# Feinpositioniersysteme

## LSTEP *express*


## LSTEP PCI *express*

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

**Sehr geehrter Kunde!**

**Vielen Dank, dass Sie sich für eine Steuerung aus unserem Hause entschieden haben!**

Mit dem Gerät haben Sie eine Positioniersteuerung gewählt, die bei einem geringen Platzbedarf anspruchsvolle Positionieraufgaben automatisiert ausführt. Die hohe Präzision der Steuerung eröffnet Ihnen weite Anwendungsmöglichkeiten. Die Schrittauflösung von bis zu 1.638.400 Schritten pro Motorumdrehung, bei einem zweihundertschrittigen Motor bieten Auflösungen im sub-µm-Bereich. Weiter bietet der „closed loop" Betrieb in Verbindung mit einer hochauflösenden Geberauswertung bei optischen und magnetischen Messsystemen eine sehr hohe Positioniergenauigkeit.
Die vielen Zusatzfunktionen, wie z.B. Snapshot, Triggerout, Takt- und Drehrichtungs- Ein- und Ausgänge machen die Steuerung zu einem idealen Partner für viele Applikationen.

Vor der Inbetriebnahme Ihrer Steuerung bitten wir Sie, diese Anleitung sorgfältig und in Ruhe zu lesen.

**Achten Sie insbesondere auf die Sicherheitshinweise!**

Inhaltliche Änderungen behalten wir uns vor. Wir haften nicht für etwaige Fehler in dieser Dokumentation. Bedingt durch ständige technische Weiterentwicklungen unserer Produkte können mitunter leichte Abweichungen zwischen den Beschreibungen in dieser Dokumentation und Ihrer Steuerung bestehen. Eine Haftung für mittelbare Schäden, die im Zusammenhang mit der Lieferung oder dem Gebrauch dieser Dokumentation entstehen, ist ausgeschlossen, soweit dies gesetzlich zulässig ist.

## Schutzvermerk nach DIN 34

# Inhaltsverzeichnis

# 1 Sicherheitshinweise

- Instandsetzungs- und Reparaturarbeiten dürfen nur von Fachpersonal durchgeführt werden, das besonders geschult und mit der Steuerung bestens vertraut ist und nur nach schriftlicher Genehmigung von Lang GmbH & Co. KG. Allen anderen Personen sind Reparaturarbeiten untersagt.

- **LSTEPexpress:** Vor dem Öffnen des Gerätes ist der Netzstecker zu ziehen!

- **LSTEP-PCIexpress: -**Schalten Sie vor und während der Montage der Karte in den PC, sowie bei der Montage des Zubehörs alle Teile spannungsfrei. –Installieren Sie die Karte derart, dass keine Späne, Flüssigkeiten oder andere Gegenstände mit der Karte in Berührung kommen. –Installieren Sie die Karte so, dass keine Hitzestaus entstehen. Die max. Umgebungstemperatur entnehmen Sie den technischen Daten. –Vergewissern Sie sich, dass die Stromversorgung Ihres PCs und falls vorhanden des externen Motorspannungsnetzteils für den Betrieb der Karte ausreicht. -Vergewissern Sie sich vor dem Einschalten über die korrekte Montage aller Komponenten.

- Die Steuerungen besitzen Teile, die gegen elektrostatische Entladung empfindlich reagieren (ESD-empfindlich). Erden Sie alle Teile die mit der Steuerung in Berührung kommen, auch sich selbst.

- Es dürfen nur Geräte angeschlossen werden die von der Fa. Lang spezifiziert sind. Zuwiderhandlungen können zu Zerstörung der Steuerung bzw. des angeschlossenen Gerätes führen!

- Stellen Sie sicher, dass die Steuerung in Verbindung mit Ihre Anwendung den dafür geltenden Sicherheitsbestimmungen und Rechtsvorschriften entspricht. Diese Steuerung ist zur EN61010-1:2001 (Sicherheitsbestimmungen für elektrische Mess-, Steuer-, Regel- und Laborgeräte) konform. Beachten Sie, dass die effektive Motorspannung in ihrer Höhe der DC-Spannung am Motorspannungsversorgungsstecker entsprechen kann.

- Gefahrbringende Bewegung: Je nach Konfiguration kann es sein, dass nach dem Einschalten der Joystick aktiv ist. Stellen Sie sicher, dass weder in dieser noch in einer anderen Betriebsart gefahrbringende Bewegungen entstehen können.

- Teile der Steuerung können im Betrieb sehr heiß werden. Verhindern Sie das Berühren der Platine während und kurz nach dem Betrieb. Eine Abkühlzeit von min. zwei Minuten bei Umgebungstemperatur ist einzuhalten. Verhindern Sie den Kontakt von Gegenständen mit der Oberfläche der Platine.

- Der Netzschalter der Steuerung, oder die Steckdose an der die Steuerung angeschlossen ist muß immer erreichbar sein, damit Jederzeit die Steuerung allpolig vom Netz getrennt werden kann!

- **In eingeschaltetem Zustand dürfen keine Kabel gesteckt oder abgezogen werden!**

# 2 Funktionsbeschreibung

Mit der Schrittmotorsteuerung LSTEPexpress (-PCIexpress) werden Koordinatentische für z.B. Mikroskope oder Fertigungsabläufe mit Auflösungen bis 0,00001 mm betrieben. Die Steuerung zeichnet sich durch hohe Laufruhe der Motoren aus. Durch das dynamische Mikrostep - Antriebs - Prinzip lassen sich trotz einer hohen Auflösung von bis zu 1.638.400 Mikroschritten je Motorumdrehung hohe Drehzahlen von bis zu 70 U/sec bei einem 200 schrittigen Motor erreichen.
Die Steuerung arbeitet mit linearer Interpolation (alle Achsen erreichen gleichzeitig die Zielposition), bei den Achsen die gemeinsam gestartet werden. Alle Achsen können aber auch asynchron verfahren werden. Die Steuerung verfügt über eine automatische frei programmierbare Rampengenerierung und Ruckbegrenzung. Beschleunigung und Verzögerung können dabei getrennt eingestellt werden. Die LSTEPexpress kann alleinstehend oder über einen PC gesteuert betrieben werden. Eine Positionsanzeige (Option) an der Frontplatte sowie ein Joy-Stick ergänzen dabei das Gerät. Der Befehlssatz der LSTEPexpress (-PCIexpress) baut auf dem Befehlssatz der LSTEP Serie auf, welcher um bedeutende neue Funktionen erweitert wurde.
Für einen sauberen Ablauf und eine sichere Positionierung sollten Motoren mit einem Schrittwinkelfehler < ± 3% eingesetzt werden. Um die maximale Drehzahl zu erreichen sollten möglichst niederohmige Motoren mit geringer Induktivität eingesetzt werden.
Zur Vermeidung einer unnötigen Erwärmung der Motoren senkt die LSTEPexpress (-PCIexpress) in jeder Betriebspause (auch bei Joy-Stick - Betrieb) den Motorstrom auf den eingestellten Ruhestrom ab.

## 2.1 Schnittstelle

Als Standard Schnittstellen zum übergeordneten PC stehen eine USB, die RS232 Schnittstelle oder der PCIexpress Bus zur Verfügung.

### 2.1.1 Betrieb ohne Steuerrechner

Einfache Bewegungen sind mit der LSTEPexpress auch ohne Steuerrechner durchführbar. Dazu wird der Joy-Stick so konfiguriert, dass er nach dem Einschalten der Steuerung aktiv ist. Nun können mit dem Joy-Stick beliebige Positionen angefahren werden. Bei Steuerungen mit LC-Display wird die augenblickliche Absolutposition dabei ständig angezeigt. Weiterhin können mit Hilfe der Schalter "CLEAR" die Achsen einzeln auf Null gesetzt werden.

## 2.2 Bedienelemente

Die Anzeige (nur bei Geräten mit Anzeige) und sämtliche Bedienungselemente mit Ausnahme des Netzschalters befinden sich an der Frontplatte.

Aktuell noch nicht verfügbar.

# 3 Inbetriebnahme

## 3.1 Anschlüsse

- Motoren über die mitgelieferten Kabel anschließen.
- Inkrementale Messsysteme anschließen (wenn vorhanden).
- Joy-Stick anschließen und mit den Schiebern verriegeln.
- Rechner über Interface-Kabel anschließen (nur bei LSTEPexpress).
- Netz anschließen (nur bei LSTEPexpress) .
- 12V Versorgung über 4pol PC-Netzteilstecker auf ST10 (nur bei LSTEP-PCIexpress)
- Bei Bedarf Motorspannung >12V…48V von externem Netzteil auf ST17 (nur bei LSTEP-PCIexpress)

## 3.2 Anschluss inkrementaler Messsysteme

An die Steuerung können inkrementale Dreh- oder Lineargebersysteme zur Erkennung bzw. Vermeidung eines Schrittversatzes angeschlossen werden. Somit wird ein closed-loop Betrieb ermöglicht. Die Einsatzmöglichkeiten sind hierbei nicht auf optische Messsysteme beschränkt. Z.B. können auch induktive oder magnetoresistive Systeme ausgewertet werden, sofern deren Ausgangssignale die spezifizierten Grenzwerte einhalten. Das optionale Geberinterface ermöglicht den Anschluss von Gebersystemen mit sinusförmigen Ausgangssignalen.

## 3.3 Funktionstest

- Gerät einschalten
  Nach jedem Einschalten führt die Steuerung selbsttätig eine Kalibrierung des angeschlossenen Joystick durch, die ca. 5s in Anspruch nimmt. Um eine korrekte Kalibrierung zu gewährleisten, darf der Joystick während dieser Zeit nicht ausgelenkt werden.
- Die LSTEPexpress hat Standart mäßig keinen Joystickschalter, kann aber so konfiguriert werden, dass nach dem Einschalten der Steuerung der Joystick aktiv ist. Ist der Joystick nicht aktiv, sollten die weiteren Tests mit dem WinCommander durchgeführt werden.
- Joy-Stick in allen Richtungen auslenken: Die Motoren laufen entsprechend der Auslenkung. Wenn keine Reaktion erfolgt, dann Motor- und Joy-Stick - Anschlüsse prüfen. Sind die Anschlüsse in Ordnung, dann sollte man über den mitgelieferten WinCommander die Endschalterpolarität überprüfen. Führt auch dies nicht zu einem Erfolg, dann sollte das Gerät auf versteckte Transportschäden untersucht werden.
- Test mit dem mitgelieferten WinCommander oder Funktionsaufruf (siehe Befehlssatz) über Ihre eigene Software.

## 3.4 Firmware update

Die Steuerung kann sehr einfach mit Programm Updates auf den neuesten Stand gebracht werden.

- Kopieren Sie das Flashtool„TIC2000“ auf Ihren PC.
- Kopieren Sie die neue Firmware „*.hex“ auf Ihren PC
- Öffnen Sie das Flashtool
- Wählen Sie die Schnittstelle des PC aus, die Sie mit der Steuerung verbunden haben,
- Machen Sie alle Einstellungen so wie unter Hilfe beschrieben.
- Der Dip-Schalter "1" auf der Rückwand der LSTEPexpress muss auf ON gestellt und dann die Steuerung eingeschaltet werden. Bei der LSTEP-PCIexpress machen Sie nach dem Einschalten von Dip-Schalter "1" mit Dip-Schalter "2" einen Reset (Ein- und Ausschalten) "
- Klicken Sie auf Button „Program*.hex
- Wählen Sie die neue Firmware aus und klicken auf „Öffnen“.
- Ist der Programmiervorgang beendet, muss der Dip-Schalter "1" wieder in seine Ausgangsposition gebracht werden.
- Nach einem erneuten Betätigen des Reset Tasters oder dem Ein.- und Ausschalten der Steuerung arbeitet die Steuerung mit der neuen Firmware.

# 4 Der Befehlssatz der LSTEP-PCIexpress und LSTEPexpress

Zur besseren Übersicht werden alle Befehle und Parameter die an die Steuerung geschickt werden, sowie alle Rückmeldungen der Steuerung als ASCII-Charakter übertragen. Dies hat den Vorteil, dass die Befehle einerseits über ein normales Terminal manuell vorgegeben werden können. Zum anderen erleichtern diese Klartextbefehle die Fehlersuche, wenn die Befehle über ein kundenspezifisches Programm vorgegeben werden.

Kommandos oder Parameter, die an die Steuerung übertragen werden, beginnen mit einem Ausrufezeichen „!“. Abfragen werden durch ein Fragezeichen „?“ gekennzeichnet. Beispielsweise bedeuten:

| | |
|---|---|
| ***!cal*** | ***Kalibrieren*** |
| ***?status*** | ***Auslesen des Status*** |

| | |
|---|---|
| **Hinweis:** | Bei Befehlen, die nur ein Schreiben oder nur ein Lesen zulassen, können die Zeichen „!“ bzw. „?“ entfallen. |

Manche Befehle, z.B. die Vorgabe von Verfahrbewegungen, erfordern die Übergabe von Parametern. Diese werden im Anschluss an den eigentlichen Befehl übertragen. Zwischen Kommandotext und Parametern sowie zwischen verschiedenen Parametern müssen Leerzeichen als Trennung eingefügt und übertragen werden.

***moa 45 13 20*** ***Verfahre x, y und z an die Positionen 45, 13 und 20***

Jeder Befehl muss mit einem Carriage Return (CR) abgeschlossen werden. Dieses Zeichen wird im ASCII-Zeichensatz folgendermaßen dargestellt:

| Symb. Name | dez. Wert | hex. Wert | bin. Wert |
|---|---|---|---|
| CR | 13 | 0xD | 00001101 |

## 4.1 Kurzbeschreibung der LSTEP express-Befehle

| Befehl | Beispiel | Bemerkung | Kap. 4 Seite |
|---|---|---|---|
| **Schnittstelle** | | | |
| baud | (?) !baud 9600 | Baudrate auf 9600 einstellen | 12 |
| **Steuerungs-Informationen** | | | |
| ver | ?ver | Versionsnummer auslesen | 9 |
| iver | ?iver | Interne Versionsnummer auslesen | 9 |
| det | ?det | detallierte Versionsnummer auslesen | 9 |
| readsn | ?readsn | Seriennummer der Steuerung auslesen | 11 |
| **Einstellungen** | | | |
| configmaxaxis | (?)!configmaxaxis 4 | Eine 4-Achsen Steuerung wird mit 4-Achsen betrieben | 17 |
| dim | (?) !dim 1 | Einstellen der Einheit in µm | 18 |
| pitch | (?) !pitch 1 1 1 (y 4) | Einstellen der Spindelsteigung X Y Z oder nur Y | 19 |
| geardenominator | (?) !geardenominator | Getriebefaktor / Nenner | 19 |
| gearnumerator | (?) !geardenominator | Getriebefaktor / Zähler | 20 |
| accel | (?) !accel 1 1 1 (x 1) | Einstellen der Beschleunigung X Y Z oder nur X | 21 |
| acceljerk | (?) ! acceljerk | Ruck Beschleunigung | 22 |
| decel | (?) ! decel | Verzögerung | 23 |
| deceljerk | (?) ! deceljerk | Ruck Verzögerungung | 24 |
| vel | (?) !vel 10 10 10 (x 20) | Einstellen der Geschwindigkeit X Y Z oder nur X | 25 |
| pot | (?) !pot 1(0) | Speedpoti Ein.-/Ausschalten | 52 |
| motorcurrent | (?) !motorcurrent 1 2 2.5 | Motorstromeinstellung: X=1A Y=2A Z=2,5A | 26 |
| maxcur | ?maxcur | alle max.Ströme werden angezeigt (=Konfiguration) | 26 |
| reduction | (?) !reduction 0.5 0.5 0.5 | Stromabsenkung auf 50% in allen Achsen | 27 |
| curdelay | (?) !curdelay 1000 | Verzögerung für die Stromabsenkung ( 0 - 10000 ms ) | 27 |
| axis | (?) !axis1 0 1 (y 1) | Achsen Ein.- und Ausschalten | 29 |
| axisdir | (?) !axisdir 0 1 0 | Motordrehrichtung für Y-Achse gedreht, incl. Endschalter | 29 |
| Motorfielddir | (?) ! Motorfielddir | Nur Motordrehrichtung tauschen | 30 |
| Swchange | (?) ! Swchange | Nur Endschalter tauschen | 31 |
| caliboffset | (?) !caliboffset 1 1 1 1 | Der Nullpunkt wird um 1mm verschoben bei Dim 2 | 44 |
| rmoffset | (?) !rmoffset 1 1 1 1 | Die Endposition wird um 1mm verschoben bei Dim 2 | 44 |
| caldir | (?) !caldir z 1 | Die Z-Achse wird in positive Richtung Kalibriert | 45 |
| calibrmbspeed | !calibrmbspeed 10 | bei "cal"+"rm" wird mit 0,1U/s aus den Endschaltern gefahren (5...100) | 46 |
| calibrmaccel | (?) ! calibrmaccel | Beschleunigung für Kalibrieren und Hubmessen | 47 |
| calibrmjerk | (?) ! calibrmjerk | Ruck für Kalibrieren und Hubmessen | 48 |
| calibrmvel | (?) ! calibrmvel | Geschwindigkeit für Kalibrieren und Hubmessen | 49 |
| Save | save | die aktuellen Parameter werden ins Flash gebrannt | 12 |

| Befehl | Beispiel | Bemerkung | Kap.4 Seite |
|---|---|---|---|
| **Einstellungen** | | | |
| reset | reset | Die Software wird in den Startzustand versetzt | 11 |
| pa | pa 1 1 1 1 | Poweramplifier 1= Endstufe eingeschaltet 0=ausgeschaltet | 11 |
| motorbrake | (?)!motorbrake z 1 | Beim Ausschalten der Z-Endstufe wird die Bremse aktiv | 34 |
| motorbrakeswitchondelay | (?)!motorbrakeswitchondelay | Verzögert den Ausgang beim Einschalten der Endstufe | 35 |
| motorbrakeswitchoffdelay | (?)!motorbrakeswitchoffdelay | Verzögert den Ausgang beim Ausschalten der Endstufe | 35 |
| stoppol | (?) !stoppol 0 oder 1 | Der Stopeingang (MFP) ist low oder high aktiv | 40 |
| stopdecel | (?) ! stopdecel 2 | Die Bremsbeschleunigung, wenn der Stopeingang aktiv wird, ist 2m/s² | 41 |
| stopdeceljerk | (?) ! stopdeceljerk | Eingestellter Ruck, für aktiven Stopeingang | 42 |
| quit | !quit | Nach einem aufgetretenen Fehler und dem Beheben, muss dies mit „quit“ bestätigt werden | 36 |
| validconfig | !validconfig | Einstellungen übernehmen | 36 |
| validpar | !validpar | Aktivierung: der Regler Parameter | 36 |
| motorpolepairs | !motorpolepairs 50 | Einstellung für einen 200 Schrittigen (1,8°) Motor | 31 |
| motorpolepairres | !motorpolepairres 1000 | Die Auflösung beträgt 1000 Mikroschritte / Signalperiode | 32 |
| motormaxvel | !motormaxvel x 1000 | Die einstellbare Geschwindigkeit für den X-Motor wird auf 1000 1/min begrenzt | 32 |
| motortype | !motortype 0 oder 1 | 0 = 2 Phasen Rotativer Schrittmotor<br>1 = 2 Phasen Linear Schrittmotor | 33 |

| | | | |
|---|---|---|---|
| **Statusabfragen** | | | |
| Autostatus | (?) !autostatus 0 (0-4) | Einstellung der Rückmeldung von der Steuerung | 13 |
| Status | ?status | Liefert den aktuellen Zustand der Steuerung | 13 |
| Statusaxis | ?statusaxis | aktuellen Zustand der einzelnen Achsen (@,M,J,C,S,A,D,-,) | 15 |
| Err | ?err | liefert die aktuelle Fehlernummer | 16 |
| Statuslimit | ?statuslimit | A=Kalibriert;D=Hubgemessen; L=Softwareendsch.;-=Grundeinst. | 38 |
| Sysstatus | ?sysstatus | Systemstatus mit Text Rückmeldung | 13 |
| Sysstat | ?sysstat | Systemstatus mit Nummerischer Rückmeldung | 14 |

| Befehl | Beispiel | Bemerkung | Kap. 4 Seite |
|---|---|---|---|
| **Fahrbefehle und Positionsverwaltung** | | | |
| cal | !cal | Kalibrieren | 43 |
| rm | !rm | Tischhubmessen | 43 |
| moa | !moa 10 10 10 (x 10) | Absolutposition anfahren X Y Z (nur X) | 50 |
| mor | !mor 4 4 4 (y 4) | Relativ-Positionieren X Y Z (nur Y) | 50 |
| m | !m | Start einer Verfahrbewegung (Strecke mit mor od.distance) | 51 |
| distance | (?) !distance | Setzen der Strecke für X Y Z (starten mit "m") | 51 |
| a | !a | Abbruch (Stop) | 53 |
| pos | (?) !pos 0 0 0 (z 0) | Position setzen oder lesen | 52 |
| **Programmieren und Anfahren von Tabellenpositionen** | | | |
| tpos | !tpos 1 10 20 30 40 | programmieren der Tabellenposition 1 für alle 4 Achsen auf die Position x y z a / 10 20 30 40 | 53 |
| | ?tpos 2 2 2 2 | abfragen der Tabellenposition 2 aller Achsen | |
| mtpa | !mtpa 1 1 1 1 | fährt alle 4 Achsen auf Tabellenposition 1 | 54 |
| idt | !itd a 50 | eine Umdrehung der a-Achse wird in 50 gleiche Winkel je 7,2° aufgeteilt | 54 |
| mita | !mita a 10 | Die a-Achse wird auf Position 10 (72° bei itd=50) gefahren. | 55 |
| **Joystick** | | | |
| speed | (?) !speed 5 5 5 (y 10) | Digitaler Joystick, alle Achsen drehen mit 5U/s od.Y mit 10 | 56 |
| joydir | (?) !joydir 1 1 -1 | Motordrehrichtung für Joystick einstellen | 57 |
| joyvel | (?) ! joyvel 10 10 10 | Die max. Geschwindigkeit im Joystickbetrieb ist 10 Umdr./s | 57 |
| joyoutpass | (?)!joyoutpass x 100 | Filterzeitkonstante für Eingangswerte ist 100µs | 58 |
| joyenable | !joyenable 0 oder 1 | Joystick ist nicht- oder angeschlossen | 59 |
| joyredcur | !joyredcur 0 oder 1 | Joystick ohne oder mit Stromreduzierung | 58 |
| joy | (?) !joy 1 oder 0 | Joystick Ein.-/Ausschalten | 60 |
| joywindow | (?) !joywindow 10 | Bereich einstellen i.d. sich die Achsen nicht bewegen (0-100) | 59 |
| joytoaxis | !joytoaxis 1 2 3 | Joystick X auf Achse 1, Y auf Achse 2, Z auf Achse 3 Jede Joystickachse kann beliebig zugeordnet werden 0 = keine Joystickachse | 60 |

| Befehl | Beispiel | Bemerkung | Kap.4 Seite |
|---|---|---|---|
| **Manuelle Bedienung über Trackball, Handrad oder Tipp-Betrieb** | | | |
| tippvel | (?)!tippvel 10 10 10 10 | Tippbetrieb Geschwindigkeit | 61 |
| tippoutpass | (?)!tippoutpass x 10 | Filterzeitkonstante ist 10µs | 61 |
| tippdir | (?)!tippdir 0 1 0 0 | Tippbetrieb-Richtung | 62 |
| tippenable | (?)!tippenable 1 1 1 0 | Tippbetrieb Freigabe | 62 |
| tippredcur | (?)!tippredcur 1 1 1 0 | Tippbetrieb mit Stromreduzierung | 63 |
| tipp | !tipp o oder 1 | Tippbetrieb einschalten | 63 |
| tbenable | !tbenable x 0 oder 1 | Trackballbetrieb Aus- und Einschalten | 65 |
| tbtoaxis | !tbtoaxis x 0 od. 1 od. 2 | Zuordnung der Trackballachse- Aus-horizontal-vertikal | 66 |
| tbdir | !tbdir x 0 oder 1 | Richtungsvorgabe der Trackballachse | 65 |
| tbredcur | !tbredcur x 0 oder 1 | Stromreduzierung für Trackballbetrieb Aus oder Ein | 66 |
| tbvel | !tbvel x 10 | Geschwindigkeitsvorgabe der max. Geschwindigkeit | 64 |
| tboutpass | (?)!tboutpass x 100 | Filterzeitkonstante für Eingangswerte ist 100µs | 64 |
| tb | !tb o oder 1 | Trackball einschalten | 67 |

| **Endschalter (Hardware u. Software)** | | | |
|---|---|---|---|
| lim | (?) !lim x 0 100 | Verfahrbereichsgrenzen für die X-Achse 0+100mm ?lim x | 37 |
| limctr | (?) !limctr x 1 | Bereichsüberwachung von Achse X ist aktiv | 37 |
| nosetlimit | (?) !nosetlimit 1 1 1 1 | Für alle Achsen werden keine Verfahrbereichsgrenzen gesetzt | 38 |
| swpol | (?) !swpol 1 0 1 (z 1 0 1) | Polarität der Endschalter für alle Achsen od. nur Z zuweisen ?swpol x geht | 39 |
| swact | (?) !swact 1 0 1 (z 1 0 1) | Endschalter für alle Achsen od. nur Z Ein.-/Ausschalten ?swact x geht | 39 |
| readsw | ?readsw | lesen aller Endschalter-Zustände | 40 |

| **Digitale und analoge Ein.- und Ausgänge** | | | |
|---|---|---|---|
| digin | ?digin od. ?digin 8 | Lesen aller Eingänge od. lesen von Eingang 8 | 68 |
| digout | !digout 5 1/?digout | Ausgang 5 wird auf 1 gesetzt / Zustand aller Ausgänge lesen | 68 |
| anain | ?anain c 2 | lesen des aktuellen Zustandes von Analogkanal 2 | 69 |
| anaout | (?) !anaout c 1 0 | setzt analogkanal 1 auf 0 | 69 |

| **Takt-Vor/Rück Eingänge** | | | |
|---|---|---|---|
| tvrtoaxis | (?)!tvrtoaxis 1 2 0 0 | Takt-Vor/rück-Achsen-Zuordnung | 73 |
| tvr | (?) !tvr 1 1 | aktiviert Takt-V/R für X + Y | 73 |
| tvrf | (?) !Tvrf 1 | Faktor Takt Vor/Rück 1= 1Takt ist 1Motorincrement | 75 |
| tvrm | (?) !tvrm 1 | Takt-Vor/Rück Modus 0 bis 4 | 74 |

| Befehl | Beispiel | Bemerkung | Kap.4 Seite |
|---|---|---|---|
| **Geber-Einstellungen** | | | |
| twi | (?) !twi 10 10 10 | Das Zielfenster für alle Achsen=10µ ( bei Dimmension=1 ) | 84 |
| poswindowrange | (?)!poswindowrange | Alternative zu“ twi“ (zur Vereinheitlichung) | 85 |
| enctype | (?)!enctyp 4 4 4 4 | TTL Encoder für alle Achsen | 77 |
| enctoaxis | (?) !entoaxis | Zuordnung von Gebereingang und Achse | 76 |
| encdir | (?)!encdir 1 1 1 1 | Zählrichtung der Encoder gedreht | 77 |
| enc | (?) enc | Antwort: 1 0= Geber-X = aktiv Geber-Y = deaktiviert | 78 |
| encperiod | (?) !encperiod 0.1 | Teilungsperiode des X-Gebers = 0,1mm | 78 |
| encpolepairs | !encpolepairs x 500 | Gibt die Anzahl der Encodersignalperioden pro Motorumdrehung an. | 79 |
| encref | (?) !encref 0 | keine Referenzsignalauswertung | 79 |
| encrefpol | (?) !encrefopol | Polarität der Referenzmarken der QEP-Encoder | 80 |
| encpos | (?) !encpos 1 | Bei der Positionsabfrage werden die Geberwerte angezeigt | 80 |
| encerr | (?) !encerr 0 | Clear Geberfehlermeldung X; ( Rückmeldung= 0 od. e ) | 81 |
| posconkp | (?) !posconkp | P-Anteil des Lageregers | 82 |
| posconoutpass | (?) !posconoutpass | Filter Zeitkonstante zum Ausgangsfilter (Tiefpass) des Lagerreglers | 82 |
| posconenable | (?) !posconenable | Achsenfreigabe für Lageregler | 82 |
| poscon | (?) !poscon | Ein- und Ausschalten des Lagereglers | 83 |
| deviationsrange | (?) !deviationsrange | Schleppfehlerüberwachung / Bereich | 83 |
| deviationstime | (?) !deviationstime | Schleppfehlerüberwachung / Zeitfenster | 83 |
| deviationcheck | (?) !deviationcheck | Schleppfehlerüberwachung / Ein- oder Ausschalten | 84 |

| | | | |
|---|---|---|---|
| **Trigger-Ausgang** | | | |
| trig | (?) !trig 1 | Schaltet den Trigger ein | 86 |
| triga | (?) !triga X | Wählt die Achse aus, auf die getriggert werden soll (z.B. X) | 86 |
| trigm | (?) !trigm 1 | Setzt den Trigger-Mode auf 1 | 86 |
| trigs | (?) !trigs 4 | Stellt die Trigger-Signal-Länge auf 4µs | 88 |
| trigd | (?) !trigd 1 | Stellt die Trigger-Distance auf 1mm (bei Dim 2) | 88 |
| Trigcount | (!)?trigcount | -lese Zählerstand Trigger 1 | 89 |

| Befehl | Beispiel | Bemerkung | Kap.4 Seite |
|---|---|---|---|
| **Snapshot-Eingang** | | | |
| sns | (?) !sns 1 | Snapshot "EIN" | 89 |
| snsl | (?) !snsl 1 | Snapshot ist high-aktiv | 90 |
| snsm | (?) !snsm 1 | Auto-Snapshot | 91 |
| snsc | ?snsc | Liefert die Anzahl der ausgelösten SnapShots | 91 |
| snsp | (!) ?snsp | Liefert die gespeicherte Position | 92 |
| snsa | ?snsa 11 | Abfrage der Snapshot-Position 11 | 92 |
| snsf | (?)!snsf 10 | Dient als Eingangsfilter bei prellenden Schaltern (Wert 0-100) | 90 |

| Erklärungen | |
|---|---|
| ! | nur schreiben ( "!" kann auch weggelassen werden ) |
| (?) ! | schreiben und lesen |
| ? | nur lesen ( "?" kann auch weggelassen werden ) |

| Eingabemöglichkeiten | |
|---|---|
| Befehl Wert Wert Wert Wert | alle Achsen werden gesetzt oder gelesen |
| Befehl Wert Wert | nur X + Y werden gesetzt oder gelesen |
| Befehl Achse Wert | nur die ausgewälte Achse wird gesetzt oder gelesen |

**Fehlermeldungen LSTEP-PCI *express*, LSTEP *express* und LSTEP-PP**

| Fehlermeldung | Fehlerbeschreibung |
|---|---|
| 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 ausgelößt |
| 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 |
| 1064 | Wegstrecke zu groß |
| 1065 | Bremse und Spannungsversorgung für Endschalter nicht gleichzeitig möglich |
| 1066 | Keine Kommutierung nötig |

**Fehlermeldungen LSTEP-PCI *express*, LSTEP *express* und LSTEP-PP**
Achsenfehler

| Fehlermeldung | Fehlerbeschreibung |
|---|---|
| 1096 | Endschalter aktiv |
| 1097 | Referenzschalter 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 | Geschwindiogkeit 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 |

**Fehlermeldungen LSTEP-PCI *express*, LSTEP *express* und LSTEP-PP**
Fahrüberwchung

| Fehlermeldung | Fehlerbeschreibung |
|---|---|
| 1117 | Achse noch am fahren |
| 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 zu klein, Achse eventuell blockiert |

**Fehlermeldungen LSTEP-PCI *express*, LSTEP *express* und LSTEP-PP**
Achsmeldungen

| Fehlermeldung | Fehlerbeschreibung |
|---|---|
| 1192 | #TRACKING_WARNING |
| 1193 | Warnug Übertemeratur Endstufe |
| 1194 | Warnung: Motortemperatur zu hoch |
| 1195 | Treiberspannung unterschritten |

## 4.2 Informationen über die Firmware und Hardware

Mit dem Befehl „ver“ kann die Version der Firmware abgefragt werden. Im speziellen kann mit dem Befehl „det“ abgefragt werden welche Optionen in der Firmware freigeschaltet sind. Jede LStep ist durch eine interne Seriennummer eindeutig gekennzeichnet. Diese Seriennummer kann mit dem Befehl „readsn“ ausgelesen werden.

| **Versionsnummer auslesen** | |
|---|---|
| **Befehl:** | ?ver oder ver |
| **Parameter:** | keine |
| **Beschreibung:** | liefert die aktuelle Versionsnummer der Firmware zurück |
| **Rückmeldung:** | LS44.xx.xxx |
| **Fehlercode:** | -- |
| **Beispiel:** | ?ver |

| **Interne Versionsnummer auslesen** | |
|---|---|
| **Befehl:** | ?iver oder iver |
| **Parameter:** | keine |
| **Beschreibung:** | Intere Versionsnummer Auslesn |
| **Rückmeldung:** | Wochentag_Kalenderwoche_Jahr-fortlaufende Nummer |
| **Fehlercode:** | -- |
| **Beispiel:** | ?iver Rückmeldung z. B.: T04_35-02-0004 |

| **Versionsnummer detailliert auslesen** | | |
|---|---|---|
| **Befehl:** | | ?det oder det |
| **Parameter:** | | keine |
| **Beschreibung:** | | Liefert die detaillierte Versionsnummer der Firmware zurück- |
| **Rückmeldung:** | | Es wird ein Dezimalwert zurück geliefert, der in einen Hexadezimalwert gewandelt werden muss: |
| | | 0x0 - - - 1 ➔ 1Vss-Geber konfiguriert |
| | | 0x0 - - - 2 ➔ MR-Geber konfiguriert |
| | | 0x0 - - - 4 ➔ TTL-Geber konfiguriert |
| | | 0x0 - - 3 - ➔ Die zweite Zahl gibt die Achsenanzahl an (hier 3) |
| | | 0x0 - 1 - - ➔ Display konfiguriert |
| | | 0x0 - 2 - - ➔ Speedpoti konfiguriert |
| | | 0x0 - 4 - - ➔ Handrad konfiguriert |
| | | 0x0 - 8 - - ➔ Snapshot konfiguriert |
| | | 0x01 - - - ➔ TVRin konfiguriert |
| | | 0x02 - - - ➔ Triggerout konfiguriert |
| | | 0x08 - - - ➔ TVRout konfiguriert |
| | | 0x1 - - - - ➔ 16 digitale I/O konfiguriert |
| | | 0x2 - - - - ➔ 32 digitale I/O konfiguriert |
| | | 0x4 - - - - ➔ Trackball |
| | | Die Kombination dieser Informationen ergibt die aktuelle Konfiguration. |
| **Fehlercode:** | | -- |
| **Beispiel:** | | ?det = 81697 ➔ $13F21_H$ |
| **Erklärung 13F21** | **1** | 16 digitale I/O konfiguriert |
| | **3** | TVR und Triggerout konfiguriert |
| | **F** | Display; Speedpoti; Handrad und Snapshot konfiguriert |
| | **2** | 2 Achsen |
| | **1** | 1Vss Geber konfiguriert |

| Seriennummer lesen | |
|---|---|
| **Befehl:** | ?readsn |
| **Parameter:** | keine |
| **Beschreibung:** | ?readsn = Welche Seriennummer? |
| **Rückmeldung:** | 9-Zeichen |
| **Fehlercode:** | -- |
| **Beispiel:** | ?readsn |

## 4.3 Reset

Es gibt drei Möglichkeiten das Steuerungsprogramm zurückzusetzen:

- Den Hardware-Reset über den Netzschalter (bei Steuerungen ohne Anzeige).
- Den Hardware-Reset über den Reset-Taster (nur bei Steuerungen mit Anzeige).
- Den Hardware-Reset über den Dip-Schalter 1 bei der PCI-Karte
- Den Software Reset

| Software - Reset | |
|---|---|
| **Befehl:** | Reset |
| **Parameter:** | keine |
| **Beschreibung:** | Die Steuerung wird in den Startzustand versetzt |
| **Rückmeldung:** | keine |
| **Fehlercode:** | -- |
| **Beispiel:** | Reset |
| **Aktivierung:** | sofort |

| Poweramplifier | |
|---|---|
| **Befehl:** | !poweramplifier oder !pa |
| **Parameter:** | 0 oder 1 |
| **Beschreibung:** | 0 = Endstufe Aus<br>1 = Endstufe Ein |
| **Rückmeldung:** | -- |
| **Fehlercode:** | -- |
| **Beispiel:** | !pa 1 1 1 1 Alle Achsen Endstufe Ein |
| **Aktivierung:** | sofort |

## 4.4 Schnittstellenkonfiguration

| Baud - Rate | |
|---|---|
| **Befehl:** | !baud oder ?baud |
| **Parameter:** | 9600, 19200, 38400, 57600 |
| **Beschreibung:** | !baud 19200 ➔ Die Übertragungsrate der Schnittstelle wird auf 19200 baud eingestellt. |
| | ?baud ➔ liefert die aktuelle Übertragungsrate |
| **Rückmeldung:** | Aktuelle Übertragungsrate |
| **Fehlercode:** | -- |
| **Beispiel:** | ?baud |
| **Aktivierung:** | sofort |

| Parameter Save Funktion | |
|---|---|
| **Befehl:** | Save |
| **Parameter:** | -- |
| **Bemerkung:** | Save bedeutet:<br>Die aktuellen Parameter (Spindelsteigung, usw.) werden in das Flash programmiert und stehen bei einem Neustart sofort zur Verfügung. |
| **Beschreibung:** | save => Die aktuellen Parameter werden ins Flash programmiert. |
| **Rückmeldung:** | Anzeige im Display<br>Mit ?err kann der Erfolg kontrolliert werden.<br>D.h.:<br>Rückmeldung = 0 => Save OK<br>Rückmeldung ungleich 0 => Save nicht OK (siehe Controller-Manual ) |
| **Fehlercode:** | -- |
| **Beispiel:** | -- |

## 4.5 Status und Fehlermeldungen

| AutoStatus | |
|---|---|
| **Befehl:** | !autostatus oder ?autostatus |
| **Parameter:** | 0,1, 2, 3 oder 4 |
| **Beschreibung:** | 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 ➔ Bei „Positionerreicht“ wird nur ein Carriage Return zurückgegeben. |
| | 4 ➔ Liefert alle Schreibbefehle mit Parametern zurück. |
| **Rückmeldung:** | |
| **Fehlercode:** | -- |
| **Beispiel:** | !autostatus 1<br>?autostatus |
| **Aktivierung:** | sofort |

| Status | |
|---|---|
| **Befehl:** | ?status oder status |
| **Parameter:** | -- |
| **Beschreibung:** | Status liefert den aktuellen Zustand der Steuerung |
| **Rückmeldung:** | OK... oder ERR und Fehlermeldung |
| **Fehlercode:** | -- |
| **Beispiel:** | ?status |

| Systemstatus | |
|---|---|
| **Befehl:** | ?sysstat oder ?sysstatus |
| **Parameter:** | -- |
| **Beschreibung:** | Liefert den aktuellen Systemstatus der Achsen |
| **Rückmeldung:** | Nummer oder Textmeldung |
| **Fehlercode:** | -- |
| **Beispiel:** | ?sysstatus |

| Systemstatus Rückmeldungen | | |
|---|---|---|
| **sysstat** | **sysstatus** | **Bemerkung** |
| 0 | Just turned On | |
| 1 | not Configured | |
| 2 | Configured | |
| 3 | Ready for Komm | |
| 4 | Autokommutating | |
| 5 | Ready for Power | Endstufen ausgeschaltet |
| 6 | Positioncontrol | |
| 7 | Speedcontrol | |
| 8 | Manual Mode | |
| 9 | Calibrating | |
| 10 | Error-Power-Off | Treiberspannung unterbrochen |
| 11 | Error-Stop-Off | |
| 12 | Error-Stop-On | Regler aus nach Schleppfehler |
| 13 | System Error | |
| 14 | Undef | |
| 15 | Changing | |

| **StatusAxis** | |
|---|---|
| **Befehl:** | ?statusaxis oder statusaxis |
| **Parameter:** | -- |
| **Beschreibung:** | Statusaxis liefert den aktuellen Zustand der einzelnen Achsen. |
| **Rückmeldung:** | z.B.: @ - M – |
| | @ ➔ Achse steht und ist bereit |
| | M ➔ Achse ist in Bewegung (Motion) |
| | J ➔ Joystick-Betrieb |
| | C ➔ in Regelung |
| | S ➔ Endschalter betätigt |
| | A ➔ Rückmeldung nach dem Kalibrieren |
| | E ➔ Rückmeldung nach dem Kalibrieren wenn ein Fehler aufgetreten ist. (Endschalter nicht korrekt freigefahren) |
| | D ➔ Rückmeldung nach dem Tischhubmessen |
| | U ➔ Einrichtbetrieb (Setting **U**p) |
| | T ➔ Timeout |
| | - ➔ Achse ist nicht freigegeben |
| **Fehlercode:** | -- |
| **Beispiel:** | ?statusaxis |

| Error | |
|---|---|
| **Befehl:** | ?err oder err |
| **Parameter:** | -- |
| **Beschreibung:** | Error liefert die aktuelle Fehlernummer<br>(siehe Beschreibung der Fehlermeldungen) |
| **Rückmeldung:** | Dezimaler Wert |
| **Fehlercode:** | -- |
| **Beispiel:** | ?err |

| Error_Nr | Beschreibung der Fehlermeldungen |
|---|---|
| 0 | Kein Fehler |
| 1 | Keine gültige Achsenbezeichnung |
| 4 | Kein gültiger Befehl |
| 5 | Außerhalb des gültigen Zahlenbereichs |
| 6 | Falsche Anzahl der Parameter |
| 7 | Kein ! oder ? |
| 8 | Kein TVR möglich, da Achse aktiv |
| 9 | Kein Ein- oder Ausschalten der Achsen, da TVR aktiv |
| 10 | Funktion nicht konfiguriert |
| 11 | Kein Move – Befehl möglich, da Joystick – Hand |
| 12 | Endschalter betätigt |
| 27 | NOT-STOP |

## 4.6 Einstellungen

Mit den folgend beschriebenen Befehlen kann die Steuerung an die eingesetzte Mechanik und die gewünschten Anforderungen angepasst werden.

| **ConfigMaxAxis** | |
|---|---|
| **Befehl:** | (?)!configmaxaxis |
| ***Funktionsbeschreibung:*** | *Mit diesem Befehl wird die Anzahl der Achsen einer Steuerung konfiguriert. z.B.: Hat man eine 4-Achs-Steuerung aber nur eine 3-Achs-Anwendung, so kann man die Steuerung als 3-Achs-Steuerung konfigurieren.* |
| **Parameter:** | 1 bis 4 |
| **Beschreibung:** | ?configmaxaxis => liefert die Anzahl der konfigurierten Achsen<br>!configmaxaxis 3 => die Steuerung wird als 3-Achs Steuerung konfiguriert |
| **Rückmeldung:** | 1 bis 4 |
| **Fehlercode:** | -- |
| **Beispiel:** | !configmaxaxis 4 |
| **Aktivierung** | !validconfig bzw. !validpar |

| **Dimension** | |
|---|---|
| **Befehl:** | !dim oder ?dim |
| **Parameter:** | X, y, z und a<br>0, 1, 2, 3 oder 4<br>Die Einheiten von Längenangaben bei Ein- und Ausgabe sind: |
| | 0 ➔ Microsteps |
| | 1 ➔ µm |
| | 2 ➔ mm |
| | 3 ➔ 360° |
| | 4 ➔ Anzahl der Umdrehungen |
| | 5 ➔ inch |
| **Beschreibung:** | |
| | !dim 4 1 ➔ Die Dimensionen für x- und y-Achse sind „Anzahl der Umdrehungen„ und „µm„. |
| | !dim z 2 ➔ Die Dimensionen für die z-Achse ist „mm„. |
| | ?dim ➔ Es werden alle Dimensionen angezeigt. |
| | ?dim a ➔ Es wird die Dimension der a-Achse angezeigt. |
| **Rückmeldung:** | Aktuelle Einstellung |
| **Fehlercode:** | |
| **Beispiel:** | !dim 1 1 1 1 (Alle Werte in µm)<br>?dim |
| **Aktivierung:** | sofort |

| | |
|---|---|
| **Hinweis:** | Für die Dimension 3 (Grad) und 4 (Umdrehungen) sollte die Spindelsteigung auf 1 mm eingestellt werden. |

| **Spindelsteigung** | |
|---|---|
| **Befehl:** | !pitch oder ?pitch |
| **Parameter:** | X, y, z und a<br>0.0001 – 68 |
| **Beschreibung:** | !pitch 4.0 1.0 ➔ Spindelsteigungen x = 4mm und y = 1mm werden programmiert.<br>!pitch z 1.0 ➔ Spindelsteigung z = 1mm wird programmiert.<br>?pitch ➔ Es werden alle Spindelsteigungen angezeigt.<br>?pitch a ➔ Es wird die Spindelsteigung der a-Achse angezeigt. |
| **Rückmeldung:** | Aktuelle Spindelsteigung |
| **Fehlercode:** | -- |
| **Beispiel:** | !pitch 10 (Spindelsteigung x = 10mm)<br>?pitch |
| **Aktivierung:** | !validconfig bzw. !validpar |

| **Getriebeeinstellunge - Geardenominator** | |
|---|---|
| **Befehl:** | !geardenominator oder ?geardenominator |
| *Funktionsbeschreibung:* | *Der Befehl kommt bei der Verwendung eines Getriebes zum Einsatz. Mittels Befehl geardenominator wird der Wert Y in einem Bruch, bzw. in einem Getriebefaktor $^{X}/_{Y}$ eingestellt.* |
| **Parameter:** | X, y, z und a<br>1, 2, 3, . . . . . ∞ (Natürliche Zahlen) |
| **Beschreibung:** | !geardenominator 4 1 ➔ Getriebe-Nominator X $^{X}/_{4}$ *und* $^{X}/_{1}$ *bei Y* werden programmiert.<br>!geardenominator z 8 ➔ Getriebe-Übersetzungen $^{X}/_{8}$ bei z wird programmiert.<br>?geardenominator r ➔ Es werden alle Getriebe-Übersetzungen angezeigt.<br>?geardenominator a ➔ Es wird der aktuelle Nominator der a-Achse angezeigt. |
| **Rückmeldung:** | Der Befehl ?geardenominator gibt den aktuellen Nominator (Nenner) der Getriebe-Übersetzungen zurück. |
| **Fehlercode:** | -- |
| **Beispiel:** | !geardenominator 2 (Getriebe-Übersetzung $^{x}/_{2}$ bei x)<br>?geardenominator |
| **Aktivierung:** | !validconfig bzw. !validpar |

| **Getriebeeinstellungen - Gearnumerator** | | |
|---|---|---|
| **Befehl:** | !gearnumerator oder ?gearnumerator | |
| *Funktionsbeschreibung:* | *Der Befehl kommt bei der Verwendung eines Getriebes zum Einsatz. Mittels Befehl gearnumerator wird der Wert X in einem Bruch, bzw. in einem Getriebefaktor $^{X}/_{Y}$ eingestellt.* | |
| **Parameter:** | X, y, z und a<br>1, 2, 3, . . . . . ∞ (Natürliche Zahlen) | |
| **Beschreibung:** | !geardenumeratorator 4 1 | ➔ Getriebe-Zähler X $^{4}/_{y}$ *und* $^{1}/_{y}$ *bei Y* werden programmiert. |
| | !gearnumeratorator z 8 | ➔ Getriebe-Übersetzungen $^{8}/_{y}$ bei z wird programmiert. |
| | ?gearnumeratorator r | ➔ Es werden alle Getriebe-Übersetzungen angezeigt. |
| | ?gearnumeratorator a | ➔ Es wird der aktuelle Zähler der a-Achse angezeigt. |
| **Rückmeldung:** | Der Befehl ?gearnumerator gibt den aktuellen Numeratorator (Zähler) der Getriebe-Übersetzungen zurück. | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !gearnumeratorator 2 (Getriebe-Übersetzung $^{2}/_{Y}$ bei x)<br>?gearnumeratorator | |
| **Aktivierung:** | !validconfig bzw. !validpar | |

| **Beschleunigung** | | |
|---|---|---|
| **Befehl:** | !accel oder ?accel | |
| *Funktionsbeschreibung:* | *Die Beschleunigung der einzelnen Achsen lassen sich durch den Befehl konfigurieren.* | |
| **Parameter:** | X, y, z und a<br>0.01 – 20.00 [m/s²] | |
| **Beschreibung:** | !accel 1.00 1.50 | ➔ Bei Achse x und y werden die Beschleunigungen (x=1.00, y=1.50 [m/s²]) eingestellt, die anderen Achsen bleiben unverändert. |
| | !accel x 1 | ➔ Die Beschleunigung für Achse x wird auf 1.00 [m/s²] eingestellt. |
| | ?accel | ➔ Alle eingestellten Beschleunigungen werden angezeigt. |
| | ?accel z | ➔ Die eingestellte Beschleunigung der z-Achse wird angezeigt. |
| **Rückmeldung:** | Eingestellte Beschleunigung | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !accel 1.00 (Setze Beschleunigungen bei x-Achse auf 1 m/s²)<br>?accel | |
| **Aktivierung:** | sofort | |

| **Ruckeinstellung während Beschleunigung [j]** | |
|---|---|
| **Befehl:** | !acceljerk oder ?acceljerk |
| *Funktionsbeschreibung:* | *Bei schwingungskritischer Maschinenmechanik sind diese Profile zu verwenden. Die Rampenzeiten für den Beschleunigungsaufbau und Abbau sollten nur so groß wie erforderlich und so klein wie möglich eingestellt werden, da diese Parameter erheblichen Einfluss auf die Positionierzeiten haben Dieser Befehl ermöglicht die Konfiguration des Rucks jeder einzelnen Achse während der Beschleuniguung.* |
| **Parameter:** | X, y, z und a<br>1, 2, 3, . . . . . ∞ [m/s³] (Natürliche Zahlen) |
| **Beschreibung:** | !acceljerk 4 1 ➔ Ruck 4 m/s³ bei X und 1 m/s³ wird bei Y programmiert |
| | !acceljerk z 10 ➔ Ruck 10 m/s³ wird bei z wird programmiert. |
| | ?acceljerk ➔ Es werden alle konfigurierten Wert angezeigt. |
| | ?acceljerk a ➔ Es wird die Ruckeinstellung der a-Achse angezeigt. |
| **Rückmeldung:** | Ruck Beschleunigung |
| **Fehlercode:** | -- |
| **Beispiel:** | !acceljerk 10 (Ein Ruck von 10m/s³ wird bei x eingestellt)<br>?acceljerk |
| **Aktivierung:** | sofort |

| **Verzögerung** | | |
|---|---|---|
| **Befehl:** | !decel oder ?decel | |
| *Funktionsbeschreibung:* | *Die Verzögerung der einzelnen Achsen lassen sich durch den Befehl konfigurieren. Damit lässt sich der Wert einstellen, mit dem die Achsen bremsen sollen.* | |
| **Parameter:** | X, y, z und a<br>0.01 – 20.00 [m/s²] | |
| **Beschreibung:** | !decel 1.00 1.50 | ➔ Bei Achse x und y werden die Verzögerungen (x=1.00, y=1.50 [m/s²]) eingestellt, die anderen Achsen bleiben unverändert. |
| | !decel x 1 | ➔ Die Verzögerung für Achse x wird auf 1.00 [m/s²] eingestellt. |
| | ?decel | ➔ Alle eingestellten Verzögerungen werden angezeigt. |
| | ?decel z | ➔ Die eingestellte Verzögerung der z-Achse wird angezeigt. |
| **Rückmeldung:** | Verzögerung | |
| **Fehlercode** | -- | |
| **Beispiel:** | !decel 10 (Verzögerung in der Achse bei 10m/s²)<br>?decel | |
| **Aktivierung:** | sofort | |

| **Ruckeinstellung während Verzögerung** | | |
|---|---|---|
| **Befehl:** | !deceljerk oder ? deceljerk | |
| *Funktionsbeschreibung:* | *Bei schwingungskritischer Maschinenmechanik sind diese Profile zu verwenden. Die Rampenzeiten für den Beschleunigungsaufbau und Abbau sollten nur so groß wie erforderlich und so klein wie möglich eingestellt werden, da diese Parameter erheblichen Einfluss auf die Positionierzeiten haben Dieser Befehl ermöglicht die Konfiguration des Rucks jeder einzelnen Achse während der Verzögerung.* | |
| **Parameter:** | X, y, z und a<br>1, 2, 3, . . . . . ∞ [m/s³] (Natürliche Zahlen) | |
| **Beschreibung:** | !deceljerk 4 1 | ➔ Ruck 4 m/s³ bei X und 1 m/s³ wird bei Y programmiert |
| | !deceljerk z 10 | ➔ Ruck 10 m/s³ wird bei z wird programmiert. |
| | ?deceljerk | ➔ Es werden alle konfigurierten Wert angezeigt. |
| | ?deceljerk a | ➔ Es wird die Ruckeinstellung der a-Achse angezeigt. |
| **Rückmeldung:** | Ruckverzögerung | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !deceljerk 10 (Ein Ruck von 10m/s³ wird bei x eingestellt)<br>?deceljerk | |
| **Aktivierung:** | sofort | |

| **Geschwindigkeit** | | |
|---|---|---|
| **Befehl:** | !vel oder ?vel | |
| *Funktionsbeschreibung:* | *Die Verfahrgeschwindigkeit der einzelnen Achsen werden mit diesem Befehl eingestellt.* | |
| **Parameter:** | X, y, z und a<br>0 – maximale Geschwindigkeit | |
| **Beschreibung:** | !vel 1.0 15 | ➔ Bei Achse x und y werden die Geschwindigkeitswerte (x=1.0, y=15 [U/s]) beschrieben, die anderen Achsen bleiben unverändert. |
| | !vel z 0.1 | ➔ Bei der z-Achse wird die Geschwindigkeit auf 0.1 [U/s] eingestellt. |
| | ?vel | ➔ Alle eingestellten Geschwindigkeiten werden angezeigt. |
| | ?vel x | ➔ Anzeigen der eingestellten Geschwindigkeit von Achse x. |
| **Rückmeldung:** | Eingestellte Geschwindigkeit | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !vel 10 (Die x-Achse wird mit maximal 10 U/s betrieben)<br>?vel | |
| **Aktivierung:** | sofort | |

Die Drehzahl der Motoren ist in Stufen (St) von 0,0001 U/sec. bis 70 U/sec einstellbar. Die oberen Drehzahlbereiche lassen sich nur bei optimaler Abstimmung der Motoren und Mechanik an die LSTEPexpress erreichen.
Die kleinste Geschwindigkeit ist möglich mit der Dimension Mikroschritte und der max. Schrittauflösung von 32768 MI/Polpaar. Für die unterschiedlichen Dimensionen gelten folgende kleinste Geschwindigkeiten: 1MI/s; 0,1µm/s; 0,0001mm/s; 0,001°/s; 0,0001U/s.

| **MaxCurrent (max. möglicher Motorstrom)** | |
|---|---|
| **Befehl:** | ?maxcur |
| *Funktionsbeschreibung:* | *Durch diesen Befehl lässt sich der maximale Motorstrom per Motorphase abfragen.* |
| **Parameter:** | X, y, z oder a |
| **Beschreibung:** | ?maxcur y => liefert den maximal möglichen Motorstrom der Y – Achse<br>?maxcur => liefert den maximal möglichen Motorstrom aller Achsen |
| **Rückmeldung:** | Motorstrom in Ampere |
| **Fehlercode:** | -- |
| **Beispiel:** | ?maxcur |

| **Motorstrom für jede Achse einstellen** | | |
|---|---|---|
| **Befehl:** | !motorcurrent oder ?motorcurrent | |
| *Funktionsbeschreibung:* | *Dieser Befehl ermöglicht die achsenunabhängige Einstellung des jeweiligen Motorstroms.* | |
| **Parameter:** | X, y, z und a<br>0.5 – 5 (vierte Achse maximal 10A) | |
| **Beschreibung:** | !motorcurrent 1 2 2.5 | ➔ Setzt folgende Werte in der Steuerung: X=1A, Y=2A, Z=2,5A |
| | !motorcurrent z 5 | ➔ Die Z-Achse hat einen maximalen Motorstrom von 5A. |
| | ?motorcurrent | ➔ Es werden alle Einstellungen angezeigt. |
| | ?motorcurrent a | ➔ Es wird der eingestellte Wert der a-Achse angezeigt. |
| **Rückmeldung:** | Wert gibt den maximalen Motorstrom an | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !motorcurrent1 2 2.5 (Motorstromeinstellung: X=1A, Y=2A, Z=2,5A) | |
| **Aktivierung:** | !validconfig bzw. !validpar | |

| **Stromabsenkung** | |
|---|---|
| **Befehl:** | !reduction oder ?reduction |
| **Parameter:** | X, y, z und a<br>0 – 100 % |
| **Beschreibung:** | Im Ruhezustand wird der Motornennrstrom auf das parametrierte Verhältnis reduziert. |
| | !reduction 10 70 ➔ x-Achse = 0.1*Nennstrom und y-Achse = 0.7*Nennstrom |
| | !reduction z 50 ➔ z-Achse = 0.5*Nennstrom |
| | ?reduction ➔ Anzeige der eingestellten Stromabsenkungen aller Achsen |
| | ?reduction x ➔ Anzeige der eingestellten Stromabsenkung der Achse x. |
| **Rückmeldung:** | Eingestellte Stromabsenkung |
| **Fehlercode:** | -- |
| **Beispiel:** | !reduction 30 50 (x- und y-Achse werden reduziert )<br>?reduction |
| **Aktivierung:** | sofort |

| **Verzögerung Stromabsenkung (Delay Reduction)** | |
|---|---|
| **Befehl:** | !curdelay oder ?curdelay |
| **Parameter:** | X, y, z und a<br>0 – 10000 (ms) |
| **Beschreibung:** | Nach dem Verfahren eines Vektors bleibt der Motornennstrom für die in curdelay eingestellte Zeit erhalten. Danach wird er auf den in der Stromabsenkung spezifizierten Wert abgesenkt.<br>!curdelay 100 300 = x-Achse = 100 ms Verzögerung und y-Achse = 300 ms Verzögerung<br>!curdelay z 450 = z-Achse = 450 ms Verzögerung<br>?curdelay = Anzeige der eingestellten Stromabsenkungs-verzögerungen aller Achsen<br>?curdelay x = Anzeige der eingestellten Stromabsenkungs-verzögerung von Achse x |
| **Rückmeldung:** | Eingestellte Verzögerung der Stromabsenkung |
| **Fehlercode:** | -- |
| **Beispiel:** | !curdelay 100 300 (x- und y-Achse werden verzögert )<br>?curdelay |

| | |
|---|---|
| **Aktivierung:** | sofort |

| **Achsenfreigabe** | |
|---|---|
| **Befehl:** | !axis oder ?axis |
| **Parameter:** | x, y, z und a<br>0 und 1 |
| **Beschreibung:** | !axis 1 0 1 0 ➔ Achse x und z sind freigegeben, Achse y und a sind nicht freigegeben. |
| | !axis y 1 ➔ Achse y freigeben |
| | ?axis ➔ Zustand aller Achsen darstellen |
| | ?axis a ➔ Zustand Achse a darstellen |
| **Rückmeldung:** | Aktueller Betriebszustand |
| **Fehlercode:** | -- |
| **Beispiel:** | !axis 1 1 1 1 (alle Achsen freigeben)<br>?axis x (lese Zustand der x-Achse) |
| **Aktivierung:** | sofort |

| **Verfahrrichtung der Achsen einstellen** | |
|---|---|
| **Befehl:** | !?axisdir |
| *Funktionsbeschreibung:* | *Mit diesem Befehl lassen sich zeitgleich Drehrichtung und Endschalter tauschen.* |
| **Parameter:** | X, y, z und a<br>0 oder 1 |
| **Bemerkung:** | Mit axisdir können die Motor - Drehrichtungen gedreht werden, die dazugehörigen Endschalter werden mit gedreht. |
| **Beschreibung:** | !axis 0 1 0 1 => Bei den Achsen y und a werden die Drehrichtungen gedreht.<br>?axisdir x = Anzeige, ob bei der x-Achse die Drehrichtung aktiviert ist. |
| **Rückmeldung:** | 0 = Kein Drehrichtungswechsel<br>1 = Drehrichtungswechsel |
| **Fehlercode:** | -- |
| **Beispiel:** | !axisdir 0 0 0 0 (Aufheben aller Drehrichtungswechsel ) |
| **Aktivierung:** | !validconfig bzw. !validpar |

| **Motordrehrichtung einstellen** | | |
|---|---|---|
| **Befehl:** | !motorfielddir oder ?motorfielddir | |
| *Funktionsbeschreibung:* | *Die Drehrichtung des Motors lässt sich mit diesem Befehl beeinflussen. Die einzelnen Achsen können somit unterschiedliche Motordrehrichtungen haben.* | |
| **Parameter:** | X, y, z und a<br>Werte 0 und 1 | |
| **Beschreibung:** | !motorfielddir 0 1 | ➔ Drehrichtung des Motor ist in Y getauscht, X ist normal |
| | !motorfielddir Z 1 | ➔ Drehrichtung des Motor ist in Z getauscht |
| | ?motorfielddir | ➔ Es werden alle eingestellten Drehrichtungen angezeigt. |
| | ?motorfielddir x | ➔ Es wird die Drehrichtung der x-Achse angezeigt. |
| **Rückmeldung:** | Aktuelle Motordrehrichtung der einzelnen Achsen<br>0 = Kein Drehrichtungswechsel<br>1 = Drehrichtungswechsel | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !motorfielddir 0 1 (setzt die entsprechenden Werte und führt eine Drehrichtungsumkehr in Y durch. X bleibt unverändert) | |
| **Aktivierung:** | !validconfig bzw. !validpar | |

| **Endschalter tauschen** | |
|---|---|
| **Befehl:** | !swchange oder ? swchange |
| *Funktionsbeschreibung:* | *Mit diesem Befehl lassen sich softwareseitig die Endschalter tauschen. Ein mechanischer Aufwand wird durch den Befehl überflüssig.* |
| **Parameter:** | X, y, z und a<br>Werte 0 und 1 |
| **Beschreibung:** | !swchange 0 1 ➔ Endschalter ist in Y getauscht, X ist normal |
| | !swchange Z 1 ➔ Endschalter ist in Z getauscht |
| | ?swchange ➔ Es werden alle Einstellungen der Endschalter angezeigt. |
| | ?swchange x ➔ Es wird die Endschaltereinstellung der x-Achse angezeigt. |
| **Rückmeldung:** | Aktuelle Einstellung der Endschalter der einzelnen Achsen<br>0 = Kein Endschalterwechsel<br>1 = Wechsel der Endschalter |
| **Fehlercode:** | -- |
| **Beispiel:** | !swchange 0 1 (setzt die entsprechenden Werte und führt eine Tausch in Y durch. X bleibt unverändert) |
| **Aktivierung:** | !validconfig bzw. !validpar |

| **Anzahl Motorpolpaare einstellen** | |
|---|---|
| **Befehl:** | !?motorpolepairs |
| *Funktionsbeschreibung:* | *Mit diesem Befehl lassen sich Schrittmotore mit unterschiedlichen Schrittwinkeln an die Steuerung anpassen.* |
| **Parameter:** | x, y, z und a<br>Polpaarzahl |
| **Bemerkung:** | Ein 200 schrittiger Motor mit einem Schrittwinkel von 1.8° hat 50 Polpaare. |
| **Beschreibung:** | !motorpolepairs x 50<br>(Der X-Motor hat 50 Polpaare = 1,8°/Vollschritt)<br>?motorpolepairs x = Anzeige der eingestellten Polparzahl für den X-Motor |
| **Rückmeldung:** | Polparzahl |
| **Fehlercode:** | -- |
| **Beispiel:** | !motorpolepairs 50 50 100 100 |
| **Aktivierung:** | !validconfig bzw. !validpar |

| Anzahl der Mikroschritte pro Motorpolpaar | |
|---|---|
| **Befehl:** | !?motorpolepairres |
| *Funktionsbeschreibung:* | *Mit diesem Befehl lässt sich die Schrittauflösung einstellen.* |
| **Parameter:** | x, y, z und a<br>Schrittauflösung: 500; 1000; 2000; 32768 |
| **Bemerkung:** | Eine Schrittauflösung von 1000 ergibt bei einem 200schrittigen Motor eine Auflösung von 50.000 Mikroschritten / Umdrehung |
| **Beschreibung:** | !motorpolepairres x 32768<br>(Der X-Motor hat 50 eine Auflösung von 1.638.400 Mikroschritten bei einem 1,8° Motor)<br>?motorpolepairres x = Anzeige der eingestellten Schrittauflösung |
| **Rückmeldung:** | Eingestellte Schrittauflösung |
| **Fehlercode:** | -- |
| **Beispiel:** | !motorpolepairres 1000 1000 1000 1000 |
| **Aktivierung:** | !validconfig bzw. !validpar |

| Maximale Motordrehzahl | |
|---|---|
| **Befehl:** | !?motormaxvel |
| *Funktionsbeschreibung:* | *Mit diesem Befehl lässt sich die max. Motordrehzahl begrenzen.* |
| **Parameter:** | x, y, z und a<br>Drehzahl 1/min. |
| **Bemerkung:** | Eine Schrittauflösung von 1000 ergibt bei einem 200schrittigen Motor eine Auflösung von 50.000 Mikroschritten / Umdrehung |
| **Beschreibung:** | !motormaxvel x 1000<br>(Für den X-Motor hat 50 eine max. Geschwindigkeit von 1000 Umdr./min)<br>?motormaxvel x = Anzeige der eingestellten Geschwindigkeit |
| **Rückmeldung:** | Eingestellte Geschwindigkeit |
| **Fehlercode:** | -- |
| **Beispiel:** | !motormaxvel 1000 1000 1000 1000 |
| **Aktivierung:** | !validconfig bzw. !validpar |

| **Motortyp** | |
|---|---|
| **Befehl:** | !?motortype |
| *Funktionsbeschreibung:* | *Mit diesem Befehl lässt sich der Motortyp einstellen.* |
| **Parameter:** | x, y, z und a<br>Motortyp 0 = rotativer 2-Phasen Schrittmotor<br>Motortyp 1 = linearer 2-Phasen Schrittmotor<br>Motortyp 4 = rotativer 2-Phasen Servomotor<br>Motortyp 5 = linearer 2-Phasen Servomotor |
| **Bemerkung:** | Eine Schrittauflösung von 1000 ergibt bei einem 200schrittigen Motor eine Auflösung von 50.000 Mikroschritten / Umdrehung |
| **Beschreibung:** | !motortype x 0 ( es ist ein 2-Phasen rotativer Schrittmotor angeschlossen )<br>?motortype x Anzeige des eingestellten Motortyps |
| **Rückmeldung:** | Eingestellter Motortyp |
| **Fehlercode:** | -- |
| **Beispiel:** | !motortype 0 0 0 0 (alle 4 Achsen 2-Phasen rotativer Schrittmotor) |
| **Aktivierung:** | !validconfig bzw. !validpar |

| **Motorbrake** | |
|---|---|
| **Befehl:** | !?motorbrake |
| *Funktionsbeschreibung:* | *Mit diesem Befehl lässt sich für jeden Achse ein Ausgang schalten, den man für eine Motorbremse oder andere Anwendungen wie z.B. eine Türverriegelung benutzen kann.* |
| **Parameter:** | x, y, z und a<br>Der Ausgang schaltet:<br>Motorbrake 0 = nach GND<br>Motorbrake 1 = in Abhängigkeit der Endstufe (pa 0 oder pa 1)<br>Motorbrake 2 = nach +12V (24V) |
| **Bemerkung:** | Eine Schrittauflösung von 1000 ergibt bei einem 200schrittigen Motor eine Auflösung von 50.000 Mikroschritten / Umdrehung |
| **Beschreibung:** | !motorbrake x 0 ( der Ausgang ist nach GND geschaltet)<br>?motorbrake x Anzeige der Einstellung |
| **Rückmeldung:** | Eingestellter Modus |
| **Fehlercode:** | -- |
| **Beispiel:** | !motorbrake 1 1 1 1 (bei allen 4 Achsen wird der Ausgang in abhängigkeit der Endstufe geschaltet) |
| **Aktivierung:** | !validconfig bzw. !validpar |

| Motorbrakeswitchondelay | |
|---|---|
| **Befehl:** | !?motorbrakeswitchondelay |
| *Funktionsbeschreibung:* | *Mit diesem Befehl lässt sich im Modus „motorbrake 1“ ein Delay einstellen, mit dem der Ausgang oder die Endstufe verzögert eingeschaltet wird.* |
| **Parameter:** | x, y, z und a<br>Wertebereich von -5000 bis 5000 ms |
| **Bemerkung:** | Bei negativen Werten wird die Endstufe verzögert eingeschaltet. Bei positiven Werten wird der Ausgang verzögert geschaltet |
| **Beschreibung:** | !motorbrakeswitchondelay z 1000 (z.B.: Beim Einschalten der Endstufe für die Z-Achse wird die Bremse erst nach 1 Sekunde geöffnet)<br>?motorbrakeswitchondelay z (Abfrage des Delays für Z) |
| **Rückmeldung:** | Eingestelltes Delay |
| **Fehlercode:** | -- |
| **Beispiel:** | !motorbrakeswitchondelay x -1000 (z.B.: Beim Einschalten der Endstufe für die X-Achse, wird 1 Sekunde vorher die Türverriegelung aktiviert) |
| **Aktivierung:** | !validconfig bzw. !validpar |

| Motorbrakeswitchoffdelay | |
|---|---|
| **Befehl:** | !?motorbrakeswitchoffdelay |
| *Funktionsbeschreibung:* | *Mit diesem Befehl lässt sich im Modus „motorbrake 1“ ein Delay einstellen, mit dem der Ausgang oder die Endstufe verzögert ausgeschaltet wird.* |
| **Parameter:** | x, y, z und a<br>Wertebereich von -5000 bis 5000 ms |
| **Bemerkung:** | Bei negativen Werten wird die Endstufe verzögert ausgeschaltet. Bei positiven Werten wird der Ausgang verzögert geschaltet |
| **Beschreibung:** | !motorbrakeswitchoffdelay z -1000 (z.B.: Beim Ausschalten der Endstufe für die Z-Achse wird die Bremse 1 Sekunde vorher aktiviert)<br>?motorbrakeswitchoffdelay z (Abfrage des Delays für Z) |
| **Rückmeldung:** | Eingestelltes Delay |
| **Fehlercode:** | -- |
| **Beispiel:** | !motorbrakeswitchoffdelay x 1000 (z.B.: Beim Ausschalten der Endstufe für die X-Achse, wird 1 Sekunde nachher die Türverriegelung deaktiviert) |
| **Aktivierung:** | !validconfig bzw. !validpar |

| Fehlerbehebung quittieren | |
|---|---|
| **Befehl:** | quit |
| **Parameter:** | keine |
| **Beschreibung:** | Nach einem aufgetretenen Fehler und dessen Behebung, muss dies mit „!quit“ bestätigt werden |
| **Rückmeldung:** | keine |
| **Fehlercode:** | -- |
| **Beispiel:** | !quit |

| Einstellungen übernehmen | |
|---|---|
| **Befehl:** | validconfig |
| **Parameter:** | keine |
| **Beschreibung:** | Nach einer Änderung von einander abhängigen Grund-einstellungen wie z.B. die Spindelsteigung und Getriebefaktor-zähler und Getriebefaktornenner, werden diese mit dem Befehl validconfig übernommen. |
| **Rückmeldung:** | keine |
| **Fehlercode:** | -- |
| **Beispiel:** | !validconfig |

| Aktivierung: der Regler Parameter | |
|---|---|
| **Befehl:** | validpar |
| **Parameter:** | keine |
| **Beschreibung:** | Nach einer Änderung von Regler Parameter wie z.B. „posconkp“, werden diese mit dem Befehl validpar aktiviert. |
| **Rückmeldung:** | keine |
| **Fehlercode:** | -- |
| **Beispiel:** | !validpar |

| **Limit** | |
|---|---|
| **Befehl:** | !lim oder ?lim |
| *Funktionsbeschreibung:* | *Mit diesem Befehl lassen sich softwareseitig die Verfahrgrenzen der einzelnen Achsen einstellen. Somit lässt sich der Verfahrbereich entsprechend den Anforderungen einstellen.* |
| **Parameter:** | x, y, z oder a<br>+- maximale Verfahrbereich |
| **Bemerkung:** | Die Werte müssen paarweise für jeweils eine Achse vorgegeben werden mit Achskennung.<br>Die Ein- und Ausgabewerte sind abhängig von der Dimension. |
| **Beschreibung:** | !lim x -1000 1000 ➔ Achse x werden Verfahrbereichsgrenzen zugewiesen. |
| | ?lim z ➔ Verfahrbereichsgrenzen der Z- Achsen lesen |
| **Rückmeldung:** | Aktuelle Verfahrbereiche |
| **Fehlercode:** | -- |
| **Beispiel:** | !lim y 0 10<br>?lim x |
| **Aktivierung:** | sofort |

| **Bereichsüberwachung** | |
|---|---|
| **Befehl:** | !limctr oder ?limctr |
| **Parameter:** | x, y, z oder a<br>0 oder 1 |
| **Beschreibung:** | !limctr 1 1 1 ➔ Bereichsüberwachung von x-, y- und z-Achse aktiv. |
| | !limctr z 1 ➔ Bereichsüberwachung von Achse z aktiv. |
| | ?limctr a ➔ Bereichsüberwachung Achse a lesen |
| | ?limctr Anzeige des Zustands der einzelnen Bereichsüberwachungen. |
| **Rückmeldung:** | 0 = Bereichsüberwachung nicht aktiv<br>1 = Bereichsüberwachung aktiv |
| **Fehlercode:** | -- |
| **Beispiel:** | ! limctr y 0 (Bereichsüberwachung der Achse y deaktivieren<br>? limctr |
| **Aktivierung:** | sofort |

| **Statuslimit** | |
|---|---|
| **Befehl:** | ?statuslimit oder statuslimit |
| **Parameter:** | -- |
| **Beschreibung:** | Statuslimit liefert den aktuellen Zustand der Software-Grenzen jeder einzelnen Achse |
| **Rückmeldung:** | A = Achse wurde kalibriert |
| | D = Tischhub wurde gemessen |
| | L = Software – Limit wurde gesetzt |
| | - = Software – Grenze wurde nicht verändert |
| | Die Reihenfolge der Rückmeldung ist zum Beispiel:<br>AA-A--DD-LL-L--L |
| | X,y und a = Kalibriert |
| | Z und a = Tischhub gemessen<br>Y und z = min. Softwarelimit gesetzt |
| | X und a = max. Softwarelimit gesetzt |
| **Fehlercode:** | -- |
| **Beispiel:** | ?statuslimit |

| **NoSetLimit** | |
|---|---|
| **Befehl:** | !?nosetlimit |
| **Parameter:** | x, y, z oder a<br>0 oder 1 |
| **Bemerkung:** | Beim Kalibrieren und Tischhubmessen werden normalerweise die internen Software - Limits gesetzt, dass kann hiermit verhindert werden. |
| **Beschreibung:** | nosetlimit 1 1 1 => Bei den Achsen x, y und z werden keine Verfahrbereichsgrenzen gesetzt.<br>!nosetlimit y 1 => Bei der Achse y wird keine Verfahrbereichs-grenze gesetzt.<br>?nosetlimit = Einstellung aller Achsen lesen<br>?nosetlimit a = Einstellung der Achse a lesen |
| **Rückmeldung:** | 0 = Software – Limits werden gesetzt (calib/rm) |
| **Fehlercode:** | -- |
| **Beispiel:** | ?nosetlimit |
| **Aktivierung:** | sofort |

| **Endschalterpolarität** | |
|---|---|
| **Befehl:** | !swpol oder ?swpol |
| **Parameter:** | x, y, z oder a<br>**0**  oder **1**  |
| **Beschreibung:** | !swpol x 1 0 1 ➔ Polarität der Endschalter der X-Achse zuweisen. (Reihenfolge: E0 REF EE). |
| | ?swpol z ➔ Polarität der Endschalter von Achse z auslesen. |
| **Rückmeldung:** | Polarität der Endschalter |
| **Fehlercode:** | -- |
| **Beispiel:** | !swpol y 1 1 1 (Alle Schalter der Achse y reagieren auf die positive Flanke)<br>?swpol x |
| **Aktivierung:** | !validconfig bzw. !validpar |

| **Endschalter Ein/Aus** | |
|---|---|
| **Befehl:** | !swact oder ?swact |
| **Parameter:** | x, y, z oder a<br>0 oder 1 |
| **Beschreibung:** | !swact x 1 0 1 ➔ Endschalter der X-Achse :<br>E0=Ein REF=Aus EE=Ein |
| | ?swact z ➔ Zustand der Endschalter von Achse z auslesen. |
| **Rückmeldung:** | Zustand der Endschalter |
| **Fehlercode:** | -- |
| **Beispiel:** | !swact y 1 1 1 (Alle Schalter der Achse y aktiv)<br>?swact x |
| **Aktivierung:** | !validconfig bzw. !validpar |

| **Endschalter einlesen** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Befehl:** | ?readsw | | | | | | | | | | | |
| **Parameter:** | | | | | | | | | | | | |
| **Beschreibung:** | ?readsw ➔ Lesen aller Endschalterzustände. | | | | | | | | | | | |
| **Rückmeldung:** | Zustand der Endschalter. | | | | | | | | | | | |
| | **Achse:** | **x** | **y** | **z** | **a** | **x** | **y** | **z** | **a** | **x** | **y** | **z** | **a** |
| | **Schalter:** | **E0** | **E0** | **E0** | **E0** | **R** | **R** | **R** | **R** | **EE** | **EE** | **EE** | **EE** |
| | E0 = Null-Endschalter | | | | | R = Referenz-Endschalter | | | | EE = End-Endschalter | | | |
| **Fehlercode:** | -- | | | | | | | | | | | |
| **Beispiel:** | ?readsw (Lesen aller Endschalter) | | | | | | | | | | | |

| **Stopeingang Polarität einstellen** | |
|---|---|
| **Befehl:** | !stoppol oder ?stoppol |
| **Parameter:** | 0 = lowaktiv 1 = highaktiv |
| **Beschreibung:** | Da der Stopeingang einen Pull Up nach 5V hat, muss man bei einem Schließer lowaktiv und bei einem Öffner highaktiv einstellen. |
| **Rückmeldung:** | -- |
| **Fehlercode:** | -- |
| **Beispiel:** | !stoppol 1 (der Stopeingang ist highaktiv) |
| **Aktivierung:** | sofort |

| **Stoppeingang Bremsbeschleunigung einstellen** | | | |
|---|---|---|---|
| **Befehl:** | !stopdecel oder ?stopdecel | | |
| *Funktionsbeschreibung:* | *Die Verzögerung der einzelnen Achsen lassen sich durch den Befehl konfigurieren. Damit lässt sich der Wert einstellen, mit dem die Achsen in Falle eines Stoppsignals bremsen sollen.* | | |
| **Parameter:** | X, y, z und a<br>0.01 – 20.00 [m/s²] | | |
| **Beschreibung:** | !stopdecel 1.00 1.50 | ➔ | Bei Achse x und y werden die Verzögerungen (x=1.00, y=1.50 [m/s²]) eingestellt, die anderen Achsen bleiben unverändert. |
| | !stopdecel x 1 | ➔ | Die Verzögerung für Achse x wird auf 1.00 [m/s²] eingestellt. |
| | ?stopdecel | ➔ | Alle eingestellten Verzögerungen werden angezeigt. |
| | ?stopdecel z | ➔ | Die eingestellte Verzögerung der z-Achse wird angezeigt. |
| **Rückmeldung:** | Eingestellter Verzögerungswert bei betätigtem Stoppeingang | | |
| **Fehlercode:** | -- | | |
| **Beispiel:** | !stopdecel 10 (Verzögerung in der Achse bei 10m/s²)<br>?decel | | |
| **Aktivierung:** | sofort | | |

| **Stoppeingang Ruckbegrenzung** | | |
|---|---|---|
| **Befehl:** | !stopdeceljerk oder ? stopdeceljerk | |
| *Funktionsbeschreibung:* | *Bei schwingungskritischer Maschinenmechanik sind diese Profile zu verwenden. Die Rampenzeiten für den Beschleunigungsaufbau und Abbau sollten nur so groß wie erforderlich und so klein wie möglich eingestellt werden, da diese Parameter erheblichen Einfluss auf die Positionierzeiten haben Dieser Befehl ermöglicht die Konfiguration des Rucks jeder einzelnen Achse während der Verzögerung, im Falle eines Notstopps.* | |
| **Parameter:** | X, y, z und a<br>1, 2, 3, . . . . . ∞ [m/s³] (Natürliche Zahlen) | |
| **Beschreibung:** | !stopdeceljerk 4 1 | ➔ Ruck 4 m/s³ bei X und 1 m/s³ wird bei Y programmiert |
| | !stopdeceljerk z 10 | ➔ Ruck 10 m/s³ wird bei z wird programmiert. |
| | ?stopdeceljerk | ➔ Es werden alle konfigurierten Werte angezeigt. |
| | ?stopdeceljerk a | ➔ Es wird die Ruckeinstellung der a-Achse angezeigt. |
| **Rückmeldung:** | Ruckeinstellung während Verzögerung des Systems im Falle eines Notstoppsignals. | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !stopdeceljerk 10 (Ein Ruck von 10m/s³ wird bei x eingestellt)<br>?stopdeceljerk | |
| **Aktivierung:** | sofort | |

## 4.7 Mechanischen Arbeitsbereich ermitteln

Nach dem initialisieren der Steuerung sollten die Befehle Kalibrieren „cal“ und Hub-Messen „rm“ausgeführt werden. Dadurch wird der maximale Mechanische Arbeitsbereich ermittelt. Hierdurch ist gewärleistet, dass die Achsen nicht mehr in die Endschalter verfahren werden können.
Das messen des Arbeitshubes ist nur möglich, wenn alle Achen einen Null- sowie einen Endendschalter besitzen.
Damit beim Anfahren der Null- oder Endposition durch ein Überschwingen der Mechanik die Endschalter nicht ansprechen, kann mit den Befehlen „caliboffset“ und „rmoffset“der Arbeitsbereich eingeengt werden.

| **Kalibrieren** | |
|---|---|
| **Befehl:** | !cal oder cal |
| **Parameter:** | X, y, z oder a |
| **Beschreibung:** | Cal ➔ Bewegt alle freigegebenen Achsen in Richtung kleinerer Positionswerte. Die Verfahrbewegung wird unterbrochen sobald die Endschalter angefahren wurden und dann langsam in entgegengesetzter Richtung gefahren bis der Schalter nicht mehr aktiv ist. Der Positionswert wird auf 0 gesetzt. Die Position wird als Softwaregrenze, wie unter dem Befehl „Limit“ beschrieben, übernommen. |
| | Cal y ➔ Wie oben jedoch nur y-Achse. |
| **Rückmeldung:** | Für jede kalibrierte Achse ein ‚A‘oder ‘E’ bei einem Fehler |
| **Fehlercode:** | -- |
| **Beispiel:** | !cal |

| **Tischhub messen** | |
|---|---|
| **Befehl:** | !rm oder rm |
| **Parameter:** | X, y, z oder a |
| **Beschreibung:** | Rm ➔ Bewegt alle freigegebenen Achsen in Richtung größerer Positionswerte. Die Verfahrbewegung wird unterbrochen sobald die Endschalter angefahren wurden und dann langsam in entgegengesetzter Richtung gefahren bis der Schalter nicht mehr aktiv ist. Der Positionswert wird gespeichert und als Softwaregrenze, wie unter dem Befehl „Limit“ beschrieben, übernommen. |
| | Rm z ➔ Wie oben jedoch nur die z-Achse |
| **Rückmeldung:** | Für jede Achse ein ‚D‘ |
| **Fehlercode:** | -- |
| **Beispiel:** | !rm |

| RM Offset | |
|---|---|
| **Befehl:** | !rmoffset oder ?rmoffset |
| **Parameter:** | X, y, z oder a<br>0 – 32*50000 (32*Spindelsteigung) |
| **Beschreibung:** | !rmoffset 1 1 1 ➔ Die Achsen X, Y und Z werden beim Tischhub messen jeweils 1mm (bei Dim 2 2 2) vom Endendschalter in Richtung Tischmitte verfahren und dann die Softwaregrenze gesetzt. |
| | ?rmoffset y ➔ Aktuellen Offset der Y-Achse lesen. |
| **Rückmeldung:** | Strecke |
| **Fehlercode:** | -- |
| **Beispiel:** | ?rmoffset |
| **Aktivierung:** | sofort |

| Kalibrier-Offset | |
|---|---|
| **Befehl:** | !caliboffset oder ?caliboffset |
| **Parameter:** | X, y, z oder a<br>0 – 32*50000 (32*Spindelsteigung) |
| **Beschreibung:** | !caliboffset 1 1 1 ➔ Die Achsen X, Y und Z werden beim Kalibrieren jeweils 1mm (bei Dim 2 2 2) vom Nullendschalter in Richtung Tischmitte verfahren und dann die Position Null gesetzt (Softwaregrenze). |
| | ?caliboffset y ➔ Aktuellen Offset der Y-Achse lesen |
| **Rückmeldung:** | Strecke |
| **Fehlercode:** | -- |
| **Beispiel:** | ?caliboffset |
| **Aktivierung:** | sofort |

| Kalibriereinstellung - Richtung | |
|---|---|
| **Befehl:** | !?caldir |
| *Funktionsbeschreibung:* | *Durch den Befehl lässt sich die Kalibrierrichtung einstellen.* |
| **Parameter:** | X, y, z oder a<br>0 oder 1 |
| **Bemerkung:** | Beim kalibrieren in positiver Richtung wird das positive Software Limit gesetzt. |
| **Beschreibung:** | !caldir 0 0 1 => Die Achsen X, Y werden in negativer Richtung und die Z-Achse in positiver Richtung kalibriert.<br>?caldir => Lese aktuelle Richtung für das kalibrieren. |
| **Rückmeldung** | 0 = negative Richtung<br>1 = positive Richtung |
| **Fehlercode:** | -- |
| **Beispiel:** | !caldir y 1 (Die Y-Achse wird in positiver Richtung kalibriert) |
| **Aktivierung:** | sofort |

| **Geschwindigkeit bei Fahrt aus dem Endschalter** | | | |
|---|---|---|---|
| **Befehl:** | !calibrmbspeed oder ?calibrmbspeed | | |
| *Funktionsbeschreibung:* | *Nachdem die Achse während des Kalibriervorgangs die Endschalter erreicht hat, fährt die Achse mit der mittels des Befehls eingestellter Geschwindigkeit wieder aus dem Endschalter heraus.*<br>***Hinweis:***<br>*Der bisherige Befehl calbspeed wird durch den neuen Befehl* calibrmbspeed ersetzt. | | |
| **Parameter:** | X, y, z und a<br>0 – maximale Geschwindigkeit | | |
| **Beschreibung:** | !calibrmbspeed 1.0 15 | ➔ | Bei Achse x und y werden die Geschwindigkeitswerte (x=1.0, y=15 [U/s]) beschrieben, die anderen Achsen bleiben unverändert. |
| | !calibrmbspeed z 0.1 | ➔ | Bei der z-Achse wird die Geschwindigkeit auf 0.1 [U/s] eingestellt. |
| | ?calibrmbspeed | ➔ | Alle eingestellten Geschwindigkeiten werden angezeigt. |
| | ?calibrmbspeed x | ➔ | Anzeigen der eingestellten Geschwindigkeit von Achse x. |
| **Rückmeldung:** | Eingestellte Geschwindigkeit | | |
| **Fehlercode:** | -- | | |
| **Beispiel:** | !calibrmbspeed 10 (Die x-Achse wird mit maximal 10 U/s bei der Fahrt aus dem Endschalter heraus betrieben)<br>?calibrmbspeed | | |
| **Aktivierung:** | sofort | | |

| **Beschleunigung während des Kalibriervorgangs** | | | |
|---|---|---|---|
| **Befehl:** | !calibrmaccel oder ? calibrmaccel | | |
| *Funktionsbeschreibung:* | *Dieser Befehl erlaubt die Konfiguration der Beschleunigung während des Kalibriervorgangs.* | | |
| **Parameter:** | X, y, z und a<br>0.01 – 20.00 [m/s²] | | |
| **Beschreibung:** | !calibrmaccel 1.00 1.50 | ➔ | Bei Achse x und y werden die Beschleunigungen (x=1.00, y=1.50 [m/s²]) eingestellt, die anderen Achsen bleiben unverändert. |
| | !calibrmaccel x 1 | ➔ | Die Beschleunigung für Achse x wird auf 1.00 [m/s²] eingestellt. |
| | ?calibrmaccel | ➔ | Anzeige aller eingestellten Beschleunigungen |
| | ?calibrmaccel z | ➔ | Die eingestellte Beschleunigung der z-Achse wird angezeigt. |
| **Rückmeldung:** | Eingestellte Beschleunigung | | |
| **Fehlercode:** | -- | | |
| **Beispiel:** | !calibrmaccel 1.00 (Setze Beschleunigungen bei x-Achse auf 1 m/s²)<br>?accel | | |
| **Aktivierung:** | sofort | | |

| **Einstellung Ruck während des Kalibriervorgangs** | |
|---|---|
| **Befehl:** | !calibrmjerk oder ? calibrmjerk |
| *Funktionsbeschreibung:* | *Bei schwingungskritischer Maschinenmechanik sind diese Profile zu verwenden. Die Rampenzeiten für den Beschleunigungsaufbau und Abbau sollten nur so groß wie erforderlich und so klein wie möglich eingestellt werden, da diese Parameter erheblichen Einfluss auf die Positionierzeiten haben Dieser Befehl ermöglicht die Konfiguration des Rucks jeder einzelnen Achse während der Beschleuniguung.* |
| **Parameter:** | X, y, z und a<br>1, 2, 3, . . . . . ∞ [m/s³] (Natürliche Zahlen) |
| **Beschreibung:** | !calibrmjerk 4 1 ➔ Ruck 4 m/s³ bei X und 1 m/s³ wird bei Y programmiert |
| | !calibrmjerk z 10 ➔ Ruck 10 m/s³ wird bei z wird programmiert. |
| | ?calibrmjerk ➔ Es werden alle konfigurierten Werte angezeigt. |
| | ?calibrmjerk a ➔ Es wird die Ruckeinstellung der a-Achse angezeigt. |
| **Rückmeldung:** | Ruckeinstellung während Kalibriervorgangs |
| **Fehlercode:** | -- |
| **Beispiel:** | !calibrmjerk 10 (Ein Ruck von 10m/s³ wird bei x eingestellt)<br>?calibrmjerk |
| **Aktivierung:** | sofort |

| **Geschwindigkeit während des Kalibriervorgangs** | | |
|---|---|---|
| **Befehl:** | !calibrmvel oder ?calibrmvel | |
| *Funktionsbeschreibung:* | *Dieser Befehl erlaubt die Konfiguration der Geschwindigkeit während des Kalibriervorgangs.* | |
| **Parameter:** | X, y, z und a<br>0 – maximale Geschwindigkeit | |
| **Beschreibung:** | !calibrmvel 1.0 15 | ➔ Bei Achse x und y werden die Geschwindigkeitswerte (x=1.0, y=15 [U/s]) beschrieben, die anderen Achsen bleiben unverändert. |
| | !calibrmvel z 0.1 | ➔ Bei der z-Achse wird die Geschwindigkeit auf 0.1 [U/s] eingestellt. |
| | ?calibrmvel | ➔ Alle eingestellten Geschwindigkeiten werden angezeigt. |
| | ?calibrmvel x | ➔ Anzeigen der eingestellten Geschwindigkeit von Achse x. |
| **Rückmeldung:** | Eingestellte Geschwindigkeit | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !calibrmvel 10 (Die x-Achse wird mit maximal 10 U/s betrieben)<br>?calibrmvel | |
| **Aktivierung:** | sofort | |

## 4.8 Verfahrbefehle und deren Kontrollfunktionen

Bei allen Positionierbefehlen wird für mit einem Befehl gleichzeitig gestartete Achsen eine Linearinterpolation durchgeführt, d.h. alle Achsen erreichen die vorgegebene Position zum gleichen Zeitpunkt. Die Achse, wo der Motor die meisten Umdrehungen zurücklegen muss, gilt als Führungsachse und fährt somit mit der eingestellten Geschwindigkeit und Beschleunigung. Müssen alle Motoren die gleiche Strecke zurücklegen, ist die x-Achse die Führungsachse. Hierbei wird aber die eingestellte Geschwindigkeit und Beschleunigung von den anderen Achsen überwacht und nicht überschritten.
Hat man Achsen mit total unterschiedlichem dynamischem Verhalten, kann man diese auch einzeln starten. Auch ein assynchrones Verfahren ist möglich. Hierbei ist zu beachten, dass bei der Einstellung Autostatus 1 die Rückmeldung erst kommt, wenn alle Achsen stehen. Möchte man während sich eine Achse bewegt eine andere Achse mehrmals starten, setzt man den Autostatus = 0 und pollt mit ?statusaxis.

| **Position absolut** | |
|---|---|
| **Befehl:** | !moa oder moa |
| **Parameter:** | X, y, z oder a<br>+- Verfahrbereich |
| **Bemerkung:** | Die Eingabe ist abhängig von der Dimension. |
| **Beschreibung:** | Moa 10 0 20 ➔ Die Achsen x, y und z werden auf die eingegebenen Positionswerte positioniert. |
| | moa y 333 ➔ Wie oben jedoch nur y-Achse. |
| **Rückmeldung:** | Für jede positionierte Achse ein ‚@' |
| **Fehlercode:** | -- |
| **Beispiel:** | Moa x 10 (Die x-Achse wird auf die eingegebene Position positioniert) |

| **Position relativ** | |
|---|---|
| **Befehl:** | !mor oder mor |
| **Parameter:** | X, y, z oder a<br>+- Verfahrbereich |
| **Bemerkung:** | Die Eingabe ist abhängig von der Dimension. |
| **Beschreibung:** | Mor 100 0 39 ➔ Die Achsen x und z werden um die eingegebenen Strecken verfahren. |
| | Mor a 298 ➔ Die a-Achse wird um die eingegebene Strecke verfahren. |
| **Rückmeldung:** | Für jede verfahrene Achse ein ‚@' |
| **Fehlercode:** | -- |
| **Beispiel:** | !mor 0 0 0 100 (Nur die a-Achse wird verfahren) |

| Position relativ (Kurzbefehl) | |
|---|---|
| **Befehl:** | !m oder m |
| **Parameter:** | |
| **Bemerkung:** | Dieser Befehl wird verwendet, wenn in sehr kurzer Abfolge immer wieder die gleiche Strecke Verfahren werden soll.<br>Die zu verfahrende Strecke muss vorher mit !distance oder mor befehlen gesetzt werden. Die Position wird nicht aktualisiert, erst wieder beim nächsten Move-Befehl. |
| **Beschreibung:** | m ➔ Start einer Verfahrbewegung aller freigegebenen Achsen. |
| **Rückmeldung:** | Je nach Einstellung von autostatus. |
| **Fehlercode:** | -- |
| **Beispiel:** | !mor 0 0 0 100 (Nur die a-Achse wird verfahren)<br>m (A-Achse wird wieder um 100 verfahren) |

| Strecke | | |
|---|---|---|
| **Befehl:** | !distance oder ?distance | |
| **Parameter:** | X,y,z und a<br>Min-/max-Verfahrbereich | |
| **Bemerkung:** | Ein- und Ausgabe ist abhängig von der Dimension. | |
| **Beschreibung:** | !distance 1 2 3 | ➔ Die Strecken für die Achsen x, y und z werden gesetzt. |
| | !distance y 20 | ➔ Die Strecke der y-Achse wird gesetzt. |
| | ?distance | ➔ Abfrage der aktuellen Strecken aller Achsen. |
| | ?distance z | ➔ Abfrage der aktuellen Strecke von Achse z. |
| **Rückmeldung:** | Strecken | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !distance 10 20 (Setzen der Strecken von x- und y-Achse)<br>?distance (Abfrage der Strecken aller Achsen) | |
| **Aktivierung:** | sofort | |

| SpeedPoti | |
|---|---|
| **Befehl:** | !pot oder ?pot |
| **Parameter:** | 0 oder 1 |
| **Bemerkung:** | |
| **Beschreibung:** | 0 ➔ Es wird die vorgegebene Geschwindigkeit (vel) als Verfahrgeschwindigkeit genutzt. |
| | 1 ➔ Es wird die vorgegebene Geschwindigkeit (vel), in Abhängigkeit von der Stellung des Potentiometers, prozentual genutzt. |
| **Rückmeldung:** | -- |
| **Fehlercode:** | -- |
| **Beispiel:** | !pot 1<br>?pot |
| **Aktivierung:** | sofort |

| Position | | |
|---|---|---|
| **Befehl:** | !pos oder ?pos | |
| **Parameter:** | X,y,z und a<br>Min-/max-Verfahrbereich | |
| **Bemerkung:** | Ein- und Ausgabe ist abhängig von der Dimension. | |
| **Beschreibung:** | !pos 1000 2000 3000 | ➔ Die Positionswerte für die Achsen x, y und z werden gesetzt. |
| | !pos y 2000 | ➔ Die Position der y-Achse wird gesetzt. |
| | ?pos | ➔ Abfrage der aktuellen Position aller Achsen. |
| | ?pos z | ➔ Abfrage der aktuellen Position von Achse z. |
| **Rückmeldung:** | Positionswerte | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !pos100 200 (Setzen der Positionen von x- und y-Achse)<br>?pos (Abfrage der Positionen aller Achsen) | |
| **Aktivierung:** | sofort | |

| Abbruch | |
|---|---|
| **Befehl:** | !a oder a |
| **Parameter:** | Keine |
| **Beschreibung:** | Es werden alle Verfahrbewegungen abgebrochen. |
| **Rückmeldung:** | Für jede Achse ein @ |
| **Fehlercode:** | -- |
| **Beispiel:** | !a |

| Tabellenposition setzen | | |
|---|---|---|
| **Befehl:** | (?)!tpos (Set Table Pos) | |
| **Parameter:** | 1 – 1.000<br>Positionen für x,y,z,a in den Verfahrbereichsgrenzen | |
| **Bemerkung:** | In der Tabelle können bis zu 1.000 Positionen hinterlegt werden. Die Eingabe ist abhängig von der Dimension. | |
| **Beschreibung:** | !tpos x 5 10 | ➔ Der Tabellenposition 5 der X-Achse ist die Position 10 hinterlegt |
| | !tpos 5 10 10 10 10 | ➔ Wie oben jedoch für alle Achsen |
| | ?tpos 5 | ➔ Abfrage der Tabellenposition 5 für alle Achsen |
| **Rückmeldung:** | -- | |
| **Fehlercode:** | -- | |
| **Beispiel:** | Moa x 10 (Die x-Achse wird auf die eingegebene Position positioniert) | |

| Tabellenposition anfahren | |
|---|---|
| **Befehl:** | !mtpa (Move Table Pos Absolut) |
| **Parameter:** | X, y, z oder a<br>1 – 1.000 |
| **Bemerkung:** | |
| **Beschreibung:** | !mtpa 5 5 5 5 ➔ Fährt alle Achsen auf die in der Position 5 der Tabelle gespeicherten Positionen |
| | !mtpa y 5 ➔ Wie oben jedoch nur y-Achse. |
| **Rückmeldung:** | Für jede positionierte Achse ein ‚@' |
| **Fehlercode:** | -- |
| **Beispiel:** | !mtpa x 10 (Die x-Achse wird auf die Position 10 der Tabelle positioniert) |

| Index Table Divider | |
|---|---|
| **Befehl:** | (?)!itd (Index Table Driver) |
| **Parameter:** | X, y, z oder a<br>1 – 1.000.000 |
| **Bemerkung:** | Mit diesem Befehl lässt sich z.B. eine Drehachse in lauter gleich große Winkel unterteilen. Ein Anwendungsbeispiel ist ein Werkzeugwechsler bei dem n-Werkzeuge auf einem Drehteller angeordnet sind mit jeweils dem gleichen Abstand. |
| **Beschreibung:** | !itd a 50 ➔ bei einer Dreheinheit die an der a-Achse angeschlossen ist, wird eine Umdrehung in 50 Abschnite unterteilt, dies entspricht 7,2° pro Abschnitt. |
| | ?itd a ➔ Auslesen des eingestellten Index der a-Achse |
| **Rückmeldung:** | -- |
| **Fehlercode:** | -- |
| **Beispiel:** | !itd x 360 (für die x-Achse wird der Index auf 360 gesetzt) |

| **Move Index Table** | | |
|---|---|---|
| **Befehl:** | !mita (Move Index Table) | |
| **Parameter:** | X, y, z oder a<br>1 – 1.000.000 | |
| **Bemerkung:** | | |
| **Beschreibung:** | !mita a 10 | ➔ Bei einer Einstellung von itd=50, wird die a-Achse auf 72° positioniert |
| | !mita a 1 | ➔ Wie oben jedoch auf 7,2° |
| **Rückmeldung:** | Für jede positionierte Achse ein ‚@' | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !mita x 10 (Die x-Achse wird auf Index 10 positioniert) | |

## 4.9 Joystick- Tippbetrieb- und Trackball-Befehle

| | |
|---|---|
| **Hinweis:** | 1. Mit dem Befehl „speed“ lässt sich per Software ein Joystick ähnliches Verhalten erreichen.<br>2. Der Joystick lässt sich per Befehl einschalten, und auch bei aktivem Joystick sind Move-Befehle möglich.<br>3. Sendet man bei eingeschaltetem Joystick den Save-Befehl, so ist der Joystick nach dem Einschalten der Steuerung oder des PCs schon aktiv.<br>4. Die Eingänge für Tippbetrieb, Trackball und die Encodereingänge Quep1 und Quep2 schließen einander aus, es ist nur eine Funktion möglich. |

| **Digitaler Joystick (Geschwindigkeit)** | | |
|---|---|---|
| **Befehl:** | !speed oder ?speed | |
| **Parameter:** | x, y, z oder a<br>+- maximale Geschwindigkeit (vel) | |
| **Beschreibung:** | Mit diesem Befehl können einzelne Achsen mit einer konstanten Geschwindigkeit verfahren werden. | |
| | !speed 0 | ➔ Alle Achsen Geschwindigkeit 0 und Joystick-Betrieb „AUS„ |
| | !speed 10 | ➔ Alle Achsen Geschwindigkeit 10.0 [U/s] und Joystick-Betrieb „EIN„. |
| | !speed 10 10 0 10 | ➔ Achsen x, y und a Geschwindigkeit 10.0 [U/s], Achse z Geschwindigkeit 0 und Joystick-Betrieb „EIN„. |
| | !speed y 25 | ➔ Achse y Geschwindigkeit 25 und Joystick-Betrieb „EIN„. |
| | !speed y -25 | ➔ Achse y Geschwindigkeit 25 in negative Richtung Joystick-Betrieb „EIN„. |
| | ?speed | ➔ Auslesen der eingestellten Geschwindigkeiten. |
| | ?speed y | ➔ Auslesen der eingestellten Geschwindigkeit von Achse y. |
| **Rückmeldung:** | Aktuelle Geschwindigkeiten | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !speed 33 11 (Achsen x Geschwindigkeit 33.0 [U/s], Achse y Geschwindigkeit 11.0 [U/s] und Joystick-Betrieb „EIN„)<br>?speed | |
| **Anmerkung:** | Will man nach dem Ausführen des speed-Befehls wieder absolut oder relativ positionieren, muss man erst mit !speed 0 den digitalen Joystick ausschalten und die Geschwindigkeit neu setzen. | |

| Joystick-Richtung | |
|---|---|
| **Befehl:** | !joydir oder ?joydir |
| **Parameter:** | x y z a<br>0 oder 1 |
| **Beschreibung:** | Mit der Eingabe von joydir wird die Drehrichtung der Motoren, bei Auslenkung des Joysticks getauscht.<br>!joydir x 1 (Bei der X-Achse wurde die Drehrichtung getauscht.) |
| **Rückmeldung:** | Eingestellte Joystick Richtungen. |
| **Fehlercode:** | -- |
| **Beispiel:** | !joydir 0 1 0 0<br>?joydir |
| **Aktivierung:** | !joy 1 |

| Joystick Geschwindigkeit | |
|---|---|
| **Befehl:** | !joyvel oder ? joyvel |
| ***Funktionsbeschreibung:*** | *Mit diesem Befehl lassen sich die maximalen Verfahrgeschwindigkeiten im Joystickbetrieb einstellen.* |
| **Parameter:** | x, y, z, a<br>0 – 70 Umdr./sec. |
| **Beschreibung** | !joyvel x 40 (die X-Achse fährt bei Vollauslenkung des Joystick mit 40 Umdr./sec.) |
| **Rückmeldung:** | Eingestellte Geschwindigkeit |
| **Fehlercode:** | -- |
| **Beispiel:** | !joyvel 10 10 10 10 (alle Achse werden mit max. 10 U/s betrieben)<br>?joyvel |
| **Aktivierung:** | !joy 1 |

| Joystick Filterzeitkonstante | |
|---|---|
| **Befehl:** | !joyoutpass oder ?joyoutpass |
| ***Funktionsbeschreibung:*** | *Mit diesem Befehl lassen sich die Eingangswerte filtern, um ruckartige Veränderungen der Geschwindigkeit zu verhindern. (Rampenfunktion)* |
| **Parameter:** | x, y, z, a<br>0 – 500000µs |
| **Beschreibung** | !joyoutpass x 40 (die Filterzeitkonstante für die X-Achse wird auf 40µs eingestellt |
| **Rückmeldung:** | Eingestellte Filterzeitkonstante |
| **Fehlercode:** | -- |
| **Beispiel:** | ! joyoutpass 10 10 10 10 (Filterzeitkonstante für alle Achse auf 10µs einstellen)<br>? joyoutpass |
| **Aktivierung:** | !joy 1 |

| Joystick mit Stromreduzierung | |
|---|---|
| **Befehl:** | !joyredcur oder ?joyredcur |
| **Parameter:** | X, y, z, a<br>0 oder 1 |
| **Beschreibung:** | Mit der Eingabe von joyredcur lässt sich die Stromreduzierung im Joystickbetrieb Ein- und Ausschalten. Bei aktiver Stromreduzierung wird bei Nichauslenkung des Joystick in Mittelstellung der Motorstrom auf den mit „reduction“ eingestellten Wert abgesenkt.<br>!joyredcur x 1 (Die Stromreduzierung für die X-Achse ist aktiv.) |
| **Rückmeldung:** | Einstellungen. |
| **Fehlercode:** | -- |
| **Beispiel:** | !joyredcur 1 1 1 0<br>?joyredcur |
| **Aktivierung:** | !joy 1 |

| Joystick Window (joywindow) | |
|---|---|
| **Befehl:** | !?joywindow |
| **Parameter:** | 0 – 100 |
| **Beschreibung:** | Mit der Eingabe von joywindow wird der Analogbereich festgelegt in dem sich die Achsen nicht bewegen. Gilt für alle Achsen. |
| | joywindow 10 => Es muss eine größere Auslenkung des Joystickes vorliegen als 10 („Punkte“), damit die Achsen sich bewegen. |
| | ?joywindow => Auslesen des Joystick - Fensters . |
| | Beispiel:<br>Nullstellung Joystick = 512 (Analogwert)<br>Joywindow = 10<br>D.h., dass die Achsen bei Werten < 502 und > 522 bewegt werden |
| **Rückmeldung:** | Eingestelltes Fenster |
| **Fehlercode:** | -- |
| **Beispiel:** | ?joywindow (Auslesen der Fenstergröße) |
| **Aktivierung:** | !joy 1 |

| Joystick Freigabe | |
|---|---|
| **Befehl:** | !joyenable oder ?joyenable |
| **Parameter:** | 0 oder 1 |
| **Beschreibung:** | Mit der Eingabe von joyenable lassen sich angeschlossene Joystickachsen freigeben oder sperren.<br>!joyenable x 1 (Der Joystick für die X-Achse ist freigegeben) |
| **Rückmeldung:** | Einstellungen. |
| **Fehlercode:** | -- |
| **Beispiel:** | !joyenable 1 1 1 0 (Freigabe für X Y Z)<br>?joyenable |
| **Aktivierung:** | !joy 1 |

| Joystick-Achsen-Zuordnung | |
|---|---|
| **Befehl:** | !joytoaxis oder ?joytoaxis |
| **Parameter:** | X y z a<br>0 bis 4 |
| **Beschreibung:** | Mit der Eingabe von joyenable lassen sich die Joystickachsen beliebig zuordnen. Da es meist nur 2 und 3 Achsen Joysticks gibt, kann man z.B. mit dem Z-Joystick die 4. Achse bewegen.<br>!joytoaxis a 3<br>Nicht zulässig ist, ein Joystickeingang gleichzeitig auf zwei Achsen zu legen. |
| **Rückmeldung:** | Eingestellte Zuordnung. |
| **Fehlercode:** | -- |
| **Beispiel:** | !joytoaxis 1 2 3 4 ( X=1 Y=2 Z=3 A=4 )<br>?joytoaxis |
| **Aktivierung:** | !joy 1 |

| Joystick | |
|---|---|
| **Befehl:** | !joy oder ?joy |
| **Parameter:** | 0 oder 1 |
| **Beschreibung** | Mit diesem befehl lässt sich der Joystick Ein- und Ausschalten<br>!joy 1 (Joystick Ein) / !joy 0 (Joystick Aus) |
| **Rückmeldung:** | @@@@ |
| **Fehlercode:** | -- |
| **Beispiel:** | !joy 1<br>?joy |

| Tippbetrieb Geschwindigkeit | |
|---|---|
| **Befehl:** | !tippvel oder ?tippvel |
| *Funktionsbeschreibung:* | *Mit diesem Befehl lassen sich die Verfahrgeschwindigkeiten im Tippbetrieb einstellen.* |
| **Parameter:** | x, y, z, a<br>0 – 70 Umdr./sec. |
| **Beschreibung** | !tippvel x 40 (die X-Achse fährt im Tippbetrieb mit 40 Umdr./sec.) |
| **Rückmeldung:** | Eingestellte Geschwindigkeit |
| **Fehlercode:** | -- |
| **Beispiel:** | !tippvel 10 10 10 10 (alle Achse werden mit max. 10 U/s betrieben)<br>?tippvel |
| **Aktivierung:** | !tipp 1 |

| Tippbetrieb Filterzeitkonstante | |
|---|---|
| **Befehl:** | !tippoutpass oder ?tippoutpass |
| *Funktionsbeschreibung:* | *Mit diesem Befehl lassen sich die Eingangswerte filtern, um ruckartige Veränderungen der Geschwindigkeit zu verhindern. (Rampenfunktion)* |
| **Parameter:** | x, y, z, a<br>0 – 500000µs |
| **Beschreibung** | !tippoutpass x 40 (Die Filterzeitkonstante für die X-Achse wird auf 40µs eingestellt) |
| **Rückmeldung:** | Eingestellte Filterzeitkonstante |
| **Fehlercode:** | -- |
| **Beispiel:** | !tippoutpass 10 10 10 10 (Filterzeitkonstante für alle Achse werden auf 10µs eingestellt)<br>?tippoutpass |
| **Aktivierung:** | !tipp 1 |

| Tippbetrieb-Richtung | |
|---|---|
| **Befehl:** | !tippdir oder ?tippdir |
| **Parameter:** | x y z a<br>0 oder 1 |
| **Beschreibung:** | Mit der Eingabe von tippdir wird die Drehrichtung der Motoren, für die Richtungstasten getauscht.<br>!tippdir x 1 (Bei der X-Achse wurde die Drehrichtung geändert.) |
| **Rückmeldung:** | Eingestellte Tippbetrieb Richtungen. |
| **Fehlercode:** | -- |
| **Beispiel:** | !tippdir 0 1 0 0<br>?tippdir |
| **Aktivierung:** | !tipp 1 |

| Tippbetrieb Freigabe | |
|---|---|
| **Befehl:** | !tippenable oder ?tippenable |
| **Parameter:** | 0 oder 1 |
| **Beschreibung:** | Mit der Eingabe von tippenable lassen sich angeschlossene Achsen für den Tippbetrieb freigeben oder sperren.<br>!tippenable x 1 (Die X-Achse ist für den Tippbetrieb freigegeben) |
| **Rückmeldung:** | Einstellungen. |
| **Fehlercode:** | -- |
| **Beispiel:** | !tippenable 1 1 1 0 (Freigabe für X Y Z)<br>?tippenable |
| **Aktivierung:** | !tipp 1 |

| Tippbetrieb mit Stromreduzierung | |
|---|---|
| **Befehl:** | !tippredcur oder ?tippredcur |
| **Parameter:** | X, y, z, a<br>0 oder 1 |
| **Beschreibung:** | Mit der Eingabe von tippredcur lässt sich die Stromreduzierung im Tippbetrieb Ein- und Ausschalten. Bei aktiver Strom-reduzierung wird bei allen Achsen im Stillstand der Motorstrom auf den mit „reduction“ eingestellten Wert abgesenkt.<br>!tippredcur x 1 (Die Stromreduzierung für die X-Achse ist aktiv.) |
| **Rückmeldung:** | Einstellungen. |
| **Fehlercode:** | -- |
| **Beispiel:** | !tippredcur 1 1 1 0<br>?tippredcur |
| **Aktivierung:** | !tipp 1 |

| Tippbetrieb Ein- oder Ausschalten | |
|---|---|
| **Befehl:** | !tipp oder ?tipp |
| **Parameter:** | 0 oder 1 |
| **Beschreibung:** | mit !tipp 1 lässt sich der Tippbetrieb einschalten<br>mit !tipp 0 lässt sich der Trackball ausschalten |
| **Rückmeldung:** | Einstellungen. |
| **Fehlercode:** | -- |
| **Beispiel:** | !tipp 1 (Tippbetrieb wird eingeschaltet)<br>?tipp (Abfrage ob der Tippbetrieb Ein- oder ausgeschaltet ist) |
| **Aktivierung:** | sofort |

| Trackball Geschwindigkeit | |
|---|---|
| **Befehl:** | !tbvel oder ?tbvel |
| *Funktionsbeschreibung:* | *Mit diesem Befehl lassen sich die Verfahrgeschwindigkeiten im Trackballbetrieb einstellen.* |
| **Parameter:** | x, y, z, a<br>0 – 70 Umdr./sec. |
| **Beschreibung** | !tbvel x 40 (die X-Achse fährt im Trackballbetrieb mit maximal 40 Umdr./sec.) |
| **Rückmeldung:** | Eingestellte Geschwindigkeit |
| **Fehlercode:** | -- |
| **Beispiel:** | !tbvel 10 10 0 0 (alle Achse werden mit max. 10 U/s betrieben)<br>?tbvel |
| **Aktivierung:** | !tb 1 |

| Trackball Geschwindigkeit | |
|---|---|
| **Befehl:** | !tboutpass oder ?tboutpass |
| *Funktionsbeschreibung:* | *Mit diesem Befehl lassen sich die Eingangswerte filtern, um ruckartige Veränderungen der Geschwindigkeit zu verhindern. (Rampenfunktion)* |
| **Parameter:** | x, y, z, a<br>0 – 500000µs |
| **Beschreibung** | !tboutpass x 40 ( die Filterzeitkonstante der X-Achse wird auf 40µs eingestellt) |
| **Rückmeldung:** | Eingestellte Filterzeitkonstante |
| **Fehlercode:** | -- |
| **Beispiel:** | !tboutpass 10 10 0 0 (die Filterzeitkonstanten aller Achsen wird auf 10µs eingestellt)<br>?tboutpass |
| **Aktivierung:** | !tb 1 |

| Trackball-Richtung | |
|---|---|
| **Befehl:** | !tbdir oder ?tbdir |
| **Parameter:** | x y z a<br>0 oder 1 |
| **Beschreibung:** | Mit der Eingabe von tbdir wird die Drehrichtung der Motoren, für den Trackball getauscht.<br>!tbdir x 1 (Bei der X-Achse wurde die Drehrichtung geändert.) |
| **Rückmeldung:** | Eingestellte Trackball Richtungen. |
| **Fehlercode:** | -- |
| **Beispiel:** | !tbdir 0 1 0 0<br>?tbdir |
| **Aktivierung:** | !tb 1 |

| Trackball Freigabe | |
|---|---|
| **Befehl:** | !tbenable oder ?tbenable |
| **Parameter:** | 0 oder 1 |
| **Beschreibung:** | Mit der Eingabe von tbenable lassen sich angeschlossene Achsen für den Trackballbetrieb freigeben oder sperren.<br>!tbenable x 1 (Die X-Achse ist für den Trackballbetrieb freigegeben) |
| **Rückmeldung:** | Einstellungen. |
| **Fehlercode:** | -- |
| **Beispiel:** | !tbenable 1 1 0 0 (Freigabe für X Y)<br>?tbenable |
| **Aktivierung:** | !tb 1 |

| Trackball mit Stromreduzierung | |
|---|---|
| **Befehl:** | !tbredcur oder ?tbredcur |
| **Parameter:** | X, y, z, a<br>0 oder 1 |
| **Beschreibung:** | Mit der Eingabe von tbredcur lässt sich die Stromreduzierung im Trackballbetrieb Ein- und Ausschalten. Bei aktiver Strom-reduzierung wird bei allen Achsen im Stillstand der Motorstrom auf den mit „reduction“ eingestellten Wert abgesenkt.<br>!tbredcur x 1 (Die Stromreduzierung für die X-Achse ist aktiv.) |
| **Rückmeldung:** | Einstellungen. |
| **Fehlercode:** | -- |
| **Beispiel:** | !tbredcur 1 1 1 0<br>?tbredcur |
| **Aktivierung:** | !tb 1 |

| Trackball-Achsen-Zuordnung | |
|---|---|
| **Befehl:** | !tbtoaxis oder ?tbtoaxis |
| **Parameter:** | X y z a<br>0 = keine Achszuordnung<br>1 = Trackball horizontal<br>2 = Trackball vertikal |
| **Beschreibung:** | Mit der Eingabe von tbenable lassen sich die Trackballachsen beliebig zuordnen. Da ein Trackball nur 2 Achsen hat, kann man ihn beliebigen Achsen zuordnen.<br>!tbtoaxis z 1 (Die Z-Achse wird mit der horizontalenTrackballrichtung verfahren.)<br>Nicht zulässig ist, ein Trackballeingang gleichzeitig auf zwei Achsen zu legen. |
| **Rückmeldung:** | Eingestellte Zuordnung. |
| **Fehlercode:** | -- |
| **Beispiel:** | !tbtoaxis 1 2 0 0 ( X=1 Y=2 Z=0 A=0 )<br>?tbtoaxis |
| **Aktivierung:** | !tb 1 |

| Trackball Ein- oder Ausschalten | |
|---|---|
| **Befehl:** | !tb oder ?tb |
| **Parameter:** | 0 oder 1 |
| **Beschreibung:** | mit !tb 1 lässt sich der Trackball einschalten<br>mit !tb 0 lässt sich der Trackball ausschalten |
| **Rückmeldung:** | Einstellungen. |
| **Fehlercode:** | -- |
| **Beispiel:** | !tb 1 (Trackball wird eingeschaltet)<br>?tb (Abfrage ob der Teackball Ein- oder ausgeschaltet ist) |
| **Aktivierung:** | sofort |

## 4.10 Ein/Ausgänge

Optional kann die LSTEPexpress oder PCIexpress mit 16 digitalen Ein- und Ausgängen ausgestattet werden. Zur Benutzung dieser Ein/Ausgänge muss die Steuerung in der entsprechenden Ausführung bestellt werden.

| **Digitale Eingänge** | |
|---|---|
| **Befehl:** | ?digin |
| **Parameter:** | 0 bis 15 |
| **Beschreibung:** | ?digin ➔ Lesen aller Inputpins |
| | ?digin 8 ➔ Lesen des Inputpins 8 |
| **Rückmeldung:** | Zustand der Inputpins |
| **Fehlercode:** | -- |
| **Beispiel:** | ?digin (Lesen aller Inputpins) |

| **Digitale Ausgänge** | | |
|---|---|---|
| **Befehl:** | !digout oder ?digout | |
| **Parameter:** | 0 bis 15 | |
| **Beschreibung:** | !digout 11110000 | ➔ Es werden die Outputpins 0,1,2,3 auf „1„ und die Outputpins 4,5,6,7 auf „0„ gesetzt. |
| | !digout 5 1 | ➔ Es wird Outputpin 5 auf „1„ gesetzt. |
| | ?digout | ➔ Lesen des aktuellen Zustands aller Outputpins. |
| | ?digout 8 | ➔ Lesen des aktuellen Zustands von Outputpin 8 |
| **Rückmeldung:** | Zustand der Outputpins | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !digout 7 0 (Setze Outputpin 7 auf „0„)<br>?digout (Lesen aller Outputpins) | |

| Analoge Ausgänge | |
|---|---|
| **Befehl:** | !anaout oder ?anaout |
| **Parameter:** | 0 bis 100 %<br>0 und 1 (Analogkanäle)<br>c (c = channel) |
| **Bemerkung:** | Channel 0 und 1 sind auf dem Multifunktionsport der Steuerung. |
| **Beschreibung:** | !anaout 100 50 ➔ Es wird der erste Analogkanal auf 100% (volle Spg.) und der zweite auf 50% (halbe Spg.) gestellt. |
| | !anaout c 1 25 ➔ Es wird Analogkanal 1 auf 25% eingestellt. |
| | ?anaout ➔ Lesen des aktuellen Zustands aller Analogkanäle. |
| | ?anaout c 0 ➔ Lesen des aktuellen Zustands von Analogkanal 0. |
| **Rückmeldung:** | Zustand der prozentualen Aussteuerung der Analogkanäle. |
| **Fehlercode:** | -- |
| **Beispiel:** | !anaout c 1 0 (Setze Analogkanal 1 auf „0„)<br>?anaout (Lesen aller Analogkanäle) |

| Analoge Eingänge | |
|---|---|
| **Befehl:** | ?anain |
| **Parameter:** | 0 bis 13 (Analogkanäle)<br>c (c = channel) |
| **Beschreibung:** | ?anain c 2 => Lesen des aktuellen Zustands von Analogkanal 2 |
| **Rückmeldung:** | Zustand je nach Analogkanal |
| **Fehlercode:** | -- |
| **Beispiel:** | ?anain (Lesen des aktuellen Zustands aller Analogkanäle) |

| Channel / Analog Eingänge | | | |
|---|---|---|---|
| **Channel** | **MFP** | **50pol (25pol)** | **Funktion** |
| **0** | MFP | Pin 1 (24) | Joystick Achse 1 |
| **1** | MFP | Pin 2 (12) | Joystick Achse 2 |
| **2** | MFP | Pin 3 (25) | Joystick Achse 3 |
| **3** | MFP | Pin 4 (n.c.) | Joystick Achse 4 |
| **4** | MFP | Pin 11 (n.c.) | Potentiometer 1 |
| **5** | | | Motorspannung |
| **6** | MFP | Pin 5 (8) | analoger Eingang / 0…5V |
| **7** | MFP | Pin 6 (20) | analoger Eingang / 0…5V |
| **8** | MFP | Pin 7 (7) | analoger Eingang / 0…5V |
| **9** | MFP | Pin 8 (19) | analoger Eingang / 0…5V |
| **10** | MFP | Pin 9 + 10 | analoger Eingang / PT 100 |
| **11** | | | Treiberspannung für Endstufen |
| **12** | MFP | Pin 12 (n.c.) | Potentiometer 2 |
| **13** | MFP | Pin 13 (n.c.) | Potentiometer 3 |

## 4.11 Auswertung von Takt und Drehrichtungsvorgaben

Optional können Achsen statt über Vektorbefehle oder manuelle Eingabe (Joystick, Trackball, Tippbetrieb, Handrad) auch mit Taktsignalen in Abhängigkeit des Drehrichtungssignales vor oder zurück verfahren werden. Dieser Betrieb ist auch asynchron zu Verfahrvorgängen möglich, die über Fahrbefehle ausgelöst worden sind. Hierzu steht der Multifunktionsport MFP zur Verfügung.

### 4.11.1 Verfahrbereichsüberwachung

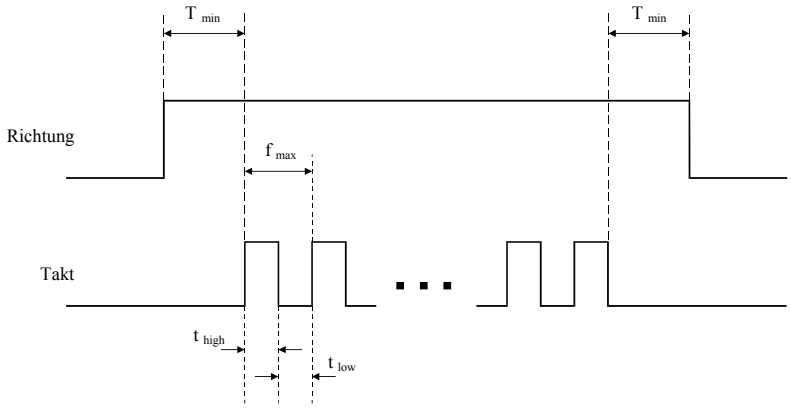
Im TVR-Betrieb wird ebenfalls überwacht, ob die zulässigen Verfahrgrenzen nicht überschritten werden. Die Verfahrgrenzen können dabei entweder über die Kombination *‚Kalibrieren'* und *‚Hub-Messen'* ermittelt worden sein. Eine andere Möglichkeit besteht darin, die Verfahrgrenzen per Befehl zu setzen.
Stellt die Steuerung fest, dass durch die aufgelaufenen Zählimpulse eine Verfahrgrenze überschritten werden würde, wird jede weitere Bewegung der Achse in dieser Richtung unterbunden. Ein Verfahren in der entgegengesetzten Richtung ist jedoch weiter möglich. Eine Meldung an den PC erfolgt nicht.

| | |
|---|---|
| **Hinweis:** | Das Anwenderprogramm ist dafür verantwortlich, dass maximale Start/Stop Frequenzen des Antriebs nicht überschritten, und die jeweilige Achse beschleunigungsmäßig nicht überlastet wird. |

### 4.11.2 Zeitliche Randbedingungen für die Signale

Die zeitliche Abfolge der Flanken von Takt- und Drehrichtungssignalen einer Achse unterliegt folgenden Randbedingungen

- frühestens $T_{min}$ nach jedem Polaritätswechsel des Drehrichtungssignals darf der nächste Taktimpuls angelegt werden.
- spätestens $T_{min}$ vor jedem Polaritätswechsel des Richtungssignales müssen die Taktimpulse ausgesetzt werden
- $T_{min}$ beträgt z. Zt. 50µs.
- die maximale Frequenz der Taktimpulse darf den Wert von $f_{max}$ = 833 kHz nicht überschreiten. Dabei müssen die Mindestzeiten $T_{low}$ = 600ns und $T_{high}$ = 600ns eingehalten werden.
- Als Schutz der Steuerungseingänge werden Eingangsfilter mit 470Ω und 220pF eingesetzt. Daher ist auf ausreichende Treiberleistung der Taktquelle zu achten.



| Takt-Vor/rück-Achsen-Zuordnung | |
|---|---|
| **Befehl:** | !tvrtoaxis oder ?tvrtoaxis |
| **Parameter:** | x y z a<br>0 = keine Achszuordnung<br>1 = QEP 1<br>2 = QEP 2 |
| **Beschreibung:** | Mit der Eingabe von tvrtoaxis lassen sich die QEP Eingänge beliebig zuordnen. Da nur zwei QEP Eingänge auf dem MFP zu Verfügung stehen, kann man diese beliebigen Achsen zuordnen.<br>!tvrtoaxis z 1 (Die Z-Achse wird über die QEP1 Eingänge verfahren.) |
| **Rückmeldung:** | Eingestellte Zuordnung. |
| **Fehlercode:** | -- |
| **Beispiel:** | !tvrtoaxis 1 2 0 0 ( X=1 Y=2 Z=0 A=0 )<br>?tvrtoaxis |
| **Aktivierung:** | sofort |

| Takt –Vor/Rück Ein/Aus | |
|---|---|
| **Befehl:** | !?tvr |
| **Parameter:** | x, y, z, a<br>0 oder 1 |
| **Beschreibung:** | !tvr x 0 Takt- Vor/Rück Aus für x-Achse<br>!tvr x 1 Takt- Vor/Rück Ein für x-Achse<br>?tvr Abfrage Zustand Takt- Vor/Rück |
| **Rückmeldung:** | O oder 1 |
| **Fehlercode:** | -- |
| **Beispiel:** | !tvr 0 1 1 0 (T-V/R für die Achsen Y + Z aktiv) |
| **Aktivierung:** | sofort |

| **Takt - Vor / Rück Modus** | |
|---|---|
| **Befehl:** | !tvrm oder ?tvrm |
| **Parameter:** | x, y, z und a<br>0, 1, 2, 3, 4 |
| **Beschreibung:** | **Funktionen:** |
| | 0 ➔ Takt Vor/Rück „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 Start/Stopp-Eingänge. |
| | 4 ➔ Kombination aus 2 & 3. |
| | **Befehle:** |
| | !tvrm 1 ➔ Normale T-V/R Bearbeitung (bei jedem Takt wird die ausgewählte Achse um ein Mikroschritt verfahren) für die X-Achse |
| | !tvrm 2 ➔ T-V/R Bearbeitung mit Faktor (bei jedem Takt wird der Wert Faktors mal 1 Mikroschritt verfahren) für die X-Achse |
| | !tvrm 3 ➔ Die Achse wird nur verfahren, wenn sie über den Start/Stopeingang frei gegeben wird. (Damit lassen sich die Achsen noch getrennt sperren, oder man kann an den Takteingang eine feste Taktfrequenz anlegen.) für die X-Achse |
| | ?tvrm ➔ Der eingestellte Modus wird angezeigt |
| **Rückmeldung:** | 0 bis 4 |
| **Fehlercode:** | -- |
| **Beispiel:** | !tvrm 4 4 4 4 (T-V/R Modus 4 ist eingestellt für alle Achsen)<br>?tvrm |
| **Aktivierung:** | !tvr 1 |

| **Faktor Takt Vor / Rück** | | |
|---|---|---|
| **Befehl:** | !tvrf oder ?tvrf | |
| **Parameter:** | x, y, z und a<br>-10000 bis + 10000<br>Mit einem negativen Faktor lässt sich die Drehrichtung ändern als Alternative zum Vor/Rück Signal | |
| **Beschreibung:** | !tvrf 1 1 0 0 | ➔ Bei den Achsen x und y soll Takt Vor/Rück mit dem Faktor 1 arbeiten [d.h. Ein Takt = Ein Motorincrement (MI)]. |
| | !tvrf a 10 | ➔ Bei Achse a soll Takt Vor/Rück mit dem Faktor 10 arbeiten (d.h. Ein Takt = 10 MI) |
| | ?tvrf | ➔ Alle eingestellten Faktoren werden angezeigt |
| | ?tvrf z | ➔ Der aktuelle Faktor z-Achse wird angezeigt |
| **Rückmeldung:** | Faktorwerte | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !tvrf 10<br>?tvrf | (Faktor = 10.00 bei der x-Achse)<br>(Ein Takt = Zehn Motorinkremente) |
| **Aktivierung:** | sofort | |

## 4.12 Auswertung von inkrementalen Messsystemen

An der Steuerung können gleichzeitig Achsen mit und ohne Geber betrieben werden. Hierzu überprüft die Steuerung während des Kalibriervorganges oder der Autokommutierung, ob Geber angeschlossen sind, sofern diese durch *enctoaxis* freigegeben wurden. Das Ergebnis dieser Überprüfung kann mit dem Befehl *?enc* überprüft werden. Die Steuerung unterscheidet hierbei allerdings nicht zwischen falsch angeschlossenem und fehlendem Geber.
Auch beim Kalibrieren auf Referenzmarke, fährt die Achse in negative Richtung in den Null-Endschalter, macht eine Richtungsumkehr und fährt mit der Geschwindigkeit die über calbspeed eingestellt wurde bis zur Referenzmarke. Hat ein System keine Endschalter (z.B. eine Drehachse) wird direkt auf die Referenzmarke kalibriert, wenn vorher alle Endschalter deaktiviert wurden.

| **Zuordnung von Gebereingang und Achse** | |
|---|---|
| **Befehl:** | enctoaxis |
| **Parameter:** | 1…6 |
| **Bemerkung:** | Die LSTEP-PCIexpress hat 6 Eingänge für QEP-Encoder, die beliebig einer Achse zugeordnet werden können.<br>Die Eingänge 1+2 können über den MFP angeschlossen werden.<br>Die Eingänge 3…6 über die Encodereingänge. |
| **Beschreibung:** | !enctoaxis 3 4 5 6 ➔ Encoder X auf Eingang 3 |
| | ➔ Encoder Y auf Eingang 4 |
| | ➔ Encoder Z auf Eingang 5 |
| | ➔ Encoder A auf Eingang 6 |
| **Rückmeldung:** | Geberzuordnung |
| **Fehlercode:** | -- |
| **Beispiel:** | !enctoaxis 1 2 (Encoder auf Eingang1, und Y auf Eingang 2)<br>?enctoaxis |
| **Aktivierung:** | sofort |

| Encoder Typ | |
|---|---|
| **Befehl:** | !enctype oder ? enctype |
| **Parameter:** | x, y, z und a<br>0 bis 8 |
| **Bemerkung:** | Vor Inbetriebnahme müssen die Encoder richtig eingestellt werden |
| **Beschreibung** | !enctype 0 bis 8<br>Rotative Encoder: 1=1Vss, 2=11µA, 3=MR, 4=TTL<br>Linear Encoder: 5=1Vss, 6=11µA, 7=MR, 8=TTL |
| **Rückmeldung:** | Encodertype |
| **Fehlercode:** | -- |
| **Beispiel:** | ?enctype |
| **Aktivierung:** | !validconfig bzw. !validpar |

| Encoder Zählrichtung | |
|---|---|
| **Befehl:** | ?encdir |
| **Parameter:** | x, y, z und a<br>0 oder 1 ( 1 = Zählrichtung tauschen ) |
| **Bemerkung:** | Bei falscher Zählrichtung kann man diese per Befehl tauschen, ohne den Encoder oder eine Phase zu drehen |
| **Beschreibung:** | !encdir 0 1 0 0 ➔ Zählrichtung für die Y-Achse gedreht. |
| | !encdir x 1 ➔ Zählrichtung für die X-Achse gedreht. |
| **Rückmeldung:** | 0 oder 1 |
| **Fehlercode:** | -- |
| **Beispiel:** | ?encdir |
| **Aktivierung:** | !validconfig bzw. !validpar |

| Gebermaske für erkannte Geber | | |
|---|---|---|
| **Befehl:** | ?enc | |
| **Parameter:** | x, y, z und a<br>0 oder 1 (On,Off) | |
| **Bemerkung:** | Für alle erkannten Geber kommt eine 1 | |
| **Beschreibung:** | ?enc | ➔ Alle Geberzustände werdenb angezeigt. |
| | ?enc x | ➔ Anzeigen der Gebermaske von Achse x. |
| **Rückmeldung:** | Geberzustand | |
| **Fehlercode:** | -- | |
| **Beispiel:** | ?enc | |

| Signalperiode / Linear Encoder | | |
|---|---|---|
| **Befehl:** | !encperiod oder ?encperiod | |
| **Parameter:** | x, y, z und a<br>Periodenlänge in der eingestellten Dimension | |
| **Beschreibung:** | !encperiod 0.5 0.020 | ➔ Die Periodenlänge des Gebersignals bei der X-Achse ist 500µm und bei y-Achse ist sie 20µm. |
| | ?encperiod | ➔ Alle Geberperiodenlängen werden angezeigt. |
| | ?encperiod x | ➔ Anzeige der Geberperiodenlänge von Achse x. |
| **Rückmeldung:** | Geberperiodenlänge in der Einstellung der Dimension | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !encperiod x 0.1 (Geberperiodenlänge der x-Achse ist 0.1mm)<br>?encperiod | |
| **Aktivierung:** | !validconfig bzw. !validpar | |

| Encoder Polpare / Drehgeber | |
|---|---|
| **Befehl:** | !encpolepairs oder ?encpolepairs |
| **Parameter:** | x, y, z, a<br>1 bis 50.000 |
| **Beschreibung:** | Gibt die Anzahl der Encodersignalperioden pro Motorumdrehung an.<br>Ist der Encoder hinter einem Getriebe montiert, sollte das Verhältnis von Perioden zum Getriebefaktor eine ganze Zahl ergeben. |
| **Rückmeldung:** | - |
| **Fehlercode:** | -- |
| **Beispiel:** | !encpolepairs 500 500 500<br>?encpolepairs |
| **Aktivierung:** | !validconfig bzw. !validpar |

| Geber-Referenzsignal | | |
|---|---|---|
| **Befehl:** | !encref oder ?encref | |
| **Parameter:** | x, y, z oder a<br>0 oder 1 | |
| **Beschreibung:** | !encref 1 1 0 | ➔ Beim Kalibrieren wird das Referenzsignal der Geber x und y ausgewertet. |
| | !encref z 1 | ➔ Beim Kalibrieren wird das Referenzsignal des Gebers der z-Achse ausgewertet. |
| | ?encref | ➔ Die aktuelle Einstellung wird angezeigt. |
| | ?encref y | ➔ Die aktuelle Einstellung der y-Achse wird angezeigt. |
| **Rückmeldung:** | 0 oder 1 | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !encref x 0 (Keine Referenzsignalauswertung der x-Achse)<br>?encref | |
| **Aktivierung:** | sofort | |

| **Polarität der Referenzmarken der QEP-Encoder** | |
|---|---|
| **Befehl:** | !encrefpol oder ?encrefpol |
| **Parameter:** | x, y, z, a<br>0 oder 1 (0 = negativer, 1 = positiver Referenzimpuls) |
| **Bemerkung:** | Mit Hilfe dieses Befehls lässt sich die Polarität der Referenzmarken der QEP-Encoder einstellen |
| **Beschreibung:** | !encrefpol 1 1 0 0 ➔ Encoder X+Y positiver Referenzimpuls<br>Encoder Z+A negativer Referenzimpuls |
| | ?encrefpol ➔ Alle Einstellungen werden angezeigt |
| | ?encrefpol x ➔ Einstellung der X-Achse wird angezeigt |
| **Rückmeldung:** | Einstellungen (z. B.: 1 1 0 0) |
| **Fehlercode:** | -- |
| **Beispiel:** | !encrefpol 1 1 (Encoder X+Y positiver Referenzimpuls)<br>?encpos |
| **Aktivierung:** | sofort |

| **Geberwertanzeige** | |
|---|---|
| **Befehl:** | !encpos oder ?encpos |
| **Parameter:** | X, y, z, a<br>0 oder 1 |
| **Beschreibung:** | !encpos 1 1 ➔ Bei der Positionsabfrage werden die Geberwerte der erkannten Geber für die Achsen X und Y angezeigt. |
| | ?encpos ➔ Die aktuellen Einstellungen für alle Achsen werden angezeigt. |
| **Rückmeldung:** | 0 oder 1 |
| **Fehlercode:** | -- |
| **Beispiel:** | !encpos 0 0 0 0 (Geberpositionsanzeige für alle Achsen „AUS„)<br>?encpos |
| **Aktivierung:** | sofort |

| Geber-Fehler | |
|---|---|
| **Befehl:** | !encerr oder ?encerr |
| **Parameter:** | X, y, z, a<br>0 oder e |
| **Beschreibung:** | !encerr 0 0 0 ➔ Clear Geberfehler-Meldungen von x-, y- und z-Achse. |
| | !encerr a 0 ➔ Clear Geberfehler-Meldung von a-Achse. |
| | ?encerr ➔ Die aktuellen Geberfehler-Meldungen aller Achsenwerden angezeigt. |
| | ?encerr z ➔ Die aktuelle Geberfehler-Meldung der z-Achse wird angezeigt. |
| **Rückmeldung:** | 0 oder e |
| **Fehlercode:** | -- |
| **Beispiel:** | !encerr 0 (Clear Geberfehler-Meldung von a-Achse)<br>?encerr |
| **Aktivierung:** | sofort |

## 4.13 Reglereinstellungen für LSTEP

| Lageregler P-Anteil | |
|---|---|
| **Befehl:** | ?poskonkp oder !posconkp |
| **Parameter:** | 0 – 4000 % |
| **Bemerkung:** | |
| **Beschreibung:** | !posconkp x 100 (P-Anteil der x-Achse wird auf 100% gesetzt) |
| **Rückmeldung:** | eingestellter Wert |
| **Fehlercode:** | |
| **Beispiel:** | !posconkp a 100 (P-Anteil der a-Achse wird auf 100% gesetzt)<br>?posconkp z (Abfrage P-Anteil der z-Achse) |
| **Aktivierung:** | !validconfig bzw. !validpar |

| Lagerregler Filter-Zeitkonstante | |
|---|---|
| **Befehl:** | !posconoutpass oder ?posconoutpass |
| **Parameter:** | 0 – 100000 µs |
| **Bemerkung:** | |
| **Beschreibung:** | !posconoutpass x 320 (Zeitkonstante des Lagereglerausgangfilter wird auf 320µs gesetzt) |
| **Rückmeldung:** | eingestellter Wert |
| **Fehlercode:** | |
| **Beispiel:** | |
| **Aktivierung:** | !validconfig bzw. !validpar |

| Lageregler Achsen-Freigabe | |
|---|---|
| **Befehl:** | !posconenable oder ?posconenable |
| **Parameter:** | 0 oder 1 |
| **Bemerkung:** | enablen oder diablen des Lagereglers |
| **Beschreibung:** | !posconenable x 1 (Freigabe für die X-Achse)<br>!posconenable 1 1 1 1 (Freigabe für alle Achsen)<br>?posconenable (Abfrage der Einstellungen) |
| **Rückmeldung:** | 0 oder 1 |
| **Fehlercode:** | |
| **Beispiel:** | !posconenable 0 1 0 (Freigabe für die Y-Achse) |
| **Aktivierung:** | !validconfig bzw. !validpar |

| Lageregler Ein- und Ausschalten | |
|---|---|
| **Befehl:** | !poscon oder ?poscon |
| **Parameter:** | 0 oder 1 |
| **Bemerkung:** | Ein- oder Ausschalten der Lageregler |
| **Beschreibung:** | !poscon 1 (Einschalten für alle enableten Achsen)<br>!poscon 0 (Ausschalten aller Lageregler) |
| **Rückmeldung:** | 0 oder 1 |
| **Fehlercode:** | |
| **Beispiel:** | ?poscon (Abfrage der Lageregler) |
| **Aktivierung:** | sofort |

| Schleppfehlerüberwachung - Bereich | |
|---|---|
| **Befehl:** | ?deviationrange oder !deviationrange |
| **Parameter:** | abhängig von der Dimension |
| **Bemerkung:** | Mit diesem Befehl wird der Bereich der Schleppfehler Überwachung eingestellt, der überschritten werden muss, damit die Überwachung auslöst. |
| **Beschreibung:** | !deviationrange x |
| **Rückmeldung:** | eingestellter Wert je nach Dimension |
| **Fehlercode:** | 5(?err) 1108(Fehlerliste) 12(?sysstat) |
| **Beispiel:** | ?deviationrange (auslesen der eingestellten Bereiche für alle Achsen) |
| **Aktivierung:** | !validconfig bzw. !validpar |

| Schleppfehlerüberwachung - Zeitfenster | |
|---|---|
| **Befehl:** | ?deviationtime oder ! deviationtime |
| **Parameter:** | 0 bis 10000 ms |
| **Bemerkung:** | Mit diesem Befehl lässt sich die Zeit einstellen, nach der der Lageregler ausgeschaltet wird bei überschreitung des Schleppfehler Bereichs. |
| **Beschreibung:** | !deviationtime x 1000 (ist der Bereich des Schleppfehlers mehr als 1000ms überschritten, wird der Regler ausgeschaltet.) |
| **Rückmeldung:** | Wert in ms |
| **Fehlercode:** | 5(?err) 1108(Fehlerliste) 12(?sysstat) |
| **Beispiel:** | ?deviationtime (auslesen der eingestellten Zeit) |
| **Aktivierung:** | !validconfig bzw. !validpar |

| Schleppfehlerüberwachung – Ein- und Ausschalten | |
|---|---|
| **Befehl:** | ?deviationcheck oder ! deviationcheck |
| **Parameter:** | 0 oder 1 |
| **Beschreibung:** | !deviationcheck x 1 (die Schleppfehlerüberwachung für die X-Achse ist aktiv) |
| **Rückmeldung:** | 0 oder 1 |
| **Fehlercode:** | |
| **Beispiel:** | ?deviationcheck (auslesen der Einstellung) |
| **Aktivierung:** | !validconfig bzw. !validpar |

| Zielfenster | | |
|---|---|---|
| **Befehl:** | !twi oder ?twi | |
| **Parameter:** | X, y, z und a<br>1 bis 25000 (Motorinkremente)<br>0.1 bis Spindelsteigung/2 (µm)<br>0.0001 bis Spindelsteigung/2 (mm) | |
| **Bemerkung:** | Ein- und Ausgabewerte sind abhängig von der Dimension. | |
| **Beschreibung:** | !twi 1.0 0.002 | ➔ Bei Achse x beträgt das Zielfenster 1mm und bei der y-Achse 2µm (bei Dim = 2). Die anderen Achsen bleiben unverändert. |
| | !twi z 0.1 | ➔ Bei der z-Achse wird das Zielfenster auf 0.1µm (bei Dim = 1) eingestellt. |
| | ?twi | ➔ Alle eingestellten Zielfenster werden angezeigt. |
| | ?twi x | ➔ Anzeigen des eingestellten Zielfenster von Achse x. |
| **Rückmeldung:** | Wirklich eingestellte Zielfenster (Rundungsfehler werden angezeigt) | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !twi 10 (Die x-Achse hat ein Zielfenster von 10 Motorincrementen (bei Dim = 0)).<br>?twi | |
| **Aktivierung:** | !validconfig bzw. !validpar | |

| **Zielfenster** | | |
|---|---|---|
| **Befehl:** | !poswindowrange oder ? poswindowrange<br>Alternative zu „twi“ | |
| **Parameter:** | X, y, z und a<br>1 bis 25000 (Motorinkremente)<br>0.1 bis Spindelsteigung/2 (µm)<br>0.0001 bis Spindelsteigung/2 (mm) | |
| **Bemerkung:** | Ein- und Ausgabewerte sind abhängig von der Dimension. | |
| **Beschreibung:** | !poswindowrange 1.0 0.002 | ➔ Bei Achse x beträgt das Zielfenster 1mm und bei der y-Achse 2µm (bei Dim = 2). Die anderen Achsen bleiben unverändert. |
| | !poswindowrange z 0.1 | ➔ Bei der z-Achse wird das Zielfenster auf 0.1µm (bei Dim = 1) eingestellt. |
| | ?poswindowrange | ➔ Alle eingestellten Zielfenster werden angezeigt. |
| | ?poswindowrange x | ➔ Anzeigen des eingestellten Zielfenster von Achse x. |
| **Rückmeldung:** | Wirklich eingestellte Zielfenster (Rundungsfehler werden angezeigt) | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !poswindowrange 10 (Die x-Achse hat ein Zielfenster von 10 Motorincrementen (bei Dim = 0)).<br>?twi | |
| **Aktivierung:** | !validconfig bzw. !validpar | |

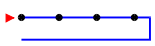
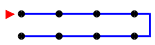
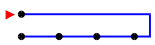
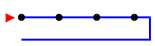
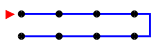
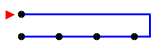
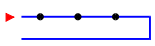
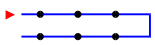
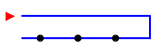
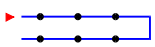
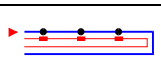
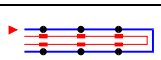
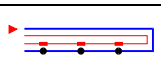
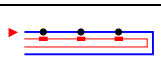
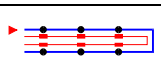
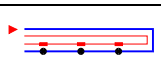
## 4.14 Konfiguration des Trigger-Ausgangssignals

Diese Befehle syncronisieren ein externes Gerät, wie z.B. eine Videokamera oder einen Laser. Diese Signale werden über den Multifunktionsport ausgegeben, welcher als Option erhältlich ist.

| **Trigger** | |
|---|---|
| **Befehl:** | ?trig oder !trig |
| **Parameter:** | 0 oder 1 (AUS / EIN) |
| **Beschreibung:** | !trig 1 ➔ Trigger „EIN“ |
| | ?trig ➔ Liefert den aktuellen Zustand der Triggerbearbeitung. |
| **Wichtig!** | Den Trigger erst einschalten, nach dem alle Einstellungen übertragen wurden. (Außer bei Triggermode 99) |
| **Rückmeldung:** | EIN oder AUS |
| **Fehlercode:** | -- |
| **Beispiel:** | !trig 0 (Triggerbearbeitung „AUS“)<br>?trig |

| **Trigger Achse** | |
|---|---|
| **Befehl:** | ?triga oder !triga |
| **Parameter:** | X, y, z oder a |
| **Beschreibung:** | !triga y ➔ Trigger bezogen auf die y-Achse. |
| | ?triga ➔ Liefert die aktuelle Bezugsachse. |
| **Rückmeldung:** | X, y, z oder a |
| **Fehlercode:** | -- |
| **Beispiel:** | !triga x (Trigger bezogen auf die x-Achse)<br>?triga |
| **Aktivierung:** | !trig 1 |

| **Trigger Modus** | |
|---|---|
| **Befehl:** | ?trigm oder !trigm |
| **Parameter:** | 0 – 17 oder 99 |

| Trigger Modus | | |
|---|---|---|
| **Beschreibung:** | !trigm 0 →  |  high active |
| | !trigm 1 →  |  high active |
| | !trigm 2 →  |  high active |
| | !trigm 3 →  |  low active |
| | !trigm 4 →  |  low active |
| | !trigm 5 →  |  low active |
| Die Trigger-Modi 6-11 erzeugen ihren ersten Trigger-Impuls nach der Hälfte der eingestellten Trigger-Strecke und dann in Abhänigkeit der Trigger-Strecke weitere Impulse. | !trigm 6 →  |  high active |
| | !trigm 7 →  |  high active |
| | !trigm 8 →  |  high active |
| | !trigm 9 →  |  low active |
| | !trigm 10 →  |  low active |
| | !trigm 11 →  |  low active |
| | !trigm 12 →  |  high active |
| | !trigm 13 →  |  high active |
| | !trigm 14 →  |  high active |
| | !trigm 15 →  |  low active |
| | !trigm 16 →  |  low active |
| | !trigm 17 →  |  low active |
| | !trigm 99 | |
| Bei dem Triggermode 99 wird am Anfang und am Ende der gleichförmigen Bewegung ein Triggerimpuls erzeugt. Bei der Ausführung dieser Funktion ist eine bestimmte Reihenfolge ein zu halten! Der Befehl !trigm 99 ist als letzter Befehl nach den gewohnten Triggereinstellungen zu senden, da er bei einer anderen Mode Einstellung gelöscht wird. | | |
| **Rückmeldung:** | 0 – 23 (Mode) | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !trigm 3 (Trigger Mode 3)<br>?trigm | |

| Legende | | | | |
|---|---|---|---|---|
| ▸ | • • • • • • | | | |
| Startpunkt | Triggerpunkte | Bahn | ExternesTriggersignal | low active |
| **Aktivierung:** | !trig 1 | | | |

| **Trigger Signal** | |
|---|---|
| **Befehl:** | ?trigs oder !trigs |
| **Parameter:** | 3 – 120 (µs) |
| **Beschreibung:** | !trigs 4 ➔ Trigger-Signallänge 4 µs |
| | ?trigs ➔ Liefert den aktuellen Zustand der eingestellten Trigger-Signallänge. |
| **Rückmeldung:** | 3 – 120 (µs) |
| **Fehlercode:** | -- |
| **Beispiel:** | !trigs 3 (Trigger-Signallänge = 3µs)<br>?trigs |
| **Aktivierung:** | !trig 1 |

| **Trigger Distanz** | |
|---|---|
| **Befehl:** | ?trigd oder !trigd |
| **Parameter:** | Abhängig von der Dimension |
| **Beschreibung:** | !trigd 1 ➔ Trigger-Distance 1mm (bei Dim 2) |
| | ?trigd ➔ Liefert die aktuelle Triggerstrecke. |
| **Rückmeldung:** | Strecke |
| **Fehlercode:** | -- |
| **Beispiel:** | !trigd 1000 (1000 MI Triggerstrecke bei Dim 0)<br>?trigd |
| **Aktivierung:** | !trig 1 |

| Trigger Counter | |
|---|---|
| **Befehl:** | ?!trigcount; |
| **Parameter:** | 0 bis 2147483647 |
| **Bemerkung:** | Es werden alle ausgegebenen Trigger gezählt |
| **Beschreibung:** | |
| **Rückmeldung:** | Anzahl der ausgeführten Trigger |
| **Fehlercode:** | -- |
| **Beispiel:** | ?trigcount ; (Lese Zählerstand) |

## 4.15 Konfiguration des Snapshot-Eingangs

Mit den Befehlen können die aktuellen Positionen während des Verfahrvorgangs in der Steuerung gespeichert werden. Diese Werte können im Anschluss ausgelesen oder angefahren werden. Dieses Signal wird über den Multifunktionsport gesetzt, welcher als Option erhältlich ist.

| Snapshot | |
|---|---|
| **Befehl:** | ?sns oder !sns |
| **Parameter:** | 0 oder 1 |
| **Beschreibung:** | !sns 1 ➔ Snapshot „EIN„ |
| | ?sns ➔ Liefert die aktuelle Snapshot-Zustand. |
| **Rückmeldung:** | Snapshot-Zustand |
| **Fehlercode:** | -- |
| **Beispiel:** | !sns 0 (Snapshot „AUS„)<br>?sns |

| Snapshot-Level (Polarität) | |
|---|---|
| **Befehl:** | ?snsl oder !snsl |
| **Parameter:** | 0 oder 1 |
| **Beschreibung:** | !snsl 1 ➔ Snapshot ist high-aktiv.  |
| | ?snsl ➔ Liefert die aktuelle Polarität |
| **Rückmeldung:** | Aktuelle Polarität |
| **Fehlercode:** | -- |
| **Beispiel:** | !snsl 0 (Snapshot ist low-aktiv)<br>?snsl  |
| **Aktivierung** | !sns 1 |

| Snapshot Filter | |
|---|---|
| **Befehl:** | ?snsf oder !snsf |
| **Parameter:** | 0 – 100 ms |
| **Bemerkung:** | Dient als Eingangsfilter bei prellenden Schaltern |
| **Beschreibung:** | !snsf 10 => 10 ms Eingangsfilter<br>?snsf => Liefert den aktuellen Wert |
| **Rückmeldung:** | Aktuelle Filterzeit |
| **Fehlercode:** | -- |
| **Beispiel:**<br><br>**Aktivierung** | !snsf 0 (Kein Eingangsfilter)<br>?snsf<br>!sns 1 |

| Snapshot-Modus | |
|---|---|
| **Befehl:** | ?snsm oder !snsm |
| **Parameter:** | 0 oder 1 |
| **Beschreibung:** | !snsm 1 ➔ Snapshot „Automatik“ Die Position wird nach dem ersten Impuls automatisch angefahren. |
| | ?snsm ➔ Liefert den aktuellen Mode |
| **Rückmeldung:** | Snapshot-Mode |
| **Fehlercode:** | -- |
| **Beispiel:** | !snsm 0 (Normaler Snapshot)<br>?snsm |
| **Aktivierung** | !sns 1 |

| Snapshot-Zähler | |
|---|---|
| **Befehl:** | ?snsc |
| **Parameter:** | – |
| **Beschreibung:** | Inhalt wird nach jedem „lesen„ gelöscht. |
| | ?snsc ➔ Liefert die Anzahl der ausgelösten SnapShot`s. |
| **Rückmeldung:** | Liefert die Anzahl der ausgelösten SnapShot`s |
| **Fehlercode:** | -- |
| **Beispiel:** | ?snsc |

| Snapshot-Position | | |
|---|---|---|
| **Befehl:** | !snsp oder ?snsp | |
| **Parameter:** | X,y,z und a<br>Min-/max-Verfahrbereich | |
| **Bemerkung:** | Ein- und Ausgabe ist abhängig von der Dimension. | |
| **Beschreibung:** | !snsp 1000 2000 3000 | ➔ Positionswerte für die Achsen x, y und z werden gesetzt. |
| | !snsp y 2000 | ➔ Position der y-Achse wird gesetzt. |
| | ?snsp | ➔ Abfrage der aktuellen Snapshot-Position aller Achsen. |
| | ?snsp z | ➔ Abfrage der aktuellen Snapshot-Position von Achse z. |
| **Rückmeldung:** | Positionswerte | |
| **Fehlercode:** | -- | |
| **Beispiel:** | !snsp 100 200 (Setzen der Positionen von x- und y-Achse)<br>?snsp (Abfrage der Snapshot-Positionen aller Achsen) | |

| Snapshot-Position-Array | | |
|---|---|---|
| **Befehl:** | ?snsa | |
| **Parameter:** | X,y,z und a<br>1 – 200 (Positionen) | |
| **Bemerkung:** | Ein- und Ausgabe ist abhängig von der Dimension. | |
| **Beschreibung:** | ?snsa 33 | ➔ Abfrage der Snapshot-Position 33 aller Achsen. |
| | ?snsa z 99 | ➔ Abfrage der Snapshot-Position 99 von Achse z. |
| **Rückmeldung:** | Positionswerte | |
| **Fehlercode:** | -- | |
| **Beispiel:** | ?snsa 1 (Abfrage der Snapshot-Positionen 1 aller Achsen) | |

# 5 Steckerbelegungen und Hardware

## 5.1 Die Pinbelegung des Multifunktionsport (ST14, 50pol Pfostenleiste auf 25pol, oder 50pol D-Sub-Buchse)

Den Multifunktionsport gibt es in zwei Ausbaustufen, als 25pol und 50pol Dsub Buchse
Bedingt durch die Funktionsvielfalt sind die Pins des Multifunktionsport **(MFP)** teilweise mehrfach belegt. Je nach Ausstattung der Steuerung bedeutet dies, dass jeweils nur ein Signalaus- bzw. Eingang auf einem Pin des MFP anliegt.
Die gewünschte Funktionalität muss bei der Bestellung mit angegeben werden.
Standart ist: Trigger, Snapshot, und Stopeingang

| 50 pol Pin | 25 pol Pin | Belegung | Signal | Bemerkungen |
|---|---|---|---|---|
| 1 | 24 | 1 | AIN-0 | Analoger Eingang/Joystick Achse 1 |
| 2 | 12 | 1 | AIN-1 | Analoger Eingang/Joystick Achse 2 |
| 3 | 25 | 1 | AIN-2 | Analoger Eingang/Joystick Achse 3 |
| 4 | - | 1 | AIN-3 | Analoger Eingang/Joystick Achse 4 |
| 5 | 8 | 1 | AIN-6 | Analoger Eingang/0…5V |
| 6 | 20 | 1 | AIN-7 | Analoger Eingang/0…5V |
| 7 | 7 | 1 | AIN-8 | Analoger Eingang/0…5V |
| 8 | 19 | 1 | AIN-9 | Analoger Eingang/0…5V |
| 9 | 6 | 1 | AIN-10 | Analoger Eingang/PT100 |
| 10 | - | 1 | AIN-10 | Analoger Eingang/PT100 |
| 11 | - | 1 | AIN-4 | Analoger Eingang/Potentiometer 1 |
| 12 | - | 1 | AIN-12 | Analoger Eingang/Potentiometer 2 |
| 13 | - | 1 | AIN-13 | Analoger Eingang/Potentiometer 3 |
| 14 | 18 | 1 | AOUT-0 | Analoger Ausgang/0…10V oder± 10V |
| 15 | - | 1 | AOUT-1 | Analoger Ausgang/0…10V oder± 10V |
| 16 | - | 1<br>2 | Takt-Ausgang A<br>TTL-Ausgang | |
| 17 | - | 1<br>2 | V/R-Ausgang A<br>TTL-Ausgang | |
| 18 | - | 1<br>2 | Takt-Ausgang B<br>TTL-Ausgang | |
| 19 | - | 1<br>2 | V/R-Ausgang B<br>TTL-Ausgang | |
| 20 | 1 | 1<br>2<br>3 | Takt-Eingang A<br>Geber A/Spur A<br>TTL-Eingang | Für: Handrad/Trackball/Geber |
| 21 | 2 | 1<br>2<br>3 | V/R-Eingang A<br>Geber A/Spur B<br>TTL-Eingang | Für: Handrad/Trackball/Geber |
| 22 | 5 | 1 | Geber A/Index | |
| 23 | 3 | 1<br>2<br>3 | Takt-Eingang B<br>Geber B/Spur A<br>TTL-Eingang | Für: Handrad/Trackball/Geber |
| 24 | 4 | 1<br>2 | V/R-Eingang B<br>Geber B/Spur B | Für: Handrad/Trackball/Geber |

| 50 pol Pin | 25 pol Pin | Belegung | Signal | Bemerkungen |
|---|---|---|---|---|
| | | 3 | TTL-Eingang | |
| 25 | 14 | 1 | Geber B/Index | |
| 26 | 15 | | Triggerout 1 | |
| 27 | - | | Triggerout 2 | |
| 28 | 22 | | Snapshot-Eingang | |
| 29 | 23 | | Stopp-Eingang | |
| 30 | 10 | | Joystick Ein | |
| 31 | - | | n.c. | |
| 32 | - | | n.c. | |
| 33 | - | | Treiberspannung | Zum Stillsetzen der Endstufentreiber |
| 34 | - | | Treiberrelais COM | Rückmeldung gemeinsamer Kontakt |
| 35 | - | | Treiberrelais NO | Rückmeldung Endstufe aktiv |
| 36 | - | | Treiberrelais NC | Rückmeldung Endstufe stillgesetzt |
| 37 | - | | Motorbremse A+ | |
| 38 | - | | Motorbremse A- | |
| 39 | - | | Motorbremse B+ | |
| 40 | - | | Motorbremse B- | |
| 41 | 13 | | +3,0Vref | |
| 42 | - | | +3,3V | |
| 43 | 17 | | +5,0V | |
| 44 | - | | +5,0Vanalog | |
| 45 | 21 | | +12V | |
| 46 | 9 | | -12V | |
| 47 | 11/16 | | GND | |
| 48 | 11/16 | | GND | |
| 49 | 11/16 | | GND | |
| 50 | 11/16 | | GND | |

## 5.2 MFP Adapter für zwei LSTEP-PCIexpress

| 25pol / ST1 | Funktion | 50pol / ST2 Karte 1 | 50pol / ST3 Karte 2 |
|---|---|---|---|
| 1 | Stopp Eingang Karte 1 | 29 | |
| 14 | Stopp Eingang Karte 2 | | 29 |
| 2 | Treiberspannung Karte 1 | 33 | |
| 15 | Treiberspannung Karte 2 | | 33 |
| 3 | Treiberrelais COM Karte 1 | 34 | |
| 4 | Treiberrelais NO Karte 1 | 35 | |
| 5 | Treiberrelais NC Karte 1 | 36 | |
| 16 | Treiberrelais COM Karte 2 | | 34 |
| 17 | Treiberrelais NO Karte 2 | | 35 |
| 18 | Treiberrelais NC Karte 2 | | 36 |
| 6 | GND Karte 1 | 47…50 | |
| 19 | GND Karte 2 | | 47…50 |
| 7 | +5V Karte 1 | 43 | |
| 20 | +5V Karte 2 | | 43 |
| 8 | +12V Karte 1 | 45 | |
| 21 | +12V Karte 2 | | 45 |
| 9 | Triggerout 1, Karte 1 | 26 | |
| 22 | Triggerout 1, Karte 2 | | 26 |
| 10 | Snapshot Karte 1 | 28 | |
| 23 | Snapshot Karte 2 | | 28 |

## 5.3 Die Pinbelegung der RS232 Schnittstelle

| Pin | Signal | Bemerkungen |
|---|---|---|
| 1 | n.c. | |
| 2 | RxD | Empfängerleitung LSTEP |
| 3 | TxD | Sendeleitung LSTEP |
| 4 | GND | |
| 5 | GND | Signalmasse |
| 6 | +5V | |
| 7 | RTS | Request to send, von LSTEP |
| 8 | CTS | Clear to send, von PC |
| 9 | entweder n.c. +5V od. +12V DC | |

## 5.4 Das RS232 - Schnittstellenkabel

| LSTEP | | PC | | |
|---|---|---|---|---|
| 9 Pol Sub-D Stecker | Belegung | 9 Pol Sub-D | 25 pol Sub-D | Belegung |
| 1 | n.c. | - | - | - |
| 2 | RxD | 3 | 2 | TxD |
| 3 | TxD | 2 | 3 | RxD |
| 4 | n.c. | - | - | - |
| 5 | GND | 5 | 7 | GND |
| 6 | n.c. | - | - | - |
| 7 | RTS | 8 | 5 | CTS |
| 8 | CTS | 7 | 4 | RTS |
| 9 | n.c. | - | - | - |

## 5.5 Die Pinbelegung der USB Schnittstelle (Steckertyp B)

| Pin | Signal | Bemerkungen |
|---|---|---|
| 1 | Vin (+5V) | |
| 2 | USB-Slave D- | |
| 3 | USB-Slave D+ | |
| 4 | GND | |

## 5.6 Die CAN Schnittstelle (ST4, 10pol Pfostenleiste auf 9pol D-Sub)

Achtung! Zur Zeit wird noch kein CAN-Protokoll unterstützt.

| Pin | Belegung | Pin-Nr. | Belegung |
|---|---|---|---|
| 1 | n.c. | 6 | CAN GND |
| 2 | CAN L | 7 | CAN H |
| 3 | CAN GND | 8 | n.c. |
| 4 | n.c. | 9 | CAN V+ (J2 gesteckt: +12V) |
| 5 | CAN Schirm (GND) | 10 | n.c. |

## 5.7 Spannungsversorgung 12V (ST10, 4pol PC-Netzteilstecker)

| Pin | Signal | Bemerkungen |
|---|---|---|
| 1 | +12Vin | |
| 2 | GND | |
| 3 | GND | |
| 4 | +5Vin | |

## 5.8 Motorspannungsversorgung bis 48V (ST17, 6pol Tyco Printstecker)

| Pin | Signal | Bemerkungen |
|---|---|---|
| 1 | + UMOT | Motospannung >24V bis 48V |
| 2 | + UMOT | Motospannung >24V bis 48V |
| 3 | + 24V | Für digitale I/O |
| 4 | - UMOT | GND / Motospannung >24V bis 48V |
| 5 | - UMOT | GND / Motospannung >24V bis 48V |
| 6 | 0V (+24V) | GND für digitale I/O |

## 5.9 Joystick Anschluss (ST1, 9pol D-Sub-Stecker)

| Pin | Signal | Bemerkungen |
|---|---|---|
| 1 | GND | Ground |
| 2 | Achse 4 | Analogeingang für 4.Achse |
| 3 | Achse 1 | Analogeingang für 1.Achse |
| 4 | Achse 2 | Analogeingang für 2.Achse |
| 5 | Achse 3 | Analogeingang für 3.Achse |
| 6 | Snap-Shot | Snap-Shot Eingang parallel zum Pin22/25pol oder Pin28/50pol des MFP |
| 7 | /Stop | Stop-Eingang parallel zum Pin23/25pol, oder Pin29/50pol des MFP |
| 8 | +5V analog | 5V Analoge Referenzspannung |
| 9 | +5V analog | 5V Analoge Referenzspannung |

## 5.10 Endschalter Eingänge (ST5, 16pol Pfostenleiste mit 15pol D-Sub-Belegung, für separaten Anschluss der Endschalter)

| Pin | Belegung | Pin | Belegung |
|---|---|---|---|
| 1 | Nullpunktschalter Achse 1 | 9 | +5V |
| 2 | Endlagenschalter Achse 1 | 10 | +5V |
| 3 | Nullpunktschalter Achse 2 | 11 | +12V |
| 4 | Endlagenschalter Achse 2 | 12 | +12V |
| 5 | Nullpunktschalter Achse 3 | 13 | GND |
| 6 | Endlagenschalter Achse 3 | 14 | GND |
| 7 | Nullpunktschalter Achse 4 | 15 | GND |
| 8 | Endlagenschalter Achse 4 | Gehäuse | Schirm |

## 5.11 Analog I/O: (ST 7, 10-pol Pfostenleiste mit 9pol D-Sub-Belegung)

| Pin | Belegung | Bemerkung |
|---|---|---|
| 1 | Analog IN Channel 6 | Analog: 0...5V, 4,7kOhm gegen +5V, RC-Filter 10KOhm/100nF (=St11,Pin8) |
| 2 | Analog IN Channel 7 | Analog: 0...5V, 4,7kOhm gegen +5V, RC-Filter 10KOhm/100nF (=St11,Pin20) |
| 3 | Analog IN Channel 8 | Analog: 0...5V, 4,7kOhm gegen +5V, RC-Filter 10KOhm/100nF (=St11, Pin7) |
| 4 | Analog IN Channel 9 | Analog: 0...5V, 4,7kOhm gegen +5V, RC-Filter 10KOhm/100nF (=St11,Pin19) |
| 5 | Analog Out Channel 0 | 0...10V oder +/-10V, R,Last >=1kOhm Ri= ca. 100 Ohm (=St11,Pin18) |
| 6, 7 | PT 100 Channel 10 Temperaturfühleranschluss | Messstrom = 10 mA, LB 5 muss geschlossen sein, St11, Pin6 ist nicht verwendbar) |
| 8 | GND | Ground |
| 9 | VAREF = +5V / 1A | Ausgang |
| 10 | Analog OUT Channel 1 | 0…10V oder +/- 10V |

## 5.12 TTL Gebereingänge: (St6, 16-pol Pfostenleiste D-Sub-Zählweise)

| Pin | Bezeichnung | Funktion |
|---|---|---|
| 1 | A 1 | Incrementalgeber 1, Spur A |
| 2 | B 1 | Incrementalgeber 1, Spur B |
| 3 | REF 1 | Incrementalgeber 1, Spur Z (Referenzsignal) |
| 4 | A 2 | Incrementalgeber 2, Spur A |
| 5 | B 2 | Incrementalgeber 2, Spur B |
| 6 | REF 2 | Incrementalgeber 2, Spur Z (Referenzsignal) |
| 7 | A 3 | Incrementalgeber 3, Spur A |
| 8 | B 3 | Incrementalgeber 3, Spur B |
| 9 | REF 3 | Incrementalgeber 3, Spur Z (Referenzsignal) |
| 10 | A4 | Incrementalgeber 4, Spur A |
| 11 | B 4 | Incrementalgeber 4, Spur B |
| 12 | REF 4 | Incrementalgeber 4, Spur Z (Referenzsignal) |
| 13 | GND | Ground |
| 14 | +5V | |
| 15 | +12V | |
| 16 | nc | |

## 5.13 Umsetzer für TTL-Gebereingänge: (16pol Pfostenleiste auf 3(4) x 9pol Buchse)

| Pin | 9pol Buchse 1 - 4 | Bezeichnung / Funktion |
|---|---|---|
| 1 | Achse 1, Pin 6 | A1 / Incrementalgeber 1, Spur A |
| 2 | Achse 1, Pin 8 | B1 / Incrementalgeber 1, Spur B |
| 3 | Achse 1, Pin 9 | REF1 / Incrementalgeber 1, Spur Z (Referenzsignal) |
| 4 | Achse 2, Pin 6 | A2 / Incrementalgeber 2, Spur A |
| 5 | Achse 2, Pin 8 | B2 / Incrementalgeber 2, Spur B |
| 6 | Achse 2, Pin 9 | REF2 / Incrementalgeber 2, Spur Z (Referenzsignal) |
| 7 | Achse 3, Pin 6 | A3 / Incrementalgeber 3, Spur A |
| 8 | Achse 3, Pin 8 | B3 / Incrementalgeber 3, Spur B |
| 9 | Achse 3, Pin 9 | REF3 / Incrementalgeber 3, Spur Z (Referenzsignal) |
| 10 | Achse 4, Pin 6 | A4 / Incrementalgeber 4, Spur A |
| 11 | Achse 4, Pin 8 | B4 / Incrementalgeber 4, Spur B |
| 12 | Achse 4, Pin 9 | REF4 / Incrementalgeber 4, Spur Z (Referenzsignal) |
| 13 | Achse 1 - 4, Pin 2 | GND / Ground |
| 14 | Achse 1 – 4, Pin 7 | +5V |
| 15 | Achse 1 – 4, Pin 4 | +12V |
| 16 | | n.c. |

## 5.14 Motorstecker Achsen 1 – 3 mit Endschalter (ST2 25pol Dsub Buchse)

| Pin | Belegung |
|---|---|
| 1 | Achse 1, Sinus + |
| 2 | Achse 1, Sinus - |
| 3 | Achse 1, Cos + |
| 4 | Achse 1, Cos - |
| 5 | Achse 2, Sinus + |
| 6 | Achse 2, Sinus - |
| 7 | Achse 2, Cos + |
| 8 | Achse 2, Cos - |
| 9 | Nullpunktschalter Achse 1 |
| 10 | Endlagenschalter Achse 1 |
| 11 | Motorbremse A+(X) |
| 12 | Motorbremse A- (Y) |
| 13 | Motorbremse B+ (Z) |
| 14 | Achse 3, Sinus + |
| 15 | Achse 3, Sinus - |
| 16 | Achse 3, Cos + |
| 17 | Achse 3, Cos - |
| 18 | Nullpunktschalter Achse 2 |
| 19 | Endlagenschalter Achse 2 |
| 20 | Nullpunktschalter Achse 3 |
| 21 | Endlagenschalter Achse 3 |
| 22 | +5V |
| 23 | +12V |
| 24 | GND |
| 25 | GND |
| Gehäuse | Schirm |

## 5.15 Motorstecker Achse 4 (ST1 oder ST2 auf der Optionskarte)

| 15pol DSub Buchse ST 1 | 5pol DSub Buchse ST2 (bis 10A) | Funktion |
|---|---|---|
| 1 | 2 | Sin + |
| 2 | 1 | Sin - |
| 3 | 5 | Cos + |
| 4 | 4 | Cos - |
| 5 | | Endlagenschalter |
| 6 | | Nullpunktschalter |
| 7 | | + 5V |
| 8 | | GND |
| 9 | | Sin + |
| 10 | | Sin - |
| 11 | | Cos + |
| 12 | | Cos - |
| 13 | | Motorbremse B+ (Z) |
| 14 | | Motorbremse B - (A) |
| 15 | | + 12V |

## 5.16 Digitale I/O (ST11 40pol Pfostenleiste Dsub-Zählweisw)

| Pin | Belegung | Pin | Belegung |
|---|---|---|---|
| 1 | Ausgang 1 | 20 | Eingang 1 |
| 2 | Ausgang 2 | 21 | Eingang 2 |
| 3 | Ausgang 3 | 22 | Eingang 3 |
| 4 | Ausgang 4 | 23 | Eingang 4 |
| 5 | Ausgang 5 | 24 | Eingang 5 |
| 6 | Ausgang 6 | 25 | Eingang 6 |
| 7 | Ausgang 7 | 26 | Eingang 7 |
| 8 | Ausgang 8 | 27 | Eingang 8 |
| 9 | Ausgang 9 | 28 | Eingang 9 |
| 10 | Ausgang 10 | 29 | Eingang 10 |
| 11 | Ausgang 11 | 30 | Eingang 11 |
| 12 | Ausgang 12 | 31 | Eingang 12 |
| 13 | Ausgang 13 | 32 | Eingang 13 |
| 14 | Ausgang 14 | 33 | Eingang 14 |
| 15 | Ausgang 15 | 34 | Eingang 15 |
| 16 | Ausgang 16 | 35 | Eingang 16 |
| 17 | GND | 36 | GND |
| 18 | GND | 37 | +24V |
| 19 | GND | 38 - 40 | +24V |

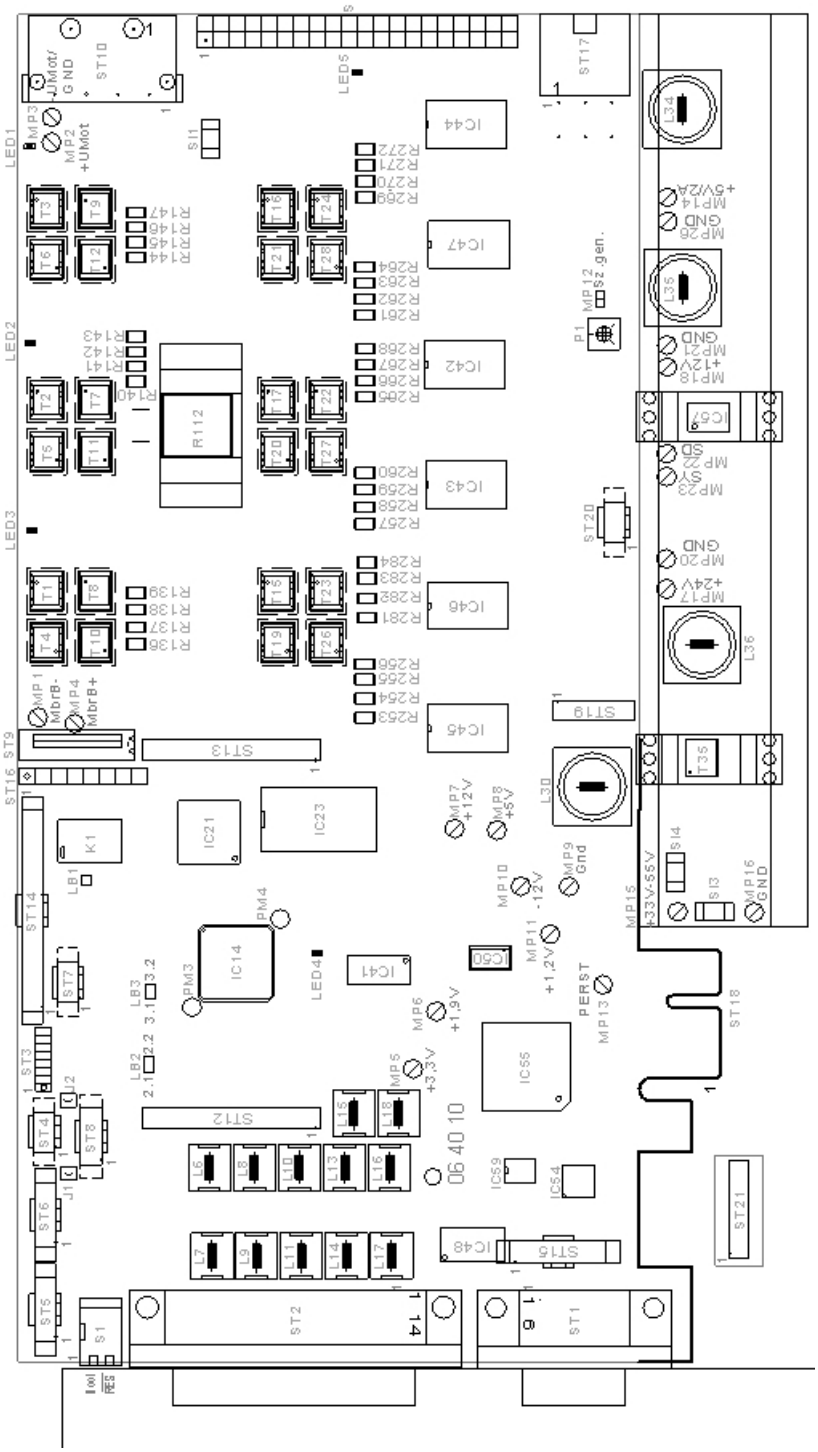
## 5.17 Technische Daten

| Allgemein | |
|---|---|
| Max. Motordrehzahl: | 70 U/sec. bei 200-schrittigem Motor |
| Max. Motorstrom: Achse 1…3<br>Max. Motorstrom Achse 4<br>(Mischbestückung ist möglich) | 0,5A; 1,25A; 2,5A; 3,75A; 5A<br>0,5A; 1,25A; 2,5A; 3,75A; 5A; max. 10A |
| Schrittauflösung | max. 1.600.000 Schritte/Umdrehung bei 200schrittigem Motor |
| Baudrate: | Bis 921.600 Kbd |
| max. Zählfrequenz für TTL-Gebereingänge | 16 Mflanken = 4 MHz für Quep 1 + 2<br>13 Mflanken = 3,25 MHz für Quep 3 bis 6 |
| Max. Motordrehzahl: | 70 U/sec. bei 200-schrittigem Motor |

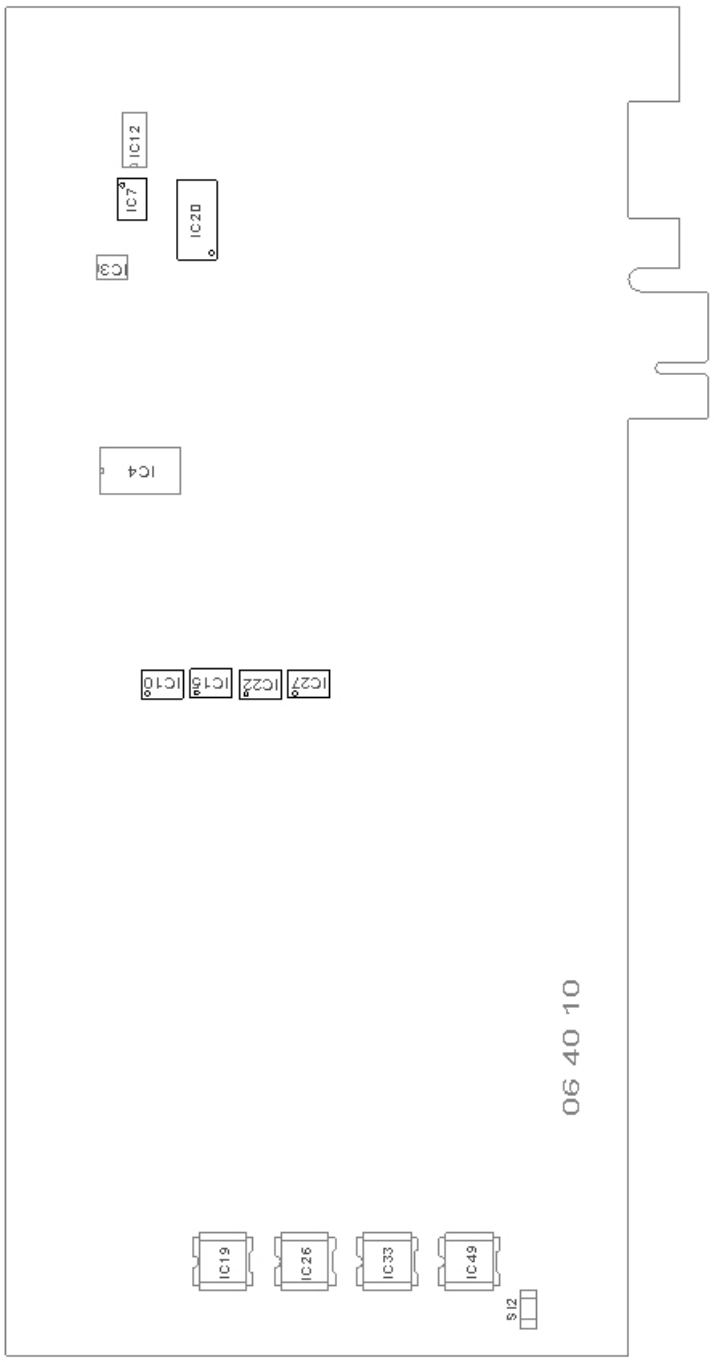
| LSTEP-PCIexpress | |
|---|---|
| Spannungsversorgung: | Logikspannung: PCIexpress-Slot vom PC und 4pol Netzteilstecker (12V)<br>Motorspannung: 2V vom PC-Netzteil oder 24 …48V über ext. Netzteil |
| Motorspannung: | 12V…48V |
| Abmessungen L x H x B (1 Slot) | 214mm x 107 x 15mm (1 Slot) |

| LSTEP-express | |
|---|---|
| Netzanschluss: | 90…264V (intern 48V) |
| Sicherungen:<br>- primär (in Eurobuchse):<br>- sekundär (auf der Platine): | • 2 A träge / 1 A träge LSTEP-1 und LSTEP-2<br>• 5 A träge / 2,25 A träge LSTEP-3<br>• Si1 LSTEP-1 und 2 ➔ 5 A träge / LSTEP-3 ➔10A träge |
| Max. Netzausfalldauer: | < 50ms<br>bei Netzausfall (<0,77 * $U_N$) schaltet die LSTEP auf Reset |
| Motorspannung: | 48V |
| Umgebungsbedingungen:<br>Lufttemperatur bei Betrieb:<br>Lufttemperatur außer Betrieb:<br>Luftfeuchtigkeit bei Betrieb<br>Luftfeuchtigkeit außer Betrieb | <br>15 ... 40 Grad C<br>0 ... 43 Grad C<br>8 ... 80 % bei 31° / Maximal 50% bei 40°<br>0 ... 80 % |
| Abmessungen B * T * H (ohne Tragegriff):<br>- Standard-Gehäuse:<br>- 19“ Tisch-Gehäuse: | <br>270 mm • 230 mm • 100 mm bei LSTEP-1x<br>482,6 mm (19“) • 375,5 mm • 90 mm (2HE) bei LSTEP-3x |
| Gewicht:<br>- Standard-Gehäuse:<br>- 19“ Tisch-Gehäuse: | <br>3,4 kg<br>5 kg |

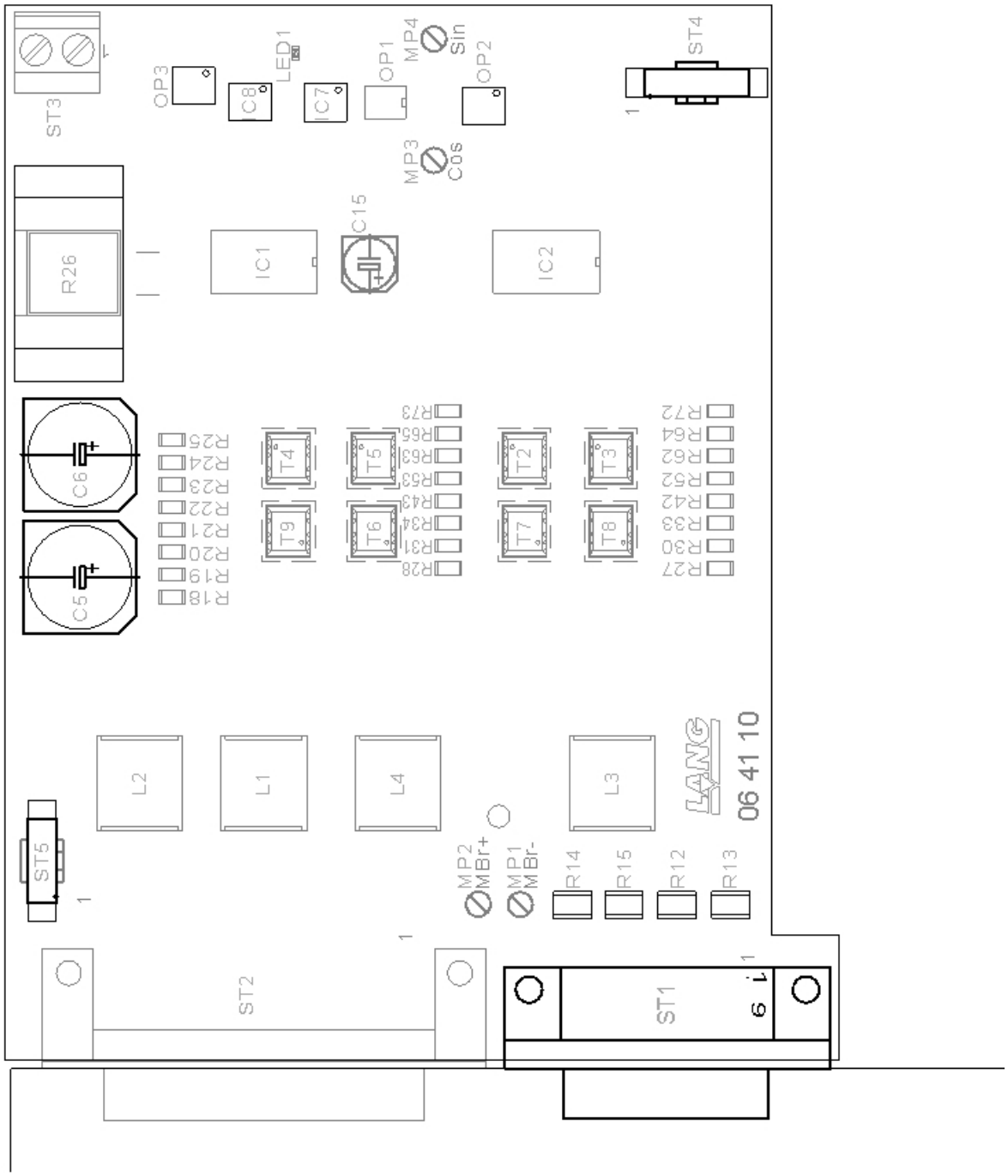
## Bestückungsplan der LSTEPexpress / LSTEP-PCIexpress



**Bestückungsplan der LSTEPexpress / LSTEP-PCIexpress** (Rückseite)



**Bestückungsplan 4. Achse (Optionskarte)**



# 6 Anhang LSTEP-API

## 6.1 Einführung

Das 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 hardware-naher Programmierung beschäftigen zu müssen. Es bietet Zugriff auf den kompletten Befehlssatz der LSTEP-Positioniersysteme.
Für neue Entwicklungen sollte man nur noch die Lstep4XDLL verwenden, die parallele Ansteuerung mehrerer LSTEPs ermöglicht. Die Lstep4DLL ist jetzt schon für Anwendungen unter LabVIEW  nicht mehr verfügbar.

### 6.1.1 Funktionsumfang

- Windows 32-bit DLL
- Unterstützung der Schrittmotorsteuerungen LSTEP xx, LSTEP xx/2, LSTEP-PC, ECO-STEP, LSTEP-44, LSTEP-PCI, LSTEP-PCIexpress, und LSTEPexpress
- Ansteuerung über RS232, USB,  ISA, PCI (DPRAM), oder PCIexpress
- Automatische Erkennung der angeschlossenen Steuerung
- Konfiguration der Steuerung
- Ausführung aller von der Steuerung unterstützten Befehle
- Bis zu 4 Achsen
- Multithreading-fähig

### 6.1.2 Systemanforderungen

Mit dem LSTEP-API, ebenso wie mit dem LSTEP4X-API können auf Intel-PCs unter MS Windows 9x, Windows NT, Windows 2000, Windows Vista, Windows XP, Windows 7 Anwendungen entwickelt werden..

### 6.1.3 unterstützte Entwicklungsumgebungen

LSTEP-API und LSTEP4X-API wurden mit den folgenden Entwicklungs- und Laufzeitumgebungen getestet:

Borland/Inprise Delphi 3-5

Microsoft Visual C++ 6.0

National Instruments LabVIEW

Sie sollten kompatibel mit allen anderen Programmier-Umgegebungen 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.)

## 6.2 DLL-Schnittstelle

### 6.2.1 LSTEP-API

Hauptbestandteil des LSTEP-APIs ist die Datei LSTEP4.DLL. Diese DLL verwenden Sie bei der Entwicklung eigener Programme, um eine LSTEP zu konfigurieren, Befehle zu senden, Positionswerte, Ein-/Ausgänge abzufragen etc. **Bitte verwenden Sie für Neuentwicklungen nur noch die LStep4XDLL.**

### 6.2.2 LSTEP4X-API

Hauptbestandteil des LStep4X-APIs ist die Datei LSTEP4X.DLL. Diese DLL verwenden Sie bei der Entwicklung eigener Programme, um eine oder mehrere LSTEPs zu konfigurieren, Befehle zu senden, Positionswerte, Ein-/Ausgänge abzufragen etc.

### 6.2.3 Allgemeine Hinweise

#### 6.2.3.1 LSTEP4.DLL

Die DLL LSTEP4.DLL implementiert die Befehle des LSTEP-API. Alle Funktionen sind mit einem 32-Bit Integer als Rückgabewert deklariert. Eine 0 als Rückgabewert zeigt die fehlerfreie Ausführung der Funktion an, bei Fehlern (z. B. Timeouts) wird der entsprechende Fehlercode (siehe Tabelle) zurückgeliefert.

Bei Funktionen wie LS_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.

#### 6.2.3.2 LSTEP4X.DLL

Die DLL LSTEP4X.DLL implementiert die Befehle des LSTEP4X-API. Alle Funktionen sind mit einem 32-Bit Integer als Rückgabewert deklariert. Eine 0 als Rückgabewert zeigt die fehlerfreie Ausführung der Funktion an, bei Fehlern (z. B. Timeouts) wird der entsprechende Fehlercode (siehe Tabelle) zurückgeliefert.

Als erster Parameter wird bei sämtlichen Funktionen des 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 kann 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.

#### 6.2.3.3 Unterschiede im Vergleich zur LSTEP4.DLL

Am Funktionsumfang hat sich bei der LSTEP4X.DLL im Vergleich zur LSTEP4.DLL nichts geändert, die Funktionsnamen sind identisch. Das normale LStep API (LSTEP4.DLL) wird weitergeführt, bestehender Quellcode, welcher das LStep API nutzt muss also nicht modifiziert werden.

Das LSTEP4X API öffnet für jede LStep ein eigenes Protokollfenster, und die Log-Dateien werden ebenfalls separat für jede LStep geschrieben.

Da das LSTEP4X API Multi-Threading unterstützt, können vom Kunden erstellte Programme von mehreren Threads aus über das API auf die LSteps zugreifen.

Die parallele Ansteuerung mehrerer LSTEP-Schrittmotorsteuerungen ist möglich.

Die Funktionsnamen haben geänderte Prefixe erhalten. Bei der LSTEP4X.DLL werden „LSX_“ anstatt „LS_“ wie bei der LSTEP4.DLL verwendet.

Bei allen Funktionsaufrufen des LSTEP4X API wird als zusätzlicher Parameter ein Integer-Wert übergeben, welcher die Steuerung bezeichnet. Die Numerierung der LSTEPs erfolgt von 1 bis 32.

| | |
|---|---|
| Hinweis: | Unter Windows NT ist die Installation des mitgelieferten Treibers GIVEIO erforderlich, damit die DPRAM-Schnittstellen der LSTEP-PCs verwendet werden können. |

### 6.2.4 Einbindung in Delphi

#### 6.2.4.1 LSTEP4-API

Allen Funktionsnamen des LSTEP4-API ist zur besseren Unterscheidung ein „LS_“ vorangestellt. Um die Funktionen des LSTEP4 APIs verwenden zu können, muss LSTEP4.pas in der uses-Klausel der entsprechenden Unit stehen, und in einem der eingestellten Suchpfade liegen.

benötigte Dateien: LSTEP4.DLL und LSTEP4.pas

Delphi-Beispiel für die Ansteuerung von einer LStep

```
...
var LStep1: Integer;
…
begin

LSX_ConnectSimple(1, 'COM1', 9600, True);

LSX_MoveAbs(10.0, 20.0, 30.0, 0.0, True);

LSX_Disconnect();

end;
```

#### 6.2.4.2 LSTEP4X API

Allen Funktionsnamen des LStep4X-API ist zur besseren Unterscheidung ein „LSX_“ vorangestellt. (siehe LStep4x.pas) Um die Funktionen des LSTEP4X APIs verwenden zu können, muss LSTEP4X.pas in der uses-Klausel der entsprechenden Unit stehen, und in einem der eingestellten Suchpfade liegen.

benötigte Dateien: LSTEP4X.DLL und LSTEP4X.pas

Delphi-Beispiel für die parallele Ansteuerung von 2 LSteps

```
...
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;
```

### 6.2.5 Einbindung in Visual C++

#### 6.2.5.1 LSTEP4-API

Für Visual C++ wurde eine Kapselung der LSTEP4.DLL erstellt. Die Klasse CLStep4 lädt die DLL und alle Zeiger auf Funktionsaufrufe dynamisch. Den Methoden des LSTEP-Objekts ist kein „LS_“ vorangestellt.
(Beispiel: LS.Calibrate() statt LS_Calibrate)
Von der Klasse CLStep4 sollte nur eine Instanz erstellt werden, zumal mit dem LSTEP API momentan nicht mehrere LSteps gleichzeitig gesteuert werden können.

benötigte Dateien: LSTEP4.DLL, LSTEP4.h und LSTEP4.cpp

Visual C++-Beispiel für die Ansteuerung von einer LStep

```
...
CLStep4 LS1;
…

LS1.ConnectSimple(1, "COM1", 9600, true);

LS1.MoveAbs(10.0, 20.0, 30.0, 0.0, true);

LS1.Disconnect();
delete LS1;
```

#### 6.2.5.2 LSTEP4X-API

Für Visual C++ wurde eine Kapselung der LSTEP4X.DLL erstellt. Die Klasse CLStep4X 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 nicht aufrufen, da die Wrapper-Klasse CLStep4X 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.

benötigte Dateien: LSTEP4X.DLL, LSTEP4X.h und LSTEP4X.cpp

Visual C++-Beispiel für die parallele Ansteuerung von 2 LSteps

```
...
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;
```

### 6.2.6 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 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 (LSTEP4.DLL und LSTEP4X.DLL) wurden VI-Bibliotheken (LSTEP4.LLB bzw LSTEP4X.LLB) erstellt, die jeweils ca. 110 VIs enthalten. Diese einzelnen VIs (z.B. LS4 ConnectSimple.vi) kapseln die entsprechenden LSTEP API-Funktionen. Die LSTEP4.DLL bzw. LSTEP4X.DLL wird mittels ‚Call Library Function‘ (Aufruf ext. Bibliotheken) verwendet.

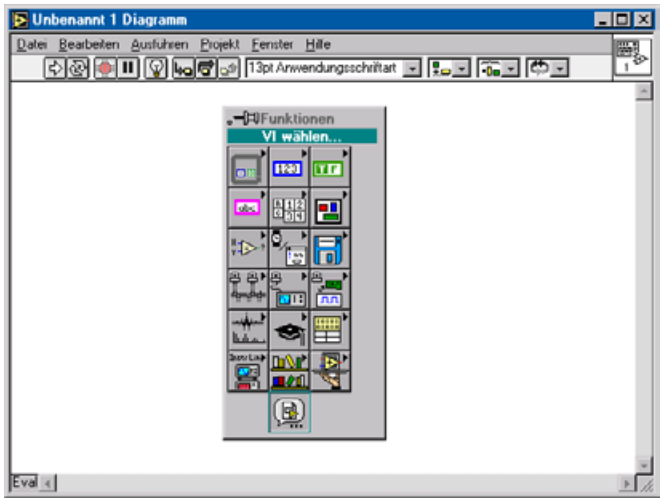
### 6.2.6.1 Unterschiede zwischen LSTEP4.LLB und LSTEP4X.LLB

Bei neuen Software-Projekten mit dem LSTEP-API unter LabView sollten Sie ausschließlich die VI-Bibliothek LSTEP4X.LLB verwenden. Dies hat folgenden Grund: Die LSTEP4.LLB unterstützt nur maximale eine LStep, ausserdem haben hier alle VI's als Rückgabewert nur eine Boolesche Variable, also keinen interpretierbaren Fehlercode.

In der (neueren) LSTEP4X.LLB haben nun die VI's als Rückgabewert einen 32-bit-Integer-Wert (Dieser ist als „error out" bezeichnet). Ist dieser gleich 0, so signalisiert dies daß die LSTEP-API-Funktion fehlerfrei ausgeführt wurde.Anderenfalls kann die Bedeutung des Fehlercodes der Tabelle in dieser Dokumentation entnommen werden. Ausserdem können durch die in den VI's aufgerufene LSTEP4X.DLL mehrere Lsteps parallel angesteuert werden. Die VI's für die LSTEP4X.DLL unterscheiden sich im Dateinamen von denen der LSTEP4.DLL dadurch, daß sie mit dem Kürzel „LS4X" statt „LS4" beginnen. In LabView haben die VI's für die LSTEP4X.DLL die Hintergrundfarbe lila, die der LSTEP4.DLL die Hintergrundfarbe blau. Die Benennung der VI's in den Symbolen ist dieselbe. Bei allen VI's kommt ein zusätzlicher Anschluß hinzu. Dieser ist ein 32-bit-Integer-Wert und gibt die Nummer der LStep an, auf die sich der Befehl, z.B. ein Verfahrbefehl oder das Auslesen der aktuellen Position, bezieht („LStep Controller ID"). Diese Nummern können Sie entweder selbst vergeben (z.B. „0" für die LStep an der seriellen Schnittstelle COM1, „1" für die LStep an COM2 etc.), oder über das VI „LS4X CreateLSID" erzeugen lassen. Mit dem VI „LS4X FreeLSID" können erzeugte LStep ID-Nummern wieder „freigegeben" werden. Wenn Sie in Ihrem LabView-Projekt nur eine LStep verwenden, können Sie den Anschluß „LStep Controller ID" bei allen verwendeten VI's aus LSTEP4X.LLB freilassen, denn dieser hat einheitlich den Default-Wert 1.benötigte Dateien in LabView: LSTEP4X.DLL und LSTEP4X.LLB
bzw. LSTEP4.DLL und LSTEP4.LLB

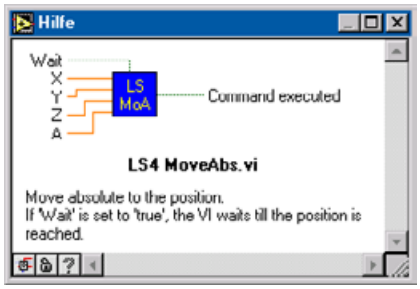
### 6.2.6.2 Vorgehensweise zur Verwendung eines LSTEP4-VI's

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

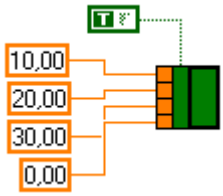



4. VI wählen...
5. Im Dateidialog die mitgelieferte VI-Bibliothek LSTEP4.LLB öffnen, daraus dann den gewünschten Befehl wählen (z.B. LS4 MoveAbs.vi)
6. VI im Diagramm plazieren

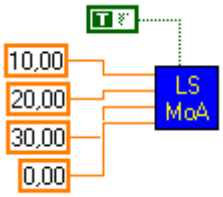

Die Tastenkombination Strg+H öffnet ein Hilfe-Fenster, dieses zeigt Informationen über das VI an, auf dem sich momentan der Mauszeiger befindet




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.

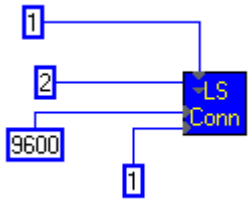



Die Belegung der Anschlüsse der VIs kann der Dokumentation der entsprechenden API-Funktion entnommen werden, dort findet sich die graphische Darstellung, welche auch im Hilfe-Fenster von LabVIEW erscheint. Die Parameter entsprechen weitgehend denen der DLL-Funktion: Lediglich bei Funktionen, denen Bitmasken 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):

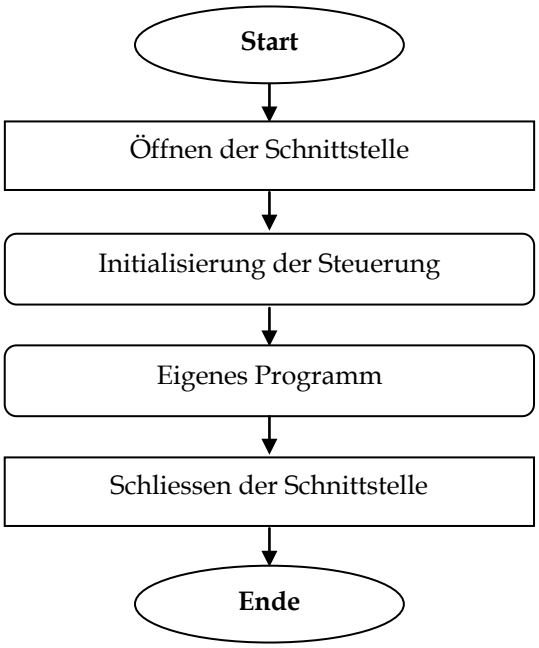


## 6.3 Hinweise zum Aufbau eigener Programme bei der Programmierung der Steuerungen über das API

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

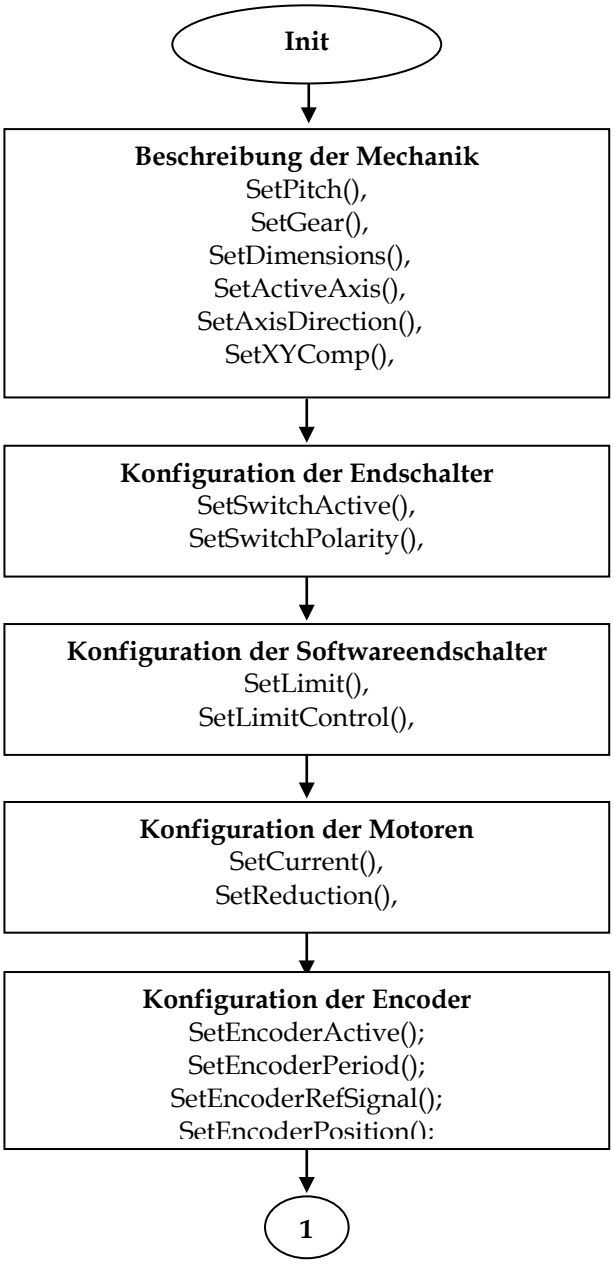
Da die vorkonfigurierten Default-Settings der Steuerung nicht für alle Anwendungen die entsprechenden Daten enthalten können sind, nachdem die verwendete Schnittstelle geöffnet worden ist, die im Punkt „Initialisierung der Steuerung“ beschriebenen Schritte vorzunehmen.

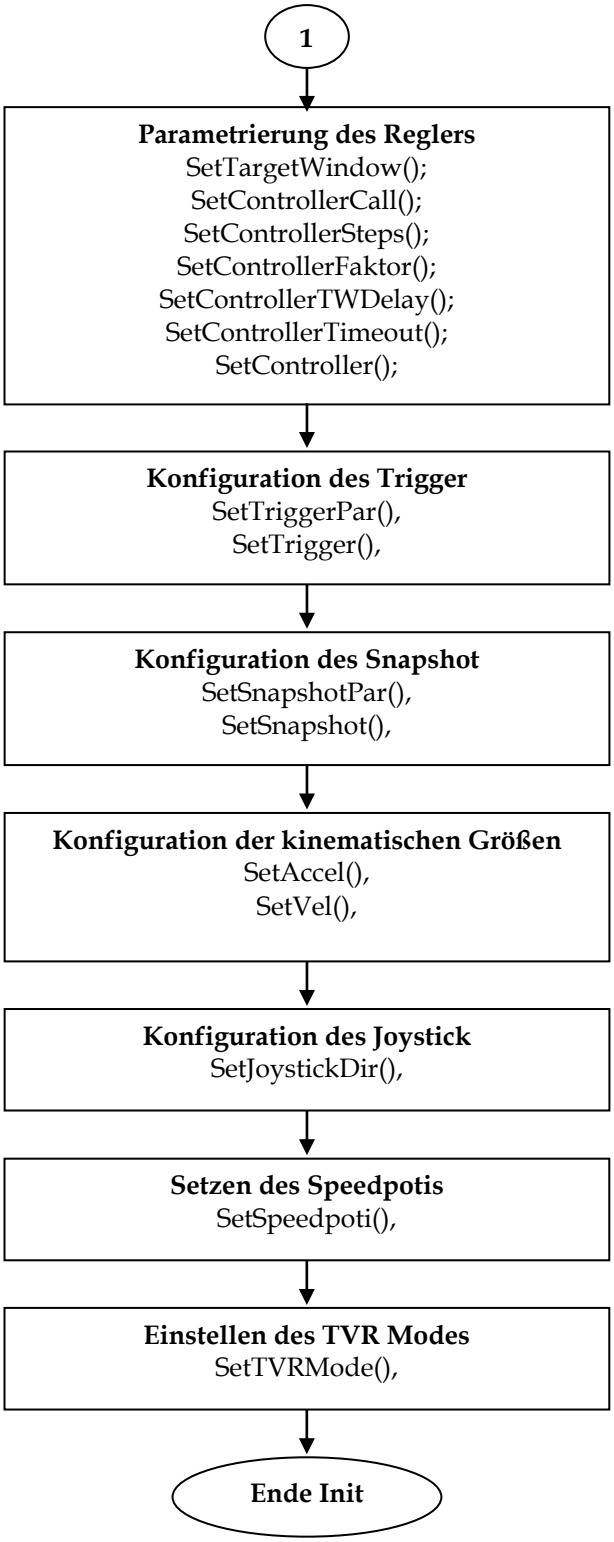
Anschliessend kann unter Verwendung des LSTEP-API ein beliebiges Anwenderprogramm geschrieben werden.



### 6.3.1 Initialisierung der Steuerung

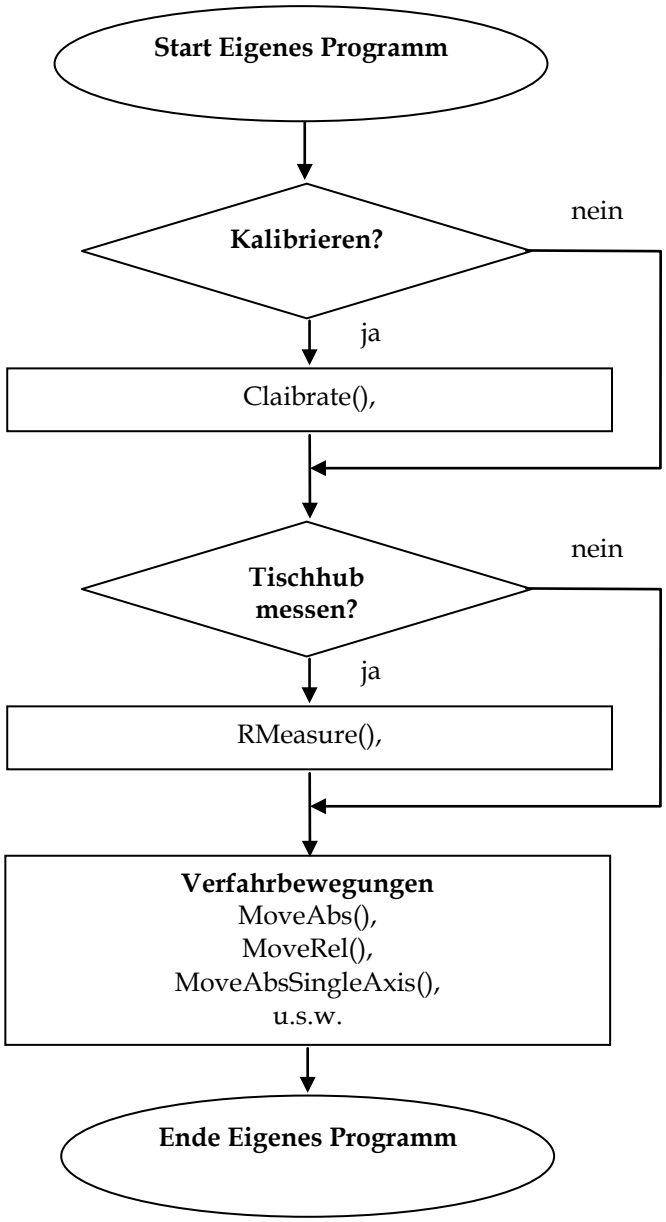
Vor dem Start des eigenen Programmteils sind bei der Initialisirung der verwendeten LSTEP die im Folgenden beschriebenen Grundeinstellungen der Positioniersteuerung vorzunehmen um einen fehlerfreien Betrieb zu gewährleisten.




1
Parametrierung des Reglers
SetTargetWindow();
SetControllerCall();
SetControllerSteps();
SetControllerFaktor();
SetControllerTWDelay();
SetControllerTimeout();
SetController();
Konfiguration des Trigger
SetTriggerPar(),
SetTrigger(),
Konfiguration des Snapshot
SetSnapshotPar(),
SetSnapshot(),
Konfiguration der kinematischen Größen
SetAccel(),
SetVel(),
Konfiguration des Joystick
SetJoystickDir(),
Setzen des Speedpotis
SetSpeedpoti(),
Einstellen des TVR Modes
SetTVRMode(),
Ende Init

### 6.3.2 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/Os ebenso wie das Setzen von Triggersignalen in Abhängigkeit von der Position usw..



## 6.4 Funktionen

### 6.4.1 Index 1(Kuzbeschreibung für API-Befehle)

Gliederung der Befehle nach Themen wie folgt:

## Achtung!

- **In dieser Kurzbeschreibung sind nur die DLL-Aufrufe beschrieben.**
- **Alle neuen Befehle müssen mit dem DLL-Aufruf „SendString" gesendet werden**
- **Eine komplette Liste aller Befehle finden Sie am Ende dieses Kapitels im Index 2.**

**API-Konfiguration/Schnittstelle**

**Steuerungs-Info:**



**Einstellungen**

**Statusabfragen**

| Befehl | Kurzbeschreibung | Seite |
|---|---|---|
| GetError | liefert die aktuelle Fehlernummer | 73 |
| GetSecurityErr | Liest alle Zustände und Ergebnisse der GAL-Sicherheitsüberwachung (nur bei LS44-Steuerungen) | 74 |
| GetSecurityStatus | Liest den aktuellen Zustand der Sicherheitsüberwachung | 75 |
| GetStatus | liefert den aktuellen Zustand der Steuerung | 76 |
| GetStatusAxis | liefert den aktuellen Zustand der einzelnen Achsen | 76 |
| GetStatusLimit | Liefert den aktuellen Zustand der Software-Grenzen jeder einzelnen Achse | 77 |
| SetAutoStatus | AutoStatus Ein/Aus | 77 |

**Fahrbefehle und Positionsverwaltung**

| Befehl | Kurzbeschreibung | Seite |
|---|---|---|
| Calibrate | Kalibrieren | 78 |
| CalibrateEx | Es werden nur die Achsen kalibriert, deren entsprechendes Bit in dem übergebenen Integer-Wert gesetzt ist. | 78 |
| Clearpos | Positionswerte werden genullt (für endlos Drehachsen) | 79 |
| GetDelay | Liefert die Verzögerung des Vektorstarts | 79 |
| GetDistance | Liefert die Strecke, die mit LS_PosRelShort gestartet wird | 80 |
| GetInputTrigMove | Liefert die Konfiguration vom Pin1 auf dem MFP | 81 |
| GetPos | Abfrage der aktuellen Position aller Achsen | 82 |
| GetPosEx | Abfrage der aktuellen Geber- bzw. Positionswerte aller Achsen | 84 |
| GetPosSingleAxis | Abfrage der aktuellen Position einer Achse | 84 |
| MoveAbs | Absolutposition anfahren | 85 |
| MoveAbsSingleAxis | Absolutposition einer Achse anfahren | 85 |
| MoveEx | Erweiterter Verfahr-Befehl | 86 |
| MoveRel | Relativen Vektor fahren | 87 |
| MoveRelShort | Positionieren Relativ (short command) | 87 |
| MoveRelSingleAxis | Relativen Vektor einer Achse verfahren | 88 |
| RMeasure | Tischhub messen | 88 |
| RmeasureEx | Tischhub messen wird nur bei den Achsen durchgeführt, deren entsprechendes Bit in dem übergebenen Integer-Wert gesetzt ist | 89 |
| SetDelay | Durch den Befehl Delay kann eine Verzögerung des Vektorstarts erzeugt werden | 80 |
| SetDistance | Strecke setzen (für MoveRelShort) | 81 |
| SetInputTrigMove | Konfiguriert den Pin 1 auf dem MFP | 82 |
| SetPos | Position setzen | 89 |
| StopAxes | alle Verfahrbewegungen werden abgebrochen | 90 |
| WaitForAxisStop | Die Funktion kehrt zurück, sobald die in der Bit-Maske AFlags gewählten Achsen ihre Zielposition erreicht haben | 90 |

### Joystick und Handrad

| Befehl | Kurzbeschreibung | Seite |
|---|---|---|
| GetDigJoySpeed | Digitaler Joystick und Geschwindigkeit lesen | 91 |
| GetHandwheel | Liest den Zustand des Handrads | 92 |
| GetJoyChangeAxis | Liest Joystickachszuordnung | 97 |
| GetJoystick | Liest den Zustand des Analog-Joysticks | 92 |
| GetJoystickAxes | Zeigt, für welche Achsen Joystick eingeschaltet ist | 93 |
| GetJoystickDir | Richtung Joystick | 94 |
| GetJoyVel | Die maximalenVerfahrgeschwindigkeitenm Joystickbetrieb abfragen | 95 |
| GetJoystickWindow | Joystick-Fenster ablesen | 96 |
| JoyChangeAxis | Setzt Joystickachszuordnung | 97 |
| SetDigJoySpeed | Digitaler Joystick und Geschwindigkeit setzen | 91 |
| SetDigJoyOff | Schaltet digitalen Joystick aus | 97 |
| SetHandwheelOff | Handrad Aus | 98 |
| SetHandwheelOn | Handrad Ein | 98 |
| SetJoystickAxes | Schaltet Joystick für angegebene Achsen ein | 93 |
| SetJoystickDir | Richtung Joystick | 95 |
| SetJoystickOff | Analog-Joystick Aus | 98 |
| SetJoystickOn | Analog-Joystick Ein | 99 |
| SetJoyVel | Die maximalen Verfahrgeschwindigkeiten im Joystickbetrieb einstellen | 96 |
| SetJoystickWindow | Joystick-Fenster setzen | 96 |

### Bedienpult mit Trackball und Joyspeed-Tasten

| Befehl | Kurzbeschreibung | Seite |
|---|---|---|
| GetBPZ | Liest Zustand des Bedienpults | 100 |
| GetBPZJoyspeed | Liest Bedienpult Joystick-Speed | 101 |
| GetBPZTrackballBackLash | Liest Bedienpult Trackball-Umkehrspiel | 101 |
| GetBPZTrackballFactor | Liest Bedienpult Trackball-Faktor | 102 |
| SetBPZ | Bedienpult Ein/Aus | 100 |
| SetBPZJoyspeed | Bedienpult Joystick-Speed | 101 |
| SetBPZTrackballBackLash | Bedienpult Trackball-Umkehrspiel | 102 |
| SetBPZTrackballFactor | Bedienpult Trackball-Faktor | 102 |

**Endschalter (Hardware u. Software)**

| Befehl | Kurzbeschreibung | Seite |
|---|---|---|
| GetAutoLimitAfterCalibRM | Gibt an, ob beim Kalibrieren und Tischhubmessendie interne Software-Limits gesetzt werden. | 103 |
| GetLimit | Liefert Verfahrbereichsgrenzen | 104 |
| GetLimitControl | Liest,ob die Bereichsüberwachung eingeschaltet ist | 105 |
| GetLimitControlMode | Liefert den Modus für die Überwachung der Softwarelimits. | 106 |
| GetSwitchActive | Gibt an, ob die Endschalter eingeschaltet sind | 107 |
| GetSwitches | Liest den Zustand aller Endschalter | 108 |
| GetSwitchPolarity | Liest Endschalterpolarität | 108 |
| GetSwChange | Zeigt die Einstellungen der Endschalter | 109 |
| SetAutoLimitAfterCalibRM | Verhindert, dass beim Kalibrieren und Tischhubmessen die internen Software-Limits gesetzt werden. | 103 |
| SetLimit | Verfahrbereichsgrenzen einstellen | 104 |
| SetLimitControl | Bereichsüberwachung | 105 |
| SetLimitControlMode | Setzt den Modus für die Überwachung der Softwarelimits | 106 |
| SetSwitchActive | Liest Status für Endschalter Ein / Aus | 107 |
| SetSwitchPolarity | Endschalterpolarität einstellen | 109 |
| SetSwChange | Endschalter tauschen | 110 |

**Digitale und analoge Ein.- und Ausgänge**

| Befehl | Kurzbeschreibung | Seite |
|---|---|---|
| GetAnalogInput | Lesen des aktuellen Zustands eines Analogkanals | 111 |
| GetAnalogInputs2 | Lesen der aktuellen Zustände der Analogkanäle PT100, MV und V24 | 111 |
| GetDigitalInputs | Alle Inputpins lesen | 112 |
| GetDigitalInputsE | Zusätzliche digitale Eingänge lesen (16-31) | 112 |
| SetAnalogOutput | Analogkanal setzen | 113 |
| SetDigIO_Distance | Aktivierung eines Ausgang in Abhängigkeit der eingestellten Strecke vor/nach der Zielposition | 113 |
| SetDigIO_EmergencyStop | Funktion der digitalen Ein-/Ausgänge Zuordnung des Not-Stop-Pins | 114 |
| SetDigIO_Off | Funktion der digitalen Ein-/Ausgänge Aus | 114 |
| SetDigIO_Polarity | Einstellung der Polarität | 115 |
| SetDigitalOutput | Digitalen Ausgang setzen | 115 |
| SetDigitalOutputs | Digitale Ausgänge setzen (0-15) | 116 |
| SetDigitalOutputsE | Zusätzliche digitale Ausgänge setzen (16-31) | 116 |

**Takt-Vor/Rück Eingänge**

| Befehl | Kurzbeschreibung | Seite |
|---|---|---|
| GetFactorTVR | Liest den Faktor Takt Vor / Rück | 117 |
| GetTVRMode | Einstellung vom Takt Vor / Rück auslesen | 118 |
| SetFactorTVR | Faktor Takt Vor / Rück | 117 |
| SetTVRMode | Takt Vor / Rück einstellen | 119 |

### Takt-Vor/Rück über Schnittstelle

| Befehl | Kurzbeschreibung | Seite |
|---|---|---|
| SetTVRInPulse | Takt-Vor/Rück über Schnittstelle | 120 |

### Takt-Vor/Rück Ausgänge für weitere Achsen

| Befehl | Kurzbeschreibung | Seite |
|---|---|---|
| GetAccelTVRO | Alle eingestellten Beschleunigungen lesen | 121 |
| GetPosTVRO | Liefert Positionswerte, in Abhängigkeit von Dimension | 122 |
| GetStatusTVRO | Liefert den aktuellen Status der Achsen | 123 |
| GetTVROutMode | Einstellung vom Takt Vor/Rück lesen | 123 |
| GetTVROutPitch | Liest Spindelsteigung | 124 |
| GetTVROutResolution | Liefert dieAuflösung der an zu steuernden Endstufe | 125 |
| GetVelTVRO | Alle eingestellten Geschwindigkeiten lesen | 126 |
| MoveAbsTVROSingleAxis | Absolutposition einer Achse anfahren | 127 |
| MoveAbsTVRO | Absolutposition anfahren | 128 |
| MoveRelTVROSingleAxis | Relativen Vector einer Achse verfahren | 128 |
| MoveRelTVRO | Relativen Vector  verfahren | 129 |
| SetAccelSingleAxisTVRO | Beschleunigung einer einzelnen Achse setzen | 129 |
| SetAccelTVRO | Beschleunigungen setzen | 121 |
| SetPosTVRO | Position setzen | 122 |
| SetTVROutMode | Takt Vor / Rück einstellen | 124 |
| SetTVROutPitch | Spindelsteigung setzen | 125 |
| SetTVROutResolution | Auflösung der an zu steuernden Endstufe | 126 |
| SetVelSingleAxisTVRO | Geschwindigkeit einer einzelnen Achse setzen | 130 |
| SetVelTVRO | Geschwindigkeiten setzen | 127 |

### Geber-Einstellungen

| Befehl | Kurzbeschreibung | Seite |
|---|---|---|
| ClearEncoder | Geberposition auf null setzen | 131 |
| GetEncoder | Liest alle Geberpositionen | 131 |
| GetEncoderActive | Liest, welche Geber nach der Kalibration aktiv werden | 132 |
| GetEncoderMask | Geberzustände auslesen | 133 |
| GetEncoderPeriod | Geberperiodenlängen auslesen | 134 |
| GetEncoderPosition | Liefert Einstellung von Geberwertanzeige | 135 |
| GetEncoderRefSignal | Gibt an, ob beim Kalibrieren Referenzsignal von Geber ausgewertet werden soll | 136 |
| SetEncoderActive | Mit dieser Funktion kann ausgewählt werden, welche Geber nach der Kalibration aktiviert werden sollen | 132 |
| SetEncoderMask | Geber (de-)aktivieren | 133 |
| SetEncoderPeriod | Geberperiodenlängen einstellen | 134 |
| SetEncoderPosition | Geberwertanzeige Ein/Aus | 135 |
| SetEncoderRefSignal | Beim Kalibrieren Referenzsignal von Geber auswerten | 136 |

**Reglereinstellungen**

| Befehl | Kurzbeschreibung | Seite |
|---|---|---|
| ClearCtrFastMoveCounter | Anzahl ausgeführter FastMove Funktionen auf 0 setzen | 137 |
| GetController | Regler-Modus auslesen | 137 |
| GetControllerCall | Einstellung vom Regleraufruf auslesen | 138 |
| GetControllerFactor | Einstellung vom Reglerfaktor auslesen | 139 |
| GetControllerSteps | Regler-Schritte auslesen | 140 |
| GetControllerTimeout | Liefert die Einstellung vom Regler-Überwachungs-Timeout | 141 |
| GetControllerTWDelay | Reglerverzögerung auslesen | 142 |
| GetCtrFastMove | Einstellung von Fast Move Funktion lesen | 143 |
| GetCtrFastMoveCounter | Anzahl ausgeführter FastMove Funktionen auf 0 auslesen | 144 |
| GetTargetWindow | Liefert Reglerzielfenster | 144 |
| SetController | Regler-Modus einstellen | 138 |
| SetControllerCall | Regleraufruf einstellen | 139 |
| SetControllerFactor | Reglerfaktor einstellen | 140 |
| SetControllerSteps | Regler-Schritte einstellen | 141 |
| SetControllerTimeout | Regler-Überwachungs-Timeout einstellen [ms] | 142 |
| SetControllerTWDelay | Reglerverzögerung setzen | 143 |
| SetCtrFastMoveOff | Fast Move Funktion „AUS“ | 145 |
| SetCtrFastMoveOn | Fast Move Funktion „EIN“ | 146 |
| SetTargetWindow | Reglerzielfenster einstellen | 145 |

**Trigger-Ausgang**

| Befehl | Kurzbeschreibung | Seite |
|---|---|---|
| GetTrigCount | Triggerzählerstand setzen | 147 |
| GetTrigger | Einstellung vom Trigger auslesen | 147 |
| GetTriggerPar | Trigger Parameter auslesen | 148 |
| SetTrigCount | Triggerzählerstand lesen | 147 |
| SetTrigger | Trigger Ein/ Aus | 148 |
| SetTriggerPar | Trigger Parameter | 149 |

**Snapshot-Eingang**

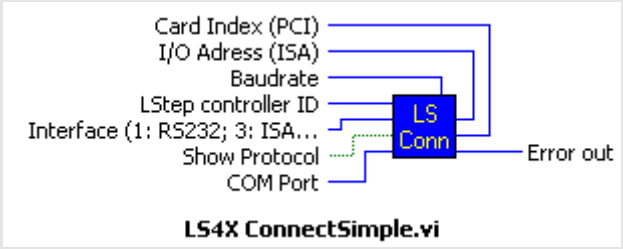
| Befehl | Kurzbeschreibung | Seite |
|---|---|---|
| GetSnapshot | Einstellung vom Snapshot auslesen | 150 |
| GetSnapshotCount | Snapshot-Zähler | 150 |
| GetSnapshotFilter | Eingangsfilter auslesen | 151 |
| GetSnapshotPar | Snapshot-Parameter auslesen | 151 |
| GetSnapshotPos | Snapshot-Position auslesen | 152 |
| GetSnapshotPosArray | Snapshot-Position aus Array auslesen | 153 |
| SetSnapshot | Snapshot Ein/Aus | 150 |
| SetSnapshotFilter | Eingangsfilter setzen | 151 |
| SetSnapshotPar | Snapshot-Parameter | 152 |

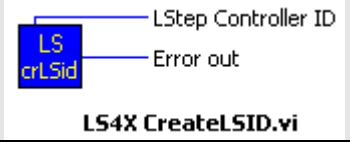
### 6.4.2 Funktionen

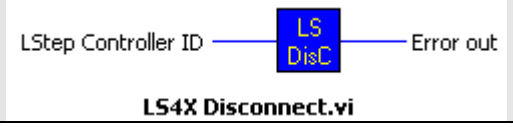
**API-Konfiguration/Schnittstelle**

| **LS_Connect** | |
|---|---|
| **Beschreibung:** | Mit LSTEP verbinden<br>Dazu werden die Schnittstellen-Parameter verwendet, welche mittels LS_LoadConfig aus der INI-Datei geladen wurden.<br>(Eine der Funktionen LS_Connect, LS_ConnectSimple oder LS_ConnectEx muß zur Initialisierung der Schnittstelle aufgerufen werden, damit die Kommunikation mit der LSTEP möglich ist.) |
| **Delphi:** | function LS_Connect: Integer;<br>function LSX_Connect(LSID: Integer): Integer; |
| **C++:** | int Connect(); |
| **LabView:** |  |
| **Parameter:** | - |
| **Beispiel:** | LS.LoadConfig("C:\LStepTest\LStep.INI");<br>LS.Connect(); |

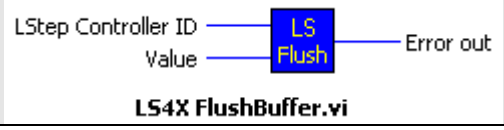
| **LS_ConnectEx** | |
|---|---|
| **Beschreibung:** | Mit LSTEP verbinden |
| | Diese Funktion bietet erweitere Möglichkeiten, übergeben wird ein Zeiger auf eine Datenstruktur, die die Schnittstellenparameter enthält. In dem Record werden außerdem Informationen über die erkannte Steuerung zurückgeliefert (Versionsnummer...)<br>(Eine der Funktionen LS_Connect, LS_ConnectSimple oder LS_ConnectEx muß zur Initialisierung der Schnittstelle aufgerufen werden, damit die Kommunikation mit der LSTEP möglich ist.) |
| **Delphi:** | function LS_ConnectEx(var AControlInitPar: TLS_ControlInitPar): Integer;<br>function LSX_ConnectEx(LSID: Integer; var AControlInitPar: TLS_ControlInitPar): Integer; |
| **C++:** | int ConnectEx (TLS_ControlInitPar *pAControlInitPar); |
| **Parameter:** | AControlInitPar: Zeiger auf einen Record des Typs TLS_ControlInitPar |
| **Beispiel:** | LS.ConnectEx(&ControlInitPar1); |

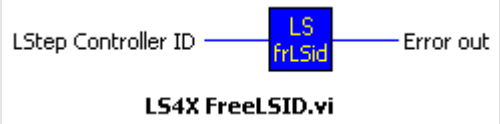
| LS_ConnectSimple | |
|---|---|
| **Beschreibung:** | Mit LSTEP verbinden |
| | Die Einstellungen der Schnittstelle werden als Parameter übergeben<br>(Eine der Funktionen LS_Connect, LS_ConnectSimple oder LS_ConnectEx muß zur Initialisierung der Schnittstelle aufgerufen werden, damit die Kommunikation mit der LSTEP möglich ist.) |
| **Delphi:** | function LS_ConnectSimple(<br>AnInterfaceType: Integer;<br>AComName: PChar;<br>ABR: Integer;<br>AShowProt: LongBool): Integer;<br>function LSX_ConnectSimple(LSID: Integer; AnInterfaceType: Integer;<br>AComName: PChar; ABaudRate: Integer; AShowProt: LongBool): Integer; |
| **C++:** | int Connect (<br>int lAnInterfaceType,<br>char *pcAComName,<br>int lABR,<br>BOOL AShowProt); |
| **LabView:** |  |
| **Parameter:** | AnInterfaceType: Schnittstellentyp<br>1 = RS232<br>2 = ArcNet<br>3 = DPRAM / ISA-Bus<br>4 = DPRAM / PCI-Bus<br>11= RS232 mit RTS/CTS Auswertung<br>AComName: Name der COM-Schnittstelle, z. B. 'COM2', bei ArcNet oder DPRAM auf NULL setzen<br>ABR: Bedeutung ist abhängig vom Schnittstellentyp<br>RS232 → Baudrate, z. B. 9600<br>ArcNet: → 0 für Koax, 1 für Twisted Pair<br>DPRAM / ISA-Bus: → Basis-I/O-Adresse der Karte, z.B. 0x0340<br>DPRAM / PCI-Bus: → O=erste Karte 1=zweite Karte<br>AShowProt: bestimmt, ob das Schnittstellenprotokoll angezeigt werden soll |
| **Beispiel:** | LS.ConnectSimple(1, "COM2", 9600, true);<br>// RS232, 9600 Baud<br>oder<br>LS_ConnectSimple(4, nil, 0, true);<br>//LStep PCI Karte 0; |

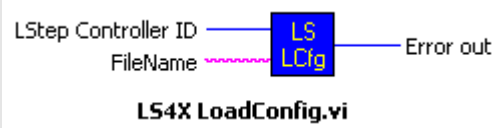
| LSX_CreateLSID (nur LSTEP4X-API) | |
|---|---|
| **Beschreibung** | Erzeugt eine LStep-ID-Nummer. Diese wird als zusätzlicher Parameter bei den LSTEP4X-API- Befehlen verwendet, um aus mehreren angeschlossenen LSteps die LStep zu wählen, auf die sich der Befehl beziehen soll. |
| **Delphi** | function 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;<br>...<br>LSX_CreateLSID(&LStep1); |

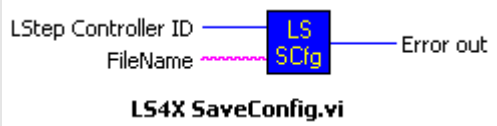
| LS_Disconnect | |
|---|---|
| **Beschreibung:** | Verbindung zu LSTEP trennen |
| | Nach Aufruf dieser Funktion können keine Befehle mehr zur LSTEP gesendet werden. Die Funktion sollte kurz vor Beendigung des Programms aufgerufen werden. |
| **Delphi:** | function LS_Disconnect: Integer;<br>function LSX_Disconnect(LSID: Integer): Integer; |
| **C++:** | int Disconnect (); |
| **LabView:** |  |
| **Parameter:** | - |
| **Beispiel:** | LS.Disconnect(); |

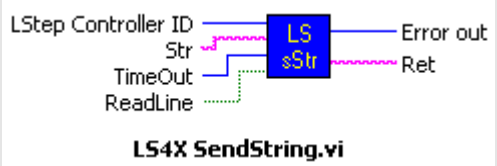
| LS_EnableCommandRetry | |
|---|---|
| **Beschreibung** | Mit dieser Funktion kann das wiederholte Senden von Kommandos im Falle von Fehlern ein-/ausgeschaltet werden (Standardmäßig ist dieses eingeschaltet) |
| **Delphi** | function LS_EnableCommandRetry(AValue: LongBool): Integer;<br>function LSX_EnableCommandRetry(LSID: Integer; AValue: LongBool): Integer; |
| **C++** | int EnableCommandRetry (BOOL bAValue); |
| **LabView:** |  |
| **Parameter** | AValue:<br>true => bei Fehlern wiederholt das LStep API das Senden bestimmter Kommandos (insbesondere bei WaitForAxisStop)<br>false => wiederholtes Senden abschalten |
| **Beispiel** | LS.EnableCommandRetry(false) ; |

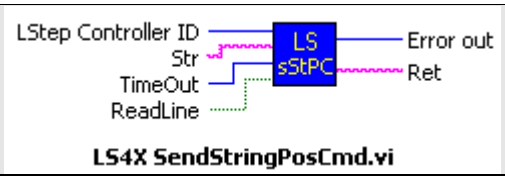
| LS_FlushBuffer | |
|---|---|
| **Beschreibung:** | Kommunikations-Eingabepuffer löschen (RS-232 und PCI)<br>kann in Fehler-Situationen verwendet werden, um nicht mehr benötigte Rückmeldungungen aus dem Eingabepuffer zu entfernen |
| **Delphi:** | function LS_FlushBuffer(AValue: Integer): Integer;<br>function LSX_FlushBuffer(LSID: Integer; AValue: Integer): Integer; |
| **C++:** | int FlushBuffer (int lAValue); |
| **LabView:** |  |
| **Parameter:** | AValue: momentan nicht verwendet, kann =0 gesetzt werden |
| **Beispiel:** | LS.FlushBuffer(0); |

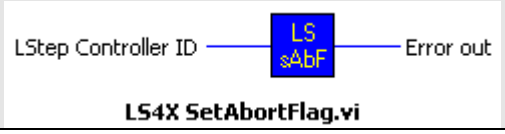
| LSX_FreeLSID (nur LSTEP4X-API) | |
|---|---|
| **Beschreibung** | Gibt eine erzeugte LStep-ID-Nummer wieder frei. Diese wird als zusätzlicher Parameter bei den LSTEP4X-API- Befehlen verwendet, um aus mehreren angeschlossenen LSteps die LStep zu wählen, auf die sich der Befehl beziehen soll. FreeLSID sollte erst nach Disconnect aufgerufen werden. |
| **Delphi** | function LSX_FreeLSID(LSID: Integer): Integer; |
| **C++** | - |
| **LabView:** |  |
| **Parameter** | LSID: freizugebende LStep-ID-Nummer; diese darf nach FreeLSID nicht mehr verwendet werden |
| **Beispiel** | var LStep1: Integer;<br>...<br>LSX_CreateLSID(&LStep1);<br>LSX_ConnectSimple(LStep1, ...);<br>...<br>LSX_Disconnect(LStep1);<br>LSX_FreeLSID(LStep1); |

| LS_LoadConfig | |
|---|---|
| **Beschreibung:** | LSTEP-Konfiguration (Schnittstelle, Achseinstellungen, Regler) aus INI-Datei laden.<br>Das Format der INI-Datei ist mit der Win-Commander-INI-Datei kompatibel, d.h. die Einstellungen können aus dem Win-Commander (Wincom4.ini) übernommen werden.<br>Die geladene Konfiguration wird in den Funktionen LS_Connect und LS_SetControlPars verwendet.<br>Achtung!<br>Für die LSTEP-PCIexpress oder LSTEPexpress kann man mit dem WinCommander 5 im Menü Steuerung / Konfiguration exportieren die Einstellungen speichern und diese mit LoadConfig laden. |
| **Delphi:** | function LS_LoadConfig(FileName: PChar): Integer;<br>function LSX_LoadConfig(LSID: Integer; FileName: PChar): Integer; |
| **C++:** | int LoadConfig (char *pcFileName); |
| **LabView:** |  |
| **Parameter:** | FileName: Dateiname der INI-Datei als nullterminierter String |
| **Beispiel:** | LS.LoadConfig("C:\LStepTest\LStep.INI"); |

| LS_SaveConfig | |
|---|---|
| **Beschreibung:** | LSTEP-Konfiguration (Schnittstelle, Achseinstellungen, Regler) in INI-Datei speichern.<br>Das Format der INI-Datei ist mit der Win-Commander-INI-Datei kompatibel. |
| **Delphi:** | function LS_SaveConfig(FileName: PChar): Integer;<br>function LSX_SaveConfig(LSID: Integer; FileName: PChar): Integer; |
| **C++:** | int SaveConfig (char *pcFileName); |
| **LabView:** |  |
| **Parameter:** | FileName: Dateiname der INI-Datei als nullterminierter String |
| **Beispiel:** | LS.SaveConfig("C:\LStepTest\LStep.INI"); |

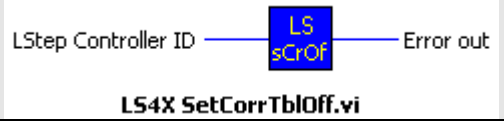
| LS_SendString | | |
|---|---|---|
| **Beschreibung:** | String an LSTEP senden | |
| **Delphi:** | function LS_SendString(Str, Ret: PChar; MaxLen: Integer; ReadLine: LongBool; TimeOut: Integer): Integer;<br>function LSX_SendString(LSID: Integer; Str, Ret: PChar; MaxLen: Integer; ReadLine: LongBool; TimeOut: Integer): Integer; | |
| **C++:** | int SendString (char *pcStr,char *pcRet,int lMaxLen,BOOL ReadLine,int lTimeOut); | |
| **LabView:** |  | |
| **Parameter:** | Str | ➔ nullterminierter String, der an die Steuerung gesendet werden soll. |
| | Ret | ➔ Puffer, der die Rückmeldung der LSTEP enthält, falls ReadLine = true |
| | MaxLen | ➔ Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen. |
| | ReadLine | ➔ Rückmeldung der LSTEP lesen: |
| | TimeOut | ➔ maximale Wartezeit auf Rückmeldung [ms] |
| **Beispiel:** | LS.SendString("?ver\r", pcLStepVer, 256, true, 1000);<br>// Versionsnummer lesen, Timeout 1s | |

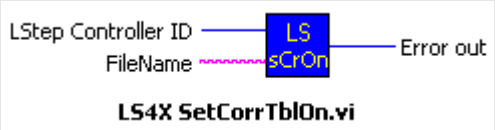
| LS_SendStringPosCmd | |
|---|---|
| **Beschreibung** | Verfahrbefehl, welcher Rückmeldung erwartet, als String an LSTEP senden |
| **Delphi** | function LS_SendStringPosCmd(Str, Ret: PChar; MaxLen: Integer; ReadLine: LongBool; TimeOut: Integer): Integer;<br>function LSX_SendStringPosCmd(LSID: Integer; Str, Ret: PChar; MaxLen: Integer; ReadLine: LongBool; TimeOut: Integer): Integer; |
| **C++** | int SendStringPosCmd (char *pcStr, char *pcRet, int lMaxLen, BOOL bReadLine, int lTimeOut); |
| **LabView:** |  |
| **Parameter:** | Str ➔ nullterminierter String, der an die Steuerung gesendet werden soll. |
| | Ret ➔ Puffer, der die Rückmeldung der LSTEP enthält, falls ReadLine = true |
| | MaxLen ➔ Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen. |
| | ReadLine ➔ Rückmeldung der LSTEP lesen: |
| | TimeOut ➔ maximale Wartezeit auf Rückmeldung [ms] |
| **Beispiel** | LS.SendStringPosCmd("!moa 1 2\r", pcLStepVer, 256, true, 100000); |

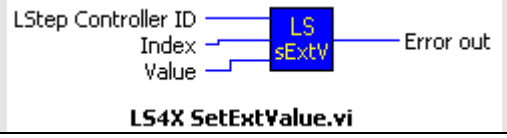
| LS_SetAbortFlag | |
|---|---|
| **Beschreibung:** | Flag setzen, damit die Kommunikation mit der LSTEP abgebrochen wird |
| | Eine Funktion, die bei Aufruf von LS_SetAbortFlag noch auf eine Rückmeldung der Steuerung warten (z.B. Verfahrbefehle), kehrt dann mit einer Fehlermeldung zurück.<br>Die Verwendung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

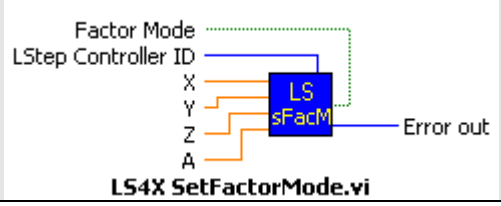
| LS_SetCommandTimeout | |
|---|---|
| **Beschreibung:** | Setzt die Timeoutzeiten für das Warten auf Rückmeldung, Positionieren und Kalibrieren. |
| **Delphi:** | function LS_ SetCommandTimeout (AtoRead, AtoMove, AtoCalibrate: Integer): Integer;<br>LSX_SetCommandTimeout(LSID: Integer; AtoRead, AtoMove, AtoCalibrate: Integer): Integer; |
| **C++:** | int SetCommandTimeout (int lAtoRead, int lAtoMove, int lAtoCalibrate) ; |
| **LabView:** |  |
| **Parameter:** | AtoRead: Timeoutzeit für das Warten auf Rückmeldung [ms]<br>AtoMove: Timeoutzeit für Positionieren [ms]<br>AtoCalibrate: Timeoutzeit für Kalibrieren [ms] |
| **Beispiel:** | LS. SetCommandTimeout (int lAtoRead, int lAtoMove, int lAtoCalibrate) ; |

| LS_SetControlPars | |
|---|---|
| **Beschreibung:** | Überträgt die mit LS_LoadConfig geladenen Parameter an die LSTEP. |
| **Delphi:** | function LS_SetControlPars: Integer;<br>function LSX_SetControlPars(LSID: Integer): Integer; |
| **C++:** | int SetControlPars (); |
| **LabView:** |  |
| **Parameter:** | - |
| **Beispiel:** | LS.SetControlPars(); |

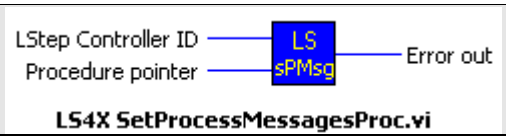
| LS_SetCorrTblOff | |
|---|---|
| **Beschreibung** | Achsenkorrektur deaktivieren |
| **Delphi** | function LS_SetCorrTblOff: Integer;<br>function LSX_SetCorrTblOff(LSID: Integer): Integer; |
| **C++** | int SetCorrTblOff (); |
| **LabView:** |  |
| **Parameter** | - |
| **Beispiel** | LS.SetCorrTblOff() ; |

| LS_SetCorrTblOn | |
|---|---|
| **Beschreibung** | Achsenkorrektur in x/y-Matrix mit linearer Interpolation aktivieren |
| **Delphi** | function LS_SetCorrTblOn(AFileName: PChar): Integer;<br>function LSX_SetCorrTblOn(LSID: Integer; AFileName: PChar): Integer; |
| **C++** | int SetCorrTblOn (char *pcAFileName); |
| **LabView:** |  |
| **Parameter** | Die Korrekturtabelle wird manuell in eine Ini-Datei eingegeben. Der Dateiname dieser Ini-Datei wird durch AFileName angegeben.<br><br>Aufbau der Korrekturtabelle:<br><br>In der Sektion [Options] wird die Achsenkorrektur mit lin. Interpolation durch die Zeile „CorrectionXY=1“ aktiviert. XCount und YCount geben die Anzahl der Korrekturwerte an. Der Parameter XDistance bestimmt den Abstand der Meßpunkte in einer Reihe (X-Achse), YDistance den Abstand der Reihen (Y-Achse).<br><br>Die Sektion [CorrTbl] enthält die Korrekturwerte. Jeder Soll-Position (x/y-Wertepaar) wird eine korrigierte Position zugeordnet, wobei die Soll-Positionen immer einer der Punkte in dem durch XCount, YCount, XDistance und YDistance festgelegten Raster sein müssen.<br>Die Zuordnungen (Soll-Position=Korrigierte Position) können in der Korrekturtabelle in beliebiger Reihenfolge erfolgen, wichtig ist nur, daß die Soll-Positionen immer in diesem Raster liegen (Der Nullpunkt der Korrekturtabelle ist (0\|0)).<br><br>Beispiel einer Korrekturtabelle:<br>[Options]<br>CorrectionXY=1<br>XCount=3<br>YCount=3<br>XDistance=1.0<br>YDistance=1.0<br>[CorrTbl]<br>0.0 0.0=0.0 0.0<br>1.0 0.0=1.0 0.0<br>2.0 0.0=2.0 0.0<br>0.0 1.0=0.0 1.0<br>1.0 1.0=0.9 1.1 (Soll-Position x=1 y=1, korrigierte Position x=0.9 y=1.1)<br>2.0 1.0=2.0 1.0<br>0.0 2.0=0.0 2.0<br>1.0 2.0=1.0 2.0<br>2.0 2.0=2.0 2.0 |
| **Beispiel** | LS.SetCorrTblOn(„C:\...\corrtbl.ini“); |

| LS_SetExtValue | |
|---|---|
| **Beschreibung:** | Schaltet Erweiterungen des API ein, teilweise handelt es sich dabei um experimentelle Modi zu Debugging-Zwecken |
| **Delphi:** | function LS_SetExtValue(AName: Integer; AValue: Integer): Integer;<br>function LSX_SetExtValue(LSID: Integer; AName, AValue: Integer): Integer; |
| **C++:** | int SetExtValue (int lAName, int lAValue); |
| **LabView:** |  |
| **Parameter:** | AName: Nummer der erweiterten Funktion<br>AValue: Parameter<br><br>**AName=2 (IFSleepTime)** stellt das Polling-Intervall für das DPRAM der LStep-PCI ein<br>AValue: Zeit-Intervall in [ms], Standard ist 10<br><br>**AName=3 (ProtMoveOnly)** schaltet Filter für Log-Datei ein, durch welches nur Moves&Fehler protokolliert werden<br>AValue=1 → Filter an<br>AValue=0 → Filter aus<br><br>**AName=4 (Max_LogLn)** begrenzt die Länge der Log-Datei, ältere Log-Datei wird in .old umbenannt<br>AValue=Maximale Zeilenzahl<br><br>**AName=5 (ThreadPriority)** ändert die Priority der Threads des LStep API. Nach Connect werden die Threads immer auf normale Priorität gesetzt, mit SetExtValue(5, ...) kann dies im nachhinein geändert werden.<br>AValue=Windows-API-Konstante für Thread-Priorität wie THREAD_PRIORITY_ABOVE_NORMAL |
| **Beispiel:** | LS.SetExtValue(3, 1); // Filter für Move-Befehle an<br>LS.SetExtValue(4, 10000); // maximale Länge der Log-Datei = 10000 Zeilen<br>LS.SetExtValue(5, THREAD_PRIORITY_HIGHEST); |

| LS_SetFactorMode | |
|---|---|
| **Beschreibung** | Positionswert-Umrechnung für ‚krumme' Spindelsteigungen |
| **Delphi** | function LS_SetFactorMode(AFactorMode: LongBool; X, Y, Z, A: Double): Integer;<br>function LSX_SetFactorMode(LSID: Integer; AFactorMode: LongBool; X, Y, Z, A: Double): Integer; |
| **C++** | int SetFactorMode (BOOL bAFactorMode, double dX, double dY, double dZ, double dA); |
| **LabView:** |  |
| **Parameter** | AFactorMode: Faktor-Modus einschalten<br><br>Der Befehl aktiviert eine API-interne Umrechnung der Positionswerte/Spindelsteigung, um so bei 'krummen' Spindelsteigungen Rundungsfehler zu vermeiden<br><br>X, Y, Z, R: Spindelsteigungswerte, die an die LStep übertragen werden (möglichst Werte wie 1.0 oder 4.0, sodass ein Mikrostep einem nichtperiodischen Dezimalbruch entspricht)<br><br>Erst **nach** SetFactorMode sollte dann SetPitch mit der tatsächlichen, physikalischen Spindelsteigung aufgerufen werden.<br><br>Alle Verfahrbefehle verwenden nach Aufruf von SetFactorMode und SetPitch eine Faktor-Umrechnung, damit die LStep korrekt positioniert.<br><br>an LStep gesendeter Positionsvektor = Positionsvektor * an LStep gesendete Spindelsteigung / physikalische Spindelsteigung |
| **Beispiel** | LS.SetFactorMode(true, 1, 1, 1, 0);<br>LS.SetPitch(1.234, 1.234, 2.345, 0);<br>LS.MoveAbs(1.234, 2.468, 2.345, 0, true); |

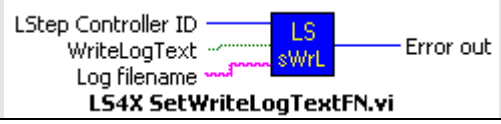
| LS_SetLanguage | |
|---|---|
| **Beschreibung:** | Sprachumschaltung LSTEP-API (Protokoll/Meldungen) |
| **Delphi:** | function LS_SetLanguage(PLN: PChar): Integer;<br>function LSX_SetLanguage(LSID: Integer; PLN: PChar): Integer; |
| **C++:** | int SetLanguage (char *pcPLN); |
| **LabView:** |  |
| **Parameter:** | PLN: Sprache (Kürzel, z.B. „DEU„ oder „ENG„)<br>Die entsprechende Textdatei (LSTEP4deu.txt oder LSTEP4eng.txt) muß im Verzeichnis des Programms liegen |
| **Beispiel:** | LS.SetLanguage('ENG'); |

| LS_SetProcessMessagesProc | |
|---|---|
| **Beschreibung** | Ermöglicht das Ersetzen der internen Message-Dispatching Prozedur des LStep API.<br>Das 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 Verwenden einer Callback-Prozedur verwenden. |
| **Delphi** | function LS_SetProcessMessagesProc(Proc: Pointer): Integer;<br>function LSX_SetProcessMessagesProc(LSID: Integer; Proc: Pointer): Integer; |
| **C++** | int SetProcessMessagesProc (void* pProc); |
| **LabView:** |  |
| **Parameter** | pProc muss ein Zeiger auf eine stdcall-Prozedur ohne Parameter sein:<br>void MyProcessMessages () { .. } |
| **Beispiel** | LS. SetProcessMessagesProc (&MyProcessMessages); |

| LS_SetShowCmdList | |
|---|---|
| **Beschreibung:** | LStep-API Befehlsliste Ein/Aus |
| **Delphi:** | function LS_SetShowCmdList(ShowCmdList: LongBool): Integer;<br>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:** | LS.SetShowCmdList(true);<br>// Schnittstellen-Protokoll zeigen falls nicht bereits sichtbar |

| LS_SetShowProt | |
|---|---|
| **Beschreibung:** | Schnittstellen-Protokoll Ein/Aus |
| **Delphi:** | function LS_SetShowProt(ShowProt: LongBool): Integer;<br>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:** | LS.SetShowProt(true);<br>// Schnittstellen-Protokoll zeigen falls nicht bereits sichtbar |

| LS_SetWriteLogText | |
|---|---|
| **Beschreibung:** | Schreiben der Protokoll-Datei LSTEP4.log ein-/ausschalten<br>(Standardmäßig ist das Schreiben in LSTEP4.log ausgeschaltet) |
| **Delphi:** | function LS_SetWriteLogText(AWriteLogText: LongBool): Integer;<br>function LSX_SetWriteLogText(LSID: Integer; AWriteLogText: LongBool): Integer; |
| **C++:** | int SetWriteLogText (BOOL AWriteLogText); |
| **LabView:** |  |
| **Parameter:** | - |
| **Beispiel:** | LS.SetWriteLogText (true); |

| LS_SetWriteLogTextFN | |
|---|---|
| **Beschreibung** | Schreiben des Schnittstellen-Protokolls in eine bestimmte Datei ein-/ausschalten (Standardmäßig ist das Schreiben ausgeschaltet) |
| **Delphi** | function LS_SetWriteLogTextFN(AWriteLogText: LongBool; ALogFN: PChar): Integer;<br>function LSX_SetWriteLogTextFN(LSID: Integer; AWriteLogText: LongBool; ALogFN: PChar): Integer; |
| **C++** | int SetWriteLogTextFN (BOOL bAWriteLogText, char *pcALogFN); |
| **LabView:** |  |
| **Parameter** | AWriteLogText: true => Protokolldatei schreiben<br>ALogFN: Dateiname der Protokolldatei |
| **Beispiel** | LS.SetWriteLogTextFN(true, „C:\Temp\prot.txt“); |

**Steuerungs-Info:**

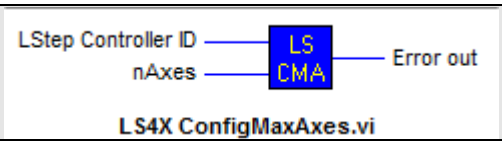
| LS_GetSerialNr | |
|---|---|
| **Beschreibung:** | Seriennummer der Steuerung auslesen |
| **Delphi:** | function LS_GetSerialNr(SerialNr: PChar; MaxLen: Integer): Integer;<br>function LSX_GetSerialNr(LSID: Integer; SerialNr: PChar; MaxLen: Integer): Integer; |
| **C++:** | int GetSerialNr (char *pcSerialNr,int lMaxLen); |
| **LabView:** |  |
| **Parameter:** | SerialNr: Zeiger auf einen Puffer, in dem die Seriennummer zurückgegeben wird<br>MaxLen: Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen |
| **Beispiel:** | LS.GetSerialNr(pcSerialNr, 256); |

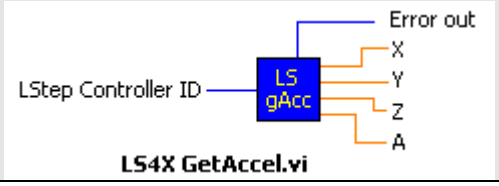
| LS_GetVersionStr | |
|---|---|
| **Beschreibung:** | liefert die aktuelle Versionsnummer der Firmware zurück |
| **Delphi:** | function LS_GetVersionStr(Vers: PChar; MaxLen: Integer): Integer;<br>function LSX_GetVersionStr(LSID: Integer; Vers: PChar; MaxLen: Integer): Integer; |
| **C++:** | int GetVersionStr (char *pcVers,int lMaxLen); |
| **LabView:** |  |
| **Parameter:** | Stat: Zeiger auf einen Puffer, in dem der Versions-String zurückgegeben wird<br>MaxLen: Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen |
| **Beispiel:** | LS.GetVersionStr(pcVers, 64);<br>// Versionsnummer auslesen |

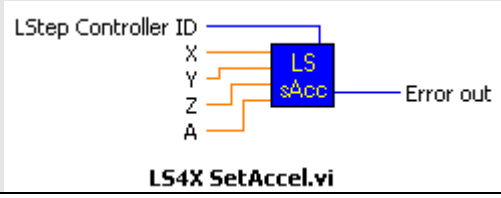
| LS_GetVersionStrDet | |
|---|---|
| **Beschreibung:** | Detaillierte Versionsnummer der Firmware auslesen |
| **Delphi:** | function LS_GetVersionStrDet(VersDet: PChar; MaxLen: Integer): Integer;<br>function LSX_GetVersionStrDet(LSID: Integer; VersDet: PChar; MaxLen: Integer): Integer; |
| **C++:** | int GetVersionStrDet (char *pcVersDet, int lMaxLen); |
| **LabView:** |  |
| **Parameter:** | VersDet: Zeiger auf einen Puffer, in dem der detaillierteVersions-String zurückgegeben wird<br>MaxLen: Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen |
| **Beispiel:** | LS.GetVersionStrDet(pcVersDet, 64);<br>// detaillierte Versionsnummer auslesen |

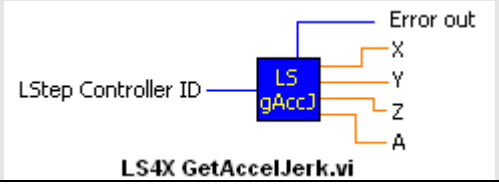
| LS_GetVersionStrInfo | |
|---|---|
| **Beschreibung:** | Liefert detaillierte Informationen zur Versionsnummer |
| **Delphi:** | function LS_ GetVersionStrInfo (VersInfo: PChar; MaxLen: Integer): Integer;<br>function LSX_ GetVersionStrInfo (LSID: Integer; VersInfo: PChar; MaxLen: Integer): Integer; |
| **C++:** | int GetVersionStrInfo (char *pcVersInfo, int lMaxLen); |
| **LabView:** |  |
| **Parameter:** | VersInfo: Zeiger auf einen Puffer, in dem Wochentag.Kalenderwoche.Jahr-fortlaufende Nummer zurückgegeben werden.<br>z. B.: T04.35.02-0004<br>MaxLen: Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen |
| **Beispiel:** | LS.GetVersionStrInfo (pcVersInfo, 64); |

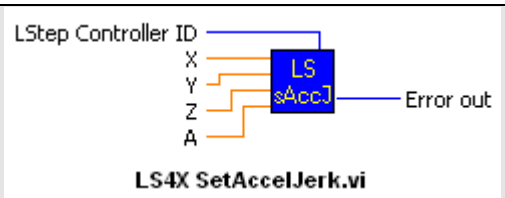
**Einstellungen**

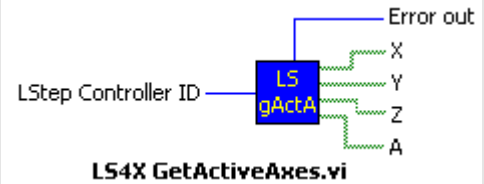
| LS_ConfigMaxAxes | |
|---|---|
| **Beschreibung:** | Konfiguriert Anzahl Achsen |
| **Delphi:** | function LS_ ConfigMaxAxes s (nAxes: Integer): Integer;<br>function LSX_ConfigMaxAxes(LSID: Integer; nAxes: Integer): Integer; |
| **C++:** | int ConfigMaxAxes (int nAxes); |
| **LabView:** |  |
| **Parameter:** | nAxes: Anzahl Achsen, 1 - 4 |
| **Beispiel:** | LS. ConfigMaxAxes (2); // Steuerung hat zwei Achsen |

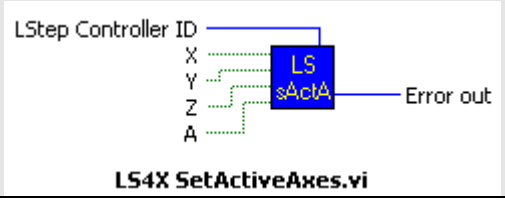
| LS_GetAccel | |
|---|---|
| **Beschreibung:** | Beschleunigung abfragen |
| **Delphi:** | function LS_GetAccel(var X, Y, Z, R: Double): Integer;<br>function LSX_GetAccel(LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetAccel (double *pdX, double *pdY, double *pdZ, double *pdA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z, A: Beschleunigungswerte [m/s² ] |
| **Beispiel:** | LS.GetAccel(&X, &Y, &Z, &A); |

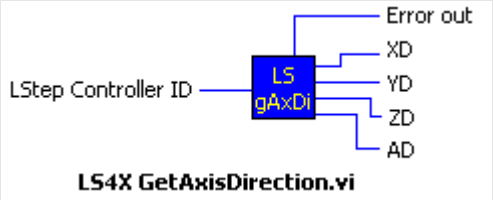
| LS_SetAccel | |
|---|---|
| **Beschreibung:** | Beschleunigung einstellen |
| **Delphi:** | function LS_SetAccel(X, Y, Z, R: Double): Integer;<br>function LSX_SetAccel(LSID: Integer; X, Y, Z, A: Double): Integer; |
| **C++:** | int SetAccel(double dX, double dY, double dZ, double dA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A<br>0.01 – 10.00 [m/s²] |
| **Beispiel:** | LS.SetAccel(1.0, 1.5, 0, 0); |

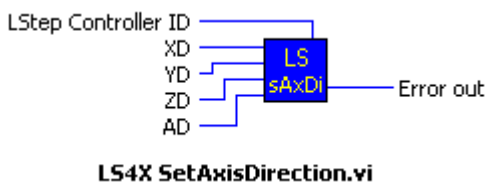
| LS_GetAccelJerk | |
|---|---|
| **Beschreibung:** | Ruck während der Beschleunigung abfragen |
| **Delphi:** | function LS_GetAccelJerk(var XD, YD, ZD, AD: Double): Integer;<br>function LSX_GetAccelJerk(LSID: Integer; var XD, YD, ZD, AD: Double): Integer; |
| **C++:** | int GetAccelJerk(double *pdXD, double *pdYD, double *pdZD, double *pdAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD, AD: Ruckwerte [m/s³] |
| **Beispiel:** | LS.GetAccelJerk(&XD, &YD, &ZD, &AD); |

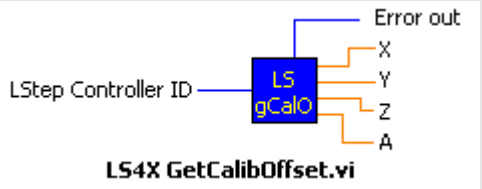
| LS_SetAccelJerk | |
|---|---|
| **Beschreibung:** | Ruck während der Beschleunigung einstellen |
| **Delphi:** | function LS_SetAccelJerk(XD, YD, ZD, AD: Double): Integer;<br>function LSX_SetAccelJerk(LSID: Integer; XD, YD, ZD, AD: Double): Integer; |
| **C++:** | int SetAccelJerk(double dXD, double dYD, double dZD, double dAD); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: 1, 2, 3, . . . . . ∞ [m/s³] (Natürliche Zahlen) |
| **Beispiel:** | LS.SetAccelJerk(1.0, 1.5, 0, 0); |

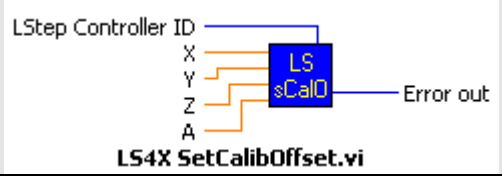
| LS_GetActiveAxes | |
|---|---|
| **Beschreibung:** | Liefert die Achsenfreigabe |
| **Delphi:** | function LS_GetActiveAxes(var Flags: Integer): Integer;<br>function LSX_GetActiveAxes(LSID: Integer; var Flags: Integer): Integer; |
| **C++:** | int GetActiveAxes (int *plFlags); |
| **LabView:** |  |
| **Parameter:** | Flags: 32-bit-Integer, welcher nach Aufruf der Funktion in den Bits 0-4 die Bit-Maske enthält.<br>Bit 0 = 1 → X-Achse freigeschaltet<br>Bit 2 = 0 → Z-Achse nicht freigeschaltet |
| **Beispiel:** | LS.GetActiveAxes(&Flags); |

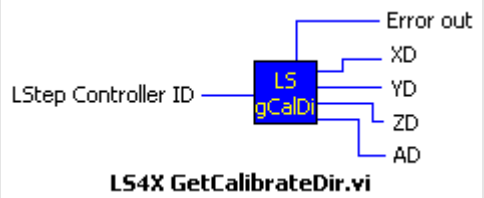
| LS_SetActiveAxes | |
|---|---|
| **Beschreibung:** | Achsenfreigabe |
| **Delphi:** | function LS_SetActiveAxes(Flags: Integer): Integer;<br>function LSX_SetActiveAxes(LSID: Integer; Flags: Integer): Integer; |
| **C++:** | int SetActiveAxes(int Flags); |
| **LabView:** |  |
| **Parameter:** | Flags: Bit-Maske<br>Bit 0 = 1 → X-Achse freigeschaltet<br>Bit 2 = 0 → Z-Achse nicht freigeschaltet |
| **Beispiel:** | LS.SetActiveAxes(3);<br>/* X- und Y-Achse freigeben (Bits 0 u. 1 gesetzt), Z-Achse nicht freigegeben (Bit 2 = 0) */ |

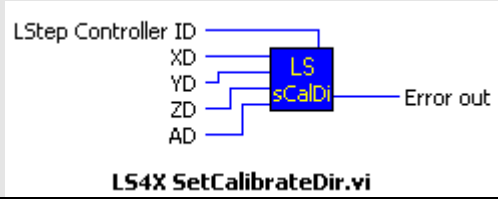
| LS_GetAxisDirection | |
|---|---|
| **Beschreibung** | Drehrichtungs-Umkehr abfragen |
| **Delphi** | function LS_GetAxisDirection(var XD, YD, ZD, AD: Integer): Integer;<br>function LSX_GetAxisDirection(LSID: Integer; var XD, YD, ZD, AD: Integer): Integer; |
| **C++** | int GetAxisDirection (int *plXD, int *plYD, int *plZD, int *plAD); |
| **LabView:** |  |
| **Parameter** | XD, YD, ZD, AD: 32-bit-Integers<br>0 => normale Drehrichtung<br>1 => Drehrichtungs-Umkehr |
| **Beispiel** | LS.GetAxisDirection(&XD, &YD, &ZD, &AD); |

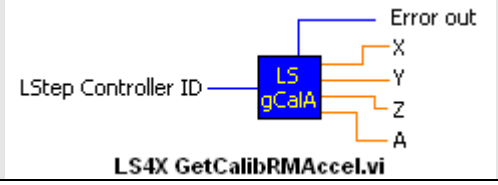
| LS_SetAxisDirection | |
|---|---|
| **Beschreibung** | Drehrichtungs-Umkehr |
| **Delphi** | function LS_SetAxisDirection(XD, YD, ZD, AD: Integer): Integer;<br>function LSX_SetAxisDirection(LSID: Integer; XD, YD, ZD, AD: Integer): Integer; |
| **C++** | int SetAxisDirection (int lXD, int lYD, int lZD, int lAD); |
| **LabView:** |  |
| **Parameter** | XY, YD, ZD, AD:<br>0 => normale Drehrichtung<br>1 => Drehrichtungs-Umkehr |
| **Beispiel** | LS.SetAxisDirection(1, 0, 0, 0); // Drehrichtung der X-Achse umkehren |

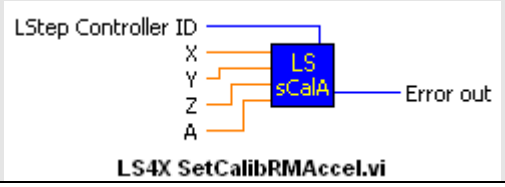
| LS_GetCalibOffset | |
|---|---|
| **Beschreibung:** | Kalibrier-Offset abfragen |
| **Delphi:** | function LS_GetCalibOffset(var X, Y, Z, A: Double): Integer;<br>function LSX_GetCalibOffset(LSID: Integer; var X, Y, Z, R: Double): Integer; |
| **C++:** | int GetCalibOffset (double *pdX, double *pdY, double *pdZ, double *pdR); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z, A: Kalibrier-Offset, abhängig von Dimension. |
| **Beispiel:** | LS.GetCalibOffset(&X, &Y, &Z, &A); |

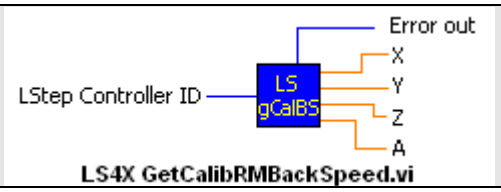
| LS_SetCalibOffset | |
|---|---|
| **Beschreibung:** | Kalibrier-Offset |
| **Delphi:** | function LS_SetCalibOffset(X, Y, Z, A: Double): Integer;<br>function LSX_SetCalibOffset(LSID: Integer; X, Y, Z, R: Double): Integer; |
| **C++:** | int SetCalibOffset (double dX,double dY,double dZ,double dA); |
| **LabView:** |  |
| **Parameter:** | X, y, z und a<br>0 – 32*50000 (32*Spindelsteigung) |
| **Beispiel:** | LS.SetCalibOffset(1, 1, 1, 1);<br>(Die Achsen X, Y und Z werden beim Kalibrieren jeweils 1mm (bei Dim 2 2 2) vom Nullendschalter in Richtung Tischmitte verfahren und dann die Position Null gesetzt (Softwaregrenze).) |

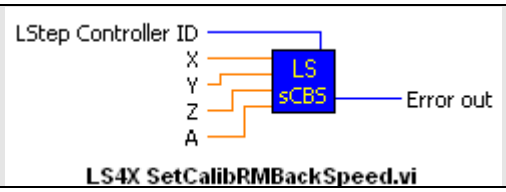
| LS_GetCalibrateDir | |
|---|---|
| **Beschreibung** | Vorzeichen-Umkehr bei Kalibration abfragen |
| **Delphi** | function LS_GetCalibrateDir(var XD, YD, ZD, AD: Integer): Integer;<br>function LSX_GetCalibrateDir(LSID: Integer; var XD, YD, ZD, AD: Integer): Integer; |
| **C++** | int GetCalibrateDir (int *plXD, int *plYD, int *plZD, int *plAD); |
| **LabView:** |  |
| **Parameter** | XD, YD, ZD, AD: 32-bit-Integers<br>0 => keine Vorzeichen-Umkehr<br>1 => Vorzeichen-Umkehr |
| **Beispiel** | LS.GetCalibrateDir(&XD, &YD, &ZD, &AD); |

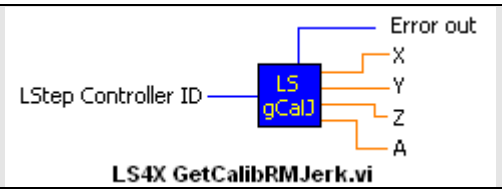
| LS_SetCalibrateDir | |
|---|---|
| **Beschreibung** | Vorzeichen-Umkehr bei Kalibration |
| **Delphi** | function LS_SetCalibrateDir(XD, YD, ZD, AD: Integer): Integer;<br>function LSX_SetCalibrateDir(LSID: Integer; XD, YD, ZD, AD: Integer): Integer; |
| **C++** | int SetCalibrateDir (int lXD, int lYD, int lZD, int lAD); |
| **LabView:** |  |
| **Parameter** | XD, YD, ZD, AD:<br>0 => keine Vorzeichen-Umkehr<br>1 => Vorzeichen-Umkehr |
| **Beispiel** | LS.SetCalibrateDir(1, 1, 0, 0); |

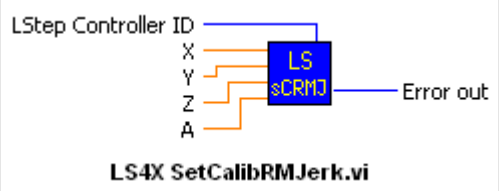
| LS_GetCalibRMAccel | |
|---|---|
| **Beschreibung:** | Beschleunigung während des Kalibriervorgangs abfragen |
| **Delphi:** | function LS_GetCalibRMAccel (var XD, YD, ZD, AD: Double): Integer;<br>function LSX_GetCalibRMAccel (LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetCalibRMAccel (double *pdXD, double *pdYD, double *pdZD, double *pdAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD, AD: Beschleunigungswerte [m/s² ] |
| **Beispiel:** | LS.GetCalibRMAccel (&XD, &YD, &ZD, &AD); |

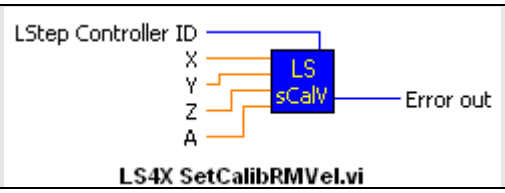
| LS_SetCalibRMAccel | |
|---|---|
| **Beschreibung:** | Beschleunigung während des Kalibriervorgangs einstellen |
| **Delphi:** | function LS_SetCalibRMAccel (XD, YD, ZD, AD: Double): Integer;<br>function LSX_SetCalibRMAccel (LSID: Integer; XD, YD, ZD, AD: Double): Integer; |
| **C++:** | int SetCalibRMAccel (double dXD, double dYD, double dZD, double dAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD und AD: Beschleunigungswerte 0.01 – 20.00 [m/s2] |
| **Beispiel:** | LS.SetCalibRMAccel (1.0, 1.5, 0, 0); |

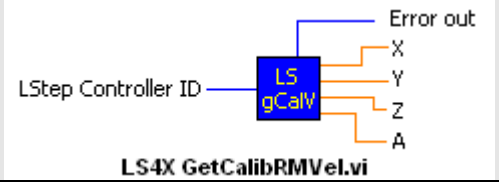
| LS_GetCalibRMBackSpeed | |
|---|---|
| **Beschreibung:** | Verfahrgeschwindigkeiten für das Herausfahren aus den Endschaltern während des Kalibriervorgangs abfragen |
| **Delphi:** | function LS_GetCalibRMBackSpeed (var XD, YD, ZD, AD: Double): Integer;<br>function LSX_GetCalibRMBackSpeed (LSID: Integer; 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 [U/s] |
| **Beispiel:** | LS.GetCalibRMBackSpeed (&XD, &YD, &ZD, &AD); |

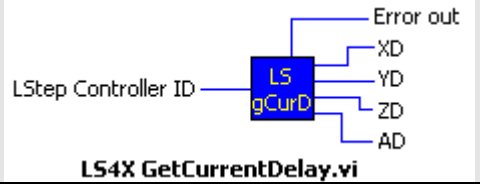
| LS_SetCalibRMBackSpeed | |
|---|---|
| **Beschreibung:** | Verfahrgeschwindigkeiten für das Herausfahren aus den Endschaltern während des Kalibriervorgangs einstellen |
| **Delphi:** | function LS_SetCalibRMBackSpeed (XD, YD, ZD, AD: Double): Integer;<br>function LSX_SetCalibRMBackSpeed (LSID: Integer; XD, YD, ZD, AD: Double): Integer; |
| **C++:** | int SetCalibRMBackSpeed (double dXD, double dYD, double dZD, double dAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD und AD: Geschwindigkeit, 5..100 [U/s] |
| **Beispiel:** | LS.SeCalibRMBackSpeed (1.0, 15.0, 0, 0); |

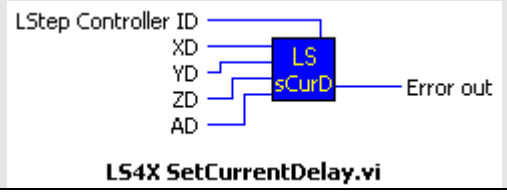
| LS_GetCalibRMJerk | |
|---|---|
| **Beschreibung:** | Ruck während des Kalibriervorgangs abfragen |
| **Delphi:** | function LS_GetCalibRMJerk(var XD, YD, ZD, AD: Double): Integer;<br>function LSX_GetCalibRMJerk(LSID: Integer; var XD, YD, ZD, AD: Double): Integer; |
| **C++:** | int GetCalibRMJerk (double *pdXD, double *pdYD, double *pdZD, double *pdAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD, AD: Ruckwerte [m/s³] |
| **Beispiel:** | LS.GetCalibRMJerk(&XD, &YD, &ZD, &AD); |

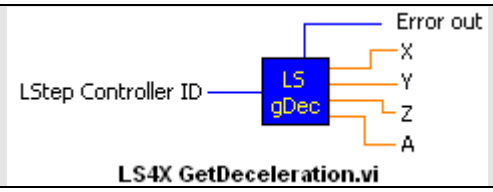
| LS_SetCalibRMJerk | |
|---|---|
| **Beschreibung:** | Ruck während des Kalibriervorgangs |
| **Delphi:** | function LS_SetCalibRMJerk(XD, YD, ZD, AD: Double): Integer;<br>function LSX_SetCalibRMJerk(LSID: Integer; XD, YD, ZD, AD: Double): Integer; |
| **C++:** | int SetCalibRMJerk(double dXD, double dYD, double dZD, double dAD); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: 1, 2, 3, . . . . . ∞ [m/s³] (Natürliche Zahlen) |
| **Beispiel:** | LS.SetCalibRMJerk(1.0, 1.5, 0, 0); |

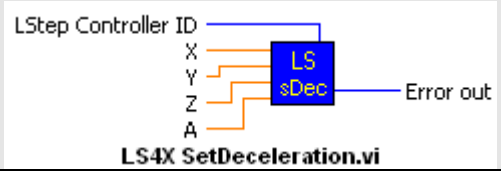
| LS_SetCalibRMVel | |
|---|---|
| **Beschreibung:** | Verfahrgeschwindigkeiten während des Kalibriervorgangs einstellen |
| **Delphi:** | function LS_SetCalibRMVel (XD, YD, ZD, AD: Double): Integer;<br>function LSX_SetCalibRMVel (LSID: Integer; XD, YD, ZD, AD: Double): Integer; |
| **C++:** | int SetCalibRMVel (double dXD, double dYD, double dZD, double dAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD und AD: Geschwindigkeit [U/s] |
| **Beispiel:** | LS.SeCalibRMVel (1.0, 15.0, 0, 0); |

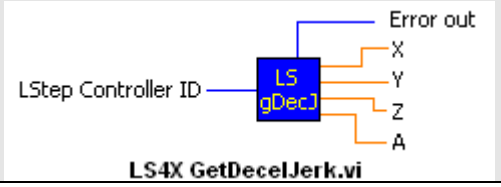
| LS_GetCalibRMVel | |
|---|---|
| **Beschreibung:** | Verfahrgeschwindigkeiten während des Kalibriervorgangs abfragen |
| **Delphi:** | function LS_GetCalibRMVel(var XD, YD, ZD, AD: Double): Integer;<br>function LSX_GetCalibRMVel(LSID: Integer; var XD, YD, ZD, AD: Double): Integer; |
| **C++:** | int GetCalibRMVel (double *pdXD, double *pdYD, double *pdZD, double *pdAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD, AD: Geschwindigkeitswerte [U/s] |
| **Beispiel:** | LS.GetCalibRMVel(&XD, &YD, &ZD, &AD); |

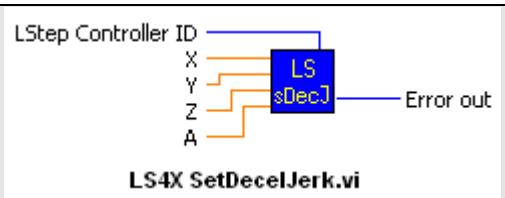
| LS_GetCurrentDelay | |
|---|---|
| **Beschreibung:** | Gibt an Zeitverzögerung für die Stromabsenkung |
| **Delphi:** | function LS_GetCurrentDelay(var X, Y, Z, R: Integer): Integer;<br>function LSX_GetCurrentDelay(LSID: Integer; var X, Y, Z, R: Integer): Integer; |
| **C++:** | int GetCurrentDelay (int *plX, int *plY, int *plZ, int *plR); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z, R: Zeitverzögerung in ms |
| **Beispiel:** | LS.SetCurrentDelay(&X, &Y, &Z, &A); |

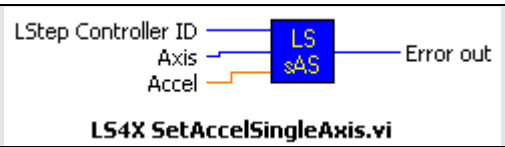
| LS_SetCurrentDelay | |
|---|---|
| **Beschreibung:** | Zeitverzögerung für die Stromabsenkung |
| **Delphi:** | function LS_SetCurrentDelay(X, Y, Z, R: Integer): Integer;<br>function LSX_SetCurrentDelay(LSID: Integer; X, Y, Z, R: Integer): Integer; |
| **C++:** | int SetCurrentDelay (int lX, int lY, int lZ, int lR); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z, R: 0-10000 [ms] |
| **Beispiel:** | LS.SetCurrentDelay(100, 300, 1000, 0) ; |

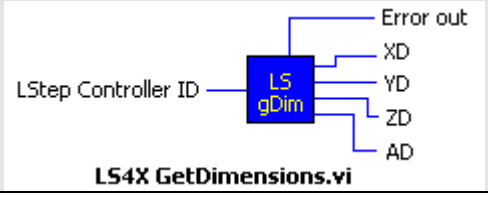
| LS_GetDeceleration | |
|---|---|
| **Beschreibung:** | Verzögerung abfragen |
| **Delphi:** | function LS_GetDeceleration (var XD, YD, ZD, AD: Double): Integer;<br>function LSX_GetDeceleration (LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetDeceleration (double *pdXD, double *pdYD, double *pdZD, double *pdAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD, AD: Verzögerungswerte [m/s² ] |
| **Beispiel:** | LS.GetDeceleration (&XD, &YD, &ZD, &AD); |

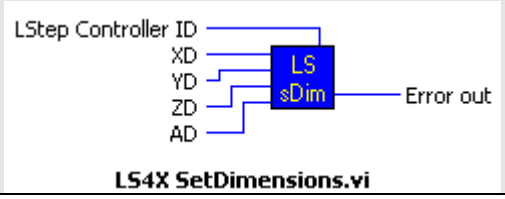
| LS_SetDeceleration | |
|---|---|
| **Beschreibung:** | Verzögerung einstellen |
| **Delphi:** | function LS_SetDeceleration (XD, YD, ZD, AD: Double): Integer;<br>function LSX_SetDeceleration (LSID: Integer; XD, YD, ZD, AD: Double): Integer; |
| **C++:** | int SetDeceleration (double dXD, double dYD, double dZD, double dAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD und AD: Verzögerungswerte 0.01 – 20.00 [m/s2] |
| **Beispiel:** | LS.SetDeceleration (1.0, 1.5, 0, 0); |

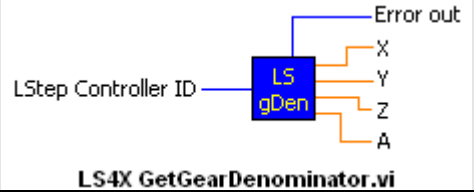
| LS_GetDecelJerk | |
|---|---|
| **Beschreibung:** | Ruck während der Verzögerung abfragen |
| **Delphi:** | function LS_GetDecelJerk(var XD, YD, ZD, AD: Double): Integer;<br>function LSX_GetDecelJerk(LSID: Integer; var XD, YD, ZD, AD: Double): Integer; |
| **C++:** | int GetDecelJerk(double *pdXD, double *pdYD, double *pdZD, double *pdAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD, AD: Ruckwerte [m/s³] |
| **Beispiel:** | LS.GetDecelJerk(&XD, &YD, &ZD, &AD); |

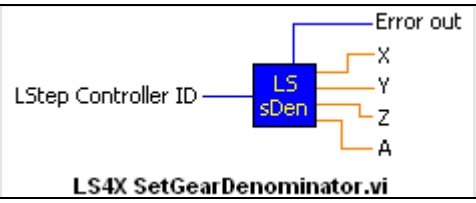
| LS_SetDecelJerk | |
|---|---|
| **Beschreibung:** | Ruck während der Verzögerung einstellen |
| **Delphi:** | function LS_SetDecelJerk(XD, YD, ZD, AD: Double): Integer;<br>function LSX_SetDecelJerk(LSID: Integer; XD, YD, ZD, AD: Double): Integer; |
| **C++:** | int SetDecellJerk(double dXD, double dYD, double dZD, double dAD); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: 1, 2, 3, . . . . . ∞ [m/s³] (Natürliche Zahlen) |
| **Beispiel:** | LS.SetDecelJerk(1.0, 1.5, 0, 0); |

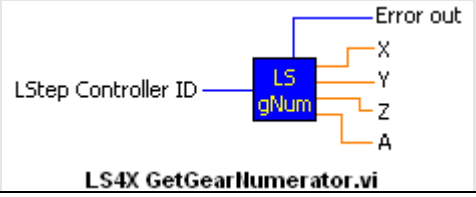
| LS_SetDecelSingleAxis | |
|---|---|
| **Beschreibung:** | Verzögerung einstellen |
| **Delphi:** | function LS_SetDecelSingleAxis(Axis: Integer; Accel: Double): Integer;<br>function LSX_SetDecelSingleAxis(LSID: Integer; Axis: Integer; Accel: Double): Integer; |
| **C++:** | int SetDecelSingleAxis (int lAxis,double dAccel); |
| **LabView:** |  |
| **Parameter:** | Axis: (X, Y, Z, A numeriert von 1 bis 4)<br>Accel Verzögerung 0.01 – 10.00 [m/s²] |
| **Beispiel:** | LS.SetDecelSingleAxis(1, 1.0);<br>// Verzögerung X-Achse 1.0 m/s² |

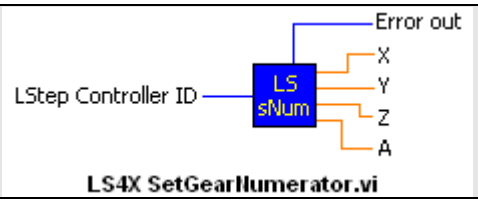
| LS_GetDimensions | |
|---|---|
| **Beschreibung:** | Abfrage der Dimensionen der Achsen |
| **Delphi:** | function LS_GetDimensions(var XD, YD, ZD, AD: Integer): Integer;<br>function LSX_GetDimensions(LSID: Integer; var XD, YD, ZD, AD: Integer): Integer; |
| **C++:** | int GetDimensions (int *plXD, int *plYD, int *plZD, int *plAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD, AD: Dimensionswerte |
| | 0 ➔ Microsteps<br>1 ➔ µm<br>2 ➔ Millimeter<br>3 ➔ Grad<br>4 ➔ Umdrehungen |
| **Beispiel:** | LS. GetDimensions (&XD, &YD, &ZD, &AD); |

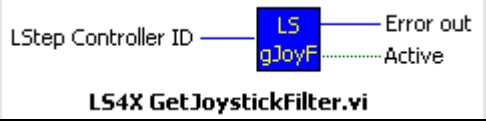
| LS_SetDimensions | |
|---|---|
| **Beschreibung:** | Dimensionen der Achsen einstellen |
| **Delphi:** | function LS_SetDimensions(XD, YD, ZD, AD: Integer): Integer;<br>function LSX_SetDimensions(LSID: Integer; XD, YD, ZD, AD: Integer): Integer; |
| **C++:** | int SetDimensions (int lXD,int lYD,int lZD,int lAD); |
| **LabView:** |  <br>**LS4X SetDimensions.vi** |
| **Parameter:** | Dimension von X, Y, Z und A-Achse:<br>0 ➔ Microsteps<br>1 ➔ µm<br>2 ➔ Millimeter<br>3 ➔ Grad<br>4 ➔ Umdrehungen |
| **Beispiel:** | LS.SetDimensions(3, 2, 2);<br>// X-Achse in Grad; Y und Z in mm |

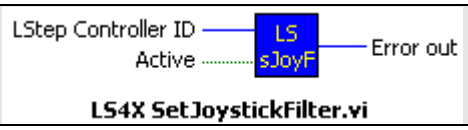
| LS_GetGearDenominator | |
|---|---|
| **Beschreibung** | Nenner der Getriebeübersetzung abfragen |
| **Delphi** | function LS_GetGearDenominator (var X, Y, Z, A: Integer): Integer;<br>function LSX_ GetGearDenominator (LSID: Integer; var X, Y, Z, A: Integer): Integer; |
| **C++** | int GetGearDenominator (int *plX, int *plY, int *plZ, int *plA); |
| **LabView:** |  <br>**LS4X GetGearDenominator.vi** |
| **Parameter** | X, Y, Z, A: Nominator (Nenner) der Getriebeübersetzung |
| **Beispiel** | LS. GetGearDenominator (&X, &Y, &Z, &A); |

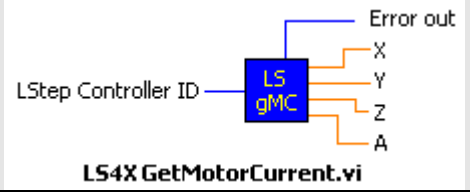
| LS_SetGearDenominator | |
|---|---|
| **Beschreibung** | Nenner der Getriebeübersetzung einstellen |
| **Delphi** | function LS_ SetGearDenominator (X, Y, Z, A: Integer): Integer;<br>function LSX_ SetGearDenominator (LSID: Integer; X, Y, Z, A: Integer): Integer; |
| **C++** | int SetGearDenominator (int lX, int lY, int lZ, int lA); |
| **LabView:** |  |
| **Parameter** | X, Y, Z, A: 1, 2, 3, . . . . . ∞ (Natürliche Zahlen) |
| **Beispiel** | LS. SetGearDenominator (2, 0, 0, 0); // Getriebeübersetzung 2/γ bei x |

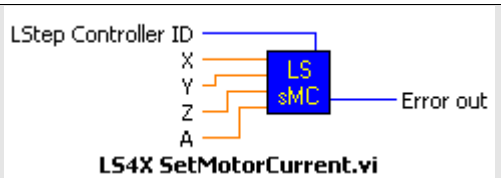
| LS_GetGearNumerator | |
|---|---|
| **Beschreibung** | Zähler der Getriebeübersetzung abfragen |
| **Delphi** | function LS_GetGearNumerator (var X, Y, Z, A: Integer): Integer;<br>function LSX_ GetGearNumerator (LSID: Integer; var X, Y, Z, A: Integer): Integer; |
| **C++** | int GetGearNumerator (int *plX, int *plY, int *plZ, int *plA); |
| **LabView:** |  |
| **Parameter** | X, Y, Z, A: Numerator (Zähler) der Getriebeübersetzung |
| **Beispiel** | LS. GetGearNumerator (&X, &Y, &Z, &A); |

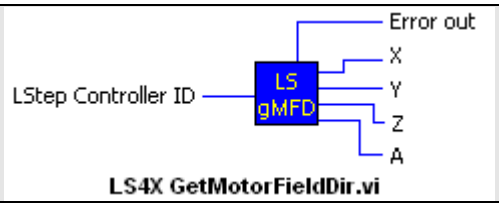
| LS_SetGearNumerator | |
|---|---|
| **Beschreibung** | Zähler der Getriebeübersetzung einstellen |
| **Delphi** | function LS_ SetGearNumerator (X, Y, Z, A: Integer): Integer;<br>function LSX_ SetGearNumerator (LSID: Integer; X, Y, Z, A: Integer): Integer; |
| **C++** | int SetGearNumerator (int lX, int lY, int lZ, int lA); |
| **LabView:** |  |
| **Parameter** | X, Y, Z, A: 1, 2, 3, . . . . . ∞ (Natürliche Zahlen) |
| **Beispiel** | LS. SetGearNumerator (4, 0, 0, 0); |

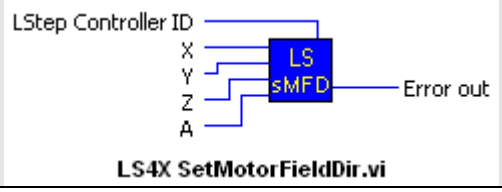
| LS_GetJoystickFilter | |
|---|---|
| **Beschreibung:** | Gibt an, ob die Filterung und Hysterese im Joystick-Betrieb aktiviert ist. |
| **Delphi:** | function LS_GetJoystickFilter(var bActive: LongBool): Integer;<br>function LSX_ GetJoystickFilter (LSID: Integer; var bActive: LongBool): Integer; |
| **C++:** | int GetJoystickFilter (BOOL *pbActive); |
| **LabView:** |  |
| **Parameter:** | bActive: True – Filterung aktiviert<br>False – deaktiviert |
| **Beispiel:** | LS. SetJoystickFilter (&Active); |

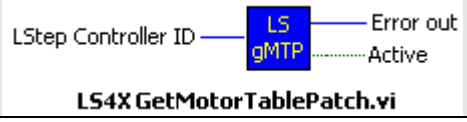
| LS_SetJoystickFilter | |
|---|---|
| **Beschreibung:** | Aktivierung/Deaktivierung der Filterung und Hysterese im Joystick-Betrieb. |
| **Delphi:** | function LS_SetJoystickFilter(bActive: LongBool): Integer;<br>function LSX_ SetJoystickFilter (LSID: Integer; bActive: LongBool): Integer; |
| **C++:** | int SetJoystickFilter (BOOL bActive); |
| **LabView:** |  |
| **Parameter:** | bActive: True – Aktivierung der Filterung<br>False – Deaktivierung |
| **Beispiel:** | LS. SetJoystickFilter (True); |

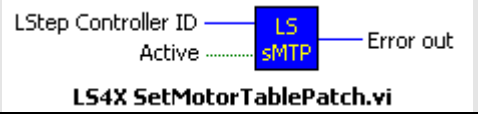
| LS_GetMotorCurrent | |
|---|---|
| **Beschreibung:** | Motorstrom abfragen |
| **Delphi:** | function LS_GetMotorCurrent(var X, Y, Z, A: Double): Integer;<br>function LSX_GetMotorCurrent(LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetMotorCurrent (double *pdX, double *pdY, double *pdZ, double *pdA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z, A: Motorstrom [A] |
| **Beispiel:** | LS.GetMotorCurrent(&X, &Y, &Z, &A); |

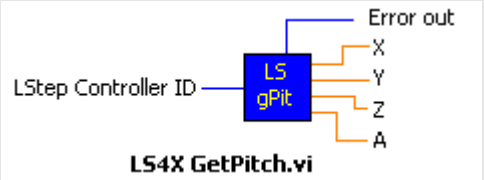
| LS_SetMotorCurrent | |
|---|---|
| **Beschreibung:** | Motorstrom einstellen |
| **Delphi:** | function LS_SetMotorCurrent(X, Y, Z, A: Double): Integer;<br>function LSX_SetMotorCurrent(LSID: Integer; X, Y, Z, A: Double): Integer; |
| **C++:** | int SetMotorCurrent (double dX,double dY,double dZ,double dA); |
| **LabView:** |  |
| **Parameter:** | Motorstrom X, Y, Z, A-Achse [A] |
| **Beispiel:** | LS.SetMotorCurrent(1.5, 1.5, 1.0, 1.0);<br>// Motorstrom X u. Y 1.5 Ampere; Z u. A 1.0 Ampere |

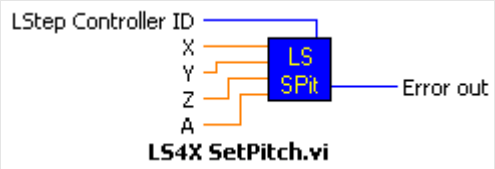
| LS_GetMotorFieldDir | |
|---|---|
| **Beschreibung** | Drehrichtung des Motors abfragen |
| **Delphi** | function LS_GetMotorFieldDir (var X, Y, Z, A: Integer): Integer;<br>function LSX_ GetMotorFieldDir (LSID: Integer; var X, Y, Z, A: Integer): Integer; |
| **C++** | int GetMotorFieldDir (int *plX, int *plY, int *plZ, int *plA); |
| **LabView:** |  |
| **Parameter** | X, Y, Z, A: 32-bit-Integers<br>0 => normale Drehrichtung<br>1 => Drehrichtungs-Umkehr |
| **Beispiel** | LS. GetMotorFieldDir (&X, &Y, &Z, &A); |

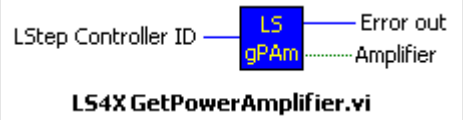
| LS_SetMotorFieldDir | |
|---|---|
| **Beschreibung** | Drehrichtung des Motors einstellen |
| **Delphi** | function LS_ SetMotorFieldDir (X, Y, Z, A: Integer): Integer;<br>function LSX_ SetMotorFieldDir (LSID: Integer; X, Y, Z, A: Integer): Integer; |
| **C++** | int SetMotorFieldDir (int lX, int lY, int lZ, int lA); |
| **LabView:** |  |
| **Parameter** | X, Y, Z, A:<br>0 => normale Drehrichtung<br>1 => Drehrichtungs-Umkehr |
| **Beispiel** | LS. SetMotorFieldDir (1, 0, 0, 0); // Drehrichtung der X-Achse umkehren |

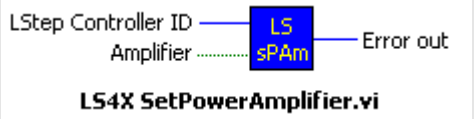
| LS_GetMotorTablePatch | |
|---|---|
| **Beschreibung:** | Gibt an, ob die Korrekturtabelle aktiviert ist. |
| **Delphi:** | function LS_GetMotorTablePatch(bActive: var LongBool): Integer;<br>function LSX_ GetMotorTablePatch (LSID: Integer; var bActive: LongBool): Integer; |
| **C++:** | int GetMotorTablePatch (BOOL *pbActive); |
| **LabView:** |  |
| **Parameter:** | bActive: True – Tabelle ist akktiviert<br>False – Deaktiviert |
| **Beispiel:** | LS. GetMotorTablePatch (&Active); |

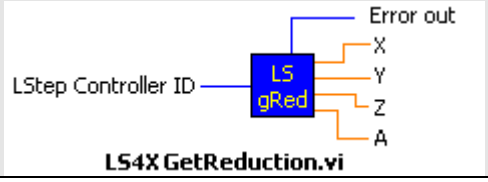
| LS_SetMotorTablePatch | |
|---|---|
| **Beschreibung:** | Die Korrekturtabelle wird aktiviert.<br>Die Korrekturtabelle wurde für einen Sondermotor durch Meßung ermittelt. Korrekturtabellen können auf Kundenwunsch ermittelt werden. |
| **Delphi:** | function LS_SetMotorTablePatch(bActive: LongBool): Integer;<br>function LSX_ SetMotorTablePatch (LSID: Integer; bActive: LongBool): Integer; |
| **C++:** | int SetMotorTablePatch (BOOL bActive); |
| **LabView:** |  |
| **Parameter:** | bActive: True – Aktivierung der Korrekturtabelle<br>False – Deaktivierung |
| **Beispiel:** | LS. SetMotorTablePatch (True); |

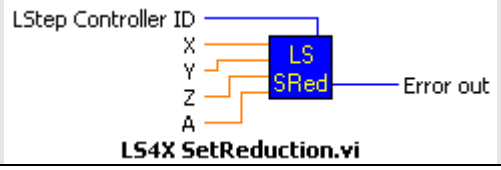
| LS_GetPitch | |
|---|---|
| **Beschreibung:** | liefert Spindelsteigung |
| **Delphi:** | function LS_GetPitch(var X, Y, Z, R: Double): Integer;<br>function LSX_GetPitch(LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetPitch (double *pdX, double *pdY, double *pdZ, double *pdA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z, A: Spindelsteigungen [mm] |
| **Beispiel:** | LS. GetPitch (&X, &Y, &Z, &A); |

| LS_SetPitch | |
|---|---|
| **Beschreibung:** | Spindelsteigung setzen |
| **Delphi:** | function LS_SetPitch(X, Y, Z, R: Double): Integer;<br>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 und A<br>0.001 – 68 [mm] |
| **Beispiel:** | LS.SetPitch(4, 4, 4, 4);<br>// Spindelsteigungen aller Achsen auf 4 mm setzen |

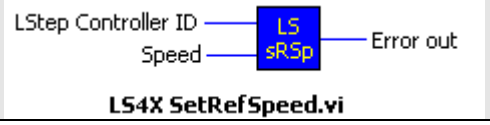
| LS_GetPowerAmplifier | |
|---|---|
| **Beschreibung:** | Gibt an, ob die Endstufen bei LS44 ein- oder ausgeschaltet sind.<br>Diesen Befehl gibt es nur bei LS44-Steuerung. |
| **Delphi:** | function LS_GetPowerAmplifier (bAmplifier: var LongBool): Integer;<br>function LSX_ GetPowerAmplifier (LSID: Integer; var bAmplifier: LongBool): Integer; |
| **C++:** | int GetPowerAmplifier (BOOL *pbAmplifier); |
| **LabView:** |  |
| **Parameter:** | Amplifier: True – die Endstufen sind eingeschaltet<br>False – ausschaltet |
| **Beispiel:** | LS. GetPowerAmplifier (&Amplifier); |

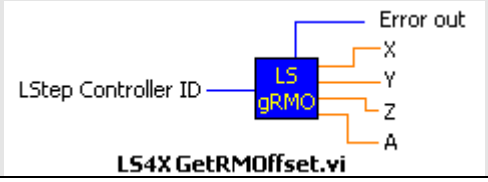
| LS_SetPowerAmplifier | |
|---|---|
| **Beschreibung:** | Schaltet bei LS44 die Endstufen Ein/Aus.<br>Diesen Befehl gibt es nur bei LS44-Steuerung. |
| **Delphi:** | function LS_SetPowerAmplifier (bAmplifier: LongBool): Integer;<br>function LSX_ SetPowerAmplifier (LSID: Integer; bAmplifier: LongBool): Integer; |
| **C++:** | int SetPowerAmplifier (BOOL bAmplifier); |
| **LabView:** |  |
| **Parameter:** | bAmplifier: True – Ein<br>False – Aus |
| **Beispiel:** | LS. SetPowerAmplifier (True); // Die Endstufen einschalten |

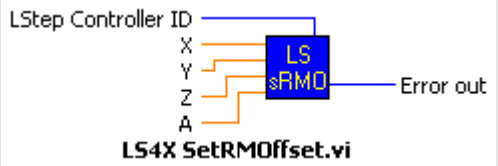
| LS_GetReduction | |
|---|---|
| **Beschreibung:** | Stromabsenkung abfragen |
| **Delphi:** | function LS_GetReduction(var X, Y, Z, R: Double): Integer;<br>function LSX_GetReduction(LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetReduction (double *pdX, double *pdY, double *pdZ, double *pdA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z, A: Stromabsenkung |
| **Beispiel:** | LS.GetReduction(&X, &Y, &Z, &A); |

| LS_SetReduction | |
|---|---|
| **Beschreibung:** | Stromabsenkung einstellen<br>Im Ruhezustand wird der Motornennstrom auf das parametrierte Verhältnis reduziert. |
| **Delphi:** | function LS_SetReduction(X, Y, Z, R: Double): Integer;<br>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 und A<br>0 – 1.0 |
| **Beispiel:** | LS.SetReduction(0.1, 0.7, 0.5, 0.5);<br>/* Ruhestrom X-Achse = 0.1*Nennstrom; Y-Achse = 0.7*Nennstrom ... */ |

| LS_GetRefSpeed | |
|---|---|
| **Beschreibung:** | Liest die Umdrehungsgeschwindigkeit, mit der die Achsen beim Suchen nach der Referenzmarke gefahren werden. Die Geschwindigkeit entspricht dem ausgegebenen Wert * 0.01 U/s. |
| **Delphi:** | function LS_GetRefSpeed(var lSpeed: Integer): Integer;<br>function LSX_ GetRefSpeed (LSID: Integer; var lSpeed: Integer): Integer; |
| **C++:** | int GetRefSpeed (int *plSpeed); |
| **LabView:** |  |
| **Parameter:** | lSpeed: Geschwindigkeitswert |
| **Beispiel:** | LS. GetRefSpeed (&lSpeed); |

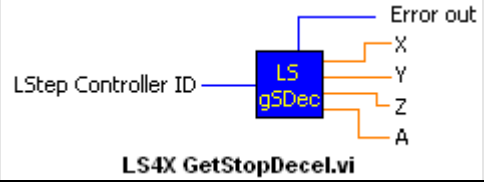
| LS_SetRefSpeed | |
|---|---|
| **Beschreibung:** | Setzt die Umdrehungsgeschwindigkeit, mit der die Achsen beim Suchen nach der Referenzmarke gefahren werden. Die Geschwindigkeit entspricht dem angegebenen Wert * 0.01 U/s. |
| **Delphi:** | function LS_SetRefSpeed(lSpeed: Integer): Integer;<br>function LSX_ SetRefSpeed (LSID: Integer; lSpeed: Integer): Integer; |
| **C++:** | int SetRefSpeed (int lSpeed); |
| **LabView:** |  |
| **Parameter:** | lSpeed: Geschwindigkeit, Wertebereich 0 bis 100 |
| **Beispiel:** | LS. SetRefSpeed (10); //Beim Suchen nach der Referenzmarke wird mit 0.1 U/s gefahren. |

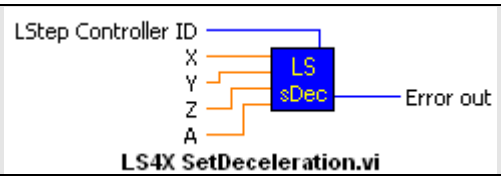
| LS_GetRMOffset | |
|---|---|
| **Beschreibung:** | RM-Offset abfragen |
| **Delphi:** | function LS_GetRMOffset(var X, Y, Z, A: Double): Integer;<br>function LSX_GetRMOffset(LSID: Integer; var X, Y, Z, R: Double): Integer; |
| **C++:** | int GetRMOffset (double *pdX, double *pdY, double *pdZ, double *pdR); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: RM-Offset, abhängig von Dimension |
| **Beispiel:** | LS.GetRMOffset(&X, &Y, &Z, &A); |

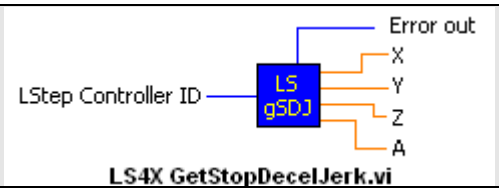
| LS_SetRMOffset | |
|---|---|
| **Beschreibung:** | RM-Offset |
| **Delphi:** | function LS_SetRMOffset(X, Y, Z, A: Double): Integer;<br>function LSX_SetRMOffset(LSID: Integer; X, Y, Z, R: Double): Integer; |
| **C++:** | int SetRMOffset (double dX,double dY,double dZ,double dA); |
| **LabView:** |  |
| **Parameter:** | X, y, z und a<br>0 – 32*50000 (32*Spindelsteigung) |
| **Beispiel:** | LS.SetRMOffset(1, 1, 1, 1);<br>(Die Achsen X, Y und Z werden beim Tischhub messen jeweils 1mm (bei Dim 2 2 2) vom Endendschalter in Richtung Tischmitte verfahren und dann die Softwaregrenze gesetzt. |

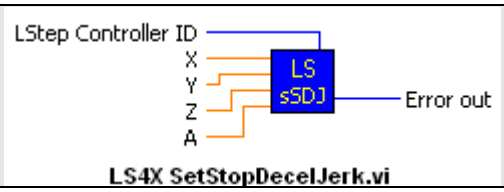
| LS_GetSpeedPoti | |
|---|---|
| **Beschreibung:** | Gibt an ob Potentiometer Ein oder Aus ist. |
| **Delphi:** | function LS_GetSpeedPoti(var SpePoti: LongBool): Integer;<br>function LSX_GetSpeedPoti(LSID: Integer; var SpePoti: LongBool): Integer; |
| **C++:** | int GetSpeedPoti (BOOL *pbSpePoti); |
| **LabView:** |  |
| **Parameter:** | Das SpePoti Flag gibt an, ob Potentiometer Ein oder Aus ist. |
| **Beispiel:** | LS.GetSpeedPoti(&flag); |

| LS_SetSpeedPoti | |
|---|---|
| **Beschreibung:** | Potentiometer Ein/Aus |
| **Delphi:** | function LS_SetSpeedPoti(SpeedPoti: LongBool): Integer;<br>function LSX_SetSpeedPoti(LSID: Integer; SpeedPoti: LongBool): Integer; |
| **C++:** | int SetSpeedPoti (BOOL SpeedPoti); |
| **LabView:** |  |
| **Parameter:** | Bei SpeedPoti = false wird die vorgegebene Geschwindigkeit (vel) als Verfahrgeschwindigkeit genutzt. Bei SpeedPoti = true wird die vorgegebene Geschwindigkeit (vel), in Abhängigkeit von der Stellung des Potentiometers, prozentual genutzt. |
| **Beispiel:** | LS.SetSpeedPoti(true);<br>// Poti An |

| LS_GetStopDecel | |
|---|---|
| **Beschreibung:** | Stoppeingang Bremsbeschleunigung abfragen |
| **Delphi:** | function LS_GetStopDecel (var XD, YD, ZD, AD: Double): Integer;<br>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 [m/s² ] |
| **Beispiel:** | LS.GetStopDecel (&XD, &YD, &ZD, &AD); |

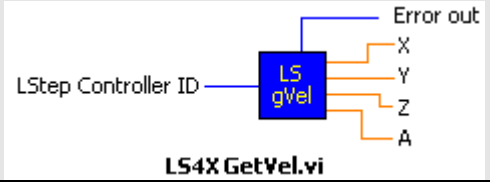
| LS_SetStopDecel | |
|---|---|
| **Beschreibung:** | Den Wert einstellen, mit dem die Achsen im Falle eines Stoppsignals bremsen sollen |
| **Delphi:** | function LS_SetStopDecel (XD, YD, ZD, AD: Double): Integer;<br>function LSX_ SetStopDecel (LSID: Integer; XD, YD, ZD, AD: Double): Integer; |
| **C++:** | int SetStopDecel (double dXD, double dYD, double dZD, double dAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD und AD: Verzögerungswerte 0.01 – 20.00 [m/s2] |
| **Beispiel:** | LS. SetStopDecel (1.0, 1.5, 0, 0); |

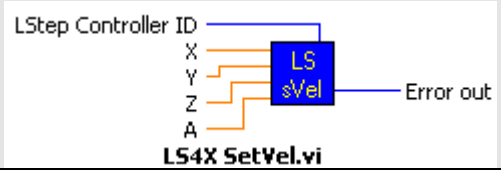
| LS_GetStopDecelJerk | |
|---|---|
| **Beschreibung:** | Ruck während Verzögerung des Systems im Falle eines Notstopsignals abfragen |
| **Delphi:** | function LS_GetStopDecelJerk (var XD, YD, ZD, AD: Double): Integer;<br>function LSX_GetStopDecelJerk (LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetStopDecelJerk (double *pdXD, double *pdYD, double *pdZD, double *pdAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD, AD: Ruckwerte [m/s³] |
| **Beispiel:** | LS.GetStopDecelJerk (&XD, &YD, &ZD, &AD); |

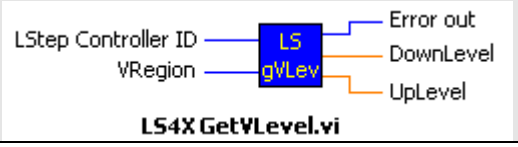
| LS_SetStopDecelJerk | |
|---|---|
| **Beschreibung:** | Ruck während Verzögerung des Systems im Falle eines Notstopsignals einstellen |
| **Delphi:** | function LS_SetStopDecelJerk (XD, YD, ZD, AD: Double): Integer;<br>function LSX_ SetStopDecelJerk (LSID: Integer; XD, YD, ZD, AD: Double): Integer; |
| **C++:** | int SetStopDecelJerk (double dXD, double dYD, double dZD, double dAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD und AD: 1, 2, 3, . . . . . ∞ [m/s³] (Natürliche Zahlen) |
| **Beispiel:** | LS. SetStopDecelJerk (1.0, 1.5, 0, 0); |

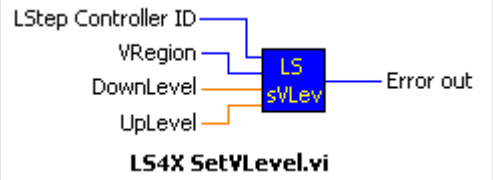
| LS_GetStopPolarity | |
|---|---|
| **Beschreibung:** | Stopeingang Polarität lesen. |
| **Delphi:** | function LS_GetStopPolarity(var bHighActiv: LongBool): Integer;<br>function LSX_ GetStopPolarity (LSID: Integer; var bHighActiv: LongBool): Integer; |
| **C++:** | int GetStopPolarity (BOOL *pbHighActiv); |
| **LabView:** |  |
| **Parameter:** | bHighActiv: True – Stopeingang highaktiv<br>False – lowaktiv |
| **Beispiel:** | LS. GetStopPolarity (&HighActiv); |

| LS_SetStopPolarity | |
|---|---|
| **Beschreibung:** | Stopeingang Polarität einstellen. Da der Stopeingang einen Pull Up nach 5V hat, muß man bei einem Schließer lowaktiv und bei einem Öffner highaktiv einstellen. |
| **Delphi:** | function LS_SetStopPolarity(bHighActiv: LongBool): Integer;<br>function LSX_ SetStopPolarity (LSID: Integer; bHighActiv: LongBool): Integer; |
| **C++:** | int SetStopPolarity (BOOL bHighActiv); |
| **LabView:** |  |
| **Parameter:** | bHighActiv: True – Stopeingang highaktiv<br>False – lowaktiv |
| **Beispiel:** | LS. SetStopPolarity (False); //Der Stopeingang ist lowaktiv. |

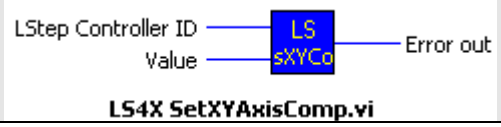
| LS_GetVel | |
|---|---|
| **Beschreibung:** | Geschwindigkeit abfragen |
| **Delphi:** | function LS_GetVel(var X, Y, Z, R: Double): Integer;<br>function LSX_GetVel(LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetVel (double *pdX, double *pdY, double *pdZ, double *pdA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z, A: Geschwindigkeitswerte [U/s] |
| **Beispiel:** | LS.GetVel(&X, &Y, &Z, &A); |

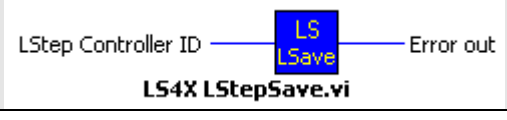
| LS_SetVel | |
|---|---|
| **Beschreibung:** | Geschwindigkeit einstellen |
| **Delphi:** | function LS_SetVel(X, Y, Z, R: Double): Integer;<br>function LSX_SetVel(LSID: Integer; X, Y, Z, A: Double): Integer; |
| **C++:** | int SetVel(double dX, double dY, double dZ, double dA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A<br>0 – maximale Geschwindigkeit [U/s] |
| **Beispiel:** | LS.SetVel(1.0, 15.0, 0, 0); |

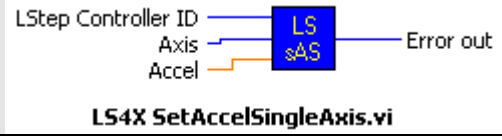
| LS_GetVLevel | |
|---|---|
| **Beschreibung:** | Liefert die Geschwindigkeitsgrenzen von dem angegebenen Geschwindigkeitsbereich. |
| **Delphi:** | function LS_GetVLevel(lVRegion: Integer; var dDownLevel, dUppLevel: Double): Integer;<br>function LSX_ GetVLevel (LSID: Integer; lVRegion: Integer; var dDownLevel, dUppLevel: Double): Integer; |
| **C++:** | int GetVLevel (int lVRegion, double *pdDownLevel, double *pdUppLevel); |
| **LabView:** |  |
| **Parameter:** | lVRegion: Wertebereich 1-4.<br>1 – Erstes/Unteres Geschwindigkeitsbereich<br>2 – Zweites/Mittleres Geschwindigkeitsbereich<br>3 – Drittes/Oberes Geschwindigkeitsbereich<br>4 – Bis zu dieser Geschwindigkeitsgrenze wird die Korrekturtabelle genutzt.<br>dDownLevel : Untere Grenze des Bereichs (bei lVRegion = 4 Geschwindig- keitsgrenze) [U/s]<br>dUppLevel : Obere Grenze des Bereichs (bei lVRegion = 4 hat keine Bedeutung) [U/s] |
| **Beispiel:** | LS. GetVLevel (2, &DownLevel, &UppLevel); // DownLevel = Untere Grenze des zweiten Geschwindigkeitsbereich, UppLevel = Obere Grenze des zweiten Geschwindigkeitsbereich. |

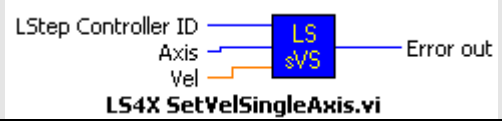
| LS_SetVLevel | |
|---|---|
| **Beschreibung:** | Geschwindigkeitsbereich ausklammern, in denen das System zu Resonanzen neigt. |
| **Delphi:** | function LS_SetVLevel(lVRegion: Integer; dDownLevel, dUppLevel: Double): Integer;<br>function LSX_ SetVLevel (LSID: Integer; lVRegion: Integer; dDownLevel, dUppLevel: Double): Integer; |
| **C++:** | int SetVLevel (int lVRegion, double dDownLevel, double dUppLevel); |
| **LabView:** |  |
| **Parameter:** | lVRegion: Wertebereich 1-4.<br>1 – Erstes/Unteres Geschwindigkeitsbereich<br>2 – Zweites/Mittleres Geschwindigkeitsbereich<br>3 – Drittes/Oberes Geschwindigkeitsbereich<br>4 – Bis zu dieser Geschwindigkeitsgrenze wird die Korrekturtabelle genutzt.<br>dDownLevel : Untere Grenze des Bereichs (bei lVRegion=4 Geschwindig- keitsgrenze) [U/s], Wertebereich 0 – max. Geschwindigkeit.<br>dUppLevel : Obere Grenze des Bereichs (bei lVRegion=4 hat keine Bedeutung) [U/s], Wertebereich 0 – max. Geschwindigkeit. |
| **Beispiel:** | LS. SetVLevel (4, 10.0, 0.0); //Die Korrekturtabelle wirkt bis zu einer Geschwindigkeit von 10 Umdrehungen/s. |

| LS_GetXYAxisComp | |
|---|---|
| **Beschreibung** | Abfrage XY-Achsüberlagerung |
| **Delphi** | function LS_GetXYAxisComp(var Value: Integer): Integer;<br>function LSX_GetXYAxisComp(LSID: Integer; var Value: Integer): Integer; |
| **C++** | int GetXYAxisComp (int *plValue); |
| **LabView:** |  |
| **Parameter** | Value: Modus der Achsüberlagerung (siehe LStep-Dokumentation) |
| **Beispiel** | LS.SetXYAxisComp(&mode) ; |

| LS_SetXYAxisComp | |
|---|---|
| **Beschreibung** | XY-Achsüberlagerung aktivieren |
| **Delphi** | function LS_SetXYAxisComp(Value: Integer): Integer;<br>function LSX_SetXYAxisComp(LSID: Integer; Value: Integer): Integer; |
| **C++** | int SetXYAxisComp (int lValue); |
| **LabView:** |  |
| **Parameter** | Value: Modus der Achsüberlagerung (siehe LStep-Dokumentation) |
| **Beispiel** | LS.SetXYAxisComp(1) ; |

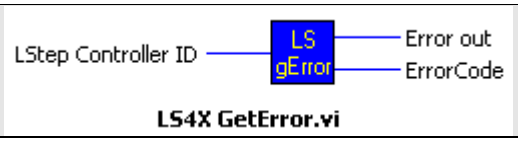
| LS_LStepSave | |
|---|---|
| **Beschreibung** | Aktuelle Konfiguration in LStep speichern (EEPROM) |
| **Delphi** | function LS_LStepSave(): Integer;<br>function LSX_LStepSave(LSID: Integer): Integer; |
| **C++** | int LStepSave (); |
| **LabView:** |  |
| **Parameter** | - |
| **Beispiel** | LS.LStepSave() ; |

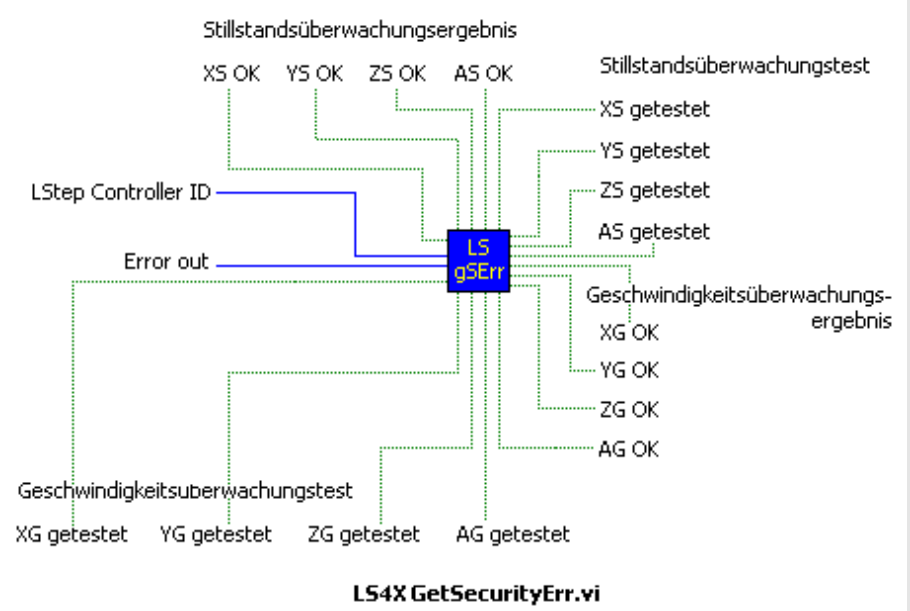
| LS_SetAccelSingleAxis | |
|---|---|
| **Beschreibung:** | Beschleunigung einstellen |
| **Delphi:** | function LS_SetAccelSingleAxis(Axis: Integer; Accel: Double): Integer;<br>function LSX_SetAccelSingleAxis(LSID: Integer; Axis: Integer; Accel: Double): Integer; |
| **C++:** | int SetAccelSingleAxis (int lAxis,double dAccel); |
| **LabView:** |  |
| **Parameter:** | Axis: (X, Y, Z, A numeriert von 1 bis 4)<br>Accel: Beschleunigung 0.01 – 10.00 [m/s²] |
| **Beispiel:** | LS.SetAccelSingleAxis(4, 1.0);<br>// Beschleunigung A-Achse 1.0 m/s² |

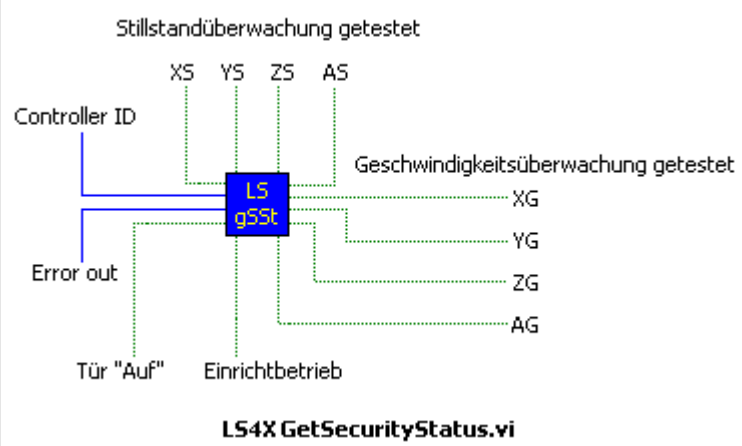
| LS_SetVelSingleAxis | |
|---|---|
| **Beschreibung:** | Geschwindigkeit für einzelne Achse einstellen |
| **Delphi:** | function LS_SetVelSingleAxis(Axis: Integer; Vel: Double): Integer;<br>function LSX_SetVelSingleAxis(LSID: Integer; Axis: Integer; Vel: Double): Integer; |
| **C++:** | int SetVelSingleAxis (int lAxis,double dVel); |
| **LabView:** |  |
| **Parameter:** | Axis: (X, Y, Z, A numeriert von 1 bis 4)<br>Vel: 0 – maximale Geschwindigkeit [U/s] |
| **Beispiel:** | LS.SetVelSingleAxis(1, 10.0)<br>// Geschwindigkeit X-Achse 10 U/s |

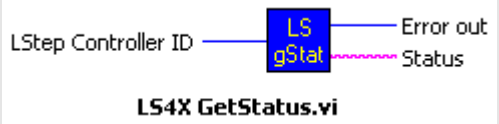
| LS_SoftwareReset | |
|---|---|
| **Beschreibung:** | Die Software wird in den Startzustand versetzt. |
| **Delphi:** | function LS_SoftwareReset: Integer;<br>function LSX_SoftwareReset(LSID: Integer): Integer; |
| **C++:** | int SoftwareReset (); |
| **LabView:** |  |
| **Parameter:** | - |
| **Beispiel:** | LS.SoftwareReset (); |

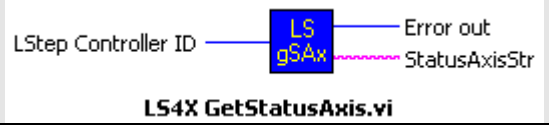
**Statusabfragen**

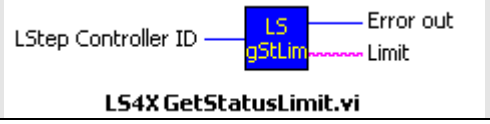
| **LS_GetError** | |
|---|---|
| **Beschreibung:** | liefert die aktuelle Fehlernummer |
| **Delphi:** | function LS_GetError(var ErrorCode: Integer): Integer;<br>function LSX_GetError(LSID: Integer; var ErrorCode: Integer): Integer; |
| **C++:** | int GetError (int *plErrorCode); |
| **LabView:** |  |
| **Parameter:** | ErrorCode: Fehlernummer |
| **Beispiel:** | LS.GetError(&ErrCode); |

| LS_GetSecurityErr | |
|---|---|
| **Beschreibung:** | Liest alle Zustände und Ergebnisse der GAL-Sicherheitsüberwachung (nur bei LS44-Steuerungen) |
| **Delphi:** | function LS_ Get SecurityErr (var Value: LongWord): Integer;<br>function LSX_ GetS SecurityErr (LSID: Integer; var Value: LongWord): Integer; |
| **C++:** | int GetSecurityErr (LongWord *pValue); |
| **LabView:** |  |
| **Parameter:** | Value: 32-Bit LongWord ohne Vorzeichen, welcher nach Aufruf der Funktion in den Bits 0-15 die Bit-Maske enthält.<br>Bit 0 X-Achse Stillstandüberwachungsergebnis (OK [1] / nicht OK [0])<br>Bit 1 Y-Achse Stillstandüberwachungsergebnis<br>Bit 2 Z-Achse Stillstandüberwachungsergebnis<br>Bit 3 A-Achse Stillstandüberwachungsergebnis<br>Bit 5 X-Achse Stillstandüberwachungstest (getestet [1] /nicht getestet [0])<br>Bit 6 Y-Achse Stillstandüberwachungstest<br>Bit 7 Z-Achse Stillstandüberwachungstest<br>Bit 8 A-Achse Stillstandüberwachungstest<br>Bit 9 X-Achse Geschwindigkeitsüberwachungsergebnis<br>Bit 10 Y-Achse Geschwindigkeitsüberwachungsergebnis<br>Bit 11 Z-Achse Geschwindigkeitsüberwachungsergebnis<br>Bit 12 A-Achse Geschwindigkeitsüberwachungsergebnis<br>Bit 13 X-Achse Geschwindigkeitsüberwachungstest<br>Bit 14 Y-Achse Geschwindigkeitsüberwachungstest<br>Bit 15 Z-Achse Geschwindigkeitsüberwachungstest |
| **Beispiel:** | LS.GetSecurityErr (&Value); |

| LS_GetSecurityStatus | |
|---|---|
| **Beschreibung:** | Liefert den aktuellen Zustand der Sicherheitsüberwachung (nur bei LS44-Steuerungen) |
| **Delphi:** | function LS_ Get SecurityStatus (var Value: LongWord): Integer;<br>function LSX_ GetS SecurityStatus (LSID: Integer; var Value: LongWord): Integer; |
| **C++:** | int GetSecurityStatus (LongWord *pValue); |
| **LabView:** |  |
| **Parameter:** | Value: 32-Bit LongWord ohne Vorzeichen, welcher nach Aufruf der Funktion in den Bits 0-15 die Bit-Maske enthält.<br>Bit 0-3 interne Merker<br>Bit 4 X-Achse Stillstandüberwachung getestet<br>Bit 5 Y-Achse Stillstandüberwachung getestet<br>Bit 6 Z-Achse Stillstandüberwachung getestet<br>Bit 7 A-Achse Stillstandüberwachung getestet<br>Bit 8 X-Achse Geschwindigkeitsüberwachung getestet<br>Bit 9 Y-Achse Geschwindigkeitsüberwachung getestet<br>Bit 10 Z-Achse Geschwindigkeitsüberwachung getestet<br>Bit 11 A-Achse Geschwindigkeitsüberwachung getestet<br>Bit 14 Zustand Einrichtbetrieb (Einrichtbetrieb = 1)<br>Bit 15 Zustand Tür (Tür „Auf“ = 1) |
| **Beispiel:** | LS.GetSecurityStatus (&Value); |

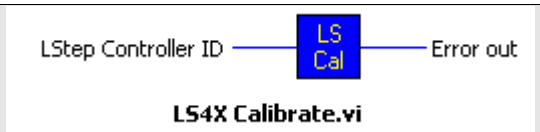
| LS_GetStatus | |
|---|---|
| **Beschreibung:** | liefert den aktuellen Zustand der Steuerung. |
| **Delphi:** | function LS_GetStatus(Stat: PChar; MaxLen: Integer): Integer;<br>function LSX_GetStatus(LSID: Integer; Stat: PChar; MaxLen: Integer): Integer; |
| **C++:** | int GetStatus (char *pcStat,int lMaxLen); |
| **LabView:** |  |
| **Parameter:** | Stat: Zeiger auf einen Puffer, in dem der Statusstring zurückgegeben wird<br>MaxLen: Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen |
| **Beispiel:** | LS.GetStatus(pcStat, 256); |

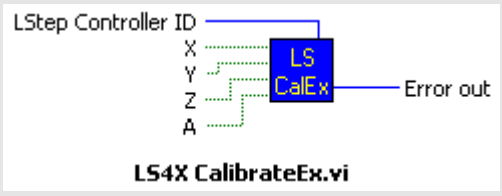
| LS_GetStatusAxis | |
|---|---|
| **Beschreibung:** | liefert den aktuellen Zustand der einzelnen Achsen |
| **Delphi:** | function LS_GetStatusAxis(StatusAxisStr: PChar; MaxLen: Integer): Integer;<br>function LSX_GetStatusAxis(LSID: Integer; StatusAxisStr: PChar; MaxLen: Integer): Integer; |
| **C++:** | int GetStatusAxis (char *pcStatusAxisStr,int lMaxLen); |
| **LabView:** |  |
| **Parameter:** | StatusAxisStr: Zeiger auf einen Puffer, in dem der Statusstring zurückgegeben wird<br>MaxLen: Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen<br><br>z.B.: @ - M – J – C – S – A – D – U - T<br>@ = Achse steht<br>M = Achse ist in Bewegung (Motion)<br>- = Achse ist nicht freigegeben<br>J = Joystick eingeschaltet<br>C = Achse ist in Regelung<br>A = Rückmeldung nach dem Kalibrieren<br>E = Fehler beim Kalibrieren (Endschalter nicht korrekt freigefahren)<br>D = Rückmeldung nach dem Tischhubmessen<br>U = Einrichtbetrieb<br>T = Timeout |
| **Beispiel:** | LS.GetStatusAxis(pcStatAxis, 256); |

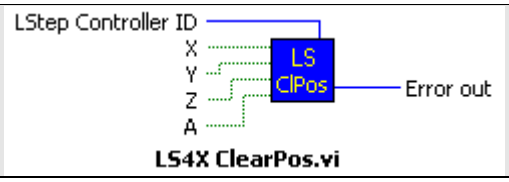
| LS_GetStatusLimit | |
|---|---|
| **Beschreibung:** | Liefert den aktuellen Zustand der Software-Grenzen jeder einzelnen Achse |
| **Delphi:** | function LS_ GetStatusLimit (Limit: PChar; MaxLen: Integer): Integer;<br>function LSX_ GetStatusLimit (LSID: Integer; Limit: PChar; MaxLen: Integer): Integer; |
| **C++:** | int GetStatusLimit (char *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<br>A = Achse wurde kalibriert<br>D = Tischhub wurde gemessen<br>L = Software-Limit wurde gesetzt<br>- = Software-Grenze wurde nicht verändert<br>MaxLen: Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen |
| **Beispiel:** | LS.GetStatusLimit (pc Limit, 64); |

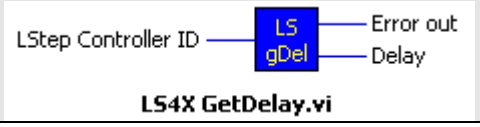
| LS_SetAutoStatus | |
|---|---|
| **Beschreibung:** | AutoStatus Ein/Aus |
| | Hinweis: der AutoStatus-Modus sollte normalerweise nicht verändert werden, da das LSTEP API bei Verfahrbefehlen etc. den richtigen Modus einstellt, eine Änderung auf 0 oder 2 könnte zu Fehlern führen |
| **Delphi:** | function LS_SetAutoStatus(Value: Integer): Integer;<br>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.<br>1 ➔ Es werden automatisch „Positionerreicht“- Meldungen von der Steuerung gesendet.<br>2 ➔ Es werden automatisch „Positionerreicht“- und Status - Meldungen von der Steuerung gesendet.<br>3 ➔ Es gibt bei „Positionerreicht“ nur ein Carriage Return zurück. |
| **Beispiel:** | LS.SetAutoStatus(3); |

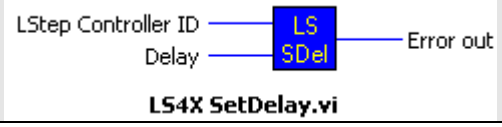
**Fahrbefehle und Positionsverwaltung**

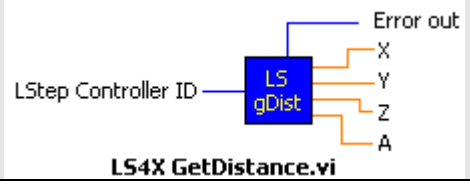
| **LS_Calibrate** | |
|---|---|
| **Beschreibung:** | Kalibrieren |
| | Bewegt alle freigegebenen Achsen in Richtung kleinerer Positionswerte. Die Verfahrbewegung wird unterbrochen sobald die Endschalter angefahren wurden. Der Positionswert wird auf 0 gesetzt. |
| **Delphi:** | function LS_Calibrate: Integer;<br>function LSX_Calibrate(LSID: Integer): Integer; |
| **C++:** | int Calibrate(); |
| **LabView:** |  |
| **Parameter:** | - |
| **Beispiel:** | LS.Calibrate(); |

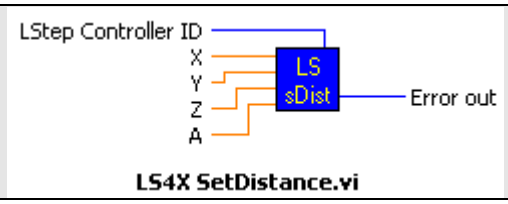
| **LS_CalibrateEx** | |
|---|---|
| **Beschreibung:** | Kalibrieren |
| | (Es werden nur die Achsen kalibriert, deren entsprechendes Bit in dem übergebenen Integer-Wert gesetzt ist.) |
| **Delphi:** | function LS_CalibrateEx(Flags: Integer): Integer;<br>function LSX_CalibrateEx(LSID: Integer; Flags: Integer): Integer; |
| **C++:** | int CalibrateEx (int lFlags); |
| **LabView:** |  |
| **Parameter:** | Flags: Bit-Maske,<br>Bit 2 = 1 → Z-Achse kalibrieren<br>Bit 2 = 0 → Z-Achse nicht kalibrieren<br>... |
| **Beispiel:** | LS.CalibrateEx(6);<br>// Nur Y- und Z-Achse kalibrieren |

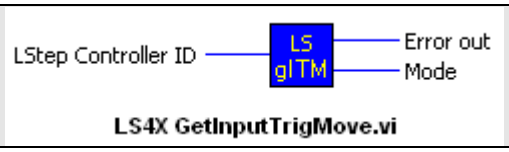
| LS_ClearPos | |
|---|---|
| **Beschreibung:** | Setzt die Position auf 0, auch den internen Zähler.<br>Diese Funktion wird für Endlosachsen gebraucht, da die Steuerung nur ±1000 Motorumdrehungen vom Werteberreich verarbeiten kann.<br>Bei erkannten Geber wird die Funktion für jeweilige Achse nicht ausgeführt. |
| **Delphi:** | function LS_ClearPos (lFlags: Integer): Integer;<br>function LSX_ClearPos (LSID: Integer; lFlags: Integer): Integer; |
| **C++:** | int ClearPos (int lFlags); |
| **LabView:** |  |
| **Parameter:** | lFlags: Bit-Maske<br>Bit 0 = 1 → Position der x-Achse wird genullt<br>Bit 1 = 0 → Für die y-Achse wird die Function nicht ausgeführt |
| **Beispiel:** | LS. ClearPos(5); //Positionen der x- und z- Achsen werden genullt. |

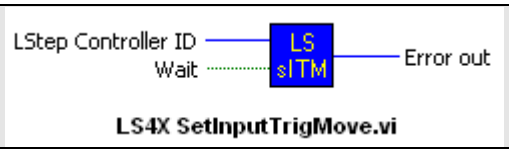
| LS_GetDelay | |
|---|---|
| **Beschreibung:** | Liest die Verzögerung des Vektorstarts. |
| **Delphi:** | function LS_GetDelay(var Delay: Integer): Integer;<br>function LSX_GetDelay(LSID: Integer; var Delay: Integer): Integer; |
| **C++:** | int GetDelay (int *plDelay); |
| **LabView:** |  |
| **Parameter:** | Delay: Verzögerung, in ms |
| **Beispiel:** | LS.GetDelay(&Delay); |

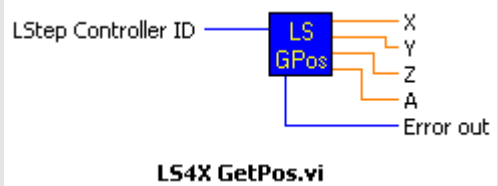
| LS_SetDelay | |
|---|---|
| **Beschreibung:** | Durch den Befehl Delay kann eine Verzögerung des Vektorstarts erzeugt werden. |
| **Delphi:** | function LS_SetDelay(Delay: Integer): Integer;<br>function LSX_SetDelay(LSID: Integer; Delay: Integer): Integer; |
| **C++:** | int SetDelay (int lDelay); |
| **LabView:** |  |
| **Parameter:** | 0 – 10000 (ms) |
| **Beispiel:** | LS.SetDelay(1000);<br>// 1s Verzögerung |

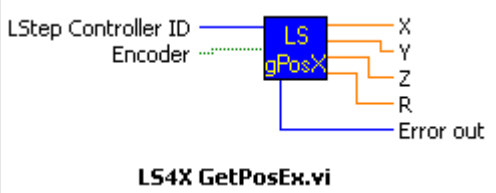
| LS_GetDistance | |
|---|---|
| **Beschreibung:** | Liefert die Strecke für LS_MoveRelShort |
| **Delphi:** | function LS_GetDistance(var X, Y, Z, A: Double): Integer;<br>function LSX_GetDistance(LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetDistance (double *pdX, double *pdY, double *pdZ, double *pdR); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: die aktuellen Strecken aller Achsen, abhängig von den Dimensionen |
| **Beispiel:** | LS.GetDistance(&X, &Y, &Z, &A); |

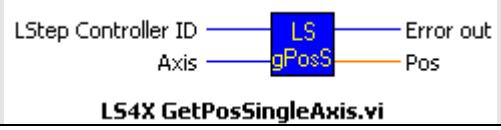
| LS_SetDistance | |
|---|---|
| **Beschreibung:** | Strecke setzen (für LS_MoveRelShort) |
| **Delphi:** | function LS_SetDistance(X, Y, Z, A: Double): Integer;<br>function LSX_SetDistance(LSID: Integer; X, Y, Z, A: Double): Integer; |
| **C++:** | int SetDistance (double dX,double dY,double dZ,double dA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A<br>Min-/max-Verfahrbereich<br>(Werte sind abhängig von der Dimension) |
| **Beispiel:** | LS.SetDistance(1, 2, 0, 0);<br>/* Strecken für die Achsen X und Y werden gesetzt, Z und A werden bei Aufruf der Funktion LS_MoveRelShort nicht bewegt. */ |

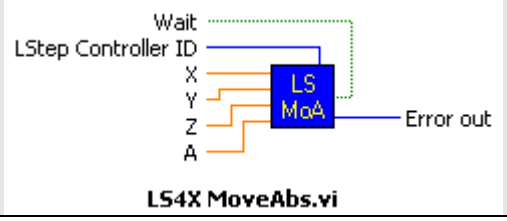
| LS_GetInputTrigMove | |
|---|---|
| **Beschreibung:** | Liefert die Konfiguration vom Pin1 auf dem MFP. |
| **Delphi:** | function LS_GetInputTrigMove (var Mode: Integer): Integer;<br>function LSX_GetInputTrigMove (LSID: Integer; var Mode: Integer): Integer; |
| **C++:** | int GetInputTrigMove (int *plMode); |
| **LabView:** |  |
| **Parameter:** | **lMode** – Modus.<br>lMode = 0 → Funktion nicht aktiv<br>lMode = 1 → absolut positionieren bei positiver Flanke<br>lMode = 2 → absolut positionieren bei negativer Flanke<br>lMode = 3 → relativ positionieren bei positiver Flanke<br>lMode = 4 → relativ positionieren bei negativer Flanke |
| **Beispiel:** | LS. GetInputTrigMove (&lMode); |

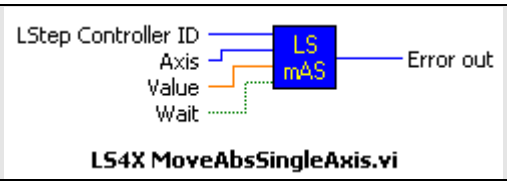
| LS_SetInputTrigMove | |
|---|---|
| **Beschreibung:** | Konfiguriert den Pin 1 auf dem MFP, so dass man mit einem externen Signal einen Move starten kann. |
| **Delphi:** | function LS_SetInputTrigMove (Mode: Integer; Wait: LongBool): Integer;<br>function LSX_SetInputTrigMove (LSID: Integer; Mode: Integer; Wait: LongBool): Integer; |
| **C++:** | int SetInputTrigMove (int lMode, BOOL bWait); |
| **LabView:** |  |
| **Parameter:** | **lMode** – Modus.<br>lMode = 0 → Funktion nicht aktiv<br>lMode = 1 → absolut positionieren bei positiver Flanke<br>lMode = 2 → absolut positionieren bei negativer Flanke<br>lMode = 3 → relativ positionieren bei positiver Flanke<br>lMode = 4 → relativ positionieren bei negativer Flanke<br><br>Verfahren wird der Wert, der in „distance“ steht.<br><br>**bWait** – das Warten auf einen Move.<br>bWait = 1 → es wird so lange gewartet, bis nach einem externen Signal ein Move ausgeführt wird, danach wird der Modus auf 0 gesetzt.<br>bWait = 0 → es wird nicht auf einen Move gewartet.<br>bWait wird nicht ausgewertet, wenn lMode = 0. |
| **Beispiel:** | LS. SetInputTrigMove (3, False); |

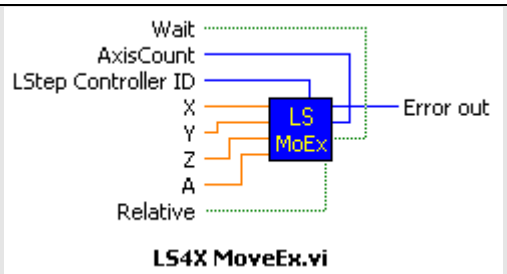
| LS_GetPos | |
|---|---|
| **Beschreibung:** | Abfrage der aktuellen Position aller Achsen |
| | Für nicht vorhandene Achsen wird der Wert 0.0 zurückgeliefert |
| **Delphi:** | function LS_GetPos(var X, Y, Z, A: Double): Integer;<br>function LSX_GetPos(LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetPos (double *pdX,double *pdY,double *pdZ,double *pdA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z, A: Positionswerte |
| **Beispiel:** | double X, Y, Z, A;<br>LS.GetPos(&X, &Y, &Z, &A); |

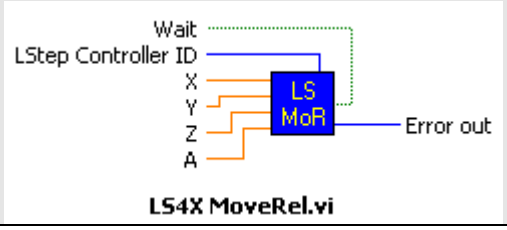
| LS_GetPosEx | |
|---|---|
| **Beschreibung:** | Abfrage der aktuellen Geber- bzw. Positionswerte aller Achsen |
| | Für nicht vorhandene Achsen wird der Wert 0.0 zurückgeliefert |
| **Delphi:** | function LS_GetPosEx(var X, Y, Z, A: Double; Encoder: LongBool): Integer;<br>function LSX_GetPosEx(LSID: 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<br>Encoder = true → Geberwerte liefern falls Geber angeschlossen<br>Encoder = false → Positionswerte liefern |
| **Beispiel:** | double X, Y, Z, A;<br>LS.GetPosEx(&X, &Y, &Z, &A, true); |

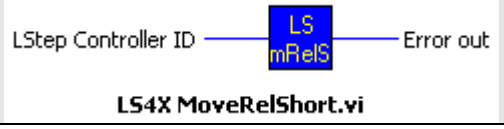
| LS_GetPosSingleAxis | |
|---|---|
| **Beschreibung:** | Abfrage der aktuellen Position einer Achse |
| | Für nicht vorhandene Achsen wird der Wert 0.0 zurückgeliefert |
| **Delphi:** | function LS_GetPosSingleAxis(Axis: Integer; var Pos: Double): Integer;<br>function LSX_GetPosSingleAxis(LSID: Integer; Axis: Integer; var Pos: Double): Integer; |
| **C++:** | int GetPosSingleAxis (int lAxis,double *pdPos); |
| **LabView:** |  |
| **Parameter:** | Axis: Achse, deren Positionswert abgefragt werden soll (X, Y, Z, A numeriert von 1 bis 4)<br>Pos: Positionswert |
| **Beispiel:** | LS.GetPosSingleAxis(2, &YPos);<br>// Position Y-Achse auslesen |

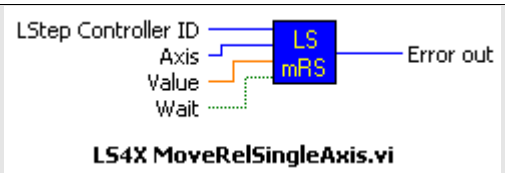
| LS_MoveAbs | |
|---|---|
| **Beschreibung:** | Absolutposition anfahren |
| | (Die Achsen x, y, z und a werden auf die übergebenen Positionswerte positioniert.) |
| **Delphi:** | function LS_MoveAbs(X, Y, Z, A: Double; Wait: LongBool): Integer;<br>function LSX_MoveAbs(LSID: Integer; X, Y, Z, A: Double; Wait: LongBool): Integer; |
| **C++:** | int MoveAbs (double dX, double dY, double dZ, double dA, BOOL Wait); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A<br>+- Verfahrbereich<br>Eingabe ist abhängig von der Dimension<br>Wait: Gibt an, ob die Funktion nachdem die Position erreicht wurde (= true) oder direkt zurückkehren soll (= false) |
| **Beispiel:** | LS.MoveAbs(10.0, 10.0, 10.0, 10.0, true); |

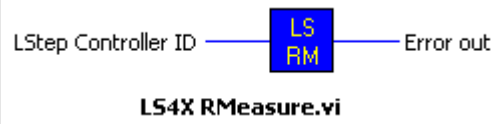
| LS_MoveAbsSingleAxis | |
|---|---|
| **Beschreibung:** | Einzelne Achse absolut positionieren |
| **Delphi:** | function LS_MoveAbsSingleAxis(Axis: Integer; Value: Double; Wait: LongBool): Integer;<br>function LSX_MoveAbsSingleAxis(LSID: Integer; Axis: Integer; Value: Double; Wait: LongBool): Integer; |
| **C++:** | int MoveAbsSingleAxis (int lAxis,double dValue,BOOL Wait); |
| **LabView:** |  |
| **Parameter:** | Axis: (X, Y, Z, A numeriert von 1 bis 4)<br>Value: Position (Eingabe ist abhängig von der eingestellten Dimension) |
| **Beispiel:** | LS.MoveAbsSingleAxis(2, 10.0);<br>// Y-Achse auf 10mm absolut positionieren |

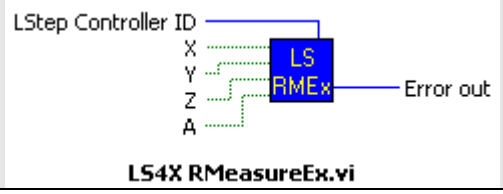
| LS_MoveEx | |
|---|---|
| **Beschreibung** | Erweiterter Verfahr-Befehl<br><br>Die Funktion LS_MoveEx kann relative und absolute Verfahrbefehle ausführen, synchron und asynchron. Die Anzahl der Achsen, die verfahren werden sollen, kann mit dem Parameter AxisCount bestimmt werden Diese Funktion kann beispielsweise genutzt werden, um bei einer LStep44 nur X und Y zu verfahren. |
| **Delphi** | function LS_MoveEx(X, Y, Z, R: Double; Relative, Wait: LongBool; AxisCount: Integer): Integer;<br>function LSX_MoveEx(LSID: Integer; X, Y, Z, R: Double; Relative, Wait: LongBool; AxisCount: Integer): Integer; |
| **C++** | int MoveEx (double dX, double dY, double dZ, double dR, BOOL bRelative, BOOL bWait, int lAxisCount); |
| **LabView:** |  |
| **Parameter** | X, Y, Z, R : Positions-Vektor<br><br>Relative : bei Relative=false werden die Werte X, Y, Z und R als absolute Koordinaten interpretiert, bei Relative=true als relative Koordinaten zur aktuellen Position<br><br>Wait: wird Wait=true gesetzt, kehrt die Funktion erst nach Erreichen der Zielposition zurück, ansonsten kehrt sie unmittelbar nach Senden des Befehls an die LStep zurück.<br><br>AxisCount: Anzahl der Achsen, die verfahren werden sollen<br>Ist AxisCount=1, wird nur X verfahren<br>Ist AxisCount=2, werden X und Y verfahren... |
| **Beispiel** | LS_MoveEx(2.0, 3.0, 0, 0, true, true, 2) ; // Es werden X und Y um 2 bzw 3 relativ verfahren |

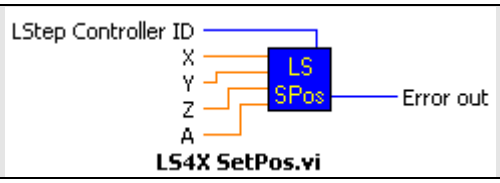
| LS_MoveRel | |
|---|---|
| **Beschreibung:** | Relativen Vektor fahren |
| | (Die Achsen x, y, z und a werden um die übergebenen Strecken verfahren.) |
| **Delphi:** | function LS_MoveRel(X, Y, Z, A: Double; Wait: LongBool): Integer;<br>function LSX_MoveRel(LSID: Integer; X, Y, Z, A: Double; Wait: LongBool): Integer; |
| **C++:** | int MoveRel (double dX, double dY, double dZ, double dA, BOOL Wait); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A<br>+- Verfahrbereich<br>Eingabe ist abhängig von der Dimension<br>Wait: Gibt an, ob die Funktion nachdem die Position erreicht wurde (= true) oder direkt zurückkehren soll (= false) |
| **Beispiel:** | LS.MoveRel(10.0, 10.0, 10.0, 10.0, true); |

| LS_MoveRelShort | |
|---|---|
| **Beschreibung:** | Positionieren Relativ (short command) |
| | Dieser Befehl sollte verwendet werden, damit aufeinander folgende relative Verfahrbefehle (mit derselben Strecke) schneller angefahren werden.<br>Die Strecke muß zuvor einmal mit LS_SetDistance gesetzt worden sein. |
| **Delphi:** | function LS_MoveRelShort: Integer;<br>function LSX_MoveRelShort(LSID: Integer): Integer; |
| **C++:** | int MoveRelShort (); |
| **LabView:** |  |
| **Parameter:** | - |
| **Beispiel:** | LS.SetDistance(1.0, 1.0, 0, 0);<br>for (i = 0; i < 10; i++) LS.MoveRelShort();<br>// 10mal X- und Y-Achse um 1 mm relativ positionieren |

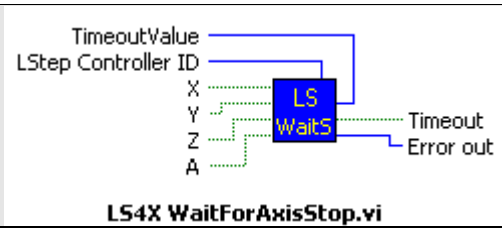
| LS_MoveRelSingleAxis | |
|---|---|
| **Beschreibung:** | Einzelne Achse relativ verfahren |
| **Delphi:** | function LS_MoveRelSingleAxis(Axis: Integer; Value: Double; Wait: LongBool): Integer;<br>function LSX_MoveRelSingleAxis(LSID: Integer; Axis: Integer; Value: Double; Wait: LongBool): Integer; |
| **C++:** | int MoveRelSingleAxis (int lAxis,double dValue,BOOL Wait); |
| **LabView:** |  |
| **Parameter:** | Axis: (X, Y, Z, A numeriert von 1 bis 4)<br>Value: Strecke (Eingabe ist abhängig von der eingestellten Dimension) |
| **Beispiel:** | LS.MoveRelSingleAxis(3, 5.0);<br>// Z-Achse um 5mm in positiver Richtung verfahren |

| LS_RMeasure | |
|---|---|
| **Beschreibung:** | Tischhub messen |
| | Bewegt alle freigegebenen Achsen in Richtung größerer Positionswerte. Die Verfahrbewegung wird unterbrochen sobald die Endschalter angefahren wurden. Der Positionswert wird gespeichert. |
| **Delphi:** | function LS_RMeasure: Integer;<br>function LSX_RMeasure(LSID: Integer): Integer; |
| **C++:** | int RMeasure(); |
| **LabView:** |  |
| **Parameter:** | - |
| **Beispiel:** | LS.RMeasure(); |

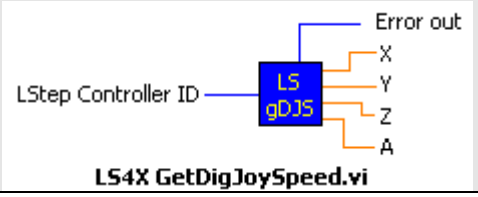
| LS_RMeasureEx | |
|---|---|
| **Beschreibung:** | Tischhub messen<br>(Tischhub messen wird nur bei den Achsen durchgeführt, deren entsprechendes Bit in dem übergebenen Integer-Wert gesetzt ist. |
| **Delphi:** | function LS_RMeasureEx(Flags: Integer): Integer;<br>function LSX_RMeasureEx(LSID: Integer; Flags: Integer): Integer; |
| **C++:** | int RMeasureEx (int lFlags); |
| **LabView:** |  |
| **Parameter:** | Flags: Bit-Maske,<br>Bit 2 = 1 → Z-Achse kalibrieren<br>Bit 2 = 0 → Z-Achse nicht kalibrieren<br>... |
| **Beispiel:** | LS.RMeasureEx(2);<br>// Tischhub messen (nur Y-Achse) |

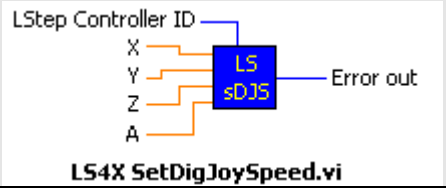
| LS_SetPos | |
|---|---|
| **Beschreibung:** | Positionswerte setzen |
| **Delphi:** | function LS_SetPos(X, Y, Z, R: Double): Integer;<br>function LSX_SetPos(LSID: Integer; X, Y, Z, A: Double): Integer; |
| **C++:** | int SetPos(double dX, double dY, double dZ, double dA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A<br>Min-/Max-Verfahrbereich<br>Eingabe ist abhängig von der Dimension |
| **Beispiel:** | LS.SetPos(10, 10, 0, 0); |

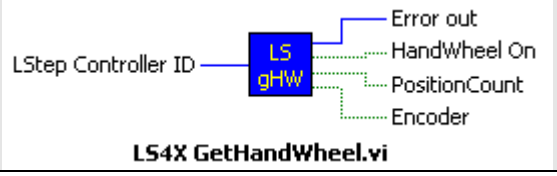
| LS_StopAxes | |
|---|---|
| **Beschreibung** | Abbruch<br>(Es werden alle Verfahrbewegungen abgebrochen) |
| **Delphi** | function LS_StopAxes: Integer;<br>function LSX_StopAxes(LSID: Integer): Integer; |
| **C++** | int StopAxes (); |
| **LabView** |  |
| **Parameter** | - |
| **Beispiel** | LS.StopAxes(); |

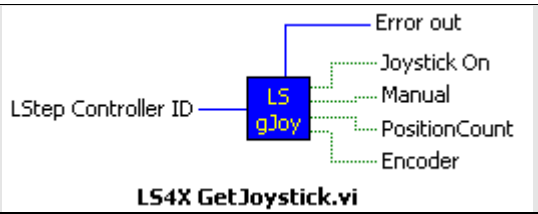
| LS_WaitForAxisStop | |
|---|---|
| **Beschreibung** | Die Funktion kehrt zurück, sobald die in der Bit-Maske AFlags gewählten Achsen ihre Zielposition erreicht haben<br><br>LS_WaitForAxisStop verwendet ‚?statusaxis', um den Status der Achsen zu pollen. |
| **Delphi** | function LS_WaitForAxisStop(AFlags: Integer; ATimeoutValue: Integer; var ATimeout: LongBool): Integer;<br>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<br>Bit 0: X-Achse<br>Bit 1: Y-Achse<br>Bit 2: Z-Achse<br>Bit 3: A-Achse<br><br>AtimeoutValue: Timeout in Millisekunden, WaitForAxisStop kehrt nach dieser Zeit mit Atimeout=true zurück, wenn die Achsen immer noch in Bewegung sind<br>AtimeoutValue = 0 setzt den Timeout auf 'Unendlich'<br>Das Atimeout Flag gibt an, ob ein Timeout aufgetreten ist. |
| **Beispiel** | LS.WaitForAxisStop(3, 0, flag);<br>// Warten bis X und Y-Achse gestoppt haben, kein Timeout<br>LS.WaitForAxisStop(7, 10000, flag);<br>// Warten bis X und Y-Achse gestoppt haben, 10 Sekunden timeout |

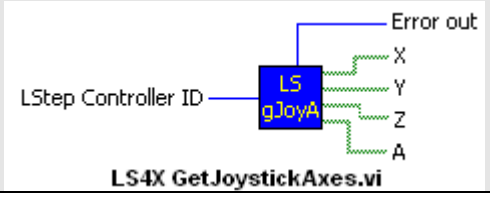
**Joystick und Handrad**

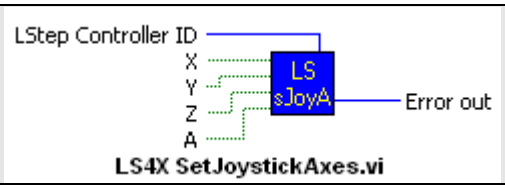
| LS_GetDigJoySpeed | |
|---|---|
| **Beschreibung:** | Auslesen der eingestellten Geschwindigkeiten |
| **Delphi:** | function LS_GetDigJoySpeed(var dX, dY, dZ, dR: Double): Integer;<br>function LSX_GetDigJoySpeed(LSID: Integer; var dX, dY, dZ, dR: Double): Integer; |
| **C++:** | int GetDigJoySpeed (double *pdX, double *pdY, double *pdZ, double *pdR); |
| **LabView:** |  |
| **Parameter:** | dX, dY, dZ, dR: Geschwindigkeitswerte [U/s] |
| **Beispiel:** | LS. GetDigJoySpeed(&X, &Y, &Z, &R); |

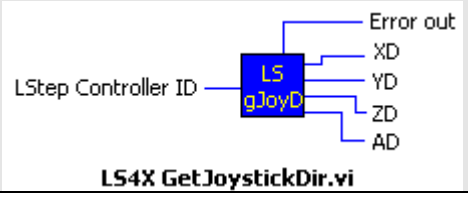
| LS_SetDigJoySpeed | |
|---|---|
| **Beschreibung:** | Mit diesem Befehl können einzelne Achsen mit einer konstanten Geschwindigkeit verfahren werden.<br>Will man nach dem Ausführen der Funktion wieder absolut oder relativ positionieren, muss man die Geschwindigkeit neu setzen. |
| **Delphi:** | function LS_SetDigJoySpeed(dX, dY, dZ, dR: Double): Integer;<br>function LSX_SetDigJoySpeed(LSID: Integer; dX, dY, dZ, dR: Double): Integer; |
| **C++:** | int SetDigJoySpeed (double dX, double dY, double dZ, double dR); |
| **LabView:** |  |
| **Parameter:** | dX, dY, dZ, dR: Geschwindigkeit [U/s],<br>Wertebereich: +- max. Geschwindigkeit |
| **Beispiel:** | LS. SetDigJoySpeed(0, 10.0, 25.0, 0); //Achsen X und R – 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“. |

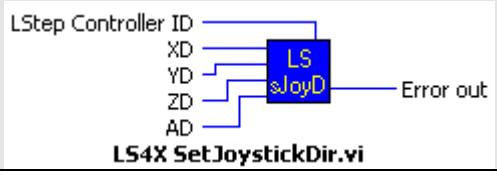
| LS_GetHandWheel | |
|---|---|
| **Beschreibung:** | Handradszustand ablesen |
| **Delphi:** | function LS_GetHandWheel(var PositionCount, Encoder: Boolean): Integer;<br>function LSX_GetHandWheel(LSID: Integer; var PositionCount, Encoder: LongBool): Integer; |
| **C++:** | int GetHandWheel (BOOL *pbHandWheelOn, BOOL *pbPositionCount, BOOL *pbEncoder); |
| **LabView:** |  |
| **Parameter:** | HWOn: True = Handrad ist eingeschaltet<br>PosCount: True = Positionszählung ist eingeschaltet<br>Encoder: True = Geberwerte, wenn vorhanden |
| **Beispiel:** | LS. GetHandWheel (&HWOn, &PosCount, &Encoder); |

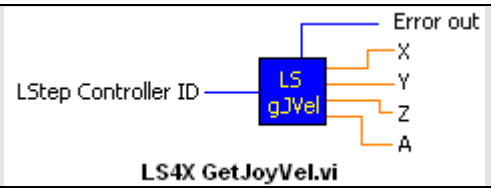
| LS_GetJoystick | |
|---|---|
| **Beschreibung:** | Abfrage des aktuellen Zustands vom Analog-Joystick. |
| **Delphi:** | function LS_GetJoystick (var JoystickOn, Manual, PositionCount, Encoder: LongBool): Integer;<br>function LSX_ GetJoystick (LSID: Integer; var JoystickOn, Manual, PositionCount, Encoder: LongBool): Integer; |
| **C++:** | int GetJoystick (BOOL *pbJoystickOn, BOOL *pbManual, BOOL *pbPositionCount, BOOL *pbEncoder); |
| **LabView:** |  |
| **Parameter:** | JoyOn: True = Joystick ist eingeschaltet<br>Manual: False = Joystickschalter steht auf Automatik<br>True = Joystick ist manual über Schalter eingeschaltet<br>PosCount: True = Positionszählung ist eingeschaltet<br>Enc: True = Geberwerte, wenn vorhanden |
| **Beispiel:** | LS. GetJoystick (&JoyOn, &Manual, &PosCount, &Enc); |

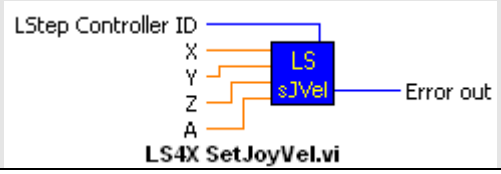
| LS_GetJoystickAxes | |
|---|---|
| **Beschreibung:** | Zeigt, für welche Achsen Joystick eingeschaltet ist |
| **Delphi:** | function LS_GetJoystickAxes(var Flags: Integer): Integer;<br>function LSX_GetJoystickAxes (LSID: Integer; var Flags: Integer): Integer; |
| **C++:** | int GetJoystickAxes (int *plFlags); |
| **LabView:** |  |
| **Parameter:** | Flags: 32-bit-Integer, welcher nach Aufruf der Funktion in den Bits 0-4 die Bit-Maske enthält.<br>Bit 0 = 1 → X-Achse Joystick „EIN“<br>Bit 2 = 0 → Z-Achse Joystick „AUS“ |
| **Beispiel:** | LS. GetJoystickAxes (&Flags); |

| LS_SetJoystickAxes | |
|---|---|
| **Beschreibung:** | Schaltet Joystick für angegebene Achsen ein |
| **Delphi:** | function LS_ SetJoystickAxes (Flags: Integer): Integer;<br>function LSX_ SetJoystickAxes (LSID: Integer; Flags: Integer): Integer; |
| **C++:** | int SetJoystickAxes (int Flags); |
| **LabView:** |  |
| **Parameter:** | Flags: Bit-Maske<br>Bit 0 = 1 → X-Achse Joystick einschalten<br>Bit 2 = 0 → Z-Achse Joystick ausschalten |
| **Beispiel:** | LS. SetJoystickAxes (3);<br>/* X- und Y-Achse – Joystick eingeschaltet (Bits 0 u. 1 gesetzt),<br>Z- und A-Achse – Joystick „AUS“ (Bit 2 = 0) */ |

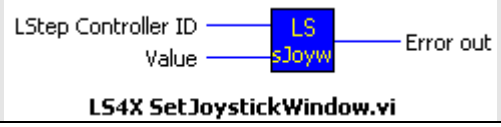
| LS_GetJoystickDir | |
|---|---|
| **Beschreibung:** | Liest Motordrehrichtung für Joystick |
| **Delphi:** | function LS_GetJoystickDir(var XD, YD, ZD, AD: Integer): Integer;<br>function LSX_GetJoystickDir(LSID: Integer; var XD, YD, ZD, AD: Integer): Integer; |
| **C++:** | int GetJoystickDir (int *plXD, int *plYD, int *plZD, int *plRD); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z, und A |
| | 0 ➔ Achse gesperrt<br>1 ➔ positive Drehrichtung<br>-1 ➔ negative Drehrichtung<br>2 ➔ mit Stromreduzierung<br>-2 ➔ mit Stromreduzierung |
| **Beispiel:** | LS.GetJoystickDir(&X, &Y, &Z, &A); |

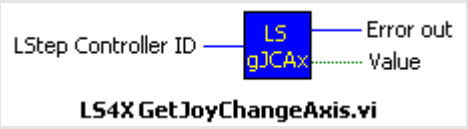
| LS_SetJoystickDir | |
|---|---|
| **Beschreibung:** | Richtung Joystick |
| **Delphi:** | function LS_SetJoystickDir(XD, YD, ZD, AD: Integer): Integer;<br>function LSX_SetJoystickDir(LSID: Integer; XD, YD, ZD, AD: Integer): Integer; |
| **C++:** | int SetJoystickDir (int lXD,int lYD,int lZD,int lAD); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z, und A |
| | 0 ➔ Achse gesperrt<br>1 ➔ positive Drehrichtung<br>-1 ➔ negative Drehrichtung<br>2 ➔ mit Stromreduzierung<br>-2 ➔ mit Stromreduzierung |
| **Beispiel:** | LS.SetJoystickDir(1, 1, -1, 0);<br>/* X- und Y-Achse positive Drehrichtung; Z-Achse negative Drehrichtung; A-Achse gesperrt */ |

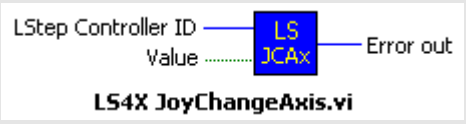
| LS_GetJoyVel | |
|---|---|
| **Beschreibung:** | Die maximalen Verfahrgeschwindigkeiten im Joystickbetriebt abfragen |
| **Delphi:** | function LS_GetJoyVel(var XD, YD, ZD, AD: Double): Integer;<br>function LSX_GetJoyVel(LSID: Integer; var XD, YD, ZD, AD: Double): Integer; |
| **C++:** | int GetJoyVel (double *pdXD, double *pdYD, double *pdZD, double *pdAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD, AD: Geschwindigkeitswerte [U/s] |
| **Beispiel:** | LS.GetJoyVel(&XD, &YD, &ZD, &AD); |

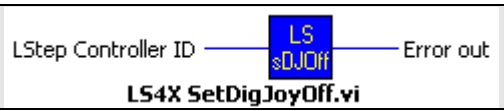
| LS_SetJoyVel | |
|---|---|
| **Beschreibung:** | Die maximalen Verfahrgeschwindigkeiten im Joystickbetrieb einstellen |
| **Delphi:** | function LS_SetJoyVel (XD, YD, ZD, AD: Double): Integer;<br>function LSX_SetJoyVel (LSID: Integer; XD, YD, ZD, AD: Double): Integer; |
| **C++:** | int SetJoyVel (double dXD, double dYD, double dZD, double dAD); |
| **LabView:** |  |
| **Parameter:** | XD, YD, ZD und AD: maximale Geschwindigkeit im Joystickbetrieb [U/s] |
| **Beispiel:** | LS.SetJoyVel (1.0, 15.0, 0, 0); |

| LS_GetJoystickWindow | |
|---|---|
| **Beschreibung** | Joystick-Fenster ablesen |
| **Delphi** | function LS_GetJoystickWindow(var AValue: Integer): Integer;<br>function LSX_GetJoystickWindow(LSID: Integer; var AValue: Integer): Integer; |
| **C++** | int GetJoystickWindow (int *plAValue); |
| **LabView:** |  |
| **Parameter** | AValue: der Analogbereich, in dem sich die Achsen nicht bewegen. |
| **Beispiel** | LS.GetJoystickWindow(&AValue) ; |

| LS_SetJoystickWindow | |
|---|---|
| **Beschreibung** | Joystick-Fenster setzen |
| **Delphi** | function LS_SetJoystickWindow(AValue: Integer): Integer;<br>function LSX_SetJoystickWindow(LSID: Integer; AValue: Integer): Integer; |
| **C++** | int SetJoystickWindow (int lAValue); |
| **LabView:** |  |
| **Parameter** | AValue: 0-100 |
| **Beispiel** | LS.SetJoystickWindow(20) ; |

| LS_GetJoyChangeAxis | |
|---|---|
| **Beschreibung:** | Liest Joystick-Achszuordnung |
| **Delphi:** | LS_GetJoyChangeAxis(var Value: LongBool): Integer;<br>LSX_GetJoyChangeAxis(LSID: Integer; var Value: LongBool): Integer; |
| **C++:** | int GetJoyChangeAxis (BOOL *pbValue); |
| **LabView:** |  |
| **Parameter:** | Value: true => Konventionelle Joystickauswertung<br>false => X- und Y-Achszuordnung ist getauscht |
| **Beispiel:** | LS. GetJoyChangeAxis (&Value); |

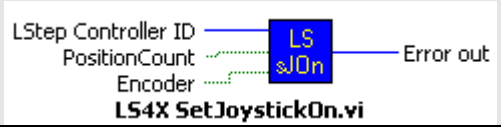
| LS_JoyChangeAxis | |
|---|---|
| **Beschreibung:** | setzt Joystick-Achszuordnung |
| **Delphi:** | LS_JoyChangeAxis(Value: LongBool): Integer;<br>LSX_JoyChangeAxis(LSID: Integer; Value: LongBool): Integer; |
| **C++:** | int JoyChangeAxis (BOOL bValue); |
| **LabView:** |  |
| **Parameter:** | Value: 0 - Ändert die Zuordnung der AD-Joystickkanäle (konventionelle Joystickauswertung)<br>1 - X- und Y-Achszuordnung wird getauscht |
| **Beispiel:** | LS. JoyChangeAxis (true); |

| LS_SetDigJoyOff | |
|---|---|
| **Beschreibung:** | Schaltet digitalen Joystick aus |
| **Delphi:** | function LS_ SetDigJoyOff: Integer;<br>function LSX_ SetDigJoyOff (LSID: Integer): Integer; |
| **C++:** | int SetDigJoyOff () ; |
| **LabView:** |  |
| **Parameter:** | |
| **Beispiel:** | LS. SetDigJoyOff () ; |

| LS_SetHandWheelOff | |
|---|---|
| **Beschreibung:** | Handrad Aus |
| **Delphi:** | function LS_SetHandWheelOff: Integer;<br>function LSX_SetHandWheelOff(LSID: Integer): Integer; |
| **C++:** | int SetHandWheelOff (); |
| **LabView:** |  |
| **Parameter:** | - |
| **Beispiel:** | LS.SetHandWheelOff(); |

| LS_SetHandWheelOn | |
|---|---|
| **Beschreibung:** | Handrad Ein |
| **Delphi:** | function LS_SetHandWheelOn(PositionCount, Encoder: Boolean): Integer;<br>function LSX_SetHandWheelOn(LSID: Integer; PositionCount, Encoder: LongBool): Integer; |
| **C++:** | int SetHandWheelOn (BOOL fPositionCount,BOOL fEncoder); |
| **LabView:** |  |
| **Parameter:** | PositionCount: Positionszählung Ein/Aus<br>Encoder: Geberwerte, wenn vorhanden |
| **Beispiel:** | LS. SetHandWheelOn (true, true);<br>// Handrad Ein mit Positionszählung (Geberwerte) |

| LS_SetJoystickOff | |
|---|---|
| **Beschreibung:** | Analog-Joystick Aus |
| **Delphi:** | function LS_SetJoystickOff: Integer;<br>function LSX_SetJoystickOff(LSID: Integer): Integer; |
| **C++:** | int SetJoystickOff(); |
| **LabView:** |  |
| **Parameter:** | - |
| **Beispiel:** | LS.SetJoystickOff(); |

| LS_SetJoystickOn | |
|---|---|
| **Beschreibung:** | Analog-Joystick Ein |
| **Delphi:** | function LS_SetJoystickOn(PositionCount, Encoder: LongBool): Integer;<br>function LSX_SetJoystickOn(LSID: Integer; PositionCount, Encoder: LongBool): Integer; |
| **C++:** | int SetJoystickOn (BOOL PositionCount,BOOL Encoder); |
| **LabView:** |  |
| **Parameter:** | PositionCount: Positionszählung Ein/Aus<br>Encoder: Geberwerte, wenn vorhanden |
| **Beispiel:** | LS.SetJoystickOn(true, true);<br>// Joystick Ein mit Positionszählung (Geberwerte) |

**Bedienpult mit Trackball und Joyspeed-Tasten**

| LS_GetBPZ | |
|---|---|
| **Beschreibung:** | Liest den Zustand des Zusatzbedienpults mit Trackball |
| **Delphi:** | function LS_GetBPZ(var AValue: Integer): Integer;<br>function LSX_GetBPZ(LSID: Integer; var AValue: Integer): Integer; |
| **C++:** | int GetBPZ (int *plAValue); |
| **LabView:** | - |
| **Parameter:** | AValue:<br>0 => Bedienpult ist „AUS“.<br>1 => Bedienpult aktiv, Trackball wird mit 0,1µ Schrittauflösung betrieben.<br>2 => Bedienpult aktiv, Trackball wird mit Faktor betrieben. |
| **Beispiel:** | LS.GetBPZ(&AValue); |

| LS_SetBPZ | |
|---|---|
| **Beschreibung** | Bedienpult Ein/Aus |
| **Delphi** | function LS_SetBPZ(AValue: Integer): Integer;<br>function LSX_SetBPZ(LSID: Integer; AValue: Integer): Integer; |
| **C++** | int SetBPZ (int lAValue); |
| **LabView:** | - |
| **Parameter** | AValue: 0-2<br>0 => Bedienpult „AUS“<br>1 => Bedienpult aktivieren und Trackball mit 0,1µ Schrittauflösung betreiben.<br>2 => Bedienpult aktivieren und Trackball mit Faktor betreiben. |
| **Beispiel** | LS.SetBPZ(1); |

| LS_GetBPZJoyspeed | |
|---|---|
| **Beschreibung:** | Bedienpult Joystick-Speed |
| **Delphi:** | function LS_GetBPZJoyspeed(APar: Integer; var AValue: Double): Integer;<br>function LSX_GetBPZJoyspeed(LSID: Integer; APar: Integer; var AValue: Double): Integer; |
| **C++:** | int GetBPZJoyspeed (int lAPar, double *pdAValue); |
| **LabView:** | - |
| **Parameter:** | APar: 1, 2 oder 3<br>AValue: maximale Geschwindigkeit [U/s] |
| **Beispiel:** | GetBPZJoyspeed(1, &AValue); // Auslesen der eingestellten Geschwindigkeit von Parameter 1. |

| LS_SetBPZJoyspeed | |
|---|---|
| **Beschreibung** | Bedienpult Joystick-Speed |
| **Delphi** | function LS_SetBPZJoyspeed(APar: Integer; AValue: Double): Integer;<br>function LSX_SetBPZJoyspeed(LSID: Integer; APar: Integer; AValue: Double): Integer; |
| **C++** | int SetBPZJoyspeed (int lAPar, double dAValue); |
| **LabView:** | - |
| **Parameter** | APar: 1, 2 oder 3<br>AValue: +- maximale Geschwindigkeit (vel) |
| **Beispiel** | SetBPZJoyspeed(1, 25) // Parameter 1 mit Geschwindigkeit 25 beschreiben |

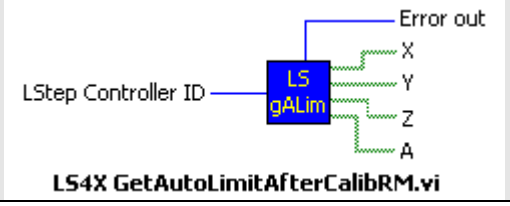
| LS_GetBPZTrackballBacklash | |
|---|---|
| **Beschreibung** | Bedienpult Trackball-Umkehrspiel auslesen |
| **Delphi** | function LS_GetBPZTrackballBackLash(var X, Y, Z, R: Double): Integer;<br>function LSX_GetBPZTrackballBackLash(LSID: Integer; var X, Y, Z, R: Double): Integer; |
| **C++** | int GetBPZTrackballBackLash (double *pdX, double *pdY, double *pdZ, double *pdR); |
| **LabView:** | - |
| **Parameter** | X, Y, Z, R: Umkehrspiel, mm. |
| **Beispiel** | LS.GetBPZTrackballBackLash(&X, &Y, &Z, &R); |

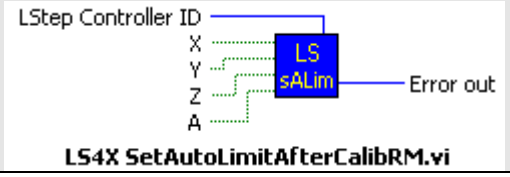
| LS_SetBPZTrackballBacklash | |
|---|---|
| **Beschreibung** | Bedienpult Trackball-Umkehrspiel |
| **Delphi** | function LS_SetBPZTrackballBackLash(X, Y, Z, R: Double): Integer;<br>function LSX_SetBPZTrackballBackLash(LSID: Integer; X, Y, Z, R: Double): Integer; |
| **C++** | int SetBPZTrackballBackLash (double dX, double dY, double dZ, double dR); |
| **LabView:** | - |
| **Parameter** | X, Y, Z, R: 0.001 bis 0.15 mm |
| **Beispiel** | LS.SetBPZTrackballBackLash(0.01, 0.01, 0.01, 0.01); |

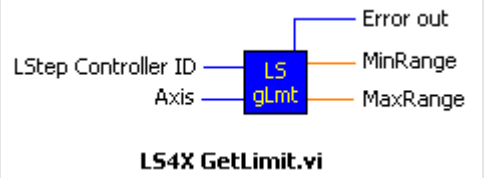
| LS_GetBPZTrackballFactor | |
|---|---|
| **Beschreibung** | Bedienpult Trackball-Faktor auslesen |
| **Delphi** | function LS_GetBPZTrackballFactor(AValue: Double): Integer;<br>function LSX_GetBPZTrackballFactor(LSID: Integer; AValue: Double): Integer; |
| **C++** | int GetBPZTrackballFactor (double *pdAValue); |
| **LabView:** | - |
| **Parameter** | AValue: Trackball – Factor. Z. B. AValue = 3 heißt: Ein Trackball-Impuls ergibt 3 Motor-Incremente. |
| **Beispiel** | LS.GetBPZTrackballFactor(&AValue) ; |

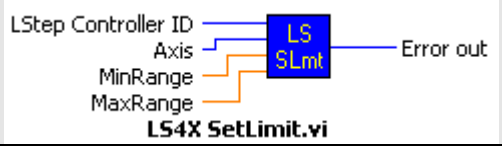
| LS_SetBPZTrackballFactor | |
|---|---|
| **Beschreibung** | Bedienpult Trackball-Faktor |
| **Delphi** | function LS_SetBPZTrackballFactor(AValue: Double): Integer;<br>function LSX_SetBPZTrackballFactor(LSID: Integer; AValue: Double): Integer; |
| **C++** | int SetBPZTrackballFactor (double dAValue); |
| **LabView:** | - |
| **Parameter** | AValue: 0.01 – 100<br>AValue=1 => Trackball – Factor = 1, d.h. Ein Trackball-Impuls ergibt ein Motor-Increment. |
| **Beispiel** | LS.SetBPZTrackballFactor(1.0) ; |

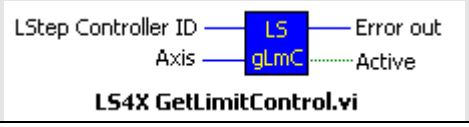
**Endschalter (Hardware u. Software)**

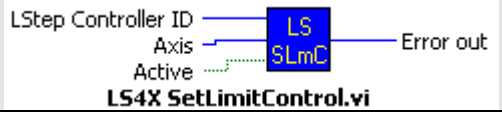
| **LS_GetAutoLimitAfterCalibRM** | |
|---|---|
| **Beschreibung:** | Gibt an, ob beim Kalibrieren und Tischhubmessendie interne Software-Limits gesetzt werden. |
| **Delphi:** | function LS_GetAutoLimitAfterCalibRM(var lFlags: Integer): Integer;<br>function LSX_ GetAutoLimitAfterCalibRM(LSID: Integer; var lFlags: Integer): Integer; |
| **C++:** | int GetAutoLimitAfterCalibRM (int *plFlags); |
| **LabView:** |  |
| **Parameter:** | lFlags: Bit-Maske<br>Bit 0 = 1 → Bei der x-Achse werden keine Verfahrbereichsgrenzen gesetzt<br>Bit 1 = 0 → Für die y-Achse werden Software-Limits gesetzt (calib/rm) |
| **Beispiel:** | LS. SetAutoLimitAfterCalibRM(&lFlags); |

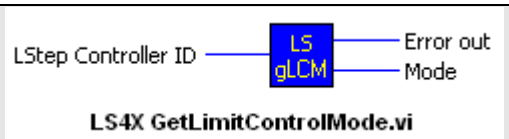
| **LS_SetAutoLimitAfterCalibRM** | |
|---|---|
| **Beschreibung:** | Verhindert, dass beim Kalibrieren und Tischhubmessendie interne Software-Limits gesetzt werden. |
| **Delphi:** | function LS_SetAutoLimitAfterCalibRM(lFlags: Integer): Integer;<br>function LSX_ SetAutoLimitAfterCalibRM(LSID: Integer; lFlags: Integer): Integer; |
| **C++:** | int SetAutoLimitAfterCalibRM (int lFlags); |
| **LabView:** |  |
| **Parameter:** | lFlags: Bit-Maske<br>Bit 0 = 1 → Bei der x-Achse werden keine Verfahrbereichsgrenzen gesetzt<br>Bit 1 = 0 → Für die y-Achse werden Software-Limits gesetzt (calib/rm) |
| **Beispiel:** | LS. SetAutoLimitAfterCalibRM(lFlags); |

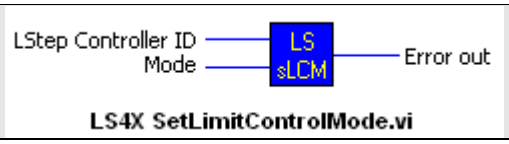
| LS_GetLimit | |
|---|---|
| **Beschreibung:** | Verfahrbereichsgrenzen lesen |
| **Delphi:** | function LS_GetLimit(Axis: Integer; var MinRange, MaxRange: Double): Integer;<br>function LSX_GetLimit(LSID: Integer; Axis: Integer; var MinRange, MaxRange: Double): Integer; |
| **C++:** | int GetLimit (int lAxis, double *pdMinRange, double *pdMaxRange); |
| **LabView:** |  |
| **Parameter:** | Axis: die Achse, welcher Verfahrbereichsgrenzen gelesen werden sollen (X, Y, Z, A numeriert von 1 bis 4)<br>MinRange: untere Verfahrbereichsgrenze, abhängig von der Dimension<br>MaxRange: obere Verfahrbereichsgrenze, abhängig von der Dimension |
| **Beispiel:** | LS.GetLimit(1, &MinRange, &MaxRange); |

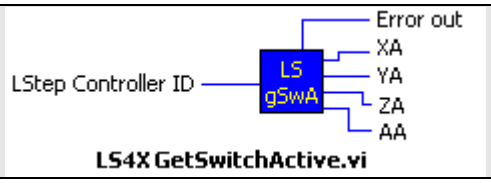
| LS_SetLimit | |
|---|---|
| **Beschreibung:** | Verfahrbereichsgrenzen einstellen |
| **Delphi:** | function LS_SetLimit(Axis: Integer; MinRange, MaxRange: Double): Integer;<br>function LSX_SetLimit(LSID: Integer; Axis: Integer; MinRange, MaxRange: Double): Integer; |
| **C++:** | int SetLimit (int lAxis,double dMinRange,double dMaxRange); |
| **LabView:** |  |
| **Parameter:** | Axis: die Achse, welcher Verfahrbereichsgrenzen zugewiesen werden sollen (X, Y, Z, A numeriert von 1 bis 4)<br>MinRange: untere Verfahrbereichsgrenze<br>MaxRange: obere Verfahrbereichsgrenze |
| **Beispiel:** | LS.SetLimit(1, -10.0, 20.0);<br>(Achse X –10 als untere und 20 als obere Verfahrbereichsgrenze zuweisen) |

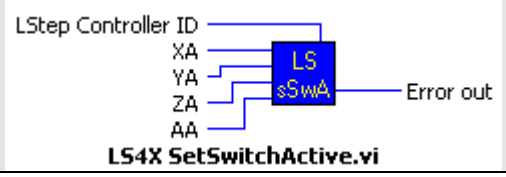
| LS_GetLimitControl | |
|---|---|
| **Beschreibung:** | Liest, ob die Bereichsüberwachung aktiv ist |
| **Delphi:** | function LS_GetLimitControl(Axis: Integer; var Active: LongBool): Integer;<br>function LSX_GetLimitControl(LSID: Integer; Axis: Integer; var Active: LongBool): Integer; |
| **C++:** | int GetLimitControl (int lAxis, BOOL *pbActive); |
| **LabView:** |  |
| **Parameter:** | Active: True = Bereichsüberwachung der jeweiligen Achse ist aktiv |
| **Beispiel:** | LS.GetLimitControl(2, &Active);<br>// Activ = False heißt: Bereichsüberwachung von Achse y ist deaktiviert. |

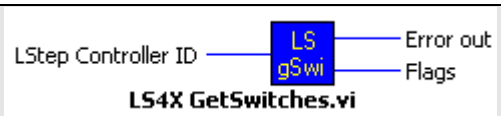
| LS_SetLimitControl | |
|---|---|
| **Beschreibung:** | Bereichsüberwachung |
| **Delphi:** | function LS_SetLimitControl(Axis: Integer; Active: LongBool): Integer;<br>function LSX_SetLimitControl(LSID: Integer; Axis: Integer; Active: LongBool): Integer; |
| **C++:** | int SetLimitControl (int lAxis,BOOL Active); |
| **LabView:** |  |
| **Parameter:** | Axis: (X, Y, Z, A numeriert von 1 bis 4)<br>Active: Bereichsüberwachung der jeweiligen Achse aktivieren |
| **Beispiel:** | LS.SetLimitControl(2, true);<br>// Bereichsüberwachung von Achse y aktiv |

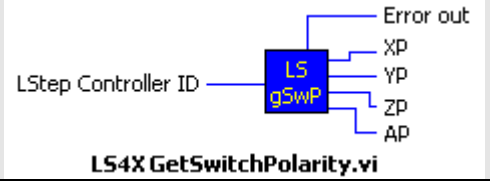
| LS_GetLimitControlMode | |
|---|---|
| **Beschreibung:** | Liefert den Modus für die Überwachung der Softwarelimits. |
| **Delphi:** | function LS_GetLimitControlMode (var Mode: Integer): Integer;<br>function LSX_GetLimitControlMode (LSID: Integer; var Mode: Integer): Integer; |
| **C++:** | int GetLimitControlMode (int *plMode); |
| **LabView:** |  |
| **Parameter:** | **lMode** – Modus.<br>lMode = 0→ Moves, die außerhalb des Verfahrbereiches liegen, werden nur bis zu den Grenzen des Verfahrbereiches ausgeführt.<br>lMode = 1 → Moves, die außerhalb des Verfahrbereiches liegen, werden nicht ausgeführt. |
| **Beispiel:** | LS. GetLimitControlMode (&lMode); |

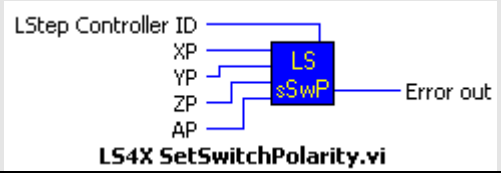
| LS_SetLimitControlMode | |
|---|---|
| **Beschreibung:** | Setzt den Modus für die Überwachung der Softwarelimits. |
| **Delphi:** | function LS_SetLimitControlMode (Mode: Integer): Integer;<br>function LSX_SetLimitControlMode (LSID: Integer; Mode: Integer): Integer; |
| **C++:** | int SetLimitControlMode (int lMode); |
| **LabView:** |  |
| **Parameter:** | **lMode** – Modus.<br>lMode = 0 → Moves, die außerhalb des Verfahrbereiches liegen, werden nur bis zu den Grenze des Verfahrbereiches ausgeführt.<br>lMode = 1 → Moves, die außerhalb des Verfahrbereiches liegen, werden nicht ausgeführt. |
| **Beispiel:** | LS. SetLimitControlMode(1); |

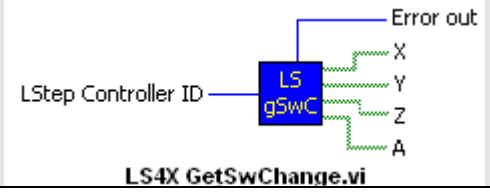
| LS_GetSwitchActive | |
|---|---|
| **Beschreibung:** | Liest Status für Endschalter Ein / Aus |
| **Delphi:** | function LS_GetSwitchActive(var XA, YA, ZA, AA: Integer): Integer;<br>function LSX_GetSwitchActive(LSID: Integer; var XA, YA, ZA, AA: Integer): Integer; |
| **C++:** | int GetSwitchActive (int *plXA, int *plYA, int *plZA, int *plRA); |
| **LabView:** |  |
| **Parameter:** | Für jede Achse wird eine Bitmaske geliefert: |
| | Bit 0 ➔ Null-Endschalter<br>Bit 1 ➔ Referenz-Endschalter<br>Bit 2 ➔ End-Endschalter |
| | Ist das Bit gesetzt - ist der jeweilige Schalter aktiv. |
| **Beispiel:** | LS.GetSwitchActive(&XA, &YA, &ZA, &RA); |

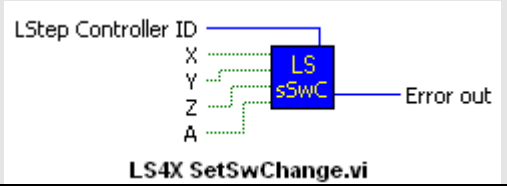
| LS_SetSwitchActive | |
|---|---|
| **Beschreibung:** | Endschalter Ein/Aus |
| **Delphi:** | function LS_SetSwitchActive(XA, YA, ZA, AA: Integer): Integer;<br>function LSX_SetSwitchActive(LSID: Integer; XA, YA, ZA, AA: Integer): Integer; |
| **C++:** | int SetSwitchActive (int lXA,int lYA,int lZA,int lAA); |
| **LabView:** |  |
| **Parameter:** | Für jede Achse wird eine Bitmaske übergeben: |
| | Bit 0 ➔ Null-Endschalter<br>Bit 1 ➔ Referenz-Endschalter<br>Bit 2 ➔ End-Endschalter |
| | Um den jeweiligen Schalter zu aktivieren, muß das Bit gesetzt werden. |
| **Beispiel:** | LS.SetSwitchActive(7, 1, 5, 0);<br>(Alle Endschalter der X-Achse Ein; Null-Endschalter der Y-Achse Ein; E0 und EE der Z-Achse ein; A-Achse: Alle Endschalter Aus) |

| LS_GetSwitches | |
|---|---|
| **Beschreibung:** | liest den Zustand aller Endschalter |
| **Delphi:** | function LS_GetSwitches(var Flags: Integer): Integer; |
| **C++:** | int GetSwitches (int *plFlags); |
| **LabView:** |  |
| **Parameter:** | Value: Zeiger auf Integerwert, der den Zustand aller Endschalter als Bitmaske enthält.<br>In der Bitmaske ist der Zustand der Endschalter wie folgt verschlüsselt:<br>Endschalter EE Ref. E0<br>Achse AZYX AZYX AZYX<br>Bit 0000 0000 0000<br>z.B.<br>Flags = 0x003 → E0 von X- und Y-Achse sind angefahren<br>Flags = 0x200 → EE von Y-Achse ist angefahren |
| **Beispiel:** | LS.GetSwitches(&Flags); |

| LS_GetSwitchPolarity | |
|---|---|
| **Beschreibung:** | Liest Endschalterpolarität |
| **Delphi:** | function LS_GetSwitchPolarity(var XP, YP, ZP, AP: Integer): Integer;<br>function LSX_GetSwitchPolarity(LSID: Integer; var XP, YP, ZP, AP: Integer): Integer; |
| **C++:** | int GetSwitchPolarity (int *plXP, int *plYP, int *plZP, int *plRP); |
| **LabView:** |  |
| **Parameter:** | Für jede Achse wird eine Bitmaske geliefert: |
| | Bit 0 → Null-Endschalter<br>Bit 1 → Referenz-Endschalter<br>Bit 2 → End-Endschalter |
| | Ist das Bit gesetzt werden - reagiert der jeweilige Schalter auf die positive Flanke |
| **Beispiel:** | LS.GetSwitchPolarity(&XP, &YP, &ZP, &RP); |

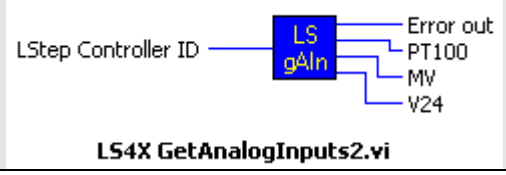
| LS_SetSwitchPolarity | |
|---|---|
| **Beschreibung:** | Endschalterpolarität einstellen |
| **Delphi:** | function LS_SetSwitchPolarity(XP, YP, ZP, AP: Integer): Integer;<br>function LSX_SetSwitchPolarity(LSID: Integer; XP, YP, ZP, AP: Integer): Integer; |
| **C++:** | int SetSwitchPolarity (int lXP,int lYP,int lZP,int lRA); |
| **LabView:** |  |
| **Parameter:** | Für jede Achse wird eine Bitmaske übergeben: |
| | Bit 0 ➔ Null-Endschalter<br>Bit 1 ➔ Referenz-Endschalter<br>Bit 2 ➔ End-Endschalter |
| | Reagiert der jeweilige Schalter auf die positive Flanke, muß das Bit gesetzt werden. |
| **Beispiel:** | LS.SetSwitchPolarity(7, 0, 0, 0);<br>(Alle Endschalter der X-Achse High-Aktiv, alle Endschalter der Y-Achse Low-Aktiv...) |

| LS_GetSwChange | |
|---|---|
| **Beschreibung:** | Zeigt die Einstellungen der Endschalter |
| **Delphi:** | function LS_GetSwChange(var Flags: Integer): Integer;<br>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 die Bit-Maske enthält.<br>Bit 0 = 1 ➔ Wechsel der Endschalter bei der X-Achse<br>Bit 2 = 0 ➔ Kein Endschalterwechsel bei der Z-Achse |
| **Beispiel:** | LS. GetSwChange (&Flags); |

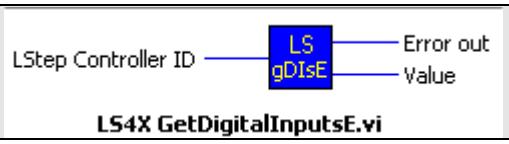
| LS_SetSwChange | |
|---|---|
| **Beschreibung:** | Endschalter tauschen |
| **Delphi:** | function LS_ SetSwChange (Flags: Integer): Integer;<br>function LSX_ SetSwChange (LSID: Integer; Flags: Integer): Integer; |
| **C++:** | int SetSwChange (int Flags); |
| **LabView:** |  |
| **Parameter:** | Flags: Bit-Maske<br>Bit 0 = 1 → X-Achse Endschalter wechseln<br>Bit 2 = 0 → Z-Achse kein Endschalterwechsel |
| **Beispiel:** | LS. SetSwChange (3);<br>/* X- und Y-Achse – Endschalter wechseln (Bits 0 u. 1 gesetzt)<br>Z- und A-Achse – kein Endschalterwechsel (Bit 2 = 0) */ |

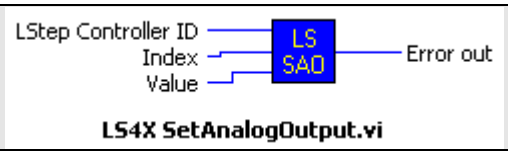
**Digitale und analoge Ein.- und Ausgänge**

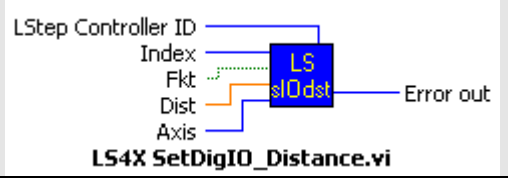
| LS_GetAnalogInput | |
|---|---|
| **Beschreibung:** | Lesen des aktuellen Zustands eines Analogkanals |
| **Delphi:** | function LS_GetAnalogInput(Index: Integer; var Value: Integer): Integer;<br>function LSX_GetAnalogInput(LSID: Integer; Index: Integer; var Value: Integer): Integer; |
| **C++:** | int GetAnalogInput (int lIndex,int *plValue); |
| **LabView:** |  |
| **Parameter:** | Index: 0-9 (Analogkanäle)<br>Value: Zeiger auf Integerwert, der den aktuellen Zustand des Analogkanals angibt |
| **Beispiel:** | LS.GetAnalogInput(0, &Eingang0); |

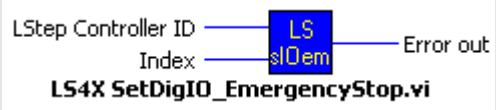
| LS_GetAnalogInputs2 | |
|---|---|
| **Beschreibung** | Lesen der aktuellen Zustände der Analogkanäle (Channel 6, 7, 8)<br>nur bei der LSTEP-PCI |
| **Delphi** | function LS_GetAnalogInputs2(var PT100, MV, V24: Integer): Integer;<br>function LSX_GetAnalogInputs2(LSID: Integer; var PT100, MV, V24: Integer): 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** | LS.GetAnalogInputs2(&PT100, &MV, &V24); |

| LS_GetDigitalInputs | |
|---|---|
| **Beschreibung:** | Alle Inputpins lesen |
| **Delphi:** | function LS_GetDigitalInputs(var Value: Integer): Integer;<br>function LSX_GetDigitalInputs(LSID: Integer; var Value: Integer): Integer; |
| **C++:** | int GetDigitalInputs (int *plValue); |
| **LabView:** |  |
| **Parameter:** | Value: Zeiger auf Integerwert, der den Zustand aller Eingänge als Bitmaske enthält. |
| **Beispiel:** | int Eingange;<br>LS.GetDigitalInputs(&Eingange);<br>if (Eingange & 16) ... // Wenn Inputpin 4 gesetzt |

| LS_GetDigitalInputsE | |
|---|---|
| **Beschreibung** | Zusätzliche digitale Eingänge lesen (16-31) |
| **Delphi** | function LS_GetDigitalInputsE(var Value: Integer): Integer;<br>function LSX_GetDigitalInputsE(LSID: Integer; var Value: Integer): Integer; |
| **C++** | int GetDigitalInputsE (int *plValue); |
| **LabView:** |  |
| **Parameter** | Value: Zeiger auf einen 32-bit-Integer, welcher nach Aufruf der Funktion in den Bits 0-15 den Status der Eingänge 16-31 enthält. |
| **Beispiel** | LS.GetDigitalInputsE(i); |

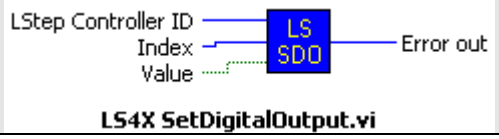
| LS_SetAnalogOutput | |
|---|---|
| **Beschreibung:** | Analogkanal setzen |
| **Delphi:** | function LS_SetAnalogOutput(Index: Integer; Value: Integer): Integer;<br>function LSX_SetAnalogOutput(LSID: Integer; Index: Integer; Value: Integer): Integer; |
| **C++:** | int SetAnalogOutput (int lIndex,int lValue); |
| **LabView:** |  |
| **Parameter:** | Index: 0-1 (Analogkanäle)<br>Value: 0-100 [%] |
| **Beispiel:** | LS.SetAnalogOutput(0, 100);<br>// Ausgang 0 auf Maximum setzen |

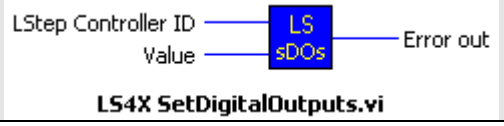
| LS_SetDigIO_Distance | |
|---|---|
| **Beschreibung:** | Funktion der digitalen Ein-/Ausgänge |
| | Aktivierung eines Ausgang in Abhängigkeit der eingestellten Strecke vor/nach der Zielposition. |
| **Delphi:** | function LS_SetDigIO_Distance(Index: Integer; Fkt: LongBool; Dist: Double; Axis: Integer): Integer;<br>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: 0 bis 15 (Outputpin) |
| | Fkt = false ➔ Aktivierung eines Ausgang in Abhängigkeit der eingestellten Strecke vor der Zielposition.<br>Fkt = true ➔ Aktivierung eines Ausgang in Abhängigkeit der eingestellten Strecke nach der Startposition. |
| | Dist: Strecke (Eingabe ist abhängig von der eingestellten Dimension)<br>Axis: (X, Y, Z, A numeriert von 1 bis 4) |
| **Beispiel:** | LS.SetDigIO_Distance(7, false, 78.9, 3);<br>/* Ausgang 7 wird 78.9mm vor erreichen der Zielposition (Z- Achse) aktiviert. */ |

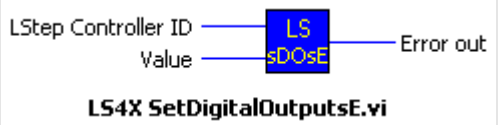
| LS_SetDigIO_EmergencyStop | |
|---|---|
| **Beschreibung:** | Funktion der digitalen Ein-/Ausgänge<br>Zuordnung des Not-Stop-Pins |
| **Delphi:** | function LS_SetDigIO_EmergencyStop(Index: Integer): Integer;<br>function LSX_SetDigIO_EmergencyStop(LSID: Integer; Index: Integer): Integer; |
| **C++:** | int SetDigIO_EmergencyStop (int lIndex); |
| **LabView:** |  |
| **Parameter:** | Index: 0 bis 15 (Input/Output) |
| **Beispiel:** | LS.SetDigIOEmergencyStop(15);<br>// Not-Stop-Pin 15 |

| LS_SetDigIO_Off | |
|---|---|
| **Beschreibung:** | Funktion der digitalen Ein-/Ausgänge Aus<br>(Keine Beeinflussung der Ein-/Ausgänge) |
| **Delphi:** | function LS_SetDigIO_Off(Index: Integer): Integer;<br>function LSX_SetDigIO_Off(LSID: Integer; Index: Integer): Integer; |
| **C++:** | int SetDigIO_Off (int lIndex); |
| **LabView:** |  |
| **Parameter:** | Index: 0 bis 15 (Input/Output), 16 (alle 16 Portpins) |
| **Beispiel:** | LS.SetDigIO_Off(0);<br>// dig. Fkt. Input-/Outputpin 0 Aus |

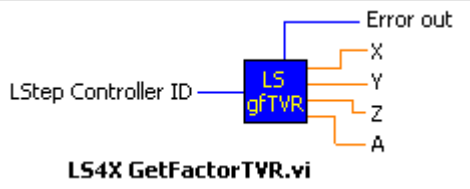
| LS_SetDigIO_Polarity | |
|---|---|
| **Beschreibung:** | Funktion der digitalen Ein-/Ausgänge<br>Einstellung der Polarität |
| **Delphi:** | function LS_SetDigIO_Polarity(Index: Integer; High: LongBool): Integer;<br>function LSX_SetDigIO_Polarity(LSID: Integer; Index: Integer; High: LongBool): Integer; |
| **C++:** | int SetDigIO_Polarity (int lIndex,BOOL High); |
| **LabView:** |  |
| **Parameter:** | Index: 0 bis 15 (Input/Output), 16 (alle 16 Portpins) |
| | High = true ➔ High-Aktiv<br>High = false ➔ Low-Aktiv |
| **Beispiel:** | LS.SetDigIO_Polarity(3, True);<br>// Input-/Outputpin 3 High-Aktiv |

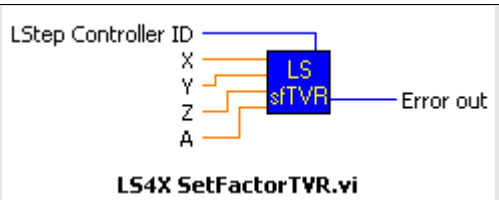
| LS_SetDigitalOutput | |
|---|---|
| **Beschreibung:** | Outputpin setzen |
| **Delphi:** | function LS_SetDigitalOutput(Index: Integer; Value: LongBool): Integer;<br>function LSX_SetDigitalOutput(LSID: Integer; Index: Integer; Value: LongBool): Integer; |
| **C++:** | int SetDigitalOutput (int lIndex,BOOL Value); |
| **LabView:** |  |
| **Parameter:** | Index: 0-15<br>Value: Zustand auf „0“ oder „1“ setzen |
| **Beispiel:** | LS.SetDigitalOutput(0, true);<br>// Outputpin 0 auf „1“ setzen |

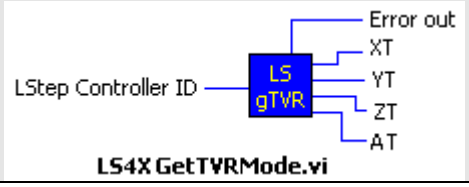
| LS_SetDigitalOutputs | |
|---|---|
| **Beschreibung** | Digitale Ausgänge setzen (0-15) |
| **Delphi** | function LS_SetDigitalOutputs(Value: Integer): Integer;<br>function LSX_SetDigitalOutputs(LSID: Integer; Value: Integer): Integer; |
| **C++** | int SetDigitalOutputs (int lValue); |
| **LabView:** |  |
| **Parameter** | Value: Bitmaske, der Wert auf den die Ausgänge 0-15 gesetzt werden, wird durch die Bits 0-15 bestimmt. |
| **Beispiel** | LS.SetDigitalOutputs($03); // Ausgänge 0 und 1 auf 1 setzen, die übrigen auf 0 |

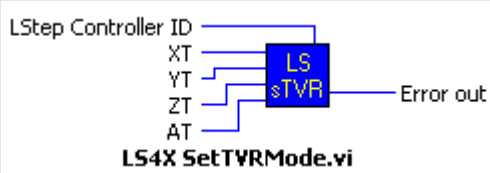
| LS_SetDigitalOutputsE | |
|---|---|
| **Beschreibung** | Zusätzliche digitale Ausgänge setzen (16-31) |
| **Delphi** | function LS_SetDigitalOutputsE(Value: Integer): Integer;<br>function LSX_SetDigitalOutputsE(LSID: Integer; Value: Integer): Integer; |
| **C++** | int SetDigitalOutputsE (int lValue); |
| **LabView:** |  |
| **Parameter** | Value: Bitmaske, der Wert auf den die Ausgänge 16-31 gesetzt werden, wird durch die Bits 0-15 bestimmt. |
| **Beispiel** | LS.SetDigitalOutputsE($03); // Ausgänge 16 und 17 auf 1 setzen, die übrigen auf 0 |

### Takt-Vor/Rück Eingänge

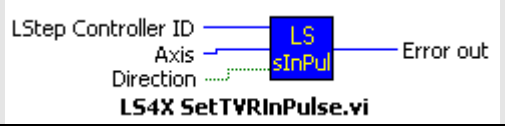
| LS_GetFactorTVR | |
|---|---|
| **Beschreibung:** | Liest den Faktor Takt Vor / Rück |
| **Delphi:** | function LS_GetFactorTVR(var X, Y, Z, A: Double): Integer;<br>function LSX_GetFactorTVR(LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetFactorTVR (double *pdX, double *pdY, double *pdZ, double *pdR); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: Factor Takt Vor/Rück.<br>Z. B. X = 10 heißt: Ein Takt = zehn Motorinkremente |
| **Beispiel:** | LS.GetFactorTVR(&X, &Y, &Z, &A); |

| LS_SetFactorTVR | |
|---|---|
| **Beschreibung:** | Faktor Takt Vor / Rück |
| **Delphi:** | function LS_SetFactorTVR(X, Y, Z, A: Double): Integer;<br>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 und A<br>0.01 – 100.00 |
| **Beispiel:** | LS.SetFactorTVR(2.0, 2.0, 0, 0);<br>/* Bei Achse X und Y soll Takt Vor/Rück mit dem Faktor 2 arbeiten */ |

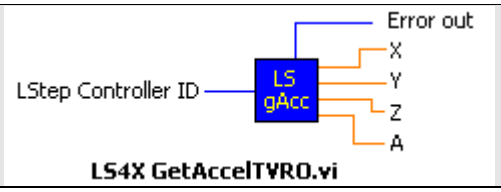
| LS_GetTVRMode | |
|---|---|
| **Beschreibung:** | Einstellung vom Takt Vor / Rück lesen |
| **Delphi:** | function LS_ GetTVRMode(var XT, YT, ZT, AT: Integer): Integer;<br>function LSX_ GetTVRMode(LSID: Integer; var XT, YT, ZT, AT: Integer): Integer; |
| **C++:** | int GetTVRMode (int *plXT, int *plYT, int *plZT, int *plRT); |
| **LabView:** |  |
| **Parameter:** | TVR-Modus von X, Y, Z und A:<br>0 ➔ Takt Vor/Rück ist „AUS“<br>1 ➔ Normale Takt Vor/Rück Bearbeitung<br>2 ➔ Takt Vor/Rück Bearbeitung arbeitet mit einem Faktor<br>3 ➔ Takt Vor/Rück Bearbeitung benötigt externe Freigabe über Triggerout Pin (MFP).<br>4 ➔ Kombination aus 2 & 3. |
| **Beispiel:** | LS. GetTVRMode(&XT, &YT, &ZT, &RT); |

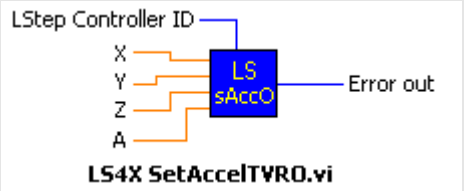
| LS_SetTVRMode | |
|---|---|
| **Beschreibung:** | Takt Vor / Rück einstellen |
| **Delphi:** | function LS_SetTVRMode(XT, YT, ZT, AT: Integer): Integer;<br>function LSX_SetTVRMode(LSID: Integer; XT, YT, ZT, AT: Integer): Integer; |
| **C++:** | int SetTVRMode (int lXT,int lYT,int lZT,int lAT); |
| **LabView:** |  |
| **Parameter:** | TVR-Modus von X, Y, Z und A:<br>0 ➔ Takt Vor/Rück „AUS“<br>1 ➔ Normale Takt Vor/Rück Bearbeitung<br>2 ➔ Takt Vor/Rück Bearbeitung arbeitet mit einem Faktor<br>3 ➔ Takt Vor/Rück Bearbeitung benötigt externe Freigabe über Triggerout Pin (MFP).<br>4 ➔ Kombination aus 2 & 3. |
| **Beispiel:** | LS.SetTVRMode(1, 1, 0, 0);<br>// TVR X- und Y-Achse Ein |

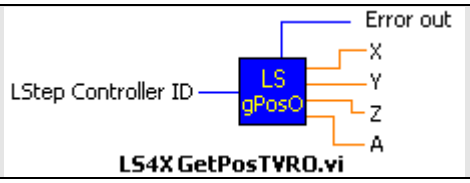
**Takt-Vor/Rück über Schnittstelle**

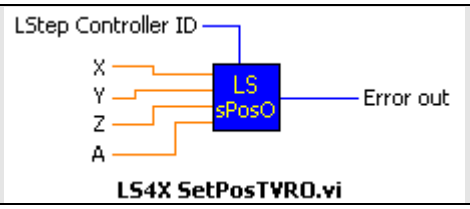
| LS_SetTVRInPulse | |
|---|---|
| **Beschreibung:** | Diese Funktion hat die gleiche Auswirkung wie ein externer Takt mit Richtungsinformation. |
| **Delphi:** | function LS_SetTVRInPulse (Axis: Integer; Direction: Boolean): Integer; function LSX_SetTVRInPulse (LSID: Integer; Axis: Integer; Direction: Boolean): Integer; |
| **C++:** | int SetTVRInPulse (int Axis, BOOL Direction); |
| **LabView:** |  |
| **Parameter:** | Wert: Anzahl der ausgeführten Trigger |
| **Beispiel:** | LS.SetTVRInPulse (2, true); // 1 Takt vorwärts bei der y-Achse. |

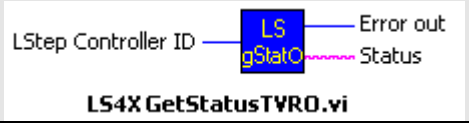
**Takt-Vor/Rück Ausgänge für weitere Achsen**

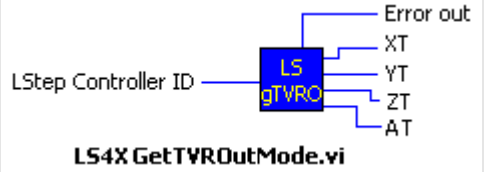
| LS_GetAccelTVRO | |
|---|---|
| **Beschreibung:** | Liest die eingestellten Beschleunigungen für die weiteren Achsen |
| **Delphi:** | function LS_GetAccelTVRO(var X, Y, Z, A: Double): Integer;<br>function LSX_GetAccelTVRO(LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetAccelTVRO (double *pdX, double *pdY, double *pdZ, double *pdA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z, A: Beschleunigungswerte, U/s² |
| **Beispiel:** | LS.GetAccelTVRO(&X, &Y, &Z, &A); |

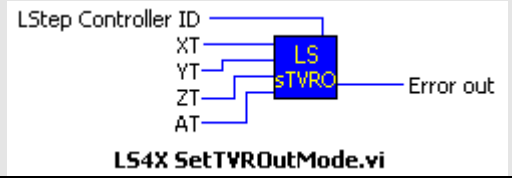
| LS_SetAccelTVRO | |
|---|---|
| **Beschreibung:** | Beschleunigung für die weiteren Achsen einstellen |
| **Delphi:** | function LS_SetAccelTVRO(X, Y, Z, A: Double): Integer;<br>function LSX_SetAccelTVRO(LSID: Integer; X, Y, Z, A: Double): Integer; |
| **C++:** | int SetAccelTVRO (double dX, double dY, double dZ, double dA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: Beschleunigungen, Wertebereich 0.01 – 1500 [U/s²] |
| **Beispiel:** | LS.SetAccelTVRO(1.0, 1.5, 0, 0); |

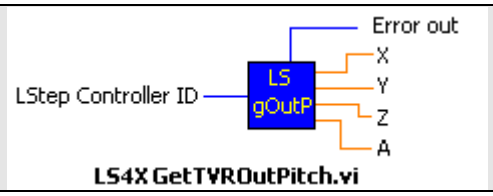
| LS_GetPosTVRO | |
|---|---|
| **Beschreibung:** | Position der zusätzlichen Achsen ablesen |
| **Delphi:** | function LS_GetPosTVRO(var dX, dY, dZ, dR: Double): Integer;<br>function LSX_GetPosTVRO(LSID: Integer; var dX, dY, dZ, dR: Double): Integer; |
| **C++:** | int GetPosTVRO (double *pdX, double *pdY, double *pdZ, double *pdR); |
| **LabView:** |  |
| **Parameter:** | dX, dY, dZ, dR: Positionswert, abhängig von der Dimension |
| **Beispiel:** | LS. GetPosTVRO(&X, &Y, &Z, &R); |

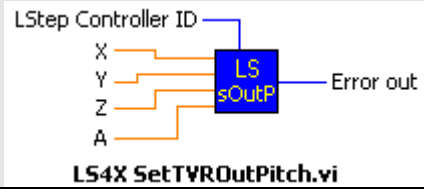
| LS_SetPosTVRO | |
|---|---|
| **Beschreibung:** | Position der zusätzlichen Achsen setzen |
| **Delphi:** | function LS_ SetPosTVRO(dX, dY, dZ, dR: Double): Integer;<br>function LSX_ SetPosTVRO(LSID: Integer; dX, dY, dZ, dR: Double): Integer; |
| **C++:** | int SetPosTVRO (double dX, double dY, double dZ, double dR); |
| **LabView:** |  |
| **Parameter:** | dX, dY, dZ, dR: Positionswert, in Abhängigkeit von Dimension<br>Wertebereich: min. Bereichsgrenze bis max. Bereichsgrenze |
| **Beispiel:** | LS. SetPosTVRO(10.0, 5.0, 0.0, 0.0); |

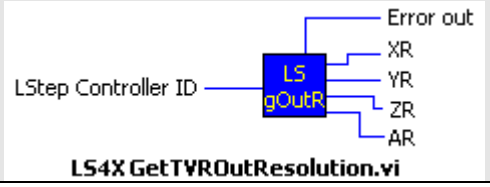
| LS_GetStatusTVRO | |
|---|---|
| **Beschreibung:** | Liefert den aktuellen Status der zusätzlichen Achsen |
| **Delphi:** | function LS_GetStatusTVRO(pcStat: PChar; MaxLen: Integer): Integer;<br>function LSX_ GetStatusTVRO(LSID: Integer; pcStat: PChar; MaxLen: Integer): Integer; |
| **C++:** | int GetStatusTVRO (char *pcStat, int lMaxLen); |
| **LabView:** |  |
| **Parameter:** | pcStat: Zeiger auf einen Puffer, in dem der Statusstring zurückgegeben wird<br>MaxLen: Maximale Anzahl von Zeichen, die in den Puffer kopiert werden dürfen<br><br>z.B.: @ - M –<br>@ = Achse steht<br>M = Achse ist in Bewegung (Motion)<br>- = Achse ist nicht freigegeben |
| **Beispiel:** | LS. GetStatusTVRO(pcStat, 256);<br>// Die zusätzliche Z-Achse um 5mm in positiver Richtung verfahren |

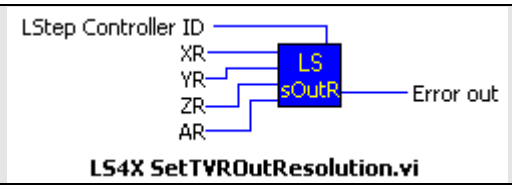
| LS_GetTVROutMode | |
|---|---|
| | |
| **Delphi:** | function LS_GetTVROutMode(var X, Y, Z, A: Integer): Integer;<br>function LSX_GetTVROutMode(LSID: Integer; var X, Y, Z, A: Integer): Integer; |
| **C++:** | int GetTVROutMode (int *plXT, int *plYT, int *plZT, int *plAT); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: 0 => Takt Vor/Rück ist "AUS"<br>1 => Takt Vor/Rück ist "EIN" |
| **Beispiel:** | LS.GetTVROutMode(&X, &Y, &Z, &A); |

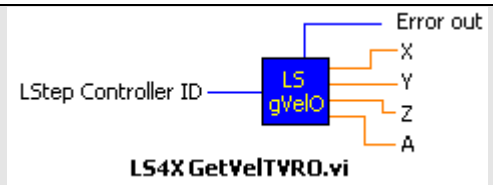
| LS_SetTVROutMode | |
|---|---|
| **Beschreibung:** | Zusetzliche Achsen X, Y, Z und A, neben den eigentlichen Hauptachsen X, Y, Z und A, setzen |
| **Delphi:** | function LS_SetTVROutMode(X, Y, Z, A: Integer): Integer;<br>function LSX_SetTVROutMode(LSID: Integer; X, Y, Z, A: Integer): Integer; |
| **C++:** | int SetTVROutMode (int lXT, int lYT, int lZT, int lAT); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: 0 oder 1 |
| **Beispiel:** | LS.SetTVROutMode(1, 0, 1, 0); //Bei Achsen x und z soll Takt Vor/Rück aktiviert werden, bei y und a – deaktiviert. |

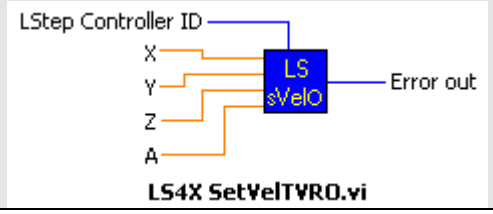
| LS_GetTVROutPitch | |
|---|---|
| **Beschreibung:** | Liest die Spindelsteigungen für die zusätzlichen Achsen |
| **Delphi:** | function LS_GetTVROutPitch(var X, Y, Z, R: Double): Integer;<br>function LSX_ GetTVROutPitch (LSID: Integer; var X, Y, Z, R: Double): Integer; |
| **C++:** | int GetTVROutPitch (double *pdX, double *pdY, double *pdZ, double *pdR); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und R: Spindelsteigungen [mm] |
| **Beispiel:** | LS. GetTVROutPitch(&X, &Y, &Z, &A); |

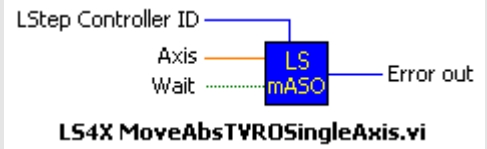
| LS_SetTVROutPitch | |
|---|---|
| **Beschreibung:** | Setzt die Spindelsteigungen für die zusätzlichen Achsen |
| **Delphi:** | function LS_SetTVROutPitch(X, Y, Z, R: Double): Integer;<br>function LSX_ SetTVROutPitch (LSID: Integer; X, Y, Z, R: Double): Integer; |
| **C++:** | int SetTVROutPitch (double dX, double dY, double dZ, double dR); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und R: Spindelsteigungen [mm], Wertebereich 0.001 bis 100 |
| **Beispiel:** | LS. SetTVROutPitch(1.0, 4.0, 1.0, 1.0);<br>/* Spindelsteigung für y-Achse beträgt 4 mm. Für x-, z- und a-Achsen werden Spindeln mit 1mm Steigung eingesetzt*/ |

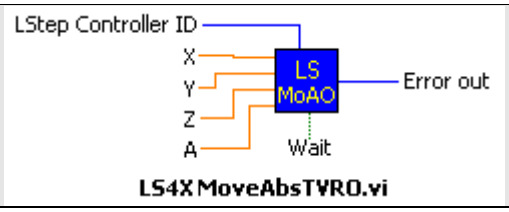
| LS_GetTVROutResolution | |
|---|---|
| **Beschreibung:** | Liest die Auflösung der an zu steuernden Endstufe |
| **Delphi:** | function LS_GetTVROutResolution(var X, Y, Z, A: Integer): Integer;<br>function LSX_ GetTVROutResolution (LSID: Integer; var X, Y, Z, A: Integer): Integer; |
| **C++:** | int GetTVROutResolution (int *plX, int *plY, int *plZ, int *plA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: Impulsen pro Umdrehung |
| **Beispiel:** | LS. GetTVROutResolution (&X, &Y, &Z, &A); |

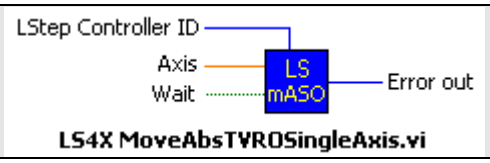
| LS_SetTVROutResolution | |
|---|---|
| **Beschreibung:** | Auflösung der an zu steuernden Endstufe einstellen |
| **Delphi:** | function LS_SetTVROutResolution(X, Y, Z, A: Integer): Integer;<br>function LSX_ SetTVROutResolution (LSID: Integer; X, Y, Z, A: Integer): Integer; |
| **C++:** | int SetTVROutResolution (int lX, int lY, int lZ, int lA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: Impulsen pro Umdrehung, Wertebereich 0 bis 51200 |
| **Beispiel:** | LS. SetTVROutResolution (1000, 1000, 0, 0);<br>/* Bei Achse X und Y wird eine Auflösung von 1000 Impulsen pro Umdrehung eingestellt*/ |

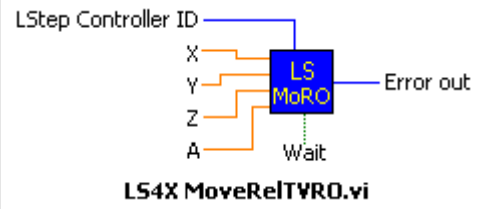
| LS_GetVelTVRO | |
|---|---|
| **Beschreibung:** | Liest die eingestellten Geschwindigkeiten für die weiteren Achsen |
| **Delphi:** | function LS_GetVelTVRO(var X, Y, Z, A: Double): Integer;<br>function LSX_GetVelTVRO(LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetVelTVRO (double *pdX, double *pdY, double *pdZ, double *pdA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z, A: Geschwindigkeitswerte, [U/s] |
| **Beispiel:** | LS.GetVelTVRO(&X, &Y, &Z, &A); |

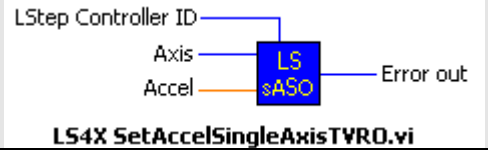
| LS_SetVelTVRO | |
|---|---|
| **Beschreibung:** | Geschwindigkeit für die weiteren Achsen einstellen |
| **Delphi:** | function LS_SetVelTVRO(X, Y, Z, A: Double): Integer;<br>function LSX_SetVelTVRO(LSID: Integer; X, Y, Z, A: Double): Integer; |
| **C++:** | int SetVelTVRO (double dX, double dY, double dZ, double dA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: Geschwindigkeiten, 0 – 40.0 [U/s] |
| **Beispiel:** | LS.SetVelTVRO(1.0, 1.5, 0, 0); |

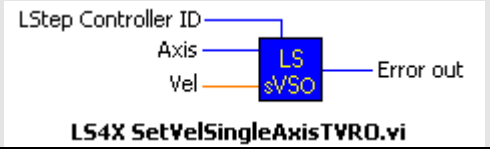
| LS_MoveAbsTVROSingleAxis | |
|---|---|
| **Beschreibung:** | Einzelne Achse absolut positionieren |
| **Delphi:** | function LS_MoveAbsTVROSingleAxis (Axis: Integer; Value: Double; Wait: LongBool): Integer;<br>function LSX_MoveAbsTVROSingleAxis (LSID: Integer; Axis: Integer; Value: Double; Wait: LongBool): Integer; |
| **C++:** | int MoveAbsTVROSingleAxis (int lAxis, double dValue, BOOL bWait); |
| **LabView:** |  |
| **Parameter:** | Axis: (X, Y, Z, A numeriert von 1 bis 4)<br>Value: Position (Eingabe ist abhängig von der eingestellten Dimension) |
| **Beispiel:** | LS.MoveAbsTVROSingleAxis (2, 10.0);<br>//Die zusätzliche Y-Achse auf 10mm absolut positionieren |

| LS_MoveAbsTVRO | |
|---|---|
| **Beschreibung:** | Absolutposition anfahren<br>(Die zusätzlichen Achsen x, y, z und a werden auf die übergebenen Positionswerte positioniert.) |
| **Delphi:** | function LS_MoveAbsTVRO(X, Y, Z, A: Double; Wait: LongBool): Integer;<br>function LSX_MoveAbsTVRO (LSID: Integer; X, Y, Z, A: Double; Wait: LongBool): Integer; |
| **C++:** | int MoveAbsTVRO (double dX, double dY, double dZ, double dR, BOOL bWait); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A<br>+- Verfahrbereich<br>Eingabe ist abhängig von der Dimension<br>Wait: Gibt an, ob die Funktion nachdem die Position erreicht wurde (= true) oder direkt zurückkehren soll (= false) |
| **Beispiel:** | LS.MoveAbsTVRO (10.0, 10.0, 10.0, 10.0, true); |

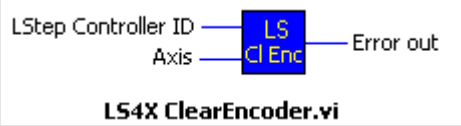
| LS_MoveRelTVROSingleAxis | |
|---|---|
| **Beschreibung:** | Einzelne Achse relativ verfahren |
| **Delphi:** | function LS_MoveRelTVROSingleAxis (Axis: Integer; Value: Double; Wait: LongBool): Integer;<br>function LSX_MoveRelTVROSingleAxis (LSID: Integer; Axis: Integer; Value: Double; Wait: LongBool): Integer; |
| **C++:** | int MoveRelTVROSingleAxis (int lAxis,double dValue,BOOL Wait); |
| **LabView:** |  |
| **Parameter:** | Axis: (X, Y, Z, A numeriert von 1 bis 4)<br>Value: Strecke (Eingabe ist abhängig von der eingestellten Dimension) |
| **Beispiel:** | LS.MoveRelTVROSingleAxis (3, 5.0);<br>// Die zusätzliche Z-Achse um 5mm in positiver Richtung verfahren |

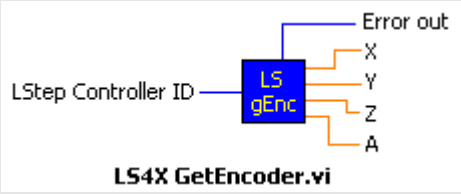
| LS_MoveRelTVRO | |
|---|---|
| **Beschreibung:** | Relativen Vektor fahren<br><br>(Die zusätzlichen Achsen x, y, z und a werden um die übergebenen Strecken verfahren.) |
| **Delphi:** | function LS_MoveRelTVRO(X, Y, Z, A: Double; Wait: LongBool): Integer;<br>function LSX_MoveRelTVRO(LSID: Integer; 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 und A<br>+- Verfahrbereich<br>Eingabe ist abhängig von der Dimension<br><br>Wait: Gibt an, ob die Funktion nachdem die Position erreicht wurde (= true) oder direkt zurückkehren soll (= false) |
| **Beispiel:** | LS.MoveRelTVRO(10.0, 10.0, 10.0, 10.0, true); |

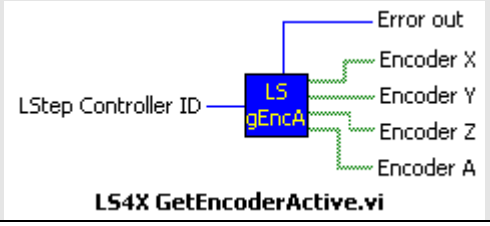
| LS_SetAccelSingleAxisTVRO | |
|---|---|
| **Beschreibung:** | Beschleunigung für einzelne zusätzliche Achse einstellen |
| **Delphi:** | function LS_SetAccelSingleAxisTVRO(Axis: Integer; Accel: Double): Integer;<br>function LSX_SetAccelSingleAxisTVRO (LSID: Integer; Axis: Integer; Accel: Double): Integer; |
| **C++:** | int SetAccelSingleAxisTVRO (int lAxis, double dAccel); |
| **LabView:** |  |
| **Parameter:** | Axis: (X, Y, Z, A numeriert von 1 bis 4)<br>Accel: 0.01 – 1500 [U/s²] |
| **Beispiel:** | LS.SetAccelSingleAxis(2, 50.0);<br>// Die Z-Achse soll wird mit 50 U/s² beschleunigt |

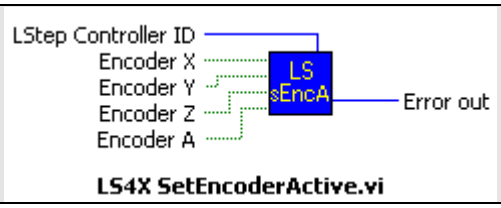
| LS_SetVelSingleAxisTVRO | |
|---|---|
| **Beschreibung:** | Geschwindigkeit für einzelne zusätzliche Achse einstellen |
| **Delphi:** | function LS_SetVelSingleAxisTVRO(Axis: Integer; Vel: Double): Integer;<br>function LSX_SetVelSingleAxisTVRO (LSID: Integer; Axis: Integer; Vel: Double): Integer; |
| **C++:** | int SetVelSingleAxisTVRO (int lAxis, double dVel); |
| **LabView:** |  |
| **Parameter:** | Axis: (X, Y, Z, A numeriert von 1 bis 4)<br>Vel: 0 – 40.0 [U/s] |
| **Beispiel:** | LS.SetVelSingleAxis(1, 10.0);<br>// Die X-Achse soll mit einer max. Geschwindigkeit von 10 U/s betrieben werden |

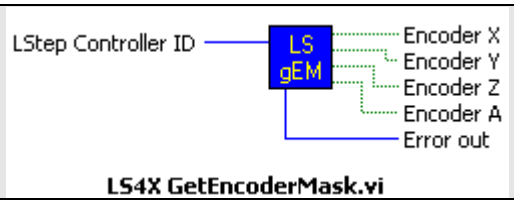
### Geber-Einstellungen

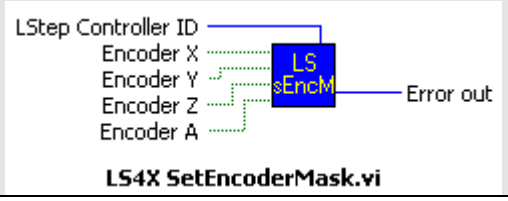
| LS_ClearEncoder | |
|---|---|
| **Beschreibung:** | Geber-Zähler auf null setzen |
| **Delphi:** | function LS_ClearEncoder(lAxis: Integer): Integer;<br>function LSX_ClearEncoder (LSID: Integer; lAxis: Integer): Integer; |
| **C++:** | int ClearEncoder (int lAxis); |
| **LabView:** |  |
| **Parameter:** | lAxis: (X, Y, Z, A numeriert von 1 bis 4) |
| **Beispiel:** | LS. ClearEncoder (2); //Geber-Zähler der y-Achse auf null setzen |

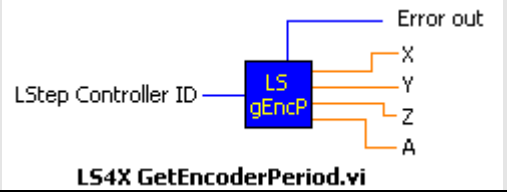
| LS_GetEncoder | |
|---|---|
| **Beschreibung:** | Liest alle Geber-Positionen |
| **Delphi:** | function LS_GetEncoder(XP, YP, ZP, AP: Double): Integer;<br>function LSX_GetEncoder (LSID: Integer; XP, YP, ZP, AP: Double): Integer; |
| **C++:** | int GetEncoder (double *pdXP, double *pdYP, double *pdZP, double *pdRP); |
| **LabView:** |  |
| **Parameter:** | XP, YP, ZP, AP: Zählerwerte, 4-fach interpoliert |
| **Beispiel:** | LS. GetEncoder (&XP, &YP, &ZP, &AP); |

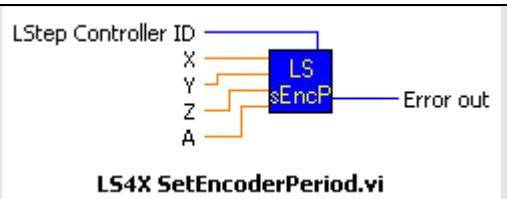
| LS_GetEncoderActive | |
|---|---|
| **Beschreibung:** | Liest, welche Geber nach der Kalibration aktiviert werden. |
| **Delphi:** | function LS_GetEncoderActive(var Flags: Integer): Integer;<br>function LSX_GetEncoderActive(LSID: Integer; var Flags: Integer): Integer; |
| **C++:** | int GetEncoderActive (int *plFlags); |
| **LabView:** |  |
| **Parameter:** | Flags: Gebermaske |
| **Beispiel:** | LS.GetEncoderActive(&Flags); |

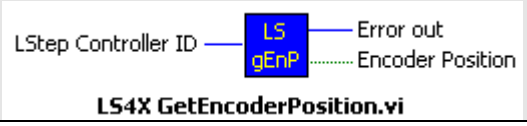
| LS_SetEncoderActive | |
|---|---|
| **Beschreibung:** | Mit dieser Funktion kann ausgewählt werden, welche Geber nach der Kalibration aktiviert werden sollen. |
| **Delphi:** | function LS_SetEncoderActive(Flags: Integer): Integer;<br>function LSX_SetEncoderActive(LSID: Integer; Flags: Integer): Integer; |
| **C++:** | int SetEncoderActive (int lFlags); |
| **LabView:** |  |
| **Parameter:** | Value: Gebermaske |
| **Beispiel:** | LS.SetEncoderActive(0);<br>// Alle Geber deaktivieren<br>LS.SetEncoderMask(2);<br>// Geber Y-Achse aktivieren |

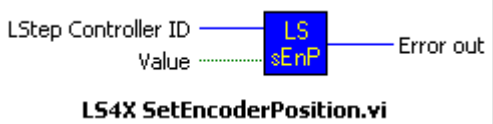
| LS_GetEncoderMask | |
|---|---|
| **Beschreibung:** | Geberzustände auslesen |
| **Delphi:** | function LS_GetEncoderMask (var Flags: Integer): Integer;<br>function LSX_GetEncoderMask(LSID: Integer; var Flags: Integer): Integer; |
| **C++:** | int GetEncoderMask (int *plFlags); |
| **LabView:** |  |
| **Parameter:** | Flags: Gebermaske |
| **Beispiel:** | int EncMask;<br>LS.GetEncoderMask(&EncMask);<br>if (EncMask & 2) ...<br>// Wenn Geber Y-Achse angeschlossen+aktiv |

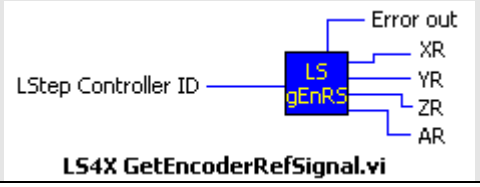
| LS_SetEncoderMask | |
|---|---|
| **Beschreibung:** | Geber (de-)aktivieren |
| **Delphi:** | function LS_SetEncoderMask(Value: Integer): Integer;<br>function LSX_SetEncoderMask(LSID: Integer; Value: Integer): Integer; |
| **C++:** | int SetEncoderMask (int lValue); |
| **LabView:** |  |
| **Parameter:** | Value: Gebermaske |
| **Beispiel:** | LS.SetEncoderMask(0);<br>// Alle Geber deaktivieren<br>LS.SetEncoderMask(2);<br>// Geber Y-Achse aktivieren |

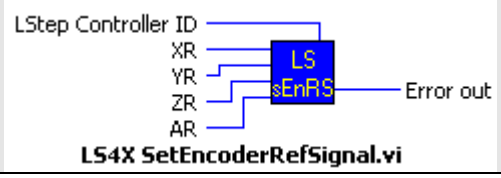
| LS_GetEncoderPeriod | |
|---|---|
| **Beschreibung:** | Geberperiodenlängen auslesen |
| **Delphi:** | function LS_GetEncoderPeriod(var X, Y, Z, A: Double): Integer;<br>function LSX_GetEncoderPeriod(LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetEncoderPeriod (double *pdX, double *pdY, double *pdZ, double *pdR); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: Periodenlängen [mm] |
| **Beispiel:** | LS.GetEncoderPeriod(&X, &Y, &Z, &A); |

| LS_SetEncoderPeriod | |
|---|---|
| **Beschreibung:** | Geberperiodenlängen einstellen |
| **Delphi:** | function LS_SetEncoderPeriod(X, Y, Z, A: Double): Integer;<br>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 und A<br>0.0001 – Spindelsteigung * 0.8 (mm) |
| **Beispiel:** | LS.SetEncoderPeriod(0.1, 0.1, 0.1, 0.1);<br>// Geberperiodenlänge aller Achsen ist 0.1mm |

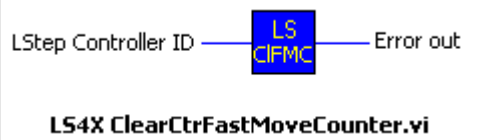
| LS_GetEncoderPosition | |
|---|---|
| **Beschreibung** | Einstellung von der Geberwertanzeige lesen |
| **Delphi** | function LS_GetEncoderPosition(Value: Boolean): Integer;<br>function LSX_GetEncoderPosition(LSID: Integer; Value: LongBool): Integer; |
| **C++** | int GetEncoderPosition (BOOL *pbValue); |
| **LabView** |  |
| **Parameter** | Value: true → Bei der Positionsabfrage werden die Geberwerte der erkannten Geber angezeigt<br>false → Geberpositionsanzeige ist "AUS" |
| **Beispiel** | LS.GetEncoderPosition(&Value); |

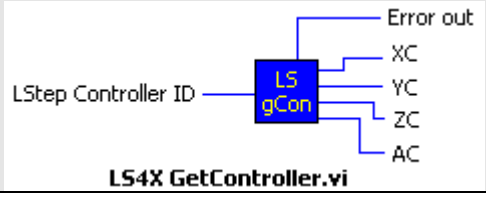
| LS_SetEncoderPosition | |
|---|---|
| **Beschreibung** | Geberwertanzeige Ein/Aus |
| **Delphi** | function LS_SetEncoderPosition(Value: Boolean): Integer;<br>function LSX_SetEncoderPosition(LSID: Integer; Value: LongBool): Integer; |
| **C++** | int SetEncoderPosition (BOOL fValue); |
| **LabView** |  |
| **Parameter** | Value = true → Bei der Positionsabfrage werden die Geberwerte der erkannten Geber angezeigt |
| **Beispiel** | LS.SetEncoderPosition(true); |

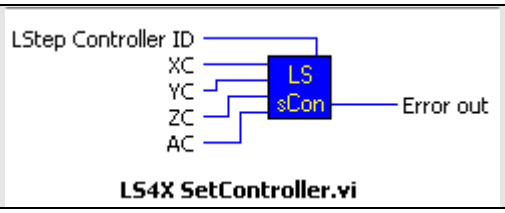
| LS_GetEncoderRefSignal | |
|---|---|
| **Beschreibung:** | Liest, ob beim Kalibrieren Referenzsignal von Geber ausgewertet wird |
| **Delphi:** | function LS_GetEncoderRefSignal(var XR, YR, ZR, AR: Integer): Integer;<br>function LSX_GetEncoderRefSignal(LSID: Integer; var XR, YR, ZR, AR: Integer): Integer; |
| **C++:** | int GetEncoderRefSignal (int *plXR, int *plYR, int *plZR, int *plRR); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: 1 => Beim Kalibrieren wird das Referenzsignal ausgewertet<br>0 => Das Referenzsignal wird nicht ausgewertet |
| **Beispiel:** | LS.GetEncoderRefSignal(&X, &Y, &Z, &A); |

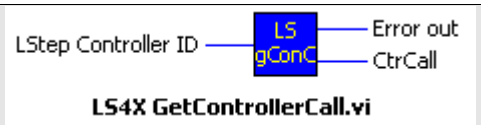
| LS_SetEncoderRefSignal | |
|---|---|
| **Beschreibung:** | Beim Kalibrieren Referenzsignal von Geber auswerten |
| **Delphi:** | function LS_SetEncoderRefSignal(XR, YR, ZR, AR: Integer): Integer;<br>function LSX_SetEncoderRefSignal(LSID: Integer; XR, YR, ZR, AR: Integer): Integer; |
| **C++:** | int SetEncoderRefSignal (int lXR,int lYR,int lZR,int lAR); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A<br>0 oder 1 |
| **Beispiel:** | LS.SetEncoderRefSignal(1, 1, 0, 0);<br>/* Beim Kalibrieren wird das Referenzsignal der Geber x und y ausgewertet. */ |

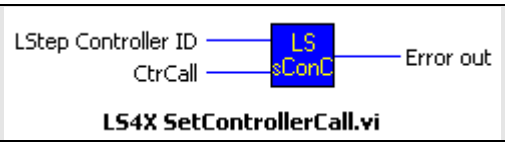
**Reglereinstellungen**

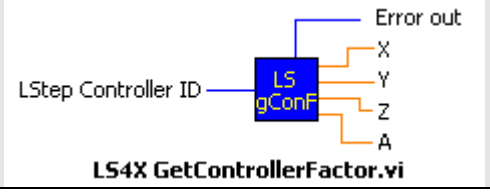
| **LS_ClearCtrFastMoveCounter** | |
|---|---|
| **Beschreibung:** | Bei einer Reglerdifferenz, die größer als der Fangbereich ist, wird ein neuer Vektor gestartet und der dazu gehörige Counter um eins erhöht.<br>Die Function setzt Fast Move Counters aller Achsen auf null. |
| **Delphi:** | function LS_ClearCtrFastMoveCounter: Integer;<br>function LSX_ClearCtrFastMoveCounter(LSID: Integer): Integer; |
| **C++:** | int ClearCtrFastMoveCounter; |
| **LabView:** |  |
| **Parameter:** | |
| **Beispiel:** | LS. ClearCtrFastMoveCounter; |

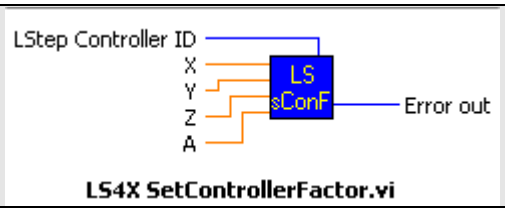
| **LS_GetController** | |
|---|---|
| **Beschreibung:** | Regler-Modus auslesen |
| **Delphi:** | function LS_GetController(var XC, YC, ZC, AC: Integer): Integer;<br>function LSX_GetController(LSID: Integer; var XC, YC, ZC, AC: Integer): Integer; |
| **C++:** | int GetController (int *plXC, int *plYC, int *plZC, int *plRC); |
| **LabView:** |  |
| **Parameter:** | Regler-Modus X, Y, Z und A : |
| | 0 ➔ Regler „AUS“ |
| | 1 ➔ Regler „AUS nach erreichen der Zielposition“ |
| | 2 ➔ Regler „Immer EIN“ |
| | 3 ➔ Regler „AUS nach erreichen der Zielposition“ mit reduziertem Strom |
| | 4 ➔ Regler „Immer EIN“ mit reduziertem Strom |
| **Beispiel:** | LS.GetController(&X, &Y, &Z, &A); |

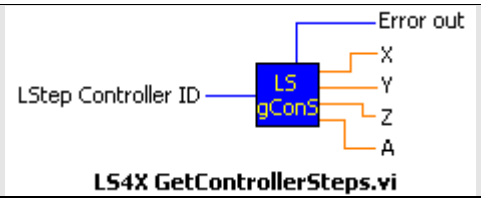
| LS_SetController | |
|---|---|
| **Beschreibung:** | Regler-Modus einstellen |
| **Delphi:** | function LS_SetController(XC, YC, ZC, AC: Integer): Integer;<br>function LSX_SetController(LSID: Integer; XC, YC, ZC, AC: Integer): Integer; |
| **C++:** | int SetController (int lXC,int lYC,int lZC,int lAC); |
| **LabView:** |  |
| **Parameter:** | Regler-Modus X, Y, Z und A : |
| | 0 ➔ Regler „AUS“<br>1 ➔ Regler „AUS nach erreichen der Zielposition“<br>2 ➔ Regler „Immer EIN“ |
| **zeit** | 3 ➔ Regler „AUS nach erreichen der Zielposition“ mit reduziertem Strom |
| | 4 ➔ Regler „Immer EIN“ mit reduziertem Strom |
| **Beispiel:** | LS.SetController(1, 2, 0, 0); |

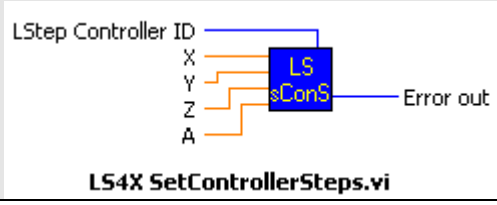
| LS_GetControllerCall | |
|---|---|
| **Beschreibung:** | Liefert Regleraufrufzeit |
| **Delphi:** | function LS_GetControllerCall(var CtrCall: Integer): Integer;<br>function LSX_GetControllerCall(LSID: Integer; var CtrCall: Integer): Integer; |
| **C++:** | int GetControllerCall (int *plCtrCall); |
| **LabView:** |  |
| **Parameter:** | CtrCall: Regleraufrufzeit [ms] |
| **Beispiel:** | LS.GetControllerCall(&CtrCall); //Nach dem Funktionsaufruf CtrCall = 10 bedeutet: Regleraufruf alle 10 ms |

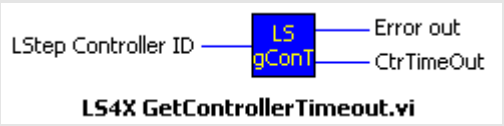
| LS_SetControllerCall | |
|---|---|
| **Beschreibung:** | Regleraufruf |
| **Delphi:** | function LS_SetControllerCall(CtrCall: Integer): Integer;<br>function LSX_SetControllerCall(LSID: Integer; CtrCall: Integer): Integer; |
| **C++:** | int SetControllerCall (int lCtrCall); |
| **LabView:** |  |
| **Parameter:** | CtrCall: Regleraufrufzeit [ms] |
| **Beispiel:** | LS.SetControllerCall(10); |

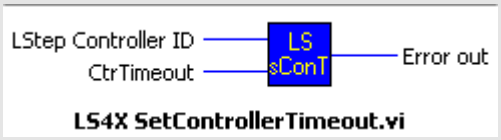
| LS_GetControllerFactor | |
|---|---|
| **Beschreibung:** | Regler-Faktoren auslesen<br>siehe Kap. 4.13 „Reglereinstellunen für LSTEP“ |
| **Delphi:** | function LS_GetControllerFactor(var X, Y, Z, A: Double): Integer;<br>function LSX_GetControllerFactor(LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetControllerFactor (double *pdX, double *pdY, double *pdZ, double *pdR); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: Regler-Faktoren |
| **Beispiel:** | LS.GetControllerFactor(&X, &Y, &Z, &A); |

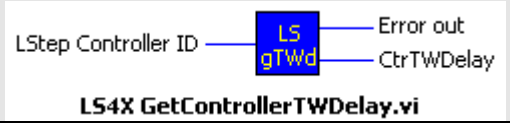
| LS_SetControllerFactor | |
|---|---|
| **Beschreibung:** | Regler-Faktor |
| **Delphi:** | function LS_SetControllerFactor(X, Y, Z, A: Double): Integer;<br>function LSX_SetControllerFactor(LSID: Integer; X, Y, Z, A: Double): Integer; |
| **C++:** | int SetControllerFactor (double dX,double dY,double dZ,double dA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A<br>1 – 64 |

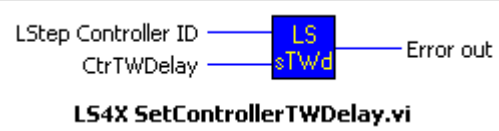
| LS_GetControllerSteps | |
|---|---|
| **Beschreibung:** | Liefert die Regler-Schrittenlänge |
| **Delphi:** | function LS_GetControllerSteps(var X, Y, Z, A: Double): Integer;<br>function LSX_GetControllerSteps(LSID: Integer; var X, Y, Z, A: Double): Integer; |
| **C++:** | int GetControllerSteps (double *pdX, double *pdY, double *pdZ, double *pdR); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A: Regler-Schrittenlänge [mm] |
| **Beispiel:** | LS.GetControllerSteps(&X, &Y, &Z, &A); |

| LS_SetControllerSteps | |
|---|---|
| **Beschreibung:** | Regler-Schritte |
| **Delphi:** | function LS_SetControllerSteps(X, Y, Z, A: Double): Integer;<br>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 und A<br>1 – Spindelsteigung<br>(Werte sind abhängig von der Dimension) |
| **Beispiel:** | LS.SetControllerSteps(4, 5, 7, 9); |

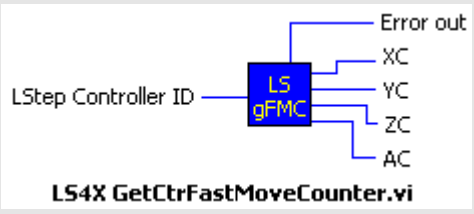
| LS_GetControllerTimeout | |
|---|---|
| **Beschreibung:** | Liest Regler-Timeout |
| **Delphi:** | function LS_GetControllerTimeout(var ACtrTimeout: Integer): Integer;<br>function LSX_GetControllerTimeout(LSID: Integer; var ACtrTimeout: Integer): Integer; |
| **C++:** | int GetControllerTimeout (int *plACtrTimeout); |
| **LabView:** |  |
| **Parameter:** | ACtrTimeout: Timeout [ms], nach dem ein Verfahrbefehl mit Fehlermeldung (Fehlercode 4013) zurückkehrt, wenn die Steuerung eine Position nicht endgültig finden konnte. |
| **Beispiel:** | LS.GetControllerTimeout(&ACtrTimeout); |

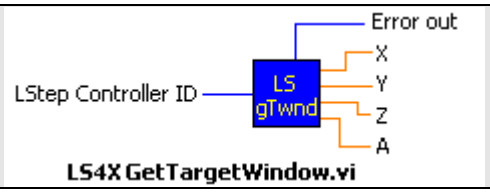
| LS_SetControllerTimeout | |
|---|---|
| **Beschreibung:** | Regler-Timeout |
| **Delphi:** | function LS_SetControllerTimeout(ACtrTimeout: Integer): Integer;<br>function LSX_SetControllerTimeout(LSID: Integer; ACtrTimeout: Integer): Integer; |
| **C++:** | int SetControllerTimeout (int ACtrTimeout); |
| **LabView:** |  |
| **Parameter:** | ACtrTimeout: Timeout [ms], nach dem ein Verfahrbefehl mit Fehlermeldung (Fehlercode 4013) zurückkehrt, wenn die Steuerung eine Position nicht endgültig finden konnte. |
| **Beispiel:** | LS.SetControllerTimeout(500); |

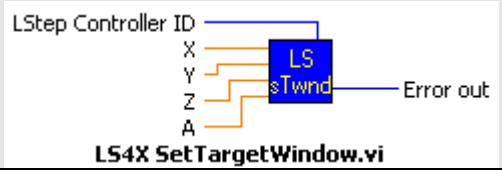
| LS_GetControllerTWDelay | |
|---|---|
| **Beschreibung** | Regler-Verzögerung auslesen |
| **Delphi** | function LS_GetControllerTWDelay(var CtrTWDelay: Integer): Integer;<br>function LSX_GetControllerTWDelay(LSID: Integer; var CtrTWDelay: Integer): Integer; |
| **C++** | int GetControllerTWDelay (int *plCtrTWDelay); |
| **LabView** |  |
| **Parameter** | CtrTWDelay: Regler-Verzögerung [ms] |
| **Beispiel** | LS.GetControllerTWDelay(&CtrTWDelay); |

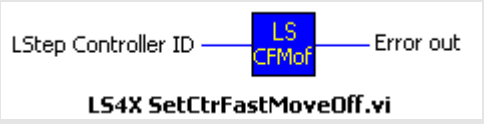
| LS_SetControllerTWDelay | |
|---|---|
| **Beschreibung** | Regler-Verzögerung |
| **Delphi** | function LS_SetControllerTWDelay(CtrTWDelay: Integer): Integer;<br>function LSX_SetControllerTWDelay(LSID: Integer; CtrTWDelay: Integer): Integer; |
| **C++** | int SetControllerTWDelay (int lCtrTWDelay); |
| **LabView** |  |
| **Parameter** | CtrTWDelay: Regler-Verzögerung 0 – 100 [ms] |
| **Beispiel** | LS.SetControllerTWDelay(0);<br>// Regler-Verzögerung Aus |

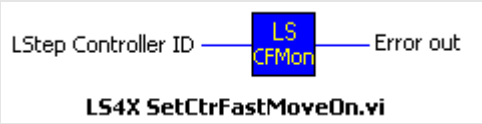
| LS_GetCtrFastMove | |
|---|---|
| **Beschreibung:** | Liest die Einstellung von der Fast Move Funktion |
| **Delphi:** | function LS_ GetCtrFastMoveOff(var bActive: LongBool): Integer;<br>function LSX_ GetCtrFastMoveOff(LSID: Integer; var bActive: LongBool): Integer; |
| **C++:** | int GetCtrFastMove (BOOL *pbActive); |
| **LabView:** |  |
| **Parameter:** | bActive: True => Fast Move Funktion aktiv |
| **Beispiel:** | LS. GetCtrFastMoveOff (&bActive); |

| LS_GetCtrFastMoveCounter | |
|---|---|
| **Beschreibung:** | Bei einer Reglerdifferenz, die größer als der Fangbereich ist, wird ein neuer Vektor gestartet und der dazu gehörige Counter um eins erhöht.<br>Die Function liefert Fast Move Counters |
| **Delphi:** | function LS_ GetCtrFastMoveCounter(var XC, YC, ZC, RC: Integer): Integer;<br>function LSX_ GetCtrFastMoveCounter(LSID: Integer; var XC, YC, ZC, RC: Integer): Integer; |
| **C++:** | int GetCtrFastMoveCounter (int *plXC, int *plYC, int *plZC, int *plRC); |
| **LabView:** |  |
| **Parameter:** | XC, YC, ZC, RC: Anzahl ausgeführter Fast Move Funktionen |
| **Beispiel:** | LS. SetCtrFastMoveCounter (&XC, &YC, &ZC, &RC); |

| LS_GetTargetWindow | |
|---|---|
| **Beschreibung:** | Liest die Zielfenster aller Achsen |
| **Delphi:** | function LS_GetTargetWindow(var X, Y, Z, A: Double): Integer;<br>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 und A: Zielfenster , abhängig von der Dimension |
| **Beispiel:** | LS.GetTargetWindow(&X, &Y, &Z, &A); |

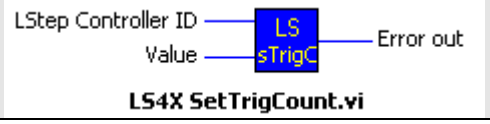
| LS_SetTargetWindow | |
|---|---|
| **Beschreibung:** | Zielfenster |
| **Delphi:** | function LS_SetTargetWindow(X, Y, Z, A: Double): Integer;<br>function LSX_SetTargetWindow(LSID: Integer; X, Y, Z, A: Double): Integer; |
| **C++:** | int SetTargetWindow (double dX,double dY,double dZ,double dA); |
| **LabView:** |  |
| **Parameter:** | X, Y, Z und A<br>1 – 25000 (Motorincremente)<br>0.1 – Spindelsteigung/2 (µm)<br>0.0001 – Spindelsteigung/2 (mm)<br>(Werte sind abhängig von der Dimension) |
| **Beispiel:** | LS.SetTargetWindow(1.0, 0.002, 1.0, 1.0); |

| LS_SetCtrFastMoveOff | |
|---|---|
| **Beschreibung:** | Fast Move Funktion deaktivieren |
| **Delphi:** | function LS_ SetCtrFastMoveOff: Integer;<br>function LSX_ SetCtrFastMoveOff(LSID: Integer): Integer; |
| **C++:** | int SetCtrFastMoveOff (); |
| **LabView:** |  |
| **Parameter:** | |
| **Beispiel:** | LS. SetCtrFastMoveOff (); |

| LS_SetCtrFastMoveOn | |
|---|---|
| **Beschreibung:** | Fast Move Funktion aktivieren, d. h. bei einer Reglerdifferenz, die größer als der Fangbereich ist, wird ein neuer Vektor gestartet. |
| **Delphi:** | function LS_ SetCtrFastMoveOn: Integer;<br>function LSX_ SetCtrFastMoveOn(LSID: Integer): Integer; |
| **C++:** | int SetCtrFastMoveOn (); |
| **LabView:** |  |
| **Parameter:** | |
| **Beispiel:** | LS. SetCtrFastMoveOn(); |

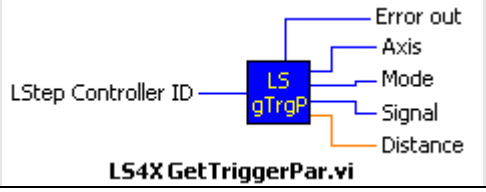
**Trigger-Ausgang**

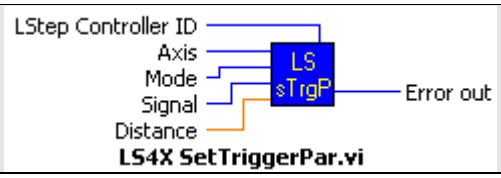
| LS_GetTrigCount | |
|---|---|
| **Beschreibung:** | Triggerzählerstand lesen. |
| **Delphi:** | function LS_GetTrigCount (var Value: Integer): Integer;<br>function LSX_GetTrigCount (LSID: Integer; var Value: Integer): Integer; |
| **C++:** | int GetTrigCount (int *pValue); |
| **LabView:** |  |
| **Parameter:** | Value: Anzahl der ausgeführten Trigger |
| **Beispiel:** | LS.GetTrigCount (&Value); |

| LS_SetTrigCount | |
|---|---|
| **Beschreibung:** | Triggerzählerstand setzen. |
| **Delphi:** | function LS_ SetTrigCount (Value: Integer): Integer;<br>function LSX_ SetTrigCount (LSID: Integer; Value: Integer): Integer; |
| **C++:** | int SetTrigCount (int Wert); |
| **LabView:** |  |
| **Parameter:** | Value: 0 bis 2147483647 |
| **Beispiel:** | LS.SetTrigCount (0); |

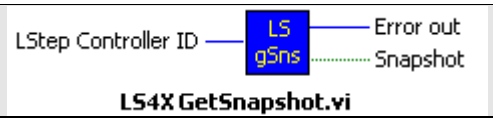
| LS_GetTrigger | |
|---|---|
| **Beschreibung:** | Liefert den aktuellen Trigger-Zustand |
| **Delphi:** | function LS_GetTrigger(var ATrigger: LongBool): Integer;<br>function LSX_GetTrigger(LSID: Integer; var ATrigger: LongBool): Integer; |
| **C++:** | int GetTrigger (BOOL *pbATrigger); |
| **LabView:** |  |
| **Parameter:** | ATrigger: True => Trigger „Ein“<br>False => Trigger „Aus“ |
| **Beispiel:** | LS.GetTrigger(&ATrigger); |

| LS_SetTrigger | |
|---|---|
| **Beschreibung:** | Trigger Ein/Aus |
| **Delphi:** | function LS_SetTrigger(ATrigger: LongBool): Integer;<br>function LSX_SetTrigger(LSID: Integer; ATrigger: LongBool): Integer; |
| **C++:** | int SetTrigger (BOOL bATrigger); |
| **LabView:** |  |
| **Parameter:** | ATrigger: Trigger Ein/Aus |
| **Beispiel:** | LS.SetTrigger(true); |

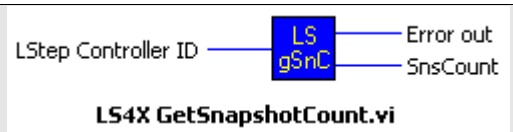
| LS_GetTriggerPar | |
|---|---|
| **Beschreibung:** | Liefert Trigger-Parameter |
| **Delphi:** | function LS_GetTriggerPar(var Axis, Mode, Signal: Integer; var Distance: Double): Integer;<br>function LSX_GetTriggerPar(LSID: Integer; var Axis, Mode, Signal: Integer; var Distance: Double): Integer; |
| **C++:** | int GetTriggerPar (int *plAxis, int *plMode, int *plSignal, double *pdDistance); |
| **LabView:** |  |
| **Parameter:** | Axis: Achse (1..4)<br>Mode: Trigger Modus (siehe Befehl !trigm)<br>Signal: Trigger Signal (siehe Befehl !trigs)<br>Distance: Trigger Distanz (siehe Befehl !trigd) |
| **Beispiel:** | LS.GetTriggerPar(&Axis, & Mode, & Signal, & Distance); |

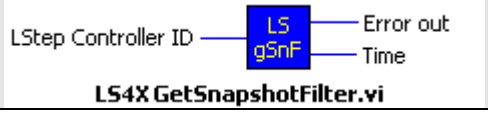
| LS_SetTriggerPar | |
|---|---|
| **Beschreibung:** | Trigger-Parameter |
| **Delphi:** | function LS_SetTriggerPar(Axis, Mode, Signal: Integer; Distance: Double): Integer;<br>function LSX_SetTriggerPar(LSID: Integer; Axis, Mode, Signal: Integer; Distance: Double): Integer; |
| **C++:** | int SetTriggerPar (int lAxis, int lMode, int lSignal, double dDistance); |
| **LabView:** |  |
| **Parameter:** | Axis: Achse (1..4)<br>Mode: Trigger Modus (siehe Befehl !trigm)<br>Signal: Trigger Signal (siehe Befehl !trigs)<br>Distance: Trigger Distanz (siehe Befehl !trigd) |
| **Beispiel:** | LS.SetTriggerPar(1, 3, 2, 5.0); |

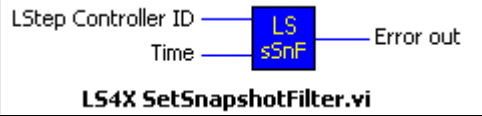
**Snapshot-Input**

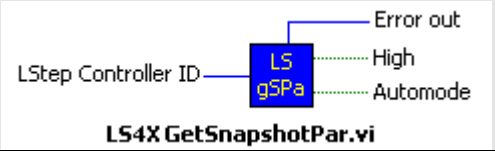
| LS_GetSnapshot | |
|---|---|
| **Beschreibung:** | Liefert den aktuellen Snapshot-Zustand |
| **Delphi:** | function LS_GetSnapshot(var ASnapshot: LongBool): Integer;<br>function LSX_GetSnapshot(LSID: Integer; var ASnapshot: LongBool): Integer; |
| **C++:** | int GetSnapshot (BOOL *pbASnapshot); |
| **LabView:** |  |
| **Parameter:** | ASnapshot: True => Snapshot „Ein“<br>False => Snapshot „Aus“ |
| **Beispiel:** | LS.GetSnapshot(&ASnapshot); |

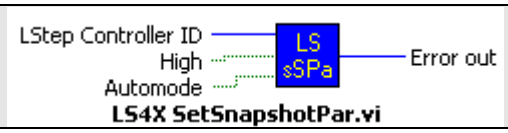
| LS_SetSnapshot | |
|---|---|
| **Beschreibung:** | Snapshot Ein/Aus |
| **Delphi:** | function LS_SetSnapshot(ASnapshot: LongBool): Integer;<br>function LSX_SetSnapshot(LSID: Integer; ASnapshot: LongBool): Integer; |
| **C++:** | int SetSnapshot (BOOL bASnapshot); |
| **LabView:** |  |
| **Parameter:** | ASnapshot: Snapshot Ein/Aus |
| **Beispiel:** | LS.SetSnapshot(true); |

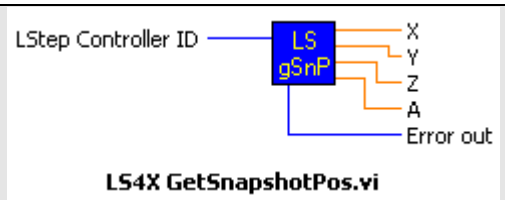
| LS_GetSnapshotCount | |
|---|---|
| **Beschreibung:** | Snapshot-Zähler |
| **Delphi:** | function LS_GetSnapshotCount(var SnsCount: Integer): Integer;<br>function LSX_GetSnapshotCount(LSID: Integer; var SnsCount: Integer): Integer; |
| **C++:** | int GetSnapshotCount (int *plSnsCount); |
| **LabView:** |  |
| **Parameter:** | SnsCount: Snapshot-Zähler |
| **Beispiel:** | LS.GetsnapshotCount(&SnsCount); |

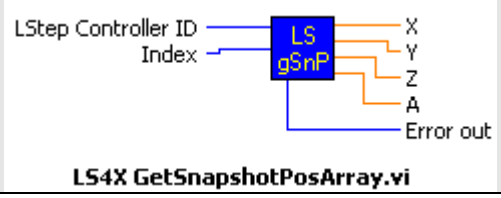
| LS_GetSnapshotFilter | |
|---|---|
| **Beschreibung** | Eingangsfilter auslesen |
| **Delphi:** | function LS_GetSnapshotFilter(var lTime: Integer): Integer;<br>function LSX_GetSnapshotFilter(LSID: Integer; var lTime: Integer): Integer; |
| **C++:** | int GetSnapshotFilter (int *plTime); |
| **LabView:** |  |
| **Parameter:** | lTime: Filterzeit [ms] |
| **Beispiel:** | LS. GetSnapshotFilter(&lTime); |

| LS_SetSnapshotFilter | |
|---|---|
| **Beschreibung** | Eingangsfilter bei prellenden Schaltern setzen |
| **Delphi:** | function LS_SetSnapshotFilter(lTime: Integer): Integer;<br>function LSX_ SetSnapshotFilter(LSID: Integer; lTime: Integer): Integer; |
| **C++:** | int SetSnapshotFilter (int lTime); |
| **LabView:** |  |
| **Parameter:** | lTime: Filterzeit, Wertebereich 0 – 100 ms |
| **Beispiel:** | LS. SetSnapshotFilter(0); // Kein Eingangsfilter |

| LS_GetSnapshotPar | |
|---|---|
| **Beschreibung** | Snapshot-Parameter auslesen |
| **Delphi:** | function LS_GetSnapshotPar(var High, AutoMode: LongBool): Integer;<br>function LSX_GetSnapshotPar(LSID: Integer; var High, AutoMode: LongBool): Integer; |
| **C++:** | int GetSnapshotPar (BOOL *pbHigh, BOOL *pbAutoMode); |
| **LabView:** |  |
| **Parameter:** | High: True => Snapshot ist High-Aktiv<br>False => Low-Aktiv<br>AutoMode: True => Snapshot „Automatik“. Die Position wird nach dem ersten Impuls automatisch angefahren. |
| **Beispiel:** | LS.GetSnapshotPar(&High, & AutoMode); |

| LS_SetSnapshotPar | |
|---|---|
| **Beschreibung** | Snapshot-Parameter |
| **Delphi:** | function LS_SetSnapshotPar(High, AutoMode: LongBool): Integer;<br>function LSX_SetSnapshotPar(LSID: Integer; High, AutoMode: LongBool): Integer; |
| **C++:** | int SetSnapshotPar (BOOL bHigh, BOOL bAutoMode); |
| **LabView:** |  |
| **Parameter:** | High: Snapshot High-Aktiv<br>AutoMode: Snapshot-Position automatisch anfahren |
| **Beispiel:** | LS.SetSnapshotPar(true, false); |

| LS_GetSnapshotPos | |
|---|---|
| **Beschreibung** | Snapshot-Position auslesen |
| **Delphi** | function LS_GetSnapshotPos(var X, Y, Z, A: Double): Integer;<br>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: Positionswerte |
| **Beispiel** | double X, Y, Z, A;<br>LS.GetSnapshotPos(&X, &Y, &Z, &A); |

| LS_GetSnapshotPosArray | |
|---|---|
| **Beschreibung** | Snapshot-Position aus Array auslesen |
| **Delphi** | function LS_GetSnapshotPosArray(Index: Integer; var X, Y, Z, R: Double): Integer;<br>function LSX_GetSnapshotPosArray(LSID: Integer; Index: Integer; var X, Y, Z, R: Double): Integer; |
| **C++** | int GetSnapshotPosArray (int lIndex, double *pdX, double *pdY, double *pdZ, double *pdR); |
| **LabView:** |  |
| **Parameter** | Index: Nummer der Snapshot-Position (1-200)<br>X, Y, Z, A: Positionswerte |
| **Beispiel** | double X, Y, Z, A;<br>LS.GetSnapshotPos(2, &X, &Y, &Z, &A); |

## 6.5 Fehlercodes

| LStep-Nr | API-Nr | Kommentar |
|---|---|---|
| 0 | 0 | Kein Fehler |
| | 4001,4002 | Interner Fehler |
| | 4003 | Undefinierter Fehler |
| | 4004 | Unbekannter Schnittstellentyp (kann bei Connect... auftreten) |
| | 4005 | Fehler beim Initialisieren der Schnittstelle |
| | 4006 | Keine Verbindung zur Steuerung (z.B. wenn SetPitch vor Connect aufgerufen wird) |
| | 4007 | Timeout während Lesen von der Schnittstelle |
| | 4008 | Fehler bei Befehlsübertragung an die LSTEP |
| | 4009 | Befehl wurde abgebrochen (mit SetAbortFlag) |
| | 4010 | Befehl wird von API nicht unterstützt |
| 11 | 4011 | Joystick-Hand eingeschaltet (kann bei SetJoystickOn/Off auftreten) |
| 11 | 4012 | Kein Verfahr-Befehl möglich, da Joystick-Hand |
| | 4013 | Regler-Timeout |
| 12 | 4015 | Endschalter in Verfahrrichtung betätigt |
| 14 | 4017 | Fehler beim Kalibrieren (Endschalter nicht korrekt freigefahren) |
| 1 | 4101 | Keine gültige Achsenbezeichnung |
| 2 | 4102 | Keine ausführbare Funktion |
| 3 | 4103 | Zu viele Zeichen in Befehls-String |
| 4 | 4104 | Kein gültiger Befehl |
| 5 | 4105 | Außerhalb des gültigen Zahlenbereiches |
| 6 | 4106 | Falsche Anzahl der Parameter |
| 7 | 4107 | Kein !oder ? |
| 8 | 4108 | Kein TVR möglich, da Achse aktiv |
| 9 | 4109 | Kein Ein- oder Ausschalten der Achsen, da TVR aktiv |
| 10 | 4110 | Funktion nicht konfiguriert |
| 11 | 4111 | Kein Move-Befehl möglich, da Joystick-Hand |
| 12 | 4112 | Endschalter betätigt |
| 13 | 4113 | Function kann nicht ausgeführt werden, da Encoder erkannt (clear pos) |
| 14 | 4114 | Fehler beim Kalibrieren (Endschalter nicht korrekt freigefahren) |
| 15 | 4115 | Wird beim Freifahren des Endschalters beim Kalibrieren oder Tischhubmessen der gegenüberliegende Endschalter aktiv, wird diese Funktion abgebrochen. |
| 20 | 4120 | Treiberrelaise defekt (Sicherheitskreis K3/K4) |
| 21 | 4121 | Es dürfen nur einzelne Vektoren verfahren werden (Einrichtbetrieb) |
| 22 | 4122 | Es darf kein Kalibrieren, Tischhubmessen oder Joystick-Betrieb durchgeführt werden (Tür offen oder Einrichtbetrieb) |
| 23 | 4123 | SECURITY Error X-Achse |
| 24 | 4124 | SECURITY Error Y-Achse |
| 25 | 4125 | SECURITY Error Z-Achse |
| 26 | 4126 | SECURITY Error A-Achse |
| 27 | 4127 | Stop aktiv |
| 28 | 4128 | Fehler im Türschaltersicherheitskreis (nur bei LS44/Solero) |
| 29 | 4129 | Endstufen nicht eingeschaltet (nur bei LS44) |
| 30 | 4130 | GAL Sicherheitsfehler (nur bei LS44) |
| 31 | 4131 | Joystick lässt sich nicht einschalten, da Move aktiv |

## 6.6 Häufige Fragen & Antworten

| |
|---|
| Wie werden die LSTEP4.DLL bzw. die LSTEP4X.DLL in einem MS Visual C++ Projekt eingebunden? |
| Wie initialisiere ich mit dem LStep API die Verbindung zur LStep?<br>Welcher der Connect-Befehle sollte verwendet werden? |
| Wie installiere ich den Treiber für die LStep-PCI? |
| Warum bekommt mein Programm mit der LSTEP4.DLL keine Verbindung zur LStep-PCI? |
| Im Ablauf, in der LSTEP4.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? |
| Wann sollte die LSTEP4.DLL, wann die LSTEP4X.DLL verwendet werden? |
| Ist das LSTEP-API kompatibel zur MCL bzw. zum alten Register-Befehlssatz? |
| Warum bekomme ich in MS Visual C++ bei Einbindung von LStep4.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 des LSTEP-API die Meldung „First chance exception“, „Exception: Timeout read RS232!“ o.ä.? |
| Wie kann ich mit dem 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 des LStep-APIs? |

**Wie werden die LSTEP4.DLL bzw. die LSTEP4X.DLL in einem MS Visual C++ Projekt eingebunden?**

- Projekt erzeugen
- LSTEP4.DLL, LSTEP4.h, LSTEP4.cpp in Projektordner kopieren
- LSTEP4.h und LSTEP4.cpp in Projekt einfügen
- Menü: Projekt\ Einstellungen\ C/C++ Option: [vorkompilierte Header nicht verwenden] wählen
- in LSTEP4.h #include „stdafx.h“ einbinden
- in Projektname_Dlg.h #include „LSTEP4.h“
- die erforderliche Instanz in public einbinden
  Beispiel: CLStep* MyLStep = new CLStep();

Die Einbindung der LSTEP4X.DLL erfolgt analog zu dieser Vorgehensweise.

**Wie initialisiere ich mit dem LStep API die Verbindung zur LStep?**
**Welcher der Connect-Befehle sollte verwendet werden?**
Die Verbindung zur LSTEP-API wird mit einem der Connect-Befehle (Connect, ConnectEx, ConnectSimple) initialisiert. Abgesehen von einigen Sonderfällen sollte immer **ConnectSimple** verwendet werden.

| | |
|---|---|
| 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 des 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 LSTEP4.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 LSTEP4.DLL, ihres Programms oder einem Windows-Systemordner liegen.** Sie finden diese Datei im Unterordner „LStepPCI" des LStep API. *(siehe auch Kapitel 9.7).*

**Im Ablauf, in der LSTEP4.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 des 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?**
Das LSTEP-API verarbeitet beim Warten auf Rückmeldungen von Verfahrbefehlen Botschaften, damit das Programm nicht „steht", denn ansonsten wäre es nicht möglich, in Fehlersituationen einen Abbruch durchzuführen bzw. die Achsen zu stoppen. SetProcessMessagesProc ermöglicht das Ersetzen der internen Message-Dispatching Prozedur des LSTEP-API. Das 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.

**Wann sind Moves mit bzw. ohne Wait zu verwenden?**
Move-Befehle mit WaitForAxisStop sind zu verwenden, wenn alle Achsen synchronund 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 dem LSTEP-API einzelne Achsen der LStep unabhängig voneinander verfahren?**
Die Verfahr-Befehle des LSTEP-API bieten zwei verschiedene Möglichkeiten: Wird der (letzte) Parameter Wait=true 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.
Indem man also zunächst MoveAbsSingleAxis mit Wait=false für die X-Achse verwendet und etwas später MoveAbsSingleAxis mit Wait=false für die Y-Achse aufruft, können Achsen separat verfahren werden. Um festzustellen, ob Achsen ihre Zielposition erreicht haben, kann man den Befehl **WaitForAxisStop** benutzen.

Beispiel:

```
LS.MoveAbsSingleAxis(Xaxis, 10, false);   // Verfahre die X-Achse asynchron
Delay(1000);                              // Warte 1s bis zum Start der Y-Achse
LS.MoveAbsSingleAxis(Yaxis, 20, false);   // Verfahre die Y-Achse asynchron
LS.WaitForAxisStop(3, 0, flag);           // Warten 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:**

```
LS.MoveAbsSingleAxis(Xaxis, 10, false);   // Verfahre die X-Achse asynchron
LS.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 LS_ConnectSimple(4, nil, 0, true) eine Verbindung zur LStep-PCI im ersten PCI-Slot des Mainboards, durch LS_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.

**Wann sollte die LSTEP4.DLL, wann die LSTEP4X.DLL verwendet werden?**
Wenn mehrere LSteps/LStep-PCI-Karten von einem PC aus gesteuert werden sollen, sollte die LSTEP4X.DLL eingesetzt werden, ansonsten ist die LSTEP4.DLL geeignet.

**Ist das LSTEP-API kompatibel zur MCL bzw. zum alten Register-Befehlssatz?**
Das LSTEP-API ist prinzipiell abwärtskompatibel zu dem Register-Befehlssatz, mit dem die MCL und ältere LSteps kommunizieren. Jedoch bietet dieser Befehlssatz viele Möglichkeiten nicht, die das 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 LStep4.cpp die Meldung "fatal error C1010"?**
Es handelt sich hierbei nicht um einen Fehler in der Datei LStep4.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 LStep4.cpp abgeschaltet werden. Sofern Sie die MFC in Ihrem Projekt nicht verwenden, sollten Sie die Zeile #include "stdafx.h" aus LStep4.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 des LSTEP-API die Meldung „First chance exception", „Exception: Timeout read RS232!" o.ä.?**
Interne Exceptions der LSTEP4.DLL, die nur im Debugger sichtbar sind haben keine Bedeutung. Sie dienen zur internen Ablaufsteuerung. Bei ConnectSimple tritt häufig eine Exception auf, da das LSTEP-API versucht, den Befehlssatz herauszufinden. Dabei kommt es zu einem Timeout, wenn die Steuerung den getesteten Befehlssatz nicht unterstützt. In Delphi können Sie in den Debugger-Optionen die entsprechenden Exception zu den von Debugger zu ignorierenden Exceptions hinzufügen.

**Wie kann ich mit dem 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 die Achse mit einem sehr langen Vektor starten
**MoveRelSingleAxis**(Xaxis, 100000, false)
Wichtig ist, den Parameter Wait=false zu setzen
Bei Loslassen der Taste den Befehl **StopAxes** aufrufen

**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, so werden die ältesten überschrieben.

**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.

## 6.7 API- und ASCII Befehle / Index 2

**Befehlskennung letzte Spalte (X):** 1 = LSTEP; 2 = LSTEPexpress; 3 = beide Steuerungen
(Diese Kennung gibt an, welche Befehle von der jeweiligen Steuerung unterstützt werden)
Die Beschreibung der Befehle für die es keinen DLL-Aufruf gibt, finden sie im Kapitel 4.
(diese müssen mit dem DLL-Aufruf „SendString“ gesendet werden)

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| Connect | Mit LSTEP verbinden | - | 3 |
| ConnectEx | Mit LSTEP verbinden | - | 3 |
| ConnectSimple | Mit LSTEP verbinden | - | 3 |
| CreateLSID | Erzeugt eine ID Nr bei der Verwendung des LSTEP4X APIs | - | 3 |
| Disconnect | Verbindung zu LSTEP trennen | - | 3 |
| EnableCommandRetry | Mit dieser Funktion kann das wiederholte Senden von Kommandos im Falle von Fehlern ein-/ausgeschaltet werden | - | 3 |
| FlushBuffer | Löscht den Eingabepuffer | - | 3 |
| FreeLSID | Gibt die erzeugte ID Nr wieder frei | - | 3 |
| LoadConfig | LSTEP-Konfiguration (Schnittstelle, Achseinstellungen, Regler) aus INI-Datei laden | - | 3 |
| SaveConfig | LSTEP-Konfiguration (Schnittstelle, Achseinstellungen, Regler) in INI-Datei speichern. | - | 1 |
| SendString | String an LSTEP senden | - | 3 |
| SendStringPosCmd | Verfahrbefehl, welcher Rückmeldung erwartet, als String an LSTEP senden | - | 3 |
| SetAbortFlag | Flag setzen, damit die Kommunikation mit der LSTEP abgebrochen wird | - | 3 |
| SetCommandTimeout | Setzt die Timeoutzeiten für das Warten auf der Rückmeldung beim Positionieren und Kalibrieren | | |
| SetControlPars | Überträgt die mit LS_LoadConfig geladenen Parameter an die LSTEP | - | 3 |
| SetCorrTblOff | Achsenkorrektur deaktivieren | - | 3 |
| SetCorrTblOn | Achsenkorrektur in x/y-Matrix mit linearer Interpolation aktivieren | - | 3 |
| SetExtValue | Erweiterungen einschalten | - | 3 |
| SetFactorMode | Positionswert-Umrechnung für ‚krumme’ Spindelsteigungen | - | 1 |
| SetLanguage | Sprachumschaltung LSTEP-API (Protokoll/Meldungen) | - | 3 |
| SetProcessMessagesProc | Ermöglicht das Ersetzen der internen Message-Dispatching Prozedur des LStep API | - | 3 |
| SetShowCmdList | LStep-API Befehlsliste Ein/Aus | - | 3 |
| SetShowProt | Schnittstellen-Protokoll Ein/Aus | - | 3 |
| SetWriteLogText | Schreiben der Protokoll-Datei LSTEP4.log ein-/ausschalten (Standardmäßig ist das Schreiben in LSTEP4.log ausgeschaltet) | - | 3 |
| SetWriteLogTextFN | Schreiben des Schnittstellen-Protokolls in eine bestimmte Datei ein-/ausschalten | - | 3 |
| | Baudrate einstellen | (?) !baud | 3 |

**Steuerungs-Info:**

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| GetSerialNr | Seriennummer der Steuerung auslesen | ?readsn | 3 |
| GetVersionStr | liefert die aktuelle Versionsnummer der Firmware zurück | ?ver | 3 |
| GetVersionStrDet | Detaillierte Versionsnummer der Firmware auslesen | ?det | 3 |
| GetVersionStrInfo | Ergänzung zur aktuellen Versionsnummer auslesen | ?iver | 3 |

**Einstellungen**

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| ConfigMaxAxes | Anzahl der verwendeten Achsen | (?) !configmaxaxis | 3 |
| GetAccel | Beschleunigung abfragen | ?accel | 3 |
| GetAccelJerk | Ruck Beschleunigung abfragen | ?acceljerk | 2 |
| GetActiveAxes | Liefert die Achsenfreigaben | ?axis | 3 |
| GetAxisDirection | Drehrichtungs-Umkehr abfragen | ?axisdir | 3 |
| GetCalibBackSpeed | Liefert die Geschwindigkeit, mit der aus den Endschaltern gefahren wird | ?calbspeed | 1 |
| GetCaliboffset | Kalibrier-Offset abfragen | ?caliboffset | 3 |
| GetCalibrateDir | Liefert Vorzeichen-Umkehr bei Kalibration | ?caldir | 3 |
| GetCalibRMAccel | Beschleunigung für Kalibrieren und Hubmessen abfragen | ?calibrmaccel | 2 |
| GetCalibRMBackSpeed | Geschwindigkeit mit der aus dem Endschalter beim Kalibrieren und Hubmessen gefahren wird. | ?calibrmbspeed | 2 |
| GetCalibRMJerk | Ruck Beschleunigung für Kalibrieren und Hubmessen abfragen | ?calibrmjerk | 2 |
| GetCalibRMVel | Geschwindigkeit für Kalibrieren und Hubmessen abfragen | ?calibrmvel | 2 |
| GetCurrentDelay | Gibt an Zeitverzögerung für die Stromabsenkung | ?curdelay | 3 |
| GetDeceleration | Verzögerung abfragen | ?decel | 2 |
| GetDeceljerk | Ruck Verzögerung abfragen | ?deceljerk | 2 |
| GetDimensions | Abfrage der  Dimensionen der Achsen | ?dim | 3 |
| GetGear | Getriebefaktor abfragen | ?gear | 1 |
| GetGearDenominator | Getriebefaktor Nenner | ?geardenominator | 2 |
| GetGearNumerator | Getriebefaktor Zähler | ?gearnumerator | 2 |
| GetJoystickFilter | Gibt an, ob Filterung im Joystick-Betrieb aktiv ist | ?joyfilter | 1 |
| GetMotorCurrent | Motorstrom abfragen | ?cur | 1 |
| GetMotorFieldDir | Motordrehrichtung positiv oder negativ | ?motorfielddir | 2 |
| GetMotorMaxVel | max. einstellbare geschwindigkeit für den Motor | ?motormaxvel | 2 |
| GetMotorTablePatch | Gibt an, ob die Korrekturtabelle aktiviert ist | ?mtpatch | 1 |
| GetMotorType | eingestellten Motortype abfragen | ?motortype | 2 |
| GetOutFuncLev | Liefert die StromUmschaltungsgeschwindigkeit | ?opfl | 1 |
| GetPitch | Liefert die Spindelsteigungen | ?pitch | 3 |
| GetPowerAmplifier | Endstufen Ein/Aus (nur bei LS44 + Express) | ?pa | 2 |
| GetReduction | Stromabsenkung abfragen | ?reduction | 3 |
| GetRefSpeed | Geschwindigkeit, mit der beim Kalibrieren die Referenzmarke gesucht wird, abfragen. | ?calrefspeed | 1 |
| GetRMOffset | RM-Offset abfragen | ?rmoffset | 3 |
| GetSpeedPoti | Gibt an, ob Speed-Poti Ein oder Aus ist. | ?pot | 3 |
| GetStopAccel | Liefert die Bremsbeschleunigung, wenn der | ?stopaccel | 1 |

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| | Stopeingang aktiv ist | | |
| GetStopDecel | Bremsbeschleunigung bei Stop aktiv | ?stopdecel | 2 |
| GetStopDecelJerk | Ruck für Bremsbeschleunigung bei Stop aktiv | ?stopdeceljerk | 2 |
| GetStopPolarity | Stopeingang Polarität lesen | ?stoppol | 3 |
| GetVel | Geschwindigkeit aller Achsen abfragen | ?vel | 3 |
| GetVelFac | Geschwindigkeitsuntersetzung abfragen | ?velfac | 1 |
| GetVLevel | Liefert die Ausgeblendete Geschwindigkeit | ?vlevel | 1 |
| GetXYAxisComp | Abfrage der XY-Achsüberlagerung | ?xycomp | 1 |
| LstepSave | Aktuelle Konfiguration in LStep speichern (EEPROM) | save | 3 |
| SetAccel | Beschleunigung einstellen | !accel | 3 |
| SetAccelJerk | Ruck für Beschleunigung einstellen | !acceljerk | 2 |
| SetAccelSingleAxis | Beschleunigung für eine einstellen | !accel x (y,z,a) | 3 |
| SetActiveAxes | Achsenfreigabe | !axis | 3 |
| SetAxisDirection | Drehrichtungs-Umkehr | !axisdir | 3 |
| SetCalibBackSpeed | Geschwindigkeit, mit der aus den Endschalter gefahren wird, setzen | !calbspeed | 1 |
| SetCaliboffset | Kalibrier-Offset | !caliboffset | 3 |
| SetCalibrateDir | Vorzeichen-Umkehr bei Kalibration | !caldir | 3 |
| SetCalibRMAccel | Beschleunigung für Kalibrieren und Hubmessen einstellen | !calibrmaccel | 2 |
| SetCalibRMBackSpeed | Geschwindigkeit mit der aus dem Endschalter beim Kalibrieren und Hubmessen gefahren wird. | !calibrmbspeed | 2 |
| SetCalibRMJerk | Ruck für Kalibrieren und Hubmessen einstellen | !calibrmjerk | 2 |
| SetCalibRMVel | Geschwindigkeit für Kalibrieren und Hubmessen einstellen | !calibrmvel | 2 |
| SetCurrentDelay | Zeitverzögerung für die Stromabsenkung | !curdelay | 3 |
| SetDeceleration | Verzögerung einstellen | !decel | 2 |
| SetDecelJerk | Ruck für Verzögerung einstellen | !deceljerk | 2 |
| SetDecelSingleAxis | Verzögerung für eine Achse einstellen | !decel x | 2 |
| SetDimensions | Dimensionen der Achsen einstellen | !dim | 3 |
| SetGear | Getriebefaktor einstellen | !gear | 1 |
| SetGearDenominator | Getriebefaktor Nenner | !geardenominator | 2 |
| SetGearNumerator | Getriebefaktor Zähler | !gearnumerator | 2 |
| SetJoystickFilter | Filterung im Joystick-Betrieb Ein/Aus | !joyfilter | 1 |
| SetMotorCurrent | Motorstrom einstellen | !cur | 1 |
| SetMotorFieldDir | Motordrehrichtung positiv oder negativ | !motorfielddir | 2 |
| SetMotorMaxVel | max. einstellbare geschwindigkeit für den Motor | !motormaxvel | 2 |
| SetMotorTablePatch | Korrekturtabelle Ein/Aus | !mtpatch | 1 |
| SetMotorType | Motortype einstellen | !motortype | 2 |
| SetOutFuncLev | Setzt die StromUmschaltungsgeschwindigkeit | !opfl | 1 |
| SetPitch | Spindelsteigung setzen | !pitch | 3 |
| SetPowerAmplifier | Schaltet bei LS44 + ExpressEndstufen Ein/Aus | !pa | 2 |
| SetReduction | Stromabsenkung einstellen | !reduction | 3 |
| SetRefSpeed | Geschwindigkeit, mit der beim Kalibrieren die Referenzmarke gesucht wird, setzen | !calrefspeed | 1 |
| SetRMOffset | RM-Offset | !rmoffset | 3 |
| SetSpeedPoti | Speed-Poti Ein/ Aus | !pot | 3 |
| SetStopAccel | Stellt die Bremsbeschleunigung ein, wenn der Stopeingang aktiv ist | !stopaccel | 1 |
| SetStopDecel | Bremsbeschleunigung bei Stop aktiv | !stopdecel | 2 |

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| SetStopDecelJerk | Ruck für Bremsbeschleunigung bei Stop aktiv | !stopdeceljerk | 2 |
| SetStopPolarity | Stopeingang Polarität einstellen | !stoppol | 3 |
| SetVel | Geschwindigkeit aller Achsen einstellen | !vel | 3 |
| SetVelFac | Geschwindigkeitsuntersetzung setzen | !velfac | 1 |
| SetVelSingleAxis | Geschwindigkeit für eine Achse einstellen | !vel x (y,z,a) | 3 |
| SetVLevel | Ausblenden von Geschwindigkeiten, bei denen Resonanzen auftreten | !vlevel | 1 |
| SetXYAxisComp | XY-Achsüberlagerung aktivieren | !xycomp | 1 |
| SoftwareReset | Software wird in den Startzustand versetzt | reset | 3 |
| | Aktivierung der Regler Parameter | !validpar | 2 |

**Statusabfragen**

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| GetError | liefert die aktuelle Fehlernummer | ?err | 3 |
| | | | |
| | | | |
| | | | |
| GetSecurityErr | Liest alle Zustände und Ergebnisse der GAL-Sicherheitsüberwachung (nur bei LS44-Steuerungen) | ?securityerror | 1 |
| GetSecurityStatus | Liest den aktuellen Zustand der Sicherheitsüberwachung | ?securitystatus | 1 |
| GetStatus | liefert den aktuellen Zustand der Steuerung | ?status | 3 |
| | Liefert den Systemstatus der Achsen als Nummer | ?sysstat | 2 |
| | Liefert den Systemstatus der Achsen als Text | ?sysstatus | 2 |
| GetStatusAxis | liefert den aktuellen Zustand der einzelnen Achsen | ?statusaxis | 3 |
| GetStatusLimit | Liefert den aktuellen Zustand der Software-Grenzen jeder einzelnen Achse | ?statuslimit | 3 |
| SetAutoStatus | AutoStatus Ein/Aus | !autostatus | 3 |

**Fahrbefehle und Positionsverwaltung**

| APIBefehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| Calibrate | Kalibrieren | !cal | 3 |
| CalibrateEx | Es werden nur die Achsen kalibriert, deren entsprechendes Bit in dem übergebenen Integer-Wert gesetzt ist. | !cal x (xy,z,a) | 3 |
| Clearpos | Positionswerte werden genullt (für endlos Drehachsen) | !clearpos | 1 |
| GetDelay | Liefert die Verzögerung des Vektorstarts | ?delay | 3 |
| GetDistance | Liefert die Strecke, die mit LS_PosRelShort gestartet wird | ?distance | 3 |
| GetInputTrigMove | Abfrage des Eingangs zum externen Vektorstart | ?itm | 1 |
| GetPos | Abfrage der aktuellen Position aller Achsen | ?pos | 3 |
| GetPosEx | Abfrage der aktuellen Geber- bzw. Positionswerte aller Achsen | | |
| GetPosSingleAxis | Abfrage der aktuellen Position einer Achse | ?pos x (y,z,a) | 3 |
| MoveAbs | Absolutposition anfahren | !moa | 3 |
| MoveAbsSingleAxis | Absolutposition einer Achse anfahren | !moa x (y,z,a) | 3 |

| APIBefehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| MoveEx | Erweiterter Verfahr-Befehl | | |
| MoveRel | Relativen Vektor fahren | !mor | 3 |
| MoveRelShort | Positionieren Relativ (short command) | m | 3 |
| MoveRelSingleAxis | Relativen Vektor einer Achse verfahren | !mor x (y,z,a) | 3 |
| RMeasure | Tischhub messen | !rm | 3 |
| RmeasureEx | Tischhub messen wird nur bei den Achsen durchgeführt, deren entsprechendes Bit in dem übergebenen Integer-Wert gesetzt ist | !rm x (xy z) | 3 |
| SetDelay | Durch den Befehl Delay kann eine Verzögerung des Vektorstarts erzeugt werden | !delay | 3 |
| SetDistance | Strecke setzen (für MoveRelShort) | !distance | 3 |
| SetInputTrigMove | Eingang setzen für externen Vektorstart | !itm | 1 |
| SetPos | Position setzen | !pos | 3 |
| StopAxes | alle Verfahrbewegungen werden abgebrochen | a | 3 |
| WaitForAxisStop | Die Funktion kehrt zurück, sobald die in der Bit-Maske Aflags gewählten Achsen ihre Zielposition erreicht haben | - | |
| | setzen und lesen einer Tabellenposition [Table Pos] | (?)!tpos | 2 |
| | Tabellenposition anfahren [Move Table Pos Absolut] | !mtpa | 2 |
| | automatisches erstellen einer Tabelle mit gleichen Abschnitten (z.B. für eine Umdrehung einer Drehachse) [Index Table Divider] | (?)!itd | 2 |
| | Anfahren des angegebenen Index [Move Index Table] | !mita | 2 |

**Joystick und Handrad**

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| GetDigJoySpeed | Digitaler Joystick und Geschwindigkeit lesen | ?speed | 3 |
| GetHandwheel | Liest den Zustand des Handrads | ?hw | 1 |
| GetJoystick | Liest den Zustand des Analog-Joysticks | ?joy | 3 |
| GetJoystickAxes | Freigabe für Joystick | ?joyenable | 2 |
| GetJoystickDir | Richtung Joystick | ?joydir | 3 |
| GetJoystickWindow | Joystick-Fenster ablesen | ?joywindow | 3 |
| GetJoyChangeAxis | Liest Joystickachszuordnung | ?joychangeaxis | 1 |
| | Filterzeitkonstante für Eingangssignale abfragen | ?joyoutpass | 2 |
| GetJoyVel | Max. Geschwindigkeit für Joystick abfragen | ?joyvel | 2 |
| SetDigJoySpeed | Digitaler Joystick und Geschwindigkeit setzen | !speed | 3 |
| SetDigJoyOff | Digitalen Joystick ausschalten | !speed 0 | 3 |
| SetHandwheelOff | Handrad Aus | !hw 0 | 1 |
| SetHandwheelOn | Handrad Ein | !hw 1 (1-4) | 1 |
| SetJoystickAxes | Freigabe für Joystick | !joyenable | 2 |
| SetJoystickDir | Richtung Joystick | !joydir | 3 |
| SetJoystickOff | Analog-Joystick Aus | !joy 0 | 3 |
| SetJoystickOn | Analog-Joystick Ein | !joy 1 (1-4) | 3 |
| SetJoystickWindow | Joystick-Fenster setzen | !joywindow | 3 |
| | Filterzeitkonstante für Eingangssignale einstellen | !joyoutpass | 2 |
| SetJoyVel | Max. Geschwindigkeit für Joystick setzen | !joyvel | 2 |
| JoyChangeAxis | Setzt Joystickachszuordnung | !joychangeaxis | 1 |

**Bedienpult mit Trackball und Joyspeed-Tasten (nur für LSTEP-xx/2)**

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | |
|---|---|---|---|
| GetBPZ | Liest Zustand des Bedienpults | ?bpz | 1 |
| GetBPZJoyspeed | Liest Bedienpult Joystick-Speed | ?joyspeed | 1 |
| GetBPZTrackballBackLash | Liest Bedienpult Trackball-Umkehrspiel | ?bpzbl | 1 |
| GetBPZTrackballFactor | Liest Bedienpult Trackball-Faktor | ?bpztf | 1 |
| SetBPZ | Bedienpult Ein/Aus | !bpz | 1 |
| SetBPZJoyspeed | Bedienpult Joystick-Speed | !joyspeed | 1 |
| SetBPZTrackballBackLash | Bedienpult Trackball-Umkehrspiel | !bpzbl | 1 |
| SetBPZTrackballFactor | Bedienpult Trackball-Faktor | !bpztf | 1 |

**Manuelle Bedienung über Trackball oder Tippbetrieb**
**(nur für LSTEP-PCIexpress / LSTEPexpress)**

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | |
|---|---|---|---|
| | Tippbetrieb Ein- oder Ausschalten | tipp | 2 |
| | Richtungsvorgabe der Tasten für Tippbetrieb | tippdir | 2 |
| | Freigabe für Tippbetrieb | tippenable | 2 |
| | Stromreduzierung für Tippbetrieb | tippredcur | 2 |
| | Geschwindigkeit für Tippbetrieb | tippvel | 2 |
| | Filterzeitkonstante für Eingangssignale | tippoutpass | 2 |
| | Trackball Ein- oder Auschalten | tb | 2 |
| | Richtungsvorgabe für den Trackball | tbdir | 2 |
| | Freigabe für den Trackball | tbenable | 2 |
| | Stromreduzierung für den Trackball | tbredcur | 2 |
| | Zuordnung der Trackballachsen | tbtoaxis | 2 |
| | Geschwindigkeit für den Trackball | tbvel | 2 |
| | Filterzeitkonstante für Eingangssignale | tboutpass | 2 |

**Endschalter (Hardware u. Software)**

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | |
|---|---|---|---|
| GetAutoLimitAfterCalibRM | Gibt an, ob beim Kalibrieren und Tischhubmessendie interne Software-Limits gesetzt werden. | ?nosetlimit | 3 |
| GetLimit | Liefert Verfahrbereichsgrenzen | ?lim | 3 |
| GetLimitControl | Liest,ob die Bereichsüberwachung eingeschaltet ist | ?limctr | 3 |
| GetLimitControlMode | Liefert den Modus für die Überwachung der Softwarelimmits | ?limmode | 1 |
| GetSwChange | Abfrage ob Endschalter getauscht wurden | ?swchange | 2 |
| GetSwitchActive | Gibt an, ob die Endschalter eingeschaltet sind | ?swact | 3 |
| GetSwitches | liest den Zustand aller Endschalter | ?readsw | 3 |
| GetSwitchPolarity | Liest Endschalterpolarität | ?swpol | 3 |
| SetAutoLimitAfterCalibRM | Verhindert, dass beim Kalibrieren und Tischhubmessen die internen Software-Limits gesetzt werden. | !nosetlimit | 3 |
| SetLimit | Verfahrbereichsgrenzen einstellen | !lim | 3 |
| SetLimitControl | Bereichsüberwachung | !limctr | 3 |
| SetLimitControlMode | Setzt den Modus für die Überwachung der | !limmode | 1 |

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | |
|---|---|---|---|
| | Softwarelimits | | |
| SetSwChange | Null- Und Endlagen-Endschalter tauschen | !swchange | 2 |
| SetSwitchActive | Liest Status für Endschalter Ein / Aus | !swact | 3 |
| SetSwitchPolarity | Endschalterpolarität einstellen | !swpol | 3 |

**Digitale und analoge Ein.- und Ausgänge**

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| GetAnalogInput | Lesen des aktuellen Zustands eines Analogkanals | ?anain | 3 |
| GetAnalogInputs2 | Lesen der aktuellen Zustände der Analogkanäle PT100, MV und V24 | -- | |
| GetDigitalInputs | Alle Inputpins lesen | ?digin | 3 |
| GetDigitalInputsE | Zusätzliche digitale Eingänge lesen (16-31) | ?edigin | 1 |
| SetAnalogOutput | Analogkanal setzen | !anaout | 3 |
| SetDigIO_Distance | Aktivierung eines Ausgang in Abhängigkeit der eingestellten Strecke vor/nach der Zielposition | (digfkt) | 1 |
| SetDigIO_EmergencyStop | Funktion der digitalen Ein-/Ausgänge Zuordnung des Not-Stop-Pins | (digfkt) | 1 |
| SetDigIO_Off | Funktion der digitalen Ein-/Ausgänge Aus | (digfkt) | 1 |
| SetDigIO_Polarity | Einstellung der Polarität | !digfkt 16 0 0 | 1 |
| SetDigitalOutput | Digitalen Ausgang setzen | !digout x | 3 |
| SetDigitalOutputs | Digitale Ausgänge setzen (0-15) | !digout 0-15 | 3 |
| SetDigitalOutputsE | Zusätzliche digitale Ausgänge setzen (16-31) | !edigout | 1 |

**Takt-Vor/Rück Eingänge**

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| GetFactorTVR | Liest den Faktor Takt Vor / Rück | ?tvrf | 3 |
| GetTVRMode | Einstellung vom Takt Vor / Rück auslesen | ?tvr | 3 |
| | Takt- Vor/Rück Modus | ?tvrm | |
| SetFactorTVR | Faktor Takt Vor / Rück | !tvrf | 3 |
| SetTVRMode | Takt Vor / Rück einstellen | !tvr (0-4) | 3 |
| | Takt- Vor/Rück Modus | !tvrm | |

**Takt-Vor/Rück über Schnittstelle**

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| SetTVRInPulse | Takt-Vor/Rück über Schnittstelle | px/nx | 1 |

**Takt-Vor/Rück Ausgänge für weitere Achsen**

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| GetAccelTVRO | Alle eingestellten Beschleunigungen lesen | ?tvroa | 1 |
| GetPosTVRO | Liefert Positionswerte, in Abhängigkeit von Dimension | ?tvropos | 1 |
| GetStatusTVRO | Liefert den aktuellen Status der Achsen | ?tvrostatus | 1 |
| GetTVROutMode | Einstellung vom Takt Vor/Rück lesen | ?tvrout | 1 |
| GetTVROutPitch | Liest Spindelsteigung | ?tvropitch | 1 |
| GetTVROutResolution | Liefert dieAuflösung der an zu steuernden Endstufe | ?tvrores | 1 |
| GetVelTVRO | Alle eingestellten Geschwindigkeiten lesen | ?tvrov | 1 |
| MoveAbsTVROSingleAxis | Absolutposition einer Achse anfahren | !tvromoa x | 1 |
| MoveAbsTVRO | Absolutposition anfahren | !tvromoa | 1 |
| MoveRelTVROSingleAxis | Relativen Vector einer Achse verfahren | !tvromor x | 1 |
| MoveRelTVRO | Relativen Vector  verfahren | !tvromor | 1 |
| SetAccelSingleAxisTVRO | Beschleunigung einer einzelnen Achse setzen | !tvroa x | 1 |
| SetAccelTVRO | Beschleunigungen setzen | !tvroa | 1 |
| SetPosTVRO | Position setzen | !tvropos | 1 |
| SetTVROutMode | Takt Vor / Rück einstellen | !tvrout | 1 |
| SetTVROutPitch | Spindelsteigung setzen | !tvropitch | 1 |
| SetTVROutResolution | Auflösung der an zu steuernden Endstufe | !tvrores | 1 |
| SetVelSingleAxisTVRO | Geschwindigkeit einer einzelnen Achse setzen | !tvrov x | 1 |
| SetVelTVRO | Geschwindigkeiten setzen | !tvrov | 1 |

**Geber-Einstellungen**

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| ClearEncoder | Geberposition auf null setzen | !pos 0 0 0 0 | 3 |
| GetEncoder | Liest alle Geberpositionen | !encpos1<br>?pos | 3 |
| GetEncoderActive | Liest, welche Geber nach der Kalibration aktiv werden | ?encmask | 3 |
| GetEncoderMask | Geberzustände auslesen | ?enc | 3 |
| GetEncoderPeriod | Geberperiodenlängen auslesen | ?encperiod | 3 |
| GetEncoderPosition | Liefert Einstellung von Geberwertanzeige | ?pos<br>(nach !encpos 1) | 3 |
| GetEncoderRefSignal | Gibt an, ob beim Kalibrieren Referenzsignal von Geber ausgewertet werden soll | ?encref | 3 |
| SetEncoderActive | Mit dieser Funktion kann ausgewählt werden, welche Geber nach der Kalibration aktiviert werden sollen | !encmask | 3 |
| SetEncoderMask | Geber aktivieren oder deaktivieren | !enc | |
| SetEncoderPeriod | Geberperiodenlängen einstellen | !encperiod | 3 |
| SetEncoderPosition | Geberwertanzeige Ein/Aus | !encpos 1<br>!pos 0 0 0 | 3 |
| SetEncoderRefSignal | Beim Kalibrieren Referenzsignal von Geber auswerten | !encref | 3 |
| | Encoder Fehler | (?) !encerr | 1 |
| | Zählrichtung des Encoders ändern | (?) !encdir | 2 |
| | Encodertype einstellen | (?) !enctyp | 2 |
| | Anzahl der Encoderperioden pro Motorumdrehung | (?) !encpolepairs | 2 |
| | Zuordnung von gebereingang und Achse | (?) !enctoaxis | 2 |

| | | | |
|---|---|---|---|
| | Polarität der Referenzmarken der QEP-Encoder | (?) !encrefpol | 2 |

**Reglereinstellungen**

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| ClearCtrFastMoveCounter | Anzahl ausgeführter FastMove Funktionen auf 0 setzen | !ctrfmc 0 | 1 |
| GetController | Regler-Modus auslesen | ?ctr | 1 |
| GetControllerCall | Einstellung vom Regleraufruf auslesen | ?ctrc | 1 |
| GetControllerFactor | Einstellung vom Reglerfaktor auslesen | ?ctrf | 1 |
| GetControllerSteps | Regler-Schritte auslesen | ?ctrs | 1 |
| GetControllerTimeout | Liefert die Einstellung vom Regler-Überwachungs-Timeout | ?ctrt | 1 |
| GetControllerTWDelay | Reglerverzögerung auslesen | ?ctrd | 1 |
| GetCtrFastMove | Einstellung von Fast Move Funktion lesen | ?ctrfm | 1 |
| GetCtrFastMoveCounter | Anzahl ausgeführter FastMove Funktionen auf 0 auslesen | ?ctrfmc | 1 |
| GetTargetWindow | Liefert Reglerzielfenster | ?twi | 3 |
| SetController | Regler-Modus einstellen | !ctr | 1 |
| SetControllerCall | Regleraufruf einstellen | !ctrc | 1 |
| SetControllerFactor | Reglerfaktor einstellen | !ctrf | 1 |
| SetControllerSteps | Regler-Schritte einstellen | !ctrs | 1 |
| SetControllerTimeout | Regler-Überwachungs-Timeout einstellen [ms] | !ctrt | 1 |
| SetControllerTWDelay | Reglerverzögerung setzen | !ctrd | 1 |
| SetCtrFastMoveOff | Fast Move Funktion „AUS“ | !ctrfm 0 | 1 |
| SetCtrFastMoveOn | Fast Move Funktion „EIN“ | !ctrfm 1 | 1 |
| SetTargetWindow | Reglerzielfenster einstellen | !twi | 3 |
| | Zielfenster (Alternative zu „twi“) | (?)!poswindowrange | |
| | P-Anteil des Lageregers | (?) !posconkp | 2 |
| | Filter Zeitkonstante zum Ausgangsfilter (Tiefpass) des Lagerreglers | (?) !posconoutpass | 2 |
| | Achsenfreigabe für Lageregler | (?) !posconenable | 2 |
| | Ein- und Ausschalten des Lagereglers | (?) !poscon | 2 |
| | Schleppfehlerüberwachung / Bereich | (?) !deviationsrange | 2 |
| | Schleppfehlerüberwachung / Zeitfenster | (?) !deviationstime | 2 |
| | Schleppfehlerüberwachung / Ein- oder Ausschalten | (?) !deviationcheck | 2 |

**Trigger-Ausgang**

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| GetTrigCount | Triggerzählerstand setzen | ?trigcount | 3 |
| GetTrigger | Einstellung vom Trigger auslesen | ?trig | 3 |
| GetTriggerPar | Trigger Parameter auslesen | ?triga<br>?trigm<br>?trigs<br>?trigd | 3 |
| SetTrigCount | Triggerzählerstand lesen | !trigcount | 3 |
| SetTrigger | Trigger Ein/ Aus | !trig | 3 |
| SetTriggerPar | Trigger Parameter | !triga<br>!trigm<br>!trigs<br>!trigd | 3 |

**Snapshot-Eingang**

| API-Befehl | Kurzbeschreibung | LSTEP-Befehl | X |
|---|---|---|---|
| GetSnapshot | Einstellung vom Snapshot auslesen | ?sns | 3 |
| GetSnapshotCount | Snapshot-Zähler | ?snsc | 3 |
| GetSnapshotFilter | Eingangsfilter auslesen | ?snsf | 3 |
| GetSnapshotPar | Snapshot-Parameter auslesen | ?snsl<br>?snsm<br>?sns | 3 |
| GetSnapshotPos | Snapshot-Position auslesen | ?snsp | 3 |
| GetSnapshotPosArray | Snapshot-Position aus Array auslesen | ?snsa | 3 |
| SetSnapshot | Snapshot Ein/Aus | !sns | 3 |
| SetSnapshotFilter | Eingangsfilter setzen | !snsf | 3 |
| SetSnapshotPar | Snapshot-Parameter | !snsl<br>!snsm<br>!sns | 3 |

## Anhang:

**Herstellererklärung LSTEPexpress**

CE-Konformitätserklärung gemäß den EG-Richtlinien 2004/108/EG und 2006/95/EG

---

Hiermit erklären wir, dass das Produkt

***Type :*** **Motorsteuerung LSTEPexpress**

in der von uns in Verkehr gebrachten Ausführung den grundlegenden Sicherheits- und Gesundheitsanforderungen den unten genannten EG-Richtlinien entspricht.

Bei einer nicht mit uns abgestimmten Änderung des Produktes verliert diese Erklärung ihre Gültigkeit.

***EMV- Richtlinie 2004/108/EG:***

***Angewandte harmonisierte Normen:***

| | |
|---|---|
| EN61800-3 - 2004 | Drehzahlveränderbare elektrische Antriebe |
| EN61000-6-3 - 2007 | Emission – Fachgrundnorm – Wohnumgebung |
| EN61000-6-2 -2005 | Störfestigkeit– Fachgrundnorm Industrie |
| EN61000-3-2 - 2006 | Emission Oberwellen |
| EN61000-3-3 – 1995+A1, A2 | Emission Flicker |
| EN60601-1-2 – 2001+A1 | Produktnorm für elektrische medizinische Geräte |

*Die Konformitätsvermutung basiert auf der Dokumentation der Testergebnisse, Prüfbericht Nr. XXXXXXX*

***Niederspannungs-Richtlinie 2006/95/EG:***

***Angewandte harmonisierte Normen:***

| | |
|---|---|
| EN 6010:2001 | Sicherheitsbestimmungen für elektrische Mess-, Steuer-, Regel- und Laborgeräte — Teil 1: Allgemeine Anforderungen |

Hüttenberg 19.04.2010

***Ort*** ***Datum*** ***Unterschrift*** ***Technischer Leiter Entwicklung***

## Herstellererklärung LSTEP-PCIexpress

Hiermit erklären wir, dass die Steuerung LSTEP-PCIexpress kein gebrauchs- oder anschlussfertiges Gerät im Sinne des „Gerätesicherheitsgesetzes“, des „EMV-Gesetzes“, oder der „EG-Maschinenrichtlinie“, sondern eine Komponente ist.

Erst durch die Einbindung in die Konstruktion des Anwenders wird die letztendliche Wirkungsweise festgelegt. Die Übereinstimmung der Konstruktion des Anwenders mit den bestehenden Sicherheitsbestimmungen und Rechtsvorschriften liegt im Verantwortungsbereich des Anwenders.

Hinweise und Empfehlungen zur Installation und zum bestimmungsgemäßen Betrieb sind in der Betriebsanleitung enthalten.

Die Inbetriebnahme der Steuerung ist so lange untersagt, bis festgestellt wurde, dass alle gesetzlichen Schutz- und Sicherheitsanforderungen eingehalten wurden. Zugrunde liegende Anforderungen entnehmen Sie der Herstellererklärung der LSTEPexpress.