

# Precision Positioning Systems

**LSTEP / ECO-STEP**



# Contents

---

	Page
<b>Foreword</b>	
<b>1 Safety Instructions</b> .....	1 • 1
1.1 General Instructions.....	1 • 1
1.2 Initial Start-Up Information.....	1 • 2
<b>2 Functional Description</b> .....	2 • 1
2.1 RS232 Interface.....	2 • 1
2.1.1 Operation Without Control Computer.....	2 • 1
2.2 Controls.....	2 • 2
<b>3 Initial Start-Up</b> .....	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
<b>4 The LSTEP Controller Instruction Set</b> .....	4 • 1
4.1 Firmware and Hardware Information.....	4 • 2
4.2 Reset.....	4 • 3
4.3 Interface Configuration.....	4 • 4
4.4 Instruction Set Used.....	4 • 5
4.5 Status and Fault / Error Messages.....	4 • 6
4.6 Settings.....	4 • 8
4.7 Determination Of The Mechanical Work Range.....	4 • 17
4.8 Travel Instructions And Their Check Functions.....	4 • 19
4.9 Joystick and Handwheel (Manual Encoder) Instructions.....	4 • 23
4.10 Inputs/Outputs.....	4 • 26
4.11 Interpretation Of Incremental Measuring Systems.....	4 • 29
4.12 Controller Settings For LSTEP.....	4 • 32
4.13 Special Instructions For The Märzhäuser MR-System.....	4 • 37
4.14 Interpretation Of Clock Pulse And Direction Of Rotation Specifications.....	4 • 40
4.14.1 Range Of Travel Monitoring.....	4 • 40
4.14.2 Temporal Marginal Conditions For The Signals.....	4 • 40
4.15 Configuration Of The Trigger Output Signal.....	4 • 43
4.16 Configuration Of The Snapshot Input.....	4 • 46

	Page
<b>5 Appendix General</b> .....	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 ( <i>Not for ECO-STEP</i> ).....	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
<b>6 Appendix LSTEP</b> .....	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 ZSS 42.200.1,2.....	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
<b>7 Appendix ECO-STEP</b> .....	7 • 1
7.1 Back Panel Of The ECO-STEP.....	7 • 1
7.2 Motor Connection X/Y/Z.....	7 • 1
7.3 Voltage Connection.....	7 • 2
7.4 DIP Switch Settings.....	7 • 2
7.5 Technical Data.....	7 • 3
7.6 View Of The Circuit Boards.....	7 • 4
7.7 The Powerpack.....	7 • 5
7.7.1 Technical Data For The Power Pack.....	7 • 5

	Page
<b>8 Appendix LSTEP-API</b> .....	<b>8 • 1</b>
8.1 Introduction .....	8 • 1
8.1.1 Included Functions .....	8 • 1
8.1.2 System Requirements .....	8 • 1
8.1.3 Supported Development Environments .....	8 • 1
8.2 DLL Interface .....	8 • 2
8.2.1 General Information .....	8 • 2
8.2.2 Integration in Delphi .....	8 • 2
8.2.3 Integration in Visual C++ .....	8 • 2
8.2.4 Integration In LabVIEW .....	8 • 3
8.3 Error Codes .....	8 • 6
8.3.1 LSTEP-Error/Fault Messages .....	8 • 6
8.3.2 API-Error/Fault Messages .....	8 • 6
8.4 Functions .....	8 • 7

**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- $\mu\text{m}$  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 Win-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. (30 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
<b>CLEAR X/Y/Z (optional)</b>	Switch for resetting the display (separate for X- Y-and Z- position). The position register is also deleted.
<b>SPEED 1..10 (optional)</b>	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.
<b>JOYSTICK MAN / AUTO</b>	Joystick selector switch MAN = Manual mode (no “move” commands can be executed) AUTO = Automatic mode with the appropriate commands
<b>RESET (optional)</b>	When the reset switch is switched up, the controller is returned to starting status (just as if you had switched it off and on).
<b>ON</b>	Shows LSTEP is on and ready for operation
<b>LCD Display (optional)</b>	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

<b>Joystick Switch Set At "AUTO"</b>		
	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
<b>Joystick Switch Set At H</b>		
	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 (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  $470\text{W}/220\text{pF}$ ,  
4.7k $\Omega$  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<sub>ss</sub>.
2. magnetic linear transducer from the company Märzhäuser.

Due to the limited data capacity of microcontrollers, 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
<b>0.40 mm</b>	X	X	X	X	X	X	X
<b>0.50 mm</b>	X	X	X	X	X	X	X
<b>1.00 mm</b>	X	X	X	X	X	X	X
<b>2.00 mm</b>	X	X	X	X	X	X	X
<b>3.00 mm</b>							
<b>4.00 mm</b>	X	X	X	X	X	X	X
<b>5.00 mm</b>	X	X	X	X	X	X	X
<b>8.00 mm</b>	X	X	X	X	X	X	X
<b>10.00 mm</b>	X	X	X	X	X	X	X
<b>15.00 mm</b>							
<b>20.00 mm</b>	X	X	X	X	X	X	
<b>25.00 mm</b>	X	X	X	X	X	X	
<b>30.00 mm</b>							
<b>35.00 mm</b>							
<b>50.00 mm</b>	X	X	X	X	X	X	
<b>100.00 mm</b>	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 o.k., 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.
- Click on the **Update** box and confirm with "Yes". The old program is deleted from the controller.
- Select the file type ( IntelHexFile ).
- Load the new control program.
- ➔ The firmware is now transmitted to the controller.
- When programming is complete, return dipswitch "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.

## 4 The LSTEP Controller Instruction Set

---

For better clarity, all instructions and parameters which are sent to the controller and all acknowledgements/feedbacks 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 the commands are set by a customized program.

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 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 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)
	0x01 -- → Display configured
	0x02 -- → Speed poti configured
	0x04 -- → Handwheel (man. encoder) configured
	0x08 -- → Snapshot configured
	0x1 --- → TVR configured
	0x2 --- → Triggerout configured
	The appropriate combination of the information gives the present configuration.
Error code:	--
Example:	?det

Read Serial Number	
<b>Instruction:</b>	?readsn
<b>Parameters:</b>	none
<b>Description:</b>	Read out serial number of the controller.
<b>Feedback:</b>	8-characters
<b>Error code:</b>	--
<b>Example:</b>	?readsn

## 4.2 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 software reset

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

### 4.3 Interface Configuration

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

CTS Interpretation Of The RS232-Interface	
<b>Instruction:</b>	?cts or !cts
<b>Parameters:</b>	0 or 1
<b>Description:</b>	!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
<b>Feedback:</b>	0 or 1
<b>Error code:</b>	--
<b>Example:</b>	?cts (Display of the curren CTS interpretation status)

### 4.4 Instruction Set Used

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	
<b>Instruction:</b>	!ipreter or ?ipreter
<b>Parameters:</b>	0, 1 and 2
<b>Description:</b>	!ipreter 0 → Register oriented instruction set
	!ipreter 1 → New instruction set
	!ipreter 2 → Multi-control instruction set (Venus)
	?ipreter → Which instruction set ?
<b>Feedback:</b>	0, 1 or 2
<b>Error code:</b>	--
<b>Example:</b>	!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 ←

## 4.5 Status and Fault / Error Messages

AutoStatus	
<b>Instruction:</b>	!autostatus or ?autostatus
<b>Parameters:</b>	0,1, 2, 3 or 4
<b>Description:</b>	0 → No status is being transmitted by the controller.
	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.
<b>Feedback:</b>	
<b>Error code:</b>	--
<b>Example:</b>	!autostatus 1 ?autostatus

Status	
<b>Instruction:</b>	?status or status
<b>Parameters:</b>	--
<b>Description:</b>	gives the present status of the controller
<b>Feedback:</b>	Text
<b>Error code:</b>	--
<b>Example:</b>	?status

StatusAxis	
<b>Instruction:</b>	?statusaxis or statusaxis
<b>Parameters:</b>	--
<b>Description:</b>	gives the present status of the individual axes.
<b>Feedback:</b>	e.g.: @ - M -
	@ → Axis stopped
	M → Axis is moving (Motion)
	J → Joystick mode
	C → in control
	- → Axis is not enabled
<b>Error code:</b>	--
<b>Example:</b>	?statusaxis

Error	
<b>Instruction:</b>	?err or err
<b>Parameters:</b>	--
<b>Description:</b>	Error gives the present error number (see error description)
<b>Feedback:</b>	0 → No errors.
	1 → No valid axis designation.
	2 → No executable function.
	3 → Too many characters in the instruction string.
	4 → No valid instruction.
	5 → Outside of the valid range of values.
	6 → Incorrect number of parameters.
	7 → None "!" or "?"
	8 → TVR not possible because axis is active.
	9 → Axes cannot be switched on or off because TVR is active.
	10 → Function not configured.
	11 → Move instruction not possible because joystick is in manual mode.
	12 → Limit switch tripped.
<b>Feedback:</b>	
<b>Error code:</b>	--
<b>Example:</b>	?err

## 4.6 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	
<b>Instruction:</b>	!dim or ?dim
<b>Parameters:</b>	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
<b>Description:</b>	
	!dim 4 → The dimensions for the X- and Y-axes are “Number of revolutions” and “μm”.
	!dim z → The dimension for the Z-axis is “mm”.
	?dim → All dimensions are displayed.
	?dim a → The dimension of the A-axis is displayed.
<b>Feedback:</b>	Present setting
<b>Error code:</b>	
<b>Example:</b>	!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	
<b>Instruction:</b>	!pitch or ?pitch
<b>Parameters:</b>	X, Y, Z and A 0.001 – 100
<b>Description:</b>	!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.
<b>Feedback:</b>	Present spindle pitch
<b>Error code:</b>	--
<b>Example:</b>	!pitch 10 (spindle pitch X = 10mm) ?pitch

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

Acceleration	
<b>Instruction:</b>	!accel or ?accel
<b>Parameters:</b>	X, Y, Z and A 0.01 – 10.00 [m/s <sup>2</sup> ]
<b>Description:</b>	!accel 1.00 1.50 → The accelerations (X=1.00, Y = 1.50 [m/s <sup>2</sup> ]) 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 <sup>2</sup> ].
	?accel → All preset accelerations are displayed.
	?accel z → The preset acceleration of the Z-axis is displayed.
<b>Feedback:</b>	Preset acceleration
<b>Error code:</b>	--
<b>Example:</b>	!accel 1.00 (set accelerations for X-axis to 1 m/s <sup>2</sup> ) ?accel

Speed (Velocity)	
<b>Instruction:</b>	!vel or ?vel
<b>Parameters:</b>	X, Y, Z and A 0 – maximum speed
<b>Description:</b>	!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.
<b>Feedback:</b>	Preset speed
<b>Error code:</b>	--
<b>Example:</b>	!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	
<b>Instruction:</b>	!velfac ?velfac
<b>Parameters:</b>	X, Y, Z or A 0.01 to 1.00
<b>Description:</b>	!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
<b>Feedback:</b>	A decimal value is returned (0.01 to 1.00)
<b>Error code:</b>	--
<b>Example:</b>	?velfac z (gives the setting for the Z-axis)

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



Current Reduction	
<b>Instruction:</b>	!reduction or ?reduction
<b>Parameters:</b>	X, Y, Z and A 0 – 1.0
<b>Description:</b>	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.
<b>Feedback:</b>	Preset current reduction
<b>Error code:</b>	--
<b>Example:</b>	!reduction 0.3 0.5 (X-and Y-axes are reduced ) ?reduction

Axis Enable	
<b>Instruction:</b>	!axis or ?axis
<b>Parameters:</b>	X, Y,Z and A 0 and 1
<b>Description:</b>	!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
<b>Feedback:</b>	Present operating status
<b>Error code:</b>	--
<b>Example:</b>	!axis 1 1 1 1 (enable all axes) ?axis x (read status of X-axis)

Output Current	
<b>Instruction:</b>	!cur or ?cur
<b>Parameters:</b>	X, Y, Z and A 0 – maximum current
<b>Description:</b>	!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.
<b>Feedback:</b>	Preset output current
<b>Error code:</b>	--
<b>Example:</b>	!cur 1.0 (The X-axis is run at maximum 1A) ?cur

Limit	
<b>Instruction:</b>	!lim or ?lim
<b>Parameters:</b>	X, Y, Z or A +- maximum range of travel
<b>Note:</b>	The values must be specified pair-wise. The input and output values depend on the dimension.
<b>Description:</b>	!lim -1000 1000 -2000 2000 → Travel limits are allocated to the X- and Y- axes.
	!lim z -500 1700 → Allocate Z-axis travel limits.
	?lim → Read travel limits for all axes
	?lim a → Show travel limit for A-axis
<b>Feedback:</b>	Present Ranges Of Travel
<b>Error code:</b>	--
<b>Example:</b>	!lim 10 (only program X-axis bottom limit) ?lim

Limit Control	
<b>Instruction:</b>	!limctr or ?limctr
<b>Parameters:</b>	X, Y, Z or A 0 or 1
<b>Description:</b>	!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.
<b>Feedback:</b>	0 = Limit control not active 1 = Limit control active
<b>Error code:</b>	--
<b>Example:</b>	! limctr y 0 (Deactivate Y-axis limit control) ? limctr

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

Limit Switch On/Off	
<b>Instruction:</b>	!swact or ?swact
<b>Parameters:</b>	X, Y, Z or A 0 or 1
<b>Description:</b>	!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.
<b>Feedback:</b>	Status of the limit switches
<b>Error code:</b>	--
<b>Example:</b>	!swact y 1 1 1 (All Y-axis switches active) ?swact x

Read Limit Switches																											
<b>Instruction:</b>	?readsw																										
<b>Parameters:</b>																											
<b>Description:</b>	?readsw → Read status of all limit switches.																										
<b>Feedback:</b>	Status of the limit switches.																										
	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"><b>Axis:</b></td> <td style="width: 5%;">x</td> <td style="width: 5%;">y</td> <td style="width: 5%;">z</td> <td style="width: 5%;">a</td> <td style="width: 5%;">x</td> <td style="width: 5%;">y</td> <td style="width: 5%;">z</td> <td style="width: 5%;">a</td> <td style="width: 5%;">x</td> <td style="width: 5%;">y</td> <td style="width: 5%;">z</td> <td style="width: 5%;">a</td> </tr> <tr> <td><b>Switch:</b></td> <td>E0</td> <td>E0</td> <td>E0</td> <td>E0</td> <td>Ref</td> <td>Ref</td> <td>Ref</td> <td>Ref</td> <td>EE</td> <td>EE</td> <td>EE</td> <td>EE</td> </tr> </table>	<b>Axis:</b>	x	y	z	a	x	y	z	a	x	y	z	a	<b>Switch:</b>	E0	E0	E0	E0	Ref	Ref	Ref	Ref	EE	EE	EE	EE
	<b>Axis:</b>	x	y	z	a	x	y	z	a	x	y	z	a														
	<b>Switch:</b>	E0	E0	E0	E0	Ref	Ref	Ref	Ref	EE	EE	EE	EE														
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">E0 = Zero limit switch</td> <td style="width: 33%;">Ref = Reference limit switch</td> <td style="width: 33%;">EE = End limit switch</td> </tr> </table>	E0 = Zero limit switch	Ref = Reference limit switch	EE = End limit switch																								
E0 = Zero limit switch	Ref = Reference limit switch	EE = End limit switch																									
<b>Error code:</b>	--																										
<b>Example:</b>	?readsw (Read all limit switches)																										

## 4.7 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 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	
<b>Instruction:</b>	!cal or cal
<b>Parameters:</b>	X, Y, Z or A
<b>Description:</b>	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.
<b>Feedback:</b>	An ‘A’ for each calibrated axis
<b>Error code:</b>	--
<b>Example:</b>	!cal

Measure Table Stroke	
<b>Instruction:</b>	!rm or rm
<b>Parameters:</b>	X, Y, Z or A
<b>Description:</b>	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
<b>Feedback:</b>	A ,D’ for every axis
<b>Error code:</b>	--
<b>Example:</b>	!rm

RM Offset	
<b>Instruction:</b>	!rmoffset or ?rmoffset
<b>Parameters:</b>	X, Y, Z or A 0 – 32*50000 (32*spindle pitch)
<b>Description:</b>	!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.
<b>Feedback:</b>	Distance
<b>Error code:</b>	--
<b>Example:</b>	?rmoffset

Calibration Offset	
<b>Instruction:</b>	!caliboffset or ?caliboffset
<b>Parameters:</b>	X, Y, Z or A 0 – 32*50000 (32*spindle pitch)
<b>Description:</b>	!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
<b>Feedback:</b>	Distance
<b>Error code:</b>	--
<b>Example:</b>	?caliboffset

## 4.8 Travel Instructions And Their Check Functions

Linear interpolation takes place for all positioning instructions, i.e. all axes reach the specified position at the same time. The axis which travels the furthest is deemed the lead axis and thus travels at the preset speed.

Position absolut	
<b>Instruction:</b>	!moa or moa
<b>Parameters:</b>	X, Y, Z or A +- Range of travel
<b>Note:</b>	The input depends on the dimension.
<b>Description:</b>	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.
<b>Feedback:</b>	A ,@' for every positioned axis
<b>Error code:</b>	--
<b>Example:</b>	Moa x 10 (The X-axis is positioned at the position which was input)

Relative Position	
<b>Instruction:</b>	!mor or mor
<b>Parameters:</b>	X, Y, Z or A +- Range of travel
<b>Note:</b>	The input depends on the dimension.
<b>Description:</b>	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.
<b>Feedback:</b>	A,@' for every axis which is travelled
<b>Error code:</b>	--
<b>Example:</b>	!mor 0 0 0 100 (Only the A-axis is travelled)

XY-Compensation	
<b>Instruction:</b>	?xycomp or !xycomp
<b>Parameters:</b>	0, 1, 2, 3 or 4
<b>Description:</b>	!xycomp 1    X 10 mm = X 10 mm Y 10 mm = X 10 mm + Y 10 mm !xycomp 2    Y 10 mm = Y 10 mm X 10 mm = X 10 mm + Y 10 mm !xycomp 3    X 10 mm = X 10 mm + Y 10 mm Y 10 mm = Y 10 mm + X -10 mm !xycomp 4    Y 10 mm = Y 10 mm + X 10 mm X 10 mm = X 10 mm + Y -10 mm
<b>Feedback:</b>	Current status of the XY- compensation
<b>Error code:</b>	--
<b>Example:</b>	?xycomp (gives the current status of the XY-compensation)

Relative Position (Shortcut)	
<b>Instruction:</b>	!m or m
<b>Parameters:</b>	
<b>Note:</b>	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 mor instructions.
<b>Description:</b>	m → Start travel of all enabled axes.
<b>Feedback:</b>	Depends on the autostatus setting.
<b>Error code:</b>	--
<b>Example:</b>	!mor 0 0 0 100 (Only the A-axis is travelled) m (A-axis is travelled by 100 again)

Distance	
<b>Instruction:</b>	!distance or ?distance
<b>Parameters:</b>	X,Y,Z and A Min-/max- travel range
<b>Note:</b>	Input and output depend on the dimension.
<b>Description:</b>	!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.
<b>Feedback:</b>	Distances
<b>Error code:</b>	--
<b>Example:</b>	!distance 10 20 (Set X- and Y-axis distances) ?distance (Inquire distances of all axes)

SpeedPoti	
<b>Instruction:</b>	!pot or ?pot
<b>Parameters:</b>	0 or 1
<b>Note:</b>	
<b>Description:</b>	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.
<b>Feedback:</b>	--
<b>Error code:</b>	--
<b>Example:</b>	!pot1 ?pot

Position	
<b>Instruction:</b>	!pos or ?pos
<b>Parameters:</b>	X,Y,Z and A Min-/max range of travel
<b>Note:</b>	Input and output depend on the dimension.
<b>Description:</b>	!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.
<b>Feedback:</b>	Positional values
<b>Error code:</b>	--
<b>Example:</b>	!pos100 200 (Set positions of X- and Y-axes) ?pos (Inquire positions of all axes)

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

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

## 4.9 Joystick and Handwheel (Manual Encoder) Instructions

**Note:** 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.

Instructions for setting the controller are possible in this mode of operation, “Move” instructions however are not.

If an inquiry is made with the instruction “Statusaxis”, the controller gives the reply “JJJ”.

Digital Joystick (Speed)	
<b>Instruction:</b>	!speed or ?speed
<b>Parameters:</b>	X, Y, Z or A +- Maximum speed (vel)
<b>Description:</b>	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 → Read the preset speeds.
	?speed y → Read the preset speed of the Y-axis.
<b>Feedback:</b>	Present speeds
<b>Error code:</b>	--
<b>Example:</b>	!speed 33 11 (X-axis speed 33.0 [r/s], Y-axis speed 11.0 [r/s] and joystick mode “ON”) ?speed

Joystick Direction	
<b>Instruction:</b>	!joydir or ?joydir
<b>Parameters:</b>	0, ±1, ±2 X, Y, Z, and A
<b>Note:</b>	As only a 3-axis joystick is provided, the 3 <sup>rd</sup> joystick axis acts on the Z- and the A-axis.
<b>Description:</b>	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 z 0 → Z-axis is disabled.
	!joydir 2 2 -1 → Für X + Y positive Drehrichtung und Stromreduzierung im Stand. Für Z negative Drehrichtung ohne Stromreduzierung im Stand.
<b>Feedback:</b>	Reset directions or status.
<b>Error code:</b>	--
<b>Example:</b>	!joydir-1 (negative preceding sign for X-axis) ?joydir

Joystick	
<b>Instruction:</b>	!joy or ?joy
<b>Parameters:</b>	0, 1, 2, 3, 4 and 5
<b>Note:</b>	The joystick switch must be set at Automatic
<b>Description:</b>	!joy0 → Joystick "OFF"
	!joy1 → Joystick "ON" without position count (tracking)
	!joy2 → Joystick "ON" with position count
	!joy3 → Joystick "ON" with position count and periodic position feedback.
	!joy4 → Joystick "ON" with position count (encoder values, if any)
	!joy5 → Joystick "ON" with position count and periodic position feedback (encoder values, if any).
	?joy → Present status
<b>Feedback:</b>	Present position or present status of joystick operation
<b>Error code:</b>	--
<b>Example:</b>	!joy 2 (Joystick "ON" with position count) ?joy (Inquire present status)

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

Handwheel	
<b>Instruction:</b>	!hw or ?hw
<b>Parameters:</b>	0, 1, 2, 3, 4 and 5
<b>Note:</b>	As an alternative to the joystick, a handwheel may be installed.
<b>Description:</b>	!hw0 → Handwheel "OFF" !hw1 → Handwheel "ON" without position count !hw2 → Handwheel "ON" with position count !hw3 → Handwheel "ON" with position count and periodic position feedback. !hw4 → Handwheel "ON" with position count (encoder values, if any). !hw5 → Handwheel "ON" with position count and periodic position feedback (encoder values, if any). ?hw → Current status
<b>Feedback:</b>	Present position or current status of the handwheel operation.
<b>Error code:</b>	--
<b>Example:</b>	!hw 2 (Handwheel "ON" with position count) ?hw (Inquire present status)

### 4.10 Inputs/Outputs *(not for ECO-STEP)*

The LSTEP may be equipped with an additional module which makes 16 digital inputs and outputs and two analog outputs available. To use these inputs and outputs, you must order the appropriate LSTEP model.

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

Digital Output	
<b>Instruction:</b>	!digout or ?digout
<b>Parameters:</b>	0 through 15
<b>Description:</b>	!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
<b>Feedback:</b>	Status of the output pins
<b>Error code:</b>	--
<b>Example:</b>	!digout 7 0 (Set output pin 7 to "0") ?digout (Read all output pins)

Function Of The Digital Inputs/Outputs	
<b>Instruction:</b>	!digfkt or ?digfkt
<b>Parameters:</b>	0 through 15 (input/output), 16 (all 16 port pins) 0 through 4 (function) 0 through 100 mm (distance) X, Y, Z and A (axes)
<b>Description:</b>	<b>Functions:</b>
	0 → No influence of the inputs/outputs and setting of the polarity (0 = High-, 1 = Low-Active).
	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
	<b>Instructions:</b>
	!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 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.	
<b>Feedback:</b>	All settings
<b>Error code:</b>	--
<b>Example:</b>	!digfkt 7 0 (Set the function of input and output 7 to "0") ?digfkt 9 (Read the function of input and output 9)

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	
<b>Instruction:</b>	!anaout or ?anaout
<b>Parameters:</b>	0 through 100 % 0, 1 and 2 (analog channels) c (c = channel)
<b>Note:</b>	Channels 0 and 1 are on the additional board (D/A-converter) Channel 2 is on the LS2000 board.
<b>Description:</b>	!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 → Rea the current status of analog channel 2.
<b>Feedback:</b>	Status of the modulation in percent of the analog channels.
<b>Error code:</b>	--
<b>Example:</b>	!anaout c 1 0 (Set analog channel 1 to "0") ?anaout (Read all analog channels)

### 4.1.1 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.

Encoder Mask For Encoders	
<b>Instruction:</b>	!encmask or ?encmask
<b>Parameters:</b>	X, Y, Z and A 0,1 (On,Off)
<b>Note:</b>	Enabling of the individual encoders.
<b>Description:</b>	!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.
<b>Feedback:</b>	Encoder mask
<b>Error code:</b>	--
<b>Example:</b>	!encmask 1 0 (X-axis encoder enabled, Y-axis encoder not enabled) ?encmask

Encoder Mask For Detected Encoders	
<b>Instruction:</b>	!enc or ?enc
<b>Parameters:</b>	X, Y, Z and A 0,1 (On,Off)
<b>Note:</b>	If encoders are activated which are not available, malfunctions may occur.
<b>Description:</b>	!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.
<b>Feedback:</b>	Encoder status
<b>Error code:</b>	--
<b>Example:</b>	!enc 1 0 (X-axis encoder active, Y-axis encoder deactivated) ?enc

Signal Periods	
<b>Instruction:</b>	!encperiod or ?encperiod
<b>Parameters:</b>	X, Y, Z and A 0.0001 – Spindle pitch * 0.8 (mm)
<b>Description:</b>	!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.
<b>Feedback:</b>	Length of encoder period in mm
<b>Error code:</b>	--
<b>Example:</b>	!encperiod 0.1 (Length of encoder period for the X-axis is 0.1mm) ?encperiod

Encoder Reference Signal	
<b>Instruction:</b>	!encref or ?encref
<b>Parameters:</b>	X, Y, Z or A 0 or 1
<b>Description:</b>	!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.
<b>Feedback:</b>	0 or 1
<b>Error code:</b>	--
<b>Example:</b>	!encref 0 (No reference signal interpretation for the X-axis) ?encref

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

Encoder Error	
<b>Instruction:</b>	!encerr or ?encerr
<b>Parameters:</b>	X, Y, Z, A 0
<b>Description:</b>	!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.
<b>Feedback:</b>	0 or e
<b>Error code:</b>	--
<b>Example:</b>	!encerr 0 (Clear encoder error message from A-axis) ?encerr

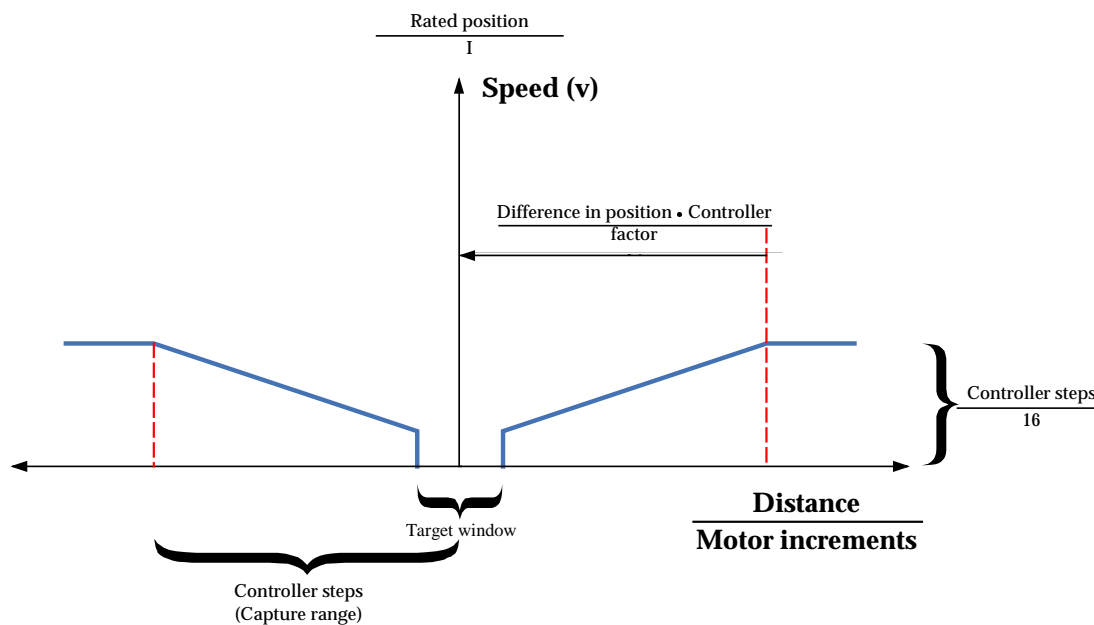
## 4.12 Controller Settings For LSTEP (not for ECO-STEP)

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. 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	
<b>Instruction:</b>	!twi or ?twi
<b>Parameters:</b>	X, Y, Z and A 1 to 25000 (motor increments) 0.1 to spindle pitch/2 (µm) 0.0001 to spindle pitch/2 (mm)
<b>Note:</b>	The input and output values depend on the dimension.
<b>Description:</b>	!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.
<b>Feedback:</b>	Target window which has actually been set (rounding errors are displayed)
<b>Error code:</b>	--
<b>Example:</b>	!twi 10 (The X-axis has a target window of 10 motor increments (when Dim = 0)). ?twi

Controller	
<b>Instruction:</b>	!ctr or ?ctr
<b>Parameters:</b>	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.
<b>Description:</b>	!ctr y 2 → Y-axis controller "Always ON".
	?ctr → All controller statuses are displayed
	?ctr x → Display of the X-axis controller status
<b>Feedback:</b>	Controller statuses
<b>Error code:</b>	--
<b>Example:</b>	!ctr 0 0 0 0 (All controllers "OFF") ?ctr

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

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

Controller Steps	
<b>Instruction:</b>	!ctrs or ?ctrs
<b>Parameters:</b>	X, Y, Z and A 1 to spindle pitch
<b>Note:</b>	Input and output values depend on the dimension
<b>Description:</b>	!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.
<b>Feedback:</b>	Controller steps
<b>Error code:</b>	--
<b>Example:</b>	!ctrs 4 5 7 9 (Controller steps for all axes, dependent on the dimension) ?ctrs

Control Factor	
<b>Instruction:</b>	!ctrf or ?ctrf
<b>Parameters:</b>	X, Y, Z and A 1 – 64
<b>Description:</b>	!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.
<b>Feedback:</b>	Control factors
<b>Error code:</b>	--
<b>Example:</b>	!ctrf 1 2 3 4 (Set all control factors) ?ctrf

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

### 4.13 Special Instructions For The Märzhäuser MR-System

MROffset	
<b>Instruction:</b>	!mro or ?mro
<b>Parameters:</b>	X, Y, Z and A +- 2048
<b>Description:</b>	!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
<b>Feedback:</b>	Always sin cos for each axis
<b>Error code:</b>	--
<b>Example:</b>	!mro 0 0 0 0 (Set the offset values for the X-and Y-axes to 0 ) ?mro

Maximum Signal Values (Peaks)	
<b>Instruction:</b>	!mrp oder ?mrp
<b>Parameters:</b>	X, Y, Z and A +- 2048
<b>Description:</b>	!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.
<b>Feedback:</b>	Always pos. sin, neg. sin and pos. cos, neg. cos for each axis.
<b>Error code:</b>	--
<b>Example:</b>	!mrp 0 0 0 0 (Set peaks for the X-axis to 0) ?mrp

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

Amplification (Gain) Factor	
<b>Instruction:</b>	!mra or ?mra
<b>Parameters:</b>	X, Y, Z and A 0.01 – 2.00
<b>Note:</b>	The amplification or gain factor always refers to the cosine signal
<b>Description:</b>	!mra 1 1.01 0.98 → Amplification factors for $\cos x = 1$ , $\cos y = 1.01$ and $\cos z = 0.98$
	!mra z 1.23 → Amplification factor $\cos z = 1.23$
	?mra → Display of the amplification factors for all axes.
	?mra x → Display of the present amplification factor for the X-axis.
<b>Feedback:</b>	Amplification (gain) factor
<b>Error code:</b>	--
<b>Example:</b>	!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	
<b>Instruction:</b>	!mrs or ?mrs
<b>Parameters:</b>	X, Y, Z and A 0 or 1
<b>Note:</b>	0 = Sine and 1 = Cosine
<b>Description:</b>	!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
<b>Feedback:</b>	Signal identification (y 0: values ->)
<b>Error code:</b>	--
<b>Example:</b>	!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)

## 4.14 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.

### 4.14.1 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).

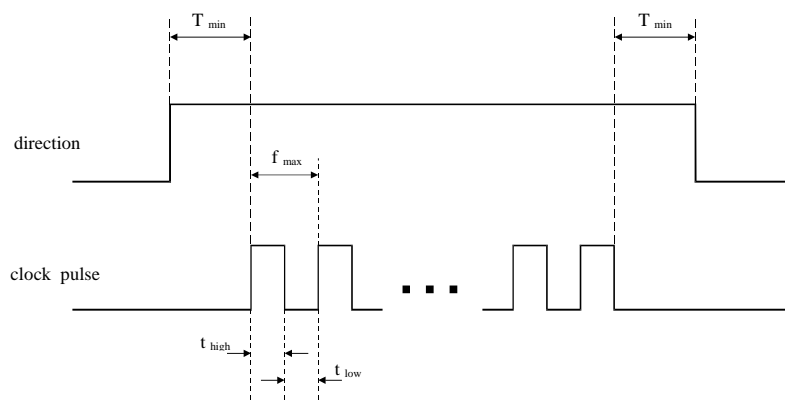
If the controller determines that the accumulated metering pulses would cause a travel limit to be exceeded, all further movement of the axis in that direction is inhibited. Travelling is however still possible in the opposite direction. No notification is given to the PC.

**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.

### 4.14.2 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  $f_{max} = 833$  kHz , whereby the minimum times  $T_{low} = 600$ ns and  $T_{high} = 600$ ns must be maintained.
- To protect the control inputs, input filters of 470 $\Omega$  and 220pF are used. You must therefore ensure that the clock pulse source has an adequate driver power.



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

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

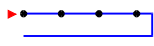

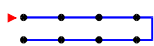



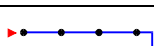

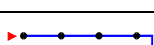

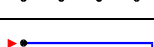

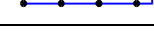

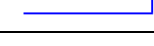

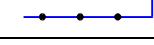



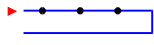

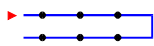

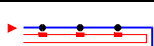





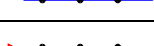

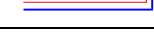

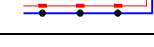

Clock Pulse Forward/Back (Internal Control)	
<b>Instruction:</b>	Px, nx, py, ny, pz, nz, pa, na
<b>Parameters:</b>	None
<b>Description:</b>	All instructions have the same effect as an external clock pulse with directional information. The first letter determines whether a positive (p) or a negative (n)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



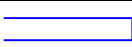


### 4.15 Configuration Of The Trigger Output Signal

These instructions synchronize an external piece of equipment, such as e.g. a video camera. This signal is output via the multi-function port, which is available as an option.

Trigger	
<b>Instruction:</b>	?trig or !trig
<b>Parameters:</b>	0 or 1 (OFF / ON)
<b>Description:</b>	!trig 1 → Trigger "ON"
	?trig → Gives the present status of the trigger processing.
<b>Feedback:</b>	ON or OFF
<b>Error code:</b>	--
<b>Example:</b>	!trig 0 (Trigger processing "OFF") ?trig

Trigger Axis	
<b>Instruction:</b>	?triga or !triga
<b>Parameters:</b>	X, Y, Z or A
<b>Description:</b>	!triga y → Trigger is referred to the Y-axis.
	?triga → Gives the current reference axis.
<b>Feedback:</b>	X, Y, Z or A
<b>Error code:</b>	--
<b>Example:</b>	!triga x (Trigger referred to the X-axis) ?triga

Trigger Mode		
<b>Instruction:</b>	?trigm or !trigm	
<b>Parameters:</b>	0 – 17	
<b>Description:</b>  The trigger modi 6-11 produce their first trigger pulse after half of the preset trigger distance and then Produce all subsequent pulses, depending on the - trigger distance .	!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
<b>Special function with external trigger signal. (Multi.function port pin 8) The external trigger signal must be “low” at the time the internal trigger signal is given.</b>	!trigm 12 → 	 high active
	!trigm 13 → 	 high active
	!trigm 14 → 	 high active
	!trigm 15 → 	 low active
	!trigm 16 → 	 low active
	!trigm 17 → 	 low active
<b>Feedback:</b>	0 – 17! (Mode)	
<b>Error code:</b>	--	
<b>Example:</b>	!trigm 3 (Trigger Mode 3) ?trigm	

Legend				
				
Starting point	Trigger points	Path	External trigger signal	 low active



Trigger Signal	
<b>Instruction:</b>	?trigs or !trigs
<b>Parameters:</b>	0 – 5 (µs) 0 = minimum trigger (some 100ns)
<b>Description:</b>	!trigs 4 → Trigger signal length 4 µs
	?trigs → Gives the present status of the set trigger signal length.
<b>Feedback:</b>	0 – 5 (µs)
<b>Error code:</b>	--
<b>Example:</b>	!trigs 3 (Trigger signal length = 3µs) ?trigs

Trigger Distance	
<b>Instruction:</b>	?trigd or !trigd
<b>Parameters:</b>	1 – 5000000 motor increments (depending on the Dim)
<b>Description:</b>	!trigd 1 → Trigger distance 1mm (for Dim 2)
	?trigd → Gives the present trigger distance.
<b>Feedback:</b>	Distance
<b>Error code:</b>	--
<b>Example:</b>	!trigd 3 (3mm trigger distance for Dim 2) ?trigd

## 4.16 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	
<b>Instruction:</b>	?sns or !sns
<b>Parameters:</b>	0 or 1
<b>Description:</b>	!sns 1 → Snapshot "ON"
	?sns → Gives the present snapshot status.
<b>Feedback:</b>	Snapshot status
<b>Error code:</b>	--
<b>Example:</b>	!sns 0 (Snapshot "OFF") ?sns

Snapshot-Level (Polarity)	
<b>Instruction:</b>	?snsl or !snsl
<b>Parameters:</b>	0 or 1
<b>Description:</b>	!snsl 1 → Snapshot is high-active. 
	?snsl → Gives the present polarity
<b>Feedback:</b>	Present polarity
<b>Error code:</b>	--
<b>Example:</b>	!snsl 0 (Snapshot is low-active)  ?snsl

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

Snapshot Counter	
<b>Instruction:</b>	?snsc
<b>Parameters:</b>	–
<b>Description:</b>	Contents are deleted after every “read”.
	?snsc → Gives the number of initiated snapshots.
<b>Feedback:</b>	Number of initiated snapshots
<b>Error code:</b>	--
<b>Example:</b>	?snsc

Snapshot Position	
<b>Instruction:</b>	!snsp or ?snsp
<b>Parameters:</b>	X,Y,Z and A Min./max. range of travel
<b>Note:</b>	Input and output depend on the dimension.
<b>Description:</b>	!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.
<b>Feedback:</b>	Positional values
<b>Error code:</b>	--
<b>Example:</b>	!snsp 100 200 (Set the X-and Z-axis positions) ?snsp (Inquire the snapshot positions of all axes)

Snapshot Position Array	
<b>Instruction:</b>	?snsa
<b>Parameters:</b>	X,Y,Z and A 0 – 200 (positions)
<b>Note:</b>	Input and output depend on the dimension.
<b>Note:</b>	
<b>Description:</b>	?snsa 33 → Inquire snapshot position 33 for all axes. ?snsa z 99 → Inquire snapshot position 99 for Z-axis.
<b>Feedback:</b>	Positional values
<b>Error code:</b>	--
<b>Example:</b>	?snsa 1 (Inquire snapshot positions 1 for all axes)

## 5 Appendix General

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

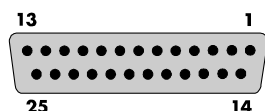


Fig.: The Multi-Function Port (25 Pole Sub-D Socket)

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.

Pin	Signal		Remarks
1	Clock input X	<b>Standard:</b>	TTL level
	Clock output X	<b>Special function:</b>	TTL level
	Encoder input X Track A	<b>Special function:</b>	+5V U-low $\leq 0.8V$ / U-High $\geq 3.6V$
2	X Forward /Back input	<b>Standard:</b>	TTL level
	X Forward/Back output	<b>Special function:</b>	TTL level
	Encoder input X Track B	<b>Special function:</b>	+5V U-low $\leq 0.8V$ / U-High $\geq 3.6V$
3	Clock input Y	<b>Standard:</b>	TTL level
	Clock output Y	<b>Special function:</b>	TTL level
	Encoder input Y Track A	<b>Special function:</b>	+5V U-low $\leq 0.8V$ / U-High $\geq 3.6V$
4	Y Forward/Back input	<b>Standard:</b>	TTL level
	Y Forward / Back output	<b>Special function:</b>	TTL level
	Encoder input Y Track B	<b>Special function:</b>	+5V U-low $\leq 0.8V$ / U-High $\geq 3.6V$
5	Clock input Z	<b>Standard:</b>	TTL level
	Clock output Z	<b>Special function:</b>	TTL level
	Encoder input Z Track A	<b>Special function:</b>	+5V U-low $\leq 0.8V$ / U-High $\geq 3.6V$
6	Analog input	<b>Special function:</b>	Measuring range 0...0.5V/ Ri = 1.1 k $\Omega$

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

## 5.2 RS232 Interface Pin Assignment

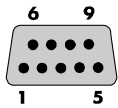


Fig.: The RS232 Interface (9 Pole, Sub-D Socket)

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

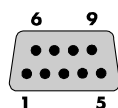


Fig.: The Joystick Connection (9 Pole, Sub-D Socket)

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	n.c.	
7	n.c.	
8	VAref (+5V)	5V analog reference voltage
9	VAref (+5V)	5V analog reference voltage

## 5.5 The CAN Interface (Not for ECO-STEP)

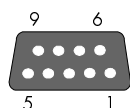


Fig.: The CAN Interface (9 Pole, Sub-D Socket)

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 .

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)

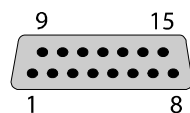


Fig.: The Handwheel Connection (15 Pole, Sub-D Socket)

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	setcloop	getcloop	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
<b>version</b>	Returns the version number of the Venus interpreter	Returns the version and revision number of the ITK interpreter.
<b>identify</b>	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
<b>setdim</b>	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
<b>status</b>	Returns the present satus of the MultiControl	Always returns 0 . <i>Note:</i> the status register of the LSTEP can be read out instead
<b>mode</b>	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
<b>save</b>	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 'Write error in flash memoryr' is set
<b>restore</b>	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 'Read error in flash memoryr' is sett
<b>setunit</b>	Sets the unit of an axis in physical terms	0 = motor increments 1 = $\mu\text{m}$ 2 = mm

Venus-Command	MultiControl Meaning	LSTEP-xx/2 Meaning
<b>setpitch</b>	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.
<b>move</b>	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.
<b>selftest</b>	Gives the result after the self-test for an axis for the controller	Always returns 0
<b>setjoysticktype</b>	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 compatibility 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> .
<b>getjoysticktype</b>	Returns the present joystick type	Returns the last value which was set with <i>setjoysticktype</i>
<b>setjoyspeed</b> <b>joyspeed</b>	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.
<b>getjoyspeed</b>	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
<b>setmotortype</b> <b>getmotortype</b>	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.
<b>setcurrent,</b> <b>getcurrent</b>	For service purposes only	Sets or reads the output current of the axes.
<b>v t setcalvel</b>	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.
<b>[r] setpos</b>	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.
<b>getpos</b>	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
<b>pos</b>	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 .
<b>devpos</b>	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> .
<b>m n l setsw</b>	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:

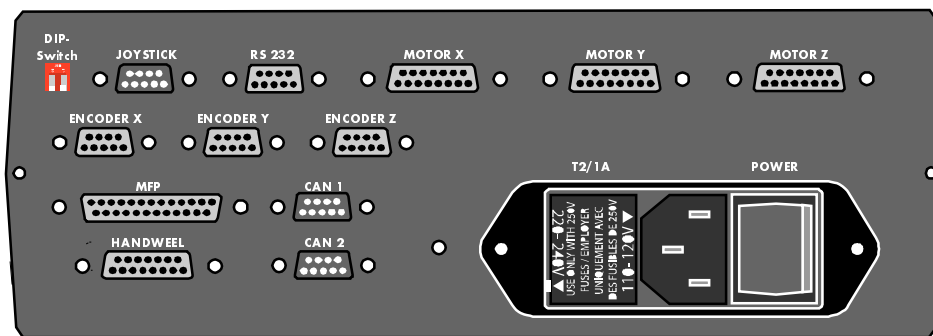
- Select low-resistance (low-impedance) motors with low inductances.
- Switch 8-conductor motors (e.g. ZSS 42.200.1,2 from Phytron) to low resistance.
- To avoid unnecessary heat errors, the motor current should however only be set as high as absolutely necessary.
- Motor currents which are at rated 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

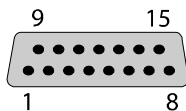


Fig.: Motor Connection (15 Pole, Sub-D Socket)

15 pole, D-SUB at LSTEP, Pin No.	Colour	12-pole Flanged Socket, Motor	Pin Assignment:
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)

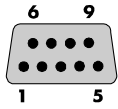


Fig.: Encoder Connection (9 Pole, Sub-D Socket)

PIN	Signal	PIN	Signal
1	U <sub>1-</sub>	6	U <sub>1+</sub>
2	0V	7	5V
3	U <sub>2-</sub>	8	U <sub>2+</sub>
4	+12V (Optional)	9	U <sub>0+</sub>
5	U <sub>0-</sub>	Housing	Outer screen

### 6.4 The Power Supply Module

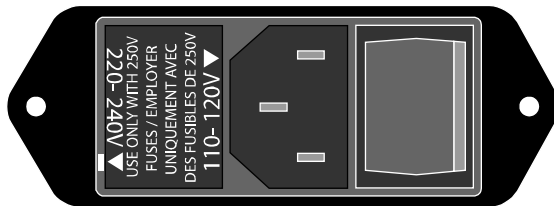


Fig.: The power supply module

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 . The side of the arrow of the relevant voltage applies in both cases.

### 6.5 DIP Switch Settings



Fig.: The LSTEP DIP Switches

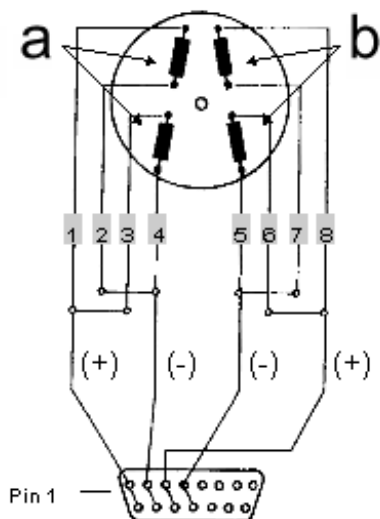
- Switch 1    ON    ➔    Firmware update switched on
- OFF   ➔    Firmware update switched off
- Switch 2    ON    ➔    Multicontrol instruction set
- OFF   ➔    Standard instruction set

## 6.6 Technical Data

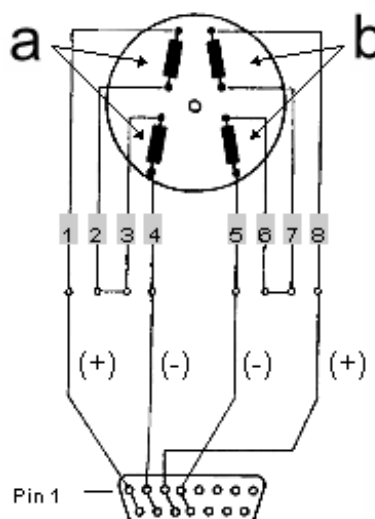
Power supply:	110V - 120V / 200V - 240V +/-10% 50/60Hz, 100VA
Fuses:	
- primary (in Euro socket):	<ul style="list-style-type: none"> <li>• 2 A time-lag / 1 A time-lag LSTEP-1 and LSTEP-2</li> <li>• 2.5 A time-lag / 1.25 A time-lag LSTEP-3</li> </ul>
- secondary (on the circuit board)	Fuse1 LSTEP-1 and 2 → 5 A time-lag / LSTEP-3 → 10A time-lag
Max. power failure duration:	< 50ms if a power failure occurs ( $<0.77 \cdot U_N$ ), the LSTEO switches to Reset
Max. motor speed:	40 r/sec. for a 200-step motor
Max. motor current:	1.25A per motor phase for LSTEP-1 2.5A per motor phase for LSTEP-2 5.0A per motorphase for LSTEP-3
Max. motor voltage:	40V
Step resolution	<ul style="list-style-type: none"> <li>• max. 50,000 (100,000) steps/revolution for a 200 step motor.</li> <li>• 2000 microsteps/full step for linear stepping motors</li> </ul>
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 ZSS 42.200.1,2

2-phase, low resistance motor



2-phase, high-resistive motor



15 pole plug or flanged plug

Motor Terminal Box			
1	redt	2	Brown
3	black	4	Yellow
5	blue	6	Violet
7	white	8	Green

## 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.



CAUTION:

**Pull out the mains plug before you open the unit!**

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

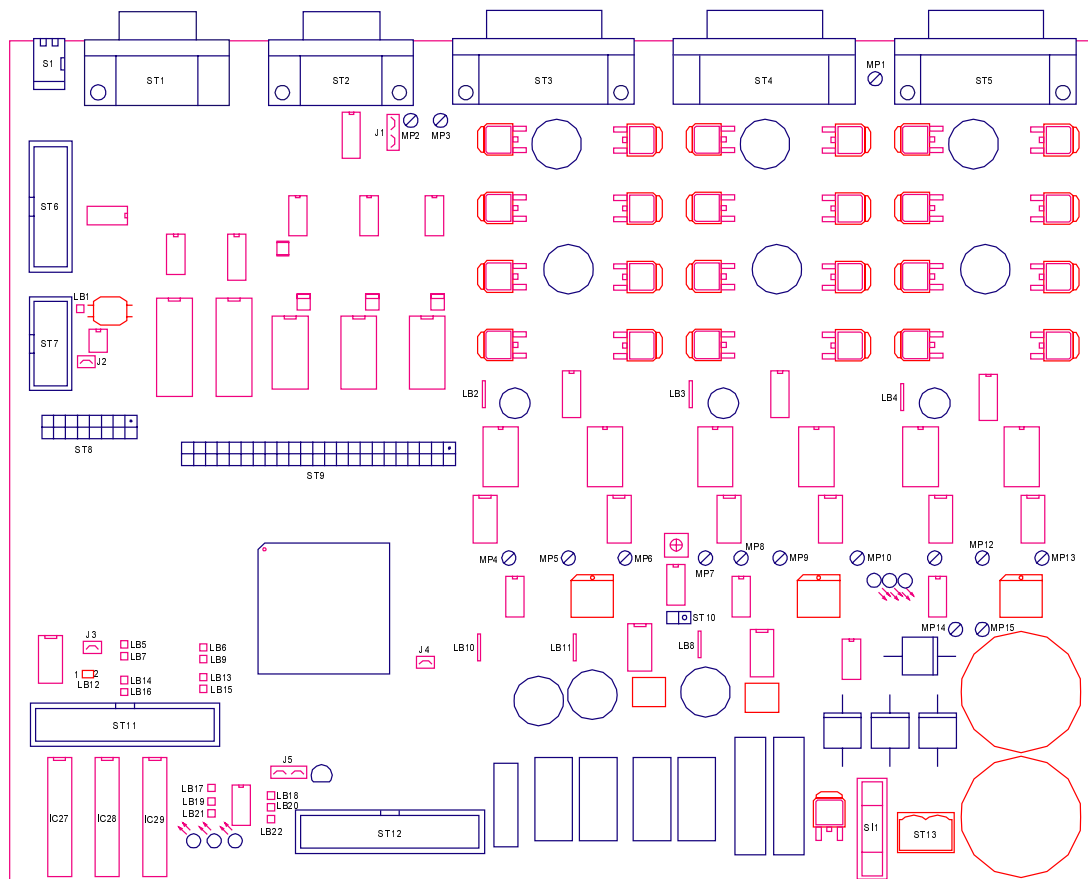


Fig.: The main circuit board

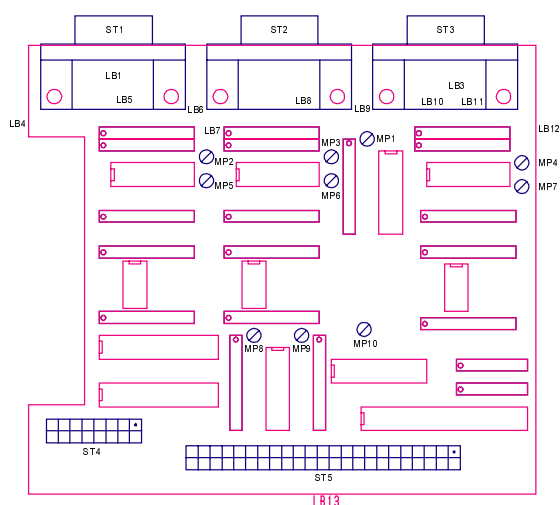
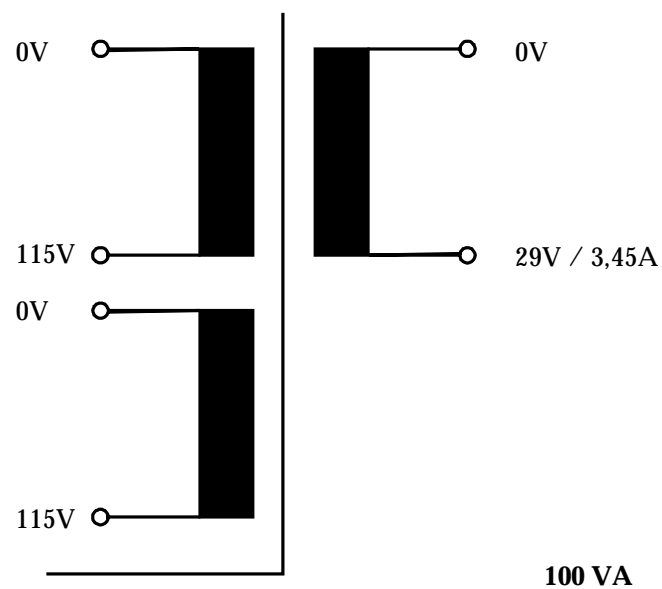


Fig.: The encoder board (optional)

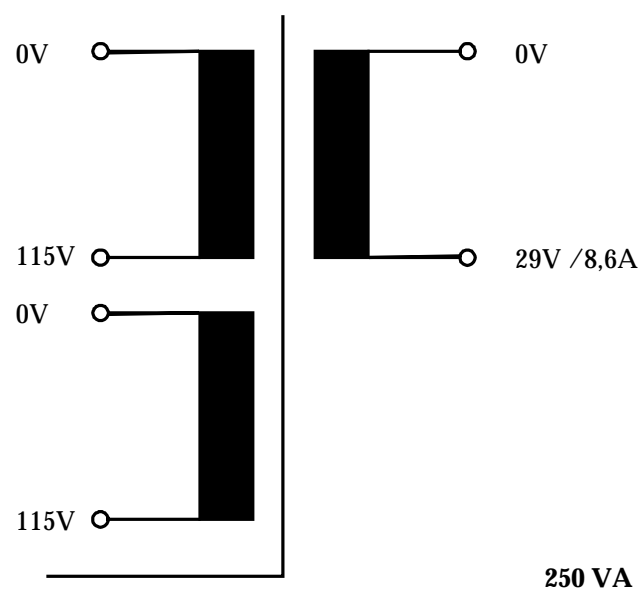
**Note:** The solder bridges of the encoder board are located on the soldered side of the circuit board.

## 6.10 Transformer Wiring

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

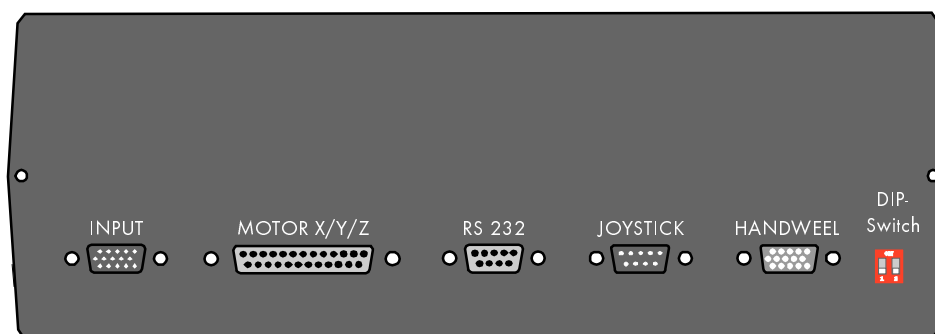


### LSTEP-3x/2



## 7 Appendix ECO-STEP

### 7.1 Back Panel Of The ECO-STEP



### 7.2 Motor Connection X/Y/Z

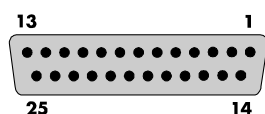


Fig.: Motor Connection (25 Pole, Sub-D Socket)

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	Gehäuse	GND

### 7.3 Voltage Connection

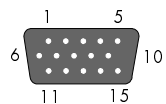


Fig.: Motor Connection (15 Pole, Sub-HD Socket)

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

### 7.4 DIP Switch Settings



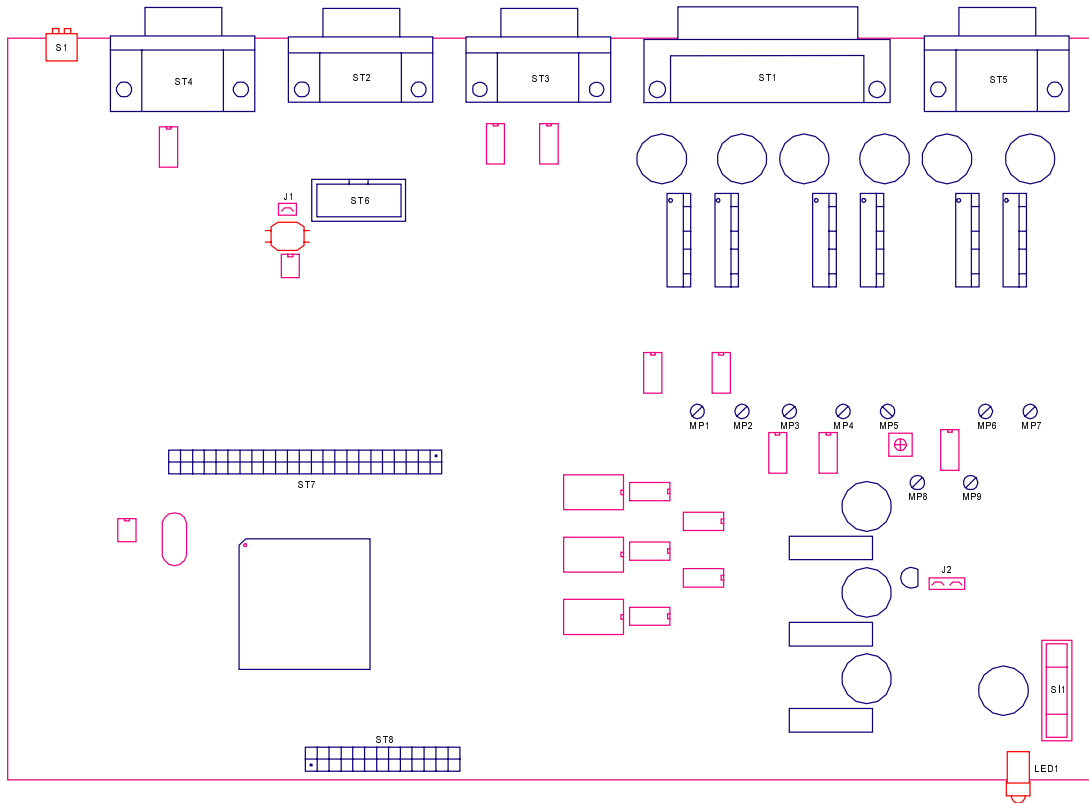
Fig.: DIP Switches Of The ECO-STEP

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

## 7.5 Technical Data

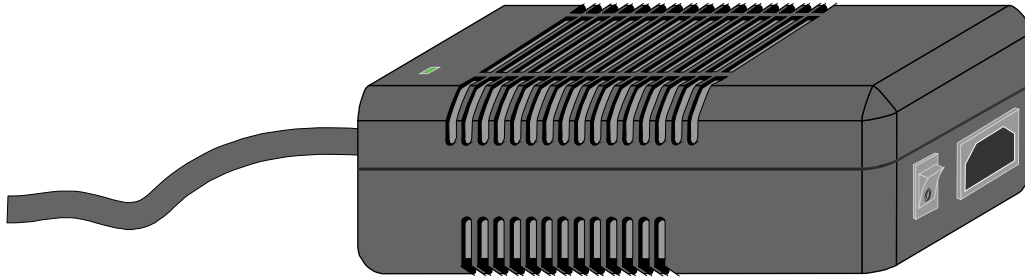
Power supply:	Table-top power pack / AC INPUT: 110V - 240V 1.5 A 47-63 Hz DC OUTPUT: +24V ---- 3A
Max. power failure duration:	< 50ms if the power fails (<0.77 * UN) the LSTEP switches to Reset
Max. motor speed:	15 r/sec. for a 200-step motor
Max. motor current:	1.25A per motor phase
Max. motor voltage:	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

## 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 Appendix LSTEP-API**

---

### **8.1 Introduction**

The LSTEP-API 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 instruction set of the LSTEP positioning systems.

#### **8.1.1 Included Functions**

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

#### **8.1.2 System Requirements**

Applications can be developed LSTEP-API on Intel-PCs under MS Windows 9x, Windows NT and Windows 2000.

#### **8.1.3 Supported Development Environments**

The LSTEP-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.)

## 8.2 DLL Interface

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 instructions, to inquire positional values, inputs/outputs, etc.

### 8.2.1 General Information

The DLL LSTEP4.DLL implements the instructions 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.

**Note:** Under Windows NT the supplied driver GIVEIO must be installed so that the ARCNET and DPRAM interfaces may be used.

### 8.2.2 Integration in Delphi

All function names of the LSTEP-API start with "LS\_" for easier differentiation. To be able to use the functions of the LSTEP APIs, LSTEP4.pas must be present in the uses-clause of the unit it question and must be in one of the preset search paths.

Required files: [LSTEP4.dll](#) and [LSTEP4.pas](#)

### 8.2.3 Integration in Visual C++

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](#)

### 8.2.4 Integration In LabVIEW

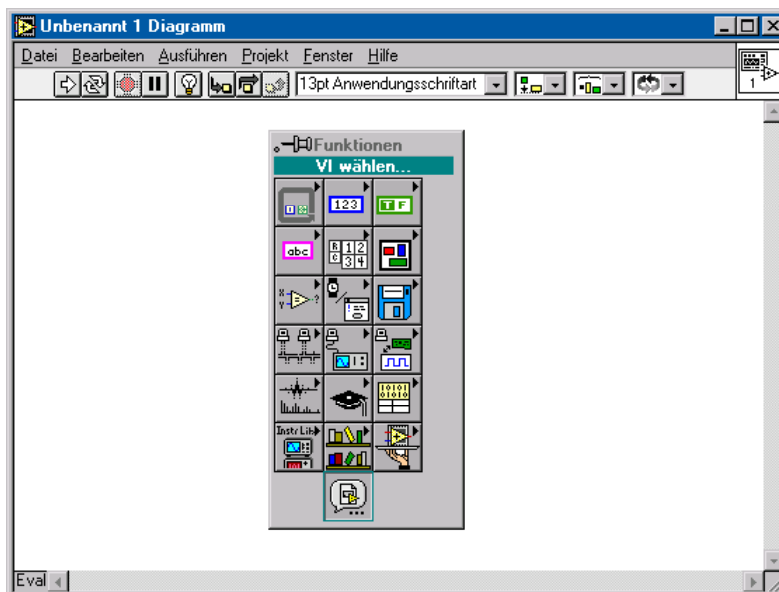
NI LabVIEW is a developing environment based on the graphic programming language G. It enables programming with graphic symbols to be done quickly and easily. Complicated 32-bit programs can be created, thus ensuring that the required speed of execution for control, test and measuring applications is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

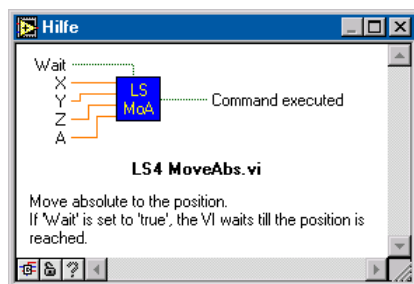
A VI library (LSTEP4.llb) has been created for LSTEP, which contains approx. 80 VIs. These individual VIs (e.g. LS4 ConnectSimple.vi) encapsulate the relevant LSTEP API-functions. The LSTEP4.dll is used by means of the "Call Library Function".

Procedure for using an LSTEP4 VIs:

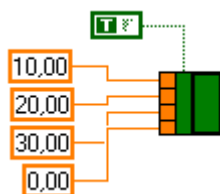
1. Create a new VI
2. Switch to the block diagram window (Ctrl+E)
3. Click on the diagram (right mouse button)



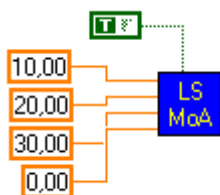
4. Select VI ...
5. Open the supplied VI Library LSTEP4.llb in the file dialog box, and then select the required instruction (e.g. LS4 MoveAbs.vi)
6. Place VI in the diagram
7. Press ctrl+H to open a help window, which gives you information about the VI at which the mouse pointer is currently located



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 instruction (X 10mm, Y 20mm, Z 30mm) is executed.

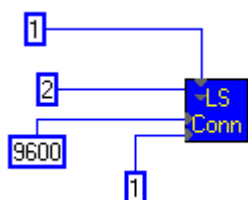


For details of the occupancy of the terminals of the Vis, please refer to the documentation for the API function in question. There, you will find a diagram, which also appears in the Help window of LabVIEW. The parameters are more or less the same as those of the DLL-function: There are only certain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

The LSTEP4 VIs have 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 initializes the interface and detects the LSTEP which is connected.

Example for RS232 (COM2 and 9600 Baud):



Required files: [LSTEP4.dll](#) and [LSTEP4.llb](#)

## 8.3 Error Codes

0: No error

### 8.3.1 LSTEP-Error/Fault Messages


- 1: Valid axis designation missing
- 2: Non-executable function
- 3: Command string has too many characters
- 4: Invalid command
- 5: Not within valid numerical range
- 6: Incorrect number of parameters
- 7: None !or ?
- 8: TVR not possible because axis is active
- 9: Axes cannot be switched on or off because TVR is active
- 10: Function not configured
- 11: Move command not possible, as joystick is in Manual
- 12: Limit switch tripped


### 8.3.2 API-Error/Fault Messages


- 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 LSTEP
- 4011: Joystick set to Manual (may occur with SetJoystickOn/Off)
- 4012: Travel command not possible, as joystick is in Manual
- 4013: Controller timeout

## 8.4 Functions

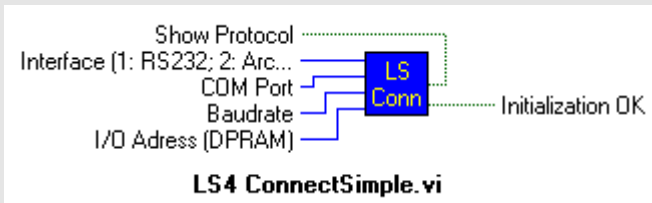
The descriptions of the functions of the LSTEP APIs are given in alphabetical order in this chapter.


LS_Calibrate	
<b>Description:</b>	Calibrate
	Moves all enabled axes towards smaller positional values. Travelling is stopped as soon as the limit switches are reached. The positional value is set to 0 .
<b>Delphi:</b>	function LS_Calibrate: Integer;
<b>C++:</b>	int Calibrate();
<b>LabView:</b>	 <span style="margin-left: 20px;">Command executed</span> <b>LS4 Calibrate.vi</b>
<b>Parameters:</b>	-
<b>Example:</b>	LS.Calibrate();


LS_CalibrateEx	
<b>Description:</b>	Calibrate
	(Only those axes are calibrated for which the relevant bit has been set in the transmitted integer valuet.)
<b>Delphi:</b>	function LS_CalibrateEx(Flags: Integer): Integer;
<b>C++:</b>	int CalibrateEx (int IFlags);
<b>LabView:</b>	 <span style="margin-left: 20px;">Command executed</span> <b>LS4 CalibrateEx.vi</b>
<b>Parameters:</b>	Flags: Bit mask, Bit 2 = 1 → Calibrate Z-axis Bit 2 = 0 → Do not calibrate Z-axis ...
<b>Example:</b>	LS.CalibrateEx(6); // Calibrate Y- and Z-axes only

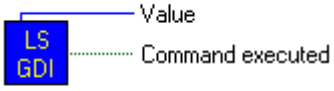
LS_Connect	
<b>Description:</b>	Connect with LSTEP
	<p>The interface parameters which were loaded from the INI file with LS_LoadConfig are used.</p> <p>(One of the functions LS_Connect, LS_ConnectSimple or LS_ConnectEx <u>must</u> be called to initialize the interface, so that communication with the LSTEP can take place.)</p>
<b>Delphi:</b>	function LS_Connect: Integer;
<b>C++:</b>	int Connect();
<b>LabView:</b>	
<b>Parameters:</b>	-
<b>Example:</b>	<pre>LS.LoadConfig("C:\LStepTest\LStep.INI"); LS.Connect();</pre>

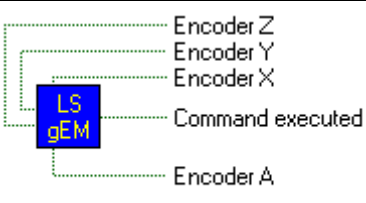
LS_ConnectEx	
<b>Description:</b>	Connect with LSTEP
	<p>This function offers extended possibilities. A pointer to a data structure which contains the interface parameters is transmitted. Information about the detected controller are also returned in the record (version number...)</p> <p>(One of the functions LS_Connect, LS_ConnectSimple or LS_ConnectEx <u>must</u> be called to initialize the interface, so that communication with the LSTEP can take place.)</p>
<b>Delphi:</b>	function LS_ConnectEx(var AControlInitPar: TLS_ControlInitPar): Integer;
<b>C++:</b>	int ConnectEx (TLS_ControlInitPar *pAControlInitPar);
<b>Parameters:</b>	AControlInitPar: Pointer to a record of the type TLS_ControlInitPar
<b>Example:</b>	LS.ConnectEx(&ControlInitPar1);

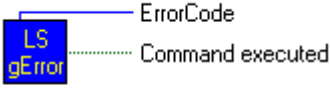
LS_ConnectSimple	
<b>Description:</b>	<p>Connect with LSTEP</p> <p>The settings of the interface are transmitted as parameters</p> <p>(One of the functions LS_Connect, LS_ConnectSimple or LS_ConnectEx <u>must</u> be called to initialize the interface, so that communication with the LSTEP can take place.)</p>
<b>Delphi:</b>	<pre>function LS_ConnectSimple(   AnInterfaceType: Integer;   AComName: PChar;   ABR: Integer;   AShowProt: LongBool): Integer;</pre>
<b>C++:</b>	<pre>int Connect (   int IAnInterfaceType,   char *pcAComName,   int IABR,   BOOL AShowProt);</pre>
<b>LabView:</b>	 <p style="text-align: center;"><b>LS4 ConnectSimple.vi</b></p>
<b>Parameters:</b>	<p>AnInterfaceType: Interface type</p> <p>1 = RS232 2 = ArcNet 3 = DPRAM</p> <p>AComName: Set name of the COM- interface, e.g.. 'COM2', for ArcNet orDPRAM to ZERO</p> <p>ABR: Meaning depends on the type of interface RS232 → Baud rate, e.g. 9600 ArcNet: → 0 for coax, 1 for twisted pair DPRAM: → Base-I/O-address of the board, e.g. 0x0340</p> <p>AShowProt: determines whether the interface protocol is to be shown</p>
<b>Example:</b>	<pre>LS.ConnectSimple(1, "COM2", 9600, true); // RS232, 9600 Baud or LS.ConnectSimple(3, NULL, 0x0340, true); // DPRAM, I/O-Address 0340 hex</pre>

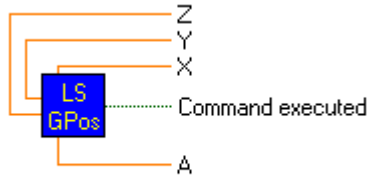
LS_Disconnect	
<b>Description:</b>	Terminate connection to LSTEP No more commands can be transmitted to the LSTEP after this function has been called. The function should be called just before ending the program.
<b>Delphi:</b>	function LS_Disconnect: Integer;
<b>C++:</b>	int Disconnect ();
<b>LabView:</b>	 Command executed <b>LS4 Disconnect.vi</b>
<b>Parameters:</b>	-
<b>Example:</b>	LS.Disconnect();

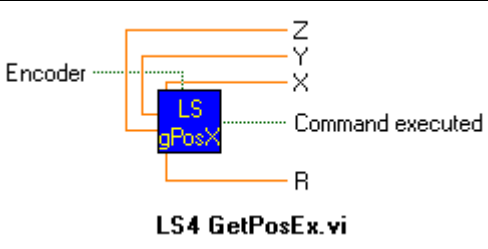
LS_GetAnalogInput	
<b>Description:</b>	Read the current status of an analog channel
<b>Delphi</b>	function LS_GetAnalogInput(Index: Integer; var Value: Integer): Integer;
<b>C++</b>	int GetAnalogInput (int lIndex,int *plValue);
<b>LabView</b>	 Command executed Value <b>LS4 GetAnalogInput.vi</b>
<b>Parameters:</b>	Index: 0-9 (Analog channels) Value: Pointer to integer value which states the current status of the analog channel
<b>Example:</b>	LS.GetAnalogInput(0, &Input0);

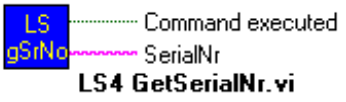
LS_GetDigitalInputs	
<b>Description:</b>	Read all input pins
<b>Delphi:</b>	function LS_GetDigitalInputs(var Value: Integer): Integer;
<b>C++:</b>	int GetDigitalInputs (int *plValue);
<b>LabView:</b>	 <b>LS4 GetDigitalInputs.vi</b>
<b>Parameters:</b>	Value: Pointer to integer value which contains the status of all inputs as a bit mask.
<b>Example:</b>	<pre>int inputs; LS.GetDigitalInputs(&amp;Inputs); if (Inputs &amp; 16) ... // when input pin 4 is set</pre>


LS_GetEncoderMask	
<b>Description:</b>	Read encoder statuses
<b>Delphi:</b>	function LS_GetEncoderMask (var Flags: Integer): Integer;
<b>C++:</b>	int GetEncoderMask (int *plFlags);
<b>LabView:</b>	 <b>LS4 GetEncoderMask.vi</b>
<b>Parameters:</b>	Flags: Encoder mask
<b>Example:</b>	<pre>int EncMask; LS.GetEncoderMask(&amp;EncMask); if (EncMask &amp; 2) ... // If Y-axis encoder is connected +active</pre>

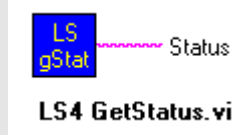
LS_GetError	
<b>Description:</b>	gives the current error number
<b>Delphi:</b>	function LS_GetError(var ErrorCode: Integer): Integer;
<b>C++:</b>	int GetError (int *pErrorCode);
<b>LabView:</b>	 <b>LS4 GetError.vi</b>
<b>Parameters:</b>	ErrorCode: error number
<b>Example:</b>	LS.GetError(&ErrCode);


LS_GetPos	
<b>Description:</b>	Inquires the current positions of all axes For non-existing axes, a value of 0.0 is returned
<b>Delphi:</b>	function LS_GetPos(var X, Y, Z, A: Double): Integer;
<b>C++:</b>	int GetPos (double *pdX,double *pdY,double *pdZ,double *pdA);
<b>LabView:</b>	 <b>LS4 GetPos.vi</b>
<b>Parameters:</b>	X, Y, Z, A: positional values
<b>Example:</b>	double X, Y, Z, A; LS.GetPos(&X, &Y, &Z, &A);


LS_GetPosEx	
<b>Description:</b>	Inquires the current encoder or positional values of all axes For non-existing axes, a value of 0.0 is returned
<b>Delphi:</b>	function LS_GetPosEx(var X, Y, Z, A: Double; Encoder: LongBool): Integer;
<b>C++:</b>	int GetPosEx (double *pdX,double *pdY,double *pdZ,double *pdA,BOOL Encoder);
<b>LabView:</b>	
<b>Parameters:</b>	X, Y, Z, A: positional values Encoder = true → Get encoder values if encoder is connected Encoder = false → Get positional values
<b>Example:</b>	double X, Y, Z, A; LS.GetPosEx(&X, &Y, &Z, &A, true);


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


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


LS_GetStatus	
<b>Description:</b>	Gives the current status of the controller.
<b>Delphi:</b>	function LS_GetStatus(Stat: PChar; MaxLen: Integer): Integer;
<b>C++:</b>	int GetStatus (char *pcStat,int lMaxLen);
<b>LabView:</b>	
<b>Parameters:</b>	Stat: Pointer to a buffer in which the status string is returned MaxLen: Maximum number of characters which may be copied into the buffer
<b>Example:</b>	LS.GetStatus(pcStat, 256);


LS_GetStatusAxis	
<b>Description:</b>	Gives the present status of the individual axes
<b>Delphi:</b>	function LS_GetStatusAxis(StatusAxisStr: PChar; MaxLen: Integer): Integer;
<b>C++:</b>	int GetStatusAxis (char *pcStatusAxisStr,int lMaxLen);
<b>LabView:</b>	 <b>LS4_GetStatusAxis.vi</b>
<b>Parameters:</b>	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 - @ = Axis at a standstill M = Axis is moving (Motion) - = Axis is not enabled
<b>Example:</b>	LS.GetStatusAxis(pcStatAxis, 256);

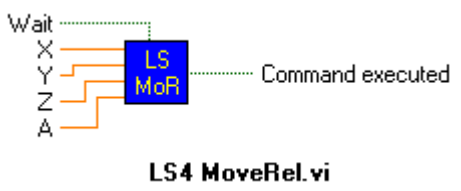
LS_GetVersionStr	
<b>Description:</b>	Returns the current version number of the Firmware
<b>Delphi:</b>	function LS_GetVersionStr(Vers: PChar; MaxLen: Integer): Integer;
<b>C++:</b>	int GetVersionStr (char *pcVers,int lMaxLen);
<b>LabView:</b>	 <b>LS4_GetVersionStr.vi</b>
<b>Parameters:</b>	Stat: Pointer to a buffer in which the version string is returned MaxLen: Maximum number of characters which can be copied into the buffer
<b>Example:</b>	LS.GetVersionStr(pcVers, 64); // Read version number


LS_GetSwitches	
<b>Description:</b>	Reads the status of all limit switches
<b>Delphi:</b>	function LS_GetSwitches(var Flags: Integer): Integer;
<b>C++:</b>	int GetSwitches (int *plFlags);
<b>LabView:</b>	 <b>LS4 GetSwitches.vi</b>
<b>Parameters:</b>	Value: Pointer to integer value which contains the status of all limit switches as a bit mask.
<b>Example:</b>	LS.GetSwitches(&Flags);


LS_LoadConfig	
<b>Description:</b>	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.
<b>Delphi:</b>	function LS_LoadConfig(FileName: PChar): Integer;
<b>C++:</b>	int LoadConfig (char *pcFileName);
<b>LabView:</b>	 <b>LS4 LoadConfig.vi</b>
<b>Parameters:</b>	FileName: File name of the INI-file as a zero-terminated string
<b>Example:</b>	LS.LoadConfig("C:\LStepTest\LStep.INI");

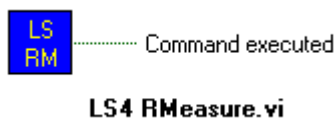
LS_MoveAbs	
<b>Description:</b>	Move to absolute position  (The X-, Y-, and Z- axes are positioned at the transmitted positional values.)
<b>Delphi:</b>	function LS_MoveAbs(X, Y, Z, A: Double; Wait: LongBool): Integer;
<b>C++:</b>	int MoveAbs (double dX, double dY, double dZ, double dA, BOOL Wait);
<b>LabView:</b>	
<b>Parameters:</b>	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)
<b>Example:</b>	LS.MoveAbs(10.0, 10.0, 10.0, 10.0, true);


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


LS_MoveRel	
<b>Description:</b>	Move to relative vector (The X-, Y-, and Z-axes are moved the transmitted distances.)
<b>Delphi:</b>	function LS_MoveRel(X, Y, Z, A: Double; Wait: LongBool): Integer;
<b>C++:</b>	int MoveRel (double dX, double dY, double dZ, double dA, BOOL Wait);
<b>LabView:</b>	
<b>Parameters:</b>	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)
<b>Example:</b>	LS.MoveRel(10.0, 10.0, 10.0, 10.0, true);


LS_MoveRelShort	
<b>Description:</b>	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 .
<b>Delphi:</b>	function LS_MoveRelShort: Integer;
<b>C++:</b>	int MoveRelShort ();
<b>LabView:</b>	
<b>Parameters:</b>	-
<b>Example:</b>	<pre> LS.SetDistance(1.0, 1.0, 0, 0); for (i = 0; i &lt; 10; i++) LS.MoveRelShort(); // Move the X- and Y- axes 10times 1 mm to the relative position           </pre>


LS_MoveRelSingleAxis	
<b>Description:</b>	Move individual axis relatively
<b>Delphi:</b>	function LS_MoveRelSingleAxis(Axis: Integer; Value: Double; Wait: LongBool): Integer;
<b>C++:</b>	int MoveRelSingleAxis (int lAxis,double dValue,BOOL Wait);
<b>LabView:</b>	 <b>LS4 MoveRelSingleAxis.vi</b>
<b>Parameters:</b>	Axis: (X, Y, Z, A numbered from 1 to 4) Value: Distance (Input depends on the preset dimensions)
<b>Example:</b>	LS.MoveRelSingleAxis(3, 5.0); // Move Z-axis 5mm in positive direction


LS_Rmeasure	
<b>Description:</b>	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.
<b>Delphi:</b>	function LS_Rmeasure: Integer;
<b>C++:</b>	int RMeasure();
<b>LabView:</b>	 <b>LS4 RMeasure.vi</b>
<b>Parameters:</b>	-
<b>Example:</b>	LS.RMeasure();

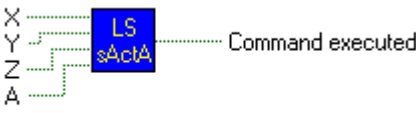
LS_RMeasureEx	
<b>Description:</b>	Measure table stroke (The table stroke is only measured for axes for which the relevant bit has been set in the transmitted integer value.)
<b>Delphi:</b>	function LS_RMeasureEx(Flags: Integer): Integer;
<b>C++:</b>	int RMeasureEx (int IFlags);
<b>LabView:</b>	 <b>LS4 RMeasureEx.vi</b>
<b>Parameters:</b>	Flags: Bit mask, Bit 2 = 1 → Calibrate Z-axis Bit 2 = 0 → Do not calibrate Z-axis ...
<b>Example:</b>	LS.RMeasureEx(2); // Measure table stroke (Y-axis only)


LS_SendString																
<b>Description:</b>	Send string to LSTEP															
<b>Delphi:</b>	function LS_SendString(Str, Ret: PChar; MaxLen: Integer; ReadLine: LongBool; TimeOut: Integer): Integer;															
<b>C++:</b>	int SendString (char *pcStr,char *pcRet,int lMaxLen,BOOL ReadLine,int lTimeOut);															
<b>LabView:</b>	 <b>LS4 SendString.vi</b>															
<b>Parameters:</b>	<table border="0" style="width: 100%;"> <tr> <td style="width: 15%;">Str</td> <td style="width: 10%;">→</td> <td>Zero terminated string which is to be transmitted to the controller.</td> </tr> <tr> <td>Ret</td> <td>→</td> <td>Buffer which contains the LSTEP feedback, if ReadLine = true</td> </tr> <tr> <td>MaxLen</td> <td>→</td> <td>Maximum number of characters which can be copied into the buffer.</td> </tr> <tr> <td>ReadLine</td> <td>→</td> <td>Read LSTEP feedback:</td> </tr> <tr> <td>TimeOut</td> <td>→</td> <td>Maximum time in which feedback must have occurred [ms]</td> </tr> </table>	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]
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]														
<b>Example:</b>	LS.SendString("?ver\r", pcLStepVer, 256, true, 1000); // Read version number, Timeout 1s															


LS_SetAbortFlag	
<b>Description:</b>	Set flag to terminate the communication with the LSTEP  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.  This function is especially useful in programs with message handling routines or several threads, if e.g. a movement is to be aborted quickly.
<b>Delphi:</b>	function LS_SetAbortFlag: Integer;
<b>C++:</b>	int SetAbortFlag ();
<b>LabView:</b>	 <b>LS4 SetAbortFlag.vi</b>
<b>Parameters:</b>	-
<b>Example:</b>	LS.SetAbortFlag(); LS.StopAxes(); (Terminate communication with the LSTEP and send the command to stop all axes)


LS_SetAccel	
<b>Description:</b>	Set acceleration
<b>Delphi:</b>	function LS_SetAccel(X, Y, Z, R: Double): Integer;
<b>C++:</b>	int SetAccel(double dX, double dY, double dZ, double dA);
<b>LabView:</b>	 <b>LS4 SetAccel.vi</b>
<b>Parameters:</b>	X, Y, Z and A  0.01 – 10.00 [m/s²]
<b>Example:</b>	LS.SetAccel(1.0, 1.5, 0, 0);


LS_SetAccelSingleAxis	
<b>Description:</b>	Set acceleration
<b>Delphi:</b>	function LS_SetAccelSingleAxis(Axis: Integer; Accel: Double): Integer;
<b>C++:</b>	int SetAccelSingleAxis (int lAxis,double dAccel);
<b>LabView:</b>	 <b>LS4 SetAccelSingleAxis.vi</b>
<b>Parameters:</b>	Axis: (X, Y, Z, A numbered from 1 to 4) Accel: Acceleration 0.01 – 10.00 [m/s <sup>2</sup> ]
<b>Example:</b>	LS.SetAccelSingleAxis(4, 1.0); // Accelerate A-axis 1.0 m/s <sup>2</sup>


LS_SetActiveAxes	
<b>Description:</b>	Enable axes
<b>Delphi:</b>	function LS_SetActiveAxes(Flags: Integer): Integer;
<b>C++:</b>	int SetActiveAxes(int Flags);
<b>LabView:</b>	 <b>LS4 SetActiveAxes.vi</b>
<b>Parameters:</b>	Flags: Bit mask Bit 0 = 1 → X-axis enabled, i.e. can be travelled Bit 2 = 0 → Z-axis not enabled
<b>Example:</b>	LS.SetActiveAxes(3); /* Enable X- and Y-axes (Bits 0 and. 1 set), do not enable Z-axis (Bit 2 = 0) */

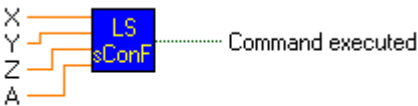
LS_SetAnalogOutput	
<b>Description:</b>	Set analog output
<b>Delphi:</b>	function LS_SetAnalogOutput(Index: Integer; Value: Integer): Integer;
<b>C++:</b>	int SetAnalogOutput (int IIndex,int IValue);
<b>LabView:</b>	 <b>LS4 SetAnalogOutput.vi</b>
<b>Parameters:</b>	Index: 0-1 (Analog channels) Value: 0-100 [%]
<b>Example:</b>	<pre>LS.SetAnalogOutput(0, 100); // Set output 0 to maximum</pre>

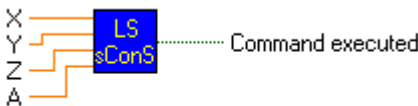
LS_SetAutoStatus	
<b>Description:</b>	AutoStatus On/Off  Note: The AutoStatus mode should not normally be changed, as LSTEP API sets the correct mode for travel commands etc. Changing to 0 or 2 could lead to errors
<b>Delphi:</b>	function LS_SetAutoStatus(Value: Integer): Integer;
<b>C++:</b>	int SetAutoStatus (int IValue);
<b>LabView:</b>	 <b>LS4 SetAutoStatus.vi</b>
<b>Parameters:</b>	Value: AutoStatus mode:  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.
<b>Example:</b>	LS.SetAutoStatus(3);

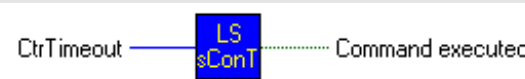
LS_SetCalibOffset	
<b>Description:</b>	Calibration offset
<b>Delphi:</b>	function LS_SetCalibOffset(X, Y, Z, A: Double): Integer;
<b>C++:</b>	int SetCalibOffset (double dX,double dY,double dZ,double dA);
<b>LabView:</b>	
<b>Parameters:</b>	X, Y, Z and A 0 – 32*50000 (32*Spindle pitch)
<b>Example:</b>	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_SetController	
<b>Description:</b>	Set controller mode
<b>Delphi:</b>	function LS_SetController(XC, YC, ZC, AC: Integer): Integer;
<b>C++:</b>	int SetController (int IXC,int IYC,int IZC,int IAC);
<b>LabView:</b>	
<b>Parameters:</b>	Controller mode X, Y, Z and A : 0 → Controller “OFF” 1 → Controller “OFF after reaching target position” 2 → Controller “Always ON”
<b>Example:</b>	LS.SetController(1, 2, 0, 0);


LS_SetControllerCall	
<b>Description:</b>	Call controller
<b>Delphi:</b>	function LS_SetControllerCall(CtrCall: Integer): Integer;
<b>C++:</b>	int SetControllerCall (int lCtrCall);
<b>LabView:</b>	 <b>LS4 SetControllerCall.vi</b>
<b>Parameters:</b>	CtrCall: Controller call time [ms]
<b>Example:</b>	LS.SetControllerCall(10);

LS_SetControllerFactor	
<b>Description:</b>	Controller factor
<b>Delphi:</b>	function LS_SetControllerFactor(X, Y, Z, A: Double): Integer;
<b>C++:</b>	int SetControllerFactor (double dX,double dY,double dZ,double dA);
<b>LabView:</b>	 <b>LS4 SetControllerFactor.vi</b>
<b>Parameters:</b>	X, Y, Z and A 1 – 64
<b>Example:</b>	LS.SetControllerFactor(1, 2, 3, 4);


LS_SetControllerSteps	
<b>Description:</b>	Controller steps
<b>Delphi:</b>	function LS_SetControllerSteps(X, Y, Z, A: Double): Integer;
<b>C++:</b>	int SetControllerSteps (double dX,double dY,double dZ,double dA);
<b>LabView:</b>	 <b>LS4 SetControllerSteps.vi</b>
<b>Parameters:</b>	X, Y, Z and A 1 – Spindle pitch (Values depend on the dimension)
<b>Example:</b>	LS.SetControllerSteps(4, 5, 7, 9);

LS_SetControllerTimeout	
<b>Description:</b>	Controller timeout
<b>Delphi:</b>	function LS_SetControllerTimeout(ACtrTimeout: Integer): Integer;
<b>C++:</b>	int SetControllerSteps (int ACtrTimeout);
<b>LabView:</b>	 <b>LS4 SetControllerTimeout.vi</b>
<b>Parameters:</b>	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.
<b>Example:</b>	LS.SetControllerTimeout(500);

LS_SetControllerTWDelay	
<b>Description:</b>	Controller delay
<b>Delphi</b>	function LS_SetControllerTWDelay(CtrTWDelay: Integer): Integer;
<b>C++</b>	int SetControllerTWDelay (int lCtrTWDelay);
<b>LabView</b>	 <b>LS4 SetControllerTWDelay.vi</b>
<b>Parameters:</b>	CtrTWDelay: Controller delay 0 - 100 [ms]
<b>Example:</b>	LS.SetControllerTWDelay(0); // Controller delay off


LS_SetControlPars	
<b>Description:</b>	Transmits the parameters which were loaded with LS_LoadConfig to the LSTEP.
<b>Delphi:</b>	function LS_SetControlPars: Integer;
<b>C++:</b>	int SetControlPars ();
<b>LabView:</b>	 <b>LS4 SetControlPars.vi</b>
<b>Parameters:</b>	-
<b>Example:</b>	LS.SetControlPars();


LS_SetCTS	
<b>Description:</b>	CTS interpretation of the RS232- interface
<b>Delphi:</b>	function LS_SetCTS(Value: Boolean): Integer;
<b>C++:</b>	int SetCTS (BOOL IValue);
<b>Parameters:</b>	True → activates the CTS interpretation of the RS232 interface
	False → deactivates the CTS interpretation of the RS232 interface
<b>Example:</b>	LS.SetCTS(true);


LS_SetDelay	
<b>Description:</b>	The delay command is used to produce a vector start delay.
<b>Delphi:</b>	function LS_SetDelay(Delay: Integer): Integer;
<b>C++:</b>	int SetDelay (int IDelay);
<b>LabView:</b>	 <b>LS4 SetDelay.vi</b>
<b>Parameters:</b>	0 – 10000 (ms)
<b>Example:</b>	LS.SetDelay(1000); // 1s delay

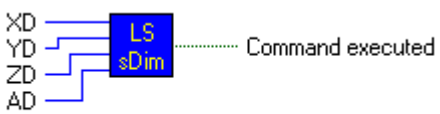
LS_SetDigIO_Distance	
<b>Description:</b>	Function of the digital inputs/outputs Activation of an output dependent on the set distance before /after the target position.
<b>Delphi:</b>	function LS_SetDigIO_Distance(Index: Integer; Fkt: LongBool; Dist: Double; Axis: Integer): Integer;
<b>C++:</b>	int SetDigIO_Distance (int lIndex,BOOL Fkt,double dDist,int lAxis);
<b>LabView:</b>	 <b>LS4 SetDigIO_Distance.vi</b>
<b>Parameters:</b>	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)
<b>Example:</b>	<pre> LS.SetDigIO_Distance(7, false, 78.9, 3); /* Output 7 is activated 78.9mm before the target position (Z-axis) is reached. */           </pre>

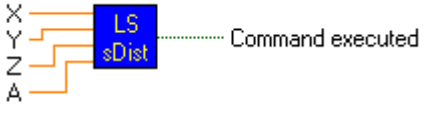
LS_SetDigIO_EmergencyStop	
<b>Description:</b>	Function of the digital inputs/outputs Allocation of the Emergency Stop pin
<b>Delphi:</b>	function LS_SetDigIO_EmergencyStop(Index: Integer): Integer;
<b>C++:</b>	int SetDigIO_EmergencyStop (int lIndex);
<b>LabView:</b>	 <b>LS4 SetDigIO_EmergencyStop.vi</b>
<b>Parameters:</b>	Index: 0 to 15 (Input/Output)
<b>Example:</b>	<pre> LS.SetDigIOEmergencyStop(15); // Emergency Stop pin 15           </pre>


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


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

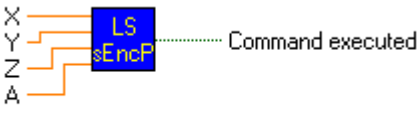
LS_SetDigitalOutput	
<b>Description:</b>	Set output pin
<b>Delphi:</b>	function LS_SetDigitalOutput(Index: Integer; Value: LongBool): Integer;
<b>C++:</b>	int SetDigitalOutput (int IIndex,BOOL Value);
<b>LabView:</b>	 <b>LS4 SetDigitalOutput.vi</b>
<b>Parameters:</b>	Index: 0-15 Value: Set status to “0” or “1”
<b>Example:</b>	LS.SetDigitalOutput(0, true); // Set output pin 0 to “1”


LS_SetDimensions	
<b>Description:</b>	Set dimensions of the axes
<b>Delphi:</b>	function LS_SetDimensions(XD, YD, ZD, AD: Integer): Integer;
<b>C++:</b>	int SetDimensions (int IXD,int IYD,int IZD,int IAD);
<b>LabView:</b>	 <b>LS4 SetDimensions.vi</b>
<b>Parameters:</b>	Dimensions of the X, Y, Z and A-axes: 0 → Microsteps 1 → μm 2 → Millimeters 3 → Degrees 4 → Revolutions
<b>Example:</b>	LS.SetDimensions(3, 2, 2); // X-axis in degrees; Y and Z in mm


LS_SetDistance	
<b>Description:</b>	Set distance (for LS_MoveRelShort)
<b>Delphi:</b>	function LS_SetDistance(X, Y, Z, A: Double): Integer;
<b>C++:</b>	int SetDistance (double dX,double dY,double dZ,double dA);
<b>LabView:</b>	 <b>LS4 SetDistance.vi</b>
<b>Parameters:</b>	X, Y, Z and A Min./max. range of travel (Values depend on the dimension)
<b>Example:</b>	<pre> 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. */ </pre>


LS_SetEncoderActive	
<b>Description:</b>	This function is used to select which encoder is to be activated after calibration.
<b>Delphi:</b>	function LS_SetEncoderActive(Flags: Integer): Integer;
<b>C++:</b>	int SetEncoderActive (int IFlags);
<b>LabView:</b>	 <b>LS4 SetEncoderActive.vi</b>
<b>Parameters:</b>	Value: Encoder mask
<b>Example:</b>	<pre> LS.SetEncoderActive(0); // Deactivate all encoders  LS.SetEncoderMask(2); // Activate Y-axis encoder </pre>

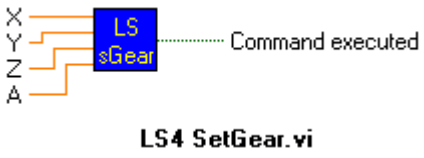
LS_SetEncoderMask	
<b>Description:</b>	(de) activate encoders
<b>Delphi:</b>	function LS_SetEncoderMask(Value: Integer): Integer;
<b>C++:</b>	int SetEncoderMask (int IValue);
<b>LabView:</b>	 <b>LS4 SetEncoderMask.vi</b>
<b>Parameters:</b>	Value: Encoder mask
<b>Example:</b>	<pre> LS.SetEncoderMask(0); // Deactivate all encoders  LS.SetEncoderMask(2); // Activate Y-axis encoder </pre>


LS_SetEncoderPeriod	
<b>Description:</b>	Set length of encoder period
<b>Delphi:</b>	function LS_SetEncoderPeriod(X, Y, Z, A: Double): Integer;
<b>C++:</b>	int SetEncoderPeriod (double dX,double dY,double dZ,double dA);
<b>LabView:</b>	 <b>LS4 SetEncoderPeriod.vi</b>
<b>Parameters:</b>	X, Y, Z and A 0.0001 – Spindle pitch * 0.8 (mm)
<b>Example:</b>	<pre> LS.SetEncoderPeriod(0.1, 0.1, 0.1, 0.1); // Encoder period length is 0.1 mm for all axes </pre>


LS_SetEncoderPosition	
<b>Description:</b>	Encoder position display On/Off
<b>Delphi</b>	function LS_SetEncoderPosition(Value: Boolean): Integer;
<b>C++</b>	int SetEncoderPosition (BOOL fValue);
<b>LabView</b>	 <b>LS4 SetEncoderPosition.vi</b>
<b>Parameters:</b>	Value = true → The encoder values of the detected encoders are displayed when the position inquiry is placed
<b>Example:</b>	LS.SetEncoderPosition(true);


LS_SetEncoderRefSignal	
<b>Description:</b>	Interpret reference signal from encoder when calibration is done
<b>Delphi:</b>	function LS_SetEncoderRefSignal(XR, YR, ZR, AR: Integer): Integer;
<b>C++:</b>	int SetEncoderRefSignal (int IXR,int IYR,int IZR,int IAR);
<b>LabView:</b>	 <b>LS4 SetEncoderRefSignal.vi</b>
<b>Parameters:</b>	X, Y, Z and A 0 or 1
<b>Example:</b>	LS.SetEncoderRefSignal(1, 1, 0, 0); /* When calibration is done, the reference signal of the encoders x and y are interpreted. */


LS_SetFactorTVR	
<b>Description:</b>	Factor for clock pulse Forward/ Back
<b>Delphi:</b>	function LS_SetFactorTVR(X, Y, Z, A: Double): Integer;
<b>C++:</b>	int SetFactorTVR (double dX,double dY,double dZ,double dA);
<b>LabView:</b>	
<b>Parameters:</b>	X, Y, Z and A 0.01 – 100.00
<b>Example:</b>	<pre> 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 */ </pre>


LS_SetGear	
<b>Description:</b>	Program gear transmission
<b>Delphi:</b>	function LS_SetGear(X, Y, Z, A: Double): Integer;
<b>C++:</b>	int SetGear (double dX,double dY,double dZ,double dA);
<b>LabView:</b>	
<b>Parameters:</b>	X, Y, Z and A 0.01 – 1000
<b>Example:</b>	<pre> 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 */ </pre>


LS_SetHandWheelOff	
<b>Description:</b>	Handwheel Off
<b>Delphi:</b>	function LS_SetHandWheelOff: Integer;
<b>C++:</b>	int SetHandWheelOff ();
<b>LabView:</b>	 Command executed
<b>Parameters:</b>	-
<b>Example:</b>	LS.SetHandWheelOff();


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


LS_SetJoystickDir	
<b>Description:</b>	Joystick direction
<b>Delphi:</b>	function LS_SetJoystickDir(XD, YD, ZD, AD: Integer): Integer;
<b>C++:</b>	int SetJoystickDir (int IxD,int IYD,int IZD,int IAD);
<b>LabView:</b>	
<b>Parameters:</b>	X, Y, Z, and A 0 → Axis disabled 1 → Positive direction of rotation -1 → Negative direction of rotation
<b>Example:</b>	<pre>LS.SetJoystickDir(1, 1, -1, 0); /* X- and Y-axis positive direction of rotation; Z-axis negative direction of rotation; A-axis disabled */</pre>

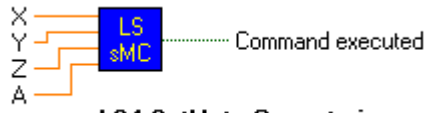
LS_SetJoystickOff	
<b>Description:</b>	Analog joystick Off
<b>Delphi:</b>	function LS_SetJoystickOff: Integer;
<b>C++:</b>	int SetJoystickOff();
<b>LabView:</b>	
<b>Parameters:</b>	-
<b>Example:</b>	LS.SetJoystickOff();

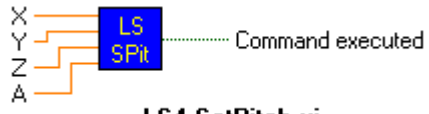
LS_SetJoystickOn	
<b>Description:</b>	Analog joystick On
<b>Delphi:</b>	function LS_SetJoystickOn(PositionCount, Encoder: LongBool): Integer;
<b>C++:</b>	int SetJoystickOn (BOOL PositionCount,BOOL Encoder);
<b>LabView:</b>	
<b>Parameters:</b>	PositionCount: Position count On/Off Encoder: Encoder values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


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


LS_SetLimit	
<b>Description:</b>	Set travel limits
<b>Delphi:</b>	function LS_SetLimit(Axis: Integer; MinRange, MaxRange: Double): Integer;
<b>C++:</b>	int SetLimit (int lAxis,double dMinRange,double dMaxRange);
<b>LabView:</b>	
<b>Parameters:</b>	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
<b>Example:</b>	LS.SetLimit(1, -10.0, 20.0); (Set 10 as minimum limit and 20 as maximum limit for the X-axis)


LS_SetLimitControl	
<b>Description:</b>	Control/monitoring of the range of travel
<b>Delphi:</b>	function LS_SetLimitControl(Axis: Integer; Active: LongBool): Integer;
<b>C++:</b>	int SetLimitControl (int IAxis,BOOL Active);
<b>LabView:</b>	 <b>LS4 SetLimitControl.vi</b>
<b>Parameters:</b>	Axis: (X, Y, Z, A numbered from 1 to 4) Active: Activate limit control for the axis in question
<b>Example:</b>	LS.SetLimitControl(2, true); // Limit control for Y-axis active


LS_SetMotorCurrent	
<b>Description:</b>	Set motor current
<b>Delphi:</b>	function LS_SetMotorCurrent(X, Y, Z, A: Double): Integer;
<b>C++:</b>	int SetMotorCurrent (double dX,double dY,double dZ,double dA);
<b>LabView:</b>	 <b>LS4 SetMotorCurrent.vi</b>
<b>Parameters:</b>	Motor current X, Y, Z, A-axis [A]
<b>Example:</b>	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_SetPitch	
<b>Description:</b>	Set spindle pitch
<b>Delphi:</b>	function LS_SetPitch(X, Y, Z, R: Double): Integer;
<b>C++:</b>	int SetPitch(double dX, double dY, double dZ, double dA);
<b>LabView:</b>	 <b>LS4 SetPitch.vi</b>
<b>Parameters:</b>	X, Y, Z and A 0.001 – 100 [mm]
<b>Example:</b>	LS.SetPitch(4, 4, 4, 4); // Set spindle pitches to 4 mm for all axes

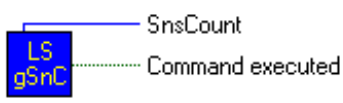
LS_SetPos	
<b>Description:</b>	Set positional values
<b>Delphi:</b>	function LS_SetPos(X, Y, Z, R: Double): Integer;
<b>C++:</b>	int SetPos(double dX, double dY, double dZ, double dA);
<b>LabView:</b>	
<b>Parameters:</b>	X, Y, Z and A Min./Max. range of travel Input depends on the dimension
<b>Example: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

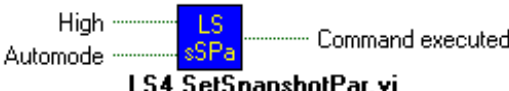
LS_SetReduction	
<b>Description:</b>	Set current reduction In quiescent state the rated motor current is reduced to the parameterized ratio.
<b>Delphi:</b>	function LS_SetReduction(X, Y, Z, R: Double): Integer;
<b>C++:</b>	int SetReduction(double dX, double dY, double dZ, double dA);
<b>LabView:</b>	
<b>Parameters:</b>	X, Y, Z and A 0 – 1.0
<b>Example:</b>	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 ... */

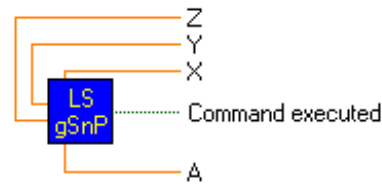
LS_SetRMOffset	
<b>Description:</b>	RM Offset
<b>Delphi:</b>	function LS_SetRMOffset(X, Y, Z, A: Double): Integer;
<b>C++:</b>	int SetRMOffset (double dX,double dY,double dZ,double dA);
<b>LabView:</b>	 <b>LS4 SetRMOffset.vi</b>
<b>Parameters:</b>	X, Y, Z and A 0 – 32*50000 (32*spindle pitch)
<b>Example:</b>	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_SetShowProt	
<b>Description:</b>	Interface protocol On/Off
<b>Delphi:</b>	function LS_SetShowProt>ShowProt: LongBool): Integer;
<b>C++:</b>	int SetShowProt (BOOL ShowProt);
<b>LabView:</b>	 <b>LS4 SetShowProt.vi</b>
<b>Parameters:</b>	ShowProt: Specifies whether the window “Interface Protocol” is to be shown or not
<b>Example:</b>	LS.SetShowProt(true); // Show interface protocol if not already visible


LS_SetSnapshot	
<b>Description:</b>	Snapshot On/Off
<b>Delphi:</b>	function LS_SetSnapshot(ASnapshot: LongBool): Integer;
<b>C++:</b>	int SetSnapshot (BOOL bASnapshot);
<b>LabView:</b>	 <b>LS4 SetSnapshot.vi</b>
<b>Parameters:</b>	ASnapshot: Snapshot On/Off
<b>Example:</b>	LS.SetSnapshot(true);


LS_GetSnapshotCount	
<b>Description:</b>	Snapshot counter
<b>Delphi:</b>	function LS_GetSnapshotCount(var SnsCount: Integer): Integer;
<b>C++:</b>	int GetSnapshotCount (int *pISnsCount);
<b>LabView:</b>	 <b>LS4 GetSnapshotCount.vi</b>
<b>Parameters:</b>	SnsCount: Snapshot counter
<b>Example:</b>	LS.GetSnapshotCount(&SnsCount);


LS_SetSnapshotPar	
<b>Description:</b>	Snapshot parameters
<b>Delphi:</b>	function LS_SetSnapshotPar(High, AutoMode: LongBool): Integer;
<b>C++:</b>	int SetSnapshotPar (BOOL bHigh, BOOL bAutoMode);
<b>LabView:</b>	 <b>LS4 SetSnapshotPar.vi</b>
<b>Parameters:</b>	High: Snapshot high-active AutoMode: approach snapshot position automatically
<b>Example:</b>	LS.SetSnapshotPar(true, false);


LS_GetSnapshotPos	
<b>Description:</b>	Read snapshot position
<b>Delphi</b>	function LS_GetSnapshotPos(var X, Y, Z, A: Double): Integer;
<b>C++</b>	int GetSnapshotPos (double *pdX, double *pdY, double *pdZ, double *pdA);
<b>LabView:</b>	 <b>LS4 GetSnapshotPos.vi</b>
<b>Parameters:</b>	X, Y, Z, A: Positional values
<b>Example:</b>	double X, Y, Z, A; LS.GetSnapshotPos(&X, &Y, &Z, &A);


LS_SetSpeedPoti	
<b>Description:</b>	Potentiometer On/Off
<b>Delphi:</b>	function LS_SetSpeedPoti(SpeedPoti: LongBool): Integer;
<b>C++:</b>	int SetSpeedPoti (BOOL SpeedPoti);
<b>LabView:</b>	
<b>Parameters:</b>	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.
<b>Example:</b>	<pre>LS.SetSpeedPoti(true); // Poti On</pre>


LS_SetSwitchActive	
<b>Description:</b>	Limit switch On/Off
<b>Delphi:</b>	function LS_SetSwitchActive(XA, YA, ZA, AA: Integer): Integer;
<b>C++:</b>	int SetSwitchActive (int IXA,int IYA,int IZA,int IAA);
<b>LabView:</b>	
<b>Parameters:</b>	<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>
<b>Example:</b>	<pre>LS.SetSwitchActive(7, 1, 5, 0);</pre> <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_SetSwitchPolarity	
<b>Description:</b>	Set limit switch polarity
<b>Delphi:</b>	function LS_SetSwitchPolarity(XP, YP, ZP, AP: Integer): Integer;
<b>C++:</b>	int SetSwitchPolarity (int IXP,int IYP,int IZP,int IRA);
<b>LabView:</b>	 <b>LS4 SetSwitchPolarity.vi</b>
<b>Parameters:</b>	<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>
<b>Example:</b>	LS.SetSwitchPolarity(7, 0, 0, 0); (All X-axis limit switches high-active, all Y-axis limit switches low-active...)


LS_SetTargetWindow	
<b>Description:</b>	Target window
<b>Delphi:</b>	function LS_SetTargetWindow(X, Y, Z, A: Double): Integer;
<b>C++:</b>	int SetTargetWindow (double dX,double dY,double dZ,double dA);
<b>LabView:</b>	 <b>LS4 SetTargetWindow.vi</b>
<b>Parameters:</b>	<p>X, Y, Z and A</p> <p>1 – 25000 (motor increments)</p> <p>0.1 – Spindle pitch/2 (µm)</p> <p>0.0001 – Spindle pitch/2 (mm)</p> <p>(Values depend on the dimension)</p>
<b>Example:</b>	LS.SetTargetWindow(1.0, 0.002, 1.0, 1.0);


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


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


LS_SetTVRMode	
<b>Description:</b>	Set clock pulse Forward /Back (= TVR Mode)
<b>Delphi:</b>	function LS_SetTVRMode(XT, YT, ZT, AT: Integer): Integer;
<b>C++:</b>	int SetTVRMode (int lXT,int lYT,int lZT,int lAT);
<b>LabView:</b>	 <b>LS4 SetTVRMode.vi</b>
<b>Parameters:</b>	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.
<b>Example:</b>	<pre>LS.SetTVRMode(1, 1, 0, 0); // TVR ON for X- and Y-axes</pre>

LS_SetVel	
<b>Description:</b>	Set speed (velocity)
<b>Delphi:</b>	function LS_SetVel(X, Y, Z, R: Double): Integer;
<b>C++:</b>	int SetVel(double dX, double dY, double dZ, double dA);
<b>LabView:</b>	 <b>LS4 SetVel.vi</b>
<b>Parameters:</b>	X, Y, Z and A 0 – maximum speed [r/s]
<b>Example:</b>	<pre>LS.SetVel(1.0, 15.0, 0, 0);</pre>

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

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

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

<b>LS_StopAxes</b>	
<b>Description:</b>	Stop (All movements are stopped)
<b>Delphi</b>	function LS_StopAxes: Integer;
<b>C++</b>	int StopAxes ();
<b>LabView</b>	 <span style="margin-left: 10px;">Command executed</span> <b>LS4 StopAxes.vi</b>
<b>Parameters:</b>	-
<b>Example:</b>	LS.StopAxes();