zustand der zweiten Trigger-Funktion.		
Delphi:	function LSX_GetTrigger_Two(LSID: Integer; var Status: Integer): Integer;		
C++:	int GetTrigger_Two (int *plStatus);		
LabView:	Nicht unterstützt		
Parameter:	Status	Zustand der Trigger-Funktion 1 = Trigger „Ein" 0 = Trigger „Aus"	
Beispiel:	LSX.GetTrigger_Two(&Value);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?trig_two	-

LSX_SetTrigger_Two			
Beschreibung:	Funktion zum Ein- / Ausschalten der zweiten Trigger-Funktion.		
Delphi:	function LSX_SetTrigger_Two(LSID: Integer; Status: Integer): Integer;		
C++:	int SetTrigger_Two (int lStatus);		
LabView:	Nicht unterstützt		
Parameter:	Status	Zustand der Trigger-Funktion 1 = Trigger „Ein" 0 = Trigger „Aus"	
Beispiel:	LSX.SetTrigger_Two(1);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!trig	-

LSX_GetTrigger_TwoCount			
Beschreibung:	Funktion zum Auslesen des zweiten Triggerzählerstandes.		
Delphi:	function LSX_GetTrigger_TwoCount (LSID: Integer; var Value: Cardinal): Integer;		
C++:	int GetTrigger_TwoCount (unsigned long *plValue);		
LabView:	Nicht unterstützt		
Parameter:	Value	Anzahl der ausgeführten Triggerevents	
Beispiel:	LSX.GetTrigger_TwoCount(&Value);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?trigcount_two	-

LSX_SetTrigger_TwoCount			
Beschreibung:	Funktion zum Setzen des zweiten Triggerzählerstands.		
Delphi:	function LSX_SetTriggerCount(LSID: Integer; Value: Integer): Integer;		
C++:	int SetTrigger_TwoCount (unsigned long lValue);		
LabView:	Nicht unterstützt		
Parameter:	Value	Gewollter Wert des zweiten Triggerzählers	
Beispiel:	LSX.SetTrigger_TwoCount(0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!trigcount_two	-

<table>
<tr><th colspan="5">LSX_GetTrigger_TwoPar</th></tr>
<tr><td>Beschreibung:</td><td colspan="4">Liefert die eingestellten Parameter der zweiten Trigger-Funktion.</td></tr>
<tr><td>Delphi:</td><td colspan="4">function LSX_GetTrigger_TwoPar(LSID: Integer; var Axis: Integer; var Mode: Integer; var Signal: Integer; var Distance: Double): Integer;</td></tr>
<tr><td>C++:</td><td colspan="4">int GetTrigger_TwoPar (int *plAxis, int *plMode, int *plSignal, double *pdDistance);</td></tr>
<tr><td>LabView:</td><td colspan="4">Nicht unterstützt</td></tr>
<tr><td rowspan="4">Parameter:</td><td>Axis</td><td colspan="3">Achszuweisung der Trigger-Funktion
X = 1
Y = 2
…</td></tr>
<tr><td>Mode</td><td colspan="3">Trigger-Modus (siehe Anleitung der Steuerung unter dem Befehl trigm)</td></tr>
<tr><td>Signal</td><td colspan="3">Trigger-Signallänge in µs</td></tr>
<tr><td>Distance</td><td colspan="3">Trigger-Distanz in der eingestellten Einheit der Achse</td></tr>
<tr><td>Beispiel:</td><td colspan="4">LSX.GetTrigger_TwoPar(&Axis, & Mode, & Signal, & Distance);</td></tr>
<tr><td rowspan="2">Sonstiges:</td><td colspan="2">Kompatibilität</td><td>“SendString” Befehl</td><td>Aktivierung (LSTEP express Serie)</td></tr>
<tr><td colspan="2">2</td><td>?triga_two / ?trigm_two / ?trigs_two / ?trigd_two</td><td>-</td></tr>
</table>

LSX_SetTrigger_TwoPar			
Beschreibung:	Setzt die Trigger-Parameter für die zweite Trigger-Funktion		
Delphi:	function LSX_SetTrigger_TwoPar(LSID: Integer; Axis: Integer; Mode: Integer; Signal: Integer; Distance: Double): Integer;		
C++:	int SetTrigger_TwoPar (int lAxis, int lMode, int lSignal, double dDistance);		
LabView:	Nicht unterstützt		
Parameter:	Axis	Achszuweisung der Trigger-Funktion X = 1 Y = 2 …	
	Mode	Trigger-Modus (siehe Anleitung der Steuerung unter dem Befehl trigm)	
	Signal	Trigger-Signallänge in µs	
	Distance	Trigger-Distanz in der eingestellten Einheit der Achse	
Beispiel:	LSX.SetTrigger_TwoPar(1, 3, 2, 5.0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!triga_two / !trigm_two / !trigs_two / !trigd_two	SetTrigger_Two

LSX_GetTrigger_TwoDim			
Beschreibung:	Fragt die ausgewählte Einheit für die Parameterübergabe zur Triggerdistanz und Triggerhysterese des zweiten Triggers ab.		
Delphi:	function LSX_GetTrigger_TwoDim (LSID: Integer;var Dimension:integer): Integer;		
C++:	int GetTrigger_TwoDim (int *plDimension);		
LabView:	Nicht unterstützt		
Parameter:	Dimension	0 → Parameter werden in der Anwendereinheit übergeben (siehe Befehl „Dim") 1 → Parameter werden unabänig von der Anwendereinheit in Microsteps übergeben	
Beispiel:	LSX.GetTrig_TwoDim (&Dimension);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?trigdim_two	-

LSX_SetTrigger_TwoDim			
Beschreibung:	Stellt die Einheit für die Parameterübergabe zur Triggerdistanz und Triggerhysterese für den zweiten Trigger um.		
Delphi:	function LSX_SetTrigger_TwoDim(LSID: Integer; Dimension: Integer): Integer;		
C++:	int SetTrigger_TwoDim (int lDimension);		
LabView:	Nicht unterstützt		
Parameter:	Dimension	0 → Parameter werden in der Anwendereinheit übergeben (siehe Befehl „Dim“) 1 → Parameter werden unabänig von der Anwendereinheit in Microsteps übergeben	
Beispiel:	LSX.SetTrigger_TwoDim(0);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	!trigdim_two	-

LSX_GetTrigger_TwoSource			
Beschreibung:	Fragt die ausgewählte Triggerquelle des zweiten Triggers ab. Es kann auf die Sollposition oder auf den Geberwert der gewählten Achse getriggert werden. Beim Triggern auf den Geberwert ist die Parametrierung einer geeigneten Triggerhysterese erforderlich.		
Delphi:	function LSX_GetTrigger_TwoSource (LSID: Integer; var Source: integer): Integer;		
C++:	int GetTrigger_TwoSource (int *plSource);		
LabView:	Nicht unterstützt		
Parameter:	Dimension	0 → Triggern auf die Sollposition 1 → Triggern auf die Geberwerte	
Beispiel:	LSX.GetTrigger_TwoSource (&Source);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	2	?trigsource_two	-

LSX_SetTrigger_TwoSource			
Beschreibung:	Wählt die Triggerquelle des zweiten Triggers aus. Es kann auf die Sollposition oder auf den Geberwert der gewählten Achse getriggert werden. Beim Triggern auf den Geberwert ist die Parametrierung einer geeigneten Triggerhysterese erforderlich.		
Delphi:	function LSX_SetTrigger_TwoSource (LSID: Integer; Source: Integer): Integer;		
C++:	int SetTrigger_TwoSource (int lSource);		
LabView:	Nicht unterstützt		
Parameter:	Dimension	0 → Triggern auf die Sollposition 1 → Triggern auf die Geberwerte	
Beispiel:	LSX.SetTrigger_TwoSource(0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!trigsource_two	SetTrigger_Two

LSX_GetTrigger_TwoHysterese			
Beschreibung:	Fragt dieTriggerhysterese des zweiten Triggers ab. Beim Triggern auf Geberwerte ist eine Hysterese um die Triggerposition erforderlich. Dadurch wird die ungewollte Ausgabe von Triggersignalen aufgrund von Störungen (z.b. Auslenkung der Achse durch externe Kräfte/ Drehmimente über die Triggerposition oder Rauschen auf dem Gebersignal,…) vermieden. Nach Ausgabe eines Triggerimpulses kann ein erneuter Impuls erst nach Zurücklegung der Triggerhysterese ausgegeben werden, die Triggerhysterese muss daher kleiner als die halbe Triggerdistanz sein.		
Delphi:	function LSX_GetTrigger_TwoHysterese (LSID: Integer; var Hysterese: Double): Integer;		
C++:	int GetTrigger_TwoHysterese (double *pdHysterese);		
LabView:	Nicht unterstützt		
Parameter:	Hysterese	0 bis Maximum mit bis zu 8 Nachkommastellen in der eingestellten Dimension	
Beispiel:	LSX.GetTrigger_TwoHysterese (&Hysterese);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?trigh_two	-

LSX_SetTrigger_TwoHysterese			
Beschreibung:	Wählt dieTriggerhysterese des zweiten Triggers aus. Beim Triggern auf Geberwerte ist eine Hysterese um die Triggerposition erforderlich. Dadurch wird die ungewollte Ausgabe von Triggersignalen aufgrund von Störungen (z.b. Auslenkung der Achse durch externe Kräfte/ Drehmimente über die Triggerposition oder Rauschen auf dem Gebersignal,…) vermieden. Nach Ausgabe eines Triggerimpulses kann ein erneuter Impuls erst nach Zurücklegung der Triggerhysterese ausgegeben werden, die Triggerhysterese muss daher kleiner als die halbe Triggerdistanz sein.		
Delphi:	function LSX_SetTrigger_TwoHysterese (LSID: Integer; Hysterese: Double): Integer;		
C++:	int SetTrigger_TwoHysterese (double dHysterese);		
LabView:	Nicht unterstützt		
Parameter:	Hysterese	0 bis Maximum mit bis zu 8 Nachkommastellen in der eingestellten Dimension	
Beispiel:	LSX.SetTrigger_TwoHysterese(0.0022);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!trigh_two	SetTrigger_Two

LSX_GetTrigger_TwoPLength			
Beschreibung:	Fragt die eingestellte Triggersignal-Periodenlänge des zweiten Triggers ab.		
Delphi:	function LSX_GetTrigger_TwoPLength (LSID: Integer; var Length: Integer): Integer;		
C++:	int GetTrigger_TwoPLength (int *plLength);		
LabView:	Nicht unterstützt		
Parameter:	Length	Triggersignal-Periodenlänge	
Beispiel:	LSX.GetTrigger_TwoPLength (&Length);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?trigp_two	-

LSX_SetTrigger_TwoPLength			
Beschreibung:	Stellt die Triggersignal-Periodenlänge des zweiten Triggers ein.		
Delphi:	function LSX_SetTrigger_TwoPLength (LSID: Integer; Length: Integer): Integer;		
C++:	int SetTrigger_TwoPLength (int lLength);		
LabView:	Nicht unterstützt		
Parameter:	Length	Triggersignal-Periodenlänge	
Beispiel:	LSX.SetTrigger_TwoPLength(0.0022);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!trigp_two	SetTrigger_Two

LSX_GetTrigger_TwoPulsCount			
Beschreibung:	Fragt die auszugebende Pulsanzahl des zweiten Triggers im Burst-Modus ab.		
Delphi:	function LSX_GetTrigger_TwoPulsCount (LSID: Integer;var Count : Integer): integer;		
C++:	int GetTrigger_TwoPulsCount (int *plCount);		
LabView:	Nicht unterstützt		
Parameter:	Count	Trigger-Puls Anzahl	
Beispiel:	LSX.GetTrigger_TwoPulsCount (&Count);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?trigc_two	-

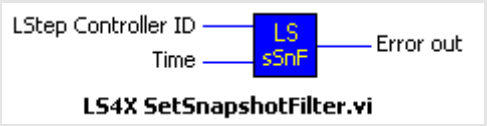
LSX_SetTrigger_TwoPulsCount			
Beschreibung:	Stellt die auszugebende Pulsanzahl des zweiten Triggers im Burst-Modus ein.		
Delphi:	function LSX_SetTrigger_TwoPulsCount (LSID: Integer; Count: Integer):Integer;		
C++:	int SetTrigger_TwoPulsCount (int lCount);		
LabView:	Nicht unterstützt		
Parameter:	Count	Trigger-Puls Anzahl	
Beispiel:	LSX.SetTrigger_TwoPulsCount(0.0022);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!trigc_two	SetTrigger_Two

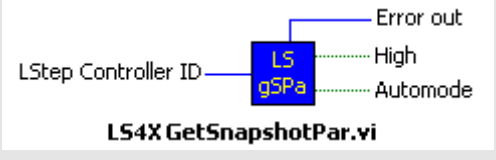
4.2.19 Snapshot-Eingang

LSX_GetSnapshot			
Beschreibung:	Liefert den aktuellen Zustand der Snapshot-Funktion.		
Delphi:	function LSX_GetSnapshot(LSID: Integer; var ASnapshot: LongBool): Integer;		
C++:	int GetSnapshot(BOOL *pbASnapshot);		
LabView:			
Parameter:	ASnapshot	Zustand der Snapshot-Funktion True = Snapshot-Funktion „Ein“ False = Snapshot-Funktion „Aus“	
Beispiel:	LSX.GetSnapshot(&ASnapshot);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	3	?sns	-

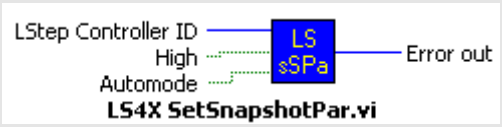
LSX_SetSnapshot			
Beschreibung:	Funktion zum Ein-/Ausschalten der Snapshot-Funktion.		
Delphi:	function LSX_SetSnapshot(LSID: Integer; ASnapshot: LongBool): Integer;		
C++:	int SetSnapshot (BOOL bASnapshot);		
LabView:			
Parameter:	ASnapshot	Zustand der Snapshot-Funktion True = Snapshot-Funktion „Ein“ False = Snapshot-Funktion „Aus“	
Beispiel:	LSX.SetSnapshot(true);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	3	!sns	-

LSX_GetSnapshotFilter			
Beschreibung:	Eingestellten Eingangsfilter der Snapshot-Funktion auslesen.		
Delphi:	function LSX_GetSnapshotFilter(LSID: Integer; var lTime: Integer): Integer;		
C++:	int GetSnapshotFilter(int *plTime);		
LabView:			
Parameter:	lTime	Filterzeit in ms	
Beispiel:	LSX.GetSnapshotFilter(&lTime);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?snsf	-

LSX_SetSnapshotFilter			
Beschreibung:	Funktion um den Eingangsfilter der Snapshot-Funktion für prellende Schalter zu setzen.		
Delphi:	function LSX_SetSnapshotFilter(LSID: Integer; lTime: Integer): Integer;		
C++:	int SetSnapshotFilter(int lTime);		
LabView:			
Parameter:	lTime	Filterzeit in ms	
Beispiel:	LSX.SetSnapshotFilter(0); // Kein Eingangsfilter		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!snsf	SetSnapshot

LSX_GetSnapshotPar	
Beschreibung:	Funktion zum Auslesen der Snapshot-Parameter.
Delphi:	function LSX_GetSnapshotPar(LSID: Integer; var High, AutoMode: LongBool): Integer;
C++:	int GetSnapshotPar(BOOL *pbHigh, BOOL *pbAutoMode);
LabView:	

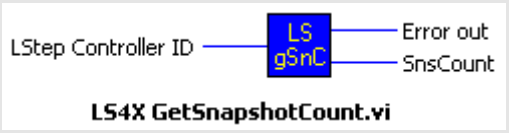
Parameter:	High	Aktivierung des Snapshot-Eingangs True = Snapshot ist high-aktiv False = Snapshot ist low-aktiv	
	AutoMode	Aktivierung des Automatikbetrieb True = Snapshot-Funktion im „Automatikbetrieb“. Die Position wird nach dem ersten Impuls automatisch angefahren. False = Snapshot-Funktion ist nicht im Automatikbetrieb	
Beispiel:	LSX.GetSnapshotPar(&High, & AutoMode);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	3	?snsl / ?snsm	-

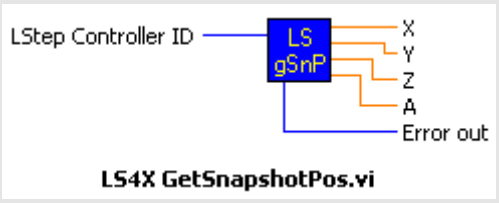
LSX_SetSnapshotPar			
Beschreibung:	Funktion zum Setzen der Snapshot-Parameter.		
Delphi:	function LSX_SetSnapshotPar(LSID: Integer; High, AutoMode: LongBool): Integer;		
C++:	int SetSnapshotPar(BOOL bHigh, BOOL bAutoMode);		
LabView:			
Parameter:	High	Aktivierung des Snapshot-Eingangs True = Snapshot ist high-aktiv False = Snapshot ist low-aktiv	
	AutoMode	Aktivierung des Automatikbetrieb True = Snapshot-Funktion im „Automatikbetrieb“. Die Position wird nach dem ersten Impuls automatisch angefahren. False = Snapshot-Funktion ist nicht im Automatikbetrieb	
Beispiel:	LSX.SetSnapshotPar(true, false);		
Sonstiges:	Kompatibilität	“SendString” Befehl	Aktivierung (LSTEP express Serie)
	3	!snsl / !snsm	SetSnapshot

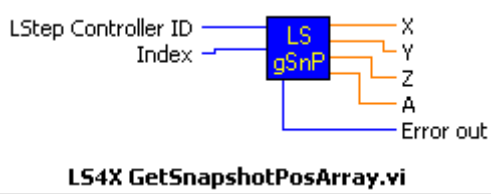
LSX_GetSnapshotSource	
Beschreibung:	Funktion zum auslesen der aktuell parametrierten Lagewertequelle
Delphi:	function LSX_GetSnapShotSource (LSID: Integer;var X,Y,Z,A: Integer): Integer;
C++:	int GetSnapShotSource (int *plX, int *plY, int *plZ, int *plA);
LabView:	Nicht unterstützt

Parameter:	X,Y,Z,A	0 → Sollposition speichern 1 → Geberwert speichern	
Beispiel:	LSX.GetSnapshotSource(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	?snssource	-

LSX_SetSnapshotSource			
Beschreibung:	Funktion zum Auswählen der Lagewertequelle Es können die Sollpositionen oder die Geberwerte der gewählten Achse gespeichert werden.		
Delphi:	function LSX_SetSnapShotSource (LSID: Integer; X,Y,Z,A: Integer): Integer; function LSX_SetSnapShotSource (LSID: Integer;Axis: Integer;Source:Integer): Integer;		
C++:	int SetSnapShotSource (int lX, int lY, int lZ, int lA);		
LabView:	Nicht unterstützt		
Parameter:	X,Y,Z,A	0 → Sollposition speichern 1 → Geberwert speichern	
	Axis,Source	Axis: 1 → X, 2 → Y, 3 → Z, 4 → A Source: 0 → Sollposition speichern 1 → Geberwert speichern	
Beispiel:	LSX.SetSnapshotSource(1,1); LSX.SetSnapshotSource(1,1,0,0);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	2	!snssource	SetSnapshot

LSX_GetSnapshotCount			
Beschreibung:	Funktion zum Auslesen des Snapshot-Zählerstands.		
Delphi:	function LSX_GetSnapshotCount(LSID: Integer; var SnsCount: Integer): Integer;		
C++:	int GetSnapshotCount(int *plSnsCount);		
LabView:			
Parameter:	SnsCount	Snapshot-Zählerstand	
Beispiel:	LSX.GetsnapshotCount(&SnsCount);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	?snsc	-

LSX_GetSnapshotPos			
Beschreibung:	Funktion zum Auslesen der Snapshot-Position.		
Delphi:	function LSX_GetSnapshotPos(LSID: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetSnapshotPos(double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	X, Y, Z, A	Snapshot-Position in der eingestellten Einheit	
Beispiel:	double X, Y, Z, A; LSX.GetSnapshotPos(&X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!snsp	-

LSX_GetSnapshotPosArray			
Beschreibung:	Funktion zum Auslesen des Arrays der Snapshot-Positionen.		
Delphi:	function LSX_GetSnapshotPosArray(LSID: Integer; Index: Integer; var X, Y, Z, A: Double): Integer;		
C++:	int GetSnapshotPosArray(int lIndex, double *pdX, double *pdY, double *pdZ, double *pdA);		
LabView:			
Parameter:	Index	Index im Snapshot-Positions-Array	
	X, Y, Z, A	Positionswerte in der eingestellten Dimension der Achse	
Beispiel:	double X, Y, Z, A; LSX.GetSnapshotPos(2, &X, &Y, &Z, &A);		
Sonstiges:	Kompatibilität	"SendString" Befehl	Aktivierung (LSTEP express Serie)
	3	!snsa	-

5 CallBack-Funktionen der LSTEP-API

Die LSTEP-API stellt für asynchrone Ereignisse CallBack-Funktionen zur Verfügung. Diese werden von einer LSTEP-Steuerung ausgelöst. Mit den CallBack-Funktionen ist es möglich ein asynchrones Ereignis aufzubereiten und eine entsprechende Aktion zu signalisieren.

Da die CallBack-Funktion während der Verarbeitung einer Eingangsnachricht aufgerufen wird hat sie Einfluss auf die Verarbeitungsgeschwindigkeit von Nachrichten. Bei der Aufbereitung der asynchronen Ereignisse sollte daher auf die Laufzeit geachtet werden. Sind größere Verarbeitungen beabsichtigt sollten diese nicht in der CallBack-Funktion geschehen, sondern über ein Signal ausgelöst werden.

Die LSTEP-API bietet eine Steuerungsbezogene und eine globale CallBack-Funktion an. Wird die steuerungsbezogene Funktion verwendet wird die globale CallBack-Funktion für diese Steuerung deaktiviert.

Nachfolgend werden die Meldungen des CallBacks sowie dessen Einsatzmöglichkeiten beschrieben.

5.1 Standard-CallBack-Funktion (OsziCallBackFct)

Die Standard-CallBack-Funktion wird in den Programmiersprachen Delphi, C++ bzw. C# wie folgt deklariert:

Deklaration Standard-CallBack-Funktion (OsziCallBackFct)		
Delphi	procedure OsziCallBackFct(Data: PAnsiChar; MaxLen: Integer; ChannelID: Integer); stdcall;	
C++	void CALLBACK OsziCallBackFct(char *pcData, int lMaxLen, int lChannelID)	
C#	void OsziCallBackFct (IntPtr Data, int MaxLen, int ChannelID)	
Parameter	Data	Zeiger auf ASCII-Zeichenkette, nullterminiert.
	MaxLen	Maximale Länge der ASCII-Zeichenkette in Data.
	ChannelID	Kanal, auf den sich die ASCII-Zeichenkette bezieht. Mögliche Kanäle siehe „5.3 Kanal Nummern“.
Rückgabewert	-	

Diese CallBack-Funktion wird über die API-Funktion LSX_SetOsziCallBackFct an die API übergeben. Bei ihr handelt es sich um eine globale Funktion, die keine Zuordnung an eine Steuerung bzw. keine Rückschlüsse auf die Steuerung zulässt, die den CallBack auslöst.

5.2 Extended-CallBack-Funktion (ExtCallBackFct)

Die Extended-CallBack-Funktion wird in den Programmiersprachen Delphi, C++ bzw. C# wie folgt deklariert:

Deklaration Extended-CallBack-Funktion (ExtCallBackFct)		
Delphi	function ExtCallBackFct(Data: PAnsiChar; MaxLen: Integer; ChannelID: Integer; pObject: Pointer): Integer; stdcall;	
C++	int CALLBACK ExtCallBackFct(char *pcData, int lMaxLen, int lChannelID, void* pObject);	
C#	int ExtCallBackFunction(IntPtr Data, int MaxLen, int ChannelID, IntPtr pObject)	
Parameter	Data	Zeiger auf ASCII-Zeichenkette, nullterminiert.
	MaxLen	Maximale Länge der ASCII-Zeichenkette in Data.
	ChannelID	Kanal, auf den sich die ASCII-Zeichenkette bezieht. Mögliche Kanäle siehe „5.3 Kanal Nummern“.
	pObject	Zeiger auf Objekt, welches beim Aufruf der CallBack-Funktion übergeben wird.
Rückgabewert	Über den Rückgabewert vom Typ Integer kann der Kommunikations-Eingabepuffer der API beeinflusst werden. 0x00000 = Führe keine Aktion aus 0x10000 = Aktuellen String des Eingabepuffers löschen	

Diese CallBack-Funktion wird über die API-Funktion LSX_SetExtCallBackFct an die API übergeben. Sie ist an eine Steuerung bzw. ein Objekt der LSTEP-API gebunden.

Über das Objekt hinter dem Zeiger pObject können Rückschlüsse auf die aufrufende Steuerung hinterlegt werden. Das Objekt zum Zeiger pObject wird mit der Funktion LSX_SetExtCallBackFct übergeben.

Durch die Zuordnung der Extended-CallBack-Funktion an eine Steuerung löst diese Steuerung keine Standard-CallBack-Funktion aus.

5.3 Kanal Nummern

Der Auslöser einer API-CallBack-Funktion wird über den Kanal unterschieden. Folgende Kanäle sind definiert:

1 = Oszilloskop-Kanal

2 = Fehler- / Informations-Kanal

3 = Oszilloskop Informations-Kanal

4 = Digitaler-Eingangs-Kanal

…

10000 = Bewegungs-Kanal

5.3.1 Oszilloskop-Kanal (1) und Oszilloskop Informations-Kanal (3)

Mit diesen Kanälen kann ein digitales Oszilloskop realisiert werden. Nähere Informationen dazu werden auf Anfrage bereitgestellt. Eine beispielhafte Umsetzung kann im WIN-Commander 5 Oszilloskop betrachtet werden.

5.3.2 Fehler- / Informations-Kanal (2)

Der Fehler- / Informations-Kanal erhält als Datenpaket eine Zeichenkette die durch Tabulatoren in Spalten aufgeteilt ist. Die erste Spalte enthält einen Subkanal, der als Ganzzahl dargestellt wird und einen Wertebreich von 32 Bit umfassen kann. Es werden die Subkanäle Fehler-Kanal (Subkanal 0) und Informations-Kanal (Subkanal 99) unterschieden.

Fehler-Kanal (Subkanal 0)

Ist der Subkanal gleich 0 enthält das asynchrone Datenpaket eine Fehlermeldung. Das Datenpaket der CallBack-Funktion setzt sich für Subkanal 0 wie folgt zusammen:

- Spalte 0 = Subkanal
- Spalte 1 = Achsnummer (0 = X-Achse, … 3 = A-Achse, 5 = Allgemeine Meldung)
- Spalte 2 = Fehlernummer
- Spalte 3 = Betriebszeit in Sekunden
- Spalte 4 = Millisekundenanteil der Betriebszeit

Die Fehlernummer kann in der LSTEP-Steuerungs-Dokumentation nachgeschlagen werden. Weiterhin bietet die LSTEP-API die Funktion LSX_TranslateErrMsg an, um eine Übersetzung durchzuführen. Als Quelle der Übersetzung dienen Sprachdateien, die bei der LSTEP-API beigelegt sind (z.B. LStep4deu.txt).

Informations-Kanal (Subkanal 99)

In diesem Subkanal folgt auf die Spalte 0 mit der Subkanalnummer 99 eine ASCII-Zeichenkette mit Zusatzinformationen. Diese beinhalten z.B. welcher Befehl einen Fehler ausgelöst hat. Im WIN-Commander 5 wird diese Information als Hinweis im Fehlerfenster angezeigt.

5.3.3 Digitaler-Eingangs-Kanal (4)

Über diesen Kanal wird der Zustand der digitalen Eingänge einer Steuerung signalisiert. Ausgelöst wird die Signalisierung durch einen Wechsel an den digitalen Eingängen, sodass jeder Wechsel eine Übertragung über diesen Kanal auslöst.

Zur Unterscheidung der digitalen Eingänge von verschiedenen Steckern bzw. Eingangstypen einer Steuerung wird vor dem Zustand der Eingänge eine Kennung übergeben. Die Zuordnung der Kennung ergibt sich aus der Steuerungsdokumentation. Sie wird über einen Tabulator vom Zustand der Eingänge getrennt.

Die Aktivierung dieses Kanals kann in der Steuerungsdokumentation nachgeschlagen werden.

Steuerung	Aktivierung siehe
LSTEP express Serie	diginstatus
LSTEP 2000 Serie	-
Andere	-

5.3.4 Bewegungs-Kanal (10000)

Dieser Kanal löst einen CallBack aus, wenn eine Achsmaske von einer Steuerung übertragen wird (siehe LSX_GetStatusAxis, LSX_GetStatusAxisW). In Kombination mit der Autostatus-Funktionalität der LSTEP-Steuerung kann so auf das Ende eines asynchronen Moves gewartet werden, ohne die Achsmaske kontinuierlich abzufragen.

Damit die Synchronität des Kommunikations-Eingabepuffers der API erhalten bleibt, muss bei einem asynchronen Move bei aktivem Autostatus der letzte Eintrag des Kommunikations-Eingabepuffers gelöscht werden. Dieser Eintrag entspricht der Achsmaske. Zum Löschen ist als Rückgabewert der Extended-CallBack-Funktion der Wert 0x10000 zu verwenden.

5.4 Unterstützte Steuerungen bzw. Schnittstellentypen

Die nachfolgenden Tabellen zeigen eine Übersicht, welche Kanäle von welchem Steuerungssystem und welcher Schnittstelle unterstützt werden:

Steuerung	Unterstützter Kanal					Aktivierung
	1	2	3	4	10000	
LSTEP express Serie	Ja	Ja	Ja	Ja	ja	!errorchannel 2
LSTEP 2000 Serie	-	-	-	Ja[1)]	-	-
Andere	-	-	-	-	-	-

1) Nur die PCI-Variante

Schnittstellentyp		Unterstützter Kanal					
Nr	Typ	1	2	3	4	…	10000
1	RS232	-	-	-	-	-	-
2	ArcNet	-	-	-	-	-	-
3	DPRAM / ISA-Bus	-	-	-	-	-	-
4	DPRAM / PCI-Bus	-	-	-	-Ja	-	-
5	RS232 mit RTS/CTS und erweitertem Protokoll	Ja	Ja	Ja	Ja	-	Ja
11	RS232 mit RTS/CTS	-	-	-	-	-	-

5.5 Beispielanwendungen

Als Unterstützung der Implementierung der CallBack-Funktionen dienen die folgenden Beispielprojekte:

C#_LogFile - Beispielprogramm zur Nutzung einer Log-Datei bzw. des Fehlerkanals

C++_MFC_LogFile - Beispielprogramm zur Nutzung einer Log-Datei bzw. des Fehlerkanals

Die Beispielprojekte sind dem Archiv der LSTEP-API beigelegt.

6 Fehlercodes

Zur Fehlererkennung bei der Nutzung einer Lang-Steuerung stehen zwei Methoden zur Verfügung. Zum einen das Abfragen der Fehlernummer aus der Steuerung mittels der Funktion LSX_GetError, zum anderen die Auswertung des Rückgabewertes der API-Funktionen.

6.1 Fehlernummern aus der Steuerung (GetError)

Mittels der API-Funktion LSX_GetError wird der zuletzt aufgetretene Fehler aus der Steuerung gelesen. Er wird als 32-Bit-Integer angegeben und hat einen Wert kleiner 4000. Ist der Wert gleich Null ist seit der letzten Abfrage kein Fehler aufgetreten. Neben der Funktion LSX_GetError werden diese Fehlernummern auch über den Fehler- / Informations-Kanal (2) (siehe Abschnitt 5.3.2) von der Steuerung an die Anwendung versendet. Die nachfolgende Liste zeigt die Zuordnung der Fehlernummern. Diese Zuordnung liegt der LSTEP-API als Textdatei in verschiedenen Sprachen bei (z.B. LStep4deu.txt):

0 = Alles OK
1 = Angegebene Achse nicht vorhanden
2 = Funktion kann nicht ausgeführt werden
3 = Zu viele Zeichen im Befehlsstring
4 = Unbekannter Befehl
5 = Außerhalb des gültigen Zahlenbereichs
6 = Anzahl Parameter falsch
7 = Befehl muss mit ! oder ? beginnen
8 = Kein TVR möglich, da Achse aktiv
9 = Kein Ein- oder Ausschalten der Achsen, da TVR aktiv
10 = Funktion nicht konfiguriert
11 = Kein Move-Befehl möglich, da Joystick aktiv
12 = Endschalter betätigt
13 = Funktion kann nicht ausgeführt werden, da Encoder erkannt
14 = Fehler beim Kalibrieren! Endschalter nicht korrekt freigefahren
15 = Fehler beim Kalibrieren auf Referenzmarke
16 = Save-Befehl fehlgeschlagen
17 = Achse noch in Benutzung
18 = Achse nicht bereit
19 = Achse nicht kalibriert
20 = Treiberrelais 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 X-Achse
24 = SECURITY Error Y-Achse
25 = SECURITY Error Z-Achse
26 = SECURITY Error A-Achse

27 = Not-STOPP
28 = Fehler im Türschaltersicherheitskreis
29 = Endstufen nicht eingeschaltet
30 = GAL Sicherheitsfehler
31 = Joystick lässt sich nicht einschalten, da move aktiv
32 = Vektor außerhalb des Verfahrbereiches

1009 = Kein Ein- oder Ausschalten der Achsen, da TVR aktiv
1010 = Anderer manueller Modus bereits aktiv
1011 = Servo- und Schrittmotor nicht koppelbar (Joystick)
1012 = Ausgang bereits anderer Funktion zugeordnet (digitaler Ausgang)
1013 = Hardwarelayer mehrfach zugewiesen
1014 = Axis mapping ist nicht möglich
1015 = Triggerausgabefrequenz überschritten

1030 = Konfigurierung ist aktiv
1031 = Achse nicht konfiguriert
1032 = Interner Fehler
1033 = Achse noch in Benutzung
1034 = Achse in Fehlerstatus
1035 = Achse nicht kalibriert
1036 = Achse ohne RoomMeasure
1037 = Min. Grenze unbekannt
1038 = Max. Grenze unbekannt
1039 = Notstopp ausgelöst
1040 = Endschalter angefahren
1041 = Verfahrweg zu klein
1042 = Geschwindigkeit zu klein
1043 = Ruck zu klein
1044 = Kein Trigger Endschalter rein
1045 = Kein Trigger Endschalter raus
1046 = Fahrweg geclippt
1047 = Endschalter überfahren

1052 = Geschwindigkeit zu hoch
1053 = Beschleunigung zu hoch
1054 = Ruck zu hoch

1064 = Wegstrecke zu groß
1065 = Bremse und Spannungsversorgung für Endschalter nicht gleichzeitig möglich
1066 = Keine Kommutierung nötig
1067 = Achse nicht kommutiert

1068 = Adaptiver Lageregler nicht im Servobetrieb zulässig
1069 = Filter-Zeitkonstante zu klein[1)]
1070 = Frequenzen der Bandsperre nicht möglich

1096 = Min. Endschalter aktiv
1097 = Max. Endschalter aktiv
1098 = Nicht bereit zur Autokommutierung
1099 = Kein interpolierender Geber gefunden
1100 = I²T Überwachung angesprochen (Langzeit)
1101 = I²T Überwachung angesprochen (Kurzzeit)
1102 = Überstrom Endstufe
1103 = Überstrom beim Einschalten
1104 = Überspannung
1105 = Sicherung Zwischenkreisspannung defekt
1106 = Encoderfehler: Amplitude zu klein
1107 = Encoderfehler: Amplitude zu groß
1108 = Schleppfehler zu groß
1109 = Geschwindigkeit zu groß
1110 = Motor blockiert
1111 = Motorbremse fehlerhaft
1112 = Übertemperatur der Endstufe
1113 = Motor überhitzt
1114 = Endschalter bei Autokommutierung geschaltet
1115 = Lesefehler Temeratur der Endstufe
1116 = Zielfenster nicht erreicht
1117 = Achse wird Verfahren
1118 = Schalter für min. Fahrbereich betätigt
1119 = Schalter für max. Fahrbereich betätigt
1120 = Zielposition außerhalb min. Fahrbereich
1121 = Zielposition außerhalb max. Fahrbereich
1122 = Mehrere Endschalter gleichzeitig betätigt
1123 = Endstufe durch Hardwareüberwachung ausgeschaltet
1124 = Spurfehler Encoder
1125 = Amplitude des Encoders zu klein, eventuell kein Geber angeschlossen.
1126 = Winkel bei Autokommutierung ausserhalb des Toleranzbereichs, Achse eventuell blockiert
1127 = Keine Rundachse
1128 = Kein Null-Endschalter / Keine Geber Referenzmarke
1129 = Kein Geber Interface
1130 = Gebereingang mehrfach zugewiesen
1131 = eQep-Gebereingänge nicht konfiguriert (Hardware-Konfiguration MFP)
1132 = Zielfenster nicht innerhalb erlaubter Zeit erreicht
1133 = Gebereingang nicht verfügbar

1134 = Autokommutierungsstrom größer als Nennstrom
1135 = Autokommutierungsstrom gleich Null

1139 = Schleppfehler nicht aktiviert
1140 = Motor-I²T-Strom kleiner /gleich Motor-Nennstrom

1160 = OTP Daten fehlen

1162 = Rechenzeit überschritten (Level 4)
1163 = Rechenzeit überschritten (Level 3)
1164 = Rechenzeit überschritten (Level 2)
1165 = Rechenzeit überschritten (Level 1)
1166 = Gerät im Status „System Error“
1167 = I²T Gerät Fehler

1171 = Amplitudenfehler Kommutierngs-Encoder (Eingänge ENC1 bis ENC8)
1172 = Frequenzfehler Kommutierungs-Encoder (Eingänge ENC1 bis ENC8)

1174 = Amplitudenfehler Lagegeber1-Encoder (Eingänge ENC1 bis ENC8)
1175 = Frequenzfehler Lagegeber1-Encoder (Eingänge ENC1 bis ENC8)

1177 = Amplitudenfehler Lagegeber2-Encoder (Eigänge ENC1 bis ENC8)
1178 = Frequenzfehler Lagegeber2-Encoder (Eingänge ENC1 bis ENC8)

1193 = Warnung: Übertemperatur Endstufe
1194 = Warnung: Motortemperatur zu hoch
1195 = Treiberspannung unterschritten
1196 = Achse deaktiviert
1197 = Zwischenkreisspannung zu niedrig
1198 = Zwischenkreisspannung zu hoch

1250 = Oszilloskop Pretrigger Position größer als Oszilloskop Datengröße

Nachfolgend sind die Befehler zur Fehlermeldung 1069 aufgeführt. Die minimalen Filterzeiten sind in den Beschreibungen der Befehle aufgeführt:

- LSX_SetAccelFeedForwardOutPass / accelfeedforwardoutpass
- LSX_SetPosConOutPass / posconoutpass
- LSX_SetSpeedConOutPass / speedconoutpass
- LSX_SetActSpeedFilterConst / actspeedfilterconst
- LSX_SetActAccelFilterConst / actaccelfilterconst
- LSX_SetMonitoringVelFilter / monitoringvelfilter
- LSX_SetJoyOutPass / joyoutpass
- LSX_SetTippOutPass / tippoutpass
- LSX_SetTrackBallOutPass / tboutpass

6.2 Rückgabewert von LSTEP-API Funktionen

Alle Befehle der LSTEP-API liefern einen 32 Bit Integer Rückgabewert. Ist dieser Wert 0 oder 4100 wurde der API-Befehl fehlerfrei ausgeführt. Ist ein Fehler aufgetreten, wird eine Fehlernummer größer 4000 in der API erzeugt und kann über den Rückgabewert weiter verarbeitet werden. Die Nachfolgende Liste zeigt die erzeugten Fehlernummern:

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 Steuerung
4008 = Fehler bei Befehlsübertragung an die LSTEP
4009 = Befehl wurde abgebrochen (mit SetAbortFlag)
4010 = Befehl wird von LSTEP nicht unterstützt
4011 = Joystick aktiv (kann bei SetJoystickOn/Off auftreten)
4012 = Kein Verfahrbefehl möglich, da Joystick aktiv
4013 = Regler-Timeout bei Move-Befehl
4014 = Fehler beim Kalibrieren, Endschalter nicht korrekt freigefahren
4015 = Endschalter in Verfahrrichtung betätigt
4016 = Wiederholter Vektorstart!! (Regelung)
4031 = Joystick lässt sich nicht einschalten, da move aktiv!
4032 = Softwarelimits undefiniert
4100 = Alles OK

Fehlernummer ab 4100

Diese Fehlernummern werden von der LSTEP-API generiert, wenn ein Fehler während der Ausführung eines API-Befehls aufgetreten ist und dieser von der Steuerung gemeldet wird. Zum Beispiel wenn in der Achsenmaske ein „F“ vorkommt oder der Statusstring „ERR FehlerNr“ (z.B. „ERR 27“) enthält.

Um diesen Fehlernummern einen beschreibenden Fehlertext zuzuordnen, ist der Wert 4100 von der Fehlernummer abzuziehen und die resultierende Nummer unter „6.1 Fehlernummern aus der Steuerung (GetError)“ nachzuschlagen.

7 Häufige Fragen & Antworten

Übersicht
Wie wird die LSTEP4X.DLL in einem MS Visual C++ Projekt eingebunden?
Wie initialisiere ich mit der LSTEP-API die Verbindung zur LSTEP?
Welcher der Connect-Befehle sollte verwendet werden?

Wie installiere ich den Treiber für die LSTEP-PCI?
Warum bekommt mein Programm mit der LSTEP4X.DLL keine Verbindung zur LSTEP-PCI?
Im Ablauf, in der LSTEP4X.DLL oder in meinem Programm ist ein Fehler aufgetreten. Wo liegt die Ursache, und wie lässt sich das Problem lösen?
Kann während Verfahrbefehlen der Status von Eingängen, die aktuelle Position u.ä. abgefragt werden?
Warum werden während der Ausführung von LSTEP-API-Funktionen Messages verarbeitet, und wie kann man dies deaktivieren?
Wann sind Moves mit bzw. ohne Wait zu verwenden?
Wie kann ich mit dem LSTEP-API einzelne Achsen der LSTEP unabhängig voneinander verfahren?
Wie kann ich mehrere LSTEP-PCI-Karten in einem PC verwenden?
Ist die LSTEP-API kompatibel zur MCL bzw. zum alten Register-Befehlssatz?
Warum bekomme ich in MS Visual C++ bei Einbindung von LStep4X.cpp die Meldung “fatal error C1010“?
Wie kann ich einen speziellen/neuen LSTEP-Befehl verwenden, für den es keine passende LSTEP-API-Funktion gibt?
Warum sehe ich im Debugger meiner Entwicklungsumgebung bei Verwendung der LSTEP-API die Meldung „First chance exception“, „Exception: Timeout read RS232!“ o.ä.?
Wie kann ich mit der LSTEP-API eine Art Joystick realisieren, also eine Achse solange fahren, bis eine Taste wieder losgelassen wird?
Wie kann ich die Einstellungen der LSTEP dauerhaft speichern?
Wie viele Einträge passen in das Protokollfenster der LSTEP-APIs?

Wie wird die LSTEP4X.DLL in einem MS Visual C++ Projekt eingebunden?

- Projekt erzeugen
- LSTEP4X.DLL, LSTEP4X.h, LSTEP4X.cpp in Projektordner kopieren
- LSTEP4X.h und LSTEP4X.cpp in Projekt einfügen
- Menü: Projekt\ Einstellungen\ C/C++ Option: [vorkompilierte Header nicht verwenden] wählen
- in LSTEP4X.h #include „stdafx.h“ einbinden
- in Projektname_Dlg.h #include „LSTEP4X.h“ einfügen
- die erforderliche Instanz in public einbinden
  Beispiel: CLStep* MyLStep = new CLStep();

Wie initiiere ich mit der LSTEP-API die Verbindung zur LSTEP?

Die Verbindung zur LSTEP-API wird mit einem der Connect-Befehle (Connect, ConnectEx, ConnectSimple) initialisiert.

Welcher der Connect-Befehle sollte dazu verwendet werden?

Abgesehen von einigen Sonderfällen sollte immer ConnectSimple verwendet werden.

Funktion	Verwendungszweck
ConnectSimple	Bei der direkten Übergabe der Schnittstellenparameter
Connect	Nach zuvorigem Laden der Schnittstellenparameter aus einer .ini Datei mittels LoadConfig
ConnectEx	Beim Laden der Schnittstellenparameter aus einer Datenstruktur

Wie installiere ich den Treiber für die LSTEP-PCI?

Nach korrektem Einbau der LSTEP-PCI fordert Windows beim Start einen Treiber für ein Gerät des Typs „Netzwerkcontroller“ an. In diesem Dialog-Fenster klicken Sie auf den Button „Durchsuchen“ o.ä. und wechseln dann in das Verzeichnis, in welches die Dateien der LSTEP-API entpackt wurden. Im Unterordner „LStepPCI“ sind die Treiber-Dateien und die Inf-Dateien, welche für die Treiber-Installation benötigt werden, enthalten.

Warum bekommt mein Programm mit der LSTEP4X.DLL keine Verbindung zur LSTEP-PCI?

Sie sollten zunächst im Windows-Geräte-Manager überprüfen, ob dort die eingebaute LSTEP-PCI als Gerät eingetragen ist. Außerdem **muss die Datei DRVX40.DLL im Verzeichnis der LSTEP4X.DLL, ihres Programms oder einem Windows-Systemordner liegen.** Sie finden diese Datei im Unterordner „LStepPCI“ der LSTEP-API.

Im Ablauf, in der LSTEP4X.DLL oder in meinem Programm ist ein Fehler aufgetreten. Wo liegt die Ursache, und wie lässt sich das Problem lösen?

Damit eine Fehlerdiagnose möglich ist, sollten sie unbedingt die Protokollierung der LSTEP-API mit SetWriteLogText einschalten. Anschließend sollten Sie versuchen,

den aufgetretenen Fehler bei laufender Protokollierung zu reproduzieren. Die Log-Datei (LStep4.log) können Sie dann zur Analyse an uns mailen.

Kann während Verfahrbefehlen der Status von Eingängen, die aktuelle Position u.ä. abgefragt werden?

Ja, zum Beispiel indem man über einen Windows-Timer oder einen zweiten Thread Funktionen wie GetPos, GetDigitalInputs während eines Verfahrbefehls aufruft. Es ist aber nicht möglich, während eines Verfahrbefehls mit Wait = True den Befehl WaitForAxisStop aufzurufen.

Warum werden während der Ausführung von LSTEP-API-Funktionen Messages verarbeitet, und wie kann man dies deaktivieren?

Die LSTEP-API verarbeitet während des Wartens auf Rückmeldungen der LSTEP im Main-Thread Messages zum Durchführen von z.B Abbrüchen bzw. Achsen Stopps. Wenn sie das Message-Dispatching abschalten wollen oder durch eigenen Code ersetzen wollen, können sie die Funktion SetProcessMessagesProc zum Setzen einer Callback-Prozedur verwenden.

Wann sind Moves mit bzw. ohne Wait zu verwenden?

Move-Befehle mit WaitForAxisStop sind zu verwenden, wenn alle Achsen synchron und linear interpoliert verfahren werden sollen. Die Steuerung nimmt neue Move-Befehle erst entgegen, wenn alle Achsen stehen.

Move-Befehle ohne WaitForAxisStop sind zu verwenden, wenn die Achsen asynchron verfahren werden sollen. Der Anwender hat in diesem Fall dafür zu sorgen, dass nur diejenige Achse die steht auch einen neuen Move-Befehl bekommt.

Wie kann ich mit der LSTEP-API einzelne Achsen der LSTEP unabhängig voneinander verfahren?

Die Verfahr-Befehle der LSTEP-API bieten zwei verschiedene Möglichkeiten:

Wird der Parameter Wait = True bei Verfahrbefehlen gesetzt, kehrt die Funktion erst zurück, nachdem die Achsen ihre Zielposition erreicht haben.

Wird hingegen dieser Parameter Wait = False gesetzt, sendet die LSTEP-API-Funktion nur den Verfahrbefehl und kehrt unmittelbar zurück, ohne auf die Ausführung der Bewegung zu warten.

Eine unabhängige Bewegung kann daher durch das Ausführen eines MoveAbsSingleAxis mit Wait = False für z.B. die X-Achse ausgeführt werden um etwas später einen MoveAbsSingleAxis mit Wait = False für die Y-Achse aufruft. Anschließend werden beide Achsen asynchron verfahren. Um festzustellen, ob die Achsen ihre Zielposition erreicht haben, kann der Befehl **WaitForAxisStop** benutzt werden.

Beispiel:

```
LSX.MoveAbsSingleAxis(Xaxis, 10, false); // Verfahre die X-Achse asynchron
Delay(1000); // Warte 1s bis zum Start der Y-Achse
LSX.MoveAbsSingleAxis(Yaxis, 20, false); // Verfahre die Y-Achse asynchron
LSX.WaitForAxisStop(3, 0, flag); // Warte bis X- und Y-Achse gestoppt haben, ohne Timeout
```

Es ist aber **nicht möglich, Move-Befehle mit Wait = True und solche mit Wait = False gleichzeitig zu verwenden**. Dies führt zu permanenten oder sporadischen Fehlern in der Kommunikation.

Beispiel (Nicht erlaubt):

LSX.MoveAbsSingleAxis(Xaxis, 10, false); // Verfahre die X-Achse asynchron
LSX.MoveAbsSingleAxis(Yaxis, 20, true); // Verfahre die Y-Achse asynchron ohne auf das Ende des asynchronen Verfahrbefehls gewartet zu haben

Wie kann ich mehrere LSTEP-PCI-Karten in einem PC verwenden?

Die Vorgehensweise bei der Installation ist diese wie bei einer einzelnen Karte. Nach dem Start fordert Windows den Treiber für sämtliche LSTEP-PCI-Karten an.

Doch es ist **problematisch festzustellen, welche physikalische Karte zu einer bestimmten Index-Nummer gehört**. Es ist nicht sichergestellt, dass man durch LSX_ConnectSimple(4, nil, 0, true) eine Verbindung zur LSTEP-PCI im ersten PCI-Slot des Mainboards erhält, durch LSX_ConnectSimple(4, nil, 1, true) eine Verbindung zur LSTEP-PCI im zweiten PCI-Slot erhält etc. Deshalb sollte zur eindeutigen Identifikation der Karten die Seriennummer mit **GetSerialNr** abgefragt werden.

Ist die LSTEP-API kompatibel zur MCL bzw. zum alten Register-Befehlssatz?

Die LSTEP-API ist prinzipiell abwärtskompatibel zu dem Register-Befehlssatz, mit dem die MCL und ältere LSTEP-Steuerungen kommunizieren. Jedoch bietet dieser Befehlssatz viele Möglichkeiten nicht, die die LSTEP-API bei Steuerungen mit neuem Befehlssatz verwenden kann. Deshalb können einige LSTEP-API-Befehle wie WaitForAxisStop bei Steuerungen mit altem Befehlssatz generell nicht verwendet werden.

Warum bekomme ich in MS Visual C++ bei Einbindung von LStep4X.cpp die Meldung "fatal error C1010"?

Es handelt sich hierbei nicht um einen Fehler in der Datei LStep4X.cpp. Die Meldung tritt gewöhnlich auf, wenn der Compiler nach der vorkompilierten Header-Datei sucht und sie nicht findet. Sollte in MS Visual C++ die Meldung „fatal error C1010 precompiled header files" auftreten, muss die Option „vorkompilierte Header-Datei" für LStep4X.cpp abgeschaltet werden. Sofern Sie die MFC in Ihrem Projekt nicht verwenden, sollten Sie die Zeile #include "stdafx.h" aus LStep4X.cpp entfernen.

Wie kann ich einen speziellen/neuen LSTEP-Befehl verwenden, für den es keine passende LSTEP-API-Funktion gibt?

Die LSTEP-API-Funktion **SendString** bietet die Möglichkeit, um neue, nicht im LSTEP-API vorgesehene LSTEP-Befehle zu benutzen. Zu beachten ist, dass alle Befehle mit dem Zeichen #13 bzw. \r abschließen!

Warum sehe ich im Debugger meiner Entwicklungsumgebung bei Verwendung der LSTEP-API die Meldung „First chance exception“, „Exception: Timeout read RS232!“ o.ä.?

Interne Exceptions der LSTEP4X.DLL, die nur im Debugger sichtbar sind haben keine Bedeutung. Sie dienen zur internen Ablaufsteuerung. Bei ConnectSimple tritt häufig eine Exception auf, da die LSTEP-API versucht, den Befehlssatz herauszufinden. Dabei kommt es zu einem Timeout, wenn die Steuerung den getesteten Befehlssatz nicht unterstützt. Die auftretenden Exceptions könnnen meist in der Entwicklungsumgebung ignoriert werden.

Wie kann ich mit der LSTEP-API eine Art Joystick realisieren, also eine Achse solange fahren, bis eine Taste wieder losgelassen wird?

Ein solcher Tasten-Joystick kann folgendermaßen implementiert werden:

Bei Tastendruck wird die Achse mit einem sehr langen Vektor gestartet, der noch innerhalb des Verfahrbereichs der Achse liegt. Anschließend ist ein **MoveRelSingleAxis(Xaxis, 100000, false)** auszuführen. Wichtig ist, den Parameter Wait = False zu setzen, damit das Programm nicht auf das Beenden der Bewegung wartet. Beim Loslassen der Taste ruft man anschließend den Befehl **StopAxes** auf.

Wie kann ich die Einstellungen der LSTEP dauerhaft speichern?

Der LSTEP-API-Befehl **LstepSave** kann eingesetzt werden, um einmal gemachte Einstellungen (Spindelsteigungen, Getriebefaktoren, Achsenströme usw.) auch nach Reset der LSTEP zu erhalten. Ob Ihre LSTEP diesen Befehl unterstützt, können Sie der Dokumentation entnehmen.

Wie viele Einträge passen in das Protokollfenster des LSTEP-APIs?

Das Protokollfenster des LSTEP-APIs kann 20.000 Einträge fassen. Treten mehr Einträge auf wird das Protokollfenster gelöscht und erneut gefüllt.

Wie viele Einträge können in die Log- Datei geschrieben werden?

In die Log- Datei wird so lange geschrieben, bis das Programm beendet wird oder die Festplatte voll ist. Dieses Verhalten kann mit der Funktion SetExtValue verändert werden.