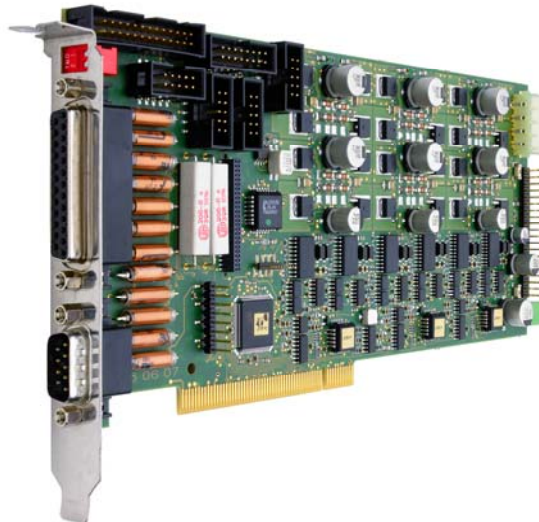


Precision Positioning Systems

LSTEP / ECO-STEP



LSTEP / PCI



LANG GMBH & CO. KG
Dillstraße 4
D-35625 Hüttenberg
Tel. +49 (0) 6403/7009-0
Telefax 06403/7009-40

Contents

	Page
Foreword	
1 Safety Instructions	1 • 1
1.1 General Instructions.....	1 • 1
1.2 Initial Start-Up Information.....	1 • 2
2 Functional Description	2 • 1
2.1 RS232 Interface	2 • 1
2.1.1 Operation Without Control Computer	2 • 1
2.2 Controls	2 • 2
3 Initial Start-up	3 • 1
3.1 Connections.....	3 • 1
3.2 Input / Output Port Data	3 • 1
3.3 Connection Of Incremental Measuring Systems.....	3 • 2
3.4 Function Test.....	3 • 3
3.5 Possible Problems When Setting Up The RS 232 Connection And Their Solutions .	3 • 4
3.6 Firmware update	3 • 5
3.6.1 Firmware update with the new Flashtool starting version 3.0.0.0.....	3 • 6

4 The LSTEP Controller Instruction Set	4 • 1
4.1 Short description of the LStep Instruction set	4 • 2
4.2 Firmware and Hardware Information	4 • 9
4.3 Reset	4 • 11
4.4 Interface Configuration	4 • 12
4.5 Instruction Set Used and Save Functions	4 • 13
4.6 Status and Fault / Error Messages	4 • 15
4.7 Settings	4 • 22
4.8 Determination Of The Mechanical Work Range	4 • 33
4.9 Travel Instructions And Their Control Functions	4 • 38
4.10 Joystick- Handwheel- and Trackball Instructions	4 • 43
4.11 In/Outputs	4 • 51
4.12 Interpretation Of Incremental Measuring Systems	4 • 57
4.13 Controller Settings For LSTEP	4 • 61
4.14 Special Instructions for the MR-System	4 • 67
4.15 Interpretation Of Clock Pulse And Direction Of Rotation Specifications	4 • 70
4.15.1 Range Of Travel Monitoring	4 • 70
4.15.2 Temporal Marginal Conditions For the Signals	4 • 70
4.15.3 Clock pulse and Turning direction-outputs for additional axes	4 • 73
4.16 Configuration Of The Trigger Output Signal	4 • 76
4.17 Configuration Of The Snapshot Input	4 • 80

5 Appendix General	5 • 1
5.1 Multi-Function Port Pin Assignment <i>(Not for ECO-STEP)</i>	5 • 1
5.2 RS232 Interface Pin Assignment	5 • 3
5.3 The Interface Cable	5 • 3
5.4 Joystick Connection Pin Assignment	5 • 4
5.5 The CAN Interface Optional	5 • 4
5.6 The Handwheel Connection (Coax Connector)	5 • 5
5.7 Interpreter For MULTICONTROL Commands	5 • 6
5.7.1 Input Of Parameters	5 • 6
5.7.2 Supported Multicontrol Commands	5 • 6
5.8 Motor Connection	5 • 10
5.9 Troubleshooting	5 • 10
6 Appendix LSTEP	6 • 1
6.1 Back Panel Of The LSTEP	6 • 1
6.2 Motor Connection X/Y/Z	6 • 1
6.3 Encoder Connection X/Y/Z <i>(Not for Eco-Step)</i>	6 • 2
6.4 The Power Supply Module	6 • 2
6.5 DIP Switch Settings	6 • 2
6.6 Technical Data	6 • 3
6.7 Wiring Of The Motor	6 • 4
6.8 Testing and Calibration Instructions	6 • 5
6.9 View Of The Circuit Boards	6 • 6
6.10 Transformer Wiring	6 • 7
6.11 I/O - Card for LSTEP Controller	6 • 8
6.12 Trackball for LSTEP	6 • 11

7 Appendix ECO-STEP / ECO-DRIVE, ECO-MOT)	7 • 1
7.1 Back Panel Of The ECO-STEP	7 • 1
7.2 Plugs	7 • 1
7.2.1 Motor Connection X/Y/Z	7 • 1
7.2.2 Voltage Connection	7 • 2
7.2.3 ST 4; 15-pol HD-Sub-Socket: Koax drive	7 • 2
7.2.4 ST 2: 9-pol D-Sub-Plug: Joy-Stick, Stop, Snap-Shot	7 • 3
7.2.5 ST3, 9-pol D-Sub-Socket: RS 232-Interface	7 • 3
7.2.6 St6, 10-pol. Connecting plug, D-Sub-Configuration: CAN-Bus	7 • 4
7.2.7 ST 8, 26-pol-Connecting plug: Connection for Control panel connector	7 • 5
7.3 Jumper Configuration	7 • 5
7.4 DIP Switch Settings	7 • 6
7.5 Technical Data	7 • 7
7.6 View Of The Circuit Boards	7 • 8
7.7 The Powerpack	7 • 9
7.7.1 Technical Data For The Power Pack	7 • 9

	Page
8 Appendix LSTEP-PCI	8 • 1
8.1 Jumper	8 • 1
8.2 Switch	8 • 1
8.3 Solder bridges	8 • 2
8.4 LED's	8 • 3
8.5 Plugs	8 • 3
8.5.1 ST1, 9-pol D-Sub-plugs: Joy-Stick, Stop, Snap-Shot	8 • 3
8.5.2 ST2, 10-pol Female connector with D-Sub-connection: RS 232-Interface	8 • 4
8.5.3 St7, 10-pol. Female connector, D-Sub-connection: CAN-Bus	8 • 4
8.5.4 St5, 8-pol Female connector measuring point 1-8.	8 • 5
8.5.5 ST3, 25-pol D-Sub-socket: Motor- and proximity switch connection	8 • 6
8.5.6 St6, 16-pol Female connector (D-Sub-counter): TTL-encoder input	8 • 7
8.5.7 St8, 16-pol-Female connector (normal counter): Encoder -plugincard	8 • 8
8.5.8 St11, 26-pol Female connector, D-Sub counter: Multi functioning port	8 • 9
8.5.9 St10, 10-pol Female connector with D-Sub-connection: Analogue I/O	8 • 11
8.5.10 ST4, 4-pol PC-supply unit plug: Motor power supply	8 • 11
8.5.11 St 9, 46-pol Female connector: System bus (for extension module)	8 • 12
8.5.12 ST8, 50-pol Female connector: For Sin.- Cos.- Encoder evaluation PCIcompact	8 • 13
8.5.13 ST12, PCI-Bus/ PCIcompact	8 • 14
8.5.14 ST14 / 10-pol Male connector with D-Sub-assignment: Encoder	8 • 15
8.5.15 STt 9 / 40-pol Male connector with DSub assignment: 16 digitale I/O's	8 • 16
8.5.16 ST15 / 2-pol Plug: 24V power supply for digital I/O's PCIcompact	8 • 16
8.6 Measuring point	8 • 17
8.7 Fuses	8 • 18
8.8 Encoder adapter card card for the LSTEP-PCI and analogue outputs	8 • 18
8.9 Description I / O - card for the LSTEP-PCI	8 • 21
8.9.1 Connections of the 46-pin-bus adapter	8 • 21
8.9.2 ST4: 2-pol Power plug for the supply of the In- and Output ST4/ PCI, ST15/ PCIcompact	8 • 22
8.9.3 40-pol Female connector with 16 inputs, 16 outputs ST2/ PCI, ST9/ PCIcompact	8 • 22
8.10 Assembly scheme	8 • 23
8.11 Appendix LSTEP-PCI Technical Data	8 • 28

9 Appendix LSTEP-API	9 • 1
9.1 Introduction	9 • 1
9.1.1 Included Functions	9 • 1
9.1.2 System requirements	9 • 1
9.1.3 Supported Development Environments	9 • 1
9.2 DLL Interface	9 • 2
9.2.1 LSTEP-API	9 • 2
9.2.2 LSTEP4X-API.....	9 • 2
9.2.3 General Information	9 • 2
9.2.3.1.LSTEP4.DLL.....	9 • 2
9.2.3.2.LSTEP4X.DLL.....	9 • 2
9.2.3.3. Difference in comparence with LSTEP4.DLL.....	9 • 2
9.2.4. Integration in Delphi	9 • 3
9.2.4.1. LSTEP4-API	9 • 3
9.2.4.2. LSTEP4X-API.....	9 • 3
9.2.5. Integration in Visual C++	9 • 4
9.2.5.1. LSTEP4-API	9 • 4
9.2.5.2. LSTEP4X-API.....	9 • 5
9.2.6. Integration in LabVIEW	9 • 5
9.2.6.1. Differences between LSTEP4. LLB and LSTEP4X.LLB	9 • 6
9.2.6.2. Procedure for using an LSTEP4 VIs	9 • 6
9.3 Notations to create own programs for programming the controller via the API	9 • 9
9.3.1. Initialising the Controller.....	9 • 10
9.3.2. Own Program part.....	9 • 12
9.4 Functions	9 • 13
9.4.1. Index for API-Instructions	9 • 13
9.4.2. Functions	9 • 20
9.5 Error Codes LSTEP / API	9 • 136
9.6 Frequent questions & answers	9 • 137
9.7 Use of the LStep PCI-card	9 • 141
9.7.1 Interrupt-controlled Communication with LStep-PCI	9 • 142
9.7.2 Readme.....	9 • 142
9.7.3. API / LSTEP Commands.....	9 • 143

Dear Customer!

Thank you for choosing one of our controllers!

With this unit, you have chosen a positioning controller which automates complex positioning tasks yet takes up a minimum of space. The high precision of the controller opens up vast application possibilities. The resolution of up to 50,000 (100,000) steps per motor revolution for a two-hundred step motor and 2000 microsteps per full step for linear stepping motors offers resolutions in the sub- μm range. In addition, „closed loop„ operation in connection with high-resolution transducer interpretation with optical and magnetic measuring systems provide a very high positioning accuracy.

The many additional functions, such as e.g. snapshot, triggerout, clock pulse and direction of rotation inputs make this controller the ideal partner for many applications.

Before putting your controller into operation, please take the time to read this manual through carefully.

Pay particular attention to the safety instructions!

Contents subject to change. We accept no liability for any errors in this documentation. Due to continuous technical development of our products, the descriptions given in this documentation may differ slightly from your machine. No liability whatsoever is accepted for direct damage arising in connection with the supply or use of this documentation, unless there is a legal obligation.

Copyright In Accordance With DIN 34

No part of this documentation may be transmitted or copied, nor may the contents thereof be used or imparted to third parties in any way without the prior, express permission of the publisher.

Failure to comply will result in a claim for damages. All rights with regards to the granting of patents and design registration reserved.

1 Safety Instructions



1.1 General Instructions

- Maintenance and repair work must only be done by duly qualified and trained experts who have sufficient knowledge of the controller!
- Pull out the mains plug before opening the unit!
- The power consumption of LSTEP-2x/2 may rise up to 200 VA for brief periods, when all three axes are being operated at 2.5 A and at maximum speed. This high power consumption is not however permissible for continuous operation, as LSTEP-2x/2 works without additional cooling (fan). An average power consumption of 100 VA should not be exceeded!
- For the controller model LSTEP 22/2 (3,75), the standing power consumption must be reduced to at least 75%!
- Only devices specified by us may be connected.
Failure to heed this instruction could cause irreparable damage to the controller or to the device connected to it!
- The main power plug for the controller or the socket into which the controller is plugged must be accessible at all times, so that the controller can be disconnected in all poles from the power supply at any time!
- Do not plug in or disconnect any cables whilst the equipment is switched on!



1.2 Initial Start-Up Information

- LSTEP Interface Assignment.**
 The Commander (terminal for LSTEP and MCL) is supplied with a controlled d.c. voltage of +5V or +12V via the interface port. For the LSTEP, the voltage is connected to PIN 1 of ST2 via the jumper J1 (behind ST2), provided that a terminal is supplied with the controller.
 Please therefore always use the original interface cable provided by Messrs. LANG.
Note for ECO-STEP: A wire strap is used instead of a jumper.
- Setting The Mains Voltage.**
 The LSTEP can be operated at 100V - 120V or at 200V - 240V . The required voltage is set on a pluggable voltage selector with fuse carrier at the power input. Make sure that the unit is always operated at the voltage which has been set. If the voltage selector has been set for 100V - 120V but the LSTEP is connected to 200V - 240V , the **control electronics could be irreparably damaged**. The power input fuse will most certainly blow!
Note for ECO-STEP: The ECO-STEP is supplied by an external power pack with a 100-240V wide range input.
- Ventilation slots in the housing.**
 Ventilation slots are provided in the housing to cool the power electronics of the controller by ventilation. You must make sure that no chips, liquids or other electrically conductive substances get inside the housing. This applies to the LSTEP-3x/2 (phase current up to 5A) only.
- Protection Of The Connected Mechanical Components**
 After the controller has been switched on, the range of travel should always be checked with the commands "Calibrate" and "Measure Table Stroke". The controller is then able to detect and prevent any movement which would exceed the maximum range of travel (see Chapter 5.1, Chapter 5.2).
 Once the travel limits have been set, the axis will only travel to the preset limits. This is necessary when you only have one limit switch available per axis.

2 Functional Description

The LSTEP stepping motor controller is used to operate coordinate tables, e.g. for microscopes or production cycles with resolutions up to 0.0001 mm. The controller excels due to extremely smooth and quiet running of the motors. Despite a high resolution of 50,000 (100,000) microsteps per motor revolution, the dynamic microstep drive principle allows high speeds of up to 40 r/sec. (7,5 r/sec) to be achieved with a 200 step motor. For linear stepping motors, individual motor tables with 2000 microsteps per full step, i.e. 8000 microsteps per tooth pitch are used.

The controller works with linear interpolation (all axes reach the target position simultaneously) and automatic, individually programmable ramp generation (Limitation of the acceleration when starting and stopping). The LSTEP can be operated as a stand-alone unit or can be controlled from a PC. A position indicator (optional) at the front panel and a "Joystick" round off the unit. A new instruction set has been developed for the LSTEP, which offers considerably more functionality. The instruction register set which has been used successfully for years on the "MCL" controller remains available.

To ensure smooth running and accurate positioning, motors with a step angle error of $< \pm 3\%$ should be used. To reach the maximum speed, low impedance motors with a low inductance should preferably be used.

To avoid unnecessary heating of the motors, LSTEP reduces the motor current to the preset zero signal current every time there is a pause in operation (even for joystick operation) (see Chapter 6.7).

2.1 RS232 Interface

A n RS232 serial interface with the following standard settings is used as the standard interface to the higher-ranking PC:

- 9600 baud, 11 bit frame, 1 start-, 8 data-, no parity-, 2 stop bits

For trouble-free operation, LSTEP needs an RS 232 port at the PC with the following signals:

RxD:	LSTEP receive line	(computer transmit line)
TxD:	LSTEP transmit line	(computer receive line)
RTS:	LSTEP ready to send	
CTS:	PC clear to send	
GND:	Signal ground	

Operation without the RTS line is possible with certain restrictions.

2.1.1 Operation Without Control Computer

Simple movements can be made with the LSTEP, without a control computer. The "Joystick" switch is set to "manual" for this purpose. Any position can now be approached using the joystick. On controllers with an LC-display, the momentary absolute position is continuously displayed. In addition, the axes can be set to zero individually with help of the "CLEAR" switch.

2.2 Controls

The display (only on units which are equipped with a display) and all controls, except the main power switch, are located on the front panel.

Control	Function
CLEAR X/Y/Z (optional)	Switch for resetting the display (separate for X- Y-and Z- position). The position register is also deleted.
SPEED 1..10 (optional)	Potentiometer for changing the motor speed when running with external clock pulse. The speed set in the software can be regulated from 0 to 100%. The parameter value set before a vector is started is valid for travelling the whole vector and cannot be changed whilst travel is in progress. If the joystick is active, the speed of travel can be changed on the potentiometer. Note: Especially interesting for joystick manual mode.
JOYSTICK MAN / AUTO	Joystick selector switch MAN = Manual mode (no “move” commands can be executed) AUTO = Automatic mode with the appropriate commands
RESET (optional)	When the reset switch is switched up, the controller is returned to starting status (just as if you had switched it off and on).
ON	Shows LSTEP is on and ready for operation
LCD Display (optional)	LCD display with 4*16 characters for displaying the mode of operation and the absolute position. Positions in a value range of $-99.999.999,9 \leq P \leq +99.999.999,9$ are displayed .

Table 1:Controls At The Front Panel Of The LSTEP

Joystick Switch Set At "AUTO"		
	READY TO RECEIVE	LSTEP waits for commands via the RS232 interface
	GO TO POSITION	LSTEP moves to a position
	RELATIVE STRAIGHT LINE	LSTEP travels a relative straight line
	CALIBRATION	LSTEP moves to zero position
	TABLE LENGTH	LSTEP moves to the end limit position
	JOYSTICK AUTO	LSTEP moves with joystick operation
Joystick Switch Set At H		
	JOYSTICK MAN	LSTEP moves with joystick operation

Table 2: LSTEP Modes Of Operation

3 Initial Start-Up

CAUTION: The ventilation grate at the back of the unit and on the bottom plate (LSTEP-3x/2) must not be covered!

3.1 Connections

- Connect the motors using the cable supplied.
- Connect the incremental measuring systems (if any).
- Connect the joystick and lock it into place with the slides.
- Connect the computer or Commander with the interface cable.
- Connect the power supply.

3.2 Input/Output Port Data *(not for ECO-STEP)*

The following power ratings must be maintained for the inputs/outputs

- Digital inputs (e.g. clock forward/back, moment trigger)

Signal level:	TTL; max. input current $\pm 5\text{mA}$
Existing input wiring	RC-low pass with $470\ \Omega$ / 220pF ,
	$4.7\ \Omega$ Pull-Up at +5V
- Digital outputs (Trigger-Out)

TTL-level with $\pm 1.6\ \text{mA}$

3.3 Connection Of Incremental Measuring Systems *(not for ECO-STEP)*

Incremental rotary or linear encoder systems for detection or avoidance of a step offset can be connected. This allows closed loop operation. The uses are not restricted to optical measuring systems. Inductive or magnetorestrictive systems can also be interpreted, provided that their output signals keep to the specified limits. The optional encoder interface allows encoder systems with sinusoidal output signals to be connected.

The following two alternatives are available:

1. sinusoidal voltage signals 1V_{SS}.
2. magnetic linear transducer.

Due to the limited data capacity of micro controllers, not all combinations of spindle pitch values and encoder graduation periods give correct position calculation results. Some of the possible spindle pitch and period graduation combinations for linear measuring systems are given in the table below. An (X) means that the combination can be used without restriction.

Spindle Pitch in mm	Encoder Graduation in mm						
	1.00 mm	0.50 mm	0.10 mm	0.020 mm	0.0080 mm	0.0040 mm	0.0001 mm
0.40 mm	X	X	X	X	X	X	X
0.50 mm	X	X	X	X	X	X	X
1.00 mm	X	X	X	X	X	X	X
2.00 mm	X	X	X	X	X	X	X
3.00 mm							
4.00 mm	X	X	X	X	X	X	X
5.00 mm	X	X	X	X	X	X	X
8.00 mm	X	X	X	X	X	X	X
10.00 mm	X	X	X	X	X	X	X
15.00 mm							
20.00 mm	X	X	X	X	X	X	
25.00 mm	X	X	X	X	X	X	
30.00 mm							
35.00 mm							
50.00 mm	X	X	X	X	X	X	
100.00 mm	X	X	X	X	X	X	

Table: Permitted encoder graduations (X) depending on the selected spindle pitch

The following equation can be used for instances not given in the table.

$$\text{encoder factor} = \frac{4 \cdot 10^5 \cdot t_p}{h}$$

h : spindle pitch in mm

t_p : encoder graduation in mm

If the selected spindle pitch and the period graduation of the measuring system results in a whole number without decimal for the *encoder factor*, the selected combination can be used without restriction.

In all other cases, please contact the manufacturer of the controller.

3.4 Function Test

- Switch on the unit
After it has been switched on, LSTEP performs an automatic calibration of the connected joystick. This takes about 5s. To ensure correct calibration, the joystick must not be deflected during this time.
- Set the "Joystick" switch to "MAN"
- Move the joystick in all directions: The motors run according to how you move the joystick. If however there is no reaction, check the motor and joystick connections. If the connections are ok., inspect the unit for possible, hidden transport damage.
- Set the "Joystick" switch to "AUTO"
- Call the functions (see instruction set)

3.5 Possible Problems When Setting Up The RS 232 Connection And Their Solutions

- LSTEP is not responding via RS 232:
 - Check the pin assignments and the connecting cable to the control computer
 - Check the interface conditions (OPEN-command) at the control computer
- Individual bytes from LSTEP messages are being lost:
 - There is no CTS line available at the control computer (RTS -line of the LSTEP is not checked): When the LSTEP is working and cannot receive data, the interface is blocked via RTS. Synchronization of the computer and the LSTEP also takes place without a check of the RTS line, when the computer is waiting for the LSTEP status signals, as described in the examples in this manual. Problems may however arise in the functions "Set resolution and spindle pitch", if a safe protocol was not established with the "Autostatus" instruction (see instruction set). In this case, the computer must be delayed, e.g. with the help of loops, so that the data or commands are not lost. The typical delay is approx. 20 msec.

3.6 Firmware Update

The controller can be updated easily with program updates. Depending on the controller used, please follow the procedure described below:

- Connect one of the serial interfaces (COM) of your PC to the serial interface at the back of the controller (LSTEP + ECO-STEP).
 - Close all programs which access the same interface.
 - Copy the self-unpacking program „LFlash.exe„ onto your PC and unpack it.
 - Copy the new control program „*.ihx„ onto your PC
 - Start „Flash.exe„ in Windows
 - Select the interface of the PC which you have connected with the controller,
 - Select the type of unit.(LSTEP; ECO-STEP)
 - Set the dip switch "1" at the back of the unit to ON and then switch on the controller. Using the PCI-card make a reset after switching on the dip switch 1 with the dip switch 2 (switching ON/OFF)
 - Click on the **Update** box and confirm with "Yes". The old program is deleted from the controller (depending on the program version only the banks 0-2 have to be deleted).
 - Select the file type (IntelHexFile).
 - Load the new control program.
- ➔ The firmware is now transmitted to the controller.
- When programming is complete, return dip switch "1" to its original position.

To subsequently operate the controller with the new firmware, you must either press the reset button, or switch the controller off and on again.



3.6.1 Firmware update with the new Flashtool starting version 3.0.0.0

How to perform an Update:

1. Select the serial interface.
2. Set the DIP-switch 1 of the control to ON:
3. Now switch on the control and actuate the reset button while the control is still switched on.
4. Start the Update. Therefore click the button with the inscription „Update“. The program automatically establishes a connection to the control and deletes the Flash. Afterwards you are requested to select the new control software. After selecting the file it will be saved in the Flash..
5. After a successful Update set the DIP-.switch 1 back to ‚OFF‘.
6. Actuate the reset-button or switch the control OFF and ON. This way you start the control with the new software.

4 The LSTEP Controller Instruction Set

For better clarity, all instruction and parameters, which are sent to the controller and all acknowledgements/feedback's from the controller, are transmitted as ASCII characters. The advantage of this is that on the one hand, the commands can be input manually at a normal terminal. On the other hand, these plain language commands make troubleshooting easier, when a customised program sets the commands.

Commands or parameters which are transmitted to the controller begin with an exclamation mark "!". Inquiries are denoted with a question mark "?". For example, the following mean:

<i>!cal</i>	<i>Calibrate</i>
<i>?status</i>	<i>Read out status</i>

Note: For write-only or read-only instructions, the characters "!" or "?" may be omitted.

Some instructions, e.g. specification of travels, require the transmission of parameters. These are then transmitted after the instruction itself. A space must be inserted and transmitted between the command text and the parameters and between the various individual parameters to separate them.

<i>moa 45 13 20</i>	<i>Move x, y and z to the positions 45, 13 and 20</i>
---------------------	---

Each instruction must be concluded with a carriage return (CR). This character is shown in the ASCII character set as follows:

Symb. Name	Dec. Value	Hex. value	Bin. value
CR	13	0xD	00001101

4.1 Short description of the LStep Instruction set

Instruction	Example	Note	Chap.4 Page
-------------	---------	------	----------------

Interface			
baud	(?) !baud 9600	set baud rate to 9600	12
cts	(?) !cts 0	the CTS-interpretation is deactivated	12
intcom	(?) !intcom 1	Communication interrupt-controlled through DPRAM	12

Interface			
ver	?ver	read version number	9
iver	?iver	addition to the current version number	9
det	?det	read out detailed version number	10
readsn	?readsn	read out serial number of the controller	11

Instruction	Example	Note	Chap. 4 Page
-------------	---------	------	--------------------

Settings			
ipreter	(?) !ipreter 0; 1; or 2	switching the command set	13
xycomp	(?) !xycomp (1-6)	for drives which influence themselves	39
dim	(?) !dim 1	setting the unit in μm	22
pitch	(?) !pitch 1 1 1 (y 4)	setting the spindle lead X Y Z or only Y	23
gear	(?) !gear	gear factor	23
accel	(?) !accel 1 1 1 (x 1)	setting the acceleration X Y Z or only X	24
vel	(?) !vel 10 10 10 (x 20)	setting the speed X Y Z or only X	24
velfac	(?) !velfac (1-100)	reduction of the set speed	25
pot	(?) !pot 1(0)	switch Speedpoti ON/OFF	40
cur	(?) !cur 1 2 2.5	motor current setting: X=1A Y=2A Z=2,5A	26
maxcur	?maxcur	all max. currents are indicated (=configuration)	25
reduction	(?) !reduction 0.5 0.5 0.5	current reduction to 50% in all axis'	26
curdelay	(?) !curdelay 1000	deceleration of the current reduction (0 - 10000 ms)	27
opfl	(?) !opfl 20 20 20 20	starting 20 rpm. the maximum current will be driven	20
axis	(?) !axis1 0 1 (y 1)	switch axis ON/OFF	27
axisdir	(?) !axisdir 0 1 0	motor turning direction turned of Y-axis	28
caliboffset	(?) !caliboffset 1 1 1 1	The zero position will be shifted 1mm with Dim 2	34
rmoffset	(?) !rmoffset 1 1 1 1	The end position will be shifted 1mm with Dim 2	34
caldir	(?) !caldir z 1	the Z-axis will be calibrated in positive direction	35
calbspeed	!calbspeed 10	when "cal"+"rm" the speed is 0,1U/s while driving out of the limit switch (5...100)	36
calrefspeed	!calrefspeed 0-100	Speed during calibrating when searching the reference mark	36
Save	save	the current parameters are burned into the Flash	14
savejoyonoff	(?) !savejoyonoff 1 (0)	after a subsequent Save and Reset, the joystick is active with the LSTEP-PC after switching on the PC.	47
saveipreter	(?) !saveipreter 0	after a subsequent Save and Reset, the LSTEP is set permanent to the register instruction set	14
Reset	Reset	the software is set back to the start condition	11
pa	pa 1	power amplifier 1 (power stage switch on) 0 = switched off	11
vlevel	(?) !vlevel 1 0.8 !vlevel 2 1.2	Fade out of speeds, with which resonances arise.	20
mtpatch	(?) !mtpatch 1	the correction table as activated	21
joyfilter	(?) !joyfilter 1	Filtration and Hysteresis activated in joystick operation	21
stoppol	(?) !stoppol 0 oder 1	The stop input (MFP) is low or high active	32
stopaccel	(?)!stopaccel 2	The stop acceleration is 2m/s^2 when the stop input is active	32

Instruction Example Note

Status inquiry			
autostatus	(?) !autostatus 0 (0-4)	setting the acknowledgement of the controller	15
Status	?status	shows the current condition of the controller	15
Statusaxis	?statusaxis	current condition of each axis (@,M,J,C,S,A,D,-,)	16
err	?err	shows the current error number	16
statuslimit	?statuslimit	A= calibrated; D= measured table stroke; L=Software end position;-=basic setting.	29
securityerror	(!)?	see description	19
securitystatus	(!)?	see description	18

Moving Command and position administration			
cal	!cal	calibrate	33
rm	!rm	measure table stroke	33
delay	(?) !delay 1000	delays the vector start by a second	42
moa	!moa 10 10 10 (x 10)	drive to absolute position X Y Z (only X)	38
mor	!mor 4 4 4 (y 4)	relative-positioning X Y Z (only Y)	38
m	!m	start of a move (track with mor or.distance)	39
distance	(?) !distance	set the track for X Y Z (start with "m")	40
a	!a	Cancel (Stop)	42
moc	!moc od. moc	all axis are centered (centerpoint of the software limits)	42
pos	(?) !pos 0 0 0 (z 0)	set or read position	41
clearpos	!clearpos	all position values are set to zero (for endless turning axis)	41
calpos	?calpos	sends back the position (depending on the period of the measuring system), where the proximity switch was left.	35
refdir	(?) !refdir x 1	X-axis moves after the instruction "ref" in positive direction	37
ref	!ref	X-axis calibrates on the reference switch (only possible with LSTEP)	37

Instruction Example Note

Chap.4
Page

Joystick and Hand wheel			
speed	(?) !speed 5 5 5 (y 10)	digital Joystick, all axis are turning with 5U/S od.Y with 10	43
joydir	(?) !joydir 1 1 -1	set motor turning direction for the Joystick	44
joy	(?) !joy 0 (1) (2) (3) (4)	switch Joystick ON/OFF with or with position counter	45
	?joy	acknowledgement "M" (Joystick manual active)	45
joywindow	(?) !joywindow 10	set range where the axis move (0-100)	46
joychangeaxis	(?) !joychangeaxis 1	Changes the allocation of the AD-Joystick channels Changes of allocation of X and Y axes	46
hw	(?) !hw 1 (0-4)	activation of the hand wheel	48
hwvel	(?) !hwvel 1.00001.0000	The max. speed in hand wheel operation = 1U/s	48
hwaccel	(?) !hwaccel 0.50.5	The acceleration in hand wheel operation =0,5 m/s ²	49

Control panal with Trackball and Joyspeed-keys or Trackball with function keys			
joyspeed	(?) !joyspeed (1-3) 25	parameter 1;2 or 3 with speed 25	45
bpz	(?) !bpz (1-4)	control panal / Trackball 0=Aus 1= On, Joyspeed keys 2= On, with trackball-factor, with Joyspeedkeys 3= On, without trackball-factor, with Function keys 4= On,with trackball-factor, with funktion keys	49
bpztf	(?) !bpztf 10	trackball-factor = 10 (value range = 0,01 to 100)	50
bpzbl	(?) !bpzbl 0.01 0.01	Trackball-Back-Lash (set reversal backlash; 1/10µ bis 15µ)	50

Instruction

Example

Note

Chap.4
Page

Limit switch (Hardware a. Software)			
lim	(?) !lim 0 10 0 10 0 10	moving range for all axis 0+10mm	28
limctr	(?) !limctr x 1	monitoring of range for all axis X is active	29
nosetlimit	(?) !nosetlimit 1 1 1 1	no value range is set for all axis	30
swpol	(?) !swpol 1 0 1 (z 1 0 1)	assign polarity of the limit switch for all axis or only for Z.	30
swact	(?) !swact 1 0 1 (z 1 0 1)	switch ON/OFF limit switch for all axis or. only Z	31
readsw	?readsw	read the staus all limit switches	31

Digital and analog Input and Output			
digin	?digin od. ?digin 8	read all inputs or input 8	51
digout	!digout 5 1/?digout	output 5 is set to 1 / read status of all outputs	51
digfkt	!digfkt 7 0 / ?digfkt 9	-no infulence to E-7/ A-7 / -read function of E 9/ A-9	52
edigin	nur bei LSTEP-44	(like digin)	53
edigout	nur bei LSTEP-44	(like digout)	53
edigfkt	nur bei LSTEP-44	(for thosel/O's only the polarity can be set)	54
anain	?anain c 2	read current status of the analog channel 2	55
anaout	(?) !anaout c 1 0	set analog cchannel 1 to 0	55

Pulse-For/Back Inputs			
tvr	(?) !tvr 1 1	aktiviert Takt-F/B for X + Y	71
tvrf	(?) !Tvrf 1	factor pulse For/Backw 1= 1pulse is 1 motor increment	72

Pulse-For/Back through Interface			
px, nx	px	1 pulse in positive diection in X	72

Pulse-For/Back Outputs for further Axis			
tvrouit	(?) !tvrouit 1 1	for X + Y pulse-V/R output is active	73
tvrores	(?) !tvrores y 1000	for Y a resolution of 1000 impulses/rotation is set	73
tvropitch	(?) !tvropitch 1	a 1 mm spindle is used for the X-axis is used	74
tvroa	(?) !tvroa 1	the acceleration of the X-axis is 1m/s ²	74
tvrov	(?) !tvrov z 10	Z speed is 10 r.p/sec.	74
tvropos	(!) ?tvropos	all current position values are shown	75
tvromoa	!tvromoa 10 10	X + Y are driven absolute to 10 10	75
tvromor	!tvromor 10 10	X + Y are driven relative 10 10 from the current position.	75
tvrostatus	?tvrostatus	gives the current status: "-"=OFF "M"=Motion "@"=Stop	76

Instruction

Example

Note

Chap.4
Page

Encounder-Setting			
twi	(?) !twi 10 10 10	The target window for all axis=10μ (for dimension=1)	63
encmask	(?) !encmask 1 0 1	Encoder: X+Z active; Y deactivated	57
enc	(?) enc	Answer: 1 0=Encoder-X = aktive Encoder-Y = deactivated	57
encperiod	(?) !encperiod 0.1	division period of the X-encoder = 0,1mm	58
encres	(?) !encres	Shows the amount of encoder signal periods per motor revolution.	58
encref	(?) !encref 0	no reference signal evaluation	59
encpos	(?) !encpos 1	the encoder values are indicated by the inquiry of the positions	59
encerr	(?) !encerr 0	clear encoder error messages X; (acknowledgement= 0 or. e)	60
ctr	(?) !ctr 2 2 2	the controller for all axes stays on	63
ctrc	(?) !ctrc 10	controller call every 10 ms	64
ctrs	(?) !ctrs 9 9 9	sets the controller steps to 9 MI/ms for all axes	64
ctrf	(?) !ctrf 2 2 2	sets the controller steps to 2	65
ctrd	(?) !ctrd 5 5 5	sets the delay for all axes to 5ms	65
ctrtr	(?) !ctrtr 1-10000	controller monitoring (Timeout)	64
ctrfm	(?) !ctrfm 1	if controller difference is larger than catching area than new vector	66
ctrfmc	(?) !ctrfmc 0	clear Fast Move Counter / (?ctrfm = Abfrage Counter	66

MR-specific			
mro	(!) ?mro	sends back the determined offset values of the controller	67
mrp	(!) ?mrp	sends back the maximum value off all measuring systems	67
mrt	(!) ?mrt x	sends back the current signal value of X	68
mra	(!) ?mra y	sends back the aplification factor of Y	68
mrs	(!) ?mrsa x 0	sends the signal form sinus-X	69

LSTEP-PCI – Encounder position			
hwcount	?hwcount	read all encoder positions	60
clearhwcount	!clearhwcount	set all encoder counter to zero	60

Trigger-Output			
trig	(?) !trig	switches the trigger on	76
triga	(?) !triga	selects the axis, that should be triggered (i.e. X)	76
trigm	(?) !trigm 1	sets the trigger-mode to 1	77
trigs	(?) !trigs 4	sets the trigger-signal-length to 4μs	78
trigd	(?) !trigd 1	sets the trigger-distance to 1mm (for Dim 2)	78
Trigoffsetone	(!) ?trigoffsetone ;	-delivers the current Trigger offset for Trigger 1	79
trigoffsettwo	(!) ?trigoffsettwo	-delivers the current Trigger offset for Trigger 2	79
Trigcount	(!)?trigcount;	-reads counter setting Trigger 1	79
trigcounttwo	(!)?trigcounttwo	-reads counter setting Trigger 2	79

Instruction
Example
Note
Chap.4
Page

Snapshot-Input			
sns	(?) !sns 1	snapshot "ON"	80
snsI	(?) !snsI 1	snapshot ist high-aktive	80
snsM	(?) !snsM 1	Auto-snapshot	81
snsC	?snsC	delivers the amount of the released snapshots	81
snsP	(!) ?snsP	delivers the saved position	81
snsA	?snsA 11	Inquiry of the snapshot-position 11	82
snsF	(?)!snsF 10	serves as input filter for all rebound switches (value 0-100)	80
snsO	(?)!snsO	Snapshot Offset	82

Explanations	
!	write only ("!" can also be left out)
(?) !	write and read
?	read only ("?" can also be left out)

Possibilities for input	
Command Value Value Value Value	alle Achsen werden gesetzt oder gelesen
Command Value Value	nur X + Y werden gesetzt oder gelesen
Command Axis Value	nur die ausgewählte Achse wird gesetzt oder gelesen

Error messages	
0	no error
1	no valid axes notation
2	no executeable function
3	to many characters in the command-string
4	no valid command
5	outside valid number area
6	incorrect amount of the parameters
7	No ! Or ?
8	no TVR possible, because axis is active
9	no switching On/Off of the axes, because TVR is active
10	function is not configured
11	no Move-command possible, because joystick-hand
12	limit switch activated
13	function cannot be carried out because Encoder was recognized
14	Fault during calibration (Limit switch was not set free correctly)
15	This function is interrupted activated while releasing the encoder during calibrating or table stroke measuring if the opposite encoder is activated.
20	driver relay defective (safty circle K3/K4)
21	only single vectors may be driven (setup mode)
22	no calibrating, measuring table stroke or joystick operation can be carried out (door open or setup mode)
23	SECURITY Error X-axis
24	SECURITY Error Y- axis
25	SECURITY Error Z- axis

26	SECURITY Error A- axis
27	Emergency-STOP
28	Fault in the door switch safty circle (only with LS44/Solero)
29	Power stages are not switched on (only with LS44)
30	GAL safty fault (only with LS44)
31	While activating the joy-stick, if Move is still active.

4.2 Firmware and Hardware Information

The Firmware version can be inquired with the “ver” instructions. Which options are released in the Firmware can be inquired with the “det” instruction. Each LStep has its own individual internal serial number. This serial number can be read out with the instruction “readsn”.

Read Out Version Number	
Instruction:	?ver or ver
Parameters:	none
Description:	gives the current Firmware version number
Feedback:	LS44.xx.xxx
Error code:	--
Example:	?ver

Read out Internal Version number	
Instruction:	?iver or iver
Parameters:	none
Description:	gives detailed information of the version number
Feedback:	weekday_calendar week_year-consecutive number
Error code:	--
Example:	?iver Rückmeldung z. B.: T04_35-02-0004

Read Out Version Number In Detail		
Instruction:		?det or det
Parameters:		none
Description:		gives the detailed Firmware version number-
Feedback:	A decimal value is given which has to be converted to a hexadecimal value:	
	0x0 - - - 1	→ 1Vss encoder configured
	0x0 - - - 2	→ MR encoder configured
	0x0 - - - 4	→ TTL encoder configured
	0x0 - - 3 -	→ The second number specifies the number of axes (here 3)
	0x0 - 1 - -	→ Display configured
	0x0 - 2 - -	→ Speed poti configured
	0x0 - 4 - -	→ Handwheel (man. encoder) configured
	0x0 - 8 - -	→ Snapshot configured
	0x01 - - -	→ TVR configured
	0x02 - - -	→ Triggerout configured
	0x1 - - - -	→ 16 digital I/O configured
	0x2 - - - -	→ 32 digital I/O configured
	0x4 - - - -	→ Trackball
	The appropriate combination of the information gives the present configuration.	
Error code:		--
Example:		?det = 81697 → 13F21 _H
Description 13F21	1	16 digitale I/O configured
	3	TVR and Triggerout configured
	F	display; speedpoti; handwheel and snapshot configured
	2	2 axis
	1	1Vss encoder configured

Read Serial Number	
Instruction:	?readsn
Parameters:	none
Description:	Read out serial number of the controller.
Feedback:	9-characters
Error code:	--
Example:	?readsn

4.3 Reset

There are three ways to reset the control program:

- The hardware reset at the main power switch (for controllers without a display).
- The hardware reset with the Reset button (for controllers with display only).
- The hardware reset with the Dip-switch 1 for the PCI-card
- The software reset

Software - Reset	
Instruction:	Reset
Parameters:	none
Description:	The controller is reset to starting status
Feedback:	none
Error code:	--
Example:	Reset

Poweramplifier	
Instruction:	!poweramplifier oder !pa
Parameters:	0 or 1
Description:	This instruction applies only to the LS44-controller. !poweramplifier or !pa switches the power stages of the LS44 On (1) or Off (0).
Feedback:	--
Error code:	--
Example:	!pa 1 => Switches on all power stages of the LS44.

4.4 Interface Configuration

Baud - Rate	
Instruction:	!baud or ?baud
Parameters:	9600, 19200, 38400, 57600 or 115200
Description:	!baud 19200 → The transmission rate of the interface is set at 19200.
	?baud → gives the present transmission rate
Feedback:	Present transmission rate
Error code:	--
Example:	?baud

CTS Interpretation Of The RS232-Interface	
Instruction:	?cts or !cts
Parameters:	0 or 1
Description:	!cts 1 => activates the CTS interpretation of the RS232 interface
	!cts 0 => deactivates the CTS interpretation of the RS232 interface
	?cts => Display of the present CTS Interpretation status
Feedback:	0 or 1
Error code:	--
Example:	?cts (Display of the current CTS interpretation status)

Interrupt-controlled Communication (relevant only for LStepPCI)	
Instruction:	!intcom or ?intcom
Parameters:	0, 1
Description:	!intcom 0 → Communication with DPRAM by Polling
	!intcom 1 → Communication interrupt-controlled with DPRAM
Feedback:	0 or 1
Error code:	--
Example:	!intcom 1 (Change-over to Communication by interrupt) ?intcom

4.5 Instruction Set Used and save Functions

The controller supports three different instruction sets.

- The instruction set introduced for the new controller generation “June 2000”, described here.
- The register instruction set used on the previous controller model until June 2000.
- The multi-control instruction set (Venus).

Use the instruction described below to select the required instruction set.

Interpreter	
Instruction:	!ipreter oder ?ipreter
Parameters:	0, 1 und 2
Description:	!ipreter 0 → Register oriented instruction set
	!ipreter 1 → New instruction set
	!ipreter 2 → instruction set ITK
	?ipreter → Which instruction set? (can only read 1)
Addition:	The commands of the register instruction set.
	U7ma → Register
	U7mb → New instruction set?
	U7mc → Venus instruction set?
Feedback:	0, 1 oder 2
Error code:	--
Example:	!ipreter 0 (Switch to old instruction set) ?ipreter

The instruction set is preset in the factory, i.e. if for reasons of compatability, the old register instruction set has been set, the following command can be used to switch to the newer instruction set.

Instruction: U7mb ◀

Configuration of the command set	
Instruction:	!saveipreter
Parameter:	0 (Register), 1 (Interpreter) oder 2 (ITK)
Comment:	This command exists only for the 168' controller and needs the „save“-function, with subsequent RESET/NEW START, to be effective.
Description::	?saveipreter => reading of the current condition !saveipreter 0 => register set is configured
Feedback::	0, 1 oder 2
Error code::	--
Example::	!saveipreter 0 (Register-command set) !save (setting is burned in the Flash) !reset (NEW START)

Parameter Save Funktion	
Instruction:	Save
Parameter:	--
Comment:	This command exists only for controllers with ST10F168 controller! Save means: The current parameters (spindl pitch, aso.) are programmed in the Flash and are available immediately for a new start
Description::	save => The current parameters are programmed in the Flash.
Feedback::	in the Display with ?err the success can be controlled. i.e.: Acknowledgement = 0 => Save OK Acknowledgement unequal 0 => Save not OK (see controller-manual)
Error code::	--
Example::	--

4.6 Status and Fault / Error Messages

AutoStatus	
Instruction:	!autostatus or ?autostatus
Parameters:	0,1, 2, 3 or 4
Description:	0 → The controller is transmitting no status.
	1 → "Position reached" signals are transmitted automatically by the controller.
	2 → "Position reached" and status signals are transmitted automatically by the controller.
	3 → For "Position reached" , only a carriage return is returned.
	4 → Returns all write errors with parameters.
Feedback:	
Error code:	--
Example:	!autostatus 1 ?autostatus

Status	
Instruction:	?status or status
Parameters:	--
Description:	gives the present status of the controller
Feedback:	OK... or ERR and error message
Error code:	--
Example:	?status

StatusAxis	
Instruction:	?statusaxis or statusaxis
Parameters:	--
Description:	gives the present status of the individual axes.
Feedback:	e.g.: @ - M -
	@ → Axis stands and is ready
	M → Axis is moving (Motion)
	J → Joystick mode
	C → in control
	S → limit switch activated
	A → Acknowledgement after calibrating
	E → Acknowledgement after calibrating if a fault occurs. (Limit switch was not set free correctly).
	D → Acknowledgement after table stroke measuring
	U → setup mode (Setting Up)
	T → Timeout
	- → Axis is not enabled
Error code:	--
Example:	?statusaxis

Error	
Instruction:	?err or err
Parameters:	--
Description:	Error gives the present error number (see error description)
Feedback:	Decimal value
Error code	--
Example:	?err

Error_Nr	Description of the error messages
0	no error
1	no valid axes notation
2	no executeable function
3	to many characters in the command-string
4	no valid command
5	outside valid number area
6	incorrect amount of the parameters
7	No ! Or ?
8	no TVR possible, because axis is active
9	no switching On/Off of the axes, because TVR is active
10	function is not configured
11	no Move-command possible, because joystick-hand
12	limit switch activated
13	function cannot be carried out because Encoder was recognized
14	Fault during calibration (Limit switch was not set free correctly)
15	This function is interrupted activated while releasing the encoder during calibrating or table stroke measuring if the opposite encoder is activated.
20	driver relay defective (safty circle K3/K4)
21	only single vectors may be driven (setup mode)
22	no calibrating, measuring table stroke or joystick operation can be carried out (door open or setup mode)
23	SECURITY Error X-axis
24	SECURITY Error Y- axis
25	SECURITY Error Z- axis
26	SECURITY Error A- axis
27	Emergency-STOP
28	Fault in the door switch safty circle (only with LS44/Solero)
29	Power stages are not switched on (only with LS44)
30	GAL security error (only with LS44)
31	While activating the joy-stick, if Move is still active.

SECURITY STATUS	
Instruction:	!?securitystatus
Parameters:	--
Note:	This instruction only exists for LS44-controller
Description:	?securitystaus => reading the current status of the security monitoring !securitystatus 0 => Clear reminder for testing
Feedback:	Bit 0 0000000000000000 Bit15 Bit 0-3 internal reminder Bit 4 X-axis standstill monitoring tested Bit 5 Y- axis standstill monitoring tested Bit 6 Z- axis standstill monitoring tested Bit 7 A- axis standstill monitoring tested Bit 8 X- axis speed monitoring tested Bit 9 Y- axis speed monitoring tested Bit 10 Z- axis speed monitoring tested Bit 11 A- axis speed monitoring tested Bit 12 Bit 13 Bit 14 condition setup mode (setup mode = 1) Bit 15 condition door (door „Open“ = 1)
Error code:	--
Example:	--

SECURITY ERROR	
Instruction:	?securityerror
Parameters:	--
Note:	This instruction only exists for LS44-controller
Description:	?securityerror => reading the current status and results of the GAL- security monitoring
Feedback:	<p>Bit 0 0000000000000000 Bit15</p> <p>Bit 0 : axis X axis standstill monitoring result (OK [1] / nicht OK [0])</p> <p>Bit 1 : axis Y axis standstill monitoring result</p> <p>Bit 2 : axis Z axis standstill monitoring result</p> <p>Bit 3 : axis A axis standstill monitoring result</p> <p>Bit 4 : axis X axis standstill monitoring test (getestet [1] / nicht getestet [0])</p> <p>Bit 5 : axis Y axis standstill monitoring test</p> <p>Bit 6 : axis Z axis standstill monitoring test</p> <p>Bit 7 : axis A axis standstill monitoring test</p> <p>Bit 8 : axis X speed monitoring result (OK [1] / not OK [0])</p> <p>Bit 9 : axis Y speed monitoring result</p> <p>Bit 10 : axis Z speed monitoring result</p> <p>Bit 11 : axis A speed monitoring result</p> <p>Bit 12 : axis X speed monitoring test (tested [1] / not tested [0])</p> <p>Bit 13 : axis Y speed monitoring test</p> <p>Bit 14 : axis Z speed monitoring test</p> <p>Bit 15 : axis A speed monitoring test</p>
Error code:	--
Example:	--

Output Function Level	
Instruction:	! ?opfl
Parameters:	X, y, z, or a 2,5-65 revolutions/second and also greater!
Note:	If the set speed is exceeded the current will be switched from parametric to maximum current.
Description:	!opfl x 25 => at the x-axis the current switch is done at 25 Rev/s. ?opfl => reading all speed limits.
Feedback:	speed in Rev/s
Error code:	--
Example:	?opfl y (read the speed limit of the y-axis)

Switch Level for Velocity	
Instruction:	! ?vlevel
Parameters:	1-7 and 0 – max. speed
Note:	<p>With this command it is possible to exclude speed areas in which the system tends to resonance.</p> <p>There are 3 speed areas and one limit that can be set with this command:</p> <p>Vlevel 1 = Lower limit of the first/lower area Vlevel 2 = Upper limit of the first/lower area Vlevel 3 = Lower limit of the second/third area Vlevel 4 = Upper limit of the second/third area Vlevel 5 = Lower limit of the third/upper area Vlevel 6 = Upper limit of the third/upper area Vlevel 7 = Up to this speed limit, the correction table is used</p> <p>These 3 speed areas are deactivated if the correction table is active. Applies to all axes!</p>
Description:	!vlevel 1 0.8 = Lower limit of the first/lower area !vlevel 2 1.2 = Upper limit of the first/lower area. ?vlevel 3 = Reading the speed limit of the second/lower area
Feedback:	Set speed
Error code:	--
Example:	!vlevel 7 10 (The correction table is valid up to a speed of 10 revolution/s)

Motor – Table Patch	
Instruction:	! ?mtpatch
Parameters:	0 or 1
Note:	With this command, the correction table is activated. The correction table was determined for Vextra Motore by measurement.
Description:	!mtpatch 1 = Activating the correction table ?mtpatch = Reading the current condition
Feedback:	0 or 1
Error code:	--
Example:	!mtpatch 0 (The correction table is not used.)

Joystick Filter	
Instruction:	! ?joyfilter
Parameters:	0 or 1
Note:	With this command, the filtration and Hysteresis in the Joystick-operation is activated.
Description:	!joyfilter 1 = Activating of the filtration ?joyfilter = Reading the current condition
Feedback:	0 or 1
Error code:	--
Example:	!joyfilter 0 (The filtration and Hysteresis is not used.)

4.7 Settings

The controller can be adapted to the mechanical components which are being used and to the desired requirements with the instructions described below.

Dimension	
Instruction:	!dim or ?dim
Parameters:	X, Y, Z and A 0, 1, 2, 3 or 4 The units for specification of lengths for input and output are:
	0 → Microsteps
	1 → μm
	2 → mm
	3 → 360°
	4 → Number of revolutions
Description:	
	!dim 4 → The dimensions for the X- and Y-axes are "Number of revolutions" and " μm ". 1
	!dim z → The dimension for the Z-axis is "mm". 2
	?dim → All dimensions are displayed.
	?dim a → The dimension of the A-axis is displayed.
Feedback:	Present setting
Error code:	
Example:	!dim 1 1 1 1 (all values in μm) ?dim

Note: The Spindle pitch should be set at 1 mm for dimensions 3 (degrees) and 4 (revolutions).

Spindle Pitch	
Instruction:	!pitch or ?pitch
Parameters:	X, Y, Z and A 0.001 – 68
Description:	!pitch 4.0 1.0 → Spindle pitches X = 4mm and Y = 1mm are programmed.
	!pitch z 1.0 → Spindle pitch Z = 1mm is programmed.
	?pitch → All spindle pitches are displayed.
	?pitch a → The spindle pitch of the A-axis is displayed.
Feedback:	Present spindle pitch
Error code:	--
Example:	!pitch 10 (spindle pitch X = 10mm) ?pitch

Gear	
Instruction:	!gear or ?gear
Parameters:	X, Y, Z and A 0.01 – 0.99 and 1-1000
Description:	!gear 4.0 1.0 → Gear transmissions 1/4 for X and 1/1 for Y are programmed.
	!gear z 10.0 → Gear transmissions 1/10 for Z is programmed.
	?gear → All gear transmissions are displayed.
	?gear a → The gear transmissions for the A-axis are displayed.
Feedback:	Present gear transmissions
Error code:	--
Example:	!gear 10 (gear transmission 1/10 for X) ?gear

Acceleration	
Instruction:	!accel or ?accel
Parameters:	X, Y, Z and A 0.01 – 20.00 [m/s ²]
Description:	!accel 1.00 1.50 → The accelerations (X=1.00, Y = 1.50 [m/s ²]) are set for the X- and Y- axes, the other axes remain unchanged.
	!accel x 1 → The acceleration for the X-axis is set to 1.00 [m/s ²].
	?accel → All preset accelerations are displayed.
	?accel z → The preset acceleration of the Z-axis is displayed.
Feedback:	Preset acceleration
Error code:	--
Example:	!accel 1.00 (set accelerations for X-axis to 1 m/s ²) ?accel

Speed (Velocity)	
Instruction:	!vel or ?vel
Parameters:	X, Y, Z and A 0 – maximum speed
Description:	!vel 1.0 15 → The speeds are described for axes X and Y (X=1.0, Y=15 [r/s]), the other axes remain unchanged.
	!vel z 0.1 → The speed for the Z-axis is set to 0.1 [r/s].
	?vel → All preset speeds are displayed.
	?vel x → Display of the preset speed of axis x.
Feedback:	Preset speed
Error code:	--
Example:	!vel 10 (The X-axis is run at max. 10 r/s) ?vel

The speed of rotation of the motors can be set in steps (St) of 0.01 r/sec. to 40 r/sec, or for the ECO-STEP, up to 15 r/sec. The top speed ranges can be reached only if the motors and the mechanical components are optimally synchronized on the LSTEP.

Value	Speed [r/sec]	Value	Speed [r/sec]	Value	Speed [r/sec]
0	0.01	2.0	2.0	12.0	12
0.1	0.1	3.0	3.0	13.0	13
0.2	0.2	9.0	9.0	15.0	15
0.9	0.9	10.0	10	20.0	20
1.0	1.0	11.0	11	40.0	40

Speed Reduction

Instruction:	!velfac ?velfac
Parameters:	X, Y, Z or A 0.01 to 1.00
Description:	!velfac x 0.1 => reduces the speed of the X-axis to 1/10 of the preset speed. ?velfac => gives the settings for all axes
Feedback:	A decimal value is returned (0.01 to 1.00)
Error code:	--
Example:	?velfac z (gives the setting for the Z-axis)

MaxCurrent (max. possible motor current)

Instruction:	?maxcur
Parameters:	X, Y, Z or A
Description:	?maxcur y => gives the maximum possible motor current for the Y-axis ?maxcur => gives the maximum possible motor current for all axes
Feedback:	Motor current in amps
Error code:	--
Example:	?maxcur

Output Current	
Instruction:	!cur or ?cur
Parameters:	X, Y, Z and A 0 – maximum current
Description:	!cur 1.0 2 → The output currents for the X- and Y-axes are set to X = 1A and Y = 2A, the other axes remain unchanged:
	!cur z 0.1 → The output current for the Z-axis is set at 0.1A.
	?cur → All preset output currents are displayed.
	?cur x → Display the preset output current for the X-axis.
Feedback:	Preset output current
Error code:	--
Example:	!cur 1.0 (The X-axis is run at maximum 1A) ?cur

Current Reduction	
Instruction:	!reduction or ?reduction
Parameters:	X, Y, Z and A 0 – 1.0
Description:	In quiescent state, the rated motor current is reduced to the parameterized ratio.
	!reduction 0.1 .7 → X-axis = 0.1*rated current and Y-axis = 0.7*rated current
	!reduction z 0.5 → Z-axis = 0.5*rated current
	?reduction → Display of the preset current reductions of all axes
	?reduction x → Display of the preset current reduction for the X-axis.
Feedback:	Preset current reduction
Error code:	--
Example:	!reduction 0.3 0.5 (X-and Y-axes are reduced) ?reduction

Delay Current Reduction	
Instruction:	!curdelay or ?curdelay
Parameters:	X, y, z and a 0 – 10000 (ms)
Description	<p>After moving a vector the motor current is maintained for the time that is set in curdelay. Afterwards it is reduced to the specified value of the current reduction.</p> <p>!curdelay 100 300 = x-axis = 100 ms delay and y-axis = 300 ms delay</p> <p>!curdelay z 450 = z-axis = 450 ms delay</p> <p>?curdelay = indication of the set current reduction of all axes</p> <p>?curdelay x = indication of the set current reduction of the x-axes</p>
Feedback:	set delay of the current reduction
Error code:	--
Example	!curdelay 100 300 (x- and y-axis are delayed) ?curdelay

Axis Enable	
Instruction:	!axis or ?axis
Parameters:	X, Y,Z and A 0 and 1
Description:	!axis 1 0 1 0 → The X- and Z-axes are enabled, the Y- and A-axes are not enabled.
	!axis y 1 → Y-axis enabled
	?axis → Show status of all axes
	?axis a → Show status of axis A
Feedback:	Present operating status
Error code:	--
Example:	!axis 1 1 1 1 (enable all axes) ?axis x (read status of X-axis)



Axis direction	
Instruction:	!axisdir
Parameter:	X, y, z and a 0 or 1
Note:	With axisdir the motor – turning direction can be turned, the corresponding limit switches are turned also.
Description:	!axis 0 1 0 1 => the turning direction of the axis y and a are turned. ?axisdir x = indication, if the turning direction of the x-axis is activated.
Feedback::	0 = no change turning direction 1 =
Error code::	--
Example::	!axisdir 0 0 0 0 (Cancel all changes of turning direction)

Limit	
Instruction:	!lim or ?lim
Parameter:	x, y, z or a +- maximale Verfahrbereich
Note:	The values must be given in pairs. The in- and output values are dependet on the dimension.
Description:	!lim -1000 1000 -2000 2000 → Moving range limits are assigned to x- and y-axis
	!lim z -500 1700 → Moving range limits are assigned to z-axis.
	?lim → Read moving range limits of all axis.
	?lim a → represent moving range limits a-axis
Feedback:	Current moving range
Error code:	--
Example:	!lim 10 (program only Lower limitx-axis) ?lim

Limit Control	
Instruction:	!limctr or ?limctr
Parameters:	X, Y, Z or A 0 or 1
Description:	!limctr 1 1 1 → Limit control active for the X-, Y- and Z-axes.
	!limctr z 1 → Limit control active for the Z-axis.
	?limctr a → Limit control active for the A-axis?
	?limctr Display of the status of the individual limit controls.
Feedback:	0 = Limit control not active 1 = Limit control active
Error code:	--
Example:	! limctr y 0 (Deactivate Y-axis limit control ? limctr

Statuslimit	
Instruction:	?statuslimit or statuslimit
Parameters:	--
Description:	Statuslimit delivers the current condition of the software-limits of each single axis.
Feedback:	A = Axis was calibratet
	D = table stroke was measured
	L = Software - Limit was set
	- = Software - Limit was not changed
	The sequence of the acknowledgement is for example: AA-A--DD-LL-L--L
	X,y and a = calibratet
	Z and a = measure table stroke
	Y and z = min. software limit set
	X and a = max. software limit set
Error code:	--
Example:	?statuslimit

NoSetLimit	
Instruction:	!nosetlimit
Parameters:	x, y, z or a 0 or 1
Note:	When calibrating and measuring table stroke normally the internal software – limits are set, this can be avoided herewith
Description:	<p>nosetlimit 1 1 1 => No moving range limits are set for the axis x, y and z.</p> <p>!nosetlimit y 1 => No moving range limit is set for the y-axis.</p> <p>?nosetlimit = Read settings of all axis</p> <p>?nosetlimit a = Read settings of axis a</p>
Feedback:	0 = Software – Limits will be set (calib/rm)
Error code:	--
Example:	?nosetlimit

Limit Switch Polarity	
Instruction:	!swpol or ?swpol
Parameters:	X, Y, Z or A <div> <div>0</div>  <div>or</div> <div>1</div>  </div>
Description:	<p>!swpol 1 0 1 → Assign polarity of the limit switches for all axes. (Order: E0 REF EE).</p> <p>!swpol z 1 0 1 → Assign polarity of the limit switch for the Z-axis. (Order: E0 REF EE)</p> <p>?swpol a → Show polarity of the limit switch for axis A.</p>
Feedback:	Polarity of the limit switches
Error code:	--
Example:	<p>!swpol y 1 1 1 (All Y-axis switches react to the positive edge)</p> <p>?swpol x</p>

Limit Switch On/Off	
Instruction:	!swact or ?swact
Parameters:	X, Y, Z or A 0 or 1
Description:	!swact 1 0 1 → Limit switches for all axes : E0=On REF=Off EE=On
	!swact z 1 0 1 → Z-axis limit switch: E0=On REF=Off EE=On
	?swact a → Show status of the A-axis limit switches.
Feedback:	Status of the limit switches
Error code:	--
Example:	!swact y 1 1 1 (All Y-axis switches active) ?swact x

Read Limit Switches													
Instruction:	?readsw												
Parameters:													
Description:	?readsw → Read status of all limit switches.												
Feedback:	Status of the limit switches.												
	Axis:	x	y	z	a	x	y	z	a	x	y	z	a
	Switch:	E0	E0	E0	E0	Ref	Ref	Ref	Ref	EE	EE	EE	EE
	E0 = Zero limit switch				Ref = Reference limit switch					EE = End limit switch			
Error code:	--												
Example:	?readsw (Read all limit switches)												

Stop input set polarity	
Instruction:	!stoppol or ?stoppol
Parameters:	0 = low active 1 = high active
Description:	Because the stop input has a Pull Up after 5 min, low active has to be set for the closer (make contact) and high active for the opener (break).
Feedback:	--
Error code:	--
Example:	!stoppol 1 (the stop input is high active)

Stop acceleration for the emergency stop	
Instruction:	!stopaccel or ?stopaccel
Parameters:	0,01 bis 20 m/s ²
Description:	During activating of the stop inputs the acceleration speed which is set with "accel" will be used for stopping, if "stopaccel" is not set. When "stopaccel" is set, this acceleration is valid unless the value in "accel" is higher. This value will not be saved with Save. The Position is not lost (if the acceleration was selected correctly) After the release of the Stop input, calibration is not necessary .
Feedback:	--
Error code:	--
Example:	!stopaccel 2 (it will be stopped with 2m/s ²)
Attention! Note:	stopaccel is only valid for vector operating not for: Joystick, calibrating and stroke measuring

4.8 Determination Of The Mechanical Work Range

After initializing the controller, the instructions calibrate "cal" and measure stroke "rm" should be performed. This will determine the maximum mechanical work range. This ensures that the axes cannot be moved into the limit switches.

The work stroke can only be measured when all axes have a zero and an end limit switch. So that the limit switches respond when the zero or the end position is reached if the mechanical components overshoot them, the work range can be limited with the instructions "caliboffset" and "rmoffset".

Calibrate	
Instruction:	!cal or cal
Parameters:	X, Y, Z or A
Description:	Cal → Moves all enabled axes towards lower positional values. Travel is stopped as soon as the limit switches have been tripped and is then resumed slowly in the opposite direction until the switch is no longer active. The positional value is set to 0. The position is taken over as a software limit, as described in the instruction "Limit".
	Cal y → As above, however for the Y-axis.
Feedback:	An 'A' for each calibrated axis or 'E' if fault occurs
Error code:	--
Example:	!cal

Measure Table Stroke	
Instruction:	!rm or rm
Parameters:	X, Y, Z or A
Description:	Rm → Moves all enabled axes towards greater positional values. The travel is stopped as soon as the limit switch has been tripped and is then resumed slowly in the opposite direction until the switch is no longer active. The positional value is saved and is taken over as the software limit, as described in the instruction "Limit".
	Rm z → As above, however for the Z-axis only
Feedback:	A ,D' for every axis
Error code:	--
Example:	!rm

RM Offset	
Instruction:	!rmoffset or ?rmoffset
Parameters:	X, Y, Z or A 0 – 32*50000 (32*spindle pitch)
Description:	!rmoffset 1 1 1 → The X-, Y-, and Z-axes are each moved 1mm (for Dim. 2 2 2) away from the limit switch towards the center of the table when the table stroke is measured and the software limit is then set.
	?rmoffset y → Read present offset of the Y-axis.
Feedback:	Distance
Error code:	--
Example:	?rmoffset

Calibration Offset	
Instruction:	!caliboffset or ?caliboffset
Parameters:	X, Y, Z or A 0 – 32*50000 (32*spindle pitch)
Description:	!caliboffset 1 1 1 → The X-, Y-, and Z-axes are each moved 1 mm (for Dim 2 2 2) away from the zero limit switch towards the center of the table when calibration is done and the zero position is then set (software limit).
	?caliboffset y → Read present offset of the Y-axis
Feedback:	Distance
Error code:	--
Example:	?caliboffset

Calibration Direction	
Instruction:	!caldir
Parameters:	X, y, z or a 0 or 1
Note:	When calibrating in positive direction, the positive software limit is set.
Description:	!caldir 0 0 1 => The axes X, Y are calibrated in negative direction and the Z-axis in positive direction. ?caldir => read current direction for calibrating.
Feedback:	0 = negative direction 1 = positive direction
Error code:	--
Example:	!caldir y 1 (The Y-axis will be calibrated in positive direction)
Attention!	This command works only with controls without measuring systems

Calibration Position	
Instruction:	!calpos (only in connection with a measuring system)
Parameters:	X, y, z or a position value
Note:	The position where the limit switch was left, is saved for each axis when calibrating.
Description:	!calpos 0 0 0 => Set the position for X-, Y- and Z-axis to 0. ?calpos => read current position
Feedback:	In range of the encoder
Error code:	--
Example:	?calpos y (The position of the Y-axis)

Calibration Backspeed	
Instruction:	!/?calbspeed
Parameters:	value range 5 bis 100
Note:	The speed equals the entered value *0.01 U/s.
Description:	calbspeed sets resp. reads the revolution speed, the axes drive after reaching the limit switch. The entered value must be 0.01 U/s multiplied.
Feedback:	--
Error code:	--
Example:	!calbspeed 10 => After reaching the limit switches when calibrating they are left with 0.1 R/s. /?calbspeed => Read current setting (given value *0.01 U/s).

Calibration Refspeed	
Instruction:	!/?calrefspeed
Parameters:	Value range 0 - 100
Note:	The basic setting = 32 This value will not be saved with Save.
Description:	This setting changes the speed for finding the reference mark
Feedback:	--
Error code:	--
Example:	!calrefspeed 5

Direction for Reference	
Instruction:	!refdir (applies only to LSTEP)
Parameters:	X, y, z or a 0 or 1
Note:	In the basic condition the reference direction is negative if no switch was activated this can be changed with „!refdir“.
Description:	!refdir 0 0 1 => The axes X, Y will be referenced in negative direction and the Z-axis in positive direction. ?refdir => Read current position for referencing
Feedback:	0 = negative direction 1 = positive direction
Error code:	--
Example:	!refdir Y (The Y-axis will be referenced in positive direction)

Reference	
Instruction:	!ref or ref (applies only to LSTEP)
Parameters:	X, y, z or a
Note:	In the basic condition the reference direction is negative if no switch was activated this can be changed with „!refdir“.
Description:	ref = Moves all released axes in the direction indicted via “refdir” The movement will be interrupted as soon as a reference switch is reached. The position value is not set. ref y = Like above but y-axis.
Feedback:	For each referenced axis a ,R’
Error code:	--
Example:	!ref

4.9 Travel Instructions And Their Control Functions

Linear interpolation takes place for all positioning instructions, i.e. all axes reach the specified position at the same time. The axis where the motor must traverse the most revolutions is deemed the lead axis and thus travels at the preset speed and acceleration. The x-axis is the lead axis, if all axes have to travel the same stretch. In this case all axes exceed the preset speed and acceleration.

If the axes have totally different dynamic behavior, they should be started individually. Also an asynchronous movement is possible. Herewith is to be noted that in the setting auto status 1 the acknowledgement first comes, if all axes came to standstill. If you want to start another axis while one is already moving the auto status = 0 and poll with ?statusaxis.

Position absolut	
Instruction:	!moa or moa
Parameters:	X, Y, Z or A +- Range of travel
Note:	The input depends on the dimension.
Description:	Moa 10 0 20 → The X-, Y- and Z-axes are positioned at the positional values which were input.
	moa y 333 → As above, however Y-axis only.
Feedback:	A ,@' for every positioned axis
Error code:	--
Example:	Moa x 10 (The X-axis is positioned at the position which was input)

Relative Position	
Instruction:	!mor or mor
Parameters:	X, Y, Z or A +- Range of travel
Note:	The input depends on the dimension.
Description:	Mor 100 0 39 → The X- and Z-axes are travelled the distances which were input.
	Mor a 298 → The A-axis is travelled the distance which was input.
Feedback:	A ,@' for every axis which is travelled
Error code:	--
Example:	!mor 0 0 0 100 (Only the A-axis is travelled)

X Y Compensation	
Instruction:	!xycomp
Parameters:	0 bis 6
Note:	0 = No compensation 1 = „X = X+Y“ 2 = „Y = X+Y“ 3 = „X = X-Y and Y = X+Y“ 4 = „X = X+Y and Y = X-Y“ 5 = „X = X-Y“ 6 = „Y = X-Y“
Description:	xycomp 1 => The axes X, Y are manipulated after above-mentioned formula. ?xycomp => Read current condition
Feedback:	Type of the compensation
Error code:	--
Example:	?xycomp (Would read current condition of the compensation)

Position relative (short command)	
Instruction:	!m or m
Parameters:	
Note:	This instruction is used when the same distance is to be travelled again and again at short intervals. The distance to be travelled must first be set with !distance or more instructions. The position is not updated, until after the next Move-command.
Description:	m → Start travel of all enabled axes.
Feedback:	Depends on the autostatus setting.
Error code:	--
Example:	!mor 0 0 0 100 (Only the A-axis is travelled) m (A-axis is travelled by 100 again)

Distance	
Instruction:	!distance or ?distance
Parameters:	X,Y,Z and A Min-/max- travel range
Note:	Input and output depend on the dimension.
Description:	!distance 1 2 3 → The distances for the X-, Y-, and Z-axes are set.
	!distance y 20 → The Y-axis distance is set.
	?distance → Inquire present distances for all axes.
	?distance z → Inquire present distance for Z-axis.
Feedback:	Distances
Error code:	--
Example:	!distance 10 20 (Set X- and Y-axis distances) ?distance (Inquire distances of all axes)

SpeedPoti	
Instruction:	!pot or ?pot
Parameters:	0 or 1
Note:	
Description:	0 → Travelling is done at the preset speed (vel).
	1 → Travelling is done at a percentage of the preset speed (vel), depending on the setting of the potentiometer.
Feedback:	--
Error code:	--
Example:	!pot 1 ?pot

Position	
Instruction:	!pos or ?pos
Parameters:	X,Y,Z and A Min-/max range of travel
Note:	Input and output depend on the dimension.
Description:	!pos 1000 2000 3000 → The positional values for the X-, Y- and Z-axes are set.
	!pos y 2000 → The position of the Y-axis is set.
	?pos → Inquire present position of all axes.
	?pos z → Inquire present position of Z-axis.
Feedback:	Positional values
Error code:	--
Example:	!pos100 200 (Set positions of X- and Y-axes) ?pos (Inquire positions of all axes)

Clear Position	
Instruction:	Clearpos
Parameters:	X,y,z and a
Note:	<p>This command sets the position to Ø, also the internal counter (is not the same function as setting position with !pos x Ø).</p> <p>This function is needed for endless axes, because the control can process only ± 1000 motor revolutions of the value range.</p> <p>In recognized encoders, the function is not carried out for the respective axis.</p>
Description:	clearpos => All position values become are set to zero
	clearpos y => Position of the y-axis is set to zero.
Feedback:	No
Error code:	--
Example:	clearpos x (Position der x-Axis wird genullt)

Central Positioning	
Instruction:	!moc or moc
Parameters:	X, y, z and a
Note:	All axes are centered. It makes sense to calibrate and measure the table stroke previously!
Description:	Moc a => The A-axis is centered.
Feedback:	For each positioned axis a ,@'
Error code:	--
Example:	Moc (All axes are centered)

Delay	
Instruction:	?delay or !delay
Parameters:	0 - 10000 (ms)
Description:	The delay instruction can be used to delay the vector start.
Feedback:	Decimal value
Error code:	--
Example:	!delay 1000 (1s delay) ?delay

Abort	
Instruction:	!a or a
Parameters:	None
Description:	All travels are stopped.
Feedback:	A @ for every axis
Error code:	--
Example:	!a

4.10 Joystick- Handwheel- and Trackball Instructions

Note:	<p>If the joystick switch on the controller is set at “manual”, all axes can be moved right up to the end limit positions using the joystick. The position is thereby also counted.</p> <p>Instructions for setting the controller are possible in this mode of operation, “Move” instructions however are not.</p> <p>If an inquiry is made with the instruction “Statusaxis”, the controller gives the reply “J J J”. In an inquiry with the instruction „?joy“ the controller gives the reply „M”.</p>
--------------	---

Digital Joystick (Speed)	
Instruction:	!speed or ?speed
Parameters:	X, Y, Z or A +- Maximum speed (vel)
Description:	The individual axes can be travelled at a constant speed with this instruction.
	!speed 0 → All axes at speed 0 and joystick mode “OFF”
	speed 10 → All axes at speed 10.0 [r/s] and joystick mode “ON”.
	!speed 10 10 0 10 → X-, Y-, and A-axes at speed 10.0 [r/s], Z-axis speed 0 and joystick mode “ON”.
	!speed y 25 → Y-axis speed 25 and joystick mode “ON”.
	!speed y -25 → Axis y speed 25 in negative direction joystick-operation „ON,,.
	?speed → Read the preset speeds.
	?speed y → Read the preset speed of the Y-axis.
Feedback:	Present speeds
Error code:	--
Example:	!speed 33 11 (X-axis speed 33.0 [r/s], Y-axis speed 11.0 [r/s] and joystick mode “ON”) ?speed
Note:	In order to position absolute or relative after carrying out the speed command, the digital joystick must be switched off with !speed 0 and the speed needs to be set new.

Joystick Direction + disable joystick	
Instruction:	!joydir or ?joydir
Parameters:	0, ±1, ±2 X, Y, Z, and A
Description:	When joydir is input, the direction of rotation of the motors is changed when the joystick is moved and the axes are disabled or enabled.
	!joydir -1 -1 1 1 = Negative direction of rotation for the X-and Y-axes, positive direction of rotation for the A-axis
	!joydir -2 -2 2 2 = Like in above-mentioned example, but if the axes have not moved longer than 1s, they will be switched to the current reduced mode.
	!joydir z 0 = Z-axis is disabled.
	Peculiarity: Because only a 3-axes joystick is planned, the third joystick-axis controls the z- and a-axis.
Feedback:	Preset directions or status.
Error code:	--
Example:	!joydir-1 (negative preceding sign for X-axis) ?joydir

Joystick	
Instruction:	!joy or ?joy
Parameters:	0, 1, 2, 3, 4 and 5
Note:	The joystick switch must be set at Automatic
Description:	!joy 0 → Joystick "OFF"
	!joy 1 → Joystick "ON" without position count (tracking)
	!joy 2 → Joystick "ON" with position count
	!joy 3 → Joystick "ON" with position count and periodic position feedback.
	!joy 4 → Joystick "ON" with position count (encoder values, if any)
	!joy 5 → Joystick "ON" with position count and periodic position feedback (encoder values, if any).
	?joy → Present status
Feedback:	Present position or present status of joystick operation. If the joy-stick is switched off for each axis comes as a return message @ if Autostatus =1
Error code:	--
Example:	!joy 2 (Joystick "ON" with position count) ?joy (Inquire present status)

Joystick Speed	
Instruction:	!joyspeed or ?joyspeed
Parameters:	X, Y, Z or A 0, 1 or 2 +- Maximum speed (vel)
Note:	For additional control panel
Description:	!joyspeed 0 25 → Parameter 0 described at speed 25 .
	?joyspeed 1 → Read preset speed of parameter 1.
Feedback:	Currently preset speeds
Error code:	--
Example:	!joyspeed 2 11 (Parameter 2 described at speed 11 .) ?joyspeed 2

Joystick Window (joywindow)	
Instruction:	!joywindow
Parameters:	0 – 100
Description:	<p>With joywindow the analog range is determined in which the axes move. Applies to all axes.</p> <p>joywindow 10 the axes move only, if the excursion of the joystick is greater than 10 (points).</p> <p>?joywindow => Read out the joystick - window .</p> <p><u>Example:</u> zero position joystick = 512 (analog value) Joywindow = 10 i.e., that the axes move within the values < 502 and > 522.</p>
Feedback:	Preset window
Error code:	--
Example:	?joywindow (reading window size)

Joystick-allocation of axes	
Instruction:	!joychangeaxis or ?joychangeaxis
Parameters:	0, 1
Description:	<p>!joychangeaxis 0 → Changes the allocation of the AD-Joystick channels (conventional evaluation of Joystick)</p> <p>!joychangeaxis 1 → Changes the allocation of the AD-Joystick channels (Changes of allocation of X and Y axes)</p>
Feedback:	0 or 1
Error code:	--
Example:	! joychangeaxis 1 (Exchange of allocation of X and Y axes) ? joychangeaxis

Configuration Joystick On/Off	
Instruction:	!savejoyonoff
Parameters:	0 = Joystick is switched OFF after switching ON the PCs 1 = Joystick is switched ON after switching ON the PCs
Note:	This command only exists for the LSTEP-PCI
Description:	?savejoyonoff => reading the current state !savejoyonoff 1 => the joystick is active after a subsequent save command and reset. of the control or (new start)
Feedback:	0 or 1
Error code:	--
Example:	!savejoyonoff 1 !save (Setting is burned into the flash) !reset (re start)

Handwheel reporting:

The movements of the table reacts dynamic to the turns of the handwheel. Slow turns are operated in micro steps, and fast turns with a speed profile. The the max. speed and the acceleration for the handwheel operation can be set with the commands hwvel und hwaccel.

Handwheel	
Instruction:	!hw or ?hw
Parameters:	0, 1, 2, 3, 4 and 5
Note:	Alternative to joystick a handwheel can be connected.
Description:	!hw 0 → Handwheel „OFF,,
	!hw 1 → Handwheel „ON,, without position count
	!hw 2 → Handwheel „ON,, with position count
	!hw 3 → Handwheel „ON,, with position count and periodic position acknowledgement.
	!hw 4 → Handwheel „ON,, with position count (encoder value, if exists).
	!hw 5 → Handwheel „ON,, with position count and periodic position acknowledgement (encoder value, if exists).
	?hw → current condition
Feedback:	Current position or current status of the hand wheel operation.
Error code:	--
Example:	!hw 2 (Handwheel „ON,, with position count) ?hw (Inquiry of the current condition)

Handwheel Speed	
Instruction:	!hwvel or ?hwvel
Parameters:	X and Y 0.0001 to 40.0000 U/s
Note:	This command only exists in connection with a handwheel.
Description:	!hwvel 1 1 The max. reachable speed for X + Y is 1U/s
Feedback:	Value of set speed.
Error code:	--
Example:	!hwvel 0.5 0.5 The axis X and Y drive with max. 0,5 U/s ?hwvel

Handwheel acceleration	
Instruction:	!hwaccel or ?hwaccel
Parameters:	X and Y 0 – max. speed
Note:	This command only exists in connection with a handwheel.
Description:	!hwaccel 0.5 0.5 Acceleration for X + Y is 0.5m/s ²
Feedback:	Value of the set acceleration.
Error code:	--
Example:	!hwaccel 1 1 The acceleration for X + Y is 1m/s ² ?hwaccel

Control panel	
Instruction:	!bpz or ?bpz
Parameters:	0,1 or 2
Note:	For an additional control panel with trackball
Description:	!bpz 0 => Control panel „OFF“
	!bpz 1 => activate control panel and operate trackball with 0,1μ step resolution, Joyspeed active.
	!bpz 2 => operate activate control panel and trackball with factor, Joyspeed active.
	!bpz 3 => activate control panel and operate track ball with 0,1μ step resolution, function keys active
	!bpz 4 => activate control panel and operate track ball with factor, function keys
	?bpz => read preset status
Feedback:	current status
Error code:	--
Example:	!bpz 1 (activate control panel and operate trackball with 0,1μ step resolution)

Control panel Trackball Factor	
Instruction:	!bpztf or ?bpztf
Parameters:	0,01 to 100,00
Note:	For an additional control panel
Description:	!bpztf 1 => Trackball - Factor = 1, i.e. A trackball-impuls = motor-increment.
	?bpztf => Reading the preset factors
Feedback:	current factor
Error code:	--
Example:	?bpztf => Reading the preset factors

Control panel Trackball Back Lash	
Instruction:	!bpzbl or ?bpzbl
Parameters:	0,0001 to 0,015 mm
Note:	Reading the preset factors
Description:	!bpzbl 0.01 0.005 => Backlash of x-axis = 10µm and y-axis = 5µm.
	!bpzbl z 0.001 => Backlash of z-axis = 1µm.
	?bpzbl => Readout set reverse backlash
Feedback:	Current reverse backlash
Error code:	--
Example:	?bpzbl => Readout Auslesen des eingestellten Lose

4.11 In/Outputs *(not withi ECO-STEP)*

The LSTEP may be equipped with an additional module which makes 16 digital inputs and outputs and two analog outputs available. An additional card for the LSTEP-PCI has only 16 switch In/Outputs.

To use these inputs and outputs, you must order the appropriate LSTEP model. At the Multi functioning port (chapter 5.1) analog In/Outputs are also available.

Digital Input	
Instruction:	?digin
Parameters:	0 through 15
Description:	?digin → Read all input pins
	?digin 8 → Read input pin 8
Feedback:	Status of the input pins
Error code:	--
Example:	?digin (Read all input pins)

Digital Output	
Instruction:	!digout or ?digout
Parameters:	0 through 15
Description:	!digout 11110000 → Output pins 0,1,2,3 are set to "1" and output pins 4,5,6,7 are set to "0".
	!digout 5 1 → Output pin 5 is set to "1".
	?digout → Read the current status of all output pins.
	?digout 8 → Read the current status of output pin 8
Feedback:	Status of the output pins
Error code:	--
Example:	!digout 7 0 (Set output pin 7 to "0") ?digout (Read all output pins)

Function Of The Digital Inputs/Outputs	
Instruction:	!digfkt or ?digfkt
Parameters:	0 through 15 (input/output), 16 (all 16 port pins) 0 through 4 (function) 0 through 100 mm (distance) or polarity of the inputs X, Y, Z and A (axes)
Description:	Functions:
	0 → switch off function
	1 → Allocation of the emergency stop pins.
	2 → Activation of an output depending on the distance set before reaching the target position.
	3 → Activation of an output depending on the distance set after the starting position.
	4 → 2&3
	Instructions:
	!digfkt 7 2 78.9 z → Output 7 is activated 78.9mm before the target position is reached.
	!digfkt 14 1 → Input 14 is used as the emergency stop.
	!digfkt 16 0 → All functions are set to 0.
	!digfkt 16 0 0 → All inputs high aktiv
	!digfkt 16 0 1 → All inputs low aktiv
	!digfkt 5 0 0 → Input 5 high aktiv
	?digfkt 16 or ?digfkt → The current function statuses of all inputs and outputs are displayed.
	?digfkt 6 The current function statuses of input 6 and output 6 are displayed.
	?digfkt 7 4 The relevant distance and axis allocation are displayed.
Feedback:	All settings
Error code:	--
Example:	!digfkt 7 0 (Set the function of input and output 7 to "0") ?digfkt 9 (Read the function of input and output 9)

EXTENDED DIGITAL INPUT	
Instruction:	?edigin
Parameters:	0 to 15
Note:	This instruction only applies to the LS44-controller.
Description:	?edigin = Read the current status of all additional input pins ?edigin 8 = Read the current status of all additional input pin 8
Feedback:	Status of the additional input pin
Error code:	--
Example:	?edigin (Read all additional inputs pins)

EXTENDED DIGITAL OUTPUT	
Instruction:	!edigout oder ?edigout
Parameters:	0 bis 15
Note:	This instruction only applies to the LS44-controller
Description:	!edigout 11110000 = All additional output pins 0,1,2,3 auf „1“ and 4,5,6,7 are set to „0“ . !edigout 5 1 = An additional output pin 5 is set to „1“ . ?edigout = Read the current status of all additional output pins ?edigout 8 = Read the current status of all additional output pin 8
Feedback:	Status of the additional output pin
Error code:	--
Example:	!edigout 7 0 (Set the additional output pin 7 to „0“) ?edigout (Read all additional output pins)

FUNCTION of the additional In and Outputs	
Instruction:	!edigfkt or ?edigfkt
Parameters:	0 to 15 (Input/Output), 16 (all 16 portpins) 0 (Function)
Note:	This instruction only applies to the LS44-controller
Description:	<p>Function:</p> <p>0 = no influence of the In-/Outputs and setting of the polarity (0 = High-, 1 = Low-Active)</p> <p>!edigfkt 16 0 = All functions are set to 0.</p> <p>?edigfkt 16 oder ?edigfkt = The current function status of all additional In- and Outputs are displayed</p> <p>?edigfkt 6 = The current function status of the additional In- and Output 6 is displayed</p>
Feedback:	All settings
Error code:	--
Example:	<p>!edigfkt 7 0 (Set the function of the additional In- and Output 7 to „0“)</p> <p>?digfkt 9 (Read the function of the additional In- and Output 9)</p>

There are also two analog outputs on the module. The outputs are standardly designed for 0...5V . Other output voltage ranges (e.g. +/- 5V, +/- 10V, 0...10V,...) are possible on request. The outputs can be loaded with +/- 5mA . The internal resistance is approx. 100 Ohm.

Analog Output	
Instruction:	!anaout or ?anaout
Parameters:	0 through 100 % 0, 1 and 2 (analog channels) c (c = channel)
Note:	Channels 0 and 1 are on the additional board (D/A-converter) Channel 2 is on the LS2000 board.
Description:	!anaout 100 50 → The first analog channel is set to 100% (full power.) and the second to 50% (half power).
	!anaout c 1 25 → Analog channel 1 is set to 25%.
	?anaout → Read the current status of all analog channels.
	?anaout c 2 → Read the current status of analog channel 2.
Feedback:	Status of the modulation in percent of the analog channels.
Error code:	--
Example:	!anaout c 1 0 (Set analog channel 1 to "0") ?anaout (Read all analog channels)

ANALOG INPUT	
Instruction:	?anain
Parameters:	0 bis 9 (analog channel) c (c = channel)
Description:	?anain c 2 => Read the current status of analog channel 2
Feedback:	Status depending on the analog channel
Error code:	--
Example:	?anain c 2 (Read the current status of analog channel 2)

Channel						
0	MFP	Pin 24	Joystick X			
1	MFP	Pin 12	Joystick Y			
2	MFP	Pin 25	Joystick Z			
3	MFP	Pin 26	ST11 (not on 25pol DSub)			
4	Speedpoti for LSTEP with display					
5	Motor current for the LSTEP-PCI					
6	MFP	Pin 8	or	LSTEP-PCI	St10	Pin 1
7	MFP	Pin 20	or	LSTEP-PCI	St10	Pin 2
8	MFP	Pin 7	or	LSTEP-PCI	St10	Pin 3
9	MFP	Pin 19	or	LSTEP-PCI	St10	Pin 4
10	MFP	Pin 6	or	LSTEP-PCI	St10	Pin 6/7

4.12 Interpretation Of Incremental Measuring Systems *(not for ECO-STEP)*

Axes with and without encoders can be operated simultaneously on the controller. During the calibration operation, the controller checks whether encoders are connected, provided that they have been enabled via *encmask*. To instruction *?enc* can be used to see the results of these checks. The controller does not however distinguish between incorrectly connected encoders and missing encoders.

Also while calibration to reference mark, the axis drives in negative direction into the zero-proximity switch, does a inversion of direction and drives with the speed which was set via *calrefspeed* to the reference mark. If a system does not have a proximity switch (i.e. a turn axis) it moves directly to the reference mark, if all proximity switches were deactivated prior.

Starting firmware version „T03.19.06-2001“ only the Sin.- Cos.- Signals need to be connected in there count direction towards motor count direction. An alignment towards the encoder reference mark is no necessary anymore.

Encoder Mask For Encoders	
Instruction:	!encmask or ?encmask
Parameters:	X, Y, Z and A 0,1 (On,Off)
Note:	Enabling of the individual encoders.
Description:	!encmask 1 0 1 → X- and Z-encoders are active, Y-encoder deactivated.
	?encmask → The encoder mask for all encoders is displayed.
	?encmask x → Display the encoder mask for the X-axis.
Feedback:	Encoder mask
Error code:	--
Example:	!encmask 1 0 (X-axis encoder enabled, Y-axis encoder not enabled) ?encmask

Encoder Mask For Detected Encoders	
Instruction:	?enc
Parameters:	X, Y, Z and A 0 or 1 (On,Off)
Note:	If encoders are activated which are not available, malfunctions may occur.
Description:	!enc 1 0 1 → X- and Z-encoder active, Y-encoder deactivated.
	?enc → All encoder statuses are displayed.
	?enc x → Display of the encoder mask for the X-axis.
Feedback:	Encoder status
Error code:	--
Example:	!enc 1 0 (X-axis encoder active, Y-axis encoder deactivated) ?enc

Signal Periods / Linear Encoder	
Instruction:	!encperiod or ?encperiod
Parameters:	X, Y, Z and A 0.0001 – Spindle pitch * 0.8 (mm)
Description:	!encperiod 0.5 0.020 → Length of the the encoder signal period is 500µm for the X-axis and 20µm for the Y-axis.
	?encperiod → All encoder period lengths are displayed.
	?encperiod x → Display of the length of the encoder period length the X-axis.
Feedback:	Length of encoder period in mm
Error code:	--
Example:	!encperiod 0.1 (Length of encoder period for the X-axis is 0.1mm) ?encperiod

Encoder Resolution/ Rotary Encoder	
Command: Instruction	!?encres
Parameter: Parameters	1 to 40000
Note:	Shows the amount of encoder signal periods per motor revolution. The ratio of the periods to the unit factor should result to a whole number (integer), if the encoder is mounted behind a gear.
Description:	?encres !encres 250 500 1000
Feedback:	-
Error code:	--
Example:	!encres 500 500 500 For the axes X,Y,Z, 500 signal periods per motor revolution are send to the control.

Encoder Reference Signal	
Instruction:	!encref or ?encref
Parameters:	X, Y, Z or A 0 or 1
Description:	!encref 1 1 0 → The reference signal of the X-and Y-axis encoders is interpreted when calibration is done.
	!encref z 1 → The reference signal of the Z-axis encoder is interpreted when calibration is done.
	?encref → The present setting is displayed.
	?encref y → The present setting for the Y-axis is displayed.
Feedback:	0 or 1
Error code:	--
Example:	!encref 0 (No reference signal interpretation for the X-axis) ?encref

Encoder Position	
Instruction:	!encpos or ?encpos
Parameters:	
Description:	!encpos 1 → When the position is inquired, the encoder positions of the detected encoders are displayed.
	?encpos → The present setting is displayed.
Feedback:	0 or 1
Error code:	--
Example:	!encpos 0 (Encoder position display "OFF") ?encpos

Encoder Error	
Instruction:	!encerr or ?encerr
Parameters:	X, Y, Z, A 0
Description:	!encerr 0 0 0 → Clear encoder error messages from X-, Y-and Z-axes.
	!encerr a 0 → Clear encoder error message from A-axis.
	?encerr → The present encoder error messages for all axes are displayed.
	?encerr z → The present encoder error message for the Z-axis is displayed.
Feedback:	0 or e
Error code:	--
Example:	!encerr 0 (Clear encoder error message from A-axis) ?encerr

Geber – Position PCI (EncoderReadPositionPCI)	
Instruction:	?hwcount
Parameters:	X, y, z, a
Description:	?hwcount => Read all encouder positions ?hwcount a => Read encouder position of a-axis
Feedback:	Counter value 4-times interpolate
Error code:	--
Example:	?hwcount x (Read encouder position of a-axis)

Encoder – Position PCI (EncoderClear-PositionPCI)	
Instruction:	!clearhwcount
Parameters:	X, y, z, a
Description:	!clearhwcount => Clear all encoder – counter. !clearhwcount a => Clear encoder – counter of the A-axis
Feedback:	--
Error code:	--
Example:	!clearhwcount x (Clear encoder – counter of the A-axis)

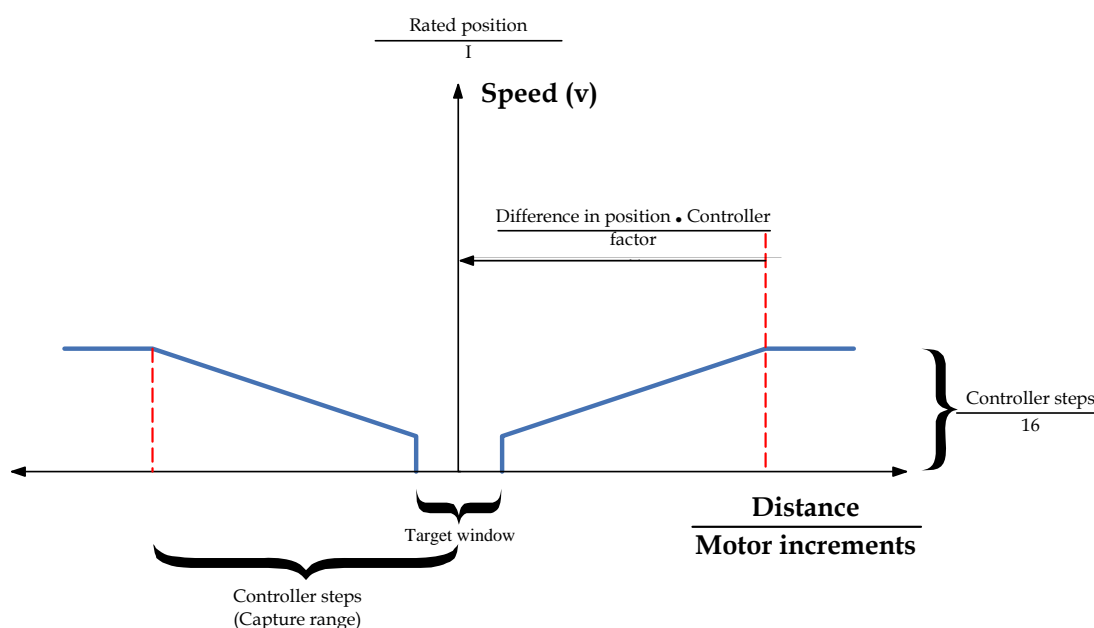
4.13 Controller Settings For LSTEP (not for ECO-STEP)

The control behaviour in closed loop operation can be influenced with the help of various parameters.

These parameters are:

1. Ctrc (Controller call)
2. Ctrs (Controller steps / capture range)
3. Ctrf (Controller factor)
4. Ctrd (Controller delay)
5. Ctrt (Controller monitoring / Timeout)

The values for **Ctrc**, **Ctrd** and **Ctrt** apply for all axes simultaneously. The values for **Ctrs** and **Ctrf** can be set individually for each axis.



The difference in position is the deviation of the present actual position from the preset target position. If the actual position is outside of the preset capture range, the controller moves at constant speed (if is set "ctrm 0"). This is set with **Ctrs**. Within the capture area, the speed of travel is adapted to the difference in position. This adaption can be influenced with the parameter **Ctrf**.

The parameters have the following meanings:

- **Ctrc** The value in *Ctrc* specifies the sampling time with which the controller is called.. Generally speaking, attenuation is increased as the sampling time increases.
- **Ctrs** The contents of *Ctrs* corresponds to a distance depending on the dimension, which specifies the capture range for the axis in question.
Example: *Ctrs* = 500, Dimension = 1 → 500 • 1µm = 500µm)
 If the difference in position is greater than *Ctrs*, the target position is approached at a constant speed.
- **Ctrf** Within the capture range, the difference in position is manipulated individually for each axis by a mathematical function. The control factor determines to what degree the respective difference in position acts on the speed at which the target position is approached.
- **Ctrd** *Ctrd* specifies how long the specified axes are not allowed to leave the target window, in order for the “position reached” signal to be transmitted.
- **Ctrt** The controller timeout limits the time available to the controller to balance out any difference in position.

Example (ctrs):

$$\frac{1\text{mm spindle pitch}}{50000 \text{ Controller steps / Capture range}} = 0,02 \mu\text{m} (= \text{one Motor increment})$$

$$\frac{\text{Capture range} = 0,1 \text{ mm} (=100)}{0,02 \mu\text{m} (\text{Motor increment})} = 5000 \text{ Motor increments}$$

$$\frac{5000 \text{ Motor increments}}{16} = 312,5 \text{ Motor increments}$$

$$\frac{312,5 \text{ Motor increments}}{\text{ctrc} (\text{Controller call})} = V_{\text{constant}}$$

Target Window	
Instruction:	!twi or ?twi
Parameters:	X, Y, Z and A 1 to 25000 (motor increments) 0.1 to spindle pitch/2 (μm) 0.0001 to spindle pitch/2 (mm)
Note:	The input and output values depend on the dimension.
Description:	!twi 1.0 0.002 → The target window is 1 mm for the X-axis and 2μm for the Y-axis(when Dim = 2). The other axes remain unchanged.
	!twi z 0.1 → The target window is set to 0.1μm for the Z-axis (when Dim = 1).
	?twi → All preset target windows are displayed.
	?twi x → The preset target window for the X-axis is displayed.
Feedback:	Target window which has actually been set (rounding errors are displayed)
Error code:	--
Example:	!twi 10 (The X-axis has a target window of 10 motor increments (when Dim = 0)). ?twi

Controller	
Instruction:	!ctr or ?ctr
Parameters:	X, Y, Z and A
	0 → Controller "OFF"
	1 → Controller "OFF after reaching target position"
	2 → Controller "Always ON"
	3 → Controller "OFF after reaching target position" at reduced current.
	4 → Controller "Always ON" with reduced current.
Description:	!ctr y 2 → Y-axis controller "Always ON".
	?ctr → All controller statuses are displayed
	?ctr x → Display of the X-axis controller status
Feedback:	Controller statuses
Error code:	--
Example:	!ctr 0 0 0 0 (All controllers "OFF") ?ctr

Controller Timeout	
Instruction:	!ctrtr or ?ctrtr
Parameters:	0 – 10000 (ms)
Description:	!ctrtr 2 → Controller timeout 2ms.
	?ctrtr → Display of the control timeout
Feedback:	Controller timeout
Error code:	--
Example:	!ctrtr 0 (Controller timeout "OFF") ?ctrtr

Controller Call	
Instruction:	!ctrc or ?ctrc
Parameters:	1 – 100 (ms)
Description:	!ctrc 2 → Controller call every 2ms.
	?ctrc → Controller call time is displayed.
Feedback:	Controller call time
Error code:	--
Example:	!ctrc 10 (Controller call every 10ms) ?ctrc

Controller Steps	
Instruction:	!ctrs or ?ctrs
Parameters:	X, Y, Z and A 1 to spindle pitch
Note:	Input and output values depend on the dimension
Description:	!ctrs y 2 → 2mm controller steps for the Y-axis (when DIM = 2).
	?ctrs → All controller steps are displayed.
	?ctrs x → Display the controller steps for the X-axis.
Feedback:	Controller steps
Error code:	--
Example:	!ctrs 4 5 7 9 (Controller steps for all axes, dependent on the dimension) ?ctrs

Control Factor	
Instruction:	!ctrf or ?ctrf
Parameters:	X, Y, Z and A 1 – 64
Description:	!ctrf y 2 → Control factor for the Y-axis is 2.
	?ctrf → All control factors are displayed.
	?ctrf x → Display of the control factor for the X-axis.
Feedback:	Control factors
Error code:	--
Example:	!ctrf 1 2 3 4 (Set all control factors) ?ctrf

Controller Delay	
Instruction:	!ctrd or ?ctrd
Parameters:	0 – 100 (ms)
Description:	!ctrd y 2 → Controller delay for Y-axis is 2ms.
	?ctrd → All controller delays are displayed.
	?ctrd x → Display of the controller delay for the X-axis.
Feedback:	Controller delay
Error code:	--
Example:	!ctrd 0 0 0 0 (All controller delays "OFF") ?ctrd

Controller (Fast Move)	
Instruction:	!?ctrfm
Parameters:	0 or 1
Note:	Meaning of Fast Move function: A new vector is startet if the controller differnce is greater than the capture range.
Description:	!ctrfm 1 => Activates the Fast Move funktion ?ctrfm => Display the status of the Fast Move function
Feedback:	0 = Fast Move function not active 1 = Fast Move function active
Error code:	--
Example:	!ctrfm 0 (Fast Move function „OFF“)

Controller (Fast Move Counter)	
Instruction:	!?ctrfmc
Parameters:	0 to 255
Note:	Meaning of Fast Move function: A new vector is startet if the controller differnce is greater than the capture range.and the corresponding counter raised by one.
Description:	!ctrfmc 0 => Clear Fast Move Counter ?ctrfmc => Display the amount of performed Move functions.
Feedback:	0 to 255
Error code:	--
Example:	!ctrfmc 0 (Clear Counter)

4.14 Special Instructions for the MR-System

MR Offset	
Instruction:	!mro or ?mro
Parameters:	X, Y, Z and A +- 2048
Description:	!mro 20 -3 56 → Offset sinx = 20, Offset cosx = -3 and Offset siny = 56 points.
	!mro y 2 9 → Offset siny = 2 and Offset cosy = 9 points.
	?mro → All offset values are displayed
	?mro x → Display of the offset values for the X-axis
Feedback:	Always sin cos for each axis
Error code:	--
Example:	!mro 0 0 0 0 (Set the offset values for the X-and Y-axes to 0) ?mro

Maximum Signal Values (Peaks)	
Instruction:	!mrp oder ?mrp
Parameters:	X, Y, Z and A +- 2048
Description:	!mrp → Error 2 (There were up to 16 values).
	!mrp y 1000 -1000 1000 → Pos.peak siny = 1000, neg. peak siny = -1000 and Pos. Peak cosy = 1000 points.
	?mrp → All peaks are displayed.
	?mrp x → Display of the peaks for the X-axis.
Feedback:	Always pos. sin, neg. sin and pos. cos, neg. cos for each axis.
Error code:	--
Example:	!mrp 0 0 0 0 (Set peaks for the X-axis to 0) ?mrp

Actual Signal Values	
Instruction:	!mrt or ?mrt
Parameters:	X, Y, Z and A
Description:	!mrt → Error 2
	!mrt z → Error 2
	?mrt → Error 2
	?mrt x → Display of the actual signal values for the X-axis
Feedback:	Always 10 * (sin, cos for the respective axis)
Error code:	--
Example:	?mrt a (Display of the actual signal values for the A-axis)

Amplification (Gain) Factor	
Instruction:	!mra or ?mra
Parameters:	X, Y, Z and A 0.01 – 2.00
Note:	The amplification or gain factor always refers to the cosine signal
Description:	!mra 1 1.01 0.98 → Amplification factors for cosx = 1, cosy = 1.01 and cosz = 0.98
	!mra z 1.23 → Amplification factor cosz = 1.23
	?mra → Display of the amplification factors for all axes.
	?mra x → Display of the present amplification factor for the X-axis.
Feedback:	Amplification (gain) factor
Error code:	--
Example:	!mra 1.11 (Amplification factor for the X-axis is = 1.11) ?mra a (Display of the present amplification factor for the A-axis)

Signal Shape	
Instruction:	!mrs or ?mrs
Parameters:	X, Y, Z and A 0 or 1
Note:	0 = Sine and 1 = Cosine
Description:	!mrs → Error 2
	!mrs z 1 → Selection of the cosine signal for the Z-axis.
	?mrs → Display of the axis identification and the signal values.
	?mrs x → Error 2
Feedback:	Signal identification (y 0: values ->)
Error code:	--
Example:	!mrs x 0 (Selection of the sine signal for the X-axis) ?mrs (Display of the signal values for the preset axis and the signal identification)

4.15 Interpretation Of Clock Pulse And Direction Of Rotation Specifications *(not for ECO-STEP)*

Instead of using vector instructions or the joystick, the axes can be moved back and forth with clock pulse signals dependent on the direction of rotation signals, at option. This mode is also possible asynchronously to travel operations which have been initiated by means of travel instructions. The multi-function port MFP is available for this purpose.

Note: As described in Clock Pulse Forward/Back (internal control) , the same function can be transmitted via the serial interface.

4.15.1 Range Of Travel Monitoring

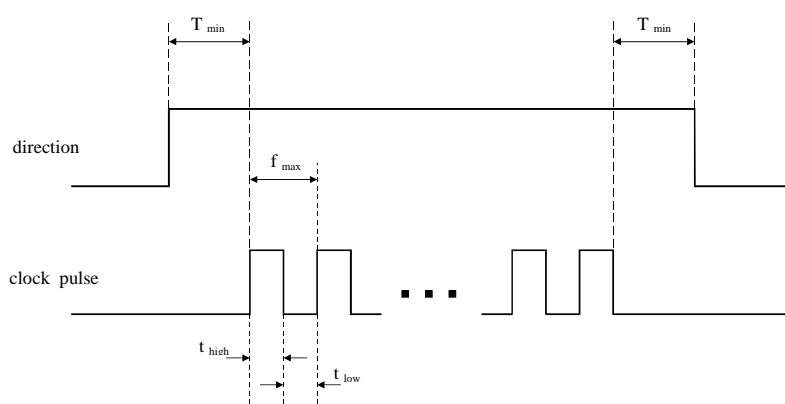
In TVR mode, it is also checked that the permissible travel limits are not exceeded. The travel limits may thereby have been determined using the combination *Calibrate* and *Measure Stroke*. Another way is to set the travel limits with a command (instruction). If the controller determines that the accumulated metering pulses would cause a travel limit to be exceeded, all further movement of the axis in that direction is inhibited. Travelling is however still possible in the opposite direction. No notification is given to the PC.

Note: The applications (user-written) program is responsible for ensuring that the maximum Start /Stop frequencies of the drive are not exceeded and that the respective axis is not overloaded acceleration-wise.

4.15.2 Temporal Marginal Conditions For the Signals

The temporal sequence of the flanks of the clock pulse and direction of rotation signals of an axis is subject to the following marginal conditions

- The next clock pulse may be applied T_{min} after every change in polarity of the direction of rotation signal at the earliest.
- The clock pluses must have been completed T_{min} prior to every change in polarity of the direction signal at the latest
- T_{min} is presently $50\mu s$.
- The maximum clock pulse frequency must not exceed $f_{max} = 833 \text{ kHz}$, whereby the minimum times $T_{low} = 600ns$ and $T_{high} = 600ns$ must be maintained.
- To protect the control inputs, input filters of 470Ω and $220pF$ are used. You must therefore ensure that the clock pulse source has an adequate driver power.



Clock Pulse Forward / Back	
Instruction:	!tvr or ?tvr
Parameters:	X, Y, Z and A 0, 1, 2, 3, 4
Description:	Functions:
	0 → Clock pulse Forward / Back "OFF".
	1 → Normal clock pulse Forward / Back processing.
	2 → Clock pulse Forward/Back process with a factor.
	3 → Clock pulse Forward/Backward processing must be enabled externally with Start/Stop inputs.
	4 → Combination of 2 & 3.
	Instructions:
	!tvr 1 1 → Activate clock pulse forward/back for the X-and Y-axes.
	!tvr a 1 → Activate clock pulse Forward/Back for the A-axis.
	?tvr → All preset status are displayed.
	?tvr z → The present status of the Z-axis is displayed.
Feedback:	Status depending on the analog channel
Error code:	--
Example:	!tvr 1 (Activate clock pulse Forward/Back for the X-axis) ?tvr

Factor Clock pulse Forward/Back	
Instruction:	!tvrfr or ?tvrfr
Parameters:	X, Y, Z and A 0.01 – 100.00
Description:	!tvrfr 1.00 1.00 → Clock pulse Forward/Back is to use a factor 1 for the X- and Y-axes (i.e. one clock pulse = one motor increment).
	!tvrfr a 1 → Clock pulse Forward /Back is to use a factor 1 for the A-axis
	?tvrfr → All preset factors are displayed
	?tvrfr z → The present factor for the Z-axis is displayed
Feedback:	Factor values
Error code:	--
Example:	!tvrfr 10.00 (Factor = 10.00 for the X-axis) (One clock pulse = ten motor increments) ?tvrfr

Clock Pulse Forward/Back (Internal Control)	
Instruction:	Px, nx, py, ny, pz, nz, pa, na
Parameters:	None
Description:	All instructions have the same effect as an external clock pulse with directional information. The first letter determines whether a positive (p) or a negative (n) movement is to be performed. The second letter denotes the axis which is to be moved.
Feedback:	None
Error code:	--
Example:	py (1clock pulse forwards for the Y-axis)

4.15.3 Clock pulse and Turning direction-Outputs for additional axes

TVR Output	
Instruction:	!tvROUT
Parameters:	X, y, z and a 0 or 1
Note:	X, y, z and a are additional axes, besides the main axes x, y, z and a
Description:	!tvROUT 1 1 = For axis x and y the clock pulse For/Back should be activated !tvROUT a 1 = For all axis the clock pulse For/Back should be activated ?tvROUT = Display all preset status ?tvROUT z = Display the current status of the z-axis
Feedback:	0 => Clock pulse For/Back „OFF“ 1 => Clock pulse For/Back „ON“
Error code:	--
Example:	!tvROUT 1 (Aktivate clock pulse For/Back for the x-axis) ?tvROUT

TVR Out resolution	
Instruction:	!tvRORES
Parameters:	X, y, z and a 0 to 51200
Note:	Here the resolution of the power stage to be controlled is entered
Description:	!tvRORES 1000 1000 = For axis x and y the resolution is set to 1000 impulses per revolution ?tvRORES = Display all set resolutions
Feedback:	0 to 51200 for each axis
Error code:	--
Example:	!tvRORES z 2500 (resolution of the z-axis is 2500 I/R)

TVR Out Pitch	
Instruction:	! ?tvropitch
Parameters:	X, y, z und a 0.001 – 100
Note:	Diese Angabe ist erforderlich, damit eine Korrekte Bewegung ausgeführt werden kann.
Description:	!tvropitch 1 1 = For axis x and y a spindle with a spindle pitch of 1mm is used. ?tvropitch = All spindle pitch's are displayed
Feedback:	0.001 to 100
Error code:	--
Example:	!tvropitch y 4 (The spindle pitch for the y-axis is 4mm)

TVR Out acceleration	
Instruction:	! ?tvroa
Parameters:	X, y, z and a 0.01 – 1500 R/s ²
Description:	tvroa 100 100 = The x- and y-axis is accelerated with 100 R/s ² ?tvroa = All preset accelerations are displayed
Feedback:	0.01 to 1500 [R/s ²]
Error code:	--
Example:	!tvroa z 50 (The z-axis accelerates with 50 R/s ²)

TVR Out velocity	
Instruction:	! ?tvrov
Parameters:	X, y, z and a 0 to 40.0 Revolution per second
Description:	!tvrov a 10 => a-Axis should be operated with max speed of. 10 R/s ?tvrov = All preset accelerations are displayed
Feedback:	0 to 40.0 [U/s] for each axis
Error code:	--
Example:	!tvrov 1 (x-Axis should be operated with max speed 1 U/s)

TVR Out position	
Instruction:	!?tvropos
Parameters:	X, y, z and a Min. range limits to max. range limits
Note:	Siehe !?pos
Description:	tvropos 45 88 => Set position of x- and y-axis ?tvropos = Display all current position
Feedback:	Position value (in dependence of the dimension)
Error code:	--
Example:	!tvropos 1 (Set position of x- axis)

TVR Out move absolute	
Instruction:	tvromoa
Parameters:	X, y, z and a +- Moving range
Note:	See moa ! The entry depends on the dimension
Description:	tvromoa 1 1 = The axis x and y are driven to position 1
Feedback:	--
Error code:	--
Example:	!tvromoa z 3.5 (Position the z-axis to the position 3.5)

TVR Out move relative	
Instruction:	tvromor
Parameters:	X, y, z and a +- Moving range
Note:	See mor ! The entry depends on the dimension.
Description:	tvromor 1 1 = The axis x and y are moved 1mm (Dim = 2)
Feedback:	--
Error code:	--
Example:	!tvromor z 3.5 (move the z-axis 3.5mm (Dim = 2))

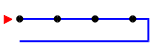

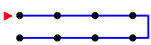

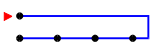

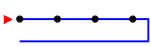

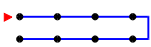

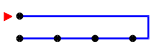



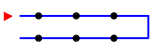





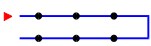



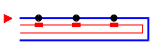

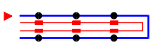

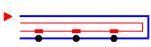

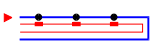

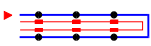

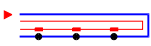



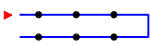



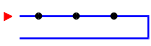

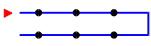



TVR Out status	
Instruction:	tvrostatus
Parameters:	
Note:	
Description:	tvrostatus => Gives the current status of the axes
Feedback:	„-“ = Axis „OFF“ „M“ = Axis in „Motion“ „@“ = Axis „Stopped“
Error code:	--
Example:	tvrostatus

4.16 Configuration of the Trigger- Output signal






These Commands synchronizes an external unit i.e. video camera or laser. The signals are send via a multi function port, which is available. Important!

Trigger	
Instruction:	?trig or !trig
Parameter:	0 or 1 (OFF / ON)
Comment:	!trig 1 → Trigger „ON“
Important!	?trig → Shows the current status of the trigger processing
	Switch on the trigger, only after all settings are transferred.. (exception with Triggermode 99)
Feedback::	ON or OFF
Error code::	--
Example::	!trig 0 (Triggerbearbeitung „OFF“) ?trig

Trigger Achse	
Instruction:	?triga or !triga
Parameter:	X, y, z or a
Comment:	!triga y → Trigger in reference to the x-axis
	?triga → Shows the current reference axis
Feedback::	X, y, z or a
Error code::	--
Example::	!triga x (Trigger referring to the x-Axis) ?triga

Trigger Modus			
Instruction:	?trigm or !trigm		
Parameter:	0 – 17 or 99		
Comment:	!trigm 0 →		 high active
	!trigm 1 →		 high active
	!trigm 2 →		 high active
	!trigm 3 →		 low active
	!trigm 4 →		 low active
	!trigm 5 →		 low active
	!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 18 →		 high active
	!trigm 19 →		 high active
	!trigm 20 →		 high active
	!trigm 21 →		 low active
	!trigm 22 →		 low active
	!trigm 23 →		 low active

!trigm 99	
With the trigger mode 99, at the beginning and at the end of the uniform motion, a trigger impulse is generated . A certain sequence has to be observed during the execution of the function. The command !trigm 99 needs to be send as the last command after the common settings, because it would be deleted with another mode setting.	
Feedback::	0 – 23! (Mode)
Error code::	--
Example::	!trigm 3 (Trigger Mode 3) ?trigm

Legende				
				
Start point	Trigger point	Bahn	External Trigger signal	 low active

Trigger Signal	
Instruction:	?trigs or !trigs
Parameter:	0 – 5 (μs) 0 = minimum Trigger (a few 100ns)
Comment:	!trigs 4 → Trigger-Signal length 4 μs
	?trigs → Shows the current status of the set Trigger-Signal length.
Feedback::	0 – 5 (μs)
Error code::	--
Example::	!trigs 3 (Trigger-Signallänge = 3μs) ?trigs

Trigger Distance	
Instruction:	?trigd or !trigd
Parameter:	1 – 5000000 Motor increments (Abhängig von der Dim)
Comment:	!trigd 1 → Trigger-Distance 1mm (bei Dim 2)
	?trigd → Shows the current Trigger distance..
Feedback::	Distance
Error code::	--
Example::	!trigd 3 (3mm Trigger distance for Dim 2) ?trigd



Trigger Offset 1 ; Trigger Offset 2	
Instruction:	?trigoffsetone ; ?trigoffsettwo !trigoffsetone ; !trigoffsettwo
Parameter:	0 – 5000000 Motor increments (depending on Dim)
Comment:	!trigoffsetone → Trigger-Distance 1mm (bei Dim 2) 20000
	?trigoffsettwo → Shows the current Trigger distance.
Feedback::	Offset (distance, degree, or Revolution)
Error code::	--
Example::	!trigoffsettwo 180 (180 Grad bei Dim 3) ?trigoffsettwo

Trigger Counter; Trigger Counter 2	
Instruction:	?!trigcount; ?trigcounttwo
Parameter:	0 bis 2147483647
Comment:	Es werden alle OFFgegebenen Trigger gezählt
	!trigcount 0 => Clear Counter 1 ?trigcounttwo 0 => Clear Counter 2 ?trigcount => Lese Zählerstand Counter 1 ?trigcounttwo => Lese Zählerstand Counter 2
Feedback::	Anzahl der OFFgeführten Trigger
Error code::	--
Example::	?trigcount ; ?trigcounttwo (Lese Zählerstand)

4.17 Configuration Of The Snapshot Input

The current positions can be saved in the controller whilst travelling is in progress with these instructions. These values can then subsequently be read out or the positions approached. This signal is set via the multi-function port, which is available as an option.

Snapshot	
Instruction:	?sns or !sns
Parameters:	0 or 1
Description:	!sns 1 → Snapshot "ON"
	?sns → Gives the present snapshot status.
Feedback:	Snapshot status
Error code:	--
Example:	!sns 0 (Snapshot "OFF") ?sns

Snapshot-Level (Polarity)	
Instruction:	?snsl or !snsl
Parameters:	0 or 1
Description:	!snsl 1 → Snapshot is high-active. 
	?snsl → Gives the current polarity
Feedback:	Current polarity
Error code:	--
Example:	!snsl 0 (Snapshot is low-active)  ?snsl

Snapshot Filter	
Instruction:	?snsf or !snsf
Parameters:	0 – 100 ms
Note:	Serves as input filter for rebounding switches
Description:	!snsf 10 => 10 ms Eingangsfilter
	?snsf => Gives the current status
Feedback:	Current filter time
Error code:	--
Example:	!snsf 0 (no input filter) ?snsf

Snapshot-Mode	
Instruction:	?snsm or !snsm
Parameters:	0 or 1
Description:	!snsm 1 → Snapshot "Automatic" The position is approached automatically after the first pulse has been given.
	?snsm → Gives the present mode
Feedback:	Snapshot mode
Error code:	--
Example:	!snsm 0 (Normal snapshot) ?snsm

Snapshot Counter	
Instruction:	?snsc
Parameters:	-
Description:	Contents are deleted after every "read".
	?snsc → Gives the number of initiated snapshots.
Feedback:	Number of initiated snapshots
Error code:	--
Example:	?snsc

Snapshot Position	
Instruction:	!snsp or ?snsp
Parameters:	X,Y,Z and A Min./max. range of travel
Note:	Input and output depend on the dimension.
Description:	!snsp 1000 2000 3000 → Positional values are set for the X-, Y-, and Z-axes.
	!snsp y 2000 → Position of the Y-axis is set.
	?snsp → Inquire present snapshot position of all axes.
	?snsp z → Inquire present snapshot position of Z-axis.
Feedback:	Positional values
Error code:	--
Example:	!snsp 100 200 (Set the X-and Z-axis positions) ?snsp (Inquire the snapshot positions of all axes)

Snapshot Position Array	
Instruction:	?snsa
Parameters:	X,y,z and a 1 – 200 (Positions)
Note:	Input and output depend on the dimension
Description:	?snsa 33 → Inquire the snapshot position 33 of all axes
	?snsa z 99 → Inquire the snapshot position 99 of z-axis.
Feedback:	Position values
Error code:	--
Example:	?snsa 1 (Inquire the snapshot position 1 of all axes)

Snapshot Offset	
Instruction:	!snsa
Parameters:	x,y,z and a
Note:	only for automatic operation
Description:	?snsa!snsa 2 0 0
Feedback:	set value
Error code:	--
Example:	!snsa -2 0 1 (the X-Axis will be moved 2mm back and the Z-axis moves 1mm forward, like the saved position.

5 Appendix General

5.1 Multi-Function Port Pin Assignment (Not for ECO-STEP)

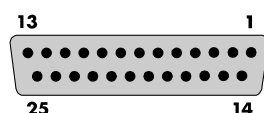


Fig.: The Multi-Function Port (25 Pole Sub-D Socket)

Due to the variety of functions, some of the pins of the multi-function port (MFP) have more than one assignment. Depending on how the controller is equipped, this means that only one signal output or input is present on a pin of the MFP.

The desired functionality has to be clarified with the order.

Standard is: Trigger, Snapshot, and Stop input.

Pin	Signal		Remarks
1	Clock input X	Standard:	TTL level
	Clock output X	Special function:	TTL level
	Encoder input X Track A	Special function:	+5V U-low $\leq 0.8V$ / U-High $\geq 3.6V$
2	X Forward /Back input	Standard:	TTL level
	X Forward/Back output	Special function:	TTL level
	Encoder input X Track B	Special function:	+5V U-low $\leq 0.8V$ / U-High $\geq 3.6V$
3	Clock input Y	Standard:	TTL level
	Clock output Y	Special function:	TTL level
	Encoder input Y Track A	Special function:	+5V U-low $\leq 0.8V$ / U-High $\geq 3.6V$
4	Y Forward/Back input	Standard:	TTL level
	Y Forward / Back output	Special function:	TTL level
	Encoder input Y Track B	Special function:	+5V U-low $\leq 0.8V$ / U-High $\geq 3.6V$
5	Clock input Z	Standard:	TTL level
	Clock output Z	Special function:	TTL level
	Encoder input Z Track A	Special function:	+5V U-low $\leq 0.8V$ / U-High $\geq 3.6V$
	Trigger out 2	Standard:	TTL-level / I _{max} = 1,6 mA
6	Analog input Ain 10 Channel 10	Special function:	Measuring range 0...0.5V / R _i = 1.1 k Ω

Pin	Signal		Remarks
7	Start / Stop Z	Standard:	TTL level $\overline{\text{Start}}$ $\overline{\text{Stop}}$ Enable for clock pulse Forward/Back
	Ain 8 / Channel 8	Special function:	0...5V
8	Start / Stop X	Standard:	TTL level $\overline{\text{Start}}$ $\overline{\text{Stop}}$ Enable for clock pulse Forward / Back
	Ain 6 / Channel 6	Special function:	0...5V
9	- 12V		$I_{\max} = 20\text{mA}$
10	Joystick on	Standard:	External switch „MAN/AUTO,,
11	VAGND	Standard:	Ground of the +5V reference voltage
12	Joystick Y Ain 1 / Channel 1	Standard:	lies parallel to ST1 pin 4
13	VAREF	Standard:	+5V reference voltage
14	Z Forward/Back input	Standard:	TTL level
	Z Forward/Back Output	Special function:	TTL level
	Encoder input Z Track B	Special function:	+5V $U_{\text{low}} \leq 0.8\text{V}$ / $U_{\text{High}} \geq 3.6\text{V}$
15	Tigger out	Standard:	TTL level / $I_{\max} = 1.6 \text{ mA}$
16	GND		
17	+5V		$I_{\max} = 300 \text{ mA}$
18	Analog output Channel 2	Standard:	Analog output 0...10V or +/-10V depending on component placement, $R_{i\min} = 1\text{kOhm}$ / $I_{\max} = 10\text{mA}$
	Digital output	Special function:	TTL level
19	Ain 9 / Channel 9	Special function:	Measuring range 0...0.5V/ $R_i = 1.1 \text{ k}\Omega$
20	Start / Stop Y	Standard:	TTL level $\overline{\text{Start}}$ $\overline{\text{Stop}}$ Enable for clock pulse Forward/Back
	Analog input	Special function:	0...5V
21	+12V		$I_{\max} = 500 \text{ mA}$
22	SnapShot input	Standard:	TTL, Pull Up = 4.7 kOhm, RC-Filter 470 Ohm/100nF
23	Stop input	Standard:	TTL, Pull Up = 4.7 kOhm, RC-Filter 470 Ohm/100nF
24	Joystick X Ain 0 / Channel 0		lies parallel to ST1 pin 3
25	Joystick Z Ain 2 / Channel 2		lies parallel to ST1 pin 5

5.2 RS232 Interface Pin Assignment

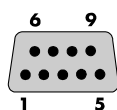


Fig.: The RS232 Interface (9 Pole, Sub-D Socket)

Pin	Signal	Remarks
1	n.c.	
2	RxD	LSTEP receive line
3	TxD	LSTEP transmit line
4	GND	
5	GND	Signal ground
6	+5V	
7	RTS	Request to send, from LSTEP
8	CTS	Clear to send, from PC
9	either n.c. +5V or +12V DC	

5.3 The Interface Cable

LSTEP		PC		
9 Pole, Sub-D Plug	Assignment	9 Pole, Sub-D	25 pole, Sub-D	Assignment
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.	-	-	-

5.4 Joystick Connection Pin Assignment

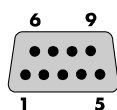


Fig.: The Joystick Connection (9 Pole, Sub-D Socket)

Pin	Signal	Remarks
1	GND	
2	Joystick On	
3	X-axis	Sliding contact of the joystick
4	Y-axis	Sliding contact of the joystick
5	Z-axis	Sliding contact of the joystick
6	Snapshot	
7	Stop	
8	VAref (+5V)	5V analog reference voltage
9	VAref (+5V)	5V analog reference voltage

5.5 The CAN Interface (Not for ECO-STEP)

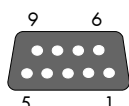


Fig.: The CAN Interface (9 Pole, Sub-D Socket)

The CAN interface is used in order to be able to operate more than one controller of the type LSTEP-xx/2 on a single PC. This is a high-speed serial connection with data rates of up to **5MBd**. In order to equip the PC with this kind of interface, an additional plug-in module is normally needed.

In theory, up to **254** different LSTEP-xx/2 controllers or other devices with a CAN port can be networked.

Physically, this interface is a twisted, two-wire cable as per RS 485 .

Attention! At the moment the CAN-Protocol is not supported.

Pin-No.	Assignment	Pin-No.	Assignment
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 plugged: +12V)
5	CAN screen (GND)	10	n.c.

5.6 The Handwheel Connection (Coax Connector)

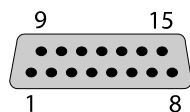


Fig.: The Handwheel Connection (15 Pole, Sub-D Socket)

Pin	Assignment	Pin	Assignment
1	Analog VCC (+5V)	9	Analog GND
2	+5V	10	Analog GND
3	A+, X-axis	11	C+, Y-axis
4	A-, X-axis	12	C-, Y-axis
5	B+, X-axis	13	D+, Y-axis
6	B-, X-axis	14	D-, Y-axis
7	TTL input resolution	15	n.c.
8	TTL input snapshot	Housing	Screen

5.7 Interpreter For MULTicontrol Commands

The LSTEP-xx/2 can also process multicontrol commands at option.

Switching to this instruction set can be done either by switching dipswitch switch no. 2

„ON,,, or by means of a command

(see Chapter 4 „Interpreter,,).

To switch to the instruction set per dip switch, the power supply to the controller must first be switched off. The dip switch of both controllers is located in the back panel.



Fig.: Dip Switch Of The LSTEP-xx/2



Fig.: Dip Switch Of The ECO-STEP

5.7.1 Input Of Parameters

Parameters can be input as integers or as floating decimals. The scientific format **cannot** be used.

23.45676	Input is supported
34.e01	Input is not supported
0.67E-1	Input is not supported

5.7.2 Supported Multicontrol Commands

The following Venus commands are presently supported:

setdim	setlimit	setaccel	calibrate	setsw
getdim	getlimit	getaccel	rmeasure	getsw
geterror	setpitch	setaxis	move	setcalvel
setvel	getpitch	getaxis	rmove	getcalvel
getvel	setpos	version	pos	setrmvel
joyspeed	setjoysticktype	identify	devpos	getrmvel
joystick	getjoysticktype	status	getpos	

If a Venus command is transmitted which the controller cannot interpret, the error code is set to 9999. Further actions are not performed.

The following commands are not supported at present:

align	getunit	setunit	Ico	scale
getec	setclloop	getclloop	Setclfactor	getclfactor
echo	And all commands which use the stack and chain.			

Deviations And Differences

Some of LSTEP Venus interpreter's instructions work a little differently to those of a multicontrol. This applies in particular to commands which display the internal status or which feed back the version number of the controller or the firmware. The known deviations are documented below:

Venus-Command	MultiControl Meaning	LSTEP-xx/2 Meaning
version	Returns the version number of the Venus interpreter	Returns the version and revision number of the ITK interpreter.
identify	Returns the identification of the controller, the switch setting at the back whilst the unit is being switched on, the internal configuration switch settings, the hardware and the software revision number	Returns the version and the revision number of the ITK interpreter. The first 4 characters of the string which is transmitted back contains the coded version number of the ITK interpreter. The next two digits specify the revision number of the version. Example.: <i>1.00-12 99 99 3d</i> means Vers. 1.00, Rev. 12
setdim	Sets the dimension of the position for the instructions with the parameters in []	Same as for MultiControl. In an incorrect number of parameter is transmitted with any of the instructions, the instructions in question are not executed
status	Returns the present status of the MultiControl	Always returns 0 . <i>Note: the status register of the LSTEP can be read out instead</i>
mode	Set the interactive mode of the Venus interpreter 1 = Terminal mode 0 = Host mode	The LSTEP-xx works exclusively in host mode . Therefore, the command <i>1 mode</i> will result in the error code 1003
save	Saves all parameters with the identification nv in the non-volatile memory	Saving of the preset parameters in the non-volatile memory is not supported at present. If this instruction is called, error code 1200 ' <i>Write error in flash memoryr</i> ' is set
restore	Overwrites all parameters with the identification nv with the values stored in the non-volatile memory	A readout of the preset parameters in the non-volatile memory is not supported at present. If this instruction is called, error code 1202 ' <i>Read error in flash memoryr</i> ' is sett
setunit	Sets the unit of an axis in physical terms	0 = motor increments 1 = μm 2 = mm

Venus-Command	MultiControl Meaning	LSTEP-xx/2 Meaning
setpitch	r, i, setpitch: Sets the spindle pitch of the axis i to r	r, i, setpitch: Sets the spindle pitch of the axis i to r. The speed axis (i=0) is not supported.
move	Absolute positioning. If the permitted range of travel is exceeded, error code 1004 is set.	Absolute positioning. Overshooting of the range of travel is monitored and may be limited to the maximum permitted range of travel. The LSTEP does <u>not</u> set the error code to 1004.
selftest	Gives the result after the self-test for an axis for the controller	Always returns 0
setjoysticktype	Defines the type of joystick which is connected	The type of joystick always depends on the number of axes of the connected controller. A 2-axis controller always assumes a two-axis joystick, a 3-axis controller always assumes a 3-axis joystick. For reasons of compatability with application programs for controllers of the type MultiControl, this instruction is permitted here. The LSTEP executes this instruction without giving any error message. It does not however have any effect, other than that the value set here can be read back with <i>getjoysticktype</i> .
getjoysticktype	Returns the present joystick type	Returns the last value which was set with <i>setjoysticktype</i>
setjoyspeed joyspeed	Sets the speed for the joystick	Has the same effect as the instruction <i>joyspeed</i> . The maximum joystick speed is specified in motor revolutions / sec. Example: 13 setjoyspeed sets the maximum speed to 13 revolutions / sec.
getjoyspeed	Returns the present joystick speed	Gives the maximum speed when the joystick is fully deflected. This is the same speed which was set with the instructions <i>joyspeed</i> or <i>setjoyspeed</i> .

Venus-Command	MultiControl Meaning	LSTEP-xx/2 Meaning
setmotortype getmotortype	Allocates certain motor types to the axes For service purposes only	The LSteps have been preset in the factory so that they can be used for all common motor types without adaption. The instructions <i>setmotortype</i> and <i>getmotortype</i> are thus not needed here and are therefore not supported.
setcurrent, getcurrent	For service purposes only	Sets or reads the output current of the axes.
v t setcalvel	Defines the calibration speed (velocity) „v“ in revolutions / sec. when approaching the limit switch position (t=1) and moving away from the limit switch position (t=2)	Defines the calibration speed „v“ in revolutions / sec when approaching the limit switch position (t=1). The speed for retracting from, i.e. moving away from the limit switch position (t=2) is preset in the LSteps. The instruction with the parameter t=2 is therefore not supported. The fault number 9999 is set.
[r] setpos	Sets the present position to [r]	Sets the present position to [r]. This corresponds to an offset of the coordinate system which is being used.
getpos	Returns the zero <u>offset</u> in microsteps	Returns the offset of the coordinate system defined by the user with <i>setpos</i> referred to the zero point after calibration in microstep
pos	Returns the present position in the current coordinate system	Returns the position with the current coordinate system (which has been offset with <i>setpos</i>) in mm .
devpos	Returns the present position from the zero position in microsteps	Returns the position in microsteps within the user coordinate system which has been offset with <i>setpos</i> .
m n l setsw	m = 0 Defines the limit switch n as normally open contact m = 1 Defines the limit switch n as normally closed contact n = 0 means the calibration limit switch n = 1 means the end limit switch l = 1,2,3 means the axis number	With the LSTEP, all limit switch inputs are wired so that depending on the type of limit switch used, the following allocation applies: Normally closed: When a limit switch is overrun a flank from 0 to 1 appears. Normally open: When the limit is overrun, a flank of 1 to 0 appears. The same allocation as that for the MultiControl applies for m, n and l

5.8 Motor Connection

The LSTEP-xx/2 is mainly designed for use with light coordinate tables, driven by 2-phase stepping motors up to 5 A. The high-resolution activation and acceleration by means of ramps in all modes of operation (including joystick), guarantees gentle running. For safe operation, you should however also heed the following points:

- Select low-resistance (low-impedance) motors with low inductances.
- Switch 8-conductor motors to low resistance.
- To avoid unnecessary heat errors, the motor current should however only be set as high as absolutely necessary.
- Motor currents which are at rated current lead to saturation of the magnetic material and step angle errors increase.

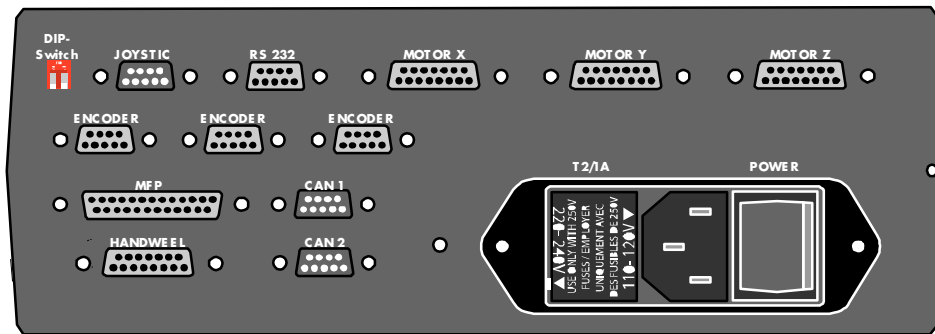
5.9 Troubleshooting

Description Of The Fault	Location / Rectification Of The Fault
1 Total failure	Check the mains power connection and fuse in the Euro-socket at the back of the unit
2 Motor overheating	Check the wiring of the motor (see Motor Connection)
3 Motor won't run at high speed	Motor is too high resistive (see Motor Connection)
4 Individual motor humming and has stopped even though a low speed has been set	Interchange the motor cables at the table, if the fault remains in the same axis: - check the cabling and motor; if the fault is now in the other axis: - there is a fault in the LSTEP
5 Individual axis not running, no humming noises	a) Check limit switches b) Check as described in 4
6 No data connection via the RS 232	a) Check the voltages at the LSTEP with the interface cable disconnected b) Check the computer and interface cable
7 LSTEP feedbacks are distorted. The correct message only appears after reading several times	LSTEP message was not read out of the receive buffer, check the application program, after a Start or Read instruction, LSTEP's response was ignored

6 Appendix LSTEP



6.1 Back Panel Of The LSTEP



6.2 Motor Connection X/Y/Z

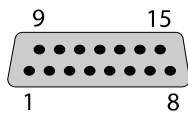


Fig.: Motor Connection (15 Pole, Sub-D Socket)

15 pole, D-SUB at LSTEP, Pin No.	Colour	12-pole Flanged Socket, Motor	Pin Connections:
1 + 9	blue	K	Phase 1R
2 + 10	orange	J	Phase 1T
3 + 11	white	B	Phase 2T
4 + 12	brown	C	Phase 2R
5	yellow	G	Limit switch end position
6	grey	H	Limit switch zero position
7	red	A	+5V
8	black	F	GND
13	green	E	Reference switch
14 (for the X- and Y-axis!)	violet	D	Temperature
14 (for the Z-axis!)	violet	D	Optional voltage for a motor brake.
15			+12V

6.3 Encoder Connection X/Y/Z (Not for ECO - STEP)

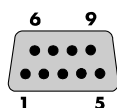


Fig.: Encoder Connection (9 Pole, Sub-D Socket)

PIN	Signal	PIN	Signal
1	U ₁₋	6	U ₁₊
2	0V	7	5V
3	U ₂₋	8	U ₂₊
4	+12V (Optional)	9	U ₀₊
5	U ₀₋	Housing	Outer screen

6.4 The Power Supply Module

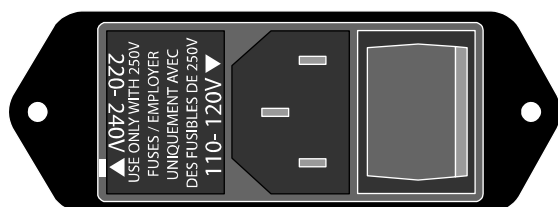


Fig.: The power supply module

The power supply consists of the device plug, the main power switch and the voltage selector with integrated power input fuses.

The controller can be operated either at 220V-240V or 110V-120V . You must set the voltage selector accordingly. The arrow for the required voltage must point to the white mark.

To change a fuse, pull the voltage selector out of the power supply module. For a voltage of 220V-240V , use a 1 amp. time-lag fuse. For a voltage of 110V-120V , use a 2 amp. time-lag fuse . (This applies for the LSTEP-1x/2 + 2x/2) The side of the arrow of the relevant voltage applies in both cases.

6.5 DIP Switch Settings



Fig.: The LSTEP DIP Switches

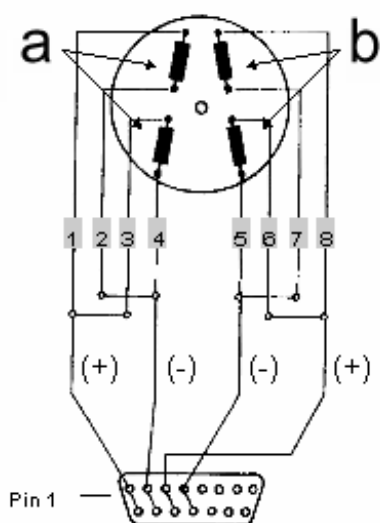
- Switch 1 ON ➔ Firmware update switched on
- OFF ➔ Firmware update switched off
- Switch 2 ON ➔ Multicontrol instruction set
- OFF ➔ Standard instruction set

6.6 Technical Data

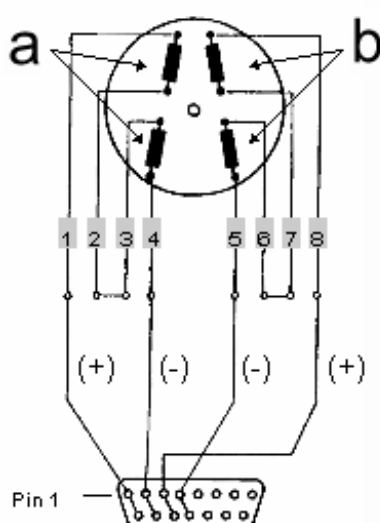
Power supply:	110V - 120V / 200V - 240V +/-10% 50/60Hz, 100VA
Fuses:	
- primary (in Euro socket):	<ul style="list-style-type: none"> • 2 A time-lag / 1 A time-lag LSTEP-1 and LSTEP-2 • 5 A time-lag / 2.5 A time-lag LSTEP-3
- secondary (on the circuit board)	Fuse1 LSTEP-1 and 2 → 5 A time-lag / LSTEP-3 → 10A time-lag
Max. power failure duration:	<p>< 50ms</p> <p>if a power failure occurs ($<0.77 \cdot U_N$), the LSTEO switches to Reset</p>
Max. motor speed:	40 r/sec. for a 200-step motor
Max. motor current:	<p>1.25A per motor phase for LSTEP-1</p> <p>2.5A per motor phase for LSTEP-2</p> <p>5.0A per motorphase for LSTEP-3</p>
Max. motor voltage:	40V
Step resolution	<ul style="list-style-type: none"> • max. 50,000 (100,000) steps/revolution for a 200 step motor. • 2000 microsteps/full step for linear stepping motors
Baud rate:	9600, 19200, 38400, 57600 or 115200
Ambient conditions:	
Air temperature when in operation:	15 ... 40 degrees C
Air temperature when not in operation:	0 ... 43 degrees C
Relative humidity when in operation	8 ... 80 % at 31° / Max. 50% at 40°
Relative humidity when not in operation	0 ... 80 %
Dimensions W * D * H (without handle):	
	250 mm • 230 mm • 100 mm for LSTEP-1x
	250 mm • 230 mm • 100 mm for LSTEP-2x
	475 mm (19") • 266 mm • 90 mm (2HE) for LSTEP-3x
Weight:	4.5 kg / LSTEP-1 and LSTEP-2
	9 kg / LSTEP-3

6.7 Wiring Of The Motor

2-phase, low resistance motor



2-phase, high-resistive motor



15 pole plug or flanged plug

6.8 Testing and Calibration Instructions

The following instructions tell you how to test and adjust the LSTEP xx/2. These jobs must only be done by duly qualified experts.



Pull out the mains plug before you open the unit!

CAUTION:

Jumper 5:	Reference voltage (+5V +/-5%)
Solder bridge 11	controlled logic voltage (+4.8V...5.25V)
Solder bridge 10	controlled logic voltage (-12V +/-5%)
Solder bridge 8	controlled logic voltage (+12V +/-5%)
Meas. point 15	Motor voltage 40 Volts

Checking The Motor Current With The Oscilloscope (X/Y-Presentation):

- X-motor current:
Connect the oscilloscope to measuring point 5 and measuring point 6.
- Y- motor current:
Connect the oscilloscope to measuring point 9 and measuring point 10.
- Z- motor current:
Connect the oscilloscope to measuring point 12 and measuring point 13.

Note:	The motor current is the measured current r_s (circle radius) and not r_{ss} (circle diameter)
--------------	---

LSTEP-1x /2	6V/A, max. 1.25A
LSTEP-2x /2	3V/A, max. 2.5A
LSTEP-3x /2	1.5V/A, max. 5.0A

Joystick Calibration

The joystick is calibrated automatically by the controller.

Note:	When switching on the controller, the joystick must not be displaced, as the controller calibrates to the zero position.
--------------	--

6.9 View Of The Circuit Boards

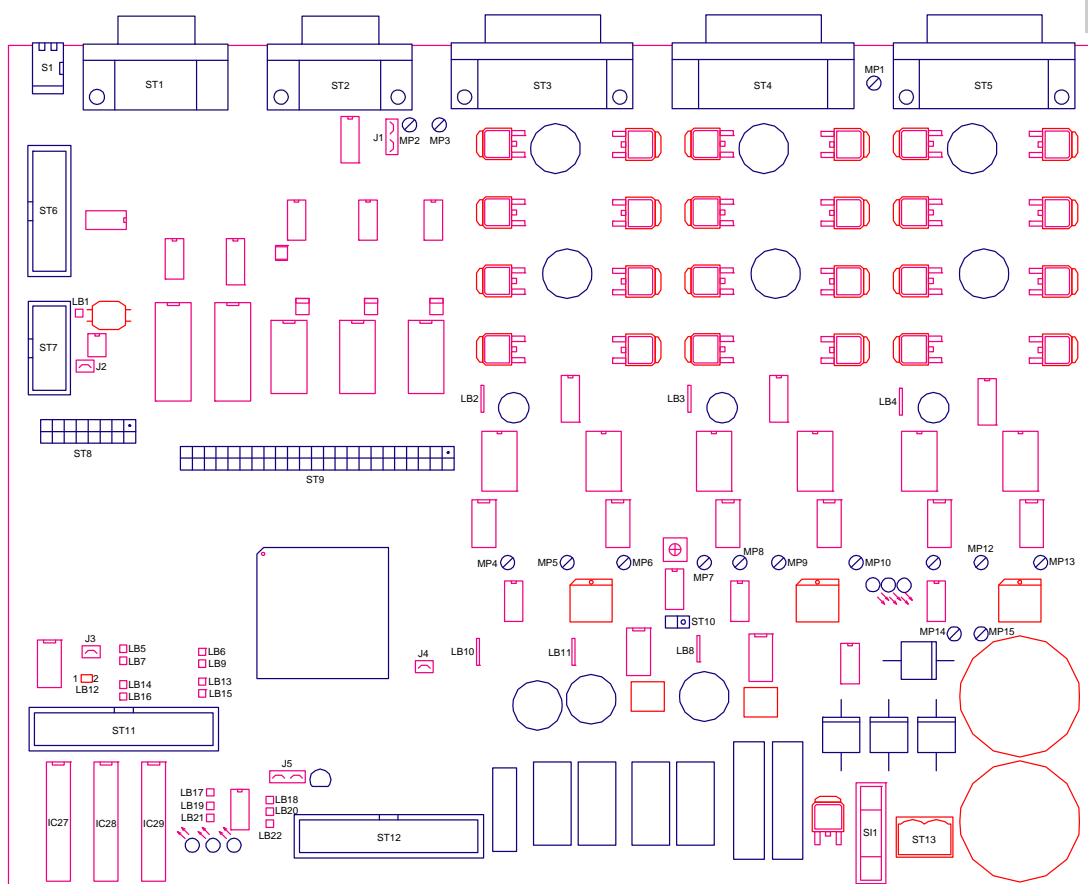


Fig.: The main circuit board

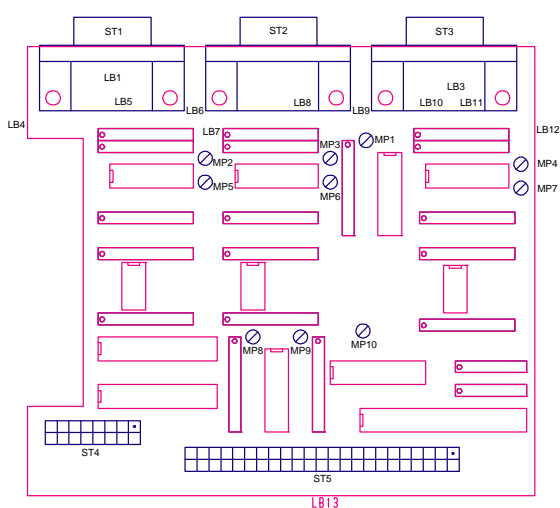
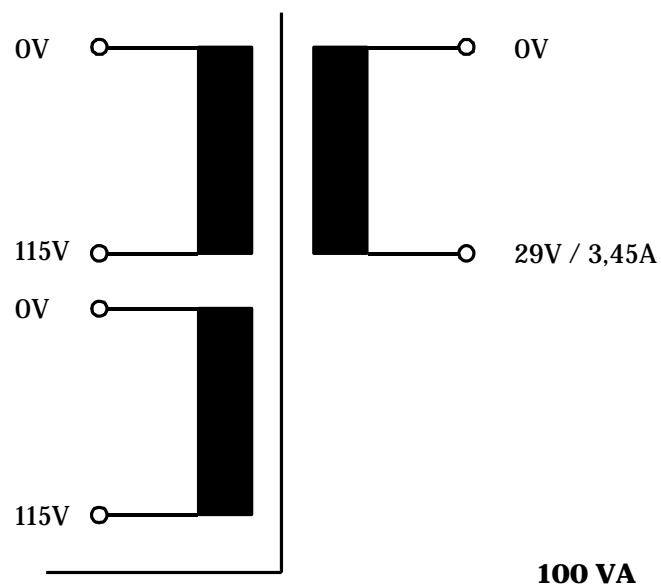


Fig.: The encoder board (optional)

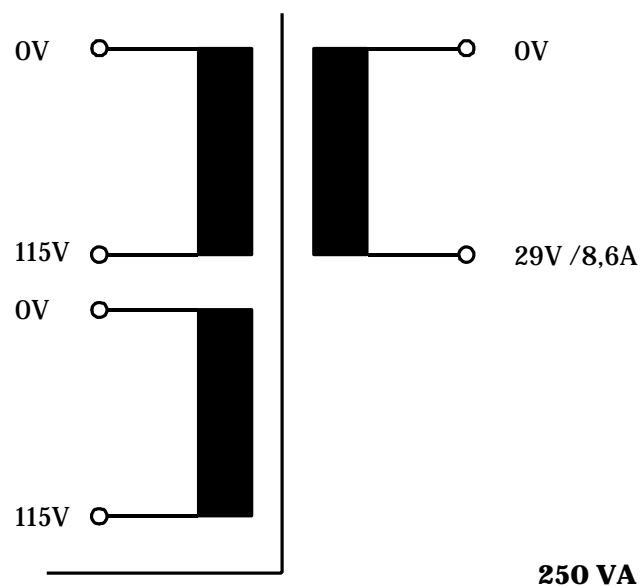
Note: The solder bridges of the encoder board are located on the soldered side of the circuit board.

6.10 Transformer Wiring

LSTEP-1x/2 ; LSTEP-2x/2



LSTEP-3x/2



6.11 I/O - Card for LSTEP Controller

Description 16 In-, 16 Outputs and 2 analogue Outputs for LSTEP 46-pin Bus adapter

The 16 in- and 16 output extensive card is suited for the LANG 46-pin female connector- bus adapter.

The connection of ST2 varies from the connection of the I/O plug from the LSTEP-PC-card. Reason: With a flat band cable, each lead of the cable can only be connected with 1A. The supply voltage of the LSTEP-PC is supplied from outside. I.e. the +11,4...32V-line carries the total current, while the GND-line stays almost non-loaded. The +11,4...32V-line is therefore 4-folded. In the existing card, the current is fed in on the card (ST4). Therefore the +11,4...32V-cable is almost non-loaded while the GND-line as a back line carries almost the total current. That is why it is 4-folded. The external power supply the 11,4...32V-power supply must be fed in through ST4. A feeding in over ST2 is inadmissible.

ST1: Connections of the 46-pin-bus adapters:

Pin No.	Function
1-9	D0 - D8
18 - 20	A1 - A3
23 - 34	A6 - A17
35	/RD
36	/WR
37	- 12V
38	+ 12V
39	+ 5V
40	GND
42	/RSTOUT

ST3: 10-pol Female connector with D-Sub-Socket-Connection: 2 Analogue Outputs

The outputs are designed for +/- 10V as a standard. Other output voltage ranges (i.e. +/- 5V, 0...5V, 0...10V,...) are possible if requested. The power handling capacity of the outputs is +/- 5mA. The internal resistance is ca. 100 Ohm.

Pin No.	Function
1,2	GND
3	Output 1
4	Output 2

Assembling of the circuit card with different voltage ranges at the analogue outputs

Output voltage	R1/R2	R4/R5	R6/R7
0...5V	10kOhm		10kOhm
0...10V	10kOhm		20kOhm
-5V...+5V	10kOhm	20kOhm	20kOhm
-10V...+10V	10kOhm	20kOhm	39 (40)kOhm

ST4: 2-pol Power plug for the supply of the In- and Outputs

The power supply of the circuit card is protected by a microfuse. The release current can not exceed 4A quick fasting fuse.

Pin No.	Function
1	+11,4...32V
2	0V

ST2: 40-pol Multipin plug with 37-pin D-Sub-socket-connection: 16 Inputs, 16 Outputs

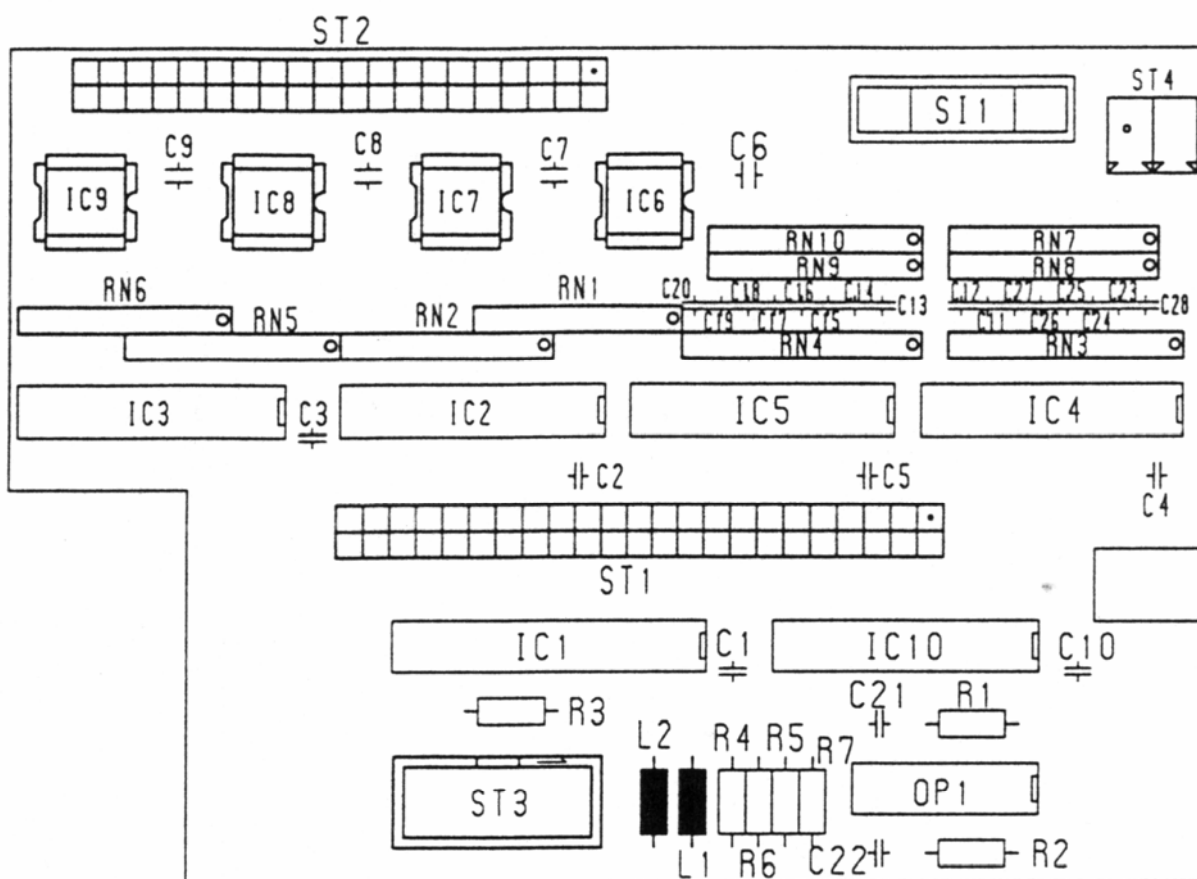
Inputs: 0...3V = „L“, 10...32V = „H“, Ri = ca. 3,3kOhm

Outputs: Switches to +Ub=11,4...32V, I_{max} = 0,5A, short circuit protected

Pin No.	Connections
1	Output 1
2	Output 2
3	Output 3
4	Output 4
5	Output 5
6	Output 6
7	Output 7
8	Output 8
9	Output 9
10	Output10
11	Output11
12	Output12
13	Output13
14	Output14
15	Output15
16	Output16
17-19	GND
20	Input 1
21	Input 2
22	Input 3
23	Input 4
24	Input 5
25	Input 6
26	Input 7
27	Input 8
28	Input 9
29	Input 10
30	Input 11
31	Input 12
32	Input 13
33	Input 14
34	Input 15
35	Input 16
36	GND
37-40	+11,4...32V

Assembly

Circuit card number 06 14 98



6.12 Documentation: Trackball for LSTEP

The trackball of the LSTEP-xx/2 was developed, to perform very fine manual movements. The trackball can be activated by switching on the joystick. The joystick is used for bigger movements, the trackball for smaller movements. It is suggested to use a LSTEP with a display., because the key functions are shown in the display.

Functions:

The trackball comes with three additional functions.

1. With the left and middle button the trackball factor can be changed.
 2. With the right button the axes X and Y can be locked individually for the trackball.
- to 1. The Trackball-Factor specifies how many motor increments are issued with a trackball-impulse. , The basic setting is 1, i.e.. 1 Impulse = 1 Motor increment. With the left button, the setting can be reduced to a factor = 0,05, with the left button increased to a factor =9.9. Pressing the left and middle button simultaneous the basic setting factor = 1 applies. The set factor is always shown for a short time in the display.
- to 2. Because it is very difficult to move only one axis, it is possible to lock and release the axes alternately with the right button. This is also displayed shortly after pressing the button.

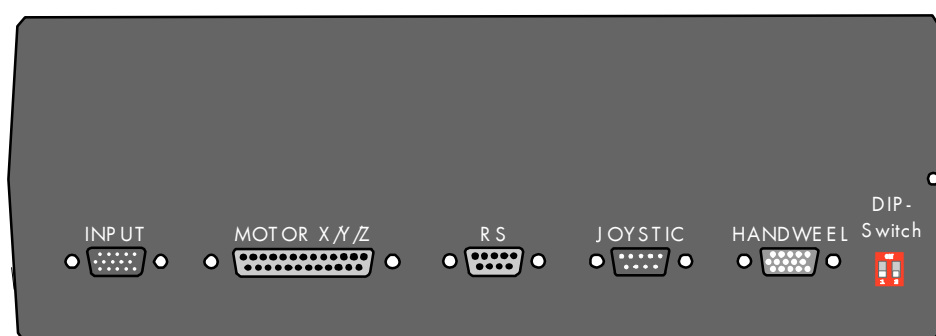
Note:

Via the command " Trackball Back Lash " the reverse back lash can be set for each axis , so that the mechanic exactly follows each change of direction with. every trackball movement. Further information to this you find in this documentation chapter 4 / command set LSTEP or in chapter 9 / Appendix LSTEP_API.



7 Appendix ECO-STEP (ECO-DRIVE, ECO-MOT)

7.1 Back Panel Of The ECO-STEP



7.2 Plug connection / Settings

7.2.1 Motor Connection X/Y/Z

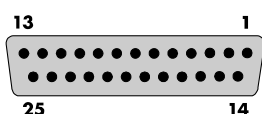


Fig.: Motor Connection (25 Pole, Sub-D Socket)

Pin	Assignment	Pin	Assignment
1	Motor X, Phase 1 +	14	Motor Z, Phase 1 +
2	Motor X, Phase 1 -	15	Motor Z, Phase 1 -
3	Motor X, Phase 2 +	16	Motor Z, Phase 2 +
4	Motor X, Phase 2 -	17	Motor Z, Phase 2 -
5	Motor Y, Phase 1 +	18	Limit switch Y zero point
6	Motor Y, Phase 1 -	19	Limit switch Y end position
7	Motor Y, Phase 2 +	20	Limit switch Z zero position
8	Motor Y, Phase 2 -	21	Limit switch Z end position
9	Limit switch X zero point	22	+5V
10	Limit switch X end position	23	+12V
11	+ Supply voltage output stage	24	GND
12	+ Supply voltage output stage	25	GND
13	+ Supply voltage output stage	Housing	GND

7.2.2 Voltage Connection

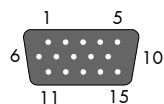


Fig.: The voltage connection (15 Pole, Sub-HD Socket)

Pin	Assignment	Pin	Assignment
1,2	GND	14,15	+24V DC controlled

7.2.3 ST 4; 15-pol HD-Sub-Socket: Koax drive

Pin No	Configuration
1	Analogue VCC (+5V)
2	+5V
3	A+, X-Axis
4	A-, X-Axis
5	B+, X-Axis
6	B-, X-Axis
7	TTL-Input Resolution
8	TTL-Input Snap-Shot
9	Analogue GND
10	Analogue GND
11	C+, Y-Axis
12	C-, Y-Axis
13	D+, Y-Axis
14	D-, Y-Axis
15	Nc
Housing	Shielding

7.2.4 ST 2: 9-pol D-Sub-Plug: Joy-Stick, Stop, Snap-Shot

Pin No.	Configuration	Comment
1	VAGND	Analogue GND
2	/Joy-Stick on	TTL, Pull Up = 4,7 k Ohm
3	Joy-Stick X	
4	Joy-Stick Y	
5	Joy-Stick Z	
6	Snap-Shot	TTL, Pull Up = 4,7 k Ohm
7	/Stop	TTL, Pull Up = 4,7 k Ohm
8	VAREF	5V Analogue Reference voltage
9	VAREF	5V Analogue Reference voltage
Housing	GND	

(Note: The connector 3-5: Joy-Stick X,Y,Z and St4, Koax drive: Axis X und Y can only be used alternative)

7.2.5 ST3, 9-pol D-Sub-Plug: RS 232-Interface

Pin-No.	Configuration
1	nc.
2	RXD
3	TXD
4	GND
5	GND
6	+5V
7	RTS
8	CTS

7.2.6 St6, 10-pol. Connecting plug, D-Sub-Configuration: CAN-Bus

Pin-No.	Configuration
1	NC
2	CAN L
3	CAN GND
4	NC
5	CAN shielding (GND)
6	CAN GND
7	CAN H
8	NC
9	CAN V+ (J1 plucked: +12V)
10	NC
Housing	Shielding

7.2.7 ST 8, 26-pol-Connecting plug: Connection for Control panel connector

Pin-No.	Configuration
1,2	GND
3	RS
4	R, /WR
5	/E
6	DB 0
7	DB 1
8	DB 2
9	DB 3
10	DB 4
11	DB 5
12	DB 6
13	DB 7
14	/STOP
15	/Joy-Stick on
16	/Clear X
17	/Clear Y
18	/Res in
19	VAGND
20	Speed
21	/CLR Z
22	+12 V
23	-12 V
24	Varef
25,26	+5V

7.3 Jumper Configuration

Identification	Function
J1	Plucked: CAN V+ = +12V
J2.1	Plucked: Varef von Precision voltage source
J2.2	Plucked: Varef von +5V

7.4 DIP Switch Settings



Fig...: DIP Switches Of The ECO-STEP

- | | | | |
|----------|-----|---|------------------------------|
| Switch 1 | ON | ➔ | Firmware update switched on |
| | OFF | ➔ | Firmware update switched off |
| Switch 2 | ON | ➔ | MultiControl instruction set |
| | OFF | ➔ | Standard instruction set |

7.5 Technical Data

Power supply:	Table-top power pack / AC INPUT: 110V - 240V 1.5 A 47-63 Hz DC OUTPUT: +24V ---- 3A
Max. power failure duration:	< 50ms if the power fails ($<0.77 \cdot U_N$) the LSTEP switches to Reset
Max. motor speed:	15 r/sec. for a 200-step motor
Max. motor current:	1.25A per motor phase
Max. motor voltage:	24V
Step resolution:	max. 50,000 steps/revolution for a 200 step motor Motor
Baud rate:	57.6 Kbd
Ambient conditions:	
Air temperature when in operation:	15 ... 40 degrees C
Air temperature when not in operation:	0 ... 43 degrees C
Relative humidity when in operation	8 ... 80 % at 31° / max.50% at 40°
Relative humidity when not in operation	0 ... 80 %
Dimensions W • D • H:	
	without display 245mm • 185mm • 90mm
	with display 245mm • 225mm • 90mm
Weight:	3.5kg

ECO-DRIVE

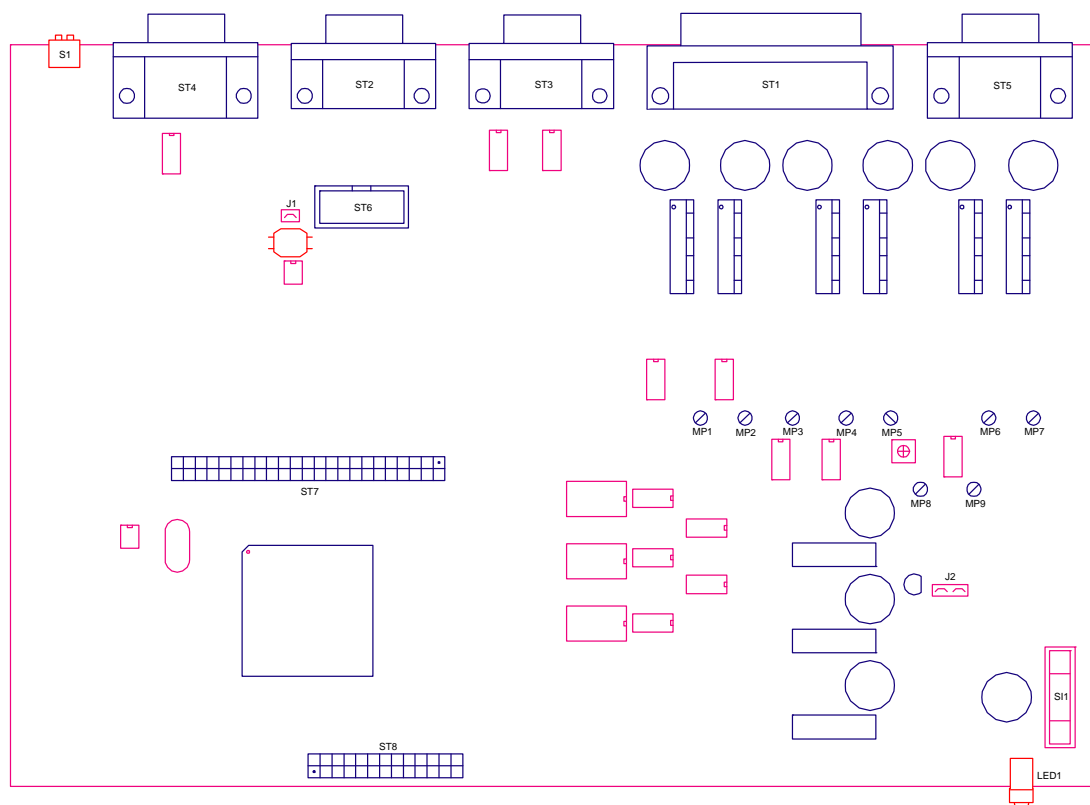
The ECO-DRIVE is only equipped with the X + Y axis and the speed is limited to 5 U/s. She is used for pinion drives and toothed racks coordinate table with a large incline (28mm/rotation)

ECO-MOT

The ECO-MOT is an single-axis controller , only equipped with the Z-axis. She is used for Focus-drives.

Attention! If the controller is set up for the register command set, the Z-axis needs to be controlled when using the API-command, except with SETVel and SetAccel in those cases the X-value is taken over.

7.6 View Of The Circuit Boards



7.7 The Powerpack

The controller is supplied complete with an external power pack.

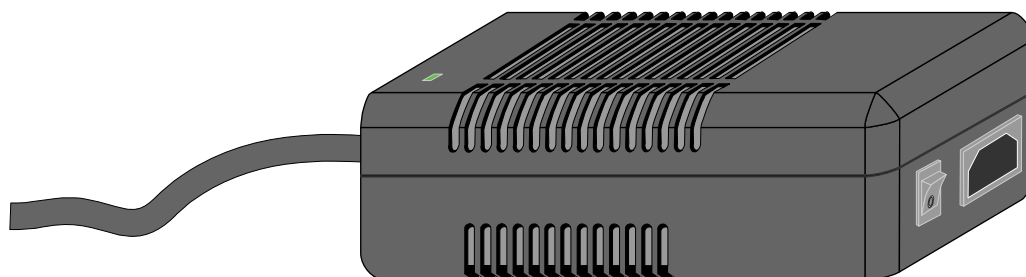


Fig.: The external power pack

7.7.1 Technical Data For The Power Pack

Power supply:	Wide range input 100 to 240V~ / 1.5A, 47-63Hz
Output:	+24V 3A

8 Hardware Documentation of the LSTEP-PCI

8.1 Jumper / PCI

Notation	Function if jumper is plugged
J1.1	RS 232 - interface St2, Pin9 = +5V
J1.2	RS 232 - interface St2, Pin9 = +12V
J2	CAN-BUS-plug St7, Pin9 = +12V
J3	If St11, Pin18 and St10, Pin5 supposed to be digital-I/O, the OP1 must be removed and J3 plugged.
J4	VPP-input controller is set to +12V (for flash-programming)
J5.1	PCI-component is booted by the content of the EEPROMS (IC21) .
J5.2	PCI- component uses its standard Vendor- and ID-number (booted internal)

0-Ohm Resistors LSTEP-PCIcompact

Notation / (R ...)	Function if resistor is equipped
0E*1 / 12; 13; 14; 15; 16; 65	NPN - Encoder (preferred equipment)
0E*2 / 32; 34; 36; 37; 38; 97	PNP - Encoder
/ 165	22V connection to the encoder OP's

8.2 Switch / PCI and PCIcompact

Notation	Function if switch is ON
S1	After reset the control goes into the bootstrapmode
S2	Reset active

8.3 Solder bridges / PCI

Solder bridge (LB)	Function closed/ Comment
1	Termination resistor CAN-Bus (120 Ohm) on St7,Pin 2 and 7
2	Power supply powerstage X (used for setup)
3	Power supply powerstage Y (used for setup)
4	Power supply powerstage Z (used for setup)
5	For separating the PT-100-Sensor amplifier of St11,Pin6. It can also be used as analogue-or digital input.
6.1	EEPROM IC21 is supplied with +5V
6.2	EEPROM IC21 is supplied with +3,3V.
12.1	Output voltage at St11,Pin18 and St10,Pin5 (Aout) = +/- 10V
12.2	Output voltage at St11,Pin18 and St10,Pin5 (Aout) = 0...10V

Solder bridges / PCIcompact

Solder bridge (LB)	Function closed / Comment
5	Termination resistor CAN-Bus (120 Ohm) on St7,Pin 2 and 7
4.1	RS 232 - Interface St2,Pin9 = +5V
4.2	RS 232 - Interface St2,Pin9 = +12V
2	CAN-BUS-Plug St7,Pin9 = +12V
3	For separating the PT-100-Sensor amplifier of St11,Pin6. It can also be used as analogue-or digital input.
1	If St11,Pin18 and St10,Pin5 supposed to be Digital-I/O, than OP1 must be removed and J3 plugged.
6.1	Output voltage at St11,Pin18 and St10,Pin5 (Aout) = +/- 10V
6.2	Output voltage at St11,Pin18 and St10,Pin5 (Aout) = 0...10V

8.4 LED's / PCI

Notation	Function
LED1	Powerstage X active
LED2	Powerstage Y active
LED3	Powerstage Z active

LED's / PCIcompact

Notation	Function
LED 3	Powerstage X aktiv
LED 1	Powerstage Y aktiv
LED 2	Powerstage Z aktiv

8.5 Plugs

8.5.1 ST1, 9-pol D-Sub-plugs: Joy-Stick, Stop, Snap-Shot / PCI and PCIcompact

Pin No	connections	Comment
1	VAGND	Analogue GND
2	/Joy-Stick on	TTL, Pull Up = 4,7 kOhm, RC-filter 470 Ohm/100nF
3	Joy-Stick X	RC-filter 10kOhm/10nF
4	Joy-Stick Y	RC-filter 10kOhm/10nF
5	Joy-Stick Z	RC-filter 10kOhm/10nF
6	Snap-Shot	TTL, Pull Up = 4,7 kOhm, RC-filter 470 Ohm/100nF
7	/Stop	TTL, Pull Up = 4,7 kOhm, RC-filter 470 Ohm/100nF
8	VAREF	5V Analogue reference voltage
9	VAREF	5V Analogue reference voltage
Housing	GND	

Note: The connections 3-5: Joystick X,Y,Z are identical with ST 11, Pin`s 24,12 and 25.

8.5.2 ST2, 10-pol Female connector with D-Sub-connection: RS 232-Interface / PCI and PCIcompact

Pin No.	Connections
1	nc.
2	RXD
3	TXD
4	GND
5	GND
6	+5V
7	RTS
8	CTS
9	J1.1 plugged: +5V; J1.2 plugged: +12V

8.5.3 St7, 10-pol. Female connector, D-Sub-connection: CAN-Bus / PCI and PCIcompact

Pin No.	Connections
1	NC
2	CAN L
3	CAN GND
4	NC
5	CAN Screen(GND)
6	CAN GND
7	CAN H
8	NC
9	CAN V+ (J2 plugged: +12V)
10	NC

8.5.4 St5, 8-pol Female connector measuring point 1-8 / PCI and PCIcompact

Pin No	Notation	Function
1	MP1	Measuring point: Port 8.0 of the control (also used internal)
2	MP2	Measuring point: Port 8.1 of the control (also used internal)
3	MP3	Measuring point: Port 8.2 of the control (also used internal)
4	MP4	Measuring point: Port 8.3 of the control (also used internal)
5	MP5	Measuring point: Port 8.4 of the control
6	MP6	Measuring point: Port 8.5 of the control
7	MP7	Measuring point: Port 8.6 of the control
8	MP8	Measuring point: Port 8.7 of the control

8.5.5 ST3, 25-pol D-Sub-socket: Motor- and proximity switch connection / PCI and PCIcompact

Pin No.	Connection
1	Motor X, phase 1 +
2	Motor X, phase 1 -
3	Motor X, phase 2 +
4	Motor X, phase 2 -
5	Motor Y, phase 1 +
6	Motor Y, phase 1 -
7	Motor Y, phase 2 +
8	Motor Y, phase 2 -
9	Proximity switch X zero point
10	Proximity switch X stop position
11	+ Power supply power stage
12	+ Power supply power stage
13	+ Power supply power stage
14	Motor Z, phase 1 +
15	Motor Z, phase 1 -
16	Motor Z, phase 2 +
17	Motor Z, phase 2 -
18	Proximity switch Y zero point
19	Proximity switch Y stop position
20	Proximity switch Z zero point
21	Proximity switch Z stop position
22	+5V
23	+12V
24	GND
25	GND
Housing	GND

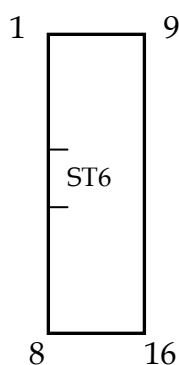
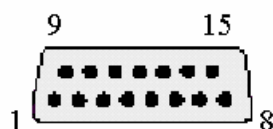
If the operating voltage is fed in at ST3, it has to be insured that there is a sufficient current load capacity of the plugs and cables (especially with flat cable).

8.5.6 St6, 16-pol Female connector (D-Sub-counter): TTL-encoder input / PCI and PCIcompact

All inputs have a TTL-level and pull-up-resistors 4,7kOhm against +5V.

St6 and St8 can only be used alternatively. The maximum count frequency is 2,5 Mflank = 625 KHz.

Pin No	Notation	Function
1	Ph1A	Incremental encoder1, Track A
2	Ph1B	Incremental encoder1, Track B
3	Ph1Z	Incremental encoder1, Track Z (Reference signal)
4	Ph2A	Incremental encoder2, Track A
5	Ph2B	Incremental encoder2, Track B
6	Ph2Z	Incremental encoder2, Track Z (reference signal)
7	GND	
8	GND	
9	Ph3A	Incremental encoder3, Track A
10	Ph3B	Incremental encoder3, Track B
11	Ph3Z	Incremental encoder3, Track Z (reference signal)
12	+5V	
13	+5V	
14	+12V	
15	+12V	
16	nc	



8.5.7 St8, 16-pol-Female connector (normal counter): Encoder-Plugin card / PCI

Pin No	Notation	Function /Comment
1	Ph1A	Incremental encoder1, Track A
2	Ph1B	Incremental encoder1, Track B
3	Ph1Z	Incremental encoder1, Track Z (Reference signal)
4	Ph2A	Incremental encoder2, Track A
5	Ph2B	Incremental encoder2, Track B
6	Ph2Z	Incremental encoder2, Track Z (reference signal)
7	ClkIn	TTL-Clock signal of T6, IC7/Pin66
8	Ph1A	Incremental encoder3, Track A
9	Ph1B	Incremental encoder3, Track B
10	Ph1Z	Incremental encoder3, Track Z (reference signal)
11	/ERRX	TTL-input error signal X (active L)
12	/ERRY	TTL-input error signal Y (active L)
13	/ERRZ	TTL-input error signal Z (active L)
14	CSAD	CS-Signal 4, IC7,Pin3: For AD-converter (active L)
15, 16	nc	

8.5.8 St11, 26-pol Female connector, D-Sub counter: Multi functioning port / PCI and PCIcompact

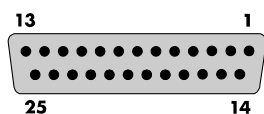


Fig.: The Multi-Function Port (25 Pole Sub-D Socket)

Due to the variety of functions, some of the pins of the multi-function port (MFP) have more than one assignment. Depending on how the controller is equipped, this means that only one signal output or input is present on a pin of the MFP.

The desired functionality has to be clarified with the order.

Standard is: Trigger, Snapshot, and Stop input

Pin No	Notation	Comment
1	Pulse X	Pulse in- or output, 10kOhm against +5V, RC-component 470Ohm/220pF
2	V/R X	V/R in- or output, 10kOhm against +5V, RC-component 470Ohm/220pF
3	Takt Y	Pulse in- or output, 10kOhm against +5V, RC-component 470Ohm/220pF
	Tigger out 2	Standard: TTL- level / $I_{\max} = 1,6 \text{ mA}$
4	V/R Y	V/R in- or output, 10kOhm against +5V, RC-component 470Ohm/220pF
5	Pulse Z	Pulse in- or output, 10kOhm against +5V, RC-component 470Ohm/220pF
6	Ain 10	Only useable if J5 is disconnected (St10, Pin6 and 7 is than inactive): Standard: TTL-input, 4,7kOhm pull-up, RC-Filter 10kOhm/100nF, Option: Analogue input0...5V
7	Ain 8	Standard: TTL-input, 4,7kOhm pull-up, RC-Filter 10kOhm/100nF Option: Analogue input0...5V (=St10, Pin3)
8	Ain 6	Standard: TTL-input, 4,7kOhm pull-up, RC-Filter 10kOhm/100nF Option: Analogue input0...5V (=St10, Pin1)
9	- 12V	
10	/Joystick on	Corresponds with St1, Pin2: TTL, Pull Up = 4,7 kOhm, RC-Filter 470 Ohm/100nF
11	VAGND	
12	AN1/Joystick Y	RC-Filter 10kOhm/100nF Joystick Y
13	VAREF	+5V Reference voltage
14	V/R Z	V/R in- or output, 10kOhm against +5V, RC-component 470Ohm/220pF
15	Tigger out	HCMOS-output: $I_{\max} = 1,6 \text{ mA}$
16	GND	
17	+5V	
18	Analogue Out	Standard: Analogue output 0...10V reps. +/-10V depending on LB12, R_i , min = 1kOhm, Option: Digital I/O (see Jumper 3) (=St10, Pin5)
19	Ain 9	Standard: TTL-input, 4,7kOhm pull-up, RC-Filter 10kOhm/100nF Option: Analogue input0...5V (=St10, Pin4)
20	Ain 7	Standard: TTL-input, 4,7kOhm pull-up, RC-Filter 10kOhm/100nF Option: Analogue input0...5V (=St10, Pin2)

21	+12V	
22	SnapShot	Input: TTL, Pull Up = 4,7 kOhm, RC-Filter 470 Ohm/100nF
23	/Stop	Input: TTL, Pull Up = 4,7 kOhm, RC-Filter 470 Ohm/100nF
24	AN0/Joystick X	RC-Filter 10kOhm/100nF Joystick X
25	AN2/Joystick Z	RC-Filter 10kOhm/100nF Joystick Z
26	Ain 3	Standard: TTL-input, 4,7kOhm pull-up, RC-Filter 10kOhm/100nF Option: Analogue input0...5V

8.5.9 St10, 10-pol Female connector with D-Sub-connection: Analogue I/O / PCI and PCIcompact

Pin No	Connection	Comment
1	Analogue In 1	Analogue: 0...5V, 4,7kOhm against +5V, RC-Filter 10KOhm/100nF (=St11,Pin8)
2	Analogue In 2	Analogue: 0...5V, 4,7kOhm against +5V, RC-Filter 10KOhm/100nF (=St11,Pin20)
3	Analogue In 3	Analogue: 0...5V, 4,7kOhm against +5V, RC-Filter 10KOhm/100nF (=St11, Pin7)
4	Analogue In 4	Analogue: 0...5V, 4,7kOhm against +5V, RC-Filter 10KOhm/100nF (=St11,Pin19)
5	Analogue Out	0...10V or +/-10V, R, Load >=1kOhm Ri= ca. 100 Ohm (=St11,Pin18)
6, 7	PT 100 Temperature sensor connection	Measuring current = 10 mA, LB 5 must be closed, St11, Pin6 is not useable)
8	GND	
9	VAREF = +5V / 1A	Output
10	NC	

8.5.10 ST4, 4-pol PC-supply unit plug: Motor power supply / PCI and PCIcompact

Pin No	Connection	Comment
1	+Um	Motor power supply: When using the PC-supply unit = 12V, for external supply unit = 11,4...48V (48V=max. use only regulated supply unit)
2,3	GND	
4	NC	

8.5.11 St 9, 46-pol Female connector: System bus (for extension module) / PCI

Pin No.	Connection
1	D0
2	D1
3	D2
4	D3
5	D4
6	D5
7	D6
8	D7
9	D8
10	D9
11	D10
12	D11
13	D12
14	D13
15	D14
16	D15
17	A0
18	A1
19	A2
20	A3
21	A4
22	A5
23	A6
24	A7
25	A8
26	A9
27	A10
28	A11
29	A12
30	A13
31	A14
32	A15
33	A16
34	/CS0
35	/RD
36	/WR
37	-12V
38	+12V
39	+5V
40	GND
41	Digital I/O 1, Interrupt capable
42	Reset Out
43	Analogue/ Digital Input1
44	Analogue/ Digital Input2
45	Analogue/ Digital Input3
46	Digital I/O 2 (Presently used internal: VPP Flash on)

8.5.12 St 8 / 50-pol Female connector: For Sin.- Cos.- Encoder evaluation PCIcompact

Pin Nr.	Connection
1	D0
2	D1
3	D2
4	D3
5	D4
6	D5
7	D6
8	D7
9	D8
10	D9
11	D10
12	D11
13	GND
14	A1
15	A2
16	A3
17	A6
18	A7
19	A8
20	A9
21	A10
22	A11
23	A12
24	A13
25	A14
26	A15
27	A16
28	CSO
29	/RD
30	/WR
31	-12V
32	+12V
33	+5V
34	GND
35	P7.4
36	/RST
37	Takt X
38	U/D X
39	Takt Y
40	U/D Y
41	Takt Z
42	U/D Z
43	CIKin
44	/Ref X
45	/Ref Y
46	/Ref Z
47	/Err X
48	/Err Y
49	/Err Z
50	CSAD

8.5.13 St12, PCI-Bus / PCI and PCIcompact

Only used pins are listed

Notation	Pin No.
AD0	A58
AD1	B58
AD2	A57
AD3	B56
AD4	A55
AD5	B55
AD6	A54
AD7	B53
AD8	B52
AD9	A49
AD10	B48
AD11	A47
AD12	B47
AD13	A46
AD14	B45
AD15	A44
AD16	A32
AD17	B32
AD18	A31
AD19	B30
AD20	A29
AD21	B29
AD22	A28
AD23	B27
AD24	A25
AD25	B24
AD26	A23
AD27	B23
AD28	A22
AD29	B21
AD30	A20
AD31	B20
C/BE0	A52
C/BE1	B44
C/BE2	B33
C/BE3	B26
/INTA	A6
PAR	A43
/SERR	B42
/PERR	B40
/STOP	A38
/DEVSEL	B37
/TRDY	B35
/IRDY	B35
/FRAME	A34
IDSEL	A26
/REQ	B18
/GNT	A17

CLK	B16
/RESET	A15
/PRSNT1	B9
/PRSNT2	B11
PCI-VIO	B19,B59,A10,A16,A59
-12V	B1
+12V	A1
+5V	A5,A8,A61,A62,B5,B6,B61,B62
GND	A:18,24,30,35,37,42,48,56 B:3,15,17,22,28,34,38,46,49,57

8.5.14 St14 / 10-pol Male connector with D-Sub-assignment: Encoder / PCI and PClcompact

Pin No.	Connection	Notation : Assembly variation
1	Encoder 0-Position X	NPN = R15 equipped / PNP = R37 equipped
2	Encoder End-Position X	NPN = R65 equipped / PNP = R97 equipped
3	Encoder 0-Position Y	NPN = R16 equipped / PNP = R38 equipped
4	Encoder End-Position Y	NPN = R14 equipped / PNP = R36 equipped
5	Encoder 0-Position Z	NPN = R12 equipped / PNP = R32 equipped
6	Encoder End-Position Z	NPN = R13 equipped / PNP = R34 equipped
7	+5V	
8	+12V	
9	GND	
10	nc	
<ul style="list-style-type: none"> Only one resistor can be equipped per encoder input. The basic equipment is only designed for the NPN Encoder. 		

8.5.15 St 9 / 40-pol Male connector with DSub assignment: 16 digitale I/O's PCIcompact

Pin No.	Connection
1	Output 1
2	Output 2
3	Output 3
4	Output 4
5	Output 5
6	Output 6
7	Output 7
8	Output 8
9	Output 9
10	Output 10
11	Output 11
12	Output 12
13	Output 13
14	Output 14
15	Output 15
16	Output 16
17	GND
18	GND
19	GND
20	Input 1
21	Input 2
22	Input 3
23	Input 4
24	Input 5
25	Input 6
26	Input 7
27	Input 8
28	Input 9
29	Input 10
30	Input 11
31	Input 12
32	Input 13
33	Input 14
34	Input 15
35	Input 16
36	GND
37	+24V
38	+24V
39	+24V
40	+24V

8.5.16 ST15 / 2-pol Plug: 24V power supply for digital I/O's PCIcompact

Pin No.	Connection
1	+24V
2	GND

8.6 Measuring point PCI / PCIcompact

Measuring point	Notation	Function
MP1	GND	
MP2	MCOSX	Measuring point motor current X-axis, phase 2 (cos)
MP3	MCOSY	Measuring point motor current Y-axis, phase 2 (cos)
MP4	MCOSZ	Measuring point motor current Z-axis, phase 2 (cos)
MP5	MSINX	Measuring point motor current X-axis, phase 1 (sin)
MP6	MSINY	Measuring point motor current Y-axis, phase 1 (sin)
MP7	MSINZ	Measuring point motor current Z-axis, phase 1 (sin)
MP8	GND	
MP9	+3,3V	+3,3V power supply
MP10	Sz.gen. 20kHz	Signal triangular generator
MP11	GND	

Measuring point/ PCIcompact Nr.	Notation	Function
MP10	GND	
MP3	MCOSX	Measuring point motor current X-axis, phase 2 (cos)
MP1	MCOSY	Measuring point motor current Y-axis, phase 2 (cos)
MP6	MCOSZ	Measuring point motor current Z-axis, phase 2 (cos)
MP4	MSINX	Measuring point motor current X-axis, phase 1 (sin)
MP2	MSINY	Measuring point motor current Y-axis, phase 1 (sin)
MP5	MSINZ	Measuring point motor current Z-axis, phase 1 (sin)
MP7	GND	
MP8	+3,3V	+3,3V power supply
MP11	Sz.gen. 20kHz	Signal triangular generator

8.7 Fuses PCI and PCIcompact

No.	Bemerkung
SI 1 PCI and PCIcompact	SI1: protects St4, Pin1 and St3, Pin11-13 (power supply power stage). Value: max. F 5A. When protecting, watch the current load capacity of the used cables (especially if ST3 is supplied with a flat cable)
SI 2 PCIcompact	Protects the 24V input voltage for the digital outputs.

8.8 Description I / O - card for the LSTEP-PCI with analogue outputs or PCIcompact

With the encoder adapter card 06 09 04 3 encoders and 2 analogue outputs with 0...10V or +/-10V can be controlled. The encoder interface is provided with difference inputs for sinus, cosine and the reference signal. Depending on the equipment of the card 1 Vss (i. e. with optical encoders), 5Vss (i. e. with MR-sensors) can be adapted.

The analogue outputs consisting of a 2-time-8-bit-converter which is amplified via OP9. Is J1.1 resp. J2.1 plugged, the output range is +/- 10V. Is J1.2 resp. J2.2 plugged, then the output voltage range is 0...10V. L1 and L2 supposed to avoid oscillation with capacity loads.

The outputs are with +/- 5mA. The inside resistor has about 100 Ohm.

Solder briges	Function (closed)
LB1	+12V on St1, Pin 4
LB2	+12V on St2, Pin 4
LB3	+12V on St3, Pin 4
LB4	120 Ohm connecting resistor St1, Pin 6 und 1 (sin)
LB5	120 Ohm connecting resistor St1, Pin 3 und 8 (cos)
LB6	120 Ohm connecting resistor St1, Pin 5 und 9 (ref)
LB7	120 Ohm connecting resistor St2, Pin 6 und 1 (sin)
LB8	120 Ohm connecting resistor St2, Pin 3 und 8 (cos)
LB9	120 Ohm connecting resistor St2, Pin 5 und 9 (ref)
LB10	120 Ohm connecting resistor St3, Pin 6 und 1 (sin)
LB11	120 Ohm connecting resistor St3, Pin 3 und 8 (cos)
LB12	120 Ohm connecting resistor St3, Pin 5 und 9 (ref)
LB 13	5,1kOhm Pull Up resistor on St5, Pin 41 (Interrupt AD-converter)

Jumper	Function
1.1	Voltage range on ANOut1 = +/- 10V
1.2	Voltage range on ANOut1 = 0...10V
2.1	Voltage range on ANOut2 = +/- 10V
2.2	Voltage range on ANOut2 = 0...10V

Plug	ST1,2,3, Plug 1(X), 2(Y), 3(Z) Encoder plug
------	---

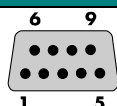


Fig.: The encoder connection (9 Pol socket)

Pin No.	Function
1	- Sin
2	GND
3	-Cos
4	when LB 1,2 or 3 are plugged, than ST 1,2,3 = +12V
5	- Ref
6	+Sin
7	+5V
8	+Cos
9	+Ref
Voltage: Encoder interface: 0,6...1,2Vss; MR-Interface max. 5Vss	

Pin No.	ST4, 16-pol Female connector PCI	
	Description	Function / Comment
1	Takt X	Clock signal of the encoder X
2	U/D X	Direction signal of the encoder X
3	Takt Y	Clock signal of the encoder Y
4	U/D Y	Direction signal of the encoder Y
5	Takt Z	Clock signal of the encoder Z
6	U/D Z	Direction signal of the encoder Z
7	Clk	TTL-Clock signal of T6
8	/Ref X	Reference signal encoder X
9	/Ref Y	Reference signal encoder Y
10	/Ref Z	Reference signal encoder Z
11	/ERRX	TTL-Error signal X (active L)
12	/ERRY	TTL-Error signal Y (active L)
13	/ERRZ	TTL-Error signal Z (active L)
14	/CSAD	CS-Signal 4: Frr AD-converter (active L)
15,16	nc	

ST5: PCIcompact
ST6: PCI
10-pol Female connector (D-Sub-connection): Analog Out

Pin No.	Function
Pin No.	Function
1,2	GND
3	Output 1
4	Output 2
7	+5V
8	+12V
9	-12V

8.9 Description I / O - card for the LSTEP-PCI (in the PCIcompact the I/O's are on board)

The 16 in- and 16 output extensive card is suited for the LANG 46-pin female connector- bus adapter. The form factor fits on the LSTEP-PCI (right-angled arrangement of 46-pin-plug and I/O-cable outlet; No further slot is blocked.) /CS, /RD and /WR-signal of 65ns length are sufficiently, while the bus 25ns after the /RD floated. For a C168 no waitstates, early read/ write, or ALE-extender are required, but a float extender is the least requirement. The connection of ST2 varies from the connection of the I/O plug from the LSTEP-PC-card. Reason: With a flat band cable, each lead of the cable can only be connected with 1A. The supply voltage of the LSTEP-PC is supplied from outside. I.e. the +11,4...32V-line carries the total current, while the GND-line stays almost non-loaded. The +11,4...32V-line is therefore 4-folded. In the existing card, the current is fed in on the card (ST4). Therefore the +11,4...32V-cable is almost non-loaded while the GND-line as a back line carries almost the total current. That is why it is 4-folded. The external power supply the 11,4...32V-power supply must be fed in through ST4. A feeding in over ST2 is inadmissible.

8.9.1 ST1: Connections of the 46-pin-bus adapter: PCI

Pin No.	Function
1-16	D0 - D15
18	A1
21 - 33	A4 - A16
34	/CS
35	/RD
36	/WR
39	+ 5V
40	GND
41	Interrupt: Low, if outputs are overloaded. (Diag. =L); LB1 closed: Pull up 10kOhm against +5V
42	/RSTOUT

8.9.2 2-pol Power plug for the supply of the In- and Output ST4/ PCI

Used with a Phoenix Mini-Combicon basic housing 2-polig.

The power supply is protected by a „Pigtail fuse“ F4A (Manufacturer: Wickmann) on the circuit card. Because the outputs are short circuit proof, a short circuit does not release the fuse at the output. LED1 (green) shows, that voltage is present behind the fuse.

Pin No.	Function
1	+11,4...32V
2	0V

8.9.3 40-pol Female connector with 37-pol D-Sub-plug-connection: 16 inputs, 16 outputs ST2/ PCI

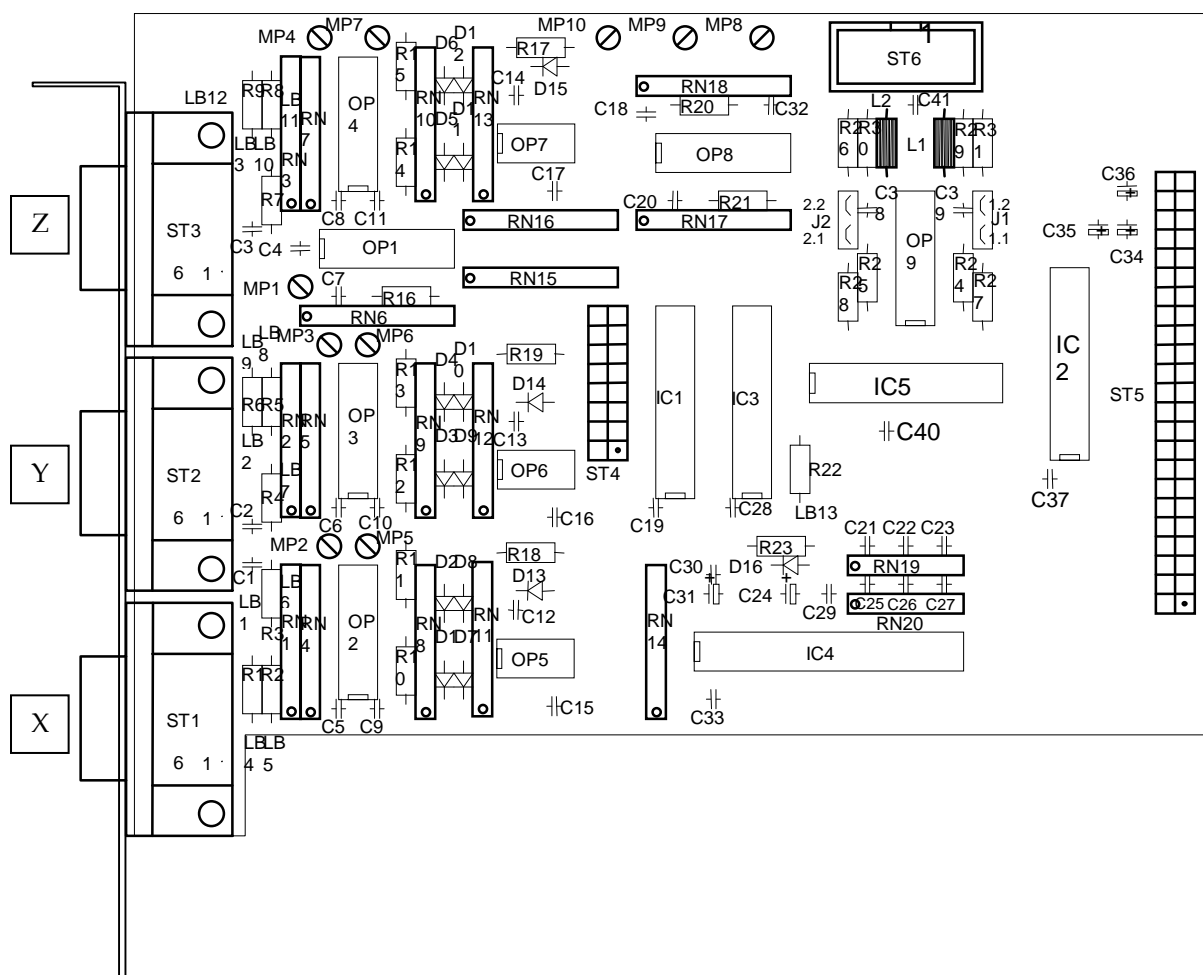
Inputs: 0...2,7V = „L“, 7,5...32V = „H“, Ri = ca. 10kOhm

Outputs: Switches to +Ub=11,4...32V, I_{max} = 0,5A, short circuit proof

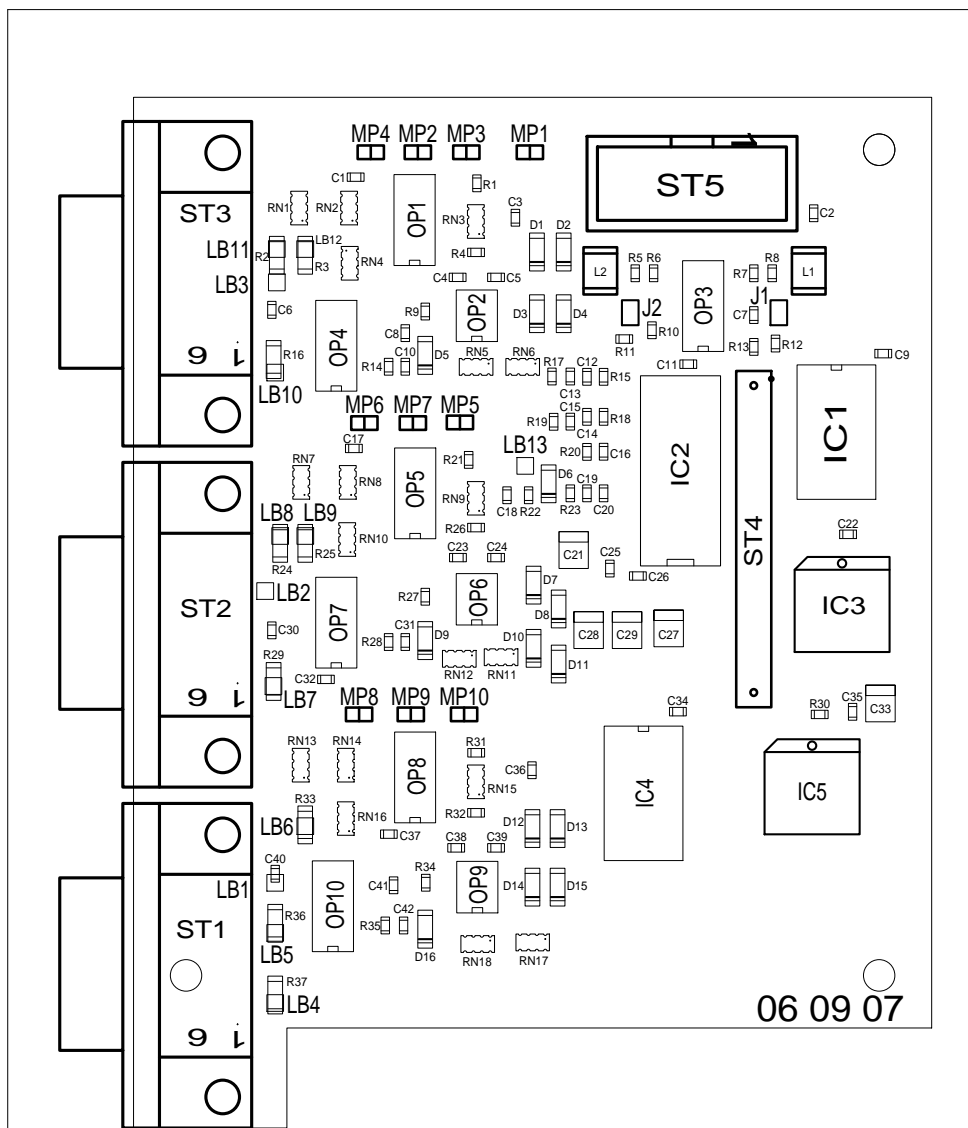
Pin No.	Connection
1	Output 1
2	Output 2
3	Output 3
4	Output 4
5	Output 5
6	Output 6
7	Output 7
8	Output 8
9	Output 9
10	Output10
11	Output11
12	Output12
13	Output13
14	Output14
15	Output15
16	Output16
17-19	GND
20	Input1
21	Input2
22	Input3
23	Input4
24	Input5
25	Input6
26	Input7
27	Input8
28	Input9
29	Input10
30	Input11
31	Input12
32	Input13
33	Input14
34	Input15
35	Input16
36	GND
37-40	+11,4...32V

8.10 Assembly scheme

LSTEP PCI- Encoder adapter card

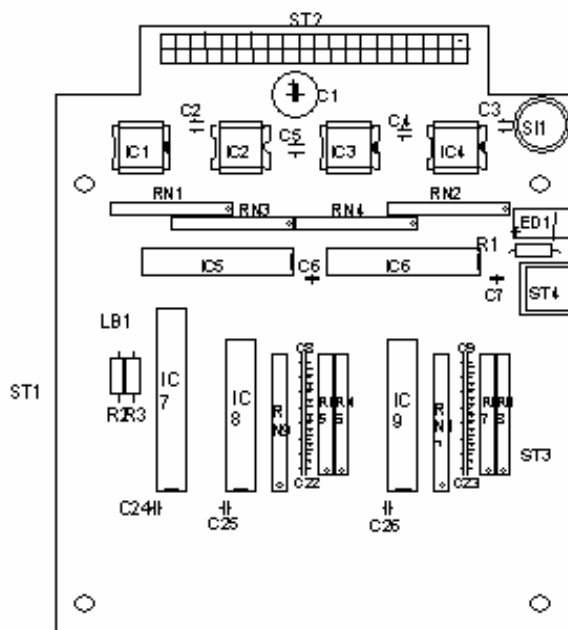


Encoder adapter card LSTEP PCIcompact 06 09 07



Die Lotbrücken (LB1-12) befinden sich auf der Lötseite der Platine./
Solder bridge (LB1-12) is located on the solder side of the circuit card

I / O – Adapter card for LSTEP PCI 06 07 00

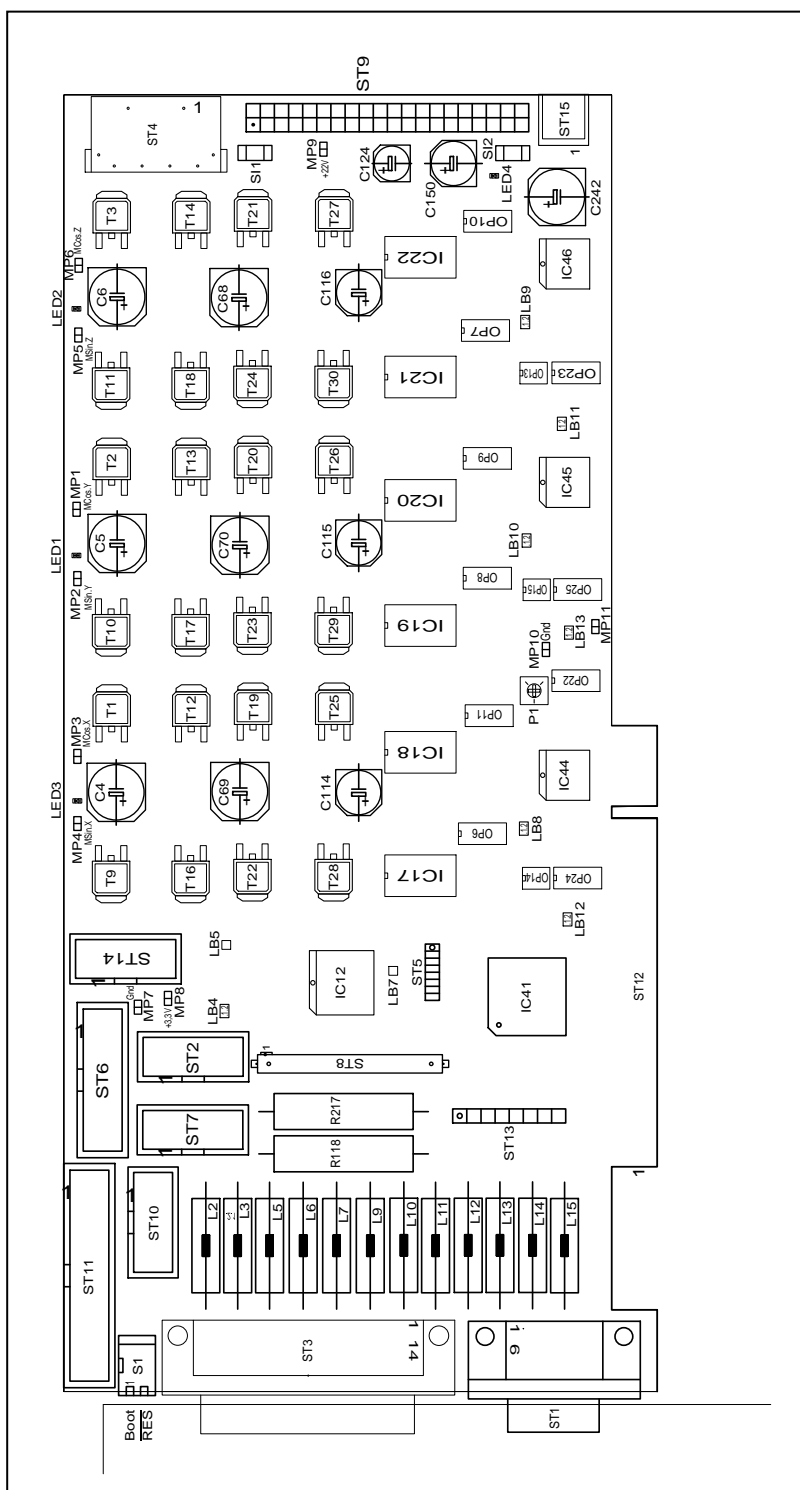


Lötbrücke LB1 befindet sich auf der Platinenlötseite.

Solder bridge LB1 is located on the solder side of the circuit card



LSTEP PCI/compact 06 06 07



8.11 Appendix LSTEP-PCI /PCIcompact

Technical Data

Power supply:	Logic voltage through PCI-Slot of PC Motor voltage: 12V from the PC-power supply unit 11,4-48V through ext. power supply unit		
Max. motor revolution:	40 U/sec. for 200-step motor		
Max. motor current:	1,25A	per motor phase	LSTEP-PCI / 1
	2,5A	per motor phase	LSTEP-PCI / 2
	3,75A	per motor phase	LSTEP-PCI / 3
Max. motor voltage:	48V		
Step resolution:	Max. 50.000 steps/revolution for 200step motor		
Baud rate:	57,6 Kbd		
Measurements L x H x B (1 Slot)			
PCI	341mm x 120mm x 20mm (1 Slot)		
PCIcompact	236mm x 107mm x 15mm (1 Slot)		
max. count frequency for TTL-Encoder inputs	2,5 Mflank= 625 KHz -Flank interpretation		

9 Appendix LSTEP-API

9.1 Introduction

The LSTEP-API (programming interface for the LStep fine positioning system) is designed to help software developers to develop applications for control of the positioning systems LSTEP xx, LSTEP xx/2, LSTEP-PC, ECO-STEP and LSTEP-44 quickly and effectively, without having to deal with hardware/machine intimate programming. It offers access to the complete command set of the LSTEP positioning systems.

The LSTEP4X API is variant of LStep APT to support the parallel control of several LSTEPs

9.1.1 Included Functions

- Windows 32-bit DLL
- Support of the stepping motor controllers LSTEP xx, LSTEP xx/2, LSTEP-PC, ECO-STEP, LSTEP-44 und LSTEP-PCI
- Activation via RS232-, ISA or PCI (DPRAM) interface
- Automatic recognition of the connected controller
- Configuration of the controller
- Execution of all commands supported by the controller
- Up to 4 axes
- Multithreading capable

9.1.2 Systemanforderungen

With LSTEP-API, as well as with LSTEP4X it is possible to develop applications with MS Windows 9x, Windows NT and Windows 2000.

9.1.3 Supported Development Environments

The LSTEP-API and LSTEP4X-API has been tested with the following development and runtime environments

Borland/Inprise Delphi 3-5

Microsoft Visual C++ 6.0

National Instruments LabVIEW

It should be compatible with all other programming environments which can use DLLs.

(DLL = Dynamic Link Library; A DLL is an executable module which contains code and resources which are used by other applications or DLLs.)

9.2 DLL Interface

9.2.1 LSTEP-API

The main component of the LSTEP-APIs is the file LSTEP4.DLL. You use this DLL for developing your own programs, to configure the LSTEP, to transmit commands, to inquire positional values, inputs/outputs, etc.

9.2.2 LSTEP4X-API

Main component of the LStep4X-APIs is the file LSTEP4X.DLL. The DLL is used for the development of several programs, to configure several LSTEPS, to send commands, to read position values, inputs and outputs etc.

9.2.3 General Information

9.2.3.1 LSTEP 4.DLL

The DLL LSTEP4.DLL implements the commands of the LSTEP-API. All functions are declared with a 32 bit integer as the return value. A return value of 0 indicates error-free execution of the function, if errors (e.g. timeouts) occur, the relevant error code (see table) is returned.

For functions such as LS_MoveAbs, values are always transmitted for 4 axes. If the controller has only 1-3 axes, the values for the non-existing axes are ignored and can be set to 0.

9.2.3.2 LSTEP4X.DLL

The DLL LSTEP4X.DLL implements the commands of the LSTEP4X-API. All functions are declared with a 32 bit integer as the return value. A return value of 0 indicates error-free execution of the function, if errors (e.g. timeouts) occur, the relevant error code (see table) is returned..

The first parameter send for all funtions of the API is an integer value (between 1 and 32), which indicates the number of the LStep, where the command is supposed to be send to.

The function LSX_CreateLSID can be used, to send such a ID-value. With a call of LSX_FreeLSID a ID-value is set free again. (see Delphi-example)

For functions such as LSX_MoveAbs, values are always transmitted for 4 axes. If the controller has only 1-3 axes, the values for the non-existing axes are ignored and can be set to 0.

9.2.3.3 Difference in comparence with LSTEP4.DLL

The function circumference LSTEP4X.DLL has not changed in compare with LSTEP4.DLL, the function names are identical. The normal LStep API (LSTEP4.DLL) are continued, existing source code, which the LStep API uses must not be modified.

LSTEP4X API opens a protocol window for each LStep, and the Log-files are also written separately for each LStep.

Because the LSTEP4X supports API Multi-Threading, programs made by the customer can access the LSteps from several Threads through the API.

The parallel control of several LSTEP motor controls is possible.

The function names received different prefixes'. For the LSTEP4X.DLL „LSX_“ is used instead „LS_“ like it is used for the LSTEP4.DLL.

A integer value is used as an additional parameter for all function calls of the LSTEP4X API which identifies the controller. The numbering of the LSTEPs starts with 1 to 32.

Note: Under Windows NT the supplied driver GIVEIO must be installed so that the DPRAM interfaces of the LSTEP-PC may be used.

9.2.4 Integration in Delphi

9.2.4.1 LSTEP4-API

All function names of the LSTEP4-API start with “LS_” for easier differentiation. To be able to use the functions of the LSTEP4 APIs, LSTEP4.pas must be present in the uses-clause of the unit it question and must be in one of the pre-set search paths.

Required files: LSTEP4.dll and LSTEP4.pas

Delphi-example for the control of a 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;
```

9.2.4.2 LSTEP4X-API

All function names of the LSTEP4X-API start with “LSX_” for easier differentiation. To be able to use the functions of the LSTEP4X APIs, LSTEP4X.pas must be present in the uses-clause of the unit it question and must be in one of the pre-set search paths.

Required files: LSTEP4X.dll and LSTEP4X.

Delphi-example for the parallel control of 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;
```

9.2.5 Integration in Visual C++

9.2.5.1 LSTEP4-API

For Visual C++ , an encapsulation of the LSTEP4.DLL has been created. The class CLStep4 loads the DLL and all pointers in response to function calls dynamically. The methods of the LSTEP-object is not preceded by "LS_" .

(Example: LS.Calibrate() instead of LS_Calibrate)

Only one instance should be created by the class CLStep4 , since at present, no more than one LStep can be controlled simultaneously with the LSTEP .

Required files: [LSTEP4.dll](#), [LSTEP4.h](#) and [LSTEP4.cpp](#)

Visual C++- example for the control of a LStep

```
...
CLStep4 LS1;
...

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

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

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

9.2.5.2 LSTEP4X-API

For Visual C++ , an encapsulation of the LSTEP4X.DLL has been created. The class CLStep4X loads the DLL and all pointers in response to function calls dynamically. The methods of the LSTEP-object is not preceded by "LSX_" .
(Example: LSX.Calibrate() instead of LSX_Calibrate)

With C++ the functions LSX_CreateLSID and LSX_FreeLSID must not be called for using the LSTEP4X.DLL, because the Wrapper-Klasse CLStep4X administers itself the integer value, which indicates the number of the LStep. The method of CLStep4X have no additional parameter for the number of the LStep.

needed files: LSTEP4X.DLL, LSTEP4X.h and LSTEP4X.cpp

Visual C++- example for the parallel control of 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;
```

9.2.6 Integration In LabVIEW

NI LabVIEW is a developing environment based on the graphic programming language G. It enables programming with graphic symbols to be done quickly and easily. Complicated 32-bit programs can be created, thus ensuring that the required speed of execution for control, test and measuring applications is given.

All LabVIEW programs (so-called VIs, Virtual Instruments) have a front panel and a block diagram and can in turn be integrated into other programs as a sub-program (SubVI).

A VI library (LSTEP4.llb) has been created for embedding (LSTEP4.DLL and LSTEP4X.DLL) which contains approx. 110 VIs. These individual VIs (e.g. LS4 ConnectSimple.vi) encapsulate the relevant LSTEP API-functions. The LSTEP4.dll is used by means of the "Call Library Function". (calling ext. libraries).

9.2.6.1 Differences between LSTEP4.LLB and LSTEP4X.LLB

In new software-projects with the LSTEP-API under LabView, you should use exclusively the VI-Library LSTEP4X.LLB. This has the following reason: Die LSTEP4.LLB only supports one, besides that all VI's have as a return value only a Boolesche variable, therefore no mistake code to be interpreted.

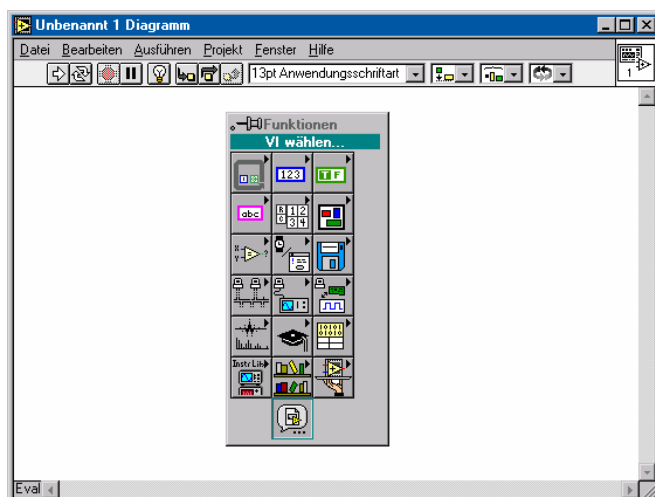
In the (newer) LSTEP4X.LLB the VI's have as a return value a 32-bit-Integer-value (which is named „error out"). If it equals 0, it indicates that the LSTEP-API-function was carried out fault-free. Otherwise the meaning of the error code is described in the table of this documentation. Several LSteps can be parallel controlled via the LSTEP4X.DLL called in the VI's. The VI's for the LSTEP4X.DLL differ in the file name from the once of the LSTEP4.DLL through the shorthand symbol at the beginning it is „LS4X“ instead „LS4“. In LabView the VI's for the LSTEP4X.DLL purple, the once of the LSTEP4.DLL has the background colour blue. The designation of the VI's in the symbols is the same.

In all VI's an additional connection comes is added. Which is a 32-bit-Integer-value and indicates the number of the LStep that the command, for example a moving command or the reading out of the current position, refers to („LStep Controller ID"). You can assign those numbers yourself (e.g. „0" for the LStep at the serial interfaces COM1, „1" for the LStep at COM2 etc.), or via the VI „LS4X CreateLSID". ID-numbers created with the VI „LS4X FreeLSID" can be released again. **If you use only one LStep in your LabView-project, you can leave the connection „LStep Controller ID" open for all used VI's from the LSTEP4X.LLB**, because it has the default-value 1 as a standard.

needed files in LabView: LSTEP4X.DLL and LSTEP4X.LLB
resp. LSTEP4.DLL and LSTEP4.LLB

9.2.6.2 Procedure for using an LSTEP4 VIs:

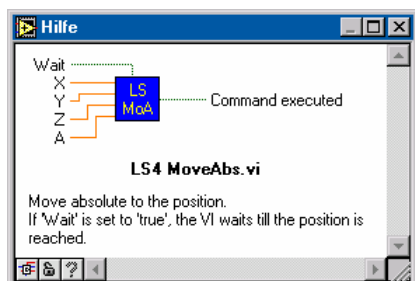
1. Create a new VI
2. Switch to the block diagram window (Ctrl+E)
3. Click on the diagram (right mouse button)



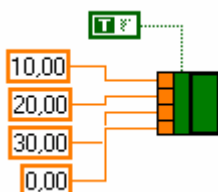
4. Select VI ...
5. Open the supplied VI Library LSTEP4.llb in the file dialog box, and then select the required command (e.g. LS4 MoveAbs.vi)
6. Place VI in the diagram



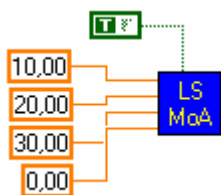
Press ctrl+H to open a help window, which gives you information about the VI at which the mouse pointer is currently located



The transmission of parameters to SubVIs is done by terminals, which have to be “cabled”. To display these terminals in the diagram, click the right mouse button on the VI and select “Display/Terminals”. You can then allocate values/sources to the terminals. There are several ways of doing this, one of them is: Click the right mouse button on the required terminal then on the menu item “produce constants”.



In this example and absolute travel command (X 10mm, Y 20mm, Z 30mm) is executed.

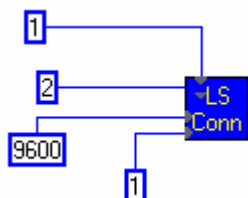


For details of the occupancy of the terminals of the Vis, please refer to the documentation for the API function in question. There, you will find a diagram, which also appears in the Help window of LabVIEW. The parameters are more or less the same as those of the DLL-function: There are only certain differences for functions to which bit masks are transmitted as the parameters (e.g. LS4 SetActiveAxes.vi)

The LSTEP4 VIs has a terminal called “Command executed”. If this logical value is “true”, the command was executed successfully. If an error has occurred, the value is set to “false”.

Before travel commands can be executed or positional values can be read out, etc., the connection to LSTEP must be opened. This is easiest with VI "LS4 ConnectSimple.vi". It initialises the interface and detects the LSTEP, which is connected.

Example for RS232 (COM2 and 9600 Baud):

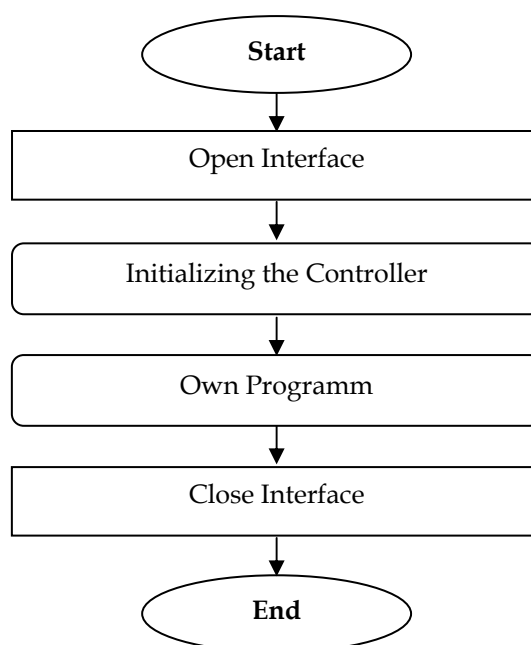


9.3 Notations to create own programs for programming the controller via the API

The following charts show the program flow diagram after that the programs for controlling positioning systems should be made. The used functions are listed in the LSTEP-API description and there they are described more detailed.

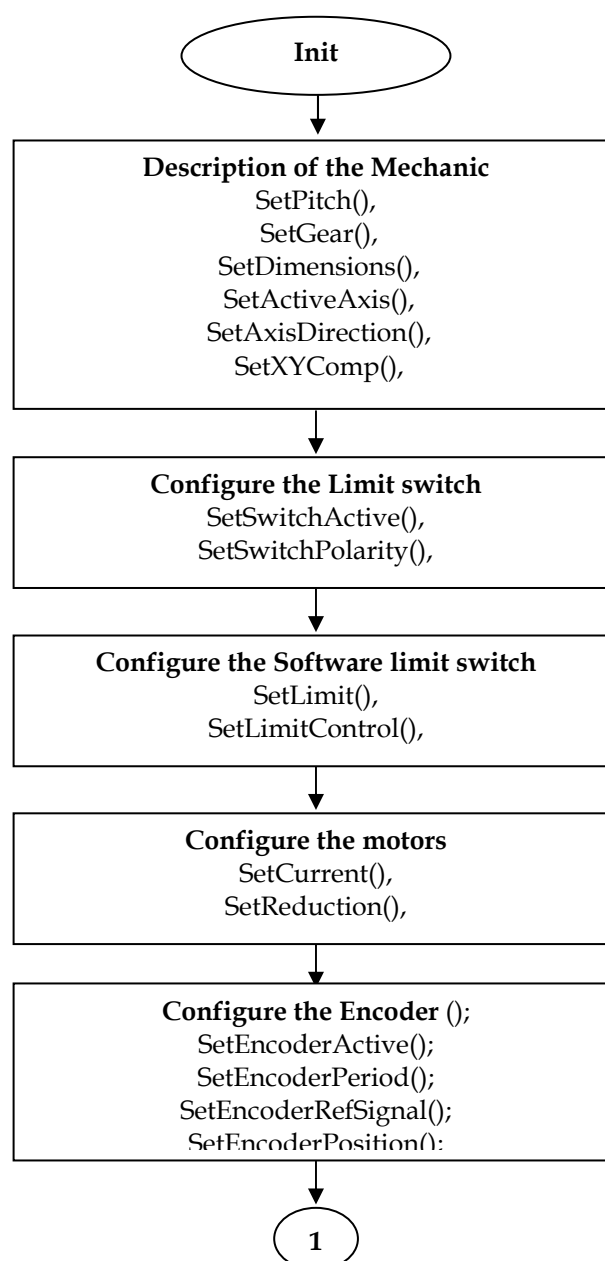
Because the pre-configured default-settings can not contain all data for every application, after the used interface was opened, follow the steps described under item „Initialising the Controller“.

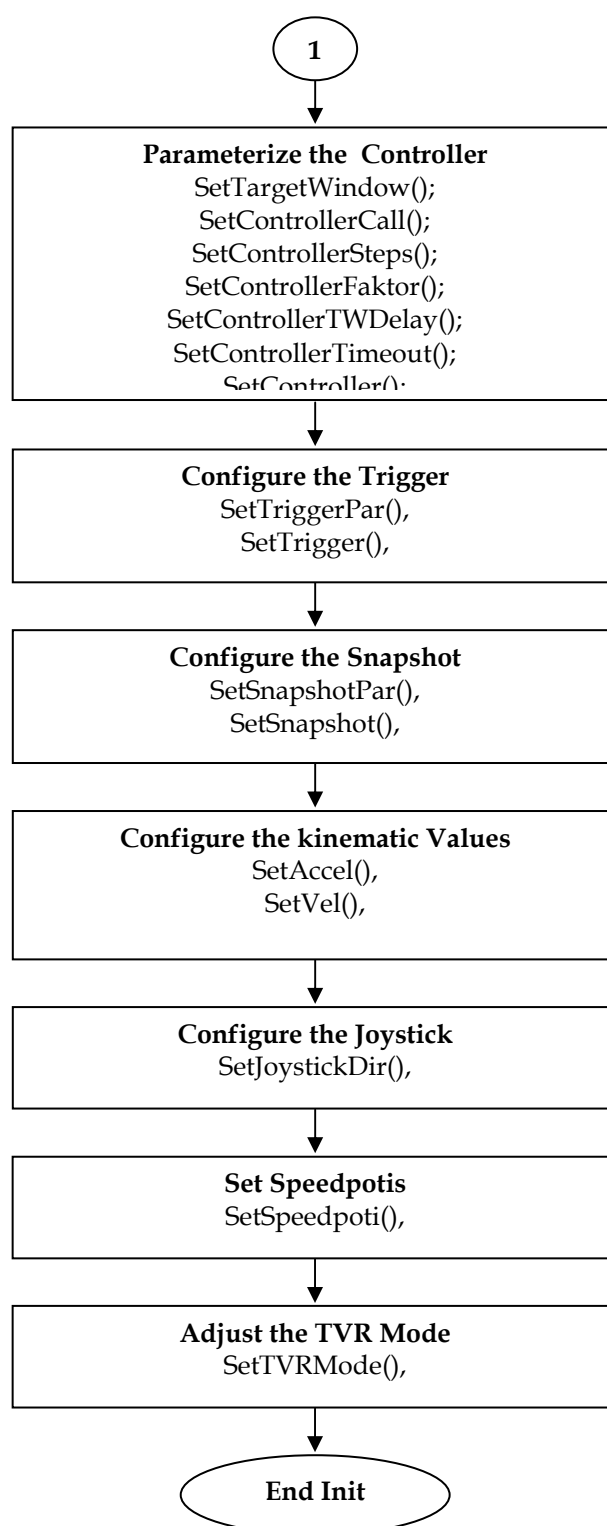
Subsequently under the use of LSTEP-API any user program can be written.



9.3.1 Initialising the Controller

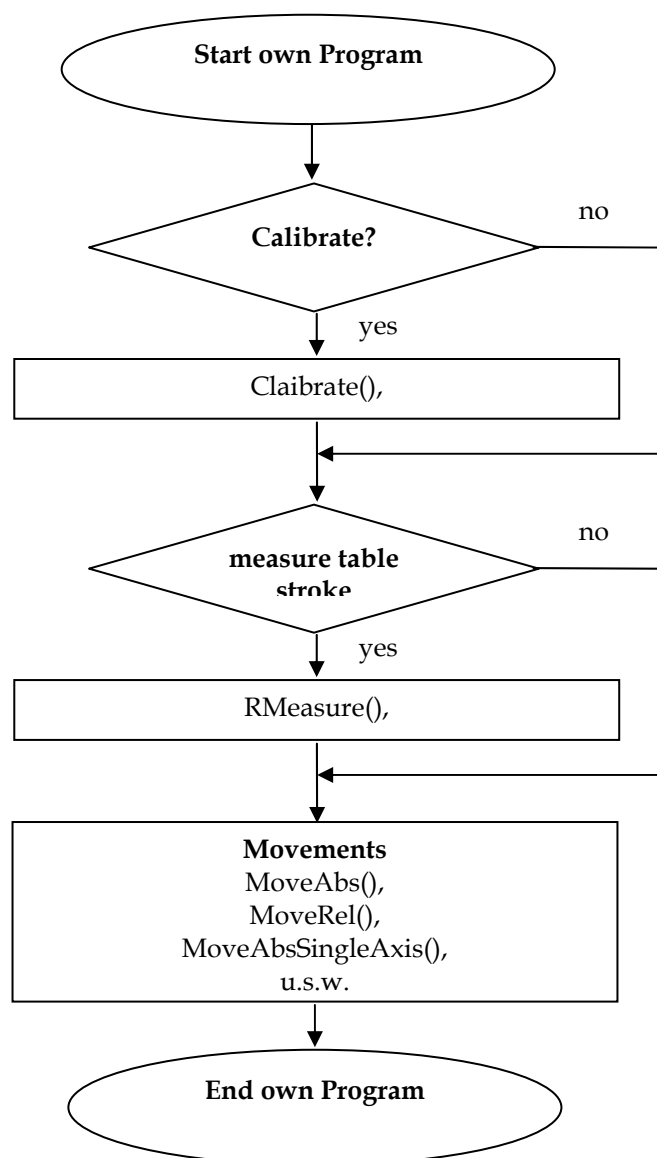
In order to have a fault free operation it is necessary to carry out the basic settings of the positioning controller prior to the start of the own program when initialising the used LSTEP.





9.3.2 Own Program part

In the own program part the user can program the desired functionality of the controller. Such as carrying out positioning movements in dependence of the I/ condition, as well as setting the trigger signal in dependence of the positions etc.



9.4 Functions

9.4.1 Index for API-Commands

Arrangement of the commands as follows:

API-Configuration/Interface

Command	Short description	Page
Connect	Connect with LSTEP	20
ConnectEx	Connect with LSTEP	20
ConnectSimple	Connect with LSTEP	21
CreateLSID	Creates an ID No for the use of the LSTEP4X APIs	22
Disconnect	Disconnect LSTEP	22
EnableCommandRetry	With this function repeated sending of commands can be switched On/Off in case of a fault.	23
FlushBuffer	Delete the input buffer	23
FreeLSID	Sets the created ID No free again	24
LoadConfig	Load LSTEP configuration (interface, axis settings, controllers) from INI-file.	24
SaveConfig	Save LSTEP configuration (interface, axis settings, controllers) into INI-file.	25
SendString	Send string to LSTEP	25
SendStringPosCmd	Moving command, which awaits confirmation , send to LSTEP as a string	26
SetAbortFlag	Set flag to terminate the communication with the LSTEP	26
SetCommandTimeout	Sets the Timeouts for waiting for the feedback signal, of positioning und calibrating.	27
SetControlPars	Transmits the parameters, which were loaded with LS_LoadConfig to the LSTEP.	27
SetCorrTblOff	deactivates axis	27
SetCorrTblOn	activates axis correction in x/y-Matrix with linear interpolation	28
SetExtValue	switch on extensions	29
SetFactorMode	Position value-Conversion for ‚krumme‘ spindle pitch	30
SetLanguage	Set language for LSTEP-API (log / messages)	31
SetProcessMessagesProc	Enables the replacement of the internal message-dispatching procedure of the LStep API	31
SetShowCmdList	LStep-API command list On/Off	32
SetShowProt	Interface protocol On/ Off	32
SetWriteLogText	Switch on / switch off write log file LSTEP4.log (Writing in LSTEP4-log is normally switched off)	32
SetWriteLogTextFN	switch On/Off writing of the interface-protocol in a certain file	33

Controller-Info

Command	Short Description	Page
GetSerialNr	Read serial number of the controller	33
GetVersionStr	Returns the current version number of the Firmware	34
GetVersionStrDet	Read detailed version number of the firmware	34
GetVersionStrInfo	Gives detailed information about version number	35

Settings

Command	Short Description	Page
GetAccel	Inquiry of acceleration	36
GetActiveAxes	Delivers enable axes	37
GetAxisDirection	Inquiry of reverse-turning direction	38
GetCalibBackSpeed	Reads the speed, with which the axis are moved back during calibration	39
GetCaliboffset	Inquiry of calibration-offset	40
GetCalibrateDir	Inquiry reverse preceding sign when calibrating	41
GetCurrentDelay	Indicates time delay for current reduction	42
GetDimensions	Inquiry dimensions of the axes	43
GetGear	Inquiry- gear transmission	44
GetJoystickFilter	Indicates, if the filtering and hysteresis is activated in joystick operation	45
GetMotorCurrent	Inquiry motor current	46
GetMotorTablePatch	Indicates, if the correction table is activated.	47
GetOutFuncLev	Indicates the speed when the current will be switched, from parameterised current to maximum current.	48
GetPitch	delivers spindle pitch	49
GetPowerAmplifier	Indicates if the amplifiers of the LS44 are switched ON or OFF. This command only exists for the LS44-controller.	50
GetReduction	Inquiry of current reduction	51
GetRefSpeed	Reads the reverse speed, the axes move while searching the reference mark.	52
GetRMOffset	Inquiry RM-Offset	53
GetSpeedPoti	Indicates if the potentiometer On/Off	54
GetStopAccel	Delivers the brake acceleration, if the stop input becomes active.	55
GetStopPolarity	Read stop entrance polarity.	56
GetVel	Inquiry speed	57
GetVelFac	Inquiry speed reduction	58
GetVLevel	Delivers the speed limits of the indicated speed range.	59
GetXYAxisComp	Inquiry XY-axis overlay	61
LstepSave	Save current configuration in LStep (EEPROM)	61
SetAccel	Set acceleration	36
SetAccelSingleAxis	Set acceleration for individual axis	62
SetActiveAxes	Enable axes	37
SetAxisDirection	Reverse turning direction	38
SetCalibBackSpeed	Sets the speed, with which the axis are moved back during calibration after reaching the limit switches.	39
SetCaliboffset	Calibration offset	40
SetCalibrateDir	Reverse preceding sign when calibrating	41
SetCurrentDelay	Time delay for current reduction	42
SetDimensions	Set dimensions of the axes	43
SetGear	Program gear transmission	44
SetJoystickFilter	Activating/Deactivating the filtering and hysteresis in joystick operation	45
SetMotorCurrent	Set motor current	46
SetMotorTablePatch	Correction table ON/OFF	47
SetOutFuncLev	Set the current switch speed	48
SetPitch	Set spindle pitch	49
SetPowerAmplifier	Switches the amplifiers of the LS44 On/Off.	50
SetReduction	Set current reduction	51

SetRefSpeed	Sets the reverse speed, the axes move while searching the reference mark.	52
SetRMOffset	RM-Offset	53
SetSpeedPoti	Potentiometer On/ Off	54
SetStopAccel	Delivers the brake acceleration, if the stop input becomes active.	55
SetStopPolarity	Adjust stop entrance polarity.	56
SetVel	Set speed (velocity)	57
SetVelFac	Set speed reduction	58
SetVelSingleAxis	Set speed for individual axis	62
SetVLevel	Exclude speed ranges, in which the system shows resonances.	60
SetXYAxisComp	Activate XY-axis overlay	61
SoftwareReset	Reset the software to starting status	62

Status report

Command	Short Description	Page
GetError	gives the current error number	63
GetSecurityErr	Reads all statuses and results of the GAL-safety monitoring (only with LS44-controller)	64
GetSecurityStatus	Delivers the current statuses the safety monitoring (only with LS44-controller)	65
GetStatus	Gives the current status of the controller	66
GetStatusAxis	Gives the present status of the individual axes	66
GetStatusLimit	Delivers the current condition of the software-limits of each axis.	67
SetAutoStatus	AutoStatus On/Off	67

Moving commands and Position administration

Command	Short Description	Page
Calibrate	Calibrate	68
CalibrateEx	Only the axes are calibrated, whose corresponding bit was set in the transmitted integer-value.	68
Clearpos	Sets the position to 0 (for endless turning axes)	69
GetDelay	Reads the delay of the vector start.	69
GetDistance	Delivers the distance for LS_MoveRelShort	70
GetPos	Inquires the current positions of all axes	71
GetPosEx	Inquires the current encoder or positional values of all axes	71
GetPosSingleAxis	Inquire the current position of an axis	72
MoveAbs	Move to absolute position	73
MoveAbsSingleAxis	Move individual axis to absolute position	73
MoveEx	extended moving command	74
MoveRel	Move to relative vector	75
MoveRelShort	Move to relative position (short command)	75
MoveRelSingleAxis	Move individual axis relatively	76
RMeasure	Measure table stroke	76
RmeasureEx	Measure table stroke (The table stroke is only measured for axes for which the relevant bit has been set in the transmitted integer value).	77
SetDelay	The delay command is used to produce a vector start delay	69
SetDistance	Set distance (for LS_MoveRelShort)	70
SetPos	Set positional values	77
StopAxes	Stop (all movements are stopped)	78
WaitForAxisStop	The function returns, as soon as the selected axes in the bit-mask	78

	A Flag reached its goal position.	
--	-----------------------------------	--

Joystick and Handwheel

Command	Short Description	Page
GetDigJoySpeed	Read out the set speeds	79
GetHandwheel	Read hand wheel condition	80
GetJoystick	Reads the delay of the vector start.	80
GetJoystickDir	Reads motor turning direction for joystick	81
GetJoystickWindow	Read Joystick-window	82
SetDigJoySpeed	Read Digital joystick and speed .	79
SetHandwheelOff	Handwheel Off	83
SetHandwheelOn	Handwheel On	83
SetJoystickDir	Joystick direction	81
SetJoystickOff	Analogue joystick Off	84
SetJoystickOn	Analogue joystick On	84
SetJoystickWindow	set joystick-window	82
GetJoyChangeAxis	Read Joystick allocation of the axes	82
JoyChangeAxis	sets allocation of axes of Joystick	83

Control panel with Trackball and Joyspeed-keys

Command	Short Description	Page
GetBPZ	Reads the condition of the additional control panel with track ball	85
GetBPZJoyspeed	Control panel joystick-speed	86
GetBPZTrackballBackLash	Read out control panel track ball-back lash	87
GetBPZTrackballFactor	Read ot control panal trackball-factor	88
SetBPZ	Control panel On/ Off	85
SetBPZJoyspeed	Control panel joystick-speed	86
SetBPZTrackballBackLash	Control panel trackball-reverse backlash	87
SetBPZTrackballFactor	Control panel trackball-factor	88

Limit switch (Hardware a. Software)

Command	Short Description	Page
GetAutoLimitAfterCalibRM	Indicates if the internal software limits will be set during calibration and table stroke measuring.	89
GetLimit	Set travel limits	90
GetLimitControl	Reads, if travel range monitoring is active	91
GetSwitchActive	Read status of limit switch	92
GetSwitches	Reads the status of all limit switches	93
GetSwitchPolarity	Reads limit switch polarity	94
SetAutoLimitAfterCalibRM	Prevents that the internal software limits are set during calibration and table stroke measuring.	89
SetLimit	Set travel limits	90
SetLimitControl	Control/ monitoring of the range of travel	91
SetSwitchActive	Limit switch On/ Off	92
SetSwitchPolarity	Set limit switch polarity	94

Digital and analogue In.- and Outputs

Command	Short Description	Page
GetAnalogInput	Reads the current status of an analogue channel	95
GetAnalogInputs2	Reads the current status of the analogue channels PT100, MV and V24	95
GetDigitalInputs	Read all input pins	96
GetDigitalInputsE	Read additional digital inputs (16-31)	96
SetAnalogOutput	Set analogue output	96
SetDigIO_Distance	Function of the digital inputs / outputs	97
SetDigIO_EmergencyStop	Function of the digital inputs / outputs Allocating of the Emergency Stop pin	97
SetDigIO_Off	"Off" function of the digital inputs/outputs	98
SetDigIO_Polarity	Set polarity	98
SetDigitalOutput	Set output pin	99
SetDigitalOutputs	Set digital outputs (0-15)	99
SetDigitalOutputsE	Set additional outputs (16-31)	99

Clock pulse Forward / Back

Command	Short Description	Page
GetFactorTVR	Reads factor for clock pulse Forward/ Back	100
GetTVRMode	Read setup of clock pulse Forward / Back (= TVR Mode)	101
SetFactorTVR	Factor for clock pulse Forward/ Back	100
SetTVRMode	Set clock pulse Forward / Back (=TVR Mode)	101

Clock pulse Forward/Back via Interface

Command	Short Description	Page
SetTVRInPulse	Clock pulse Forward / Back via Interface	102

Clock pulse Forward/Back for the additional axes.

Command	Short Description	Page
GetAccelTVRO	Reads the set acceleration for the additional axes.	103
GetPosTVRO	Read position of the additional axis	104
GetStatusTVRO	Delivers the current status of the additional axis	105
GetTVROOutMode	Read settings of the additional axis	106
GetTVROOutPitch	Reads the spindle pitch of the additional axis	107
GetTVROOutResolution	Reads the resolution of the amplifier which is to be controlled	108
GetVelTVRO	Reads the set speed of the additional axis	109
MoveAbsTVROSingleAxis	Position single axis absolute	110
MoveAbsTVRO	Move to absolute position	110
MoveRelTVROSingleAxis	Move single axis absolute	111
MoveRelTVRO	Move relative vector	111
SetAccelSingleAxisTVRO	Acceleration of single additional axis	112
SetAccelTVRO	Set acceleration	103
SetPosTVRO	Set position of the additional axis	104
SetTVROOutMode	Set additional axis X, Y, Z and A, beside the actual main axis X, Y, Z and A	106
SetTVROOutPitch	Sets the spindle pitch for the additional axis	107
SetTVROOutResolution	Sets the resolution of the amplifier which is to be controlled	108
SetVelSingleAxisTVRO	set speed of the additional axis	112

SetVelTVRO	set speed of the additonal axis	109
------------	---------------------------------	-----

Encoder-Settings

Command	Short Description	Page
ClearEncoder	Set encoder-counter to zero	113
GetEncoder	Reads all encoder positions	113
GetEncoderActive	Reads , which encoders are activated after the calibration.	114
GetEncoderMask	Read encoder statuses	115
GetEncoderPeriod	Read length of encoder period	116
GetEncoderPosition	Read encoder position setting	117
GetEncoderRefSignal	Reads if interpret reference signal from encoder when calibration is done	118
SetEncoderActive	This function is used to select which encoder is to be activated after calibration.	114
SetEncoderMask	(de-)activate encoder	115
SetEncoderPeriod	Set length of encoder period	116
SetEncoderPosition	Encoder position display On/Off	117
SetEncoderRefSignal	Interpret reference signal from encoder when calibration is done	118

Controller Setting

Command	Short Description	Page
ClearCtrFastMoveCounter	This function sets Fast Move Counters of all axis to zero.	119
GetController	Read controller mode	120
GetControllerCall	Reads controller call time	121
GetControllerFactor	Reads controller factor	122
GetControllerSteps	Reads controller steps length	123
GetControllerTimeout	Reads controller timeout	124
GetControllerTWDelay	Read controller relay	125
GetCtrFastMove	Reads setting of the Fast Move Function	126
GetCtrFastMoveCounter	Read amount od executed FastMove functions to 0	126
GetTargetWindow	Reads the target window	127
SetController	Set controller mode	120
SetControllerCall	Call controller	121
SetControllerFactor	Controller factor	122
SetControllerSteps	Controller steps	123
SetControllerTimeout	Controller timeout	124
SetControllerTWDelay	Controller delay	125
SetCtrFastMoveOff	Fast Move Funktion „OFF“	128
SetCtrFastMoveOn	Fast Move Funktion „ON“	128
SetTargetWindow	Target window	127

Trigger-Output

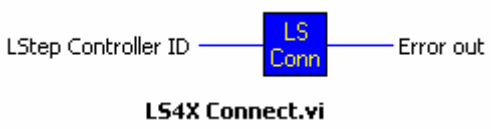
Command	Short Description	Page
GetTrigCount	Read Trigger counter.	129
GetTrigger	Read Trigger setting Einstellung vom Trigger auslesen	130
GetTriggerPar	Reads Trigger-Parameter	131
SetTrigCount	Read Trigger counter	129
SetTrigger	Trigger On/ Off	130
SetTriggerPar	Trigger parameters	131

Snapshot-Input

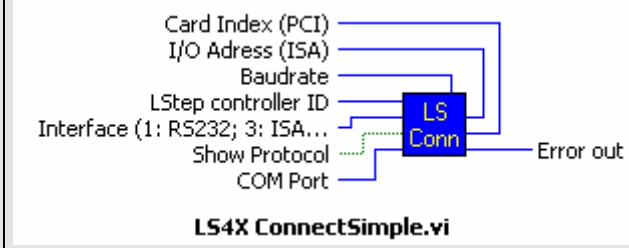
Command	Short Description	Page
GetSnapshot	Reads the current Snapshot-condition	132
GetSnapshotCount	Snapshot counter	133
GetSnapshotFilter	Reads input filter (snapshot-filter)	133
GetSnapshotPar	Read Snapshot-Parameter	134
GetSnapshotPos	Read snapshot position	135
GetSnapshotPosArray	Read snapshot-position from array	135
SetSnapshot	Snapshot On/Off	132
SetSnapshotFilter	Set input filter for rebounding switches.	133
SetSnapshotPar	Snapshot parameters	134

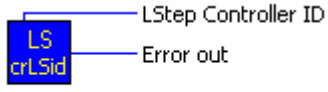
9.4.2 Functions


API-Configuration/Interface

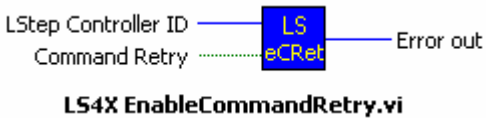
LS_Connect	
Description:	Connect with LSTEP
	<p>Therefore the interfaces-parameters are used, which are loaded from the INI-file via LS_LoadConfig.</p> <p>(One of the functions LS_Connect, LS_ConnectSimple or LS_ConnectEx <u>must</u> be called for initialising of the interface, so that the communication with the LSTEP is possible.)</p>
Delphi:	<pre>function LS_Connect: Integer; function LSX_Connect(LSID: Integer): Integer;</pre>
C++:	<pre>int Connect();</pre>
LabView:	 <p>LS4X Connect.vi</p>
Parameters:	-
Example:	<pre>LS.LoadConfig("C:\LStepTest\LStep.INI"); LS.Connect();</pre>

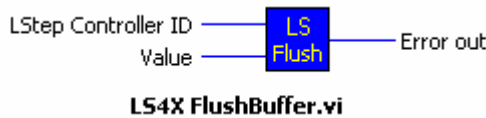
LS_ConnectEx	
Description:	Connect with LSTEP
	<p>This function offers more possibilities, a pointer is transferred to a data structure, that contains the interface parameter. In this record also informations about identified controller (version number...)</p> <p>(One of the functions LS_Connect, LS_ConnectSimple or LS_ConnectEx <u>must</u> be called for initialising of the interface, so that the communication with the LSTEP is possible.)</p>
Delphi:	<pre>function LS_ConnectEx(var AControlInitPar: TLS_ControlInitPar): Integer; function LSX_ConnectEx(LSID: Integer; var AControlInitPar: TLS_ControlInitPar): Integer;</pre>
C++:	<pre>int ConnectEx (TLS_ControlInitPar *pAControlInitPar);</pre>
Parameters:	AControlInitPar: Pointer to a record of the type Typs TLS_ControlInitPar
Example:	<pre>LS.ConnectEx(&ControlInitPar1);</pre>


LS_ConnectSimple	
Description:	<p>Connect with LSTEP</p> <p>The settings of the interface are delivered as a parameter.</p> <p>(One of the functions LS_Connect, LS_ConnectSimple or LS_ConnectEx <u>must</u> be called for initialising of the interface, so that the communication with the LSTEP is possible.)</p>
Delphi:	<pre>function LS_ConnectSimple(AnInterfaceType: Integer; AComName: PChar; ABR: Integer; AShowProt: LongBool): Integer; function LSX_ConnectSimple(LSID: Integer; AnInterfaceType: Integer; AComName: PChar; ABaudRate: Integer; AShowProt: LongBool): Integer;</pre>
C++:	<pre>int Connect (int lAnInterfaceType, char *pcAComName, int lABR, BOOL AShowProt);</pre>
LabView:	 <p>LS4X ConnectSimple.vi</p>
Parameters:	<p>AnInterfaceType: Interface type 1 = RS232 2 = ArcNet 3 = DPRAM / ISA-Bus 4 = DPRAM / PCI-Bus 11= RS232 with RTS/CTS evaluation</p> <p>AComName: Name of the COM-interface, i.e. 'COM2', for ArcNet or DPRAM set to ZERO</p> <p>ABR: Meaning is dependent on the interface type RS232 → Baud rate, z. B. 9600 ArcNet: → 0 für Koax, 1 for Twisted Pair DPRAM / ISA-Bus: → Basis-I/O-Adress of the, for example 0x0340 DPRAM / PCI-Bus: → 0=first card 1=second card</p> <p>AShowProt: determines whether the interfaces protocol is supposed to be indicated</p>
Example:	<pre>LS.ConnectSimple(1, "COM2", 9600, true); // RS232, 9600 Baud or LS_ConnectSimple(4, nil, 0, true); // LStep PCI card 0;</pre>

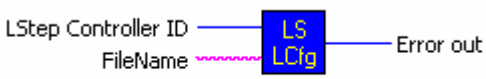
LSX_CreateLSID (nur LSTEP4X-API)	
Description:	Creates a LStep-ID-Number. This is used as an additional parameter in the LSTEP4X-API- commands, in order to select the LStep out of several connected LSteps, on which the command should refer to.
Delphi	function LSX_CreateLSID(var LSID: Integer): Integer;
C++	-
LabView	 <p>LS4X CreateLSID.vi</p>
Parameters:	LSID: contains after calling CreateLSID a new LStep-ID-Number, this number can be used for Connect-, moving commands and other commands
Example:	<pre>var LStep1: Integer; ... LSX_CreateLSID(&LStep1);</pre>


LS_Disconnect	
Description:	<p>Disconnect LSTEP</p> <p>After calling this function, no more commands can be sent to the LSTEP. The function should be called shortly before termination of the program.</p>
Delphi:	<pre>function LS_Disconnect: Integer; function LSX_Disconnect(LSID: Integer): Integer;</pre>
C++:	<pre>int Disconnect ();</pre>
LabView:	 <p>LS4X Disconnect.vi</p>
Parameters:	-
Example:	<pre>LS.Disconnect();</pre>

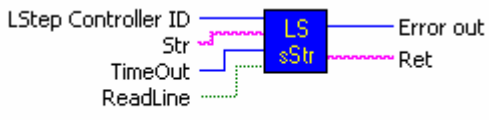
LS_EnableCommandRetry	
Description:	With this function repeated sending of commands can be switched On/Off in case of a fault. (it is turned on as a standard)
Delphi:	function LS_EnableCommandRetry(AValue: LongBool): Integer; function LSX_EnableCommandRetry(LSID: Integer; AValue: LongBool): Integer;
C++:	int EnableCommandRetry (BOOL bAValue);
LabView:	
Parameters:	AValue: true => if faults occur the LStep API repeats the sending of certain commands (especially with WaitForAxisStop) false => switch off repeated sending
Example:	LS.EnableCommandRetry(false) ;

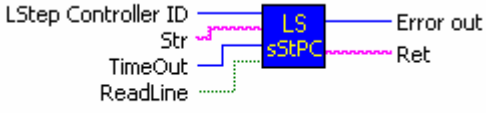
LS_FlushBuffer	
Description:	Delete communication input buffer (RS-232 und PCI) can be used in case of a fault, to erase acknowledgements from the input buffer that are no longer needed.
Delphi:	function LS_FlushBuffer(AValue: Integer): Integer; function LSX_FlushBuffer(LSID: Integer; AValue: Integer): Integer;
C++:	int FlushBuffer (int lAValue);
LabView:	
Parameters:	AValue: currently not used, can be set to =0
Example:	LS.FlushBuffer(0);

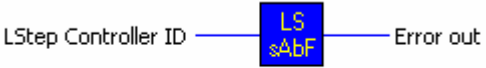
LSX_FreeLSID (only LSTEP4X-API)	
Description:	Releases a created LStep-ID-number. This is used as an additional parameter in the LSTEP4X-API- commands, in order to select the LStep out of several connected LSteps, on which the command should refer to. FreeLSID should be called after Disconnect.
Delphi	function LSX_FreeLSID(LSID: Integer): Integer;
C++	-
LabView	 <p>LS4X FreeLSID.vi</p>
Parameters:	LSID: LStep-ID-number to be released; this may not used after FreeLSID
Example:	<pre>var LStep1: Integer; ... LSX_CreateLSID(&LStep1); LSX_ConnectSimple(LStep1, ...); ... LSX_Disconnect(LStep1); LSX_FreeLSID(LStep1);</pre>


LS_LoadConfig	
Description:	Load LSTEP configuration (interface, axis settings, controllers) from INI-file. The format of the INI-file is compatible with the Win-Commander-INI-file, i.e. the settings can be taken over from the Win-Commander (Wincom4.ini) The loaded configuration is used in the functions LS_Connect and LS_SetControlPars.
Delphi:	function LS_LoadConfig(FileName: PChar): Integer; function LSX_LoadConfig(LSID: Integer; FileName: PChar): Integer;
C++:	int LoadConfig (char *pcFileName);
LabView:	 <p>LS4X LoadConfig.vi</p>
Parameters:	FileName: File name of the INI-file as a zero-terminated string
Example:	LS.LoadConfig("C:\LStepTest\LStep.INI");

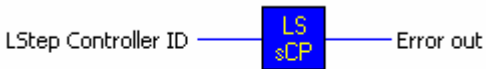
LS_SaveConfig	
Description:	Save LSTEP-configuration (interface, axes setting, controller) in the INI-file. The format of the INI-file is compatible with the Win-Commander-INI-file.
Delphi	function LS_SaveConfig(FileName: PChar): Integer; function LSX_SaveConfig(LSID: Integer; FileName: PChar): Integer;
C++	int SaveConfig (char *pcFileName);
LabView	
Parameters:	FileName: File name of the INI-file as a zero terminating String
Example:	LS.SaveConfig("C:\LStepTest\LStep.INI");

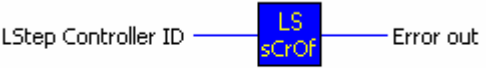
LS_SendString	
Description:	Send string to LSTEP
Delphi:	function LS_SendString(Str, Ret: PChar; MaxLen: Integer; ReadLine: LongBool; TimeOut: Integer): Integer; function LSX_SendString(LSID: Integer; Str, Ret: PChar; MaxLen: Integer; ReadLine: LongBool; TimeOut: Integer): Integer;
C++:	int SendString (char *pcStr,char *pcRet,int lMaxLen,BOOL ReadLine,int lTimeOut);
LabView:	
Parameters:	Str → Zero terminated string which is to be transmitted to the controller.
	Ret → Buffer which contains the LSTEP feedback, if ReadLine = true
	MaxLen → Maximum number of characters which can be copied into the buffer.
	ReadLine → Read LSTEP feedback:
	TimeOut → Maximum time in which feedback must have occurred [ms]
Example:	LS.SendString("?ver\r", pcLStepVer, 256, true, 1000); // Read version number, Timeout 1s


LS_SendStringPosCmd	
Description:	Moving command, which awaits confirmation , send to LSTEP as a string
Delphi:	function LS_SendStringPosCmd(Str, Ret: PChar; MaxLen: Integer; ReadLine: LongBool; TimeOut: Integer): Integer; 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:	 <p>LS4X SendStringPosCmd.vi</p>
Parameters:	<p>Str → Zero terminated String, that is to be send to the controller</p> <p>Ret → Buffer, that contains the acknowledgement of the LSTEP, if ReadLine = true</p> <p>MaxLen → Maximum amount of characters, that can be copied into the buffer.</p> <p>ReadLine → Read acknowledgement of the LSTEP:</p> <p>TimeOut → maximum waiting time for acknowledgement [ms]</p>
Example:	LS.SendStringPosCmd("!moa 1 2\r", pcLStepVer, 256, true, 100000);


LS_SetAbortFlag	
Description:	<p>Set flag to terminate the communication with the LSTEP</p> <p>A function which is still waiting for a feedback from the controller when LS_SetAbortFlag is called (e.g. travel commands), comes back with a fault message.</p> <p>This function is especially useful in programs with message handling routines or several threads, if e.g. a movement is to be aborted quickly.</p>
Delphi:	function LS_SetAbortFlag: Integer; function LSX_SetAbortFlag(LSID: Integer): Integer;
C++:	int SetAbortFlag ();
LabView:	 <p>LS4X SetAbortFlag.vi</p>
Parameters:	-
Example:	<p>LS.SetAbortFlag();</p> <p>LS.StopAxes();</p> <p>(Terminate communication with the LSTEP and send the command to stop all axes)</p>

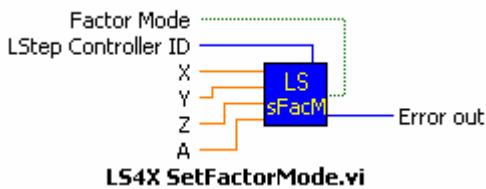
LS_SetCommandTimeout	
Description:	Sets the Timeouts for waiting for the feedback signal, of positioning and calibrating.
Delphi:	function LS_SetCommandTimeout (AtoRead, AtoMove, AtoCalibrate: Integer): Integer; LSX_SetCommandTimeout(LSID: Integer; AtoRead, AtoMove, AtoCalibrate: Integer): Integer;
C++:	int SetCommandTimeout (int lAtoRead, int lAtoMove, int lAtoCalibrate) ;
LabView:	 LS4X SetCommandTimeout.vi
Parameters:	AtoRead: Timeout for waiting for the feedback signal [ms] AtoMove: Timeout für Positioning [ms] AtoCalibrate: Timeout für calibrating [ms]
Example:	LS.SetCommandTimeout (int lAtoRead, int lAtoMove, int lAtoCalibrate) ;


LS_SetControlPars	
Description:	Transmits the parameters which were loaded with LS_LoadConfig to the LSTEP.
Delphi:	function LS_SetControlPars: Integer; function LSX_SetControlPars(LSID: Integer): Integer;
C++:	int SetControlPars ();
LabView:	 LS4X SetControlPars.vi
Parameters:	-
Example:	LS.SetControlPars();

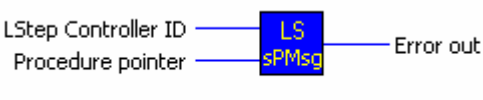
LS_SetCorrTblOff	
Description:	deactivate axis correction
Delphi:	function LS_SetCorrTblOff: Integer; function LSX_SetCorrTblOff(LSID: Integer): Integer;
C++:	int SetCorrTblOff ();
LabView:	 LS4X SetCorrTblOff.vi
Parameters:	-
Example:	LS.SetCorrTblOff() ;

LS_SetCorrTblOn	
Description:	Activate axes correction in x/y-matrix with linear interpolation
Delphi:	function LS_SetCorrTblOn(AFileName: PChar): Integer; function LSX_SetCorrTblOn(LSID: Integer; AFileName: PChar): Integer;
C++:	int SetCorrTblOn (char *pcAFileName);
LabView:	
Parameters:	<p>The correction table is entered manually in the Ini-file. The file name of this file is indicated by AFileName.</p> <p>Structure of a correction table:</p> <p>In the section [Options] the axes correction with linear interpolation is activated via the line „CorrectionXY=1“. XCount and YCount indicate the amount of correction values. The Parameter XDistance determines the distance of measuring points in a row (X-Axis), YDistance the distance of the rows (Y-Axis).</p> <p>The section [CorrTbl] contains the correction values. A corrected position is assigned to each desired value position (x/y-pair of varites), the desired value position must be a point which is determined by the screen XCount, YCount, XDistance and YDistance.</p> <p>The allocations (desired value position=corrected position) can be performed in any desired sequence, important is, that the desired value position is always within the screen. (Zero point of the correction table is (0 0)).</p> <p>Example of a correction table:</p> <pre>[Options] CorrectionXY=1 XCount=3 YCount=3 XDistance=1.0 YDistance=1.0 [CorrTbl] 0.0 0.0=0.0 0.0 1.0 0.0=1.0 0.0 2.0 0.0=2.0 0.0 0.0 1.0=0.0 1.0 1.0 1.0=0.9 1.1 (desired value position x=1 y=1, corrected Position x=0.9 y=1.1) 2.0 1.0=2.0 1.0 0.0 2.0=0.0 2.0 1.0 2.0=1.0 2.0 2.0 2.0=2.0 2.0</pre>
Example:	LS.SetCorrTblOn(„C:\...\corrtbl.ini“);


LS_SetExtValue	
Description:	Switches on the extension of the API, partly it concerns experimental Modes for Debugging purposes.
Delphi:	function LS_SetExtValue(AName: Integer; AValue: Integer): Integer; function LSX_SetExtValue(LSID: Integer; AName, AValue: Integer): Integer;
C++:	int SetExtValue (int lAName, int lAValue);
LabView:	 <p>LS4X SetExtValue.vi</p>
Parameters:	<p>AName: Number of the extended function AValue: Parameter</p> <p>AName=2 (IFSleepTime) setup of the Polling-Interval for the DPRAM of the LStep-PCI AValue: Time-interval in [ms], standard is 10</p> <p>AName=3 (ProtMoveOnly) switches on filter for Log-file, which only protocols Moves&Errors. AValue=1 → Filter on AValue=0 → Filter off</p> <p>AName=4 (Max_LogLn) limits the length of the Log-file, older Log-file will be renamed in .old AValue=Maximale number of line</p> <p>AName=5 (ThreadPriority) changes the priority of Threads of the LStep API. After Connect the Threads are always set to normal Priority, with SetExtValue(5, ...) they can be changed one after the other. AValue=Windows-API-constant for Thread-Priority like THREAD_PRIORITY_ABOVE_NORMAL</p>
Example:	<pre>LS.SetExtValue(3, 1); // Filter for Move-commands on LS.SetExtValue(4, 10000); // maximum length of the Log-file = 10000 lines LS.SetExtValue(5, THREAD_PRIORITY_HIGHEST);</pre>


LS_SetFactorMode	
Description:	Position value-conversion for , odd' spindle pitch
Delphi:	function LS_SetFactorMode(AFactorMode: LongBool; X, Y, Z, A: Double): Integer; 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:	 <p>LS4X SetFactorMode.vi</p>
Parameters:	<p>AFactorMode: Switch on factor-mode</p> <p>This command acitivates a API-internal conversion of the Position values/spindle pitch, to avoid rounding errors with 'odd' spindle pitch</p> <p>X, Y, Z, R: Spindle pitch values, that are transferred to the LStep (if possible values like 1.0 or 4.0, so that a micro step corresponds with a non periodic decimal fraction)</p> <p>Only after SetFactorMode, SetPitch should be called with the actual physical spindel pitch.</p> <p>All moving commands use after calling SetFactorMode and SetPitch a factor-conversion, so that the LStep is positioned correctly.</p> <p>send to LStep Position vector = Positionsvektor * send to LStep spindle pitch /physical spindel pitch</p>
Example:	<pre> LS.SetFactorMode(true, 1, 1, 1, 0); LS.SetPitch(1.234, 1.234, 2.345, 0); LS.MoveAbs(1.234, 2.468, 2.345, 0, true); </pre>


LS_SetLanguage	
Description:	Set language for LSTEP-API (log / messages)
Delphi:	function LS_SetLanguage(PLN: PChar): Integer; function LSX_SetLanguage(LSID: Integer; PLN: PChar): Integer;
C++:	int SetLanguage (char *pcPLN);
LabView:	
Parameters:	<p>PLN: Language (Abbreviated, e.g. "DEU" or "ENG")</p> <p>The appropriate text file (LSTEP4deu.txt or LSTEP4eng.txt) must be in the program directory</p>
Example:	LS.SetLanguage('ENG');

LS_SetProcessMessagesProc	
Description:	<p>Enables the replacement of the internal message-dispatching procedure of the LStep API.</p> <p>The LStep API processes during waiting for confirmation of the LStep in the main-thread messages. If you want to switch of the Message-Dispatching or replace with your own Code, you can use SetProcessMessagesProc for using a callback-procedure.</p>
Delphi:	function LS_SetProcessMessagesProc(Proc: Pointer): Integer; function LSX_SetProcessMessagesProc(LSID: Integer; Proc: Pointer): Integer;
C++:	int SetProcessMessagesProc (void* pProc);
LabView:	
Parameters:	<p>pProc must be a pointer to a stdcall-procedure without a parameter :</p> <p>void MyProcessMessages () { .. }</p>
Example:	LS.SetProcessMessagesProc (&MyProcessMessages);


LS_SetShowCmdList	
Description:	LStep-API Command list On/Off
Delphi	function LS_SetShowCmdList>ShowCmdList: LongBool): Integer; function LSX_SetShowCmdList(LSID: Integer; ShowCmdList: LongBool): Integer;
C++	int SetShowCmdList (BOOL bShowCmdList);
LabView	-
Parameters:	ShowProt: Indicates, if the window „LStep-API command list“ should be shown
Example:	LS.SetShowCmdList(true); // shows interface-protocol if not visible


LS_SetShowProt	
Description:	Interface protocol On/Off
Delphi:	function LS_SetShowProt>ShowProt: LongBool): Integer; function LSX_SetShowProt(LSID: Integer; ShowProt: LongBool): Integer;
C++:	int SetShowProt (BOOL ShowProt);
LabView:	
Parameters:	ShowProt: Specifies whether the window “Interface Protocol” is to be shown or not
Example:	LS.SetShowProt(true); // Show interface protocol if not already visible


LS_SetWriteLogText	
Description:	Switch on / switch off write log file LSTEP4.log (Writing in LSTEP4-log is normally switched off)
Delphi:	function LS_SetWriteLogText(AWriteLogText: LongBool): Integer; function LSX_SetWriteLogText(LSID: Integer; AWriteLogText: LongBool): Integer;
C++:	int SetWriteLogText (BOOL AWriteLogText);
LabView:	
Parameters:	-
Example:	LS.SetWriteLogText (true);

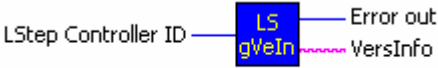
LS_SetWriteLogTextFN	
Description:	switch On/Off the writing a interface –protocol in certain file (Standard mode the writing is turned off.)
Delphi:	function LS_SetWriteLogTextFN(AWriteLogText: LongBool; ALogFN: PChar): Integer; function LSX_SetWriteLogTextFN(LSID: Integer; AWriteLogText: LongBool; ALogFN: PChar): Integer;
C++:	int SetWriteLogTextFN (BOOL bAWriteLogText, char *pcALogFN);
LabView:	
Parameters:	AWriteLogText: true => write protocol file ALogFN: filename of the protocol file
Example:	LS.SetWriteLogTextFN(true, „C:\Temp\prot.txt“);

Controller-Info

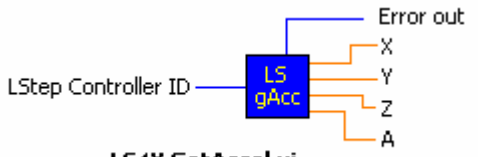
LS_GetSerialNr	
Description:	Read serial number of the controller
Delphi:	function LS_GetSerialNr(SerialNr: PChar; MaxLen: Integer): Integer; function LSX_GetSerialNr(LSID: Integer; SerialNr: PChar; MaxLen: Integer): Integer;
C++:	int GetSerialNr (char *pcSerialNr,int lMaxLen);
LabView:	
Parameters:	SerialNr: Pointer to a buffer in which the serial number is returned MaxLen: Maximum number of characters which can be copied into the buffer
Example:	LS.GetSerialNr(pcSerialNr, 256);

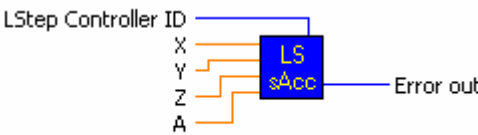
LS_GetVersionStr	
Description:	Returns the current version number of the Firmware
Delphi:	function LS_GetVersionStr(Vers: PChar; MaxLen: Integer): Integer; function LSX_GetVersionStr(LSID: Integer; Vers: PChar; MaxLen: Integer): Integer;
C++:	int GetVersionStr (char *pcVers,int lMaxLen);
LabView:	 <p>LS4X GetVersionStr.vi</p>
Parameters:	Stat: Pointer to a buffer in which the version string is returned MaxLen: Maximum number of characters which can be copied into the buffer
Example:	LS.GetVersionStr(pcVers, 64); // Read version number

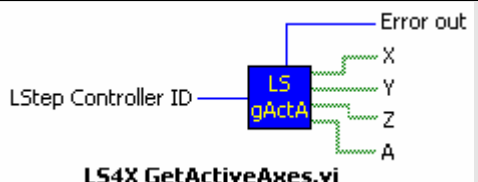
LS_GetVersionStrDet	
Description:	Read out detailed version number of Firmware
Delphi:	function LS_GetVersionStrDet(VersDet: PChar; MaxLen: Integer): Integer; function LSX_GetVersionStrDet(LSID: Integer; VersDet: PChar; MaxLen: Integer): Integer;
C++:	int GetVersionStrDet (char *pcVersDet, int lMaxLen);
LabView:	 <p>LS4X GetVersionStrDet.vi</p>
Parameters:	VersDet: Points to buffer, where the detailed version-string is returned to MaxLen: Maximum allowed amount of characters, that can be copied into the buffer.
Example:	LS.GetVersionStrDet(pcVersDet, 64); // read out detailed version number

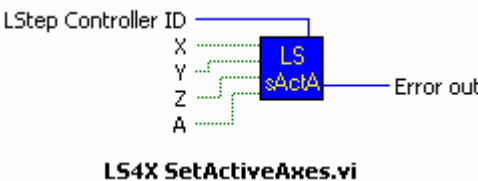
LS_GetVersionStrInfo	
Description:	Gives detailed information about version number
Delphi:	function LS_GetVersionStrInfo (VersInfo: PChar; MaxLen: Integer): Integer; function LSX_GetVersionStrInfo (LSID: Integer; VersInfo: PChar; MaxLen: Integer): Integer;
C++:	int GetVersionStrInfo (char *pcVersInfo, int lMaxLen);
LabView:	 <p>LS4X GetVersionStrInfo.vi</p>
Parameters:	<p>VersInfo: Points to buffer, in which the day of the week, calender week, year, consecutive number is returned to</p> <p>i. e.: T04.35.02-0004</p> <p>MaxLen: Maximum allowed amount of characters, that can be copied into the buffer.</p>
Example:	LS.GetVersionStrInfo (pcVersInfo, 64);

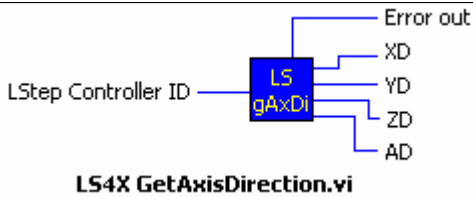
Settings

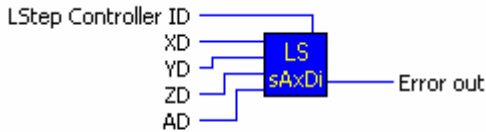
LS_GetAccel	
Description:	Inquiry of acceleration
Delphi:	function LS_GetAccel(var X, Y, Z, R: Double): Integer; function LSX_GetAccel(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetAccel (double *pdX, double *pdY, double *pdZ, double *pdA);
LabView:	 LS4X GetAccel.vi
Parameters:	X, Y, Z, A: Acceleration values [m/s ²]
Example:	LS.GetAccel(&X, &Y, &Z, &A);

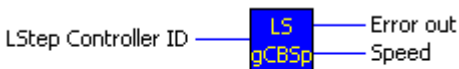
LS_SetAccel	
Description:	Set acceleration
Delphi:	function LS_SetAccel(X, Y, Z, R: Double): Integer; function LSX_SetAccel(LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetAccel(double dX, double dY, double dZ, double dA);
LabView:	 LS4X SetAccel.vi
Parameters:	X, Y, Z and A 0.01 – 10.00 [m/s ²]
Example:	LS.SetAccel(1.0, 1.5, 0, 0);


LS_GetActiveAxes	
Description:	Delivers enable axes
Delphi:	function LS_GetActiveAxes(var Flags: Integer): Integer; function LSX_GetActiveAxes(LSID: Integer; var Flags: Integer): Integer;
C++:	int GetActiveAxes (int *pIFlags);
LabView:	 LS4X GetActiveAxes.vi
Parameters:	Flags: 32-bit-Integer which after activation of the function in the Bits 0-4 contains the bit-mask. Bit 0 = 1 → X-axis axis enabled Bit 2 = 0 → Z-axis not enabled
Example:	LS.GetActiveAxes(&Flags);

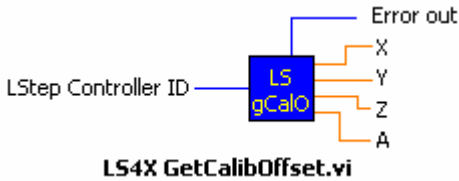
LS_SetActiveAxes	
Description:	Enable axes
Delphi:	function LS_SetActiveAxes(Flags: Integer): Integer; function LSX_SetActiveAxes(LSID: Integer; Flags: Integer): Integer;
C++:	int SetActiveAxes(int Flags);
LabView:	 LS4X SetActiveAxes.vi
Parameters:	Flags: Bit mask Bit 0 = 1 → X-axis enabled, i.e. can be travelled Bit 2 = 0 → Z-axis not enabled
Example:	LS.SetActiveAxes(3); /* Enable X- and Y-axes (Bits 0 and. 1 set), do not enable Z-axis (Bit 2 = 0) */

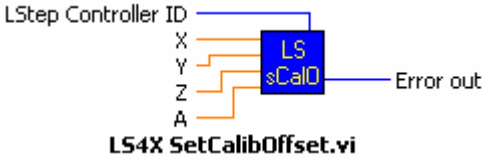
LS_GetAxisDirection	
Description:	Inquiry of reverse-turning direction
Delphi:	function LS_GetAxisDirection(var XD, YD, ZD, AD: Integer): Integer; function LSX_GetAxisDirection(LSID: Integer; var XD, YD, ZD, AD: Integer): Integer;
C++:	int GetAxisDirection (int *plXD, int *plyD, int *plZD, int *plAD);
LabView:	 <p style="text-align: center;">LS4X GetAxisDirection.vi</p>
Parameters:	XY, YD, ZD, AD: 0 => normal turning direction Drehrichtung 1 => Reverse-turning direction
Example:	LS.GetAxisDirection(&XD, &YD, &ZD, &AD);

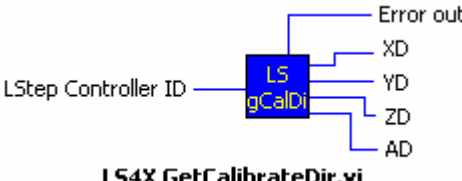
LS_SetAxisDirection	
Description:	Reverse-turning direction
Delphi:	function LS_SetAxisDirection(XD, YD, ZD, AD: Integer): Integer; function LSX_SetAxisDirection(LSID: Integer; XD, YD, ZD, AD: Integer): Integer;
C++:	int SetAxisDirection (int lXD, int lYD, int lZD, int lAD);
LabView:	 <p style="text-align: center;">LS4X SetAxisDirection.vi</p>
Parameters:	XY, YD, ZD, AD: 0 => normal turning direction Drehrichtung 1 => Reverse-turning direction
Example:	LS.SetAxisDirection(1, 0, 0, 0); // Reverse turning direction of X-Achse

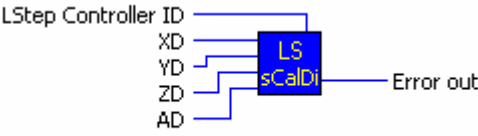
LS_GetCalibBackSpeed	
Description:	Reads the rotational speed, with which the axis are moved back during calibration after reaching the limit switches. The speed equivalent to the issued value * 0.01 U/s.
Delphi:	function LS_GetCalibBackSpeed(var ISpeed: Integer): Integer; function LSX_GetCalibBackSpeed (LSID: Integer; var ISpeed: Integer): Integer;
C++:	int GetCalibBackSpeed (int *pISpeed);
LabView:	 LS4X GetCalibBackSpeed.vi
Parameters:	ISpeed: Speed value
Example:	LS. GetCalibBackSpeed (&ISpeed);

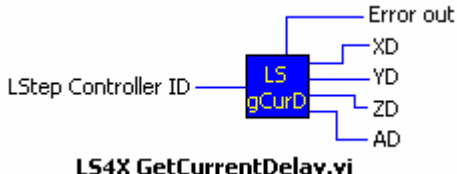
LS_SetCalibBackSpeed	
Description:	Sets the rotational speed, with which the axis are moved back during calibration after reaching the limit switches. The speed to equivalent to the issued value * 0.01 U/s.v
Delphi:	function LS_SetCalibBackSpeed(ISpeed: Integer): Integer; function LSX_SetCalibBackSpeed (LSID: Integer; ISpeed: Integer): Integer;
C++:	int SetCalibBackSpeed (int ISpeed);
LabView:	 LS4X SetCalibBackSpeed.vi
Parameters:	ISpeed: Speed, value range 5 to 100
Example:	LS. SetCalibBackSpeed (10); //The limit switches are left with 0.1 U/s.during calibration after connecting them

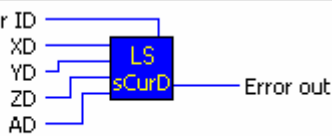
LS_GetCalibOffset	
Description:	Inquiry of calibration-offset
Delphi:	function LS_GetCalibOffset(var X, Y, Z, A: Double): Integer; function LSX_GetCalibOffset(LSID: Integer; var X, Y, Z, R: Double): Integer;
C++:	int GetCalibOffset (double *pdX, double *pdY, double *pdZ, double *pdR);
LabView:	
Parameters:	X, Y, Z, A: calibration-offset dependent on dimension.
Example:	LS.GetCalibOffset(&X, &Y, &Z, &A);

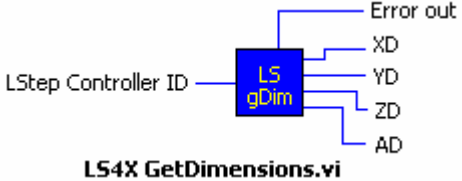
LS_SetCalibOffset	
Description:	Calibration offset
Delphi:	function LS_SetCalibOffset(X, Y, Z, A: Double): Integer; function LSX_SetCalibOffset(LSID: Integer; X, Y, Z, R: Double): Integer;
C++:	int SetCalibOffset (double dX,double dY,double dZ,double dA);
LabView:	
Parameters:	X, Y, Z and A 0 – 32*50000 (32*Spindle pitch)
Example:	LS.SetCalibOffset(1, 1, 1, 1); (When calibration is done, the X-, Y- and Z- axes are each moved 1 mm (for Dim. 2 2 2) away from the zero limit switch towards the center of the table and the zero position is then set (software limit).

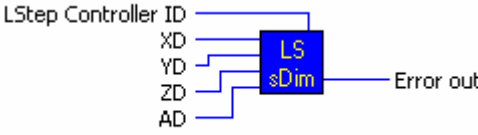
LS_GetCalibrateDir	
Description:	Inquiry reverse preceding sign when calibrating
Delphi:	function LS_GetCalibrateDir(var XD, YD, ZD, AD: Integer): Integer; function LSX_GetCalibrateDir(LSID: Integer; var XD, YD, ZD, AD: Integer): Integer;
C++:	int GetCalibrateDir (int *plXD, int *plyD, int *plZD, int *plAD);
LabView:	
Parameters:	XD, YD, ZD, AD: 0 => no reversing of preceding sign 1 => reverse preceding sign Vorzeichen-Umkehr
Example:	LS.GetCalibrateDir(&XD, &YD, &ZD, &AD);

LS_SetCalibrateDir	
Description:	Reverse preceding sign when calibrating
Delphi:	function LS_SetCalibrateDir(XD, YD, ZD, AD: Integer): Integer; function LSX_SetCalibrateDir(LSID: Integer; XD, YD, ZD, AD: Integer): Integer;
C++:	int SetCalibrateDir (int lXD, int lYD, int lZD, int lAD);
LabView:	
Parameters:	XD, YD, ZD, AD: 0 => no reversing of preceding sign 1 => reverse preceding sign Vorzeichen-Umkehr
Example:	LS.SetCalibrateDir(1, 1, 0, 0);

LS_GetCurrentDelay	
Description:	Indicates time delay for current reduction
Delphi:	function LS_GetCurrentDelay(var X, Y, Z, R: Integer): Integer; function LSX_GetCurrentDelay(LSID: Integer; var X, Y, Z, R: Integer): Integer;
C++:	int GetCurrentDelay (int *plX, int *plY, int *plZ, int *plR);
LabView:	 <p style="text-align: center;">LS4X GetCurrentDelay.vi</p>
Parameters:	X, Y, Z, R: Time delay in ms
Example:	LS.SetCurrentDelay(&X, &Y, &Z, &A);

LS_SetCurrentDelay	
Description:	Time delay for current reduction
Delphi:	function LS_SetCurrentDelay(X, Y, Z, R: Integer): Integer; function LSX_SetCurrentDelay(LSID: Integer; X, Y, Z, R: Integer): Integer;
C++:	int SetCurrentDelay (int lX, int lY, int lZ, int lR);
LabView:	 <p style="text-align: center;">LS4X SetCurrentDelay.vi</p>
Parameters:	X, Y, Z, R: 0-10000 [ms]
Example:	LS.SetCurrentDelay(100, 300, 1000, 0) ;

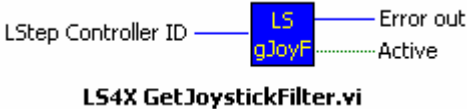
LS_GetDimensions	
Description:	Inquiry dimensions of the axes
Delphi:	function LS_GetDimensions(var XD, YD, ZD, AD: Integer): Integer; function LSX_GetDimensions(LSID: Integer; var XD, YD, ZD, AD: Integer): Integer;
C++:	int GetDimensions (int *plXD, int *plYD, int *plZD, int *plAD);
LabView:	 <p>LS4X GetDimensions.vi</p>
Parameters:	XD, YD, ZD, AD: Dimension values 0 → Microsteps 1 → μm 2 → Millimeters 3 → Degrees 4 → Revolutions
Example:	LS. GetDimensions (&XD, &YD, &ZD, &AD);


LS_SetDimensions	
Description:	Set dimensions of the axes
Delphi:	function LS_SetDimensions(XD, YD, ZD, AD: Integer): Integer; function LSX_SetDimensions(LSID: Integer; XD, YD, ZD, AD: Integer): Integer;
C++:	int SetDimensions (int lXD,int lYD,int lZD,int lAD);
LabView:	 <p>LS4X SetDimensions.vi</p>
Parameters:	Dimensions of the X, Y, Z and A-axes: 0 → Microsteps 1 → μm 2 → Millimeters 3 → Degrees 4 → Revolutions
Example:	LS.SetDimensions(3, 2, 2);

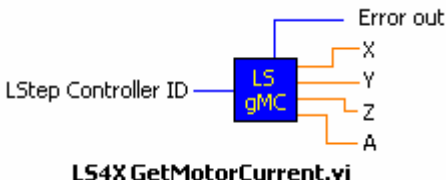
	// X-axis in degrees; Y and Z in mm
--	-------------------------------------

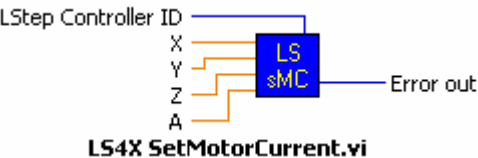
LS_GetGear	
Description:	Inquiry- gear transmission
Delphi:	function LS_GetGear(var X, Y, Z, A: Double): Integer; function LSX_GetGear(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetGear (double *pdX, double *pdY, double *pdZ, double *pdA);
LabView:	<p>LS4X GetGear.vi</p>
Parameters:	X, Y, Z, A: gear transmission values
Example:	LS.GetGear (&X, &Y, &Z, &A);

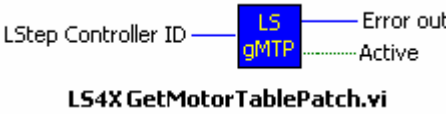
LS_SetGear	
Description:	Program gear transmission
Delphi:	function LS_SetGear(X, Y, Z, A: Double): Integer; function LSX_SetGear(LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetGear (double dX,double dY,double dZ,double dA);
LabView:	<p>LS4X SetGear.vi</p>
Parameters:	X, Y, Z and A 0.01 – 1000
Example:	LS.SetGear(4.0, 2.0, 1.0, 1.0); /* Gear transmissions of ¼ for Z, ½ for Y and 1/1 for Z and A are programmed */

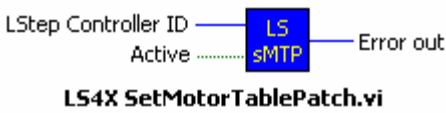
LS_GetJoystickFilter	
Description:	Indicates, if the filtering and hysteresis is activated in joystick operation
Delphi:	function LS_GetJoystickFilter(var bActive: LongBool): Integer; function LSX_GetJoystickFilter (LSID: Integer; var bActive: LongBool): Integer;
C++:	int GetJoystickFilter (BOOL *pbActive);
LabView:	
Parameters:	bActive: True – filtering activates False – deactivated
Example:	LS.SetJoystickFilter (&Active);

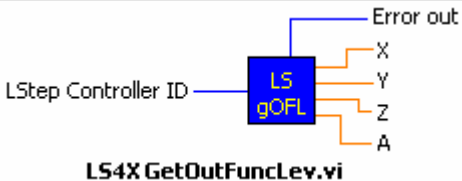
LS_SetJoystickFilter	
Description:	Activating/Deactivating the filtering and hysteresis in joystick operation
Delphi:	function LS_SetJoystickFilter(bActive: LongBool): Integer; function LSX_SetJoystickFilter (LSID: Integer; bActive: LongBool): Integer;
C++:	int SetJoystickFilter (BOOL bActive);
LabView:	
Parameters:	bActive: True – filtering activates False – deactivated
Example:	LS.SetJoystickFilter (True);

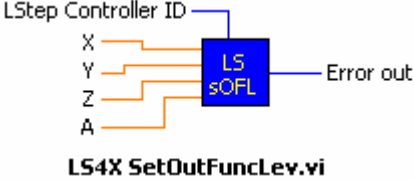
LS_GetMotorCurrent	
Description:	Inquiry motor current
Delphi:	function LS_GetMotorCurrent(var X, Y, Z, A: Double): Integer; function LSX_GetMotorCurrent(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetMotorCurrent (double *pdX, double *pdY, double *pdZ, double *pdA);
LabView:	 LS4X GetMotorCurrent.vi
Parameters:	X, Y, Z, A: Motor current [A]
Example:	LS.GetMotorCurrent(&X, &Y, &Z, &A);

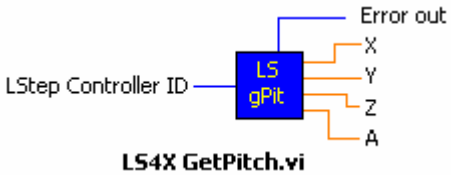
LS_SetMotorCurrent	
Description:	Set motor current
Delphi:	function LS_SetMotorCurrent(X, Y, Z, A: Double): Integer; function LSX_SetMotorCurrent(LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetMotorCurrent (double dX,double dY,double dZ,double dA);
LabView:	 LS4X SetMotorCurrent.vi
Parameters:	Motor current X, Y, Z, A-axis [A]
Example:	LS.SetMotorCurrent(1.5, 1.5, 1.0, 1.0); // Motor current for X and Y is 1.5 amperes; for Z and A, 1.0 amperes

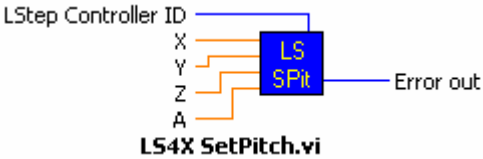
LS_GetMotorTablePatch	
Description:	Indicates, if the correction table is activated.
Delphi:	function LS_GetMotorTablePatch(bActive: var LongBool): Integer; function LSX_GetMotorTablePatch (LSID: Integer; var bActive: LongBool): Integer;
C++:	int GetMotorTablePatch (BOOL *pbActive);
LabView:	
Parameters:	bActive: True – table is activated False – deactivated
Example:	LS. GetMotorTablePatch (&Active);

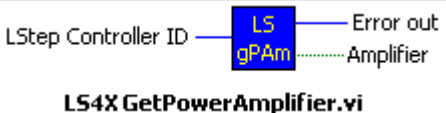
LS_SetMotorTablePatch	
Description:	The correction table becomes activated The correction table was determined for a special motor by measurement. Correction tables can be determined on customer wish.
Delphi:	function LS_SetMotorTablePatch(bActive: LongBool): Integer; function LSX_SetMotorTablePatch (LSID: Integer; bActive: LongBool): Integer;
C++:	int SetMotorTablePatch (BOOL bActive);
LabView:	
Parameters:	bActive: True – table is activated False – deactivated
Example:	LS. SetMotorTablePatch (True);

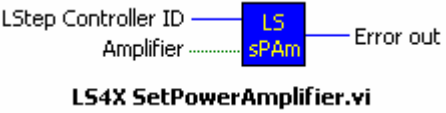
LS_GetOutFuncLev	
Description:	Indicates the speed when the current will be switched, from parameterised current to maximum current.
Delphi:	function LS_GetOutFuncLev(var X, Y, Z, R: Double): Integer; function LSX_GetOutFuncLev (LSID: Integer; var X, Y, Z, R: Double): Integer;
C++:	int GetOutFuncLev (double *pdX, double *pdY, double *pdZ, double *pdR);
LabView:	 LS4X GetOutFuncLev.vi
Parameters:	X, Y, Z, R: Speed in rp/s
Example:	LS.GetCurrentDelay(&X, &Y, &Z, &A);

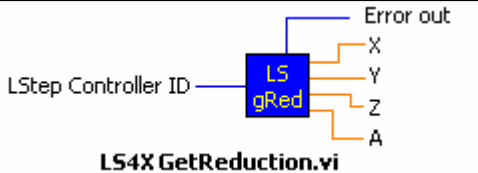
LS_SetOutFuncLev	
Description:	If the set speed is exceeded the current will be switched, from parameterised current to maximum current.
Delphi:	function LS_SetOutFuncLev(X, Y, Z, R: Double): Integer; function LSX_SetOutFuncLev (LSID: Integer; X, Y, Z, R: Double): Integer;
C++:	int SetOutFuncLev (double dX, double dY, double dZ, double dR);
LabView:	 LS4X SetOutFuncLev.vi
Parameters:	X, Y, Z, R: Speed in rp/s
Example:	LS.SetCurrentDelay(25, 25, 25, 25); /* Bei allen Achsen erfolgt eine Stromschaltung bei 25 U/s */

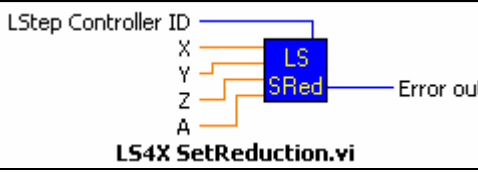
LS_GetPitch	
Description:	delivers spindle pitch
Delphi:	function LS_GetPitch(var X, Y, Z, R: Double): Integer; function LSX_GetPitch(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetPitch (double *pdX, double *pdY, double *pdZ, double *pdA);
LabView:	
Parameters:	X, Y, Z, A: Spindle pitch [mm]
Example:	LS.GetPitch (&X, &Y, &Z, &A);


LS_SetPitch	
Description:	Set spindle pitch
Delphi:	function LS_SetPitch(X, Y, Z, R: Double): Integer; function LSX_SetPitch(LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetPitch(double dX, double dY, double dZ, double dA);
LabView:	
Parameters:	X, Y, Z and A 0.001 – 68 [mm]
Example:	LS.SetPitch(4, 4, 4, 4); // Set spindle pitches to 4 mm for all axes

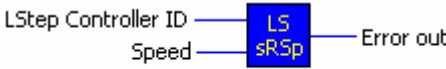
LS_GetPowerAmplifier	
Description:	Indicates if the amplifiers of the LS44 are switched ON or OFF. This command only exists for the LS44-controller.
Delphi:	function LS_GetPowerAmplifier (bAmplifier: var LongBool): Integer; function LSX_GetPowerAmplifier (LSID: Integer; var bAmplifier: LongBool): Integer;
C++:	int GetPowerAmplifier (BOOL *pbAmplifier);
LabView:	
Parameters:	Amplifier: True – the amplifiers are switched on False – switched off
Example:	LS. GetPowerAmplifier (&Amplifier);

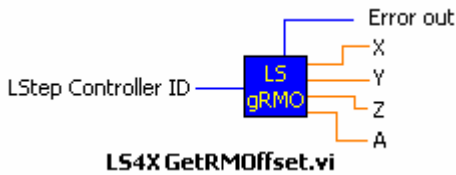
LS_SetPowerAmplifier	
Description:	15
Delphi:	function LS_SetPowerAmplifier (bAmplifier: LongBool): Integer; function LSX_SetPowerAmplifier (LSID: Integer; bAmplifier: LongBool): Integer;
C++:	int SetPowerAmplifier (BOOL bAmplifier);
LabView:	
Parameters:	bAmplifier: True – On False – Off
Example:	LS. SetPowerAmplifier (True); // The amplifiers are switched on

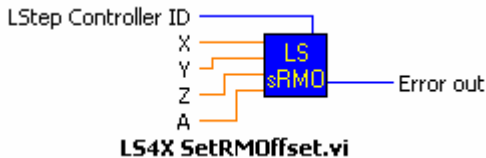
LS_GetReduction	
Description:	Inquiry of current reduction
Delphi:	function LS_GetReduction(var X, Y, Z, R: Double): Integer; function LSX_GetReduction(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetReduction (double *pdX, double *pdY, double *pdZ, double *pdA);
LabView:	
Parameters:	X, Y, Z, A: Current reduction
Example:	LS.GetReduction(&X, &Y, &Z, &A);


LS_SetReduction	
Description:	Set current reduction In quiescent state the rated motor current is reduced to the parameterized ratio.
Delphi:	function LS_SetReduction(X, Y, Z, R: Double): Integer; function LSX_SetReduction(LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetReduction(double dX, double dY, double dZ, double dA);
LabView:	
Parameters:	X, Y, Z and A 0 - 1.0
Example:	LS.SetReduction(0.1, 0.7, 0.5, 0.5); /* Quiescent current for X-axis = 0.1*Rated current; Y-axis = 0.7*rated current ... */


LS_GetRefSpeed	
Description:	Reads the reverse speed, the axes move while searching the reference mark. The speed is equivalent to the issued value * 0.01 rp/s.
Delphi:	function LS_GetRefSpeed(var ISpeed: Integer): Integer; function LSX_GetRefSpeed (LSID: Integer; var ISpeed: Integer): Integer;
C++:	int GetRefSpeed (int *pISpeed);
LabView:	 <p style="text-align: center;">LS4X GetRefSpeed.vi</p>
Parameters:	ISpeed: Speed value
Example:	LS. GetRefSpeed (&lSpeed);

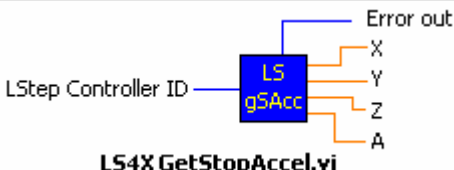
LS_SetRefSpeed	
Description:	Sets the reverse speed, the axes move while searching the reference mark. The speed is equivalent to the issued value * 0.01 rp/s.
Delphi:	function LS_SetRefSpeed(ISpeed: Integer): Integer; function LSX_SetRefSpeed (LSID: Integer; ISpeed: Integer): Integer;
C++:	int SetRefSpeed (int ISpeed);
LabView:	 <p style="text-align: center;">LS4X SetRefSpeed.vi</p>
Parameters:	ISpeed: Speed , value range 0 to 100
Example:	LS. SetRefSpeed (10); //Speed while searching the reference mark is 0.1 rp/s

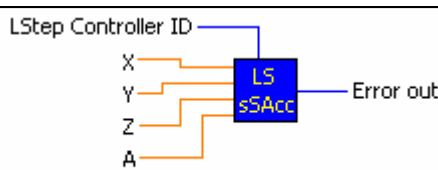
LS_GetRMOffset	
Description:	Inquiry RM-Offset
Delphi:	function LS_GetRMOffset(var X, Y, Z, A: Double): Integer; function LSX_GetRMOffset(LSID: Integer; var X, Y, Z, R: Double): Integer;
C++:	int GetRMOffset (double *pdX, double *pdY, double *pdZ, double *pdR);
LabView:	
Parameters:	X, Y, Z and A: RM-Offset, depending on dimension
Example:	LS.GetRMOffset(&X, &Y, &Z, &A);


LS_SetRMOffset	
Description:	RM Offset
Delphi:	function LS_SetRMOffset(X, Y, Z, A: Double): Integer; function LSX_SetRMOffset(LSID: Integer; X, Y, Z, R: Double): Integer;
C++:	int SetRMOffset (double dX,double dY,double dZ,double dA);
LabView:	
Parameters:	X, Y, Z and A 0 - 32*50000 (32*spindle pitch)
Example:	LS.SetRMOffset(1, 1, 1, 1); (When the table stroke is measured, the X, Y, and Z-axes are each moved 1 mm (for Dim 2 2 2) away from the end limit switch towards the center of the table and the software limit is then set.


LS_GetSpeedPoti	
Description:	Indicates if the potentiometer On/Off
Delphi:	function LS_GetSpeedPoti(var SpePoti: LongBool): Integer; function LSX_GetSpeedPoti(LSID: Integer; var SpePoti: LongBool): Integer;
C++:	int GetSpeedPoti (BOOL *pbSpePoti);
LabView:	
Parameters:	The SpePoti Flag indicates, if the potentiometer is On or Off.
Example:	LS.GetSpeedPoti(&flag);

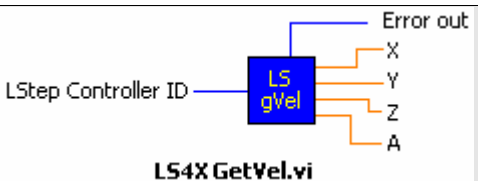
LS_SetSpeedPoti	
Description:	Potentiometer On/Off
Delphi:	function LS_SetSpeedPoti(SpeedPoti: LongBool): Integer; function LSX_SetSpeedPoti(LSID: Integer; SpeedPoti: LongBool): Integer;
C++:	int SetSpeedPoti (BOOL SpeedPoti);
LabView:	
Parameters:	If SpeedPoti = false, the preset speed (vel) is used as the speed of travel. If SpeedPoti = true, travelling is done at a percentage of the preset speed (vel), depending on the setting of the potentiometer.
Example:	LS.SetSpeedPoti(true); // Poti On

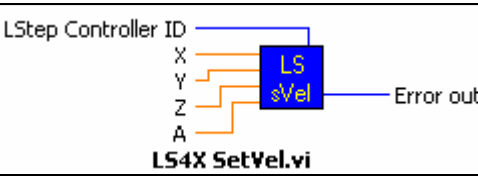
LS_GetStopAccel	
Description:	Delivers the brake acceleration, if the stop input becomes active.
Delphi:	function LS_GetStopAccel (var dXD, dYD, dZD, dAD: Double): Integer; function LSX_GetStopAccel (LSID: Integer; var dXD, dYD, dZD, dAD: Double): Integer;
C++:	int GetStopAccel (double *pdXD, double *pdYD, double *pdZD, double *pdAD);
LabView:	 <p>LS4X GetStopAccel.vi</p>
Parameters:	XD, YD, ZD, AD: Brake acceleration values, [m/s ²]
Example:	LS. GetStopAccel (&XD, &YD, &ZD, &AD);

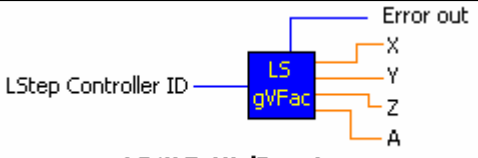
LS_SetStopAccel	
Description:	<p>the brake acceleration, if the stop input is set active</p> <p>The acceleration is staop if the stop input is avice unless the value LS_SetAccel with the set acceleration is bigger.</p> <p>The set brake acceleration is only valid for vector opertating, not for joystick, calibrating and stroke measuring</p> <p>The value is not saved with LStepSave.</p>
Delphi:	function LS_SetStopAccel (dXD, dYD, dZD, dAD: Double): Integer; function LSX_SetStopAccel (LSID: Integer; dXD, dYD, dZD, dAD: Double): Integer;
C++:	int SetStopAccel (double dXD, double dYD, double dZD, double dAD);
LabView:	 <p>LS4X SetStopAccel.vi</p>
Parameters:	dXD, dYD, dZD, dAD: Brake acceleration, value range 0.01 to 20 m/s ²
Example:	LS. SetStopAccel (15.0, 15.0, 15.0, 15.0);

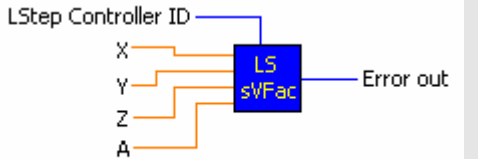
LS_GetStopPolarity	
Description:	Read stop entrance polarity.
Delphi:	function LS_GetStopPolarity(var bHighActiv: LongBool): Integer; function LSX_GetStopPolarity (LSID: Integer; var bHighActiv: LongBool): Integer;
C++:	int GetStopPolarity (BOOL *pbHighActiv);
LabView:	 <p>LS4X GetStopPolarity.vi</p>
Parameters:	bHighActiv: True – Stop input high active False – low active
Example:	LS. GetStopPolarity (&HighActiv);


LS_SetStopPolarity	
Description:	Adjust stop entrance polarity. Because the stop entrance has a Pull Up after 5V, a closer has to be set low active and an opener high active.
Delphi:	function LS_SetStopPolarity(bHighActiv: LongBool): Integer; function LSX_SetStopPolarity (LSID: Integer; bHighActiv: LongBool): Integer;
C++:	int SetStopPolarity (BOOL bHighActiv);
LabView:	 <p>LS4X SetStopPolarity.vi</p>
Parameters:	bHighActiv: True – Stop input high active False – low active
Example:	LS. SetStopPolarity (False); // The stop input is low active

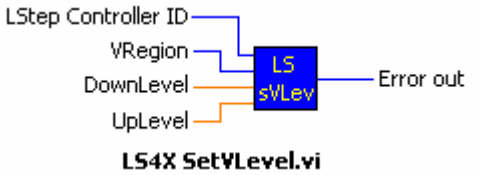
LS_GetVel	
Description:	Inquiry speed
Delphi:	function LS_GetVel(var X, Y, Z, R: Double): Integer; function LSX_GetVel(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetVel (double *pdX, double *pdY, double *pdZ, double *pdA);
LabView:	 LS4X GetVel.vi
Parameters:	X, Y, Z, A: Speed values [rp/s]
Example:	LS.GetVel(&X, &Y, &Z, &A);


LS_SetVel	
Description:	Set speed (velocity)
Delphi:	function LS_SetVel(X, Y, Z, R: Double): Integer; function LSX_SetVel(LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetVel(double dX, double dY, double dZ, double dA);
LabView:	 LS4X SetVel.vi
Parameters:	X, Y, Z and A 0 – maximum speed [r/s]
Example:	LS.SetVel(1.0, 15.0, 0, 0);


LS_GetVelFac	
Description:	Inquiry speed reduction
Delphi:	function LS_GetVelFac (var X, Y, Z, R: Double): Integer; function LSX_GetVelFac (LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int SetVelFac (double dX, double dY, double dZ, double dR);
LabView:	 <p style="text-align: center;">LS4X GetVelFac.vi</p>
Parameters:	X, Y, Z, A: Speed reduction values
Example:	LS.SetVelFac (&X, &Y, &Z, &A);


LS_SetVelFac	
Description:	Set speed reduction
Delphi:	function LS_SetVelFac (X, Y, Z, R: Double): Integer; function LSX_SetVelFac (LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetVelFac (double dX, double dY, double dZ, double dR);
LabView:	 <p style="text-align: center;">LS4X SetVelFac.vi</p>
Parameters:	X, Y, Z, A: Speed reduction, values range 0.01 – 1.00
Example:	LS.SetVelFac (0.1, 0.1, 0.1, 0.1); /* reduces the speed of all axes to 1/10 of the set speeds.

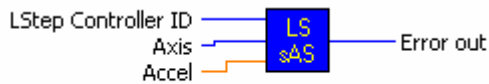
LS_GetVLevel	
Description:	Delivers the speed limits of the indicated speed range.
Delphi:	function LS_GetVLevel(IVRegion: Integer; var dDownLevel, dUpLevel: Double): Integer; function LSX_GetVLevel (LSID: Integer; IVRegion: Integer; var dDownLevel, dUpLevel: Double): Integer;
C++:	int GetVLevel (int IVRegion, double *pdDownLevel, double *pdUpLevel);
LabView:	
Parameters:	IVRegion: Wertebereich 1-4. 1 – First/lowest speed range 2 – Second/middle speed range 3 – Third/highes speed range 4 – Up to this speed limit a correction table is used. dDownLevel : Lower limit of the range(for IVRegion = 4 speed limit) [rp/s] dUpLevel : Upper limit of the range (for IVRegion = 4 has no meaning) [rp/s]
Example:	LS. GetVLevel (2, &DownLevel, &UpLevel); // DownLevel = Lower limit of the second speed range, UpLevel = Upper limit of the second speed range.

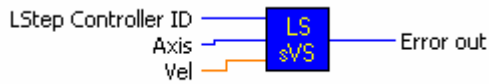
LS_SetVLevel	
Description:	Exclude speed ranges, in which the system shows resonances.
Delphi:	function LS_SetVLevel(IVRegion: Integer; dDownLevel, dUpLevel: Double): Integer; function LSX_SetVLevel (LSID: Integer; IVRegion: Integer; dDownLevel, dUpLevel: Double): Integer;
C++:	int SetVLevel (int IVRegion, double dDownLevel, double dUpLevel);
LabView:	
Parameters:	<p>IVRegion: Value range 1-4.</p> <ul style="list-style-type: none"> 1 – First/lowest speed range 2 – Second/middle speed range 3 – Third/highes speed range 4 – Up to this speed limit a correction table is used. <p>dDownLevel : Lower limit of the range(for IVRegion = 4 speed limit) [rp/s], value range 0 – max. speed.</p> <p>dUpLevel : Upper limit of the range (for IVRegion = 4 has no meaning) [rp/s], value range 0 – max. speed.</p>
Example:	LS. SetVLevel (4, 10.0, 0.0); //The correction table is active to a speed of 10 rp/s.

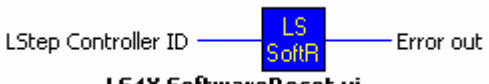
LS_GetXYAxisComp	
Description:	Inquiry XY-axis overlay
Delphi:	function LS_GetXYAxisComp(var Value: Integer): Integer; function LSX_GetXYAxisComp(LSID: Integer; var Value: Integer): Integer;
C++:	int GetXYAxisComp (int *pIValue);
LabView:	 <p>LS4X GetXYAxisComp.vi</p>
Parameters:	Value: Modus der Achsüberlagerung (see LStep-documentation)
Example:	LS.SetXYAxisComp(&mode) ;

LS_SetXYAxisComp	
Description:	activate XY-axis overlay
Delphi:	function LS_SetXYAxisComp(Value: Integer): Integer; function LSX_SetXYAxisComp(LSID: Integer; Value: Integer): Integer;
C++:	int SetXYAxisComp (int IValue);
LabView:	 <p>LS4X SetXYAxisComp.vi</p>
Parameters:	Value: Mode for axis overlay (see LStep-documentation)
Example:	LS.SetXYAxisComp(1) ;


LS_LstepSave	
Description:	Save current configuration in LStep (EEPROM)
Delphi:	function LS_LStepSave(): Integer; function LSX_LStepSave(LSID: Integer): Integer;
C++:	int LStepSave ();
LabView:	 <p>LS4X LStepSave.vi</p>
Parameters:	-
Example:	LS.LStepSave() ;

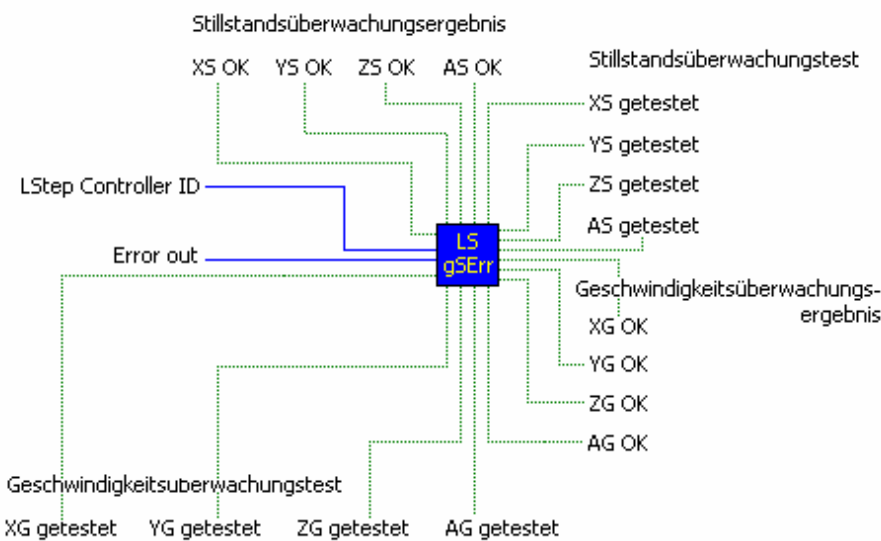
LS_SetAccelSingleAxis	
Description:	Set acceleration for individual axis
Delphi:	function LS_SetAccelSingleAxis(Axis: Integer; Accel: Double): Integer; function LSX_SetAccelSingleAxis(LSID: Integer; Axis: Integer; Accel: Double): Integer;
C++:	int SetAccelSingleAxis (int lAxis,double dAccel);
LabView:	 <p>LS4X SetAccelSingleAxis.vi</p>
Parameters:	Axis: (X, Y, Z, A numbered from 1 to 4) Accel: Acceleration 0.01 – 10.00 [m/s ²]
Example:	LS.SetAccelSingleAxis(4, 1.0); // Accelerate A-axis 1.0 m/s ²

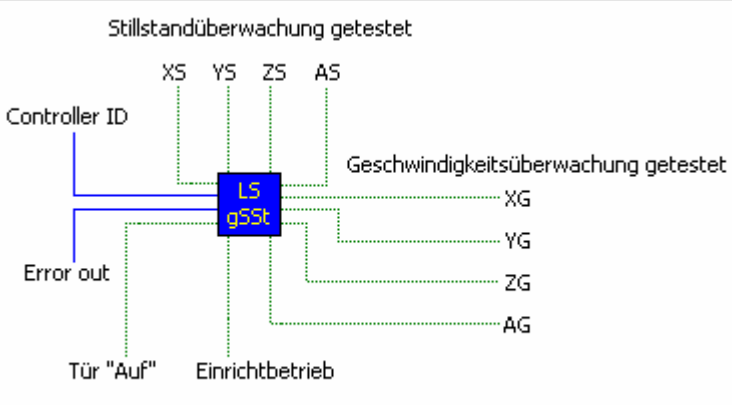
LS_SetVelSingleAxis	
Description:	Set speed for individual axis
Delphi:	function LS_SetVelSingleAxis(Axis: Integer; Vel: Double): Integer; function LSX_SetVelSingleAxis(LSID: Integer; Axis: Integer; Vel: Double): Integer;
C++:	int SetVelSingleAxis (int lAxis,double dVel);
LabView:	 <p>LS4X SetVelSingleAxis.vi</p>
Parameters:	Axis: (X, Y, Z, A numbered from 1 to 4) Vel: 0 – maximum speed [r/s]
Example:	LS.SetVelSingleAxis(1, 10.0) // Speed of X-axis 10 r/s

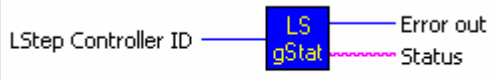
LS_SoftwareReset	
Description:	Reset the software to starting status.
Delphi:	function LS_SoftwareReset: Integer; function LSX_SoftwareReset(LSID: Integer): Integer;
C++:	int SoftwareReset ();
LabView:	 <p>LS4X SoftwareReset.vi</p>
Parameters:	-
Example:	LS.SoftwareReset ();


Status report


LS_GetError	
Description:	gives the current error number
Delphi:	function LS_GetError(var ErrorCode: Integer): Integer; function LSX_GetError(LSID: Integer; var ErrorCode: Integer): Integer;
C++:	int GetError (int *plErrorCode);
LabView:	 <p>LS4X GetError.vi</p>
Parameters:	ErrorCode: error number
Example:	LS.GetError(&ErrCode);


LS_GetSecurityErr	
Description:	Reads all statuses and results of the GAL-safety monitoring (only with LS44-controller)
Delphi:	function LS_GetSecurityErr (var Value: LongWord): Integer; function LSX_GetSecurityErr (LSID: Integer; var Value: LongWord): Integer;
C++:	int GetSecurityErr (LongWord *pValue);
LabView:	 <p style="text-align: center;">LS4XGetSecurityErr.vi</p>
Parameters:	Value: 32-Bit LongWord ohne Vorzeichen, welcher nach Aufruf der Funktion in den Bits 0-15 die Bit-Maske enthält. Bit 0 X-axis standstill monitoring result (OK [1] / not OK [0]) Bit 1 Y- axis standstill monitoring result Bit 2 Z- axis standstill monitoring result Bit 3 A- axis standstill monitoring result Bit 5 X- axis standstill monitoring test (tested [1] /not tested [0]) Bit 6 Y- axis standstill monitoring test Bit 7 Z- axis standstill monitoring test Bit 8 A- axis standstill monitoring test Bit 9 X- axis standstill monitoring result Bit 10 Y- axis standstill monitoring result Bit 11 Z- axis standstill monitoring result Bit 12 A- axis standstill monitoring result Bit 13 X- axis standstill monitoring test Bit 14 Y- axis standstill monitoring test Bit 15 Z- axis standstill monitoring test
Example:	LS.GetSecurityErr (&Value);

LS_GetSecurityStatus	
Description:	Delivers the current statuses the safety monitoring (only with LS44-controller)
Delphi:	function LS_GetSecurityStatus (var Value: LongWord): Integer; function LSX_GetSecurityStatus (LSID: Integer; var Value: LongWord): Integer;
C++:	int GetSecurityStatus (LongWord *pValue);
LabView:	 <p style="text-align: center;">LS4X GetSecurityStatus.vi</p>
Parameters:	Value: 32-Bit LongWord without preceding sign, which contains the bit mask in the bits 0-15 after activating the function. Bit 0-3 internal memory Bit 4 X-axis standstill monitoring tested Bit 5 Y- axis standstill monitoring tested Bit 6 Z- axis standstill monitoring tested Bit 7 A- axis standstill monitoring tested Bit 8 X-axis speed monitoring tested Bit 9 Y- axis speed monitoring tested Bit 10 Z- axis speed monitoring tested Bit 11 A- axis speed monitoring tested Bit 14 Condition setup mode Einrichtbetrieb (setup mode = 1) Bit 15 Condition door (door „Open“ = 1)
Example:	LS.GetSecurityStatus (&Value);


LS_GetStatus	
Description:	Gives the current status of the controller.
Delphi:	function LS_GetStatus(Stat: PChar; MaxLen: Integer): Integer; function LSX_GetStatus(LSID: Integer; Stat: PChar; MaxLen: Integer): Integer;
C++:	int GetStatus (char *pcStat,int IMaxLen);
LabView:	 LS4X GetStatus.vi
Parameter:	Stat: Pointer to a buffer in which the status string is returned MaxLen: Maximum number of characters which may be copied into the buffer
Beispiel:	LS.GetStatus(pcStat, 256);

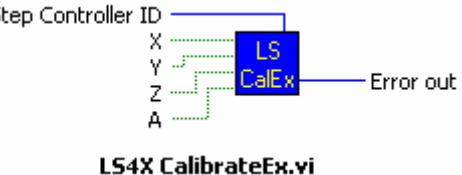
LS_GetStatusAxis	
Description:	Gives the present status of the individual axes
Delphi:	function LS_GetStatusAxis(StatusAxisStr: PChar; MaxLen: Integer): Integer; function LSX_GetStatusAxis(LSID: Integer; StatusAxisStr: PChar; MaxLen: Integer): Integer;
C++:	int GetStatusAxis (char *pcStatusAxisStr,int IMaxLen);
LabView:	 LS4X GetStatusAxis.vi
Parameters:	StatusAxisStr: Pointer to a buffer in which the status string is returned MaxLen: Maximum number of characters which can be copied into the buffer e.g.: @ - M - J - C - S - A - D - U - T @ = Axis at a standstill M = Axis is moving (Motion) - = Axis is not enabled J = Joystick switched on C = Axis in control A = feedback after calibration E = Fault during calibration (Limit switch was not set free correctly) D = feedback after table stroke measuring U = Set up mode T = Timeout
Example:	LS.GetStatusAxis(pcStatAxis, 256);

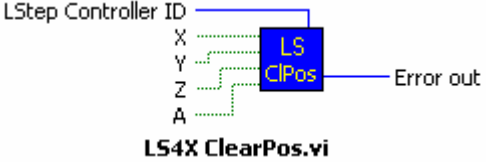
LS_GetStatusLimit	
Description:	Delivers the current condition of the software-limits of each axis.
Delphi:	function LS_GetStatusLimit (Limit: PChar; MaxLen: Integer): Integer; function LSX_GetStatusLimit (LSID: Integer; Limit: PChar; MaxLen: Integer): Integer;
C++:	int GetStatusLimit (char *pcLimit, int lMaxLen);
LabView:	 <p>LS4X GetStatusLimit.vi</p>
Parameters:	<p>pc Limit: Pointer to a buffer, in which the condition of the axis is returned to. Z. B.: AA- A- - DD - LL- L- - L</p> <p>A =Axis was calibrated D = Table stroke was measured L = Software-limit was set - = Software-limit was not changed</p> <p>MaxLen: Maximum number of characters, that can be copied into the buffer</p>
Example:	LS.GetStatusLimit (pc Limit, 64);


LS_SetAutoStatus	
Description:	<p>AutoStatus On/Off</p> <p>Note: The AutoStatus mode should not normally be changed, as LSTEP API sets the correct mode for travel commands etc. Changing to 0 or 2 could pitch to errors</p>
Delphi:	function LS_SetAutoStatus(Value: Integer): Integer; function LSX_SetAutoStatus(LSID: Integer; Value: Integer): Integer;
C++:	int SetAutoStatus (int lValue);
LabView:	 <p>LS4X SetAutoStatus.vi</p>
Parameters:	<p>Value: AutoStatus mode:</p> <p>0 → No status is transmitted by the controller. 1 → "Position reached" signals are sent automatically by the controller. 2 → "Position reached" and status messages are sent automatically by the controller. 3 → For "Position reached" only a Carriage Return is returned.</p>
Example:	LS.SetAutoStatus(3);

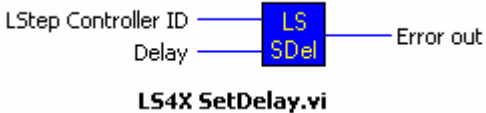
Moving commands and position administration

LS_Calibrate	
Description:	Calibrate
	Moves all released axes to smaller position values. The movements are interrupted as soon as the limit switch is reached. The position values are set to 0.
Delphi:	function LS_Calibrate: Integer; function LSX_Calibrate(LSID: Integer): Integer;
C++:	int Calibrate();
LabView:	 <p>LS4X Calibrate.vi</p>
Parameters:	-
Example:	LS.Calibrate();

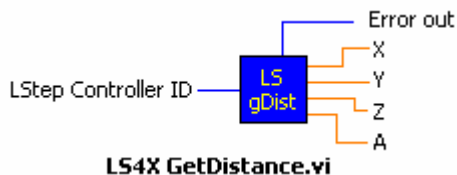
LS_CalibrateEx	
Description:	Calibrate
	(Only the axes are calibrated, whose corresponding bit was set in the transmitted integer-value)
Delphi:	function LS_CalibrateEx(Flags: Integer): Integer; function LSX_CalibrateEx(LSID: Integer; Flags: Integer): Integer;
C++:	int CalibrateEx (int IFlags);
LabView:	 <p>LS4X CalibrateEx.vi</p>
Parameters:	Flags: Bit-mask, Bit 2 = 1 → calibrate Z-axis Bit 2 = 0 → no calibrating of Z-axis ...
Example:	LS.CalibrateEx(6); // Calibrate only Y- and Z-axis

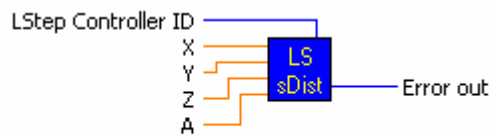
LS_ClearPos	
Description:	<p>Sets the position to 0, also the internal counter.</p> <p>This function is needed for endless axes, because the control can process only + 1000 motor revolutions of the value range.</p> <p>In recognised encoders, the function is not carried out for corresponding axis.</p>
Delphi:	function LS_ClearPos (IFlags: Integer): Integer; function LSX_ClearPos (LSID: Integer; IFlags: Integer): Integer;
C++:	int ClearPos (int IFlags);
LabView:	
Parameters:	IFlags: Bit-Mask Bit 0 = 1 → Position of the x-axis is zeroized Bit 1 = 0 → For the y-axis the function is not carried out.
Example:	LS. ClearPos(5); //Positions of the x- and z- axes are zeroized.

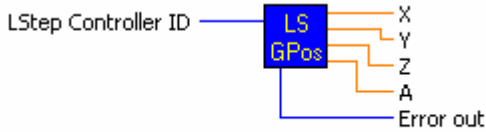
LS_GetDelay	
Description:	Reads the delay of the vector start.
Delphi:	function LS_GetDelay(var Delay: Integer): Integer; function LSX_GetDelay(LSID: Integer; var Delay: Integer): Integer;
C++:	int GetDelay (int *plDelay);
LabView:	
Parameters:	Delay: in ms
Example:	LS.GetDelay(&Delay);

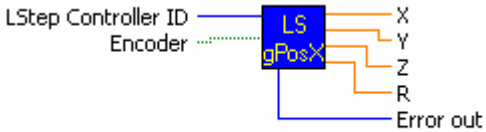
LS_SetDelay	
Description:	The delay command is used to produce a vector start delay.
Delphi:	function LS_SetDelay(Delay: Integer): Integer; function LSX_SetDelay(LSID: Integer; Delay: Integer): Integer;
C++:	int SetDelay (int lDelay);
LabView:	
Parameters:	0 – 10000 (ms)
Example:	LS.SetDelay(1000);

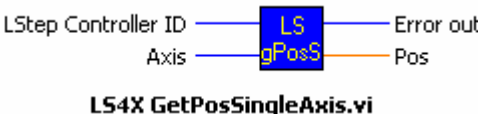
	// 1s delay
--	-------------

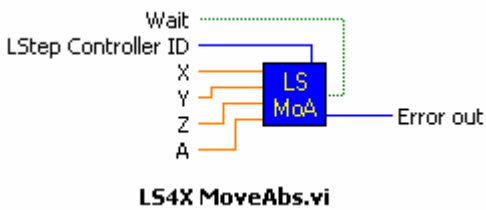
LS_GetDistance	
Description:	Delivers the distance for LS_MoveRelShort
Delphi:	function LS_GetDistance(var X, Y, Z, A: Double): Integer; function LSX_GetDistance(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetDistance (double *pdX, double *pdY, double *pdZ, double *pdR);
LabView:	 <p>LS4X GetDistance.vi</p>
Parameters:	X, Y, Z and A: the current distances of all axes, dependent on the dimensions.
Example:	LS.GetDistance(&X, &Y, &Z, &A);

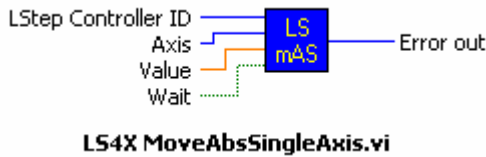
LS_SetDistance	
Description:	Set distance (for LS_MoveRelShort)
Delphi:	function LS_SetDistance(X, Y, Z, A: Double): Integer; function LSX_SetDistance(LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetDistance (double dX,double dY,double dZ,double dA);
LabView:	 <p>LS4X SetDistance.vi</p>
Parameters:	X, Y, Z and A Min./max. range of travel (Values depend on the dimension)
Example:	LS.SetDistance(1, 2, 0, 0); /* Distances are set for the X-, and Y-axes, Z and A are not moved when the function LS_MoveRelShort is called. */

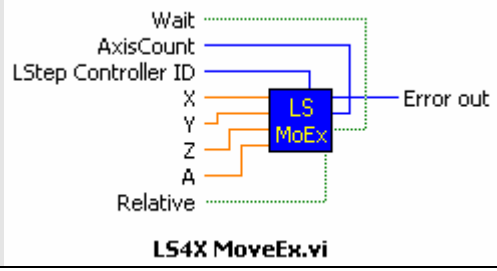
LS_GetPos	
Description:	Inquires the current positions of all axes
	For non-existing axes, a value of 0.0 is returned
Delphi:	function LS_GetPos(var X, Y, Z, A: Double): Integer; function LSX_GetPos(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetPos (double *pdX,double *pdY,double *pdZ,double *pdA);
LabView:	 <p style="text-align: center;">LS4X GetPos.vi</p>
Parameters:	X, Y, Z, A: positional values
Example:	double X, Y, Z, A; LS.GetPos(&X, &Y, &Z, &A);

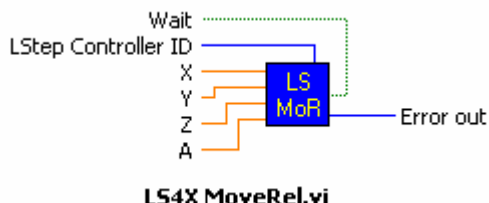
LS_GetPosEx	
Description:	Inquires the current encoder or positional values of all axes
	For non-existing axes, a value of 0.0 is returned
Delphi:	function LS_GetPosEx(var X, Y, Z, A: Double; Encoder: LongBool): Integer; function LSX_GetPosEx(LSID: Integer; var X, Y, Z, R: Double; Encoder: LongBool): Integer;
C++:	int GetPosEx (double *pdX,double *pdY,double *pdZ,double *pdA,BOOL Encoder);
LabView:	 <p style="text-align: center;">LS4X GetPosEx.vi</p>
Parameters:	X, Y, Z, A: Positionswerte Encoder = true → Geberwerte liefern falls Geber angeschlossen Encoder = false → Positionswerte liefern
Example:	double X, Y, Z, A; LS.GetPosEx(&X, &Y, &Z, &A, true);

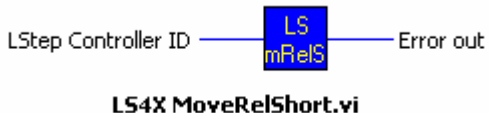
LS_GetPosSingleAxis	
Description:	Inquire the current position of an axis
	For non-existing axes, a value of 0.0 is returned
Delphi:	function LS_GetPosSingleAxis(Axis: Integer; var Pos: Double): Integer; function LSX_GetPosSingleAxis(LSID: Integer; Axis: Integer; var Pos: Double): Integer;
C++:	int GetPosSingleAxis (int lAxis,double *pdPos);
LabView:	
Parameter:	Axis: Axis for which the position is to be inquired (X, Y, Z, A numbered from 1 to 4) Pos: positional value
Example:	LS.GetPosSingleAxis(2, &YPos); // Read position of Y-axis

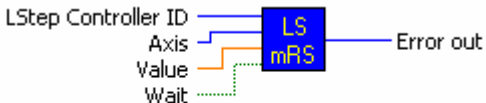
LS_MoveAbs	
Description:	Move to absolute position (The X-, Y-, and Z- axes are positioned at the transmitted positional values.)
Delphi:	function LS_MoveAbs(X, Y, Z, A: Double; Wait: LongBool): Integer; function LSX_MoveAbs(LSID: Integer; X, Y, Z, A: Double; Wait: LongBool): Integer;
C++:	int MoveAbs (double dX, double dY, double dZ, double dA, BOOL Wait);
LabView:	
Parameters:	X, Y, Z and A +/- Range of travel Input depends on the dimension Wait: Specifies whether the function should return after the position has been reached (= true) or directly (= false)
Example:	LS.MoveAbs(10.0, 10.0, 10.0, 10.0, true);


LS_MoveAbsSingleAxis	
Description:	Move individual axis to absolute position
Delphi:	function LS_MoveAbsSingleAxis(Axis: Integer; Value: Double; Wait: LongBool): Integer; function LSX_MoveAbsSingleAxis(LSID: Integer; Axis: Integer; Value: Double; Wait: LongBool): Integer;
C++:	int MoveAbsSingleAxis (int lAxis,double dValue,BOOL Wait);
LabView:	
Parameters:	Axis: (X, Y, Z, A numbered from 1 to 4) Value: Position (input depends on the preset dimension)
Example:	LS.MoveAbsSingleAxis(2, 10.0); // Move Y-axis to 10mm absolute position

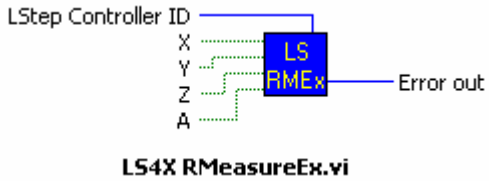
LS_MoveEx	
Description:	<p>Extended move command</p> <p>The function LS_MoveEx can carry out relative and absolute move commands, synchronically and asynchronous. The amount of axes, that are supposed to move, can be determined with the parameter AxisCount. This function for example can be used, to move only X and Y of the LStep44.</p>
Delphi:	<p>function LS_MoveEx(X, Y, Z, R: Double; Relative, Wait: LongBool; AxisCount: Integer): Integer;</p> <p>function LSX_MoveEx(LSID: Integer; X, Y, Z, R: Double; Relative, Wait: LongBool; AxisCount: Integer): Integer;</p>
C++:	<p>int MoveEx (double dX, double dY, double dZ, double dR, BOOL bRelative, BOOL bWait, int lAxisCount);</p>
LabView:	
Parameters:	<p>X, Y, Z, R: Position-vector</p> <p>Relative: with relative=false all values X, Y, Z and R are interpreted as absolute co-ordinates.</p> <p>Wait: is Wait=true, the function returns only after reaching the target position otherwise it returns directly after sending the command to the LStep.</p> <p>AxisCount: Amount of axes, that should move Is AxisCount=1, only X moves Is AxisCount=2, X and Y move...</p>
Example:	<p>LS_MoveEx(2.0, 3.0, 0, 0, true, true, 2) ; // X and Y are moved relative by 2 resp. 3</p>

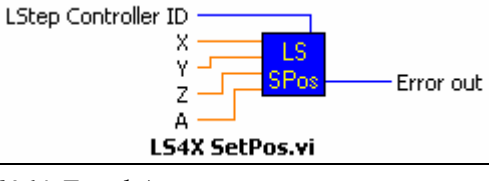
LS_MoveRel	
Description:	Move to relative vector (The X-, Y-, and Z-axes are moved the transmitted distances.)
Delphi:	function LS_MoveRel(X, Y, Z, A: Double; Wait: LongBool): Integer; function LSX_MoveRel(LSID: Integer; X, Y, Z, A: Double; Wait: LongBool): Integer;
C++:	int MoveRel (double dX, double dY, double dZ, double dA, BOOL Wait);
LabView:	
Parameters:	X, Y, Z and A +- Range of travel Input depends on the dimension Wait: Specifies whether the function should return after the position has been reached (= true) or directly (= false)
Example:	LS.MoveRel(10.0, 10.0, 10.0, 10.0, true);


LS_MoveRelShort	
Description:	Move to relative position (short command) This command should be used, so that a series of consecutive relative travel commands (of the same distance) are approached more quickly The distance must have been set beforehand with LS_SetDistance .
Delphi:	function LS_MoveRelShort: Integer; function LSX_MoveRelShort(LSID: Integer): Integer;
C++:	int MoveRelShort ();
LabView:	
Parameters:	-
Example:	LS.SetDistance(1.0, 1.0, 0, 0); for (i = 0; i < 10; i++) LS.MoveRelShort(); // Move the X- and Y- axes 10times 1 mm to the relative position

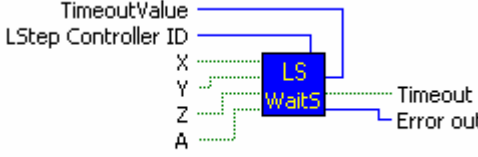
LS_MoveRelSingleAxis	
Description:	Move individual axis relatively
Delphi:	function LS_MoveRelSingleAxis(Axis: Integer; Value: Double; Wait: LongBool): Integer; function LSX_MoveRelSingleAxis(LSID: Integer; Axis: Integer; Value: Double; Wait: LongBool): Integer;
C++:	int MoveRelSingleAxis (int lAxis,double dValue,BOOL Wait);
LabView:	 <p>LS4X MoveRelSingleAxis.vi</p>
Parameters:	Axis: (X, Y, Z, A numbered from 1 to 4) Value: Distance (Input depends on the preset dimensions)
Example:	LS.MoveRelSingleAxis(3, 5.0); // Move Z-axis 5mm in positive direction

LS_RMeasure	
Description:	Measure table stroke
	Moves all enabled axes towards greater positional values. Travel is stopped as soon as the limit switches are reached. The positional value is saved.
Delphi:	function LS_RMeasure: Integer; function LSX_RMeasure(LSID: Integer): Integer;
C++:	int RMeasure();
LabView:	 <p>LS4X RMeasure.vi</p>
Parameters:	-
Example:	LS.RMeasure();

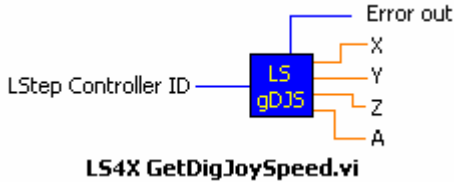
LS_RMeasureEx	
Description:	Measure table stroke (The table stroke is only measured for axes for which the relevant bit has been set in the transmitted integer value.
Delphi:	function LS_RMeasureEx(Flags: Integer): Integer; function LSX_RMeasureEx(LSID: Integer; Flags: Integer): Integer;
C++:	int RMeasureEx (int IFlags);
LabView:	
Parameters:	Flags: Bit mask, Bit 2 = 1 → Calibrate Z-axis Bit 2 = 0 → Do not calibrate Z-axis ...
Example:	LS.RMeasureEx(2); // Measure table stroke (Y-axis only)

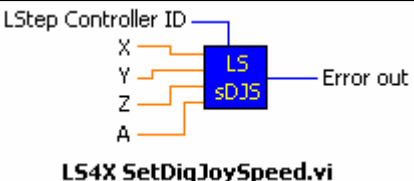
LS_SetPos	
Description:	Set positional values
Delphi:	function LS_SetPos(X, Y, Z, R: Double): Integer; function LSX_SetPos(LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetPos(double dX, double dY, double dZ, double dA);
LabView:	
Parameters:	X, Y, Z and A Min./Max. range of travel Input depends on the dimension
Example:	LS.SetPos(10, 10, 0, 0);

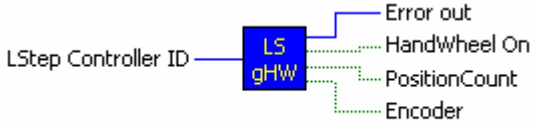
LS_StopAxes	
Description:	Stop (All movements are stopped)
Delphi	function LS_StopAxes: Integer; function LSX_StopAxes(LSID: Integer): Integer;
C++	int StopAxes ();
LabView	
Parameters:	-
Example:	LS.StopAxes();

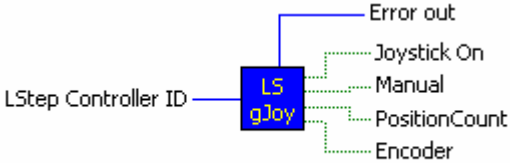
LS_WaitForAxisStop	
Description:	<p>The function returns, as soon as the selected axes in the bit-mask AFlags reached its goal position.</p> <p>LS_WaitForAxisStop uses ,?statusaxis', to pollen the status of the axes.</p>
Delphi:	function LS_WaitForAxisStop(AFlags: Integer; ATimeoutValue: Integer; var ATimeout: LongBool): Integer; function LSX_WaitForAxisStop(LSID: Integer; AFlags: Integer; ATimeoutValue: Integer; var ATimeout: LongBool): Integer;
C++:	int WaitForAxisStop (int lAFlags, int lATimeoutValue, BOOL *pbATimeout);
LabView:	
Parameters:	<p>AFlags: Bit-mask Bit 0: X-axis Bit 1: Y- axis Bit 2: Z- axis Bit 3: A- axis</p> <p>ATimeoutValue: Timeout in milliseconds, WaitForAxisStop returns after this time with Atimeout=true, if the axes are still in motion AtimeoutValue = 0 set the timeout to 'endless' The Atimeout Flag indicates, if a timeout occurred.</p>
Example:	LS.WaitForAxisStop(3, 0, flag); // Wating until X and Y-Achse stopped, no timeout LS.WaitForAxisStop(7, 10000, flag); // Wating until X and Y-Achse stopped, 10 seconds timeout

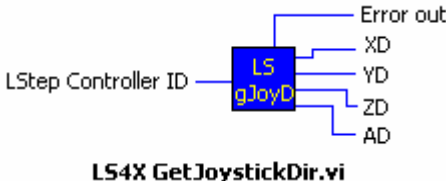
Joystick and Handwheel

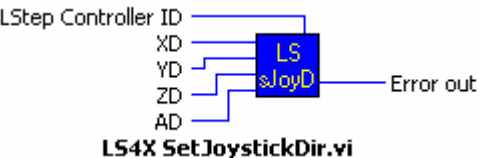
LS_GetDigJoySpeed	
Description:	Read out the set speeds
Delphi	function LS_GetDigJoySpeed(var dX, dY, dZ, dR: Double): Integer; function LSX_GetDigJoySpeed(LSID: Integer; var dX, dY, dZ, dR: Double): Integer;
C++	int GetDigJoySpeed (double *pdX, double *pdY, double *pdZ, double *pdR);
LabView	
Parameters:	dX, dY, dZ, dR: Speed values [rp/s]
Example:	LS. GetDigJoySpeed(&X, &Y, &Z, &R);

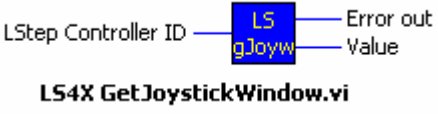
LS_SetDigJoySpeed	
Description:	With this command, single axes can be operated with a constant speed. If the positioning should be done absolutely or relatively after carrying out this function, the speed needs to be set new.
Delphi:	function LS_SetDigJoySpeed(dX, dY, dZ, dR: Double): Integer; function LSX_SetDigJoySpeed(LSID: Integer; dX, dY, dZ, dR: Double): Integer;
C++:	int SetDigJoySpeed (double dX, double dY, double dZ, double dR);
LabView:	
Parameters:	dX, dY, dZ, dR: Speed [rp/s], Value range: +- max. speed
Example:	LS. SetDigJoySpeed (0, 10.0, 25.0, 0); //axes X and R – speed 0 and Joystick-operation „OFF“, Axis Y – speed 10.0 rp/s and Joystick-operation „ON“, Axes Z – speed 25.0 rp/s and Joystick-operation „ON“.

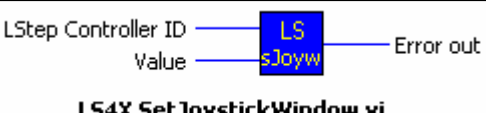
LS_GetHandWheel	
Description:	Read hand wheel condition
Delphi	function LS_GetHandWheel(var PositionCount, Encoder: Boolean): Integer; function LSX_GetHandWheel(LSID: Integer; var PositionCount, Encoder: LongBool): Integer;
C++	int GetHandWheel (BOOL *pbHandWheelOn, BOOL *pbPositionCount, BOOL *pbEncoder);
LabView	 <p>LS4X GetHandWheel.vi</p>
Parameters:	HWOn: True = Hand wheel is switched on PosCount: True = Position counter is switched on Encoder: True = Encoder values, if available
Example:	LS. GetHandWheel (&HWOn, &PosCount, &Encoder);


LS_GetJoystick	
Description:	Inquiry of the current condition of the analogy-Joystick.
Delphi	function LS_GetJoystick (var JoystickOn, Manual, PositionCount, Encoder: LongBool): Integer; function LSX_GetJoystick (LSID: Integer; var JoystickOn, Manual, PositionCount, Encoder: LongBool): Integer;
C++	int GetJoystick (BOOL *pbJoystickOn, BOOL *pbManual, BOOL *pbPositionCount, BOOL *pbEncoder);
LabView	 <p>LS4X GetJoystick.vi</p>
Parameters:	JoyOn: True = Joystick is switched on Manual: False = Joystick switch is set to automatic True = Joystick ist manually switched on via switch PosCount: True = Position counter is switched on Enc: True = Encoder values, if available
Example:	LS. GetJoystick (&JoyOn, &Manual, &PosCount, &Enc);

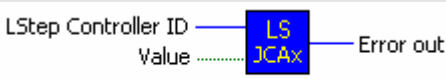
LS_GetJoystickDir	
Description:	Reads motor turning direction for joystick
Delphi	function LS_GetJoystickDir(var XD, YD, ZD, AD: Integer): Integer; function LSX_GetJoystickDir(LSID: Integer; var XD, YD, ZD, AD: Integer): Integer;
C++	int GetJoystickDir (int *plXD, int *plYD, int *plZD, int *plRD);
LabView	 <p>LS4X GetJoystickDir.vi</p>
Parameters:	<p>X, Y, Z, and A</p> <p>0 → Axis locked</p> <p>1 → positive turning direction</p> <p>-1 → negative turning direction</p> <p>2 → with current reduction</p> <p>-2 → with current reduction</p>
Example:	LS.GetJoystickDir(&X, &Y, &Z, &A);


LS_SetJoystickDir	
Description:	Joystick direction
Delphi:	function LS_SetJoystickDir(XD, YD, ZD, AD: Integer): Integer; function LSX_SetJoystickDir(LSID: Integer; XD, YD, ZD, AD: Integer): Integer;
C++:	int SetJoystickDir (int lXD,int lYD,int lZD,int lAD);
LabView:	 <p>LS4X SetJoystickDir.vi</p>
Parameters:	<p>X, Y, Z, and A</p> <p>0 → Axis disabled</p> <p>1 → Positive direction of rotation</p> <p>-1 → Negative direction of rotation</p> <p>2 → with current reduction</p> <p>-2 → with current reduction</p>
Example:	LS.SetJoystickDir(1, 1, -1, 0); /* X- and Y-axis positive direction of rotation; Z-axis negative direction of rotation; A-axis disabled */


LS_GetJoystickWindow	
Description:	Read Joystick-window
Delphi	function LS_GetJoystickWindow(var AValue: Integer): Integer; function LSX_GetJoystickWindow(LSID: Integer; var AValue: Integer): Integer;
C++	int GetJoystickWindow (int *pIAValue);
LabView	
Parameters:	AValue: the analogous range in which the axes do not move.
Example:	LS.GetJoystickWindow(&AValue) ;


LS_SetJoystickWindow	
Description:	Set joystick window
Delphi:	function LS_SetJoystickWindow(AValue: Integer): Integer; function LSX_SetJoystickWindow(LSID: Integer; AValue: Integer): Integer;
C++:	int SetJoystickWindow (int IAValue);
LabView:	
Parameters:	AValue: 0-100
Example:	LS.SetJoystickWindow(20) ;


LS_GetJoyChangeAxis	
Description:	Read Joystick allocation of the axes
Delphi:	LS_GetJoyChangeAxis(var Value: LongBool): Integer; LSX_GetJoyChangeAxis(LSID: Integer; var Value: LongBool): Integer;
C++:	int GetJoyChangeAxis (BOOL *pbValue);
LabView:	
Parameters:	Value: true => Conventional evaluation of Joystick false => allocation of X and Y axes have been exchanged
Example:	LS. GetJoyChangeAxis (&Value);

LS_JoyChangeAxis	
Description:	sets allocation of axes of Joystick
Delphi:	LS_JoyChangeAxis(Value: LongBool): Integer; LSX_JoyChangeAxis(LSID: Integer; Value: LongBool): Integer;
C++:	int JoyChangeAxis (BOOL bValue);
LabView:	 <p>LS4X JoyChangeAxis.vi</p>
Parameters:	Value: 0 – Changes the allocation of the AD-Joystick channels (conventional evaluation of the Joystick) 1 – Allocation of X and Y axes will be exchanged
Example:	LS. JoyChangeAxis (true);

LS_SetHandWheelOff	
Description:	Handwheel Off
Delphi:	function LS_SetHandWheelOff: Integer; function LSX_SetHandWheelOff(LSID: Integer): Integer;
C++:	int SetHandWheelOff ();
LabView:	 <p>LS4X SetHandWheelOff.vi</p>
Parameters:	-
Example:	LS.SetHandWheelOff();

LS_SetHandWheelOn	
Description:	Handwheel On
Delphi:	function LS_SetHandWheelOn(PositionCount, Encoder: Boolean): Integer; function LSX_SetHandWheelOn(LSID: Integer; PositionCount, Encoder: LongBool): Integer;
C++:	int SetHandWheelOn (BOOL fPositionCount,BOOL fEncoder);
LabView:	 <p>LS4X SetHandWheelOn.vi</p>
Parameters:	PositionCount: Position count On/Off Encoder: Encoder values, if any
Example:	LS. SetHandWheelOn (true, true); // Handwheel On with position count (encoder values)

LS_SetJoystickOff	
Description:	Analog joystick Off
Delphi:	function LS_SetJoystickOff: Integer; function LSX_SetJoystickOff(LSID: Integer): Integer;
C++:	int SetJoystickOff();
LabView:	
Parameters:	-
Example:	LS.SetJoystickOff();

LS_SetJoystickOn	
Description:	Analog joystick On
Delphi:	function LS_SetJoystickOn(PositionCount, Encoder: LongBool): Integer; function LSX_SetJoystickOn(LSID: Integer; PositionCount, Encoder: LongBool): Integer;
C++:	int SetJoystickOn (BOOL PositionCount,BOOL Encoder);
LabView:	
Parameters:	PositionCount: Position count On/Off Encoder: Encoder values (positions), if any
Example:	LS.SetJoystickOn(true, true); // Joystick On with position count (encoder values)

Control panel with Trackball and Joyspeed-keys

LS_GetBPZ	
Description:	Reads the condition of the additional control panel with track ball
Delphi:	function LS_GetBPZ(var AValue: Integer): Integer; function LSX_GetBPZ(LSID: Integer; var AValue: Integer): Integer;
C++:	int GetBPZ (int *plAValue);
LabView:	-
Parameters:	AValue: 0 => Control panel is „OFF“. 1 => Control panel active, track ball runs with 0,1μ step resolution. 2 => Control panel active, track ball runs with factor.
Example:	LS.GetBPZ(&AValue);

LS_SetBPZ	
Description:	Bedienpult Ein/ Aus
Delphi:	function LS_SetBPZ(AValue: Integer): Integer; function LSX_SetBPZ(LSID: Integer; AValue: Integer): Integer;
C++:	int SetBPZ (int lAValue);
LabView:	-
Parameters:	AValue: 0-2 0 => Control panel „OFF“ 1 => activate operating control panel and trackball with 0,1μ step resolution 2 => activate operating control panel and trackball with factor.
Example:	LS.SetBPZ(1);

LS_GetBPZJoyspeed	
Description:	Control panel joystick-speed
Delphi:	function LS_GetBPZJoyspeed(APar: Integer; var AValue: Double): Integer; function LSX_GetBPZJoyspeed(LSID: Integer; APar: Integer; var AValue: Double): Integer;
C++:	int GetBPZJoyspeed (int lAPar, double *pdAValue);
LabView:	-
Parameters:	APar: 1, 2 or 3 AValue: maximum speed [rp/s]
Example:	GetBPZJoyspeed(1, &AValue); // Read out the set speed of parameter 1.

LS_SetBPZJoyspeed	
Description:	Control panel joystick-speed
Delphi:	function LS_SetBPZJoyspeed(APar: Integer; AValue: Double): Integer; function LSX_SetBPZJoyspeed(LSID: Integer; APar: Integer; AValue: Double): Integer;
C++:	int SetBPZJoyspeed (int lAPar, double dAValue);
LabView:	-
Parameters:	APar: 1, 2 or 3 AValue: +- maximum speed (vel)
Example:	SetBPZJoyspeed(1, 25) // Set parameter 1 to speed 25

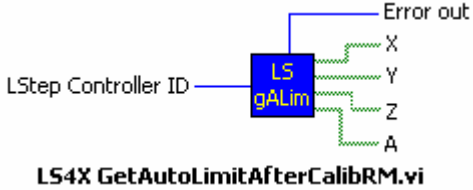
LS_GetBPZTrackballBacklash	
Description:	Read out control panel track ball-back lash
Delphi:	function LS_GetBPZTrackballBackLash(var X, Y, Z, R: Double): Integer; function LSX_GetBPZTrackballBackLash(LSID: Integer; var X, Y, Z, R: Double): Integer;
C++:	int GetBPZTrackballBackLash (double *pdX, double *pdY, double *pdZ, double *pdR);
LabView:	-
Parameters:	X, Y, Z, R: Back lash, mm.
Example:	LS.GetBPZTrackballBackLash(&X, &Y, &Z, &R);

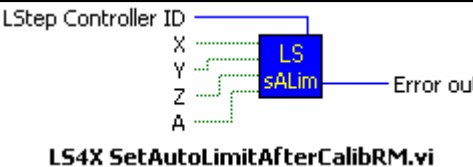
LS_SetBPZTrackballBacklash	
Description:	Control panal trackball-reverse backlash
Delphi:	function LS_SetBPZTrackballBackLash(X, Y, Z, R: Double): Integer; function LSX_SetBPZTrackballBackLash(LSID: Integer; X, Y, Z, R: Double): Integer;
C++:	int SetBPZTrackballBackLash (double dX, double dY, double dZ, double dR);
LabView:	-
Parameters:	X, Y, Z, R: 0.001 to 0.15 mm
Example:	LS.SetBPZTrackballBackLash(0.01, 0.01, 0.01, 0.01);

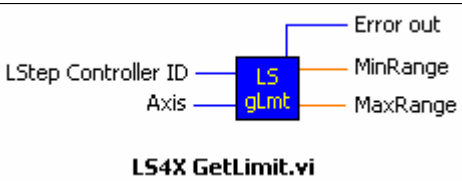
LS_GetBPZTrackballFactor	
Description:	Read ot control panal trackball-factor
Delphi:	function LS_GetBPZTrackballFactor(AValue: Double): Integer; function LSX_GetBPZTrackballFactor(LSID: Integer; AValue: Double): Integer;
C++:	int GetBPZTrackballFactor (double *pdAValue);
LabView:	-
Parameters:	AValue: Trackball – Factor. Z. B. AValue = 3 means: One trackball-impulse results 3 motor-Increment.
Example:	LS.GetBPZTrackballFactor(&AValue) ;

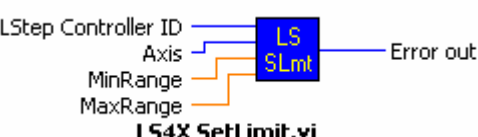
LS_SetBPZTrackballFactor	
Description:	Control panal trackball-factor
Delphi:	function LS_SetBPZTrackballFactor(AValue: Double): Integer; function LSX_SetBPZTrackballFactor(LSID: Integer; AValue: Double): Integer;
C++:	int SetBPZTrackballFactor (double dAValue);
LabView:	-
Parameters:	AValue: 0.01 – 100 AValue=1 => Trackball – Factor = 1, i.e. One trackball-impulse results one motor-Increment.
Example:	LS.SetBPZTrackballFactor(1.0) ;

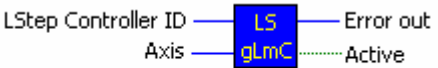
Limit switch (Hardware a. Software)

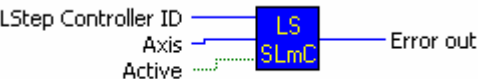
LS_GetAutoLimitAfterCalibRM	
Description:	Indicates if the internal software limits will be set during calibration and table stroke measuring.
Delphi:	function LS_GetAutoLimitAfterCalibRM(var IFlags: Integer): Integer; function LSX_GetAutoLimitAfterCalibRM(LSID: Integer; var IFlags: Integer): Integer;
C++:	int GetAutoLimitAfterCalibRM (int *pIFlags);
LabView:	 <p>LS4X GetAutoLimitAfterCalibRM.vi</p>
Parameters:	IFlags: Bit-Mask Bit 0 = 1 → No travel limits are set for the x-axis Bit 1 = 0 → Software limits are set for the y-axis (calib/rm)
Example:	LS. SetAutoLimitAfterCalibRM(&IFlags);

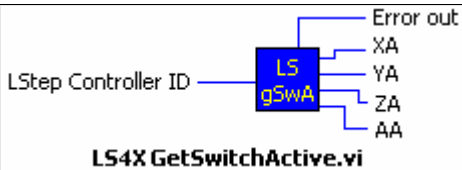
LS_SetAutoLimitAfterCalibRM	
Description:	Prevents that the internal software limits are set during calibration and table stroke measuring.
Delphi:	function LS_SetAutoLimitAfterCalibRM(IFlags: Integer): Integer; function LSX_SetAutoLimitAfterCalibRM(LSID: Integer; IFlags: Integer): Integer;
C++:	int SetAutoLimitAfterCalibRM (int IFlags);
LabView:	 <p>LS4X SetAutoLimitAfterCalibRM.vi</p>
Parameters:	IFlags: Bit-Mask Bit 0 = 1 → No travel limits are set for the x-axis Bit 1 = 0 → Software limits are set for the y-axis (calib/rm)
Example:	LS. SetAutoLimitAfterCalibRM(IFlags);

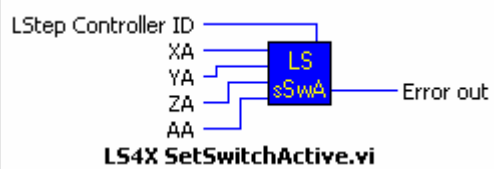
LS_GetLimit	
Description:	Set travel limits
Delphi:	function LS_GetLimit(Axis: Integer; var MinRange, MaxRange: Double): Integer; function LSX_GetLimit(LSID: Integer; Axis: Integer; var MinRange, MaxRange: Double): Integer;
C++:	int GetLimit (int lAxis, double *pdMinRange, double *pdMaxRange);
LabView:	 LS4X GetLimit.vi
Parameters:	Axis: the axis for which the travel limits are to be read (X, Y, Z, A numbered from 1 to 4) MinRange: minimum travel limit, depending on dimension MaxRange: maximum travel limit, depending on dimension
Example:	LS.GetLimit(1, &MinRange, &MaxRange);

LS_SetLimit	
Description:	Set travel limits
Delphi:	function LS_SetLimit(Axis: Integer; MinRange, MaxRange: Double): Integer; function LSX_SetLimit(LSID: Integer; Axis: Integer; MinRange, MaxRange: Double): Integer;
C++:	int SetLimit (int lAxis,double dMinRange,double dMaxRange);
LabView:	 LS4X SetLimit.vi
Parameters:	Axis: the axis for which the travel limits are to be set (X, Y, Z, A numbered from 1 to 4) MinRange: minimum travel limit MaxRange: maximum travel limit
Example:	LS.SetLimit(1, -10.0, 20.0); (Set 10 as minimum limit and 20 as maximum limit for the X-axis)

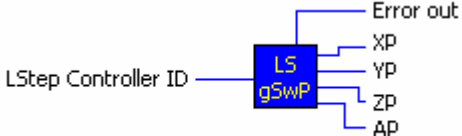
LS_GetLimitControl	
Description:	Reads, if travel range monitoring is active
Delphi:	function LS_GetLimitControl(Axis: Integer; var Active: LongBool): Integer; function LSX_GetLimitControl(LSID: Integer; Axis: Integer; var Active: LongBool): Integer;
C++:	int GetLimitControl (int lAxis, BOOL *pbActive);
LabView:	 <p>LS4X GetLimitControl.vi</p>
Parameters:	Active: True = travel range monitoring is active
Example:	LS.GetLimitControl(2, &Active); // Activ = False means: travel range monitoring of axis y is deactivated.

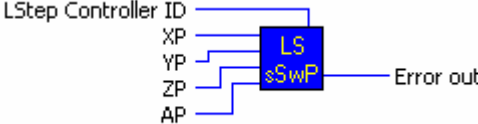
LS_SetLimitControl	
Description:	Monitoring of travel range
Delphi:	function LS_SetLimitControl(Axis: Integer; Active: LongBool): Integer; function LSX_SetLimitControl(LSID: Integer; Axis: Integer; Active: LongBool): Integer;
C++:	int SetLimitControl (int lAxis,BOOL Active);
LabView:	 <p>LS4X SetLimitControl.vi</p>
Parameters:	Axis: (X, Y, Z, A numbered from 1 to 4) Active: Activate limit control for the axis in question
Example:	LS.SetLimitControl(2, true); // Limit control for Y-axis active

LS_GetSwitchActive	
Description:	Read status of limit switch
Delphi:	function LS_GetSwitchActive(var XA, YA, ZA, AA: Integer): Integer; function LSX_GetSwitchActive(LSID: Integer; var XA, YA, ZA, AA: Integer): Integer;
C++:	int GetSwitchActive (int *plXA, int *plYA, int *plZA, int *plRA);
LabView:	 <p>LS4X GetSwitchActive.vi</p>
Parameters:	<p>A bit maks is transmitted for each axis:</p> <p>Bit 0 → Zero limit switch</p> <p>Bit 1 → Reference limit switch</p> <p>Bit 2 → End limit switch</p> <p>To activate the respective switch, the appropriate bit must be set.</p>
Example:	LS.GetSwitchActive(&XA, &YA, &ZA, &RA);

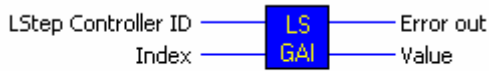
LS_SetSwitchActive	
Description:	Limit switch On/Off
Delphi:	function LS_SetSwitchActive(XA, YA, ZA, AA: Integer): Integer; function LSX_SetSwitchActive(LSID: Integer; XA, YA, ZA, AA: Integer): Integer;
C++:	int SetSwitchActive (int IXA,int IYA,int IZA,int IAA);
LabView:	 <p>LS4X SetSwitchActive.vi</p>
Parameters:	<p>A bit maks is transmitted for each axis:</p> <p>Bit 0 → Zero limit switch</p> <p>Bit 1 → Reference limit switch</p> <p>Bit 2 → End limit switch</p> <p>To activate the respective switch, the appropriate bit must be set.</p>
Example:	<p>LS.SetSwitchActive(7, 1, 5, 0);</p> <p>(All X-axis limit switches ON; Y-axis zero limit switch ON; Z-axis E0 and EE ON; A-axis: all limit switches OFF)</p>

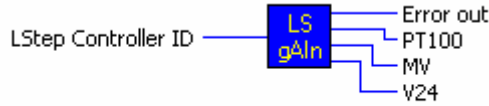
LS_GetSwitches													
Description:	Reads the status of all limit switches												
Delphi:	function LS_GetSwitches(var Flags: Integer): Integer												
C++:	int GetSwitches (int *plFlags);												
LabView:	<div><div><div>LStep Controller ID</div><div>LSgSwi</div><div>LS4X GetSwitches.vi</div></div><div><div>Error out</div><div>Flags</div></div></div>												
Parameters:	<p>Value: Pointer to integer value which contains the status of all limit switches as a bit mask.</p> <p>The condition of the limit switch is coded in the bit mask as follows:</p> <table><tr><td>Limit switch</td><td>EE</td><td>Ref.</td><td>E0</td></tr><tr><td>Axis</td><td>AZYX</td><td>AZYX</td><td>AZYX</td></tr><tr><td>Bit</td><td>0000</td><td>0000</td><td>0000</td></tr></table> <p>z.B.</p> <p>Flags = 0x003 → E0 of X- and Y-Axis are reached</p> <p>Flags = 0x200 → EE of Y-Axis is reached</p>	Limit switch	EE	Ref.	E0	Axis	AZYX	AZYX	AZYX	Bit	0000	0000	0000
Limit switch	EE	Ref.	E0										
Axis	AZYX	AZYX	AZYX										
Bit	0000	0000	0000										
Example:	LS.GetSwitches(&Flags);												


LS_GetSwitchPolarity	
Description:	Reads limit switch polarity
Delphi:	function LS_GetSwitchPolarity(var XP, YP, ZP, AP: Integer): Integer; function LSX_GetSwitchPolarity(LSID: Integer; var XP, YP, ZP, AP: Integer): Integer;
C++:	int GetSwitchPolarity (int *plXP, int *plYP, int *plZP, int *plRP);
LabView:	 <p>LS4X GetSwitchPolarity.vi</p>
Parameters:	<p>A bit mask is transmitted for each axis:</p> <p>Bit 0 → Zero limit switch</p> <p>Bit 1 → Reference limit switch</p> <p>Bit 2 → End limit switch</p> <p>To activate the respective switch, the appropriate bit must be set.</p>
Example:	LS.GetSwitchPolarity(&XP, &YP, &ZP, &RP);


LS_SetSwitchPolarity	
Description:	Set limit switch polarity
Delphi:	function LS_SetSwitchPolarity(XP, YP, ZP, AP: Integer): Integer; function LSX_SetSwitchPolarity(LSID: Integer; XP, YP, ZP, AP: Integer): Integer;
C++:	int SetSwitchPolarity (int IXP,int IYP,int IZP,int IRA);
LabView:	 <p>LS4X SetSwitchPolarity.vi</p>
Parameters:	<p>A bit mask is transmitted for each axis:</p> <p>Bit 0 → Zero limit switch</p> <p>Bit 1 → Reference limit switch</p> <p>Bit 2 → End limit switch</p> <p>If the respective switch reacts to the positive flank, the bit must be set.</p>
Example:	LS.SetSwitchPolarity(7, 0, 0, 0); (All X-axis limit switches high-active, all Y-axis limit switches low-active...)

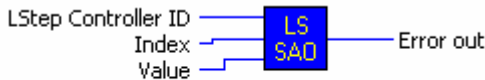
Digital and analogue In.- and Outputs

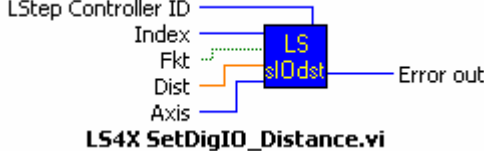
LS_GetAnalogInput	
Description:	Reading the current condition of an analogous channel
Delphi:	function LS_GetAnalogInput(Index: Integer; var Value: Integer): Integer; function LSX_GetAnalogInput(LSID: Integer; Index: Integer; var Value: Integer): Integer;
C++:	int GetAnalogInput (int lIndex,int *plValue);
LabView	 <p>LS4X GetAnalogInput.vi</p>
Parameters:	Index: 0-9 (Analog channels) Value: Pointer to integer value, that indicates the current condition of the analogous channel.
Example:	LS.GetAnalogInput(0, &Inputs0);


LS_GetAnalogInputs2	
Description:	read the current condition of the analogous channels (Channel 6, 7, 8) only with the LSTEP-PCI
Delphi	function LS_GetAnalogInputs2(var PT100, MV, V24: Integer): Integer; function LSX_GetAnalogInputs2(LSID: Integer; var PT100, MV, V24: Integer): Integer;
C++	int GetAnalogInputs2 (int *plPT100, int *plMV, int *plV24);
LabView	 <p>LS4X GetAnalogInputs2.vi</p>
Parameters:	PT100, MV, V24: Pointer to integer value, in which GetAnalogInputs2 supposed to write the current condition of the analogue channels.
Example:	LS.GetAnalogInputs2(&PT100, &MV, &V24);


LS_GetDigitalInputs	
Description:	Read all input pins
Delphi:	function LS_GetDigitalInputs(var Value: Integer): Integer; function LSX-GetDigitalInputs(LSID: Integer;var Value: Integer): Integer;
C++:	int GetDigitalInputs (int *pIValue);
LabView:	 <p>LS4X GetDigitalInputs.vi</p>
Parameters:	Value: Pointer to integer value which contains the status of all inputs as a bit mask.
Example:	<pre>int inputs; LS.GetDigitalInputs(&Inputs); if (Inputs & 16) ... // when input pin 4 is set</pre>


LS_GetDigitalInputsE	
Description:	read additional digital inputs (16-31)
Delphi:	function LS_GetDigitalInputsE(var Value: Integer): Integer; function LSX_GetDigitalInputsE(LSID: Integer; var Value: Integer): Integer;
C++:	int GetDigitalInputsE (int *pIValue);
LabView:	 <p>LS4X GetDigitalInputsE.vi</p>
Parameters:	Value: Pointer to a 32-bit-Integer, which after activating the function contains the status of the inputs 16-31 in the Bits 0-15.
Example:	LS.GetDigitalInputsE(i);

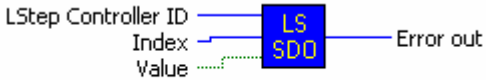
LS_SetAnalogOutput	
Description:	Set analog output
Delphi:	function LS_SetAnalogOutput(Index: Integer; Value: Integer): Integer; function LSX_SetAnalogOutput(LSID: Integer; Index: Integer; Value: Integer): Integer;
C++:	int SetAnalogOutput (int IIndex,int IValue);
LabView:	 <p>LS4X SetAnalogOutput.vi</p>
Parameters:	Index: 0-1 (Analog channels) Value: 0-100 [%]
Example:	<pre>LS.SetAnalogOutput(0, 100); // Set output 0 to maximum</pre>

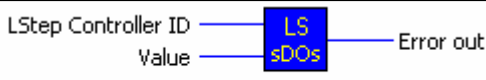
LS_SetDigIO_Distance	
Description:	Function of the digital inputs/outputs
	Activation of an output dependent on the set distance before / after the target position.
Delphi:	function LS_SetDigIO_Distance(Index: Integer; Fkt: LongBool; Dist: Double; Axis: Integer): Integer; function LSX_SetDigIO_Distance(LSID: Integer; Index: Integer; Fkt: LongBool; Dist: Double; Axis: Integer): Integer;
C++:	int SetDigIO_Distance (int lIndex,BOOL Fkt,double dDist,int lAxis);
LabView:	
Parameters:	Index: 0 to 15 (Output pin)
	Fct = false → Activation of an output dependent on the set distance <u>before</u> the target position. Fct = true → Activation of an output dependent on the set distance <u>after</u> the start position.
	Dist: Distance (Input depends on the preset dimension) Axis: (X, Y, Z, A numbered from 1 to 4)
Example:	LS.SetDigIO_Distance(7, false, 78.9, 3); /* Output 7 is activated 78.9mm before the target position (Z-axis) is reached. */

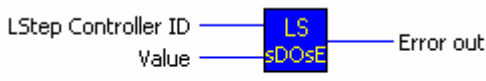
LS_SetDigIO_EmergencyStop	
Description:	Function of the digital inputs/outputs Allocation of the Emergency Stop pin
Delphi:	function LS_SetDigIO_EmergencyStop(Index: Integer): Integer; function LSX_SetDigIO_EmergencyStop(LSID: Integer; Index: Integer): Integer;
C++:	int SetDigIO_EmergencyStop (int lIndex);
LabView:	
Parameters:	Index: 0 to 15 (Input/Output)
Example:	LS.SetDigIOEmergencyStop(15); // Emergency Stop pin 15

LS_SetDigIO_Off	
Description:	“Off” function of the digital inputs/outputs (no influence of the inputs/outputs)
Delphi:	function LS_SetDigIO_Off(Index: Integer): Integer; function LSX_SetDigIO_Off(LSID: Integer; Index: Integer): Integer;
C++:	int SetDigIO_Off (int lIndex);
LabView:	 <p>LS4X SetDigIO_Off.vi</p>
Parameters:	Index: 0 to 15 (Input/Output), 16 (all 16 port pins)
Example:	LS.SetDigIO_Off(0); // dig. fct. Input/Output pin 0 off

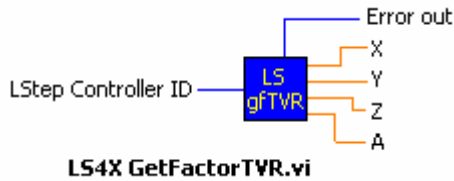
LS_SetDigIO_Polarity	
Description:	Function of the digital inputs/outputs Set polarity
Delphi:	function LS_SetDigIO_Polarity(Index: Integer; High: LongBool): Integer; function LSX_SetDigIO_Polarity(LSID: Integer; Index: Integer; High: LongBool): Integer;
C++:	int SetDigIO_Polarity (int lIndex,BOOL High);
LabView:	 <p>LS4X SetDigIO_Polarity.vi</p>
Parameters:	Index: 0 to 15 (Input/Output), 16 (all 16 port pins) High = true → High-active High = false → Low-active
Example:	LS.SetDigIO_Polarity(3, True); // Input/Output pin 3 high-active

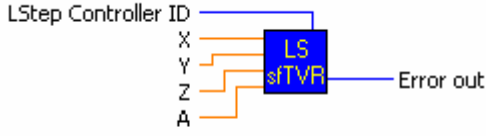
LS_SetDigitalOutput	
Description:	Set output pin
Delphi:	function LS_SetDigitalOutput(Index: Integer; Value: LongBool): Integer; function LSX_SetDigitalOutput(LSID: Integer; Index: Integer; Value: LongBool): Integer;
C++:	int SetDigitalOutput (int IIndex,BOOL Value);
LabView:	 <p>LS4X SetDigitalOutput.vi</p>
Parameters:	Index: 0-15 Value: Set status to "0" or "1"
Example:	LS.SetDigitalOutput(0, true); // Set output pin 0 to "1"

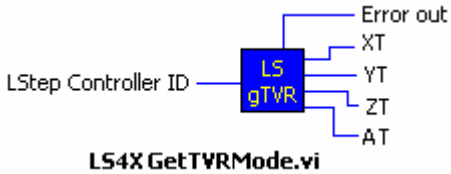
LS_SetDigitalOutputs	
Description:	set digital outputs (0-15)
Delphi:	function LS_SetDigitalOutputs(Value: Integer): Integer; function LSX_SetDigitalOutputs(LSID: Integer; Value: Integer): Integer;
C++:	int SetDigitalOutputs (int IValue);
LabView:	 <p>LS4X SetDigitalOutputs.vi</p>
Parameters:	Value: Bit mask, the value that the outputs 0-15 are set to, is determined via the bits 0-15.
Example:	LS.SetDigitalOutputs(\$03); // Set outputs 0 and 1 to 1, the remaining 0.

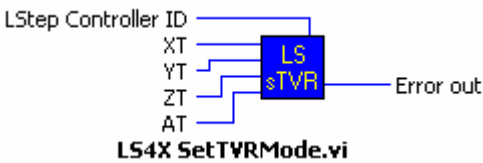
LS_SetDigitalOutputsE	
Description:	Set additional digital outputs (16-31)
Delphi:	function LS_SetDigitalOutputsE(Value: Integer): Integer; function LSX_SetDigitalOutputsE(LSID: Integer; Value: Integer): Integer;
C++:	int SetDigitalOutputsE (int IValue);
LabView:	 <p>LS4X SetDigitalOutputsE.vi</p>
Parameters:	Value: Bit mask, the values that the outputs 16-31 are set to, determined via the Bits 0-15.
Example:	LS.SetDigitalOutputsE(\$03); // Set outputs 16 and 17 to 1, the remaining to 0

Clock pulse Forward / Back

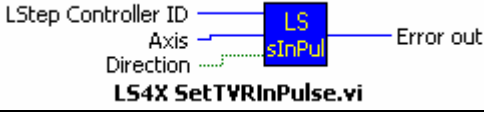
LS_GetFactorTVR	
Description:	Reads factor for clock pulse Forward/ Back
Delphi:	function LS_GetFactorTVR(var X, Y, Z, A: Double): Integer; function LSX_GetFactorTVR(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetFactorTVR (double *pdX, double *pdY, double *pdZ, double *pdR);
LabView:	 <p>LS4X GetFactorTVR.vi</p>
Parameters:	X, Y, Z and A: Factor clock pulse Forward/ Back. Z. B. X = 10 means: One puls = ten Motor increments
Example:	LS.GetFactorTVR(&X, &Y, &Z, &A);

LS_SetFactorTVR	
Description:	Factor for clock pulse Forward/ Back
Delphi:	function LS_SetFactorTVR(X, Y, Z, A: Double): Integer; function LSX_SetFactorTVR(LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetFactorTVR (double dX,double dY,double dZ,double dA);
LabView:	 <p>LS4X SetFactorTVR.vi</p>
Parameters:	X, Y, Z and A 0.01 – 100.00
Example:	LS.SetFactorTVR(2.0, 2.0, 0, 0); /* Clock pulse Forward/Back is to work with the factor 2 for the X- and Y-axis */

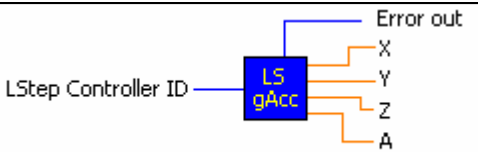
LS_GetTVRMode	
Description:	Read setup of clock pulse Forward / Back (= TVR Mode)
Delphi:	function LS_GetTVRMode(var XT, YT, ZT, AT: Integer): Integer; function LSX_GetTVRMode(LSID: Integer; var XT, YT, ZT, AT: Integer): Integer;
C++:	int GetTVRMode (int *plXT, int *plYT, int *plZT, int *plRT);
LabView:	
Parameters:	TVR-mode for X, Y, Z and A: 0 → Clock pulse Forward / Back (= TVR mode) "OFF" 1 → Normal clock pulse Forward/Back processing 2 → Processing of clock pulse Forward/Back with a factor 3 → Clock pulse Forward / Back processing must be externally enabled with the triggerout pin (MFP). 4 → Combination of 2 & 3.
Example:	LS.GetTVRMode(&XT, &YT, &ZT, &RT);

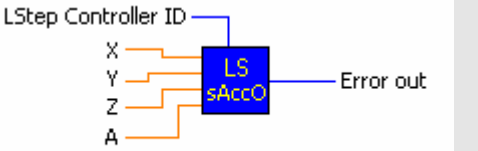
LS_SetTVRMode	
Description:	Set clock pulse Forward / Back (= TVR Mode)
Delphi:	function LS_SetTVRMode(XT, YT, ZT, AT: Integer): Integer; function LSX_SetTVRMode(LSID: Integer; XT, YT, ZT, AT: Integer): Integer;
C++:	int SetTVRMode (int lXT,int lYT,int lZT,int lAT);
LabView:	
Parameters:	TVR-mode for X, Y, Z and A: 0 → Clock pulse Forward / Back (= TVR mode) "OFF" 1 → Normal clock pulse Forward/Back processing 2 → Processing of clock pulse Forward/Back with a factor 3 → Clock pulse Forward / Back processing must be externally enabled with the triggerout pin (MFP). 4 → Combination of 2 & 3.
Example:	LS.SetTVRMode(1, 1, 0, 0); // TVR ON for X- and Y-axes

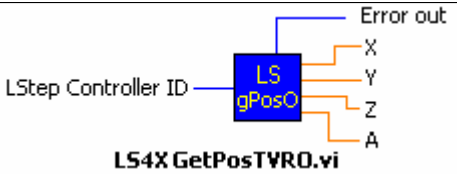
Clock pulse Forward /Back via Interface

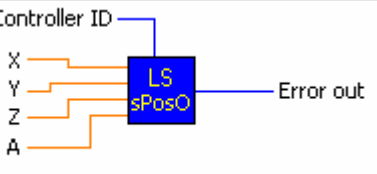
LS_SetTVRInPulse	
Description:	This function has the same influence as an external clock pulse with direction information.
Delphi:	function LS_SetTVRInPulse (Axis: Integer; Direction: Boolean): Integer; function LSX_SetTVRInPulse (LSID: Integer; Axis: Integer; Direction: Boolean): Integer;
C++:	int SetTVRInPulse (int Axis, BOOL Direction);
LabView:	
Parameters:	Value: Amount of executed Trigger
Example:	LS.SetTVRInPulse (2, true); // 1 clock pulse Forward y-Axis.

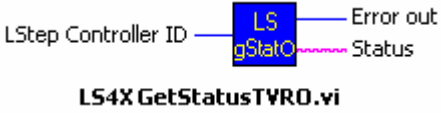
Clock Pulse Forward / Back for additional axes

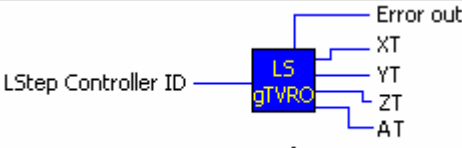
LS_GetAccelTVRO	
Description:	Reads the set acceleration for the additional axes.
Delphi:	function LS_GetAccelTVRO(var X, Y, Z, A: Double): Integer; function LSX_GetAccelTVRO(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetAccelTVRO (double *pdX, double *pdY, double *pdZ, double *pdA);
LabView:	 <p>LS4X GetAccelTVRO.vi</p>
Parameters:	X, Y, Z, A: Acceleration values, U/s ²
Example:	LS.GetAccelTVRO(&X, &Y, &Z, &A);

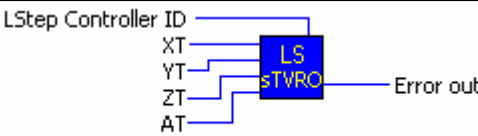
LS_SetAccelTVRO	
Description:	Set acceleration for the additional axes
Delphi:	function LS_SetAccelTVRO(X, Y, Z, A: Double): Integer; function LSX_SetAccelTVRO(LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetAccelTVRO (double dX, double dY, double dZ, double dA);
LabView:	 <p>LS4X SetAccelTVRO.vi</p>
Parameters:	X, Y, Z and A: Acceleration, value range 0.01 – 1500 [U/s ²]
Example:	LS.SetAccelTVRO(1.0, 1.5, 0, 0);

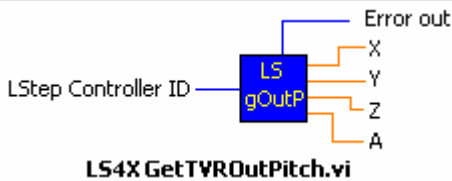
LS_GetPosTVRO	
Description:	Read position of the additional axis
Delphi:	function LS_GetPosTVRO(var dX, dY, dZ, dR: Double): Integer; function LSX_GetPosTVRO(LSID: Integer; var dX, dY, dZ, dR: Double): Integer;
C++:	int GetPosTVRO (double *pdX, double *pdY, double *pdZ, double *pdR);
LabView:	 <p style="text-align: center;">LS4X GetPosTVRO.vi</p>
Parameters:	dX, dY, dZ, dR: Position value, depending on the dimension
Example:	LS. GetPosTVRO(&X, &Y, &Z, &R);

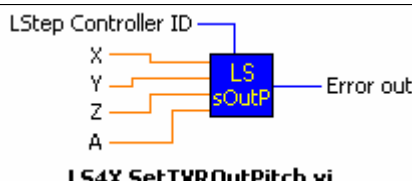
LS_SetPosTVRO	
Description:	Set position of the additional axis
Delphi:	function LS_SetPosTVRO(dX, dY, dZ, dR: Double): Integer; function LSX_SetPosTVRO(LSID: Integer; dX, dY, dZ, dR: Double): Integer;
C++:	int SetPosTVRO (double dX, double dY, double dZ, double dR);
LabView:	 <p style="text-align: center;">LS4X SetPosTVRO.vi</p>
Parameters:	dX, dY, dZ, dR: Position value, depending on the dimension. Value range: min. range limit to max. range limit
Example:	LS. SetPosTVRO(10.0, 5.0, 0.0, 0.0);

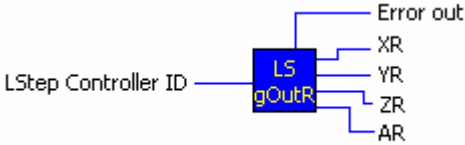
LS_GetStatusTVRO	
Description:	Delivers the current status of the additional axis
Delphi:	function LS_GetStatusTVRO(pcStat: PChar; MaxLen: Integer): Integer; function LSX_GetStatusTVRO(LSID: Integer; pcStat: PChar; MaxLen: Integer): Integer;
C++:	int GetStatusTVRO (char *pcStat, int lMaxLen);
LabView:	
Parameters:	pcStat: Pointert to a buffer, in which the status string is returned MaxLen: Maximum amount of characters, that can be copied into the buffer z.B.: @ - M - @ = Axis standing M = Axis in motion - = Axis are enabled
Example:	LS. GetStatusTVRO(pcStat, 256); // Move the additional Z-axis 5mm in positive direction

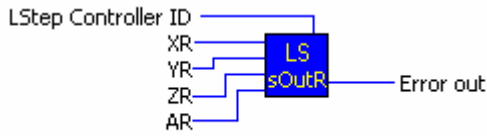
LS_GetTVROutMode	
Description:	Read settings of the additional axis
Delphi:	function LS_GetTVROutMode(var X, Y, Z, A: Integer): Integer; function LSX_GetTVROutMode(LSID: Integer; var X, Y, Z, A: Integer): Integer;
C++:	int GetTVROutMode (int *plXT, int *plYT, int *plZT, int *plAT);
LabView:	 LS4X GetTVROutMode.vi
Parameters:	X, Y, Z and A: 0 => Puls Forw/Back is "OFF" 1 => Puls Forw/Back is "ON"
Example:	LS.GetTVROutMode(&X, &Y, &Z, &A);

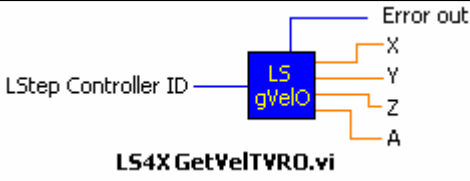
LS_SetTVROutMode	
Description:	Set additional axis X, Y, Z and A, beside the actual main axis X, Y, Z and A
Delphi:	function LS_SetTVROutMode(X, Y, Z, A: Integer): Integer; function LSX_SetTVROutMode(LSID: Integer; X, Y, Z, A: Integer): Integer;
C++:	int SetTVROutMode (int lXT, int lYT, int lZT, int lAT);
LabView:	 LS4X SetTVROutMode.vi
Parameters:	X, Y, Z and A: 0 or 1
Example:	LS.SetTVROutMode(1, 0, 1, 0); // Puls Forw/Back of the x and z is activated, and deactivated for y and a.

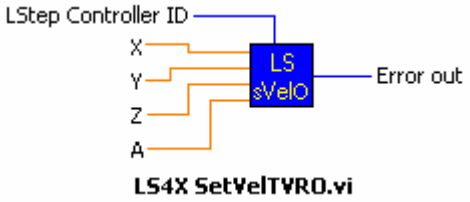
LS_GetTVROutPitch	
Description:	Reads the spindle pitch of the additional axis
Delphi:	function LS_GetTVROutPitch(var X, Y, Z, R: Double): Integer; function LSX_GetTVROutPitch (LSID: Integer; var X, Y, Z, R: Double): Integer;
C++:	int GetTVROutPitch (double *pdX, double *pdY, double *pdZ, double *pdR);
LabView:	
Parameters:	X, Y, Z and R: Spindle pitch [mm]
Example:	LS. GetTVROutPitch(&X, &Y, &Z, &A);

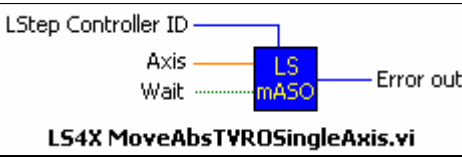
LS_SetTVROutPitch	
Description:	Sets the spindle pitch for the additional axis
Delphi:	function LS_SetTVROutPitch(X, Y, Z, R: Double): Integer; function LSX_SetTVROutPitch (LSID: Integer; X, Y, Z, R: Double): Integer;
C++:	int SetTVROutPitch (double dX, double dY, double dZ, double dR);
LabView:	
Parameters:	X, Y, Z and R: Spindle pitch [mm], value range 0.001 to 100
Example:	LS. SetTVROutPitch(1.0, 4.0, 1.0, 1.0); /* Spindle pitch of y-axis is 4 mm. For x-, z- and a-axis spindle with 1mm pitch are used */

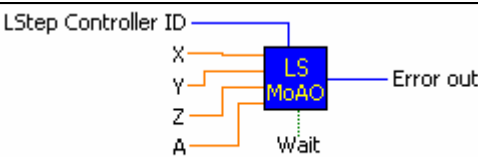
LS_GetTVROutResolution	
Description:	Reads the resolution of the amplifier which is to be controlled
Delphi:	function LS_GetTVROutResolution(var X, Y, Z, A: Integer): Integer; function LSX_GetTVROutResolution (LSID: Integer; var X, Y, Z, A: Integer): Integer;
C++:	int GetTVROutResolution (int *plX, int *plY, int *plZ, int *plA);
LabView:	 <p style="text-align: center;">LS4X GetTVROutResolution.vi</p>
Parameters:	X, Y, Z and A: Impulses per rotation
Example:	LS. GetTVROutResolution (&X, &Y, &Z, &A);

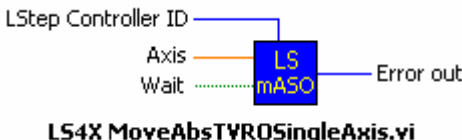
LS_SetTVROutResolution	
Description:	Sets the resolution of the amplifier which is to be controlled
Delphi:	function LS_SetTVROutResolution(X, Y, Z, A: Integer): Integer; function LSX_SetTVROutResolution (LSID: Integer; X, Y, Z, A: Integer): Integer;
C++:	int SetTVROutResolution (int lX, int lY, int lZ, int lA);
LabView:	 <p style="text-align: center;">LS4X SetTVROutResolution.vi</p>
Parameters:	X, Y, Z and A: Impulses per rotation value range 0 to 51200
Example:	LS. SetTVROutResolution (1000, 1000, 0, 0); /* The resolution of axis X and Y is 1000 impulses per rotation */

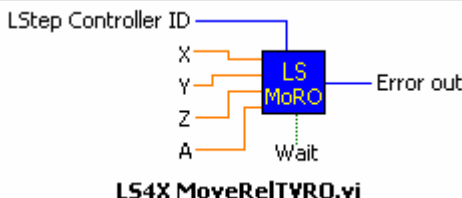
LS_GetVelTVRO	
Description:	Reads the set speed of the additional axis
Delphi:	function LS_GetVelTVRO(var X, Y, Z, A: Double): Integer; function LSX_GetVelTVRO(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetVelTVRO (double *pdX, double *pdY, double *pdZ, double *pdA);
LabView:	
Parameters:	X, Y, Z, A: Speed values, [rp/s]
Example:	LS.GetVelTVRO(&X, &Y, &Z, &A);

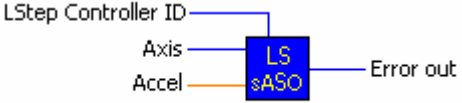
LS_SetVelTVRO	
Description:	set speed of the additional axis
Delphi:	function LS_SetVelTVRO(X, Y, Z, A: Double): Integer; function LSX_SetVelTVRO(LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetVelTVRO (double dX, double dY, double dZ, double dA);
LabView:	
Parameters:	X, Y, Z and A: speed, 0 - 40.0 [rp/s]
Example:	LS.SetVelTVRO(1.0, 1.5, 0, 0);

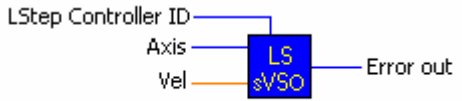
LS_MoveAbsTVROSingleAxis	
Description:	Position single axis absolute
Delphi:	function LS_MoveAbsTVROSingleAxis (Axis: Integer; Value: Double; Wait: LongBool): Integer; function LSX_MoveAbsTVROSingleAxis (LSID: Integer; Axis: Integer; Value: Double; Wait: LongBool): Integer;
C++:	int MoveAbsTVROSingleAxis (int IAxis, double dValue, BOOL bWait);
LabView:	 <p>LS4X MoveAbsTVROSingleAxis.vi</p>
Parameters:	Axis: (X, Y, Z, A numeriert von 1 bis 4) Value: Position (Input depends on the set dimension)
Example:	LS.MoveAbsTVROSingleAxis (2, 10.0); //Position additional Y-axis auf 10mm absolut

LS_MoveAbsTVRO	
Description:	Move to absolute position
	(The additional axes x, y, z and a are positioned on the given position values)
Delphi:	function LS_MoveAbsTVRO(X, Y, Z, A: Double; Wait: LongBool): Integer; 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:	 <p>LS4X MoveAbsTVRO.vi</p>
Parameters:	X, Y, Z and A +- Moving range Input depends on the set dimension Wait: Indicates, if the function after reaching the position (= true) or should directly return(= false)
Example:	LS.MoveAbsTVRO (10.0, 10.0, 10.0, 10.0, true);

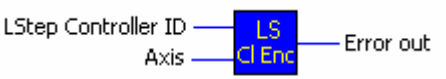
LS_MoveRelTVROSingleAxis	
Description:	Move single axis absolute
Delphi:	function LS_MoveRelTVROSingleAxis (Axis: Integer; Value: Double; Wait: LongBool): Integer; function LSX_MoveRelTVROSingleAxis (LSID: Integer; Axis: Integer; Value: Double; Wait: LongBool): Integer;
C++:	int MoveRelTVROSingleAxis (int lAxis,double dValue,BOOL Wait);
LabView:	
Parameters:	Axis: (X, Y, Z, A numbered starting 1 to 4) Value: Stretch (Input depends on the set dimension)
Example:	LS.MoveRelTVROSingleAxis (3, 5.0); // The additional Z-axis moves 5mm in positive direction

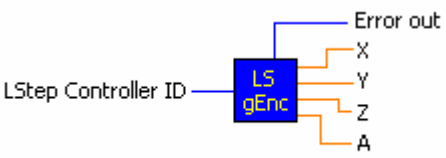
LS_MoveRelTVRO	
Description:	Move relative vector (The additional axes x, y, z and a are moved the length of the set vector)
Delphi:	function LS_MoveRelTVRO(X, Y, Z, A: Double; Wait: LongBool): Integer; function LSX_MoveRelTVRO(LSID: Integer; X, Y, Z, A: Double; Wait: LongBool): Integer;
C++:	int MoveRelTVRO (double dX, double dY, double dZ, double dR, BOOL bWait);
LabView:	
Parameters:	X, Y, Z and A +- Moving range Input depends on the set dimension) Wait: Indicates, if the function after reaching the position (= true) or should directly return(= false)
Example:	LS.MoveRelTVRO(10.0, 10.0, 10.0, 10.0, true);

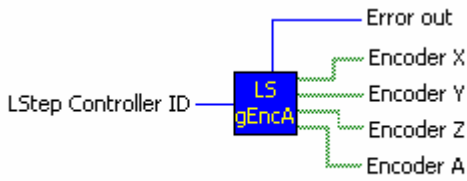
LS_SetAccelSingleAxisTVRO	
Description:	Acceleration of single additional axis
Delphi:	function LS_SetAccelSingleAxisTVRO(Axis: Integer; Accel: Double): Integer; function LSX_SetAccelSingleAxisTVRO (LSID: Integer; Axis: Integer; Accel: Double): Integer;
C++:	int SetAccelSingleAxisTVRO (int lAxis, double dAccel);
LabView:	
Parameters:	Axis: (X, Y, Z, A numbered starting 1 to 4) Accel: 0.01 – 1500 [U/s ²]
Example:	LS.SetAccelSingleAxis(2, 50.0); // The Z-axis will be accelerated with 50 rp/s ²

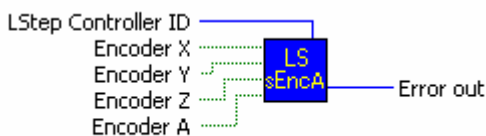
LS_SetVelSingleAxisTVRO	
Description:	Set acceleration of single additional axis
Delphi:	function LS_SetVelSingleAxisTVRO(Axis: Integer; Vel: Double): Integer; function LSX_SetVelSingleAxisTVRO (LSID: Integer; Axis: Integer; Vel: Double): Integer;
C++:	int SetVelSingleAxisTVRO (int lAxis, double dVel);
LabView:	
Parameters:	Axis: (X, Y, Z, A numbered starting 1 to 4) Vel: 0 – 40.0 [U/s]
Example:	LS.SetVelSingleAxis(1, 10.0); //The X-axis should run with max. speed of 10 rp/s

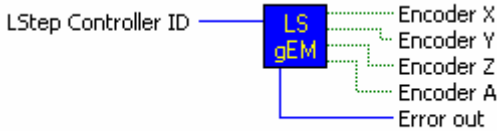
Encoder-Settings

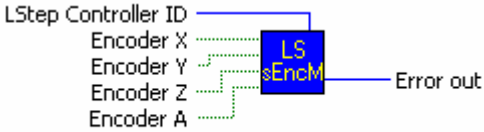
LS_ClearEncoder	
Description:	Set encoder-counter to zero
Delphi:	function LS_ClearEncoder(lAxis: Integer): Integer; function LSX_ClearEncoder (LSID: Integer; lAxis: Integer): Integer;
C++:	int ClearEncoder (int lAxis);
LabView:	 LS4X ClearEncoder.vi
Parameters:	lAxis: (X, Y, Z, A numbered starting 1 to 4)
Example:	LS. ClearEncoder (2); // Set encoder-counter of y-axis to zero

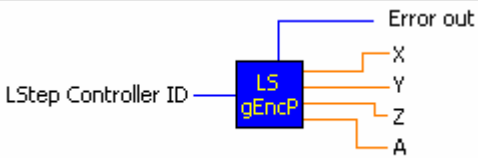
LS_GetEncoder	
Description:	Reads all encoder positions
Delphi:	function LS_GetEncoder(XP, YP, ZP, AP: Double): Integer; function LSX_GetEncoder (LSID: Integer; XP, YP, ZP, AP: Double): Integer;
C++:	int GetEncoder (double *pdXP, double *pdYP, double *pdZP, double *pdRP);
LabView:	 LS4X GetEncoder.vi
Parameters:	XP, YP, ZP, AP: Meter value, 4-fold interpoliert
Example:	LS. GetEncoder (&XP, &YP, &ZP, &AP);

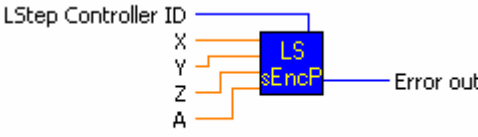
LS_GetEncoderActive	
Description:	Reads , which encoders are activated after the calibration.
Delphi:	function LS_GetEncoderActive(var Flags: Integer): Integer; function LSX_GetEncoderActive(LSID: Integer; var Flags: Integer): Integer;
C++:	int GetEncoderActive (int *plFlags);
LabView:	 <p style="text-align: center;">LS4X GetEncoderActive.vi</p>
Parameters:	Flags: Encoder mask
Example:	LS.GetEncoderActive(&Flags);

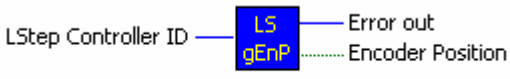
LS_SetEncoderActive	
Description:	This function is used to select which encoder is to be activated after calibration.
Delphi:	function LS_SetEncoderActive(Flags: Integer): Integer; function LSX_SetEncoderActive(LSID: Integer; Flags: Integer): Integer;
C++:	int SetEncoderActive (int IFlags);
LabView:	 <p style="text-align: center;">LS4X SetEncoderActive.vi</p>
Parameters:	Value: Encoder mask
Example:	LS.SetEncoderActive(0); // Deactivate all encoders LS.SetEncoderMask(2); // Activate Y-axis encoder


LS_GetEncoderMask	
Description:	Read encoder statuses
Delphi:	function LS_GetEncoderMask (var Flags: Integer): Integer; function LSX_GetEncoderMask (LSID: Integer; var Flags: Integer): Integer;
C++:	int GetEncoderMask (int *plFlags);
LabView:	 <p>LS4X GetEncoderMask.vi</p>
Parameters:	Flags: Encoder mask
Example:	<pre>int EncMask; LS.GetEncoderMask(&EncMask); if (EncMask & 2) ... // If Y-axis encoder is connected +active</pre>

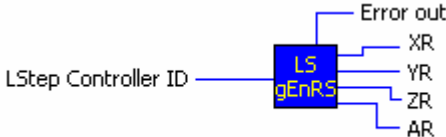
LS_SetEncoderMask	
Description:	(de-)activate encoder
Delphi:	function LS_SetEncoderMask(Value: Integer): Integer; function LSX_SetEncoderMask(LSID: Integer; Value: Integer): Integer;
C++:	int SetEncoderMask (int IValue);
LabView:	 <p>LS4X SetEncoderMask.vi</p>
Parameters:	Value: Encoder mask
Example:	<pre>LS.SetEncoderMask(0); // deactivate all encoder LS.SetEncoderMask(2); // activate encoder Y-axis</pre>

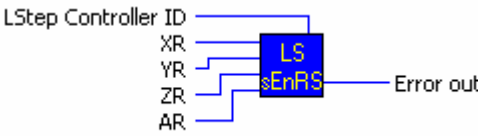
LS_GetEncoderPeriod	
Description:	Read length of encoder period
Delphi:	function LS_GetEncoderPeriod(var X, Y, Z, A: Double): Integer; function LSX_GetEncoderPeriod(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetEncoderPeriod (double *pdX, double *pdY, double *pdZ, double *pdR);
LabView:	 LS4X GetEncoderPeriod.vi
Parameters:	X, Y, Z and A: Period length [mm]
Example:	LS.GetEncoderPeriod(&X, &Y, &Z, &A);

LS_SetEncoderPeriod	
Description:	Set length of encoder period
Delphi:	function LS_SetEncoderPeriod(X, Y, Z, A: Double): Integer; function LSX_SetEncoderPeriod(LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetEncoderPeriod (double dX,double dY,double dZ,double dA);
LabView:	 LS4X SetEncoderPeriod.vi
Parameters:	X, Y, Z and A 0.0001 – Spindle pitch * 0.8 (mm)
Example:	LS.SetEncoderPeriod(0.1, 0.1, 0.1, 0.1); // Encoder period length is 0.1 mm for all axes

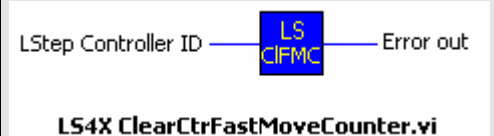
LS_GetEncoderPosition	
Description:	Read encoder position setting
Delphi:	function LS_GetEncoderPosition(Value: Boolean): Integer; function LSX_GetEncoderPosition(LSID: Integer; Value: LongBool): Integer;
C++:	int GetEncoderPosition (BOOL *pbValue);
LabView:	 <p>LS4X_GetEncoderPosition.vi</p>
Parameters:	Value = true → The encoder values of the detected encoders are displayed when the position inquiry is placed false → encoder position display is "OFF"
Example:	LS.GetEncoderPosition(&Value);

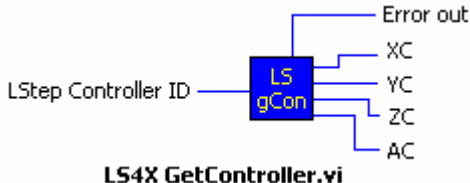
LS_SetEncoderPosition	
Description:	Encoder position display On/Off
Delphi	function LS_SetEncoderPosition(Value: Boolean): Integer; function LSX_SetEncoderPosition(LSID: Integer; Value: LongBool): Integer;
C++	int SetEncoderPosition (BOOL fValue);
LabView	 <p>LS4X_SetEncoderPosition.vi</p>
Parameters:	Value = true → The encoder values of the detected encoders are displayed when the position inquiry is placed
Example:	LS.SetEncoderPosition(true);

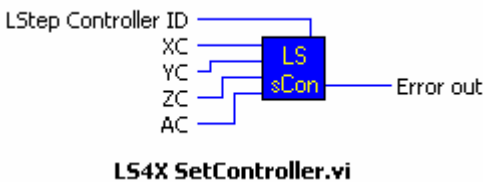
LS_GetEncoderRefSignal	
Description:	Reads if interpret reference signal from encoder when calibration is done
Delphi:	function LS_GetEncoderRefSignal(var XR, YR, ZR, AR: Integer): Integer; function LSX_GetEncoderRefSignal(LSID: Integer; var XR, YR, ZR, AR: Integer): Integer;
C++:	int GetEncoderRefSignal (int *pIXR, int *pIYR, int *pIZR, int *pIAR);
LabView:	 LS4X GetEncoderRefSignal.vi
Parameters:	X, Y, Z and A: 1 => When calibration the reference signal is interpreted. 0 => The reference signal is not interpreted
Example:	LS.GetEncoderRefSignal(&X, &Y, &Z, &A);

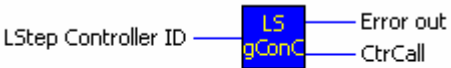
LS_SetEncoderRefSignal	
Description:	Interpret reference signal from encoder when calibration is done
Delphi:	function LS_SetEncoderRefSignal(XR, YR, ZR, AR: Integer): Integer; function LSX_SetEncoderRefSignal(LSID: Integer; XR, YR, ZR, AR: Integer): Integer;
C++:	int SetEncoderRefSignal (int IXR,int IYR,int IZR,int IAR);
LabView:	 LS4X SetEncoderRefSignal.vi
Parameters:	X, Y, Z and A 0 or 1
Example:	LS.SetEncoderRefSignal(1, 1, 0, 0); /* When calibration is done, the reference signal of the encoders x and y are interpreted. */


Controller Setting

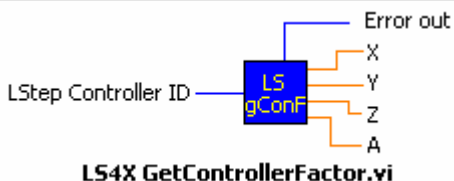
LS_ClearCtrFastMoveCounter	
Description:	<p>If the controller difference, is larger than the catch range, a new vector is started and the corresponding counter is extended by one.</p> <p>This function sets Fast Move Counters of all axis to zero.</p>
Delphi:	<pre>function LS_ClearCtrFastMoveCounter: Integer; function LSX_ClearCtrFastMoveCounter(LSID: Integer): Integer;</pre>
C++:	<pre>int ClearCtrFastMoveCounter;</pre>
LabView:	 <p>LS4X ClearCtrFastMoveCounter.vi</p>
Parameters:	
Example:	<pre>LS. ClearCtrFastMoveCounter;</pre>

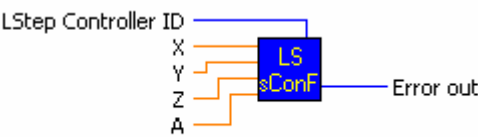
LS_GetController	
Description:	Read controller mode
Delphi:	function LS_GetController(var XC, YC, ZC, AC: Integer): Integer; function LSX_GetController(LSID: Integer; var XC, YC, ZC, AC: Integer): Integer;
C++:	int GetController (int *plXC, int *plYC, int *plZC, int *plRC);
LabView:	
Parameters:	Controller mode X, Y, Z and A : 0 → Controller "OFF" 1 → Controller "OFF after reaching target position" 2 → Controller "Always ON" 3 → Controller "OFF after reaching target position" with reduced current 4 → Controller "Always ON" with reduced current
Example:	LS.GetController(&X, &Y, &Z, &A);

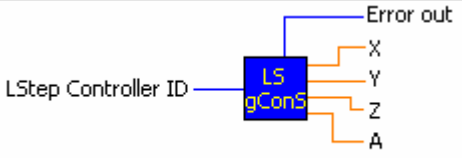
LS_SetController	
Description:	Set controller mode
Delphi:	function LS_SetController(XC, YC, ZC, AC: Integer): Integer; function LSX_SetController(LSID: Integer; XC, YC, ZC, AC: Integer): Integer;
C++:	int SetController (int lXC,int lYC,int lZC,int lAC);
LabView:	
Parameters:	Controller mode X, Y, Z and A : 0 → Controller "OFF" 1 → Controller "OFF after reaching target position" 2 → Controller "Always ON"
Example:	LS.SetController(1, 2, 0, 0);

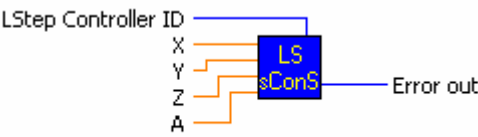
LS_GetControllerCall	
Description:	Reads controller call time
Delphi:	function LS_GetControllerCall(var CtrCall: Integer): Integer; function LSX_GetControllerCall(LSID: Integer; var CtrCall: Integer): Integer;
C++:	int GetControllerCall (int *pICtrCall);
LabView:	 <p style="text-align: center;">LS4X_GetControllerCall.vi</p>
Parameters:	CtrCall: Controller call time [ms]
Example:	LS.GetControllerCall(&CtrCall); // After function call CtrCall = 10 means: Controller call every 10 ms


LS_SetControllerCall	
Description:	Call controller
Delphi:	function LS_SetControllerCall(CtrCall: Integer): Integer; function LSX_SetControllerCall(LSID: Integer; CtrCall: Integer): Integer;
C++:	int SetControllerCall (int ICtrCall);
LabView:	 <p style="text-align: center;">LS4X_SetControllerCall.vi</p>
Parameters:	CtrCall: Controller call time [ms]
Example:	LS.SetControllerCall(10);

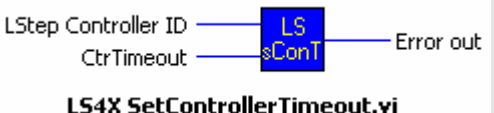
LS_GetControllerFactor	
Description:	Reads controller factor see chapt. 4.13 „Controller setting for LSTEP“
Delphi:	function LS_GetControllerFactor(var X, Y, Z, A: Double): Integer; function LSX_GetControllerFactor(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetControllerFactor (double *pdX, double *pdY, double *pdZ, double *pdR);
LabView:	 <p style="text-align: center;">LS4X GetControllerFactor.vi</p>
Parameters:	X, Y, Z and A: Controller factor
Example:	LS.GetControllerFactor(&X, &Y, &Z, &A);


LS_SetControllerFactor	
Description:	Controller factor
Delphi:	function LS_SetControllerFactor(X, Y, Z, A: Double): Integer; function LSX_SetControllerFactor(LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetControllerFactor (double dX,double dY,double dZ,double dA);
LabView:	 <p style="text-align: center;">LS4X SetControllerFactor.vi</p>
Parameters:	X, Y, Z and A 1 – 64
Example:	LS.SetControllerFactor(1, 2, 3, 4);

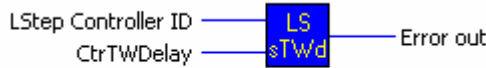
LS_GetControllerSteps	
Description:	Reads controller steps length
Delphi:	function LS_GetControllerSteps(var X, Y, Z, A: Double): Integer; function LSX_GetControllerSteps(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetControllerSteps (double *pdX, double *pdY, double *pdZ, double *pdR);
LabView:	 <p style="text-align: center;">LS4X GetControllerSteps.vi</p>
Parameters:	X, Y, Z and A: Controller steps length [mm]
Example:	LS.GetControllerSteps(&X, &Y, &Z, &A);

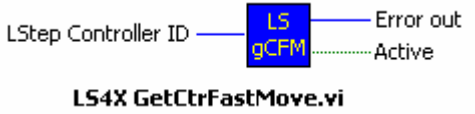
LS_SetControllerSteps	
Description:	Controller steps
Delphi:	function LS_SetControllerSteps(X, Y, Z, A: Double): Integer; function LSX_SetControllerSteps(LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetControllerSteps (double dX,double dY,double dZ,double dA);
LabView:	 <p style="text-align: center;">LS4X SetControllerSteps.vi</p>
Parameters:	X, Y, Z and A 1 – Spindle pitch (Values depend on the dimension)
Example:	LS.SetControllerSteps(4, 5, 7, 9);

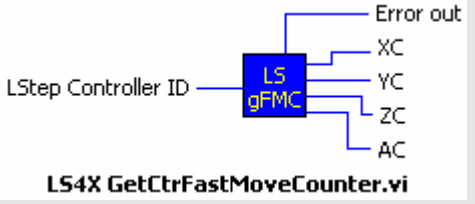
LS_GetControllerTimeout	
Description:	Reads controller timeout
Delphi:	function LS_GetControllerTimeout(var ACtrTimeout: Integer): Integer; function LSX_GetControllerTimeout(LSID: Integer; var ACtrTimeout: Integer): Integer;
C++:	int GetControllerTimeout (int *plACtrTimeout);
LabView:	 <p style="text-align: center;">LS4X GetControllerTimeout.vi</p>
Parameters:	ACtrTimeout: Timeout [ms], time after which a travel command returns with an error message (error code 4013) , if the controller could not definitively find a position.
Example:	LS.GetControllerTimeout(&ACtrTimeout);

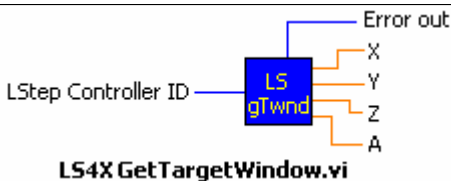
LS_SetControllerTimeout	
Description:	Controller timeout
Delphi:	function LS_SetControllerTimeout(ACtrTimeout: Integer): Integer; function LSX_SetControllerTimeout(LSID: Integer; ACtrTimeout: Integer): Integer;
C++:	int SetControllerTimeout (int ACtrTimeout);
LabView:	 <p style="text-align: center;">LS4X SetControllerTimeout.vi</p>
Parameters:	ACtrTimeout: Timeout [ms], time after which a travel command returns with an error message (error code 4013) , if the controller could not definitively find a position.
Example:	LS.SetControllerTimeout(500);

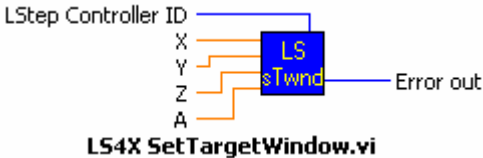
LS_GetControllerTWDelay	
Description:	Read controller relay
Delphi:	function LS_GetControllerTWDelay(var CtrTWDelay: Integer): Integer; function LSX_GetControllerTWDelay(LSID: Integer; var CtrTWDelay: Integer): Integer;
C++:	int GetControllerTWDelay (int *pICtrTWDelay);
LabView:	 <p style="text-align: center;">LS4X GetControllerTWDelay.vi</p>
Parameters:	CtrTWDelay: Controller delay [ms]
Example:	LS.GetControllerTWDelay(&CtrTWDelay);


LS_SetControllerTWDelay	
Description:	Controller delay
Delphi	function LS_SetControllerTWDelay(CtrTWDelay: Integer): Integer; function LSX_SetControllerTWDelay(LSID: Integer; CtrTWDelay: Integer): Integer;
C++	int SetControllerTWDelay (int lCtrTWDelay);
LabView	 <p style="text-align: center;">LS4X SetControllerTWDelay.vi</p>
Parameters:	CtrTWDelay: Controller delay 0 - 100 [ms]
Example:	LS.SetControllerTWDelay(0); // Controller delay off


LS_GetCtrFastMove	
Description:	Reads setting of the Fast Move Function
Delphi:	function LS_GetCtrFastMoveOff(var bActive: LongBool): Integer; function LSX_GetCtrFastMoveOff(LSID: Integer; var bActive: LongBool): Integer;
C++:	int GetCtrFastMove (BOOL *pbActive);
LabView:	 <p style="text-align: center;">LS4X GetCtrFastMove.vi</p>
Parameters:	bActive: True => Fast Move Funktion active
Example:	LS. GetCtrFastMoveOff (&bActive);

LS_GetCtrFastMoveCounter	
Description:	<p>In a regulator difference, that is larger than the capture area, a new vector is started and the Counter is raised by one.</p> <p>The Function delivers Fast Move Counters</p>
Delphi:	function LS_GetCtrFastMoveCounter(var XC, YC, ZC, RC: Integer): Integer; function LSX_GetCtrFastMoveCounter(LSID: Integer; var XC, YC, ZC, RC: Integer): Integer;
C++:	int GetCtrFastMoveCounter (int *plXC, int *plYC, int *plZC, int *plRC);
LabView:	 <p style="text-align: center;">LS4X GetCtrFastMoveCounter.vi</p>
Parameters:	XC, YC, ZC, RC: Amount of finished Fast Move functions
Example:	LS. SetCtrFastMoveCounter (&XC, &YC, &ZC, &RC);


LS_GetTargetWindow	
Description:	Reads the target window
Delphi:	function LS_GetTargetWindow(var X, Y, Z, A: Double): Integer; function LSX_GetTargetWindow(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++:	int GetTargetWindow (double *pdX, double *pdY, double *pdZ, double *pdA);
LabView:	 <p style="text-align: center;">LS4X GetTargetWindow.vi</p>
Parameters:	X, Y, Z and A: Target window , depend on the dimension)
Example:	LS.GetTargetWindow(&X, &Y, &Z, &A);

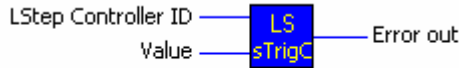
LS_SetTargetWindow	
Description:	Target window
Delphi:	function LS_SetTargetWindow(X, Y, Z, A: Double): Integer; function LSX_SetTargetWindow(LSID: Integer; X, Y, Z, A: Double): Integer;
C++:	int SetTargetWindow (double dX,double dY,double dZ,double dA);
LabView:	 <p style="text-align: center;">LS4X SetTargetWindow.vi</p>
Parameters:	X, Y, Z and A 1 – 25000 (motor increments) 0.1 – Spindle pitch/2 (µm) 0.0001 – Spindle pitch/2 (mm) (Values depend on the dimension)
Example:	LS.SetTargetWindow(1.0, 0.002, 1.0, 1.0);


LS_SetCtrFastMoveOff	
Description:	Deactivate Fast Move Function
Delphi:	function LS_SetCtrFastMoveOff: Integer; function LSX_SetCtrFastMoveOff(LSID: Integer): Integer;
C++:	int SetCtrFastMoveOff ();
LabView:	 LS4X SetCtrFastMoveOff.vi
Parameters:	
Example:	LS. SetCtrFastMoveOff ();


LS_SetCtrFastMoveOn	
Description:	Fast Move function activated, i. e. In a regulator difference, that is larger than the capture area, a new vector is started.
Delphi:	function LS_SetCtrFastMoveOn: Integer; function LSX_SetCtrFastMoveOn(LSID: Integer): Integer;
C++:	int SetCtrFastMoveOn ();
LabView:	 LS4X SetCtrFastMoveOn.vi
Parameters:	
Example:	LS. SetCtrFastMoveOn();

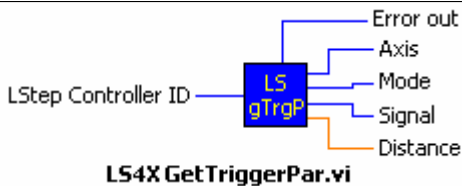
Trigger-Output

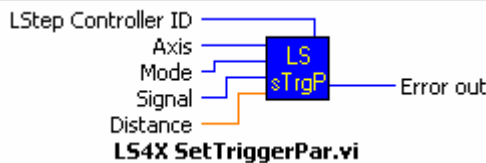
LS_GetTrigCount	
Description:	Read Trigger counter.
Delphi:	function LS_GetTrigCount (var Value: Integer): Integer; function LSX_GetTrigCount (LSID: Integer; var Value: Integer): Integer;
C++:	int GetTrigCount (int *pValue);
LabView:	 LS4X GetTrigCount.vi
Parameters:	Value: Amount of the executed Trigger
Example:	LS.GetTrigCount (&Value);

LS_SetTrigCount	
Description:	Set Trigger counter.
Delphi:	function LS_SetTrigCount (Value: Integer): Integer; function LSX_SetTrigCount (LSID: Integer; Value: Integer): Integer;
C++:	int SetTrigCount (int Wert);
LabView:	 LS4X SetTrigCount.vi
Parameters:	Value: 0 to 2147483647
Example:	LS.SetTrigCount (0);


LS_GetTrigger	
Description:	Reads the current Trigger-Condition
Delphi:	function LS_GetTrigger(var ATrigger: LongBool): Integer; function LSX_GetTrigger(LSID: Integer; var ATrigger: LongBool): Integer;
C++:	int GetTrigger (BOOL *pbATrigger);
LabView:	
Parameters:	ATrigger: True => Trigger „ON“ False => Trigger „OFF“
Example:	LS.GetTrigger(&ATrigger);


LS_SetTrigger	
Description:	Trigger On/Off
Delphi:	function LS_SetTrigger(ATrigger: LongBool): Integer; function LSX_SetTrigger(LSID: Integer; ATrigger: LongBool): Integer;
C++:	int SetTrigger (BOOL bATrigger);
LabView:	
Parameters:	ATrigger: Trigger On/Off
Example:	LS.SetTrigger(true);


LS_GetTriggerPar	
Description:	Reads Trigger-Parameter
Delphi:	function LS_GetTriggerPar(var Axis, Mode, Signal: Integer; var Distance: Double): Integer; function LSX_GetTriggerPar(LSID: Integer; var Axis, Mode, Signal: Integer; var Distance: Double): Integer;
C++:	<pre>int GetTriggerPar (int *plAxis, int *plMode, int *plSignal, double *pdDistance);</pre>
LabView:	
Parameters:	Axis: Axis (1..4) Mode: Trigger mode (see command !trigm) Signal: Trigger signal (see command !trigs) Distance: Trigger distance (see command !trigd)
Example:	LS.GetTriggerPar(&Axis, & Mode, & Signal, & Distance);


LS_SetTriggerPar	
Description:	Trigger parameters
Delphi:	function LS_SetTriggerPar(Axis, Mode, Signal: Integer; Distance: Double): Integer; function LSX_SetTriggerPar(LSID: Integer; Axis, Mode, Signal: Integer; Distance: Double): Integer;
C++:	<pre>int SetTriggerPar (int lAxis, int lMode, int lSignal, double dDistance);</pre>
LabView:	
Parameters:	Axis: Axis (1..4) Mode: Trigger mode (see command !trigm) Signal: Trigger signal (see command !trigs) Distance: Trigger distance (see command !trigd)
Example:	LS.SetTriggerPar(1, 3, 2, 5.0);


Snapshot-Input

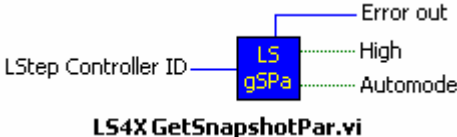
LS_GetSnapshot	
Description:	Reads the current Snapshot-condition
Delphi:	function LS_GetSnapshot(var ASnapshot: LongBool): Integer; function LSX_GetSnapshot(LSID: Integer; var ASnapshot: LongBool): Integer;
C++:	int GetSnapshot (BOOL *pbASnapshot);
LabView:	 <p>LS4X GetSnapshot.vi</p>
Parameters:	ASnapshot: True => Snapshot „ON“ False => Snapshot „OFF“
Example:	LS.GetSnapshot(&ASnapshot);


LS_SetSnapshot	
Description:	Snapshot On/Off
Delphi:	function LS_SetSnapshot(ASnapshot: LongBool): Integer; function LSX_SetSnapshot(LSID: Integer; ASnapshot: LongBool): Integer;
C++:	int SetSnapshot (BOOL bASnapshot);
LabView:	 <p>LS4X SetSnapshot.vi</p>
Parameters:	ASnapshot: Snapshot On/Off
Example:	LS.SetSnapshot(true);

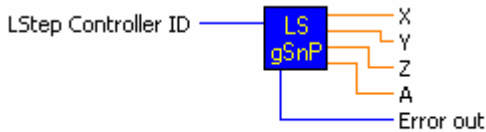
LS_GetSnapshotCount	
Description:	Reads Snapshot counter
Delphi:	function LS_GetSnapshotCount(var SnsCount: Integer): Integer; function LSX_GetSnapshotCount(LSID: Integer; var SnsCount: Integer): Integer;
C++:	int GetSnapshotCount (int *pSnsCount);
LabView:	 LS4X GetSnapshotCount.vi
Parameters:	SnsCount: Snapshot counter
Example:	LS.GetsnapshotCount(&SnsCount);

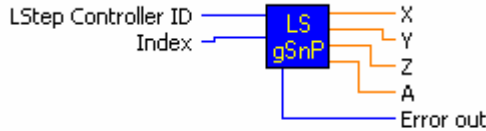
LS_GetSnapshotFilter	
Description:	Reads input filter (snapshot-filter)
Delphi:	function LS_GetSnapshotFilter(var lTime: Integer): Integer; function LSX_GetSnapshotFilter(LSID: Integer; var lTime: Integer): Integer;
C++:	int GetSnapshotFilter (int *plTime);
LabView:	 LS4X GetSnapshotFilter.vi
Parameters:	lTime: Filter time [ms]
Example:	LS. GetSnapshotFilter(&lTime);

LS_SetSnapshotFilter	
Description:	Set input filter for rebounding switches.
Delphi:	function LS_SetSnapshotFilter(lTime: Integer): Integer; function LSX_SetSnapshotFilter(LSID: Integer; lTime: Integer): Integer;
C++:	int SetSnapshotFilter (int lTime);
LabView:	 LS4X SetSnapshotFilter.vi
Parameters:	lTime: Filter time, value range 0 - 100 ms
Example:	LS. SetSnapshotFilter(0); // no Snapshot filter

LS_GetSnapshotPar	
Description:	Read Snapshot-Parameter
Delphi:	function LS_GetSnapshotPar(var High, AutoMode: LongBool): Integer; function LSX_GetSnapshotPar(LSID: Integer; var High, AutoMode: LongBool): Integer;
C++:	int GetSnapshotPar (BOOL *pbHigh, BOOL *pbAutoMode);
LabView:	
Parameters:	High: Snapshot high-active False => Low- active AutoMode: True => Snapshot „Automatic “.The position is automatically approached after the first impulse.
Example:	LS.GetSnapshotPar(&High, & AutoMode);

LS_SetSnapshotPar	
Description:	Snapshot parameters
Delphi:	function LS_SetSnapshotPar(High, AutoMode: LongBool): Integer; function LSX_SetSnapshotPar(LSID: Integer; High, AutoMode: LongBool): Integer;
C++:	int SetSnapshotPar (BOOL bHigh, BOOL bAutoMode);
LabView:	
Parameters:	High: Snapshot high-active AutoMode: approach snapshot position automatically
Example:	LS.SetSnapshotPar(true, false);

LS_GetSnapshotPos	
Description:	Read snapshot position
Delphi	function LS_GetSnapshotPos(var X, Y, Z, A: Double): Integer; function LSX_GetSnapshotPos(LSID: Integer; var X, Y, Z, A: Double): Integer;
C++	int GetSnapshotPos (double *pdX, double *pdY, double *pdZ, double *pdA);
LabView:	 <p style="text-align: center;">LS4X GetSnapshotPos.vi</p>
Parameters:	X, Y, Z, A: Positional values
Example:	double X, Y, Z, A; LS.GetSnapshotPos(&X, &Y, &Z, &A);

LS_GetSnapshotPosArray	
Description:	Read snapshot-position from array
Delphi:	function LS_GetSnapshotPosArray(Index: Integer; var X, Y, Z, R: Double): Integer; 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:	 <p style="text-align: center;">LS4X GetSnapshotPosArray.vi</p>
Parameters:	Index: Number of snapshot-position (1-200) X, Y, Z, A: Position value
Example:	double X, Y, Z, A; LS.GetSnapshotPos(2, &X, &Y, &Z, &A);

9.5 Error Codes

LStep-number.	API-number	Description:
0	0	No error
	4001,4002	Internal error
	4003	undefined error
	4004	Interface type unknown (may occur with Connect..)
	4005	Interface initialization error
	4006	No connection to the controller (e.g. when SetPitch is called before Connect)
	4007	Timeout whilst reading from the interface
	4008	Command transmission error to LSTEP
	4009	Command terminated (with SetAbortFlag)
	4010	Command not supported by API
11	4011	Joystick set to Manual (may occur with SetJoystickOn/Off)
11	4012	Travel command not possible, as joystick is in Manual
	4013	Controller timeout
12	4015	actuates limit switch in moving direction
14	4017	Fault during calibration (Limit switch was not set free correctly)
1	4101	Valid axis designation missing
2	4102	Non-executable function
3	4103	Command string has too many characters
4	4104	Invalid command
5	4105	Not within valid numerical range
6	4106	Incorrect number of parameters
7	4107	None !or ?
8	4108	TVR not possible because axis is active
9	4109	Axes cannot be switched on or off because TVR is active
10	4110	Function not configured
11	4111	Move command not possible, as joystick is in Manual
12	4112	Limit switch tripped
13	4113	Function cannot be carried out because Encoder was recognized
14	4114	Fault during calibration (Limit switch was not set free correctly)
15	4115	This function is interrupted activated while releasing the encoder during calibrating or table stroke measuring if the opposite encoder is activated.
20	4120	Driver relay defective (safty circle K3/K4)
21	4121	Only single vectors may be driven (setup mode)
22	4122	No calibrating, measuring table stroke or joystick operation can be carried out (door open or setup mode)
23	4123	SECURITY Error X-axis
24	4124	SECURITY Error Y- axis
25	4125	SECURITY Error Z- axis
26	4126	SECURITY Error A- axis
27	4127	Stop activ
28	4128	Fault in the door switch safty circle (only with LS44/Solero)
29	4129	Power stages are not switched on (only with LS44)
30	4130	GAL security error (only with LS44)
31	4131	Joy-stick can not be activated because Move is active

9.6 Frequent questions & answers

How are the LSTEP4.DLL and/or the LSTEP4X.DLL are combined in an M Visual C + + project?
How do I initialise with the LStep API the connection to the LStep?
Which of the Connect-commands should be used?
How do I install the driver for the LStep-PCI?
Why does my program with the LSTEP4.DLL get no connection to the LStep-PCI?
A fault occurred in the process, in the LSTEP4.DLL or in my program. What is the cause for that problem, and how can I solve it.
Can I make an inquiry during moving commands about the status of inputs, the current position and something similar to that?
Why are messages processed during the run of LSTEP-API-functions and how can I deactivate this.
When should Moves with or without Wait be used?
How can I move single axes independently from one another with the LSTEP-API?
How can I use several LStep-PCI-cards in a PC?
Why does Windows requires occasional the driver for the LStep-PCI after new starts, although it was already installed?
When should LSTEP4.DLL, when the LSTEP4X.DLL be used?
Is the LSTEP-API compatible to the MCL resp. to the old register-command set?
Why do I get in MS Visual C++ while connecting the LStep4.cpp the message "fatal error C1010"?
How can I use a special /new LStep-command, which has no suitable LSTEP-API-function?
Why do I see in my debugger of my development environment when using the LSTEP-API the message „First chance exception“, „Exception: Timeout read RS232!“ or something similar?
How can I simulate a sort of joystick with the LSTEP-API, which means moving an axis until a key is released?
How can I save durable the adjustments of the LStep?

How are the LSTEP4. DLL and/or the LSTEP4X.DLL are embedded in a M Visual C bound + + project?

- create project
- copy LSTEP4.DLL, LSTEP4.h, LSTEP4.cpp in a project folder
- insert LSTEP4.h and LSTEP4.cpp in the project
- Menu: Select Project\Adjustment\ C/C++ Option: [do not use pre compiled]
- embed in LSTEP4.h #include „stdafx.h“
- in project name_Dlg.h #include „LSTEP4.h“
- embed the required instance in public

Example: CLStep* MyLStep = new CLStep();

The embedding of the LSTEP4X.DLL is analgue to this \procedure.

How do I initialise with the LStep API the connection to the LStep?

Which of the Connect-commands should be used?

The connection to the LSTEP-API is initialised with a Connect-command (Connect, ConnectEx, ConnectSimple). Besides some special cases, **ConnectSimple** should always be used.

ConnectSimple	when transferring the interfaces parameter directly
Connect	after pre loading of the interfaces parameter out of a .ini file using LoadConfig
ConnectEx	when loading the interface parameter out of a data structure

How do I initialise the driver for the LStep-PCI?

After the correct installation of the LStep-PCI Windows requests a driver during the start for a device of the type „network controller“. In this dialog-window click to button „Search“ o.s.s and switch to the folder, in which the files of the LSTEP-API where unpacked. The driver-files and the Inf-files are In the sub folder „LStepPCI“, which are required for the driver-installation.

Why does my program with the LSTEP4.DLL get no connection to the LStep-PCI?

They should review first of all in the Windows-device-manager, whether the installed LStep-PCI is registered as a device. Also the **file DRVX40.DLL must be in the folder of the LSTEP4.DLL, of your program or in a Windows-system folder**. You find this file in the sub folder „LStepPCI“ of the LStep API (*see also chapter 9.7*).

A fault occurred during the process, in the LSTEP4.DLL or in my program. What is the cause for that problem, and how can I solve it.

So that a fault diagnosis is possible, you should unconditional switch on the recording of the LSTEP-API with SetWriteLogText. Afterwards you should try to reproduce the fault while recording it. You can send the Log-file (LStep4.log) to us, to e analysed.

Can I make an inquiry during moving commands about the status of inputs, the current position and somthig similar to that?

Yes, for example by calling GetDigitalInputs via a Windows-timer or a second Thread function like GetPos, during the moving command. But it is not possible to call the function WaitForAxisStop, during a moving command with Wait=true.

Why are messages processed during the run of LSTEP-API-functions and how can I deactivate this?

While from moving commands, the LSTEP-API processes messages, so that the program does not „stand still“, otherwise it would not be possible to perform an interrupt in case of a fault or to stop the axes. SetProcessMessagesProc enables the replcement of a internal Message-Dispatching procedure of the LSTEP-API. The LSTEP-API processes messages during waiting for acknowlegements of the LStep in the Main-Thread. If you want to turn off the Message-Dispatching or replace it with your own code, you can use SetProcessMessagesProc to set Callback-procedure.

When should Moves with or without Wait be used?

Move-commands with WaitForAxisStop are to be used, if all axes supposed to move synchronic and linearly interpolated. The controller accepts new Move-commands only after all axes are stopped.

Move-commands without WaitForAxisStop are to be used, if all axes supposed to move asynchronous. In this case the user has to make sure, that only the axis stands still that gets a new Move-command.

How can I move single axes independently from one another with the LSTEP-API?

The Move-command of the LSTEP-API offers two possibilities: If the (last) parameter is set Wait=true, the function only returns, after the axes reach there goal position. If the parameter is set Wait=false, the LSTEP-API-function only sends the Move-command and returns immediatly, without performing the movement.

That means by using MoveAbsSingleAxis with Wait=false for the X-axis verwendet and some time later call MoveAbsSingleAxis with Wait=false for the Y-axis, the axis can be moved seperately. In order to find out, if the axis reached there goal position the command WaitForAxisStop can be used.

Example:

```
LS.MoveAbsSingleAxis(Xaxis, 10, false);    // Move X-axis asynchronous
Delay(1000);                               // Wait 1s to start the Y-axis
LS.MoveAbsSingleAxis(Yaxis, 20, false);    // Move the Y-axis asynchronous
LS.WaitForAxisStop(3, 0, flag);            // Wait until X- and Y-axis stopped,
                                           without timeout
```

But it is **not possible to use Move-commands with Wait=true and the once with Wait=false simultaneous**. This leads to permanent or sporadic faults in the communication.

Example:

Not allowed:

```
LS.MoveAbsSingleAxis(Xaxis, 10, false);    // Move the X-axis asynchronous
LS.MoveAbsSingleAxis(Yaxis, 20, true);     // Move the Y-axis asynchronous without
                                           waiting for the end of the asynchronous
                                           Move-command.
```

How can I use several LStep-PCI-cards in a PC?

The procedure for the installation is like the one for single card. After the start, Windows requests the driver for all LStep-PCI-cards.

Yet it is **problematically to identify, which physical card belongs to a certain index-number**. It is not guaranteed, that with LS_ConnectSimple(4, nil, 0, true) a connection to LStep-PCI in the first PCI-Slot of the mainboard, with LS_ConnectSimple(4, nil, 1, true) a connection to LStep-PCI in the second PCI-Slot will be achieved etc. Therefore to the unambiguous identification of the cards, the series number with GetSerialNr should be queried.

When should LSTEP4.DLL, when the LSTEP4X.DLL be used?

If several LSteps/LStep-PCI-cards supposed to be controlled from one PC, LSTEP4X.DLL should be used; otherwise LSTEP4.DLL is more suitable.

Is the LSTEP-API compatible to the MCL resp. to the old register-command-set?

The LSTEP-API is principle down compatible to the Register-command-set, that other MCL and older LSteps communicate with. However this command-set does not offer many possibilities, that the LStep API can in controllers related to new command-set. Therefore some LSTEP-API-commands cannot generally be used as well as WaitForAxisStop in controls with an old command-set

Why do I get in MS Visual C++ while connecting the LStep4.cpp the message "fatal error C1010"?

IN this case it is not a fault in the file LStep4.cpp. The message usually appears, if the compiler is looking for a pre compiled header-file and can not find it. If in MS Visual C++ the message „fatal error C1010 precompiled header files“ appears, the option „pre compiled Header-file“ for LStep4.cpp must be turned off. If you don't want to use the MFC in your project, you should erase the #include "stdafx.h" from LStep4.cpp.

How can I use a special /new LStep-command, which has no suitable LSTEP-API-function?

The LSTEP-API-function **SendString** offers the possibility, to use new LStep commands, that were not planned for the LSTEP-API. Note, that all commands close with #13 resp. \r !

Why do I see in my debugger of my development environment when using the LSTEP-API the message „First chance exception“, „Exception: Timeout read RS232!“ or something similar?

Internal Exceptions of the LSTEP4.DLL, which are only visible in the debugger have no meaning. They serve the internal process control. In ConnectSimple frequently an Exception appears, because the LSTEP-API tries, to find out the command-set. At the same time it comes to a Timeout if the controller does not support the tested command-set. In Delphi, you can add in the Debugger-options the corresponding Exception to the Exceptions to be ignored by Debugger.

How can I simulate a sort of joystick with the LSTEP-API, which means moving an axis until a key is released?

Such a key-Joystick can be implemented as follows:

Start the axis with a very long vector during a keystroke

MoveRelSingleAxis(Xaxis, 100000, false)

Important is to set the parameter Wait=false

When releasing the key call the command **StopAxes**

How can I save durable the adjustments of the LStep?

The LSTEP-API-command **LstepSave** can be used to keep settings (spindle pitch, gear factor, axes current a.s.o.) made once, even after a reset of the LStep. If your LStep supports this command you can read in your documentation.

How many entries will fit into the log window of the LStep-API?

The log window of the LStep-API can contain more than 20.000 logs. Arise more logs, the oldest one will overwritten.

How many logs can be written into the log file?

You can write into the log file so long until the programme will be finished or the hard disk is full.

9.7 Use of the LStep PCI-card

The LStep API (LSTEP4.DLL and LSTEP4X.DLL) supports starting version 1.0.7.0 the PCI-plugging card **LStep-PCI** under the operating systems Windows 95, 98, NT 4.0 and 2000.

For the operation a driver is required, that is located in the sub folder 'LStepPCI'. Under Windows 9x/2000 the installation of the driver after the automatic recognition of the LStep-PCI via the operating system the .INF-file LSPCIW9X.INF resp. LSPCIW2K.INF must be selected. The installation of the driver under Windows NT 4.0 based on operating systems (Windows NT, Windows 2000, Windows XP) can be done with help of the Tools SetupDrvXNT.exe (from API-version 1.2.0.20 SetupDrvX.exe) Therefore the file needs to be executed one time after the installation of the card

Example to the Initialisation of the LStep API with a LStep PCI:

S_ConnectSimple (4, nil, 0, true);
(4 = LS_if_PCI)

The third parameter indicates the index of the card. If in a PC several LStep-PCI-cards are installed, they will be numbered starting 0 to n-1.

Besides the initialisation with LS_ConnectSimple the function calls of the LStep API do not differ from the once that are used with a normal LStep, so that one and the same program when using the LStep API with minnum changes in the source code can control both a LStep via RS232 as well a LStep-PCI.

The file DRVX40.DLL (located in the sub folder 'LStepPCI') should be in the same folder as the LSTEP4.DLL, so that the LStep-PCI can be used. (Or in a path that is registered in enviroment variable PATH)

9.7.1 Interrupt-controlled Communication with LStep-PCI

The LStep4.D11 (LStep4X.D1f) communicates from version 1.2.0.20 with the LStep-PCI card by interrupt. This increases the transfer rate of Move and Status-commands significant.

It will be not recommended to put 2 LStep-PCI cards into the slots which divides one Interrupt Request Line

(IRQ) This can be verified in the hardware manager. (Start => set-up => system control => system => hardware => equipment manager => LStep => LStep-PCI => resources)

In case 2 LStep-PCI cards divides one Interrupt Request Line and communication shall take place with both cards at the same time, then Lstep4X.D11 will communicate with one of both cards through pollen. With this the communication will be a little bit more slowly.

If a LStep-PCI card still works with the old firmware, the new Lstep4DLL (Lstep4X.D11) will communicate with pollen.

Starting the Firmware version 38 / internal Version T02.21.05 the communication is interrupt controlled

After installation of the driver the new set-up driverX.exe has to be executed. With this the driver will be configured in that way, that the interrupt-controlled communication will be possible.

Please pay attention to the Readme.txt file !

9.7.2 Readme

Installation of the driver for LStep-

Windows NT

- 1) Copy the files DRIVERX.SYS, DRVX40.DLL, SetupDrvX.exe in a folder.
- 2) 2) Start SetupDrvX.exe.

Windows 9x

- 1) Copy the files DRIVERX.VXD, DRVX40.DLL, LSPCIW9X.INF, SetupDrvX.exe in a folder
- 2) Install driver with the hardware assistance
- 3) start SetupDrvX.exe, to configure the driver.

It doesn't matter if step 2) is carried out before step 3) or afterwards.

For X LStep-PCI- card, step 2) must be carried out X times..

Windows 2000, XP

- 1) Copy the files DRIVERX.VXD, DRVX40.DLL, LSPCIW9X.INF, SetupDrvX.exe in a folder
- 2) Install driver with the hardware assistance
- 3) start SetupDrvX.exe, to configure the driver.

It doesn't matter if step 2) is carried out before step 3) or afterwards.

For X LStep-PCI- card, step 2) must be carried out X times..

9.7.3 API / LSTEP Commands

API-Command	Short description	LSTEP-Command
Connect	Connect with LSTEP	-
ConnectEx	Connect with LSTEP	-
ConnectSimple	Connect with LSTEP	-
CreateLSID	Creates an ID No for the use of the LSTEP4X APIs	-
Disconnect	Disconnect LSTEP	-
EnableCommandRetry	With this function repeated sending of commands can be switched On/Off in case of a fault.	-
FlushBuffer	Delete the input buffer	-
FreeLSID	Sets the created ID No free again	-
LoadConfig	Load LSTEP configuration (interface, axis settings, controllers) from INI-file.	-
SaveConfig	Save LSTEP configuration (interface, axis settings, controllers) into INI-file.	-
SendString	Send string to LSTEP	-
SendStringPosCmd	Moving command, which awaits confirmation , send to LSTEP as a string	-
SetAbortFlag	Set flag to terminate the communication with the LSTEP	-
SetControlPars	Transmits the parameters, which were loaded with LS_LoadConfig to the LSTEP.	-
SetCorrTblOff	deactivates axis	-
SetCorrTblOn	activates axis correction in x/y-Matrix with linear interpolation	-
SetExtValue	switch on extensions	-
SetFactorMode	Position value-Conversion for ‚krumme‘ spindle pitch	-
SetLanguage	Set language for LSTEP-API (log / messages)	-
SetProcessMessagesProc	Enables the replacement of the internal message-dispatching procedure of the LStep API	-
SetShowCmdList	LStep-API command list On/Off	-
SetShowProt	Interface protocol On/ Off	-
SetWriteLogText	Switch on / switch off write log file LSTEP4.log (Writing in LSTEP4-log is normally switched off)	-
SetWriteLogTextFN	switch On/Off writing of the interface-protocol in a certain file	-

Controller-Info

API-Command	Short description	LSTEP-Command
GetSerialNr	Read serial number of the controller	?readsn
GetVersionStr	Returns the current version number of the Firmware	?ver
GetVersionStrDet	Read detailed version number of the firmware	?det
GetVersionStrInfo	Gives detailed information about version number	?iver

Settings

API-Command	Short description	LSTEP-Command
GetAccel	Inquiry of acceleration	?accel
GetActiveAxes	Delivers enable axes	?axis
GetAxisDirection	Inquiry of reverse-turning direction	?axisdir
GetCalibBackSpeed	Reads the speed, with which the axis are moved back during calibration	?calbspeed
GetCaliboffset	Inquiry of calibration-offset	?caliboffset
GetCalibrateDir	Inquiry reverse preceding sign when calibrating	?caldir
GetCurrentDelay	Indicates time delay for current reduction	?curdelay
GetDimensions	Inquiry dimensions of the axes	?dim
GetGear	Inquiry- gear transmission	?gear
GetJoystickFilter	Indicates, if the filtering and hysteresis is activated in joystick operation	?joyfilter
GetMotorCurrent	Inquiry motor current	?cur
GetMotorTablePatch	Indicates, if the correction table is activated.	?mtpatch
GetOutFuncLev	Indicates the speed when the current will be switched, from parameterised current to maximum current.	?opfl
GetPitch	delivers spindle pitch	?pitch
GetPowerAmplifier	Indicates if the amplifiers of the LS44 are switched ON or OFF. This command only exists for the LS44-controller.	?pa
GetReduction	Inquiry of current reduction	?reduction
GetRefSpeed	Reads the reverse speed, the axes move while searching the reference mark.	?calrefspeed
GetRMOffset	Inquiry RM-Offset	?rmoffset
GetSpeedPoti	Indicates if the potentiometer On/Off	?pot
GetStopAccel	Delivers the brake acceleration, if the stop input becomes active.	?stopaccel
GetStopPolarity	Read stop entrance polarity.	?stoppol
GetVel	Inquiry speed	?vel
GetVelFac	Inquiry speed reduction	?velfac
GetVLevel	Delivers the speed limits of the indicated speed range.	?vlevel
GetXYAxisComp	Inquiry XY-axis overlay	?xycomp
LstepSave	Save current configuration in LStep (EEPROM)	save
SetAccel	Set acceleration	!accel
SetAccelSingleAxis	Set acceleration for individual axis	!accel x (y,z,a)
SetActiveAxes	Enable axes	!axis
SetAxisDirection	Reverse turning direction	!axisdir
SetCalibBackSpeed	Sets the speed, with which the axis are moved back during calibration after reaching the limit switches.	!calbspeed
SetCaliboffset	Calibration offset	!caliboffset
SetCalibrateDir	Reverse preceding sign when calibrating	!caldir
SetCurrentDelay	Time delay for current reduction	!curdelay
SetDimensions	Set dimensions of the axes	!dim
SetGear	Program gear transmission	!gear
SetJoystickFilter	Activating/Deactivating the filtering and hysteresis in joystick operation	!joyfilter
SetMotorCurrent	Set motor current	!cur
SetMotorTablePatch	Correction table ON/OFF	!mtpatch
SetOutFuncLev	Set the current switch speed	!opfl
SetPitch	Set spindle pitch	!pitch

SetPowerAmplifier	Switches the amplifiers of the LS44 On/Off.	!pa
SetReduction	Set current reduction	!reduction
SetRefSpeed	Sets the reverse speed, the axes move while searching the reference mark.	!calrefspeed
SetRMOffset	RM-Offset	!rmoffset
SetSpeedPoti	Potentiometer On/ Off	!pot
SetStopAccel	Delivers the brake acceleration, if the stop input becomes active.	!stopaccel
SetStopPolarity	Adjust stop entrance polarity.	!stoppol
SetVel	Set speed (velocity)	!vel
SetVelFac	Set speed reduction	!velfac
SetVelSingleAxis	Set speed for individual axis	!vel x (y,z,a)
SetVLevel	Exclude speed ranges, in which the system shows resonances.	!vlevel
SetXYAxisComp	Activate XY-axis overlay	!xycomp
SoftwareReset	Reset the software to starting status	reset

Status report

API-Command	Short description	LSTEP-Command
GetError	gives the current error number	?err
GetSecurityErr	Reads all statuses and results of the GAL-safety monitoring (only with LS44-controller)	?securityerror
GetSecurityStatus	Delivers the current statuses the safety monitoring (only with LS44-controller)	?securitystatus
GetStatus	Gives the current status of the controller	?status
GetStatusAxis	Gives the present status of the individual axes	?statusaxis
GetStatusLimit	Delivers the current condition of the software-limits of each axis.	?statuslimit
SetAutoStatus	AutoStatus On/Off	!autostatus

Moving commands and Position administration

API-Command	Short description	LSTEP-Command
Calibrate	Calibrate	!cal
CalibrateEx	Only the axes are calibrated, whose corresponding bit was set in the transmitted integer-value.	!cal x (xy,z,a)
Clearpos	Sets the position to 0 (for endless turning axes)	!clearpos
GetDelay	Reads the delay of the vector start.	?delay
GetDistance	Delivers the distance for LS_MoveRelShort	?distance
GetPos	Inquires the current positions of all axes	?pos
GetPosEx	Inquires the current encoder or positional values of all axes	
GetPosSingleAxis	Inquire the current position of an axis	?pos x (y,z,a)
MoveAbs	Move to absolute position	!moa
MoveAbsSingleAxis	Move individual axis to absolute position	!moa x (y,z,a)
MoveEx	extended moving command	
MoveRel	Move to relative vector	!mor
MoveRelShort	Move to relative position (short command)	m
MoveRelSingleAxis	Move individual axis relatively	!mor x (y,z,a)
RMeasure	Measure table stroke	!rm
RmeasureEx	Measure table stroke (The table stroke is only measured for axes for which the relevant bit has been set in the transmitted integer value).	!rm x (xy z)

SetDelay	The delay command is used to produce a vector start delay	!delay
SetDistance	Set distance (for LS_MoveRelShort)	!distance
SetPos	Set positional values	!pos
StopAxes	Stop (all movements are stopped)	a
WaitForAxisStop	The function returns, as soon as the selected axes in the bit-mask A Flag reached its goal position.	-

Joystick and Handwheel

API-Command	Short description	LSTEP-Command
GetDigJoySpeed	Read out the set speeds	?speed
GetHandwheel	Read hand wheel condition	?hw
GetJoystick	Reads the delay of the vector start.	?joy
GetJoystickDir	Reads motor turning direction for joystick	?joydir
GetJoystickWindow	Read Joystick-window	?joywindow
SetDigJoySpeed	Read Digital joystick and speed .	!speed
SetHandwheelOff	Handwheel Off	!hw 0
SetHandwheelOn	Handwheel On	!hw 1 (1-4)
SetJoystickDir	Joystick direction	!joydir
SetJoystickOff	Analogue joystick Off	!joy 0
SetJoystickOn	Analogue joystick On	!joy 1 (1-4)
SetJoystickWindow	set joystick-window	!joywindow
GetJoyChangeAxis	Read Joystick allocation of the axes	?joychangeaxis
JoyChangeAxis	sets allocation of axes of Joystick	!joychangeaxis

Control panel with Trackball and Joyspeed-keys

API-Command	Short description	LSTEP-Command
GetBPZ	Reads the condition of the additional control panel with track ball	?bpz
GetBPZJoyspeed	Control panel joystick-speed	?joyspeed
GetBPZTrackballBackLash	Read out control panel track ball-back lash	?bpzbl
GetBPZTrackballFactor	Read ot control panel trackball-factor	?bpztf
SetBPZ	Control panel On/ Off	!bpz
SetBPZJoyspeed	Control panel joystick-speed	!joyspeed
SetBPZTrackballBackLash	Control panel trackball-reverse backlash	!bpzbl
SetBPZTrackballFactor	Control panel trackball-factor	!bpztf

Limit switch (Hardware a. Software)

API-Command	Short description	LSTEP-Command
GetAutoLimitAfterCalibRM	Indicates if the internal software limits will be set during calibration and table stroke measuring.	?nosetlimit
GetLimit	Set travel limits	?lim
GetLimitControl	Reads, if travel range monitoring is active	?limctr
GetSwitchActive	Read status of limit switch	?swact
GetSwitches	Reads the status of all limit switches	?readsw
GetSwitchPolarity	Reads limit switch polarity	?swpol
SetAutoLimitAfterCalibRM	Prevents that the internal software limits are set during calibration and table stroke measuring.	!nosetlimit
SetLimit	Set travel limits	!lim
SetLimitControl	Control/ monitoring of the range of travel	!limctr

SetSwitchActive	Limit switch On/ Off	!swact
SetSwitchPolarity	Set limit switch polarity	!swpol

Digital and analogue In.- and Outputs

API-Command	Short description	LSTEP-Command
GetAnalogInput	Reads the current status of an analogue channel	?anain
GetAnalogInputs2	Reads the current status of the analogue channels PT100, MV and V24	--
GetDigitalInputs	Read all input pins	?digin
GetDigitalInputsE	Read additional digital inputs (16-31)	?edigin
SetAnalogOutput	Set analogue output	!anaout
SetDigIO_Distance	Function of the digital inputs / outputs	(digfkt)
SetDigIO_EmergencyStop	Function of the digital inputs / outputs Allocating of the Emergency Stop pin	(digfkt)
SetDigIO_Off	"Off" function of the digital inputs/outputs	(digfkt)
SetDigIO_Polarity	Set polarity	!digfkt 16 0 0
SetDigitalOutput	Set output pin	!digout x
SetDigitalOutputs	Set digital outputs (0-15)	!digout 0-15
SetDigitalOutputsE	Set additional outputs (16-31)	!edigout

Clock pulse Forward / Back

API-Command	Short description	LSTEP-Command
GetFactorTVR	Reads factor for clock pulse Forward/ Back	?tvrf
GetTVRMode	Read setup of clock pulse Forward /Back (= TVR Mode)	?tvr
SetFactorTVR	Factor for clock pulse Forward/ Back	!tvrf
SetTVRMode	Set clock pulse Forward / Back (=TVR Mode)	!tvr (0-4)

Clock pulse Forward / Back for the additional axes.

API-Command	Short description	LSTEP-Command
SetTVRInPulse	Clock pulse Forward /Back via Interface	px/nx

Clock pulse Forward / Back for the additional axes.

API-Command	Short description	LSTEP-Command
GetAccelTVRO	Reads the set acceleration for the additional axes.	?tvroa
GetPosTVRO	Read position of the additional axis	?tvropos
GetStatusTVRO	Delivers the current status of the additional axis	?tvrostatus
GetTVROOutMode	Read settings of the additional axis	?tvROUT
GetTVROOutPitch	Reads the spindle pitch of the additional axis	?tvropitch
GetTVROOutResolution	Reads the resolution of the amplifier which is to be controlled	?tvrores
GetVelTVRO	Reads the set speed of the additional axis	?tvrov
MoveAbsTVROSingleAxis	Position single axis absolute	!tvromoa x
MoveAbsTVRO	Move to absolute position	!tvromoa
MoveRelTVROSingleAxis	Move single axis absolute	!tvromor x
MoveRelTVRO	Move relative vector	!tvromor
SetAccelSingleAxisTVRO	Acceleration of single additional axis	!tvroa x
SetAccelTVRO	Set acceleration	!tvroa
SetPosTVRO	Set position of the additional axis	!tvropos
SetTVROOutMode	Set additional axis X, Y, Z and A, beside the actual main axis X, Y, Z and A	!tvROUT
SetTVROOutPitch	Sets the spindle pitch for the additional axis	!tvropitch
SetTVROOutResolution	Sets the resolution of the amplifier which is to be controlled	!tvrores
SetVelSingleAxisTVRO	set speed of the additional axis	!tvrov x
SetVelTVRO	set speed of the additional axis	!tvrov

Encoder-Settings

API-Command	Short description	LSTEP-Command
ClearEncoder	Set encoder-counter to zero	!pos 0 0 0 0
GetEncoder	Reads all encoder positions	!encpos1 ?pos
GetEncoderActive	Reads , which encoders are activated after the calibration.	?encmask
GetEncoderMask	Read encoder statuses	?enc
GetEncoderPeriod	Read length of encoder period	?encperiod
GetEncoderPosition	Read encoder position setting	
GetEncoderRefSignal	Reads if interpret reference signal from encoder when calibration is done	?encref
SetEncoderActive	This function is used to select which encoder is to be activated after calibration.	!encmask
SetEncoderPeriod	Set length of encoder period	!encperiod
SetEncoderPosition	Encoder position display On/Off	!encpos 1 !pos 0 0 0
SetEncoderRefSignal	Interpret reference signal from encoder when calibration is done	!encref

Controller Setting

API-Command	Short description	LSTEP-Command
ClearCtrFastMoveCounter	This function sets Fast Move Counters of all axis to zero.	!ctrfmc 0
GetController	Read controller mode	?ctr
GetControllerCall	Reads controller call time	?ctrc
GetControllerFactor	Reads controller factor	?ctrf
GetControllerSteps	Reads controller steps length	?ctrs
GetControllerTimeout	Reads controller timeout	?crtt
GetControllerTWDelay	Read controller relay	?ctrd
GetCtrFastMove	Reads setting of the Fast Move Function	?ctrfm
GetCtrFastMoveCounter	Read amount of executed FastMove functions to 0	?ctrfmc
GetTargetWindow	Reads the target window	?twi
SetController	Set controller mode	!ctr
SetControllerCall	Call controller	!ctrc
SetControllerFactor	Controller factor	!ctrf
SetControllerSteps	Controller steps	!ctrs
SetControllerTimeout	Controller timeout	!crtt
SetControllerTWDelay	Controller delay	!ctrd
SetCtrFastMoveOff	Fast Move Funktion „OFF“	!ctrfm 0
SetCtrFastMoveOn	Fast Move Funktion „ON“	!ctrfm 1
SetTargetWindow	Target window	!twi

Trigger-Output

API-Command	Short description	LSTEP-Command
GetTrigCount	Read Trigger counter.	?trigcount
GetTrigger	Read Trigger setting Einstellung vom Trigger auslesen	?trig
GetTriggerPar	Reads Trigger-Parameter	?triga ?trigm ?trigs ?trigd
SetTrigCount	Read Trigger counter	!trigcount
SetTrigger	Trigger On/ Off	!trig
SetTriggerPar	Trigger parameters	!triga !trigm !trigs !trigd

Snapshot-Input

API-Command	Short description	LSTEP-Command
GetSnapshot	Reads the current Snapshot-condition	?sns
GetSnapshotCount	Snapshot counter	?snscl
GetSnapshotFilter	Reads input filter (snapshot-filter)	?snsf
GetSnapshotPar	Read Snapshot-Parameter	?snsl ?snsm ?sns
GetSnapshotPos	Read snapshot position	?snspl
GetSnapshotPosArray	Read snapshot-position from array	?snspla
SetSnapshot	Snapshot On/Off	!sns
SetSnapshotFilter	Set input filter for rebounding switches.	!snsf
SetSnapshotPar	Snapshot parameters	!snsl !snsm !sns