

# The Instruction Set of the TANGO Controller



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



# 1. Table of Contents

1.	<b>Table of Contents</b>	2
2.	<b>Introduction</b>	7
3.	<b>Hint for controller initialization</b>	9
4.	<b>Brief Description of the Tango Instruction Set</b>	10
5.	<b>Instruction Syntax Description</b>	16
6.	<b>Error Numbers and their possible Root Cause</b>	16
7.	<b>Controller Informations</b>	17
7.1.	version (Read detailed Version information)	17
7.2.	det (Read detailed Configuration)	18
7.3.	detext (Read Extended detailed Configuration)	18
7.4.	readsn (Read Serial Number)	19
7.5.	ver (Read default Version Number)	19
7.6.	iver (Read internal Version Number)	19
7.7.	uptime (Read Controller Up Time)	20
7.8.	temp (Read Case Temperature)	20
7.9.	maxaxis (Read number of available Axes)	20
7.10.	maxcur (Read Maximum Motor Current)	20
7.11.	etspresent (Read ETS Detect State)	21
7.12.	stagesn (Read Connected Devices Serial Number)	22
8.	<b>Communication Interface Settings</b>	23
8.1.	baud (Baud Rate)	23
8.2.	cts (Enable/Disable RS232 Hardware Handshake)	23
9.	<b>System Instructions</b>	24
9.1.	save (Save Parameters)	24
9.2.	restore (Restore Saved Parameters)	24
9.3.	reset (Force a Software Reset)	25
9.4.	pa (Enable or Disable the Power Amplifiers)	25
9.5.	ipreter (Select Instruction Set)	26
10.	<b>Operating Modes</b>	27
10.1.	Extended Mode	27
10.1.1	extmode (Switch to Extended Mode)	27
10.2.	Scan Mode	28
10.2.1	scanmode (Switch to Scan Mode)	28
10.2.2	scanvel (Scanmode Vector Velocity)	28
10.2.3	modulomode (Define Linear or Turntable Modes)	29
11.	<b>Controller States and Error Messages</b>	30
11.1.	autostatus (Set Autostatus to required behavior)	30
11.2.	statusaxis (Read State of Axis)	31
11.3.	status (Read the Controller Error State)	31
11.4.	err (Read Error Number)	32
11.5.	help (Read Error Number with Description String)	32
11.6.	service (Print Service Information to Terminal)	32
11.7.	pci (Is PCI Bus)	33
11.8.	isvel (Read Actual Velocities)	33
11.9.	maxpos (Maximum Position)	33
12.	<b>General Adjustments</b>	34
12.1.	dim (Unit for Positions and Velocities)	34



- 12.2. pitch (Spindle Pitch).....35
- 12.3. gear (Gear Ratio).....35
- 12.4. motorsteps (Motor Steps Per Revolution).....36
- 12.5. accel (Acceleration).....37
- 12.6. accelfunc (Acceleration Ramp Function).....37
- 12.7. stopaccel (Emergency Stop Deceleration).....38
- 12.8. vel (Velocity).....39
- 12.9. velfac (Velocity Factor).....40
- 12.10. secvel (Secure Velocity).....40
- 12.11. cur (Motor Current).....41
- 12.12. reduction (Motor Current Reduction Factor).....42
- 12.13. curdelay (Delay for Current Reduction).....42
- 12.14. ecomove (EcoMove Current Level).....43
- 12.15. axis (Enable, Disable, Switch Off Axis).....44
- 12.16. axisdir (Axis Direction).....44
- 12.17. motortable (Motor Correction Table).....45
- 12.18. usteps (Microstep Resolution).....45
- 12.19. resolution (Position Number Format).....46
- 12.20. backlash (Mechanical Backlash Compensation).....46
- 12.21. lock (Select Parameters to Lock).....47
- 12.22. lockaxis (Apply the Parameter Lock to Axes).....47
- 12.23. lockstate (Read all internal Lock States).....48
- 12.24. stout (Select Status Signal Output).....48
- 12.25. noled (Select Status Signal Output).....49
- 12.26. updelay (Power Up Delay).....49
- 13. Limit Switch Instructions (Hardware and Software).....50**
- 13.1. lim (Software Limits).....50
- 13.2. limctr (Enable or Disable Limit Control).....50
- 13.3. nosetlimit (Do not set limits by cal/rm).....51
- 13.4. swtyp (Type of Limit Switch).....51
- 13.5. swpol (Polarity of Limit Switch).....52
- 13.6. swdir (swap assignment of cal and rm switch).....52
- 13.7. swact (enable or disable limit switches).....53
- 13.8. readsw (Read Status of Limit Switches).....54
- 13.9. swin (Read Limit Switch Input Level).....54
- 13.10. statuslimit (Limit Status).....55
- 14. Calibration and Range Measure Instructions.....56**
- 14.1. cal (Perform Calibration to lower Limit Switch).....56
- 14.2. rm (Perform Range Measure to upper Limit Switch).....57
- 14.3. vrm (Virtual Range Measure).....58
- 14.4. calmode (Closed Loop/Calibration Behavior).....59
- 14.5. carequired (Calibration Required).....59
- 14.6. caltimeout (Calibration Timeout).....60
- 14.7. caliboffset (Calibration Offset).....61
- 14.8. rmooffset (Range Measure Position Offset).....61
- 14.9. caldir (Calibration Direction).....61
- 14.10. calbspeed (Calibration Speed for Retraction).....62
- 14.11. calrefspeed (Reference Signal Calibration Speed).....62
- 14.12. encrefvel (Encoder Ref Signal Calibration Velocity).....63
- 14.13. refdir (Direction for Searching Reference Signal).....63
- 14.14. calpos (Calibration Position).....64



14.15.	calvel (Calibration Velocities for CAL Instruction)	65
14.16.	rmvel (Range Measure Velocities for RM Instruction)	66
14.17.	autopitch (Measure Pitch after CAL Instruction)	66
<b>15.</b>	<b>Move Instructions</b>	<b>67</b>
15.1.	moa (Move Absolute)	68
15.2.	mor (Move Relative)	68
15.3.	m (Move Relative Shortcut)	69
15.4.	distance (Distance for m)	69
15.5.	moc (Move to Center)	69
15.6.	go (Go To Position)	70
15.7.	speed (Speed Move)	70
15.8.	a (Abort the Current Move)	71
15.9.	delay (Set the Delay Time for Consecutive Moves)	71
15.10.	pause (Set the Pause after Position Reached)	71
15.11.	pos (Read or Set Position)	72
15.12.	posclr (Clear Position Offset)	72
15.13.	zero (Set Internal Position to Zero)	73
15.14.	clearpos (Set Internal Position to Zero)	73
<b>16.</b>	<b>HDI: Joystick, Tackball and Handwheel Instructions</b>	<b>74</b>
16.1.	joy (Generally Enable/Disable HDI)	74
16.2.	joydir (Joystick Direction or Assign Joystick per axis)	75
16.3.	joychangeaxis (Change Joystick X and Y Axis)	75
16.4.	joywindow (Joystick Window)	76
16.5.	joyvel (Joystick Velocity)	76
16.6.	joyspeed (Joystick Speed Presets for BPZ Device)	76
16.7.	keymode (Joystick Key Mode)	77
16.8.	keyspeed (Joystick Key Speed Presets)	78
16.9.	joycurve (Joystick Characteristic)	78
16.10.	key (Read HDI Device Key State)	79
16.11.	keyl (Read HDI Device Latched Key State)	79
16.12.	hwfactor (Handwheel Transmission Factor)	80
16.13.	hwfactorb (Alternate Handwheel Factor)	80
16.14.	hwfilter (Handwheel Noise Filter)	80
16.15.	tbfactor (Trackball Factor)	81
16.16.	zwheel (Is Z-Wheel Available)	82
16.17.	zwtravel (Z-Wheel Travel per wheel revolution)	82
16.18.	zwaxis (Z-Wheel Axis)	83
16.19.	zwfactor (Z-Wheel Factor)	83
16.20.	tvrjoy (Pulse and Direction Joystick Functionality)	84
16.21.	tvrjoyf (Pulse and Direction Joystick Factor)	84
16.22.	hdi (Read HDI ID)	85
16.23.	hdimode (HDI Mode Options)	86
16.24.	configaxsel (Joystick Axis Select Option)	87
<b>17.</b>	<b>Digital and Analogue I/O</b>	<b>88</b>
17.1.	digin (Digital Input)	88
17.2.	digout (Digital Output)	88
17.3.	adigin (AUX-I/O Digital Input)	89
17.4.	adigout (AUX-I/O Digital Output)	89
17.5.	anain (Analogue Input)	90
17.6.	anaout (Analogue Output)	91
17.7.	stoppol (Mode and Polarity of Stop Input Signal)	92



- 17.8. stop (Release, Force or Check Stop Condition) ..... 93
- 17.9. shutter (Shutter Out Signal of AUX-IO) ..... 93
- 17.10. flash (Defined Pulse at AUX-IO Takt Out) ..... 93
- 18. Encoder Instructions ..... 94**
- 18.1. encmask (Encoder Mask) ..... 94
- 18.2. enc (Encoder Active) ..... 95
- 18.3. encperiod (Encoder Signal Period) ..... 95
- 18.4. encdir (Encoder Counting Direction) ..... 96
- 18.5. encvel (Encoder Auto-Ajust Velocity) ..... 96
- 18.6. encctl (Encoder has TTL Signal) ..... 97
- 18.7. encref (Use Encoder Reference Signal) ..... 97
- 18.8. encnas (Use Encoder NAS Error Signal) ..... 98
- 18.9. encrefstatus (Encoder REF Signal State) ..... 98
- 18.10. encrefstatusl (Latched Encoder REF Signal State) ..... 98
- 18.11. encnasstatus (Encoder NAS Error Signal State) ..... 99
- 18.12. encerr (Encoder Error State) ..... 99
- 18.13. encamp (Encoder Signal Amplitude) ..... 99
- 18.14. encpos (Encoder Position) ..... 100
- 18.15. hwcount (Hardware Counter) ..... 100
- 18.16. clearhwcount (Clear Hardware Counter) ..... 100
- 19. MR Encoder Instructions ..... 101**
- 19.1. mra (MR Amplitude Correction Factor) ..... 101
- 19.2. mro (MR Offset Correction Value) ..... 101
- 19.3. mrp (MR Signal Peak-To-Peak Measuring Result) ..... 102
- 19.4. mrt (MR Signal Level) ..... 102
- 20. Closed Loop Instructions ..... 103**
- 20.1. ctr (Control Enable) ..... 103
- 20.2. ctrf (Control Factor) ..... 104
- 20.3. ctrff (Extended Control Factor) ..... 104
- 20.4. ctrc (Control Call) ..... 105
- 20.5. ctrd (Control Target Window Delay) ..... 105
- 20.6. ctrt (Control Timeout) ..... 105
- 20.7. twi (Target Window) ..... 106
- 20.8. ctrsm (Control behavior outside Lock-in Range) ..... 106
- 20.9. ctrs (Control Lock-in Range) ..... 107
- 20.10. ctrstatus (Control Status) ..... 108
- 20.11. ctrdiff (Control Position Difference) ..... 109
- 21. Trigger Output Functionality (option) ..... 110**
- 21.1. trig (Trigger) ..... 110
- 21.2. triga (Trigger Axis) ..... 110
- 21.3. trigm (Trigger Mode) ..... 111
- 21.4. trigo (Trigger Output) ..... 112
- 21.5. trigs (Trigger Signal Length) ..... 113
- 21.6. trigd (Trigger Distance) ..... 113
- 21.7. trigcomp (Trigger Compensation) ..... 113
- 21.8. trigenc (Trigger on Encoder) ..... 114
- 21.9. trigf (Trigger Frequency) ..... 114
- 21.10. trigbdelay (Precise Trigger Delay for second output) ..... 115
- 21.11. trigbwidth (Precise Signal Width for second output) ..... 115
- 21.12. trigbf (Precise Trigger Frequency for second output) ..... 115
- 21.13. trigcount (Trigger Counter) ..... 116



- 21.14. trigger (Force Trigger Signal) ..... 116
- 22. Snapshot -Trigger Input Functionality (option)..... 117**
- 22.1. sns (Snapshot enable/disable)..... 118
- 22.2. snsl (Snapshot Level / Polarity) ..... 118
- 22.3. snsf (Snapshot Filter)..... 118
- 22.4. snsm (Snapshot Mode)..... 119
- 22.5. Snapshot Mode Description and Examples ..... 120
- 22.6. snsc (Snapshot Counter) ..... 121
- 22.7. snsi (Snapshot Index) ..... 121
- 22.8. snsaxis (Snapshot Axis) ..... 122
- 22.9. snsp (Snapshot Position)..... 123
- 22.10. snsas (Snapshot Array)..... 123
- 22.11. snse (Snapshot Event) ..... 124
- 22.12. sns (Snapshot Voltage) ..... 124
- 22.13. prehome (Snapshot PreHome Position) ..... 125
- 22.14. home (Snapshot Home Position)..... 125
- 23. Document Revision History ..... 126**

## 2. Introduction

### Axes :

Tango controllers provide up to 4 axes. The axis specifiers used in the Tango instruction set are ASCII x, y, z, and a. Axes can be addressed separately by using the axis specifier or combined if no axis is specified in the instruction.

### Instruction syntax:

All Tango controllers communicate via a standard serial COM port interface, independent of the controller type (RS232C, USB, PCI, PCI-E). The instructions and parameters are sent as cleartext ASCII strings with a terminating carriage return [CR], which is 0x0d hex. Characters may be upper-, lower- or camel-case. The parameters are separated by a space character.

This provides easy access to all functions by using a simple terminal program such as HyperTerminal. A typical instruction syntax is as follows:

```
[!,?][instruction][SP][optional axis] [parameter1][SP][parameter2] [etc...] [CR]
```

**[!,?]** Read/write specifier, required by most instructions:

! (exclamation mark) = to write parameter, execute an instruction etc.

? (question mark) = to read data (returns settings, or status, etc.)

**[instruction]** Is the instruction word itself.

**[SP]** Space (ASCII 0x20 hex) as separation.

**[optional axis]** Axis character x, y, z or a if only one axis must be addressed.

**[parameter]** Usually integer or floating point numbers, floating point uses decimal point, no comma.

**[CR]** Termination (ASCII 0x0d hex), causes instruction execution.

A read instruction may return more than one parameter. In many cases the number of returned parameters depends on the amount of available axes:

```
[axis X] [if available: axis Y] [if available: axis Z] [if available: axis A]
```

For some instructions that return fractional numbers (e.g. ?pitch, ?gear, ?vel, ?encperiod and more) the number of returned fractional digits can be specified in order to increase resolution when reading back the value. For '?pos' and similar position returning functions (e.g. 'lim'), the number of fractional digits can be set by the '**resolution**' parameter.

### Syntax examples:

```
!vel 10 1.5      set velocities for the first two axes
!cal            command all axes to perform a calibration move
!moa y 10.1     move y axis to absolute position 10.1
?pos           returns position of all axes (e.g. 0.0000 0.0000 0.0000)
?vel x         returns velocity setting of X axis only (e.g. 10.000)
```

### Moves :

Move instructions are executed as a vector move. If several axes are started with one instruction they will reach their destinations at the same time. This means that - depending on velocity, acceleration and travel distance - one leading axis travels at its full velocity while the others follow synchronously. To move axes independently with their individual velocities, they have to be started separately by using single axis instructions. Please refer to the "move" instruction descriptions.

**Settings:**

Most settings can be stored permanently in the Tango Controller, so they are available from power on. When stored once, this reduces initialization overhead of the application software. Refer to the "save" instruction for further information. Parameters that are saved can be identified by a 'Y' in the Save column of the **brief instruction set description** later in this document.

**Character limits:**

To prevent the input buffer from overflow, please do not send more than 255 characters at once.

Such may occur when sending the setup sequence to the Tango controller. A good practice is to request the "?err" state after each setup instruction. This will return the information if the parameters were accepted or not while preventing overflow.

Another solution is to activate the "!cts" handshake (available only with Desktop RS232C and some USB versions). This will automatically halt the PC transmission for as long as the input buffer is full. The PC COM port then must be opened with hardware handshake on, as well. Please refer to the "!cts" instruction description.

**Important: Security speed limitation!**

The Tango controllers have a built in security function, which reduces the maximum travel velocity to a secure 10mm/s for as long as no initial "cal" and "rm" moves have been executed. This is to safeguard the driven axis against damage that could be caused by moving fast into its end positions. After calibrating the axis into its endswitches (cal and/or rm if switches are mounted and enabled) the travel velocity is no longer limited.

If it is not wanted or impossible to do a calibration and range measure move on each power on:

- A) The speed limit may be increased to up to 100mm/s at own risk. Please refer to the "secvel" instruction for further information.
- B) The !rm can be skipped by instead using !vrm (please read remarks).
- C) Non existent limit switches can be deactivated (swact) and then do not require a !rm. In such case secvel will be released after !cal.

**Important: Measuring units!**

The measuring unit is set by the "dim" instruction, where dim 2 [mm] is the default value.

In all dim settings, except of dim 9, the velocity is motor revolutions per second! Only dim 9 provides the millimeter unit for most parameters<sup>1</sup>, positions and velocities.

**Extended mode:**

In addition to the improvements when using dim 9 units, there is an option to enable extended mode behavior. It enables more functionality, like separate calibration, rm and joystick velocities, which else are the same as the axis velocity (vel). Please refer to the "extmode" instruction for further details.

---

<sup>1</sup> Only 'calbspeed' and 'calrefspeed' are always in 1/100 turns/sec, even in dim mode 9



### 3. Hint for controller initialization

Please make sure that first of all the following parameters have to be set:

- The axis units (here called "**dim**")
- If the controller firmware version is 1.32 or above: the **extmode**
- The axis **pitch** and - if used - also the **gear**, which are always in [mm] independent of dim

Using *dim=9* and *extmode=1* will turn all (even the vel and joyvel) units to linear axis related [mm] and [mm/s]. Extmode=1 also offers bugfixes, more features and flexibility. But it has a slightly different behavior. Please refer to the **Extended Mode** description in this document.

## 4. Brief Description of the Tango Instruction Set

<b>Controller Informations</b>					
Instruction	Example	Save	Brief description	Page	
(?)	version	version	-	Read detailed firmware and controller version	17
(?)	det	det	-	Read the controller configuration	18
(?)	detext	detext	-	Read the controller configuration with description	18
?	readsn	?readsn	-	Read the controller serial number	19
(?)	ver	ver	-	Read default version number	19
(?)	iver	iver	-	Read further version number information	19
(?)	uptime	uptime	-	Read how long the controller is running	20
(?)	temp	temp	-	Read case temperature (avail. depends on controller type)	20
?	maxaxis	?maxaxis	-	Read number of available axes	20
?	maxcur	?maxcur	-	Show the maximum possible motor currents of all axes	20
?	etspresent	?etspresent	-	Check availability of ETS	21
?	stagesn	?stagesn	-	Read the axis serial numbers from ETS	22

<b>Communication Interface Settings</b>					
Instruction	Example	Save	Brief description	Page	
? !	baud	!baud 9600	Y	Set RS232 baud rate to 9600 Bd (default=57600)	23
? !	cts	!cts 1	Y	Enable CTS hardware handshake	23

<b>System Instructions</b>					
Instruction	Example	Save	Brief description	Page	
(!)	save	save	-	Save parameters to controller nonvolatile memory	24
(!)	restore	restore	-	Reload saved controller parameters from n.v. memory	24
(!)	reset	reset	-	Reset controller (forces restart, similar to cycle power)	25
!	pa	!pa 1	-	Enable power amplifiers (disable = 0), see also 'axis' instr.	25
? !	ipreter	!ipreter 2	Y	Select optional Venus instruction set	26

<b>Operating Modes</b>					
Instruction	Example	Save	Brief description	Page	
? !	extmode	!extmode 1	Y	Enable extended controller behavior	27
? !	scanmode	!scanmode 1	Y	Set positioning behavior to scanmode	28
? !	scanvel	!scanvel 0.5	Y	Set scanmode vector velocity to 0.5 mm/s	28
? !	modulomode	!modulomode a 1	Y	Set positioning behavior of A axis to turntable mode 1	29

<b>Controller States and Error Messages</b>					
Instruction	Example	Save	Brief description	Page	
? !	autostatus	!autostatus 0	-	Select autostatus response type 0 (=disabled), range: [0-4]	30
(?)	statusaxis	statusaxis	-	Read axis state [@,M,J,C,S,A,D,-]	31
(?)	status	status	-	Read controller error state	31
(?)	err	err	-	Read error number	32
(?)	help	help	-	Read error number with additional description text	32
(?)	service	service	-	Returns a detailed parameter and state list, for debugging	32
(?)	pci	pci	-	Returns 1 if controller is plugged in a PCI slot (desktop=0)	33
(?)	isvel	?isvel x	-	Read actual velocity of the X axis	33
?	maxpos	?maxpos x	-	Read maximum available position range for X axis	33

<b>General Adjustments</b>				
Instruction	Example	Save	Brief description	Page



### General Adjustments

Instruction	Example	Save	Brief description	Page
? ! dim	!dim 1 1 1	Y	Set position units of X Y Z to $\mu\text{m}$	34
? ! pitch	!pitch 1 1 1	Y	Set spindle pitch of X Y Z to 1 [mm/revolution]	35
? ! gear	!gear 1 1 1	Y	Set gear factor of X Y Z to 1	35
? ! motorsteps	!motorsteps x 200	Y	Set X axis motor has 200 steps per revolution	36
? ! accel	!accel 0.1 0.1 0.1	Y	Set acceleration of X Y Z to $0.1\text{m/s}^2$	37
? ! accelfunc	!accelfunc 1 1 0	Y	Set acceleration function X and Y to $\sin^2$ , Z to linear	37
? ! stopaccel	!stopaccel 2 2	Y	Set X and Y deceleration during stop condition to $2\text{m/s}^2$	38
? ! vel	!vel 10 10 10	Y	Adjust speed of X Y Z to 10 [revolutions/s]	39
? ! velfac	!velfac 1 1 1	Y	Set velocity reduction factor for X Y Z to 1 (= no reduction), range is [0.01-1]	40
? ! secvel	!secvel x 20	Y	Set secure speed limit X to 20mm/s (unit is always mm/s)	40
? ! cur	!cur 0.5 0.6 1	Y	Set motor current in Ampere: X=0.5 Y=0.6 and Z=1 A	41
? ! reduction	!reduction 0.5 0.5 0.5	Y	Select 50% motor current reduction for X Y Z	42
? ! curdelay	!curdelay 1000	Y	Delay X axis motor current reduction by 1000 [ms]	42
? ! ecomove	!ecomove 30 30 0	Y	Set eco-level of X and Y to 30% current saving	43
? ! axis	!axis 1 0 -1	Y	Enable X, disable Y and switch off Z axis	44
? ! axisdir	!axisdir 0 1 0	Y	Reverse rotating direction of Y motor (caution!)	44
? ! motortable	!motortable x 2	Y	Select custom motor correction table type 2 for X axis	45
? ! usteps	!usteps 50000	Y	Set dim 0 microsteps to 50000/rev for all axes	45
? ! resolution	!resolution 6	Y	Set position return string resolution to 1 nm	46
? ! backlash	!backlash 12.3 0 0	Y	Set backlash compensation to $12.3\mu\text{m}$ in X and 0 in Y & Z	46
! ? lock	!lock 2 1	Y	Set write protection for parameter 2 (here: motor current)	47
! ? lockaxis	!lockaxis 0 0 0 0	Y	Remove lock protection from all axes (lock has no effect)	47
? ? lockstate	?lockstate x	-	Read extended locked parameters, including internal limitations currently applied to X axis	48
? ! stout	!stout 2	Y	Make Status-LED state available at AUX-I/O Pin VR_OUT	48
? ! noled	!noled 1	Y	Switch Status-LED permanently off	49
? ! updelay	!updelay -5000	Y	Wait to a maximum of 5 seconds for valid external power	49

### Limit Switch Instructions (Hardware and Software)

Instruction	Example	Save	Brief description	Page
? ! lim	!lim 0 10 0 10 0 10	-	Set lower position limit to 0 and upper limit to 10 (assume unit is [mm] if dim was set to 2) for X Y Z	50
? ! limctr	!limctr x 1	-	Enable hardware limit switches for X axis, default = 1	50
? ! nosetlimit	!nosetlimit 1 1 1 1	Y	Disable setting/overwriting of software limits during cal and rm for all axes (here: X Y Z A), default = 0	51
? ! swtyp	!swtyp 1 0 1 !swtyp y 0 0 0	Y	Set limit switch type for all axes to NPN (pull-up) Set limit switch type for Y to PNP (pull-down)	51
? ! swpol	!swpol 1 0 1 !swpol z 1 0 1	Y	Set polarity of limit switches for all axes to active high (=1) Set polarity of limit switches for Z to active high	52
? ! swact	!swact 1 0 1 !swact y 1 0 0	Y	Enable cal and rm limit switches for all axes Enable cal limit switch for Y, disable ref and rm	53
? ! swdir	!swdir x 1	Y	Swap CAL and RM endswitch assignment of X axis	52
? ? readsw	?readsw	-	Read states of all limit switches (1=active and actuated)	54
(?) ? swin	swin	-	Read TTL signal level of all limit switch inputs (1=high)	54
(?) ? statuslimit	statuslimit	-	Read current limit status „A“ = calibration done „D“ = rm done „L“ = limit switch modified by software „-“ = not yet modified	55

### Calibration and Range Measure Instructions



Instruction	Example	Save	Brief description	Page
(!) cal	cal	-	Perform a calibration move for all enabled axes, see 'axis'	56
(!) rm	rm x	-	Perform a range measure move in X	57
(!) vrm	vrm 75 50	-	Virtual range measure for a 75x50 microscope stage	58
? ! calmode	!calmode 2 2	Y	Set calibration/closed loop behavior X, Y to type 2	59
? ! calrequired	!calrequired z 1	Y	Z axis does not move until !cal is executed	59
? ! caltimeout	!caltimeout 60 60 10	Y	Set calibration timeout for X and Y to 1 minute, Z to 10s	60
? ! caliboffset	!caliboffset 1 1 1	Y	Set the cal zero-point 1mm aside lower limit switch (dim 2)	61
? ! rmooffset	!rmooffset 1 1 1	Y	Set rm end-position 1mm aside upper limit switch (dim 2)	61
? ! caldir	!caldir z 1	Y	Calibrate the Z-axis in positive direction	61
? ! calbspeed	!calbspeed 20	Y	Set the speed for move out of 'cal' and 'rm' limit switches for all axes to 0.2 [revolutions/s], range is [1...100]	62
? ! calrefspeed	!calrefspeed 10	Y	Set the speed for calibrating to the encoder reference for all axes to 0.1 [revolutions/s], range is [1...100]	62
? ! encrefvel	!encrefvel 5 5 5	Y	Set the speed for calibrating to the encoder reference for axes X,Y,Z to 5 [rev/s] if dim=2, [mm/s] if dim=9	63
? ! calpos	calpos	-	Read back the encoder position where the calibration switch was released	64
? ! refdir	?refdir y	Y	Read the direction for encoder reference search in Y axis	63
? ! calvel	!calvel x 10 0.5	Y	<b>Only if extmode = 1:</b> Set calibration velocities in X	65
? ! rmvel	!rmvel x 10 0.5	Y	<b>Only if extmode = 1:</b> Set range measure velocities in X	66
? ! autopitch	!autopitch x 1	Y	Measure pitch after cal move of X axis	66

### Move Instructions

Instruction	Example	Save	Brief description	Page
(!) moa	moa 10 10 10 moa y 20	-	Move X Y Z absolute to positions 10 10 10 Move Y axis to position 20 (unit depends on dim setting)	68
(!) mor	mor 4 4 4 mor y -10.5	-	Move X Y Z relative by 4 (unit depends on dim setting) Move Y axis relative 10.5 backwards	68
(!) m	m	-	Move relative again (use same parameters as defined by last '!mor' or '!distance' instruction)	69
? ! distance	!distance 1 1 1	-	Set distance for X Y Z 'm'-move (start with 'm' or '!m')	69
(!) moc	moc x	-	Move X to center position between lower and upper limit switch, or between lower and upper software limits	69
(!) go	go x 12.5	-	Move X to pos. 12.5, overwriteable, for tracking applications	70
? ! speed	!speed 5 5 5 !speed y 0	-	Let X Y Z axis travel at 5 [revolutions/s] Stop the Y axis speed move	70
(!) a	a	-	Abort move (Stop)	71
? ! delay	!delay 1000	Y	Delay the start of move instructions by 1000 ms	71
? ! pause	!pause 10	Y	Delay "position reached" autostatus response by 10 ms	71
? ! pos	!pos 0 0 1.5 ?pos z	-	Set current X Y positions to 0 and Z position to 1.5 Read current Z position	72
(!) posclr	!posclr x	-	Reset the position offset that was manipulated by !pos	72
(!) zero	!zero z	-	Set Z position and internal counter to 0 (e.g. filter wheel application)	73
(!) clearpos	!clearpos z	-	Set Z position and internal counter to 0 (e.g. filter wheel application), not executable with measuring system	73

### HDI: Joystick, Tackball and Handwheel Instructions

Instruction	Example	Save	Brief description	Page
? ! joy	!joy 0 !joy 2	Y	Switch joystick ON=2 or OFF=0 (Switching off may cause an autostatus response)	74
? ! joydir	!joydir 2 -2 0	Y	Set HDI axis direction: Y reversed, Z disabled	75
? ! joychangeaxis	!joychangeaxis 1	Y	Change Joystick X and Y axis (X→Y, Y→X)	75
? ! joywindow	!joywindow 14	Y	Set idle window of the joystick center position, where a joystick deflection has no effect [0..100]	75

**HDI: Joystick, Tackball and Handwheel Instructions**

Instruction	Example	Save	Brief description	Page	
? !	joyvel	!joyvel z 1.5	Y	<b>Only if extmode = 1:</b> Set joystick velocity for Z to 1.5	76
?(!)	joyspeed	joyspeed 2 25	Y	Set joystick speed for speed button 2 "medium" to 25 rev/s	76
? !	keymode	!keymode 2	Y	Select joystick key mode 2 = high speed preselection	77
? !	keyspeed	!keyspeed x 5 20	Y	Set keymode joystick speed X low=5mm/s, high=20mm/s	78
?(!)	joycurve	!joycurve z 1	Y	Set joystick characteristic for Z ot linear	78
(?)	key	key	-	Read state of all joystick buttons (0=released, 1=pressed)	79
(?)	keyl	keyl	-	Read and clear latched state of all joystick buttons	79
? !	hwfactor	!hwfactor x 1	Y	One handwheel revolution in X is 1mm stage travel	80
? !	hwfactorb	!hwfactorb x 14	Y	One handwheel revolution in X is 14mm stage travel	80
? !	hwfilter	!hwfilter 0	Y	Switch off handwheel noise reduction	80
? !	tbfactor	!tbfactor 1 1	Y	Set trackball transmission factor in X and Y to default	81
(?)	zwheel	?zwheel	-	Returns 1 if HDI device has a Z-Wheel attached	82
?(!)	zwtravel	!zwtravel 1 0.25	Y	Set default Z-Wheel travel to 2.5 mm/rev	82
? !	zwaxis	!zwaxis a	Y	Assign Z-Wheel to A-axis	83
? !	zwfactor	!zwfactor 1	Y	Set Z-Wheel factor to 1:1 (default for Tango HDI devices)	83
? !	tvrjoy	!tvrjoy z	Y	Assign AUX-IO pulse&direction joystick to Z axis	84
? !	tvrjoyf	!tvrjoyf 1	Y	Set tvrjoy transmission factor to 1	84
(?)	hdi	hdi	-	Read ID number of the connected HDI device	85
! ?	hdimode	!hdimode 0 1	Y	Set hdimode bit 0 to 1 for ErgoDrive Toggle Mode	86
! ?	configaxsel	!configaxsel	Y	Toggle joystick Z-axis between axes Z and A by F4 key	87

**Digital and Analogue I/O**

Instruction	Example	Save	Brief description	Page	
(?)	digin	digin digin 8	-	I/O Extension board: Read all digital inputs I/O Extension board: Read digital input 8	88
? !	digout	!digout 5 1 ?digout	-	I/O Extension board: Set digital output 5 to logic level 1 I/O Extension board: Read back all digital output levels	88
(?)	adigin	adigin adigin 2	-	Read all AUX-I/O digital inputs Read logic level of AUX-I/O digital input 2 only	89
? !	adigout	!adigout 3 1 ?adigout	-	Set AUX-I/O digital output 3 to logic level 1 Read back all digital output levels	89
(?)	anain	anain c 2	-	Read input of analogue channel 2	90
? !	anaout	!anaout c 1 17.5	-	Set analogue voltage of channel 1 to 17.5 percent (1.75V)	90
? !	stoppol	!stoppol 1	Y	Set AUX-IO stop input to active high	92
!	stop	!stop 0	-	Release stop condition (in latched stoppol modes 4 or 5)	93
? !	shutter	!shutter 1	-	Set AUX-IO shutter out signal to TTL high	93
(!)	flash	flash 0.1	-	Send a 100µs high pulse to AUX-IO TAKT_OUT (LED)	93

**Encoder Instructions**

Instruction	Example	Save	Brief description	Page	
? !	encmask	!encmask 1 1 0	Y	Enable activation of X and Y encoders, disable Z	94
? !	enc	!enc 1 0	-	Manually activate X encoder (caution!), set Y to inactive	95
? !	encperiod	!encperiod 0.1	Y	Set signal period of X encoder to 100 µm	95
? !	encttl	!encttl x 1	Y	X encoder is TTL type (has no analogue sin/cos signal)	97
? !	encdir	!encdir y 1	(Y)	Reverse counting direction for Y encoder	96
? !	encvel	!encvel x 0.5	Y	Set auto-adjust velocity of X encoder to 0.5mm/s	96
? !	encref	!encref 0	Y	Disable usage of X encoder reference signal	97
? !	encnas	!encnas 1 0 0	Y	Enable NAS error signal input encoding for X encoder only	98
(?)	encrefstatus	encrefstatus x	-	Read X encoder reference signal state (1=on reference)	98
(?)	encrefstatusl	encrefstatusl x	-	Read latched X encoder reference signal state	98
(?)	encnasstatus	encnasstatus x	-	Read X encoder NAS signal state (1=NAS error)	99
? !	encerr	!encerr 0	-	Clear encoder error state for X axis (? response is 0 or e)	99

**Encoder Instructions**

Instruction	Example	Save	Brief description	Page
? ! encamp	?encamp x	-	Read X encoder signal amplitude in percent	99
? ! encpos	!encpos 1	-	?pos instruction returns for the encoder positions, if enc=1	100
(?) hwcount	hwcount	-	Read all encoder positions (TTL counter, no interpolation)	100
(!) clearhwcount	clearhwcount x	-	Set X axis hwcount to zero	100

**MR Encoder Instructions**

Instruction	Example	Save	Brief description	Page
? ! mra	?mra x	-	Read amplitude correction factor (sin/cos ratio) of X	101
? ! mro	?mro	-	Read offset correction value for all encoders	101
? ! mrp	!mrp x 0 0 0 0	-	Reset MR-signal peak-to-peak measurement result of X	102
? mrt	?mrt z 2	-	List two measurement results of the Z input signals	102

**Closed Loop Instructions**

Instruction	Example	Save	Brief description	Page
? ! ctr	!ctr 1 1 1	Y	Set closed loop circuit X Y Z to "active until reached" mode	103
? ! ctrf	!ctrf 2.0	Y	Closed loop factor for X axis is set to 2.0	104
? ! ctrff	!ctrff 2 3.5	Y	Closed loop factors for X axis are set to 2 and 3.5	104
? ! ctrc	!ctrc 3	Y	Closed loop control is called every 3 millisecond	105
? ! ctrd	!ctrd 100	Y	Closed loop in target window for 100 milliseconds	105
? ! ctrt	!ctrt 200	Y	Closed loop control timeout after 200 milliseconds	105
? ! twi	!twi 0.01 0.01 0.01	Y	Set target window for X Y Z to 10µm (assume dim=2)	106
? ! ctrsm	!ctrsm x 1	Y	Set X behavior outside lock-in range to "slow closed loop"	106
? ! ctrs	!ctrs x 0.2	Y	Set lock-in range for X to 0.2mm (assume dim=2)	107
? ctrstatus	?ctrstatus 1	-	Get Closed Loop active state of all axes	108
? ctrdiff	?ctrdiff	-	Get Closed Loop position difference of all axes	109

**Trigger Output Functionality<sup>1</sup>**

Instruction	Example	Save	Brief description	Page
? ! trig	!trig 1	-	Enable trigger functionality (should be the last instruction)	110
? ! triga	!triga x	Y	Trigger function is related to X axis	110
? ! trigm	!trigm 0	Y	Select trigger mode 0	111
! ? trigo	!trigo 1	Y	Select the default trigger output	112
? ! trigs	!trigs 40	Y	Set trigger output signal length to 40 microseconds	113
? ! trigd	!trigd 10	Y	Set trigger distance to 10 (mm if dim=2)	112
? ! trigcomp	!trigcomp 50	Y	Compensate a trigger signal delay of 50µs	113
? ! trigenc	!trigenc 1	Y	Select encoder signal as trigger source (if available)	114
? ! trigf	!trigf 1000	Y	Generate periodic trigger pulses with 1kHz	114
? ! trigbwidth	!trigbwidth 4.35	Y	Set Precise secondary trigger out signal length to 4.35µs	115
? ! trigbdelay	!trigbdelay 15.05	Y	Set Precise secondary trigger out to 15.05µs delayed	115
? ! trigbf	!trigbf 66000000	Y	Set Precise secondary trigger out frequency to 66 MHz	115
? ! trigcount	?trigcount	-	Read number of generated trigger events	116
(!) trigger	trigger	-	Manually set trigger output (available in trigm 102, 103)	116

**Snapshot - Trigger Input Functionality<sup>1</sup>**

Instruction	Example	Save	Brief description	Page
? ! sns	!sns 1	-	Enable snapshot functionality (always 1 after power up)	118
? ! sns1	!sns1 0	Y	Set snapshot input signal to active low	118
? ! snsf	!snsf 10	Y	Set snapshot signal debounce filter to 10 milliseconds	118

<sup>1</sup> Function has to be enabled by factory, it is not available per default.

**Snapshot - Trigger Input Functionality<sup>1</sup>**

Instruction	Example	Save	Brief description	Page
? ! snsm	!snsm 0	Y	Set snapshot mode to 0 (0=capture pos, 1=move to pos)	119
? ! snsc	?snsc	-	Read number of snapshot events (=array fill size)	121
? ! snsi	!snsi 5	-	Set snapshot index to 6 <sup>th</sup> entry (snsa 6)	121
? ! snsaxis	!snsaxis 1 1 0 0	-	X and Y axis moves wait for snapshot (in snsm mode 6)	122
? ! snsp	?snsp x	-	Read last captured X position	123
? ! snsas	?snsas 1	-	Read first position entry of snapshot array (all axes)	123
(!) snse	snse 2	-	Generate SnapShot event F2	124
? ! snsv	?snsv 3	-	Read captured ANIN0 voltage in mV of 3 <sup>rd</sup> snapshot entry	124
? ! prehome	!prehome 10 20 1	-	Set prehome positions X Y Z to 10 20 1 (unit depends on dim setting)	125
? ! home	!home 5 5 0	-	Set home positions X Y Z to 5 5 0 (unit depends on dim setting)	125

## 5. Instruction Syntax Description

Most instructions work in both directions (reading and writing). (?)! means the instruction accepts write and read access. The controller identifies a read instruction by a preceding '?', while '!' indicates writing to a parameter or executing an instruction. More information can be found in the **Introduction** chapter of this document.

Some examples of legal instruction syntax:

```
!Instruction parameter1 parameter2 parameter3 parameter4
!Instruction parameter1 parameter2
!Instruction axis parameter
!Instruction
?Instruction axis parameter
?Instruction
```

## 6. Error Numbers and their possible Root Cause

```
0 no error
1 no valid axis name
2 no executable instruction
3 too many characters in command line
4 invalid instruction
5 number is not inside allowed range
6 wrong number of parameters
7 either ! or ? is missing
8 no TVR possible, while axis active
9 no ON or OFF of axis possible, while TVR active
10 function not configured
11 no move instruction possible, while joystick enabled
12 limit switch active
13 function not executable, because encoder detected
21 multiple axis moves are forbidden (e.g. during initialization)
22 automatic or manual move is not allowed (e.g. door open or initialization)
27 emergency STOP is active
29 servo amplifier are disabled (switched OFF)
30 safety circuit out of order

70 wrong CPLD data
71 ETS error
72 parameter is write protected (check lock bits)
73 internal error, e.g. eeprom data corruption
74 closed loop switched off due to parameter change
75 could not enable axis correction, or axis correction was disabled
76 io extension card error
```





## 7. Controller Informations

You may read the firmware version by sending the instruction `'version'` to the controller. The instruction `'det'` gives you further details of which options and features are enabled. Each controller has its own unique serial number readable with the instruction `'readsn'`.

### 7.1. version (Read detailed Version information)

Syntax:                    ?version or version  
Parameter:                none or 1

Description:             This instruction gives detailed information about the firmware version.

Sending the version instruction with parameter 1 returns the tango firmware version number only.

Example:                    ?version  
TANGO-DT-S, Version 1.37, Aug 12 2008 , 16:39:01

                              ?version 1  
1.37

Response syntax:        Character string including controller type, firmware version and build date separated by a comma:

TANGO	Fixed string identifying the Tango controller
-DT	Desktop version
-PCI	PCI card version
-S	Tango short card version (PCI-S, DT-S)
-MINI	TANGOmini
-C	Motorized Stage with integrated Controller
e	Tango PCI-E card version (PCIE, DTe)
Version 1.37	Firmware version number
Aug 12 2008	Firmware build date
16:39:01	Firmware build time

## 7.2. det (Read detailed Configuration)

Syntax:               ?det or det  
Parameter:           none

Description:         Read Detailed information of the controller configuration.

Response:            The response is a decimal integer number. Its bit pattern represents the configuration as described below:

```
0x0 - - - 1    1Vpp encoder is configured
0x0 - - - 2    MR encoder is configured
0x0 - - - 4    TTL encoder is configured
0x0 - - 3 -    this is the number of configured axes (e.g. 3)
0x0 - 1 - -    Display is configured
0x0 - 2 - -    Speedpoti is configured
0x0 - 4 - -    Hand wheel is configured
0x0 - 8 - -    Snapshot is configured
0x0 1 - - -    TVRin is configured
0x0 2 - - -    Trigger out is configured
0x0 8 - - -    TVRout is configured
0x1 - - - -    digital I/O extension 24in+8out
0x2 - - - -    digital I/O extension 12in+8out
0x4 - - - -    Trackball is configured
0x8 - - - -    ETS available
```

Individual configured options can be identified by applying a logic AND mask to the returned value.

E.g. (val & 0x0800) to identify if Snapshot instructions are available/configured by factory, (val & 0x02000) for Trigger.

Example:             Assume the ?det response is 81697, which is 0x13F21 hex. This number means in detail, that the controller is configured for:

```
1 => Built in digital I/O extension with 24in + 8out
3 => TVRin and Trigger out
F => Display, Speedpoti, Hand wheel and Snapshot
2 => 2 axes
1 => 1Vpp encoder
```

## 7.3. detext (Read Extended detailed Configuration)

Syntax:               ?detext or detext  
Parameter:           none

Description:         Read Detailed information of the controller configuration as multi-line readable text (description strings). Each line is terminated by a [CR].

Response:            Multi line reply, containing the configuration number as hex value (bit representation as describet with 'det'), and the meaning of it as multi line text.

Example:  
detext           =>    02832  
                  = MR Encoder  
                  = 3 Axes  
                  = Snapshot  
                  = Trigger out

## 7.4. readsn (Read Serial Number)

Syntax: ?readsn or readsn  
Parameter: none

Description: Reads the serial number of the Tango controller.

Response: The controller returns its unique serial number as ASCII character string. The syntax is YYWWTNXXX.

YY year of manufacturing  
WW week of manufacturing  
T type identifier  
N in hardware available axes (may differ from ?maxaxis)  
XXX index number

Example: ?readsn => 083303007

## 7.5. ver (Read default Version Number)

Syntax: ?ver or ver  
Parameter: none

Description: Read the default controller version info. The first digit is the number of configured axes. The second digit is the maximum possible motor current in ampere.  
To read the Tango firmware version, please use "version".

Response syntax: Vers:LSnm.xx.xxx (in some cases "Vers:ESnm.xx.xxx")

"Vers:LS" Fixed character string (also "Vers:ES" possible)  
n Number of configured axes: 1, 2, 3, or 4  
m Maximum Current: 1=1.25A, 2=2.5A, 3=3.75A  
x Fixed numbers

Example: ?ver => Ver:LS32.00.038

## 7.6. iver (Read internal Version Number)

Syntax: ?iver or iver  
Parameter: none

Description: Read the internal version information string. Mostly unused.  
To read the Tango firmware version, please use "version".

Response syntax: 14 characters, e.g. T[DD].[WW].[YY]-[NNNN]  
[DD] = Day of Week  
[WW] = Week  
[YY] = Year  
[NNNN] = Number

Example: ?iver => T04.35.02-0004

## 7.7. uptime (Read Controller Up Time)

Syntax:           ?uptime or uptime  
Parameter:       none

Description:      This instruction returns the power-on time of the controller since it was switched on or resetted.

Response:        Time in seconds.

Example:          uptime           => 8731

## 7.8. temp (Read Case Temperature)

Syntax:           ?temp or temp  
Parameter:        none

Description:      Read the ambient temperature inside the controller.  
Remarks:         Not all Tango controllers provide a temperature sensor.

Response:        Temperature in [°C] with one decimal place.

Example:          temp             => 28.9

## 7.9. maxaxis (Read number of available Axes)

Syntax:           ?maxaxis  
Parameter:        none

Description:      Read the number of available (factory configured) axes.  
1, 2, 3 or 4.

Response:        Number of available axes.

Example:          ?maxaxis       => 3

## 7.10. maxcur (Read Maximum Motor Current)

Syntax:           ?maxcur  
Parameter:        x, y, z, a or none  
Optional parameter 1

Description:      Read the maximum possible motor current which can be set by  
**!cur**.

A) Maxcur called without parameter:  
Maximum motor current which the power amplifier can deliver.

B) Maxcur called with parameter "1":  
Additional soft motor current limitation, defined from ETS.

Response:        maximum motor current in Ampere [A] (e.g. "1.25" or "1.00")

Examples:

?maxcur y        read maximum adjustable motorcurrent of Y axis only  
?maxcur         read maximum adjustable motorcurrent of all available axes  
(e.g. returns 1.25 1.25 1.25 1.00)

?maxcur y 1     read motor current soft limit of Y axis only (e.g. set by ETS)  
?maxcur 1       read motor current soft limit of all axes       (e.g. set by ETS)



## 7.11. etspresent (Read ETS Detect State)

Syntax:               ?etspresent  
Parameter:           none

Description:         Check if an ETS was detected by the Tango (during power-up).

0 = No ETS found at the corresponding address  
1 = ETS was found at the corresponding address (is available)

Remarks:            If more than one ETS is found, the first one (lowest address) is treated as the "main" ETS, the ETS with higher address only for the there defined axis specific parameters (e.g. axis 3).

Response:            ASCII string of 4 characters, 0 or 1  
                      [ETS ADR0][ETS ADR1][ETS ADR2][ETS ADR3]

Example:             ?etspresent   => 0000   (no ETS connected)  
                      ?etspresent   => 1000   (ETS found at address 0, usual case)  
                      ?etspresent   => 1010   (ETS found at address 0 and 2)



## 7.12. stagesn (Read Connected Devices Serial Number)

Syntax: ?stagesn  
Parameter: none or -1,0,1,2,3

Description: Read the device serial number from the ETS.  
If the motor axis provides the ETS identification circuit, the serial number of the axis or connected unit (e.g. a microscope stage) is returned.

Call parameter options:

- None: 4 serial numbers are returned, one for each possible ETS address (0,1,2,3)
- -1 : 1 Serial number is returned, of the main ETS (usually also the only connected ETS, recommendet instruction)
- 0-3 : Addresses an individual ETS, usually only ETS#0 is Available (which then also responds with -1 param.)

Response: ASCII string(s) of 8 or 9 characters.  
The syntax is YYMMDDXXX or "-----" if no ETS available.

YY year of manufacturing  
MM month of manufacturing  
DD day of manufacturing  
XX(X) index number

```

Example: ?stagesn      => 120328015 -----
         ?stagesn -1 => 120328015
         ?stagesn 0  => 120328015
         ?stagesn 1  => -----

         ?stagesn -1 => 07090305

         ?stagesn      => ----- 120328015 -----
         ?stagesn -1  => 120328015
         ?stagesn 0   => -----
         ?stagesn 1   => 120328015

```

## 8. Communication Interface Settings

### 8.1. baud (Baud Rate)

Syntax:               !baud or ?baud  
Parameter:           1200, 2400, 480, 9600, 19200, 38400, 57600 or 115200

Description:         Set or read the baudrate of the serial COM Port interface.

It applies to devices with a true RS232 serial connection, as available with Tango-DT and Tango-mini, and the Pilot stage or Tango-mini with USB cable.

Remarks:            For PCI/PCI-E card versions or Tango-DT with USB interface this instruction has no effect, as communication is managed by the driver at a very high, internally fixed baudrate. In this case it does not matter which baudrate the virtual COM port is opened with, it has no effect on performance.

After sending this instruction the PC has to re-open the COM Port with the new baudrate, else no communication is possible. Then a "**!save**" instruction may be sent to permanently store the new baudrate in the controller.

Response:            Current baud rate of the controller

Examples:  
!baud 57600           Set the baud rate to 57600 [Bd]  
?baud                 Read currently set baud rate

### 8.2. cts (Enable/Disable RS232 Hardware Handshake)

Syntax:               ?cts or !cts  
Parameter:            0 or 1

Description:         Enable or disable RTS/CTS hardware handshake of the RS232 and USB Virtual COM Port interfaces. For PCI bus communication this instruction has no effect.

Remarks:            The COM port of the PC has to be opened in the same hardware handshake mode.

0 : no handshake (default)  
1 : RTS/CTS handshake

Response:            Currently selected cts mode

Examples:  
?cts                 read current handshake mode  
!cts 0               disable RTS/CTS handshake (default, recommended)  
!cts 1               enable RTS/CTS handshake

## 9. System Instructions

### 9.1. save (Save Parameters)

Syntax:               !save or save  
Parameter:           none

Description:         The save instruction permanently stores the parameter settings (e.g. spindle pitch, motor current) to the Tango controller. These parameters will be applied as default values after each consecutive power-on or reset. Executing a save comand always returns the "OK..." string when writing to the internal memory has completed successfully.

Response:            ASCII string "OK..." or "ERR"

Example:             save  
                      ==> OK... (The currently used controller parameters are saved  
                                  and from now on used as defaults)

### 9.2. restore (Restore Saved Parameters)

Syntax:               !restore, restore or ?restore  
Parameter:            none or -1

Description:         Reload or reset the saved parameters.

**When called without parameter (restore, !restore, ?restore)**

The controller reloads the parameter settings from its nonvolatile memory, like performed at power-on or reset. But restore does not affect or restart the entire hardware.

?restore returns "OK..." or "ERR" (like the save instruction), In the other cases there is no reply. ?err might be sent once after the restore instruction and will reply after restore has completed.

**When called with parameter "-1" (restore -1, !restore -1)**

The saved parameters (nonvolatile memory) will be deleted and set to the firmware defaults. The parameters then have to be checked and set to the hardware requirements by the user. Else it may cause damage (due to overcurrent, wrong pitch, limit switch types etc). Software reset is performed automatically.

There is no reply to the restore -1 instruction, ?err polling may be required as described with the software "reset".

Response:            none, or "OK..." or "ERR" when using ?restore

Example:             restore                   (reload the saved parameter set)  
                      ?restore                (like restore, but with OK/ERR reply)  
                      restore -1              (reset the saved parameters to default)



### 9.3. reset (Force a Software Reset)

Syntax:               !reset or reset  
Parameter:           none

Description:         The controller is forced to perform a software reset. It is a restart similar to power on. Rebooting from reset will take more than 1 second, where the controller is not responding. There is no reply to a software reset.

Remarks:            Wait for reboot after Reset:  
For knowing if the controller has rebooted and is ready, data may be polled until it responds again. E.g. send "**?err**" until the controller responds:  
Send ?err [wait 0.5s for response]  
Send ?err [wait 0.5s for response]  
...  
Send ?err [wait 0.5s for response]  
Send ?err => 0 [Response, controller is ready]

Response:            none

Example:             reset

### 9.4. pa (Enable or Disable the Power Amplifiers)

Syntax:               !pa or !poweramplifier  
                      ?pa or ?poweramplifier  
Parameter:            0 or 1

Description:         Switch all motor amplifiers on/off or read amplifier state.  
  
If switched off, no motor current is flowing (high impedance). To switch off individual axes, use the '**axis -1**' instructions.  
  
Amplifier switch off can also be caused by a short circuit, the PSE pin functionality or loss of motor supply voltage. Then an internal error (error 29) is generated and the status LED flashes. ?pa will return 0. After removing the switch-off cause, the amplifiers can be switched on again by !pa 1.

1 = Amplifiers on  
0 = Amplifiers off

Remarks:            With amplifiers off it can not be ensured that the axis position will be retained. A closed loop will be deactivated.

Response:            amplifier on state, 0 or 1

Example:             !pa 1            Switch on all amplifiers  
                      ?pa             Read amplifier on state

Sequence example:   !pa 0  
                      ?pa             => 0  
                      ?err            => 0

External event: PSE, short circuit or no motor voltage applied  
?pa                => 0  
?err               => 29

## 9.5. ipreter (Select Instruction Set)

Syntax:            !ipreter or ?ipreter  
Parameter:        1, 2, 3 or 4

Description:        Instruction set of the Tango controller.  
The Tango controller provides 4 different instruction sets, also called "interpreter".  
It is recommended to use the default interpreter 1, as it is the native language and supports the most complex instruction set available.  
Other instruction sets may be used for compatibility to existing applications. They are described in separate manuals.

INTERPRETER NR.

```
-----  
0   Not supported! (old register instruction set)  
  
1   Default instruction set (Native),  
    as described in this manual  
  
2   VENUS-1 and VENUS-2  
  
3   LUDL MAC5000  
  
4   ASI MS-2000 (Available with Firmware >= 1.46)  
-----
```

To return from the VENUS instruction set (2), please enter the string "1 setipreter" and press enter (or send an ASCII [CR]). For other instruction sets please refer to the corresponding instruction set description.

Response:         1, 2, 3 or 4

Example:  
!ipreter 2 => Switch the interpreter to the VENUS instruction set  
?ipreter   => Responds the currently selected interpreter

## 10. Operating Modes

### 10.1. Extended Mode

Activating Extended Mode will change the controller's behavior. Also there are new instructions available for setting calibrate and range measure velocities.

Note: When initializing the controller, the desired Extended Mode should be set directly after setting **dim** and before setting gear, pitch, vel etc.

#### Calibration in extmode = 0:

**!vel** --> The velocity for moving towards an endswitch has to be set everytime before starting a **cal** / **rm** move.  
**!calbspeed** --> There is only one velocity for all axes to travel out of the endswitch. The unit is 1/100 rev/s.

#### Calibration in extmode = 1:

**!vel** has no influence to the **!cal** and **!rm** move, **calbspeed** is no longer used. Now the calibrate (**cal**) and range measure (**rm**) velocities can be assigned once and will be used as speed especially for this instructions.

**!calvel** --> Set velocities for moving towards and out of the cal endswitch (E0)

**!rmvel** --> Set velocities for moving towards and out of the rm endswitch (EE)

#### Additional differences when in extmode = 1:

If the pitch or gear parameter is changed, all parameters which are in revolutions/s (e.g. vel) are recalculated internally. So the axis velocities will remain the same.

**joyvel** --> The joystick velocity can (and has to be) set independently from **vel** by the **joyvel** instruction.

The **?lim** instruction, when requested without an axis specifier, now returns all limits in a correctly formatted way.

#### 10.1.1 extmode (Switch to Extended Mode)

Syntax: **!extmode** or **?extmode**

Parameter: 0 or 1

Description: This instruction switches the Tango controller into extended mode. This mode offers improved behavior and more instructions than the standard interpreter. For further information please refer to the **Extended Mode** Chapter 10.1.

0 = default, compatible interpreter mode

1 = extended interpreter mode

Response: currently used extmode, 0 or 1

Examples:

**!extmode 1** Set controller behavior to extended mode

**?extmode** Read extended mode setting

## 10.2. Scan Mode

In Scan Mode the controller executes move instructions with a vector velocity.

### 10.2.1 scanmode (Switch to Scan Mode)

Syntax:            !`scanmode` or ?`scanmode`  
Parameter:        0, 1 or 2

Description:       This instruction switches the Tango controller into scan mode, which may be used for continuous path control. This mode applies a constant vector velocity for automatic moves (`moa`, `mor`) which is set by '`scanvel`'.

0 = normal operation (default, no scan mode)  
1 = scan mode 1  
2 = scan mode 2

#### Scan mode 1:

- The resulting travel velocity of automatic moves is **scanvel**.
- The individual '`vel`' settings are ignored.
- Applies to single axis and vector moves, e.g. "`!moa x 10`", "`!moa 10 20`"

#### Scan mode 2:

- Similar to scanmode 1, but individually started axes now travel at their original '`vel`' settings. May be useful e.g. when the Z-axis controls the focus.
- The resulting travel velocity of a vector move is **scanvel**.
- The individual '`vel`' settings are used for single axis move. e.g. "`!moa z -10`"
- Applies to vector moves of 2 or more axes only, e.g. "`!moa 10 20`"

Response:         Scanmode (automatic move mode) as integer

Examples:

`!scanmode 1`        Set controller into scanmode 1  
`?scanmode`         Read controller scanmode

`!vel 20 20 10` "`!scanvel 0.1`" "`!scanmode 2`"  
`!moa 50 100`       (vector move with scanvel)  
`!mor z 1.5`        (single axis move executed with vel z: 10mm/s)

### 10.2.2 scanvel (Scanmode Vector Velocity)

Syntax:            !`scanvel` or ?`scanvel`  
Parameter:        0.000001 to 1000 [mm/s]

Description:       This instruction sets or reads the '`scanmode`' vector velocity in millimeter per second. It is also used for the snapshot dissection mode (`snsm 3`) continuous path velocity. There is only one parameter.

Response:         Currently selected velocity in [mm/s]

Examples:

`!scanvel 10`       Set scanmode vector velocity to 10 mm/s  
`?scanvel`         Read scanmode velocity

### 10.2.3 modulomode (Define Linear or Turntable Modes)

Syntax: !modulomode or ?modulomode

Parameter: x, y, z, a or none  
0, 1, 2, 3 or 4

Description: This instruction sets or reads the axis modulo mode. Modulo mode switches the specified axes from linear into a turntable mode that remains within [0...360[° or [0...1[ depending on '**dim**'. 1 linear and 4 turntable modes are available:

- 0 : Modulo Mode off (default mode, e.g. for linear axes)
- 1 : Travel shortest distance to target position  
*(automatically decides to travel forward or back)*
- 2 : Only travel in positive direction  
*(may result in traveling up to one additional revolution)*
- 3 : Only travel in negative direction  
*(may result in traveling up to one additional revolution)*
- 4 : Do not travel over Zero  
*e.g. for swiveling axes <360° with limited operation range or as cable tear-off protection)*

Modes 1,2 and 3 ignore the upper and lower limits of the axis. While mode 4 uses the limits (**cal,rm,lim**) in order to narrow the possible operating range.

Remarks: Modulo Mode 1, in conjunction with the !speed instruction, can be used to achieve an endless rotating mode (pump, fan, etc).

Response: Currently selected modulo mode

Examples:

```
!modulomode 0 0 1 (set Z axis to modulo mode 1, X and Y to standard linear mode)
!modulomode a 4 (set A axis to modulo mode 4)
?modulomode => 0 0 0 (returns modulo mode of all axes, e.g. 0s)
?modulomode x => 1 (returns modulo mode of X axis, e.g. 1)
```

## 11. Controller States and Error Messages

### 11.1. autostatus (Set Autostatus to required behavior)

Syntax: !autostatus or ?autostatus

Parameter: 0, 1, 2, 3 or 4

Description: Select auto response behavior for completion of move and calibration instructions. The power-on default is mode 1. Autostatus mode can not be saved by the **'save'** instruction.

0 : Disable automatic status replies  
(Except **'save'** instructions will still return either "OK..." or "ERR")

1 : The Default.  
Position reached response after each automatic move (moa, mor, cal, rm)  
(a 5 character string with e.g. '@' for each configured axis is returned by the controller. Please also refer to **'statusaxis'** for further information. Autostatus 1 is the default configuration after power-on.

2 : The controller responds with the 'position reached' message of mode 1, plus the **status message** "OK..." or "ERR".

3 : A simple <CR> (0x0d hex) is returned to indicate that the move is completed. It can be used to improve performance for higher vector throughput, but contains less information e.g. concerning possible errors.

4 : Echoes the sent instruction including parameters.

Example: Assume a controller with 3 axes and autostatus set to 1.  
After completion of a move (moa, mor, m, a) the controller will return a 5 ASCII character string "@@@-." which means move completed.

!autostatus 0 Switch off autostatus ("**statusaxis**" has to be polled to find out if the axis moving or stopped)

!autostatus 1 Set autostatus to the default behavior

!autostatus 3 Set autostatus to <CR> response for low communication overhead

?autostatus Read the currently selected autostatus

## 11.2. statusaxis (Read State of Axis)

Syntax:                   ?statusaxis or statusaxis or sa  
Parameter:                x, y, z, a or none

Description:             Statusaxis responds the state of each axis.  
Similar to the '**autostatus 1**' response for move instructions,  
but with an additional '-' after the dot.  
It can be used for polling move states when in 'autostatus 0'  
mode, where no automatic response is generated.  
Every response except of 'M' means the axis has stopped for  
some reason and may be ready for a new move instruction.  
It is recommended to check the returned ASCII character for  
!= 'M' (not equal to 'M', 0x4D hex).

Response:                6 ASCII characters: [STATUS X][STATUS Y][STATUS Z][STATUS A].-

@ => Axis is not moving and ready  
**M => Axis is moving**  
J => Axis is ready and may also be controlled manually (by joystick)  
C => Axis is in closed loop  
S => Limit switches are actuated and prevent further automatic move  
A => ok response after cal instruction  
D => ok response after rm instruction  
E => not o.k. response after cal or rm, if an error occurred during cal  
instruction (e.g. a limitswitch is not working properly)  
U => manual adjustment (e.g. 1st setup)  
T => Timeout occurred (refer to '**caltimeout**' instruction)  
- => Axis is not enabled, not available in hardware

Example:                Assume ?statusaxis returns @@@-.-  
This means three axes are enabled and ready to move.

?statusaxis           (JM--.- = 2 axis controller, X is ready, Y is moving)  
sa                    (e.g. may return A@@-.- after calibrating the X-Axis)  
sa x                  (e.g. may return M if axis is moving)

## 11.3. status (Read the Controller Error State)

Syntax:                   ?status or status  
Parameter:                none

Description:             The ?status instruction responds with the current state of the  
controller. Which is either 'OK...' or an 'ERR' with the error  
number. The error number description can be found in chapter  
**Error Numbers**. Also refer to '**err**' and '**help**' instructions.

Response:                OK... or ERR with error number

Example:                ?status => ERR 4  
                         ?status => OK...

## 11.4. err (Read Error Number)

Syntax:               ?err or err, !err  
Parameter:           none

Description:           The instructions err or ?err return the controller error state or 0, if no error occurred. The error state will be updated or re-set by the next instruction.  
If not a permanent error (like e.g. 29) the error state may be cleared to zero by sending !err.

Response:             Error number as decimal value  
                      (refer to Chapter 6. "**Error Numbers**")

Example:              err       (may return a 0)  
                      ?err     (same as err)  
                      !err     (clear error state if no permanent error)

## 11.5. help (Read Error Number with Description String)

Syntax:               ?help or help  
Parameter:            none or requested error number

Description:           The instruction help returns a text string. It contains the error state with appended error description. The error state is not cleared to zero. Please also refer to the '**err**' instruction.

**Called without a parameter:**

It returns the controller's error state with description

**Called with a parameter (error number):**

It returns this error number with the corresponding comment

Response:             Error number as decimal value, error description as ASCII text

Example:              help     => ERROR 0,no error     (assuming there is no error)  
                      help     => ERROR 5,number outside range  
                      help 29 => ERROR 29,servo amplifier off

## 11.6. service (Print Service Information to Terminal)

Syntax:               ?service or service  
Parameter:            none

Description:           The instruction service returns a multi-line parameter and state list of the controller. It may be used for debugging or in case of service requests. Either a terminal program or SwitchBoard version 1.19 and above can be used.

Response:             Many lines of text including e.g. serial number, parameters, states etc. Each line terminated by a [CR].

Example:              service



## 11.7. pci (Is PCI Bus)

Syntax:               ?pci or pci  
Parameter:           none

Description:           Check if the Tango controller is used as PCI/PCI-E card  
                         (plugged into a PC slot).

0 = Controller is a desktop version  
1 = Controller is a PCI or PCI-E card, plugged in the PCI(-E)  
   slot of a computer (here: no RS232 or USB communication)

Response:             0 or 1

Example:              pci => 0     (e.g. a Tango Desktop)

## 11.8. isvel (Read Actual Velocities)

Syntax:               ?isvel or isvel  
Parameter:           x, y, z, a or none

Description:           Read the actual velocity(ies) at which the axis is currently  
                         traveling. Unlike `'?vel'` or `'?speed'` this instruction returns  
                         the currently traveled (true) speed of the axes, even when  
                         controlled by a HDI device.

Optional read-resolution: As an option to read the value with  
higher precision, the number of required decimal places can be  
specified with the query `"?isvel [0...16 decimal places]"`. If no  
precision is defined, the default resolution is 3 decimal  
places.

Response:             Actual axis velocity in [mm/s]

Example:  
?isvel               => Read actual velocity of all axes  
?isvel y            => Read actual velocity of the Y axis (e.g returns 10.000)  
?isvel 4            => Read actual velocity of all axes with 4 decimal places (0.0000)  
?isvel y 6          => Read actual velocity of the Y axis with 6 decimal places

## 11.9. maxpos (Maximum Position)

Syntax:               ?maxpos  
Parameter:           x, y, z, a or none

Description:           Read the maximum position value which the controller  
                         can accept due to internal limitations. It depends on e.g. the  
                         selected pitch, gear or motorsteps.

Response:             Maximum position value of the axes  
                         (unit depends on `'dim'` setting)

Example:  
?maxpos             => 2600.0000 2600.0000 2600.0000  
?maxpos x           => 2600.0000 (X axis accepts positions from -2600mm to +2600mm)

## 12. General Adjustments

With the following instructions the parameters of the controller are widely scalable to the given mechanic construction and to customer requirements. The controller is adaptable to the requested requirements.

### 12.1. dim (Unit for Positions and Velocities)

Syntax:           !dim or ?dim  
Parameter:       x, y, z, a or none  
                  0 to 9

Description:       The dim instruction sets the unit (here "dimension") of the input and output parameters related to length, e.g. position or move instructions. Dim 9 also sets the velocity parameters to mm/s (e.g. `'vel'`, `'speed'`), which else remain in motor revolutions per second.

```
0  => Micro steps
1  => µm
2  => mm      (default, velocities in motor revolutions/s)
3  => 360°
4  => revolutions
5  => cm
6  => m
7  => inch
8  => mil
9  => mm      (difference to mode 2: all velocity units in mm/s)
```

Remarks:         For dim 3 (=360°) and 4 (=revolutions) it is recommended to set the **spindle pitch** to 1mm.  
                  In dim mode 0 the Microstep resolution can be specified by the `'usteps'` instruction, in order to match the required amount of steps per revolution (e.g. 40000, 54000, 360). Fractional values can be used for positioning in Microsteps mode also.

Response:         Current dim settings

Examples:

```
!dim 4 1   set dim unit for X to [revolutions] and for Y to [µm]
!dim z 9   set dim unit for Z to [mm and mm/s]
!dim 2 2 2 set dim unit for axes X, Y and Z to [mm]
?dim      read dim unit for all axes
?dim a    read dim unit of A-axis only
```

## 12.2. pitch (Spindle Pitch)

Syntax:           !pitch or ?pitch  
Parameter:       x, y, z, a or none  
                  0.0001 to 72 [mm/rev]

Description:       This instruction sets or reads the spindle pitch which defines the axis travel distance in millimeter per motor (or gear) revolution.

Optional read-resolution: As an option to read the parameter with higher precision, the number of required decimal places can be specified with the query "?pitch [0...16 decimal places]". If no precision is defined, the default resolution is 4 decimal places.

Remarks:          If the required pitch value is an infinite number due to a 1/x function, the '**gear**' parameter can be used instead or in combination.

Response:         currently used spindle pitch in [mm per motor revolution]

Examples:  
!pitch 4 1         set spindle pitch X=4[mm] and Y=1[mm]  
!pitch z 28.895    set spindle pitch Z=28.895[mm]  
?pitch            read spindle pitch of all axes (e.g. returns 1.0000 1.0000)  
?pitch a          read spindle pitch of A-axis only  
?pitch 9          read spindle pitch of all axes with 9 decimal places  
?pitch y 6        read spindle pitch of Y-axis with 6 decimal places (1.000000)

## 12.3. gear (Gear Ratio)

Syntax:           !gear or ?gear  
Parameter:        x, y, z, a or none  
                  0.001 to 1000

Description:       This instruction sets or reads the axis gear ratio. If there is no gearbox mounted to the axis, this parameter should be left at its default value of 1.

Optional read-resolution: As an option to read the parameter with higher precision, the number of required decimal places can be specified with the query "?gear [0...16 decimal places]". If no precision is defined, the default resolution is 3 decimal places. Please also refer to the examples below.

Remarks:          If the required pitch value is an infinite number due to a 1/x function, the '**pitch**' parameter can be used instead or in combination.

Response:         currently used gear ratio

Examples:  
!gear 10          set gear ratio X=1:10  
!gear 4 1         set gear ratio X=1:4 and Y=1:1  
!gear z 12.5      set gear ratio Z=1:12.5  
?gear            read gear ratio of all axes  
?gear a          read gear ratio of A-axis only (e.g. returns 1.000)  
?gear 9          read gear ratio of all axes with 9 decimal places  
?gear z 10        read gear ratio of Z-axis only with 10 decimal places

## 12.4. motorsteps (Motor Steps Per Revolution)

Syntax: !motorsteps or ?motorsteps

Parameter: x, y, z, a or none  
[multiples of 4]

Description: This instruction sets the steps per revolution of the motor, which can be found in the datasheet. Common motors have 200 steps per revolution (1.8° full step). This is the Tango default value. Other motors may have e.g. 400, 500 or 24 steps per revolution. It is essential for operation to have this parameter set according to the datasheet. The motor steps parameter must be a multiple of 4 in the range of 4 to 65534.

Response: Selected motorsteps of the stepper motor(s)

Examples:

```
!motorsteps 200 200 400      set motor steps for X and Y to 200 and Z to 400
!motorsteps x 500           set motor steps for X to 500
?motorsteps                 read motorsteps of all axes
?motorsteps a               read motorsteps of A-axis only
```

## 12.5. accel (Acceleration)

Syntax: !accel or ?accel  
Parameter: x, y, z, a or none  
0.0001 to 20 [m/s<sup>2</sup>]

Description: This instruction sets or reads the acceleration which is used for all moves, the speed instruction and manual control by HDI devices.

Optional read-resolution: As an option to read the parameter with higher precision, the number of required decimal places can be specified with the query “?accel [0...10 decimal places]”. If no precision is defined, the default resolution is 2 decimal places.

Remarks: In case of a stop event, **'stopaccel'** is used instead.

Response: Currently used acceleration in m/s<sup>2</sup>

Examples:

```
!accel 0.5      set acceleration X=0.5[m/s2]. Other axes are not affected
!accel 1 0.55   set acceleration X=1.0[m/s2] and Y=0.55[m/s2]
!accel z 0.2    set acceleration Z=0.2[m/s2]. Other axes are not affected
?accel         read all axes for their current acceleration
?accel z       read Z axis for its acceleration
?accel 6       => 0.500000 0.123456 0.200000
?accel y 4     => 0.1235
```

## 12.6. accelfunc (Acceleration Ramp Function)

Syntax: !accelfunc or ?accelfunc  
Parameter: x, y, z, a or none  
0, 1 or 2

Description: Select the acceleration ramp type for automatic moves (e.g. m, moa, mor, moc, cal, rm).

0 = Linear acceleration/deceleration ramp  
1 = sin<sup>2</sup> acceleration/deceleration ramp  
2 = reserved, currently same as 1: sin<sup>2</sup>

Remarks: The acceleration ramp for **'go'** and **'speed'** instructions and manual control (HDI) always remains linear accelerated.

Response: Currently used acceleration type

Examples:

```
!accelfunc 1   set accel function X to sin2, other axes are not affected
!accelfunc x 1 set accel function X to sin2, other axes are not affected
!accelfunc 1 1 0 set accel function in X and Y to sin2, Z to linear accel.
?accelfunc     read acceleration ramp function of all axes
?accelfunc z   read acceleration ramp function of Z axis only
```

## 12.7. stopaccel (Emergency Stop Deceleration)

Syntax: !stopaccel or ?stopaccel

Parameter: x, y, z, a or none  
0.001 to 200 m/s<sup>2</sup>

Description: This instruction sets the deceleration for emergency stop conditions. It will be used by:

- abort instructions
- active stop input
- a '**cal**' or '**rm**' move (at the limit switch)
- when detecting an unexpected limit switch

Optional read-resolution: As an option to read the parameter with higher precision, the number of required decimal places can be specified with the query "?stopaccel [0...10 decimal places]". If no precision is defined, the default resolution is 2 decimal places.

Response: Deceleration for stop conditions

Examples:

```
!stopaccel 1 1 2 Set the stop deceleration for X and Y to 1 and Z to 2 [m/s2]  
!stopaccel x 1.5 Set the X stop deceleration to 1.5[m/s2]  
?stopaccel Returns the stop deceleration of all axes  
?stopaccel z Returns the stop deceleration of Z axis only (e.g. 1.50)  
?stopaccel 6 Returns stop deceleration of all axes with 6 fractional digits  
?stopaccel z 9 Returns stop deceleration of Z axis with 9 fractional digits
```

## 12.8. vel (Velocity)

Syntax:	!vel or ?vel
Parameter:	x, y , z, a or none 0.000001 to 200 [rev/s] (or up to 3000 [mm/s] if dim = 9)
Description:	Velocity for automatic moves, cal**, rm** and HDI**  Except of dim=9 the unit is always motor revolutions per second. In dim=9 the unit is [mm/s].  Optional read-resolution: As an option to read the parameter with higher precision, the number of required decimal places can be specified with the query "?vel [0...16 decimal places]". If no precision is defined, the default resolution is 3 decimal places.
Remarks:	If <b>extmode</b> =0 (default), vel is also used for 1) the HDI (joystick) velocity 2) the cal and rm instructions In extmode=1 there are separate parameters (joyvel,calvel,...).  The ' <b>velfac</b> ' instruction can be used in addition to 'vel', but is not necessary or recommended.
Response:	Currently selected velocity
Examples:	<pre>!vel 1.0 15      set velocity X=1[revolution/s] and Y=15[revolution/s] !vel z 0.1      set velocity Z=0.1[revolution/s] ?vel           read velocity of all axes ?vel x         read velocity of X axis only ?vel 6         =&gt; 20.000000 0.123456 5.000000 ?vel y 4       =&gt; 0.1235</pre>

## 12.9. velfac (Velocity Factor)

Syntax:               !velfac or ?velfac  
Parameter:           x, y , z, a or none  
                      0.01 to 1.00

Description:         This instruction sets or reads the velocity factor, which is used for all consecutive automatic moves. It is internally multiplied to the velocity (vel).

Response:            Currently used velocity factor [0.01 to 1.00]

Examples:  
?velfac               read velocity factor of all axes  
?velfac z             read velocity factor of Z axis only  
!velfac x 0.1         set velocity factor of X axis to 1/10 of specified velocity  
!velfac 1 1 1         set velocity factor of X,Y,Z to specified velocity (default)

## 12.10.    secvel (Secure Velocity)

Syntax:               !secvel or ?secvel  
Parameter:            x, y , z, a or none  
                      0.000001 to 100 [mm/s]

Description:         The security speed limitation is used as long as the axis is not calibrated and range measured ('cal', 'rm'). The unit is always mm/s and does not depend on the 'dim' setting. It is intended to prevent mechanical damage as long as the controller does not know the mechanical limits of the axis. (The breaking distance behind the hardware limit switches often is not sufficient to stop the axis under all velocities).

Setting this parameter to higher values may also be used as a workaround at own risk, if executing a cal/rm is not wanted. Axes which only have one (E0/cal) or do not provide any limit switch (refer to 'swact' settings) disable the security speed limit after a cal or do not even apply the limit.

Optional read-resolution: As an option to read the parameter with higher precision, the number of required decimal places can be specified with the query "?secvel [0...16 decimal places]". If no precision is defined, the default resolution is 2 decimal places.

Response:            Currently used secure velocity in [mm/s]

Examples:  
!secvel 100 100 100 => Set maximum possible velocity of X Y Z  
!secvel y 14.5       => Set maximum possible velocity of Y to 14.5 mm/s





## 12.11. cur (Motor Current)

Syntax: !cur or ?cur  
Parameter: x, y, z, a or none  
0.03 to [maximum current]

Description: This instruction sets or reads the motor current. The maximum current is limited by hardware or additionally by factory settings (ETS) and may be checked using '**maxcur**' instruction.

Remarks: Please check the motor datasheet first in order not to destroy the motor by overcurrent/overtemperature. Also, if the motor current set too low it can cause the axis to move incorrectly and lead to mechanical damage. At least for open loop systems (without encoder feedback) it is required to ensure the axis can travel under all required velocities and load situations.

Response: Motor current in Ampere (e.g. 1.00)

Examples:  
!cur 1.1 set X motor current to 1.1[A]  
!cur 1 2 set motor current for X=1[A] and Y=2[A]  
!cur z 0.3 set Z motor current to 0.3[A]  
?cur read motor current of all axes  
?cur x read motor current of X axis only

## 12.12. reduction (Motor Current Reduction Factor)

Syntax: !reduction or ?reduction

Parameter: x, y, z, a or none  
0 to 1.00

Description: Motor current reduction factor when idle, used to reduce the dissipated heat from the motor.  
When the axis is idle (stopped), the motor current (**cur**) is reduced by this factor.  
Floating point numbers from 0 to 1.00 represent 0 to 100% of the motor current. Reduction is disabled when set to 1 (default).

Remarks: The current reduction can be delayed by the '**curdelay**' instruction.  
Without a delay, reduction might have a slight impact on vector throughput, as it takes some extra time reduce and increase the current after and ahead of each move instruction. While reducing, the axis might also show slight "waggle", depending on mechanical load.

Response: Reduction factor(s) [0.00 to 1.00]

Examples:

```
!reduction .3 .7 Set idle current reduction X=0.3*cur[A] and Y=0.7*cur[A]
!reduction 1 1 1 Disable reduction for axes X,Y,Z
!reduction z 0.5 Set Z idle current reduction factor to 0.5*cur[A]
?reduction Read idle current reduction factor of all axes
?reduction x Read idle current reduction factor of X axis only (e.g. 1.00)
```

## 12.13. curdelay (Delay for Current Reduction)

Syntax: !curdelay or ?curdelay

Parameter: x, y, z, a or none  
0 to 65000 [ms]

Description: At the end of each move the axis enters the idle state. If the motor current **reduction factor** is set to a value less than 1.0 this reduction will take effect after the curdelay time.

Remarks: A delay might be necessary in cases of  
- long time exposure (to avoid waggle)  
- high vector throughput (to avoid reduction between the move)  
- heat reduction when not operating (as a "savesaver")

Response: Selected delay time for the current reduction in [ms]

Examples:

```
!curdelay 100 450 Set delay for motor current reduction X=100[ms] and Y=450[ms]
!curdelay z 15000 Set delay for motor current reduction Z=15 seconds
!curdelay 0 0 Set immediate reduction for X and Y axis
?curdelay Read motor current reduction delay of all axes
?curdelay x Read motor current reduction delay of X axis only
```



## 12.14. ecomove (EcoMove Current Level)

Syntax: !ecomove or ?ecomove

Parameter: x, y , z, a or none  
0 to 70

Description: EcoMove reduces the motor current while the axis travels at constant velocity. Like "**reduction**" it can be used to greatly reduce the dissipated heat of the motor. But the EcoMove parameter works in a different way: Greater values = greater power saving, which means a eco level value of 0 is 100% current and a eco level of 30 is 70% current.

0 = Full motor current (default)

70 = Maximum power saving level (low motor heating and force)

Response: EcoMove level

Examples: !ecomove 0 0 0 0 (disable ecomove for all axes / 0%)  
!ecomove x 25 (reduce X current by 25% when moving)  
?ecomove (return all ecomove levels)

## 12.15. axis (Enable, Disable, Switch Off Axis)

Syntax: !axis or ?axis  
Parameter: x, y, z, a or none  
-1, 0, 1

Description: This instruction enables, disables and switches off axes. The currently selected state can also be read.  
A disabled axis still powers the motor with its current, while a switched off axis loses its torque.

1 = enabled  
0 = disabled  
-1 = axis power stage off

Response: Axis enable state

Examples:  
!axis 1 1 1 1 enable all axes  
!axis 1 0 1 0 disable Y and A axis, enable X and Z  
!axis y -1 switch off Y axis: power stage Y off  
?axis x read axis state of X axis only  
?axis read axis state of all axes

## 12.16. axisdir (Axis Direction)

Syntax: !axisdir or ?axisdir  
Parameter: x, y, z, a or none  
0 or 1

Description: This instruction sets and reads the travel direction of the axes.  
Please make sure to first set the desired axis direction before setting the end switch types, polarity etc.!  
It is not recommended to change direction during operation!

0 = Normal direrction, CAL switch => E0, RM switch => EE  
1 = Reversed direrction, CAL switch => EE, RM switch => E0

Remarks: The hardware limit switches CAL/E0, RM/EE will automatically be reassigned when switching the axis direction. Also swact, swpol, swtyp, readsw etc. Exception: The 'swin' function is not affected.  
Closed loop will be deactivated when changing the diredtion and has to be reenabled by cal or reset/power-on.

Response: Current axis direction

Examples:  
!axisdir 0 1 0 1 Reverse travel directions of Y and A axis  
!axisdir z 1 Set reverse travel direction for Z axis  
?axisdir Read axis direction of all axes  
?axisdir x Read axis direction of X axis only

## 12.17. motortable (Motor Correction Table)

Syntax: !motortable or ?motortable  
Parameter: x, y , z, a or none  
0 or number specified by factory

Description: This instruction adds a motor correction, which can be used to reduce resonances and vibration. The motor has to be measured for the specific application by factory. Then a table number will be assigned and the customer may activate it by setting the corresponding motortable number. Using a wrong motortable will lead to increased noise and position error.

0 = No correction

Response: Currently used motortable(s)

Examples:  
!motortable 1 1 2 0      Select motortable 1 for X and Y, 2 for Z and no for A  
!motortable x 0          Disable correction for x  
?motortable              Read the currently used tables for all axes

## 12.18. usteps (Microstep Resolution)

Syntax: !usteps or ?usteps  
Parameter: 360 ... 1638400

Description: This instruction sets the microsteps for one motor revolution, used in axis unit "**dim 0**" (Microsteps). It offers compatibility to application software that is written for e.g. 40000 or 54000 microsteps. One value applies to all axes that have **dim 0** selected.

Remarks: For positioning (moa, mor etc.) in microsteps dim mode 0, the microstep position can also be specified with fractional digits (e.g. !mor 40000 4500 0.5).

Response: Currently used **dim 0** microstepping resolution in [steps/rev]

Examples:  
!usteps 40000      Set microstep resolution to 40000/revolution (for dim=0)  
?usteps            Read the microstep resolution

## 12.19. resolution (Position Number Format)

Syntax: !resolution or ?resolution  
Parameter: 0, 1, ... 6

Description: This instruction sets the resolution of '?pos' and similar position returning instructions for **dim** 1, 2 and 9. It affects the amount of returned decimal places, as listed below. One value applies to all axes, default = 4 (100nm resolution).

Value	Resolution dim 2,9	Resolution dim 1
0	= 1mm	0.1 $\mu$ m
1	= 0.1mm	0.1 $\mu$ m
2	= 0.01mm	0.1 $\mu$ m
3	= 0.001mm	0.1 $\mu$ m
4 (default)	= 0.0001mm	0.1 $\mu$ m
5	= 0.00001mm	0.01 $\mu$ m
6	= 0.000001mm	0.001 $\mu$ m

Affected instructions are: ?pos, ?lim, ?maxpos, ?posclr, ?distance, ?twi, ?ctrs, ?ctrdiff, ?caliboffset, ?rmoffset, ?calpos, ?trigd, ?snsa, ?snsp.

Response: Responded decimal places for the 'pos' and other position returning instructions.

Examples:  
!resolution 5 Set position read resolution to 10 nm (5 decimal places if mm)  
e.g. "?pos x" returns 0.00000 in dim 2 and 9, 0.00 in dim 1.  
?resolution Read the resolution of

## 12.20. backlash (Mechanical Backlash Compensation)

Syntax: !backlash or ?backlash  
Parameter: x, y, z, a or none  
-100.0 ... 100.0 [ $\mu$ m]

Description: This instruction compensates mechanical backlash for each axis. Unit is  $\mu$ m, independent from dim.

0 = Backlash compensation off

Response: Currently used backlash in  $\mu$ m

Examples:  
!backlash 12.7 21.3 0 Set backlash for X to 12.7 $\mu$ m, Y=21.3 $\mu$ m and Z=none  
!backlash x 0 Disable backlash compensation for X  
!backlash x 5 Compensate a backlash of 5 $\mu$ m in X  
?backlash Read the backlash compensation value of all axes  
?backlash z Read the backlash compensation value of Z axis only

## 12.21. lock (Select Parameters to Lock)

Syntax: ?lock or !lock  
Parameter: 0 to 15, 0 or 1

Description: Select write protection for Tango parameters (lock state).  
Either bitwise: !lock [bit number] [0 or 1]  
or multi bits : !lock [bit field of 0s and 1s]  
After selecting the parameters to lock, these have to be applied to the desired axes by '**lockaxis**'.

Response: Specified lock bit state or entire lock bit field, LSB first.  
The bit positions represent the following parameters:

Bit Nr.	Parameter
0:	Pitch
1:	Gear
2:	Cur
3:	MotorSteps
4:	SwPol
5:	SwTyp
6:	SwDir
7:	EncTTL
8:	EncPeriod
9:	AxisDir
10:	MotorTable
11:	BackLash
12:	Anglecorr
13:	CalLrnPos

Example: !lock 111 => Set lock bits 0 1 and 2, leave others unaffected  
!lock 2 0 => Clear lock condition for parameter 2 (=current)  
!lock 0 1 => Set lock bit for parameter 0 (pitch)  
?lock => Read lock bit field (e.g. "0000000000000000")  
?lock 5 => Read lock bit #5 state

## 12.22. lockaxis (Apply the Parameter Lock to Axes)

Syntax: ?lockaxis or !lockaxis  
Parameter: x, y , z, a or none

Description: Apply the parameter lock, selected by the '**lock**' instruction, to the specified axes. If the lockbits or lockaxis are zero nothing will be locked.

Response: Axes to which the lock bits are currently applied.

Example: !lockaxis y 1 => Apply lock bits to Y axis  
!lockaxis 1 1 => Apply lock bits to X and Y axis  
?lockaxis x => Read if lock bits are applied to the X axis  
?lockaxis => Read all axes (returns e.g. "1 1 0 0")

## 12.23. lockstate (Read all internal Lock States)

Syntax: ?lockstate  
Parameter: x, y, z, a or none

Description: Set/read the internal parameter write protection (lock) state caused by the ETS (factory) and user lock/lockaxis settings. The bit positions represent the following parameters:

Bit Nr.	Parameter
0:	Pitch
1:	Gear
2:	Cur
3:	MotorSteps
4:	SwPol
5:	SwTyp
6:	SwDir
7:	EncTTL
8:	EncPeriod
9:	AxisDir
10:	MotorTable
11:	BackLash
12:	Anglecorr
13:	CalLrnPos

Response: Lock state as 16 bits ASCII string(s), 0s and 1s, LSB first

Example: ?lockstate => Read lock state of all axes  
?lockstate x => Lock state of X axis e.g. "1100000000000000"

## 12.24. stout (Select Status Signal Output)

Syntax: !stout or ?stout  
Parameter: 0,1,2,3,4

Description: Makes the state of the Tango Status LED available to the optional AUX-I/O connector:

0 = Just Status LED, no AUX-I/O used (Standard)  
~~1 = AUX-I/O Pin 5 (TAKT\_OUT) may not be available!~~  
2 = AUX-I/O Pin 6 (VR\_OUT)  
3 = AUX-I/O Pin 7 (SHUTTER\_OUT)  
4 = AUX-I/O Pin 8 (TRIGGER\_OUT)

Remarks: To turn off the Status LED, instruction 'noled' may be used.

Response: Selected status output mode

Example: !stout 0 => Only use Status LED (default)  
?stout => Read status output mode (returns 0,1,2,3 or 4)



## 12.25. noled (Select Status Signal Output)

Syntax: !noled or ?noled  
Parameter: 0 or 1

Description: Set Tango Status LED to "used" or "forced off". Forcing the LED off may be necessary in low light applications where no external light source is wanted.

0 Normal operation, Status-LED on (default)  
1 Status-LED permanently off

Response: Status LED mode

Example: !noled 1 => Force Status-LED permanently off  
!noled 0 => Status-LED normal operation (default)  
?noled => Read status output mode (returns 0 or 1)

## 12.26. updelay (Power Up Delay)

Syntax: !updelay or ? updelay  
Parameter: -5000 to 5000

Description: Delay time of the Tango controller on power up in [ms].

This parameter is ment for fixing problems of Tango PCI/PCI-E card versions with external power supply or long PCI reset times of the computer mainboard.

Applications:

Use negative values to wait for valid motor voltage (e.g. when using master-slave power switches for the external power supply).

Use positive values to wait a fixed time (e.g. when the mainboard generates a too long reset signal it causes the PCI card to start as a Desktop version. So the COM port is not accessible.)

Positive values: The controller waits for the specified time.

Negative values: The controller waits for valid motor voltage for a maximum of this time or shorter.

Response: Power up delay time in [ms]

Example: !updelay -2000 => Wait max. 2s for valid motor voltage level  
!updelay 2500 => Wait 2.5s extra on power up  
?updelay => Read the power up delay

## 13. Limit Switch Instructions (Hardware and Software)

### 13.1. lim (Software Limits)

Syntax: `!lim or ?lim`  
Parameter: `x, y, z, a or none`  
`+-maximum position range (unit depends on 'dim')`

Description: This instruction sets or reads the movement range limitations. The upper and lower software limits must send together in a single `!lim` instruction. The position unit depends on the `'dim'` setting.

Remarks: In **Extended Mode** (`extmode = 1`) the `'?lim'` instruction returns the limits as a correctly formatted string, as shown in the example below.

Response: `Currently used software limits [lower] [upper]`

Examples:

```
!lim -1000 1000 -2000 2000    set the software limits for X and Y
!lim z -500 1700             set the software limits for Z
!lim z -45.37                set the lower software limit of Z
?lim y                       read software limits of Y-axis only
?lim                         read software limits of all axes
                             only recommended in extmode=1, as shown below:
```

`?lim` response example for 3 axes in

```
--> extmode=0: -1000 1000, [CR]-1000 1000, -1000 100[CR]
--> extmode=1: -1000 1000 -1000 1000 -1000 100[CR]
```

### 13.2. limctr (Enable or Disable Limit Control)

Syntax: `!limctr or ?limctr`  
Parameter: `x, y, z, a or none`  
`0 or 1`

Description: This instruction enables or disables the limit control or returns the current state of it.

Attention: If limit controls are disabled, the controller doesn't care about limits, which may cause mechanical damage! Limit control is enabled by default from power on.

`0 = disabled`  
`1 = enabled (default)`

Response: `Limit control state`

Example:

```
!limctr y 0                 disable Y limit control, Y axis limit switches are ignored
!limctr 1 1 1              enable X, Y and Z limit control
!limctr z 1                enable Z limit control
?limctr a                  read limit control state of A axis only
?limctr                    read limit control state of all axes
```

### 13.3. nosetlimit (Do not set limits by cal/rm)

Syntax:                   !nosetlimit or ?nosetlimit  
Parameter:               x, y, z, a or none  
                          0 or 1

Description:             Enables or disables the setting of software limits by the calibration (**cal**) and range measure (**rm**) functions. The default is nosetlimit=0 which means that the software limits are set by the cal/rm moves to these min/max positions.

Response:                0 = set software limits to !cal and !rm positions (default)  
                          1 = do not change software limits after !cal or !rm

Examples:  
!nosetlimit 1 1       X and Y axis do not take software limits after !cal and !rm  
!nosetlimit y 1      Y axis is does not set software limits of !cal and !rm move  
?nosetlimit           read nosetlimit state of all axes  
?nosetlimit a        read nosetlimit state of A axis only

### 13.4. swtyp (Type of Limit Switch)

Syntax:                   !swtyp or ?swtyp  
Parameter:                x, y, z, a or none  
                          0 or 1

Description:             Set or read the type of the limit switches. The sequence is [E0] [REF] [EE] for all axes. The [REF] switch is not used by the Tango controller.

Remarks:                When using no axis parameter (x,y,z or a), the 3 values will be used for all axes! To set individual axes, please do this separately, use the axis parameter x,y,z or a. Please note that the E0 and EE switches are reassigned by a change of the '**axisdir**' instruction.

0 = PNP, which adds a pull-down resistor to the switch input  
1 = NPN, which adds a pull-up resistor (default)

Response:                Currently selected limit switch type

Examples:  
!swtyp z 0 0 1       Set Z axis limit switches E0=PNP, REF(don't care), EE=NPN  
?swtyp y             Read switch types of Y axis  
  
!swtyp 1 0 1         Set all! limit switches to NPN type (not recommended)  
?swtyp               Not recommended,  
                      in extmode=0 returns e.g. 1 1 11 1 11 1 1  
                      in extmode=1 returns e.g. 1 1 1 1 1 1 1 1 1

## 13.5. swpol (Polarity of Limit Switch)

Syntax:               !swpol or ?swpol  
Parameter:            x, y, z, a or none  
                          0 or 1

Description:           Set or read the polarity of the limit switches.  
                          The sequence is [E0] [REF] [EE] for all axes.  
                          The REF switch is not used by the Tango controller.  
                          Important: When using no axis parameter (x,y,z or a), the 3  
                          values will be used for all axes! To set individual axes,  
                          please do this separately, use the axis parameter x,y,z or a.  
                          Please note that the E0 and EE switch are reassigned by the  
                          '**axisdir**' instruction.

0 = switch has active low signal  
1 = switch has active high signal

Response:             Polarity of the limit switches

Examples:

```
!swpol y 1 1 1     set polarity of Y limit switches (E0 REF EE) to positive edge
!swpol z 0 0 0     set polarity of Y limit switches (E0 REF EE) to negative edge
!swpol 1 0 1       set polarity of limit switches (E0 REF EE) for all axes
?swpol a           read limit switch polarity of the A axis
```

## 13.6. swdir (swap assignment of cal and rm switch)

Syntax:               !swdir or ?swdir  
Parameter:            x, y, z, a or none  
                          0 or 1

Description:           Swap the cal(E0) and rm(EE) switch assignment.

0 = switches are not swapped (default)  
1 = switches are swapped

In opposite to the axisdir instruction, which swaps motor direction and endswitch assignment, swdir only swaps the switches E0<->EE without changing the axis direction. This may become necessary due to wiring of the axis and depends on the axis hardware. It is independent of axisdir, which works with and without a swapped swdir.

Attention:            swdir should only be used to compensate different wiring of the stage endswitches. **Swapping the switches to the wrong assignment may result in mechanical damage!**

Response:             Current state of endswith assignment(s)

Examples:

```
!swdir 1 1 0       Swap E0<->EE switch assignment in X and Y, not in Z
!swdir x 1          Swap E0<->EE switch assignment in X (E0 switch is now EE etc.)
?swdir              Read switch assignment of all axes
?swdir z            Read switch assignment of Z axis only
```

## 13.7. swact (enable or disable limit switches)

Syntax:                   !swact or ?swact  
Parameter:               x, y, z, a or none  
                          0 or 1

Description:             This instruction enables or disables the limit switches.  
                          The sequence is always:

E0 REF EE

0 = switch is inactive (actuation state is ignored)  
1 = switch is active

The REF switch is not used by the Tango controller.  
Disabling limit switches may damage the hardware.  
When using no axis parameter, the 3 values will be used for all axes! To set individual axes please do this separately, use the axis parameter x, y, z or a.  
Inactive switches always return a non actuated state when using '?readsw', while the 'swin' instruction still returns the switches TTL logic level.  
Please note that the hardware E0 and EE switches are reassigned by the 'axisdir' instruction.

Remarks:                If all switches of an axis are set to inactive, 'secvel' will not be applied (no velocity limitation at all). If EE (rm) switch is set to inactive, the secvel limitation will be released after the cal. If both switches E0+EE are activated, cal+rm must be executed in order to release the secure velocity.

Response:                Limit switch enable states [E0] [REF] [EE] (e.g. 1 0 1)

Examples:  
!swact 1 0 1            Enable cal and rm limit switches for all axes (REF disabled)  
!swact z 1 0 1         Set Z limit switches E0=enabled REF=disabled EE=enabled  
?swact a                Read limit switches enable state of A axis only

## 13.8. readsw (Read Status of Limit Switches)

Syntax:               ?readsw or readsw  
Parameter:           none

Description:         Readsw returns the actuation state of the limit switch (while swin returns the signal level). Also readsw exchanges the E0 and EE switch assignment (cal direction is always E0) depending on 'axisdir'. Disabled switches (swact) are read as 0.

0 = limit switch is currently not actuated or disabled  
1 = limit switch is currently actuated (axis is in switch)

Please note that the switch state is only valid when the swtyp, swpol parameters are set correctly and the switch is activated by swact.

Sequence of the 12 characters is:

Axis:     X   Y   Z   A   X   Y   Z    A   X   Y   Z   A  
Switch: [E0][E0][E0][E0][Ref][Ref][Ref][Ref][EE][EE][EE][EE]

E0 = lower limit switch (!cal instruction)  
Ref = Reference switch  
EE = upper limit switch (!rm instruction)

Response:            Actuation state of limit switches as 12 character ASCII string

Examples:            readsw => 000000001000  
                      (read all limit switch actuation states, EE of X is actuated)

## 13.9. swin (Read Limit Switch Input Level)

Syntax:               ?swin or swin  
Parameter:            none or 0...7

Description:         This instruction reads the limit switch signal directly. The response is a string of 8 characters, either 0 or 1.

0 = limit switch input signal is TTL low  
1 = limit switch input signal is TTL high

In opposite to the "readsw" instruction, swin reflects the TTL input levels. Also disabled switches are represented with their current TTL input signal level. Swin is not affected by axisdir setting (does not exchange E0 and EE switches). The Ref signals are not used.

Sequence of the 8 characters is:

Axis:     X           Y           Z           A  
Switch: [E0][EE][E0][EE][E0][EE][E0][EE]  
Index :  0   1   2   3   4   5   6   7  
E0 = lower limit switch (!cal)  
EE = upper limit switch (!rm)

Response:            Limit switch input TTL level as 1 or 8 ASCII characters

Examples:            swin     => 11111111 (read all 8 limit switch signal levels)  
                      swin 1 => 1           (read EE of X Axis signal level)

## 13.10. statuslimit (Limit Status)

Syntax:                ?statuslimit or statuslimit  
Parameter:            none

Description:          Status of the known limits (hardware and software).

They are arranged in 3 groups. Character string positions are:

```
0 ... 3: Group 1 => cal state of axis 0-3 (x,y,z,a)
4 ... 7: Group 2 => rm  state of axis 0-3 (x,y,z,a)
8 ... 11: Group 3 => lower software limit state of axis 0-3 (x,y,z,a)
12 ... 15: Group 4 => upper software limit state of axis 0-3 (x,y,z,a)
```

The characters represent the state with 4 possible characters:

```
- => limit not set/modified since power on
A => axis is calibrated (!cal)
D => axis is range measured (!rm)
L => software limit has been modified by (!lim)
```

Response:             ASCII string of 16 characters

Example:              Assume `?statuslimit` returns the string "AA-A---D-LL-L--L"

This means in detail:

```
[ 0] A -> X-axis is calibrated
[ 1] A -> Y-axis is calibrated
[ 2] - -> Z-axis is not calibrated
[ 3] A -> A-axis is calibrated
[ 4] - -> X-axis is not range measured
[ 5] - -> Y-axis is not range measured
[ 6] - -> Z-axis is not range measured
[ 7] D -> A-axis is range measured
[ 8] - -> X-axis lower software limit is not modified
[ 9] L -> Y-axis lower software limit is modified
[10] L -> Z-axis lower software limit is modified
[11] - -> A-axis lower software limit is not modified
[12] L -> X-axis upper software limit is modified
[13] - -> Y-axis upper software limit is not modified
[14] - -> Z-axis upper software limit is not modified
[15] L -> A-axis upper software limit is modified
```

## 14. Calibration and Range Measure Instructions

After power on or '!reset' of the controller, a calibration (instruction **!cal**) followed by a range measure (instruction **!rm**) should be executed in order to set the axis origin and hardware position limits.

From then on the controller will stop the axes automatically at the end of the available positioning range. It also disables the secure travel speed limitation of '**secvel**', which else is applied to protect the axes from possible damage.

The cal/rm instructions set the limits close to the limit switch positions. An additional position offset for the these limits can be specified with the instructions **!caliboffset** and **!rmoffset**.

Long axes and/or slow velocities may exceed the default calibration timeout of 40 seconds. Therefore the timeout can be set to the desired value by **caltimeout**. Please also refer to the optional **extmode** enhancements and **calmode** options for calibration.

### 14.1. cal (Perform Calibration to lower Limit Switch)

Syntax:                !**cal** or **cal**  
Parameter:            **x, y, z, a** or none

Description:           This instruction moves either the specified or all axes towards lower positions until the limitswitch E0 is detected. If the corresponding switch is disabled (swact), cal will not be performed.  
The behavior of CAL depends on the setting of '**extmode**'.

**Extmode=0 (mostly the default setting):**

- CAL travels towards switch with the axis velocity '**vel**'
- CAL travels out of switch with '**calbspeed**'

**Extmode=1:**

- CAL travels towards switch with '**calvel**' parameter 1
- CAL travels out of switch with '**calvel**' parameter 2

The movement stops slightly ahead of the position where the switch was released. If required this travel distance can be increased by specifying a '**caliboffset**'.

Depending on the '**calmode**' setting this position is used as origin (position 0) and if '**nosetlimit**' is set to 0 (default) also as the new "lower software limit".

Remarks:             The calibration status can be read by checking '**statuslimit**'. Highest repeatability (in combination with MR encoders only) can be achieved by teaching a precise '**callrn**' position.

Response:             **Autostatus** dependent, similar to !moa, !mor etc. but instead of the '@' the axis status return string contains an '**A**' after a successful calibration or '**E**' if an error occurred (cal was unsuccessful) '**T**' if a timeout occurred (cal was unsuccessful) '- ' the axis is not present

Examples:

```
!cal                   execute a calibration for all enabled axes => "AAA-."  
cal y                  execute a calibration for Y axis           => "@A@-."
```

```
This sequence calibrates the X and Z axis => "A@A-."  
"!axis 1 0 1"  
"!cal"  
"!axis 1 1 1"
```



## 14.2. rm (Perform Range Measure to upper Limit Switch)

Syntax: !rm or rm

Parameter: x, y, z, a or none

Description: This instruction moves either the specified or all axes towards higher positions until the limitswitch EE is detected. If the corresponding switch is disabled (swact), rm will not be performed.  
The behavior of RM depends on the setting of 'extmode'.

**Extmode=0 (mostly the default setting):**

- RM travels towards switch with the axis velocity 'vel'
- RM travels out of switch with 'calbspeed'

**Extmode=1:**

- RM travels towards switch with 'rmvel' parameter 1
- RM travels out of switch with 'rmvel' parameter 2

The movement stops slightly under the position where the switch was released. If required this travel distance can be increased by specifying a 'rmoffset'.

Depending on 'nosetlimit' (=0, default) also as the new "upper software limit" is set.

Remarks: The rangemeasure status can be read by checking 'statuslimit'.

Response: **Autostatus** dependent, similar to !moa, !mor etc. but instead of the '@' the axis status return string contains an 'D' after a successful rangemeasure or 'E' if an error occurred (cal was unsuccessful) 'T' if a timeout occurred (cal was unsuccessful) '-' the axis is not present

Examples:

```
!rm          execute a range measure for all enabled axes => "DDD-."
rm y        execute a range measure for Y axis           => "@D@-."
```

Or the sequence

```
"!axis 1 0 1"
```

```
"!rm"
```

```
"!axis 1 1 1"
```

range measure the X and Z axis, while Y is not used => "D@D-."

## 14.3. vrm (Virtual Range Measure)

Syntax: !vrm or vrm

Parameter: x, y, z, a or none  
Axis length(s) in user dim

Description: Can replace the need of a rm move, by just defining the axis length without axis travel (time saving).

Same as with a rm instruction the

- SecVel is released
- corresponding statuslimit 'E' entry is set
- hardware limit is set

Vrm requires that a !cal was performed on the axis.  
Vrm does not return the @@@ autostatus reply.

Remarks: Axes with MR encoders (nanoScale) should always perform a true Rm for high accuracy measurement.  
Also, specifying a wrong axis length can cause mechanical damage.

Check ?**err** response or **statuslimit** 'E' entries to make sure the parameters were accepted.

If axes are not calibrated the vrm function causes error 2.

Response: None

Examples:  
vrm 75 50 virtual range measure (define axis lengths of X=75, Y=50)  
vrm y 150 virtual range measure for Y axis (define Y=150)

Sequence example:

```
cal x
cal y
cal z
?statuslimit    => "AAA-----"
vrm 300 300 20
?statuslimit    => "AAA-DDD-----"

cal x
cal y
!moa 10 10
!pos 0 0        (the lower limit is now at position -10 -10)
vrm 300 300    (the upper limit is set to 290 290)
```

## 14.4. calmode (Closed Loop/Calibration Behavior)

Syntax: !calmode or ?calmode

Parameter: x, y, z, a or none  
0, 1 or 2

Description: This instruction reads or sets the closed loop behavior on power-up and also affects the calibration behavior:

0 = Cal instruction sets the zero position (default setting)  
Closed loop enabled after cal move

1 = the power-up position remains the zero position even after calibration  
Closed loop is enabled from power-up

2 = Cal instruction sets the zero position  
Closed loop is enabled from power-up

Remarks: For activating the closed loop, the '**encmask**' of the corresponding axes must be set to one. If encmask is not set or no encoder is present, the calmode only affects the axis zero position behavior.

Response: Calmode of the axes

Examples:

```
!calmode 0 0 0    Calmode behavior of axes X,Y,Z set to default operation
!calmode z 2      Set Z axis to enter closed loop instantly after power-up
?calmode          Read calmode of all axes
?calmode y        Read calmode of Y axis only
```

## 14.5. carequired (Calibration Required)

Syntax: !calrequired or ?calrequired

Parameter: x, y, z, a or none  
0 or 1

Description: The calibration required mode offers a mechanism to avoid axis travel until the !cal instruction is executed. It can be set for each axis individually.

0 = Axes can move even without !cal (default)

1 = Axes are halted as long as !cal was not executed (

Response: Calrequired setting of the axes

Examples:

```
!calrequired 0 0 0    No cal required to enable X,Y,Z axis move (default)
!calrequired z 1      Cal required to enable axis move in Z
?calrequired          Read calrequired mode of all axes
?calrequired y        Read calrequired mode of Y axis only
```



## 14.6. caltimeout (Calibration Timeout)

Syntax: !caltimeout or ?caltimeout  
Parameter: x, y, z, a or none  
1 to 120 (seconds, as integer)

Description: This instruction specifies the timeout for calibration (cal) and range measure (rm) instructions. It can be set for each axis individually, depending on axis length and velocities. The default value is 40 or 60 seconds.

Remarks: For long axes (over 350mm) the default timeout of 40s is insufficient and must be increased, as a typical cal/rm travel velocity is 10mm/s.

Response: Calibration+RangeMeasure timeout in seconds

### Examples:

```
!caltimeout 60 60 60    set the cal/rm timeout for X,Y,Z to 60 seconds
!caltimeout x 40       set the cal/rm timeout for X to 40 seconds
?caltimeout            read timeout of all axes
?caltimeout z         read timeout of Z axis only
```

## 14.7. caliboffset (Calibration Offset)

Syntax: !caliboffset or ?caliboffset

Parameter: x, y, z, a or none  
Position (0.0 ... 100mm, depends on 'dim')

Description: This instruction specifies a calibration position offset. When executing a 'cal' instruction, the axis travels this extra distance away from the limit switch E0 and sets the origin (axis zero position) there. The unit depends on the current 'dim' settings. Valid position range is "0.0 to 100mm equivalent". The default value is 0.

Response: Calibration Cal switch position offset

Examples:

```
!caliboffset 10 5.5 0.1 set the calibration offset for X, Y and Z axis
!caliboffset x 0.05    set the calibration offset for X axis to 0.05
?caliboffset          read the calibration offset of all axes
?caliboffset y        read the calibration offset of Y axis only
```

## 14.8. roffset (Range Measure Position Offset)

Syntax: !roffset or ?roffset

Parameter: x, y, z, a or none  
Position (0.0 ... 100mm, depends on 'dim')

Description: Similar to caliboffset, this instruction specifies an extra position offset for the 'rm' instruction. The axis travels this extra distance away from the upper limit switch EE and sets upper limit position there. The unit depends on the current 'dim' settings. Valid position range is "0.0 to 100mm equivalent". The default value is 0.

Response: Range Measure RM switch position offset

Examples:

```
!roffset 1 1 1 set the range measure offset to 1 (mm at dim 2) for X, Y and Z
!roffset z 0.1 set the range measure offset to 0.1 (mm at dim 2) for Z only
?roffset      read range measure position offset of all axes
?roffset z    read range measure position offset of Z axis only
```

## 14.9. caldir (Calibration Direction)

Syntax: !caldir or ?caldir

Parameter: x, y, z, a or none  
0 or 1

Description: This instruction set the calibration direction to either positive or negative positions. Default is negative direction. If set to positive(=1), the upper software limit is set. This instruction is not possible for systems with encoders.

Response: 0 = cal move to negative direction  
1 = cal move to positive direction

Examples:

```
?caldir          read calibration directions of all axes
!caldir y 1      set Y axis calibration direction to positive
!caldir 0 0 1    set Z axis calibration direction to negative
```

## 14.10. calbspeed (Calibration Speed for Retraction)

Syntax: !calbspeed or ?calbspeed  
Parameter: x, y, z, a or -1  
0.1 to 8000 [\*0.01 revolutions/s]

Description: **(Available in EXTMODE=0 only, else use calvel/rmvel)**  
Set or read the cal/rm calibration speed which is used for traveling out of the limit switches E0 and EE.

Remarks: RESTRICTIONS APPLY DUE TO BACKWARDCOMPATIBILITY. See examples. Setting the calbspeed without specifying an axis will set all axes to this one value. It is recommended to access axes individually. Refer to examples below. Calbspeed is only available in **extmode=0**. For improved behavior and flexibility please refer to the **calvel**, **rmvel** instructions, which become available with **extmode=1** and replace the calbspeed and vel.

Response: Currently used calibration back speed [in 1/100 motor rev/s]

Examples:  
!calbspeed 50 50 *ILLEGAL INSTRUCTION!*  
!calbspeed 15 *COMPATIBILITY RESTRICTIONS! 0.15 [revolutions/s] for all axes!*  
?calbspeed *COMPATIBILITY RESTRICTIONS! returns one parameter!*

?calbspeed -1 read the the limit switch retraction speed of all axes  
!calbspeed z 20 set the limit switch retraction speed for Z only (recommended)  
?calbspeed x read the limit switch retraction speed for X (recommended)

## 14.11. calrefspeed (Reference Signal Calibration Speed)

Syntax: !calrefspeed or ?calrefspeed  
Parameter: x, y, z, a or -1  
0.1 to 8000 [\*0.01 revolution/s]

Description: Reference mark calibration speed. This speed is used when searching the encoder reference mark. The default value is 32 (0.32 rev/s). There is only one value for all axes.

Remarks: RESTRICTIONS APPLY DUE TO BACKWARDCOMPATIBILITY. See examples. **It is recommended to use the new instruction 'encrefvel'.**

Response: Currently used calrefspeed [in 1/100 motor rev/s]

Examples:  
!calrefspeed 5 5 *ILLEGAL INSTRUCTION!*  
!calrefspeed 15 *COMPATIBILITY RESTRICTIONS! 0.15 [revolutions/s] for all axes!*  
?calrefspeed *COMPATIBILITY RESTRICTIONS! returns one parameter!*

?calrefspeed -1 read the the referencing speed of all axes  
!calrefspeed z 20 set the referencing speed for Z only (recommended)  
?calrefspeed x read the referencing speed for X (recommended)

## 14.12. encrefvel (Encoder Ref Signal Calibration Velocity)

Syntax: !encrefvel or ? encrefvel

Parameter: x, y, z, a or none  
0.000001 to 3000 [rev/s] (or [mm/s] if dim=9)

Description: Replaces the !calrefspeed instruction.  
Set or read the velocity, at which the reference mark search is performed (during !cal, after releasing the limit switch).

Response: Currently used encrefvel in [rev/s] or [mm/s] when dim=9

Examples:

```
!encrefvel 5 5 5 Set encoder reference mark search velocity to 5 (mm/s @ dim=9)
!encrefvel z 0.5 Set encoder reference mark search velocity for Z axis to 0.5
?encrefvel Read velocities of all axes
?encrefvel y Read velocities of Y axis only
```

## 14.13. refdir (Direction for Searching Reference Signal)

Syntax: !refdir or ?refdir

Parameter: x, y, z, a or none  
0 or 1

Description: DUMMY INSTRUCTION.  
Specifies or reads the direction in which to search the reference switch [REF]. This switch is not available with Tango controllers.

Response: 0 = search in negative direction (default)  
1 = search in positive direction

Examples:

```
!refdir y 1 set the Y-axis reference search to positive direction
!refdir 1 1 0 set the reference search direction of X, Y and Z axis
?refdir read reference search directions of all axes
?refdir x read reference search direction of X axis only
```

## 14.14. calpos (Calibration Position)

Syntax:                   !calpos or ?calpos  
Parameter:               x, y, z, a or none  
                          position value (unit depends on 'dim')

Description:            Used, in combination with an encoder, to measure calibration position accuracy (repeatability of the origin). During calibration the encoder signal period, where the limit switch E0 was released and the origin (pos=0) was set, is stored. This position value within an encoder period can be read with '?calpos'.  
The position may also be set to another value. The unit depends on the setting of 'dim'. Valid range is 0 to 100mm equivalent.

Remarks:               This instruction is used for systems with encoders only.

Response:               A position within the range of one encoder signal period

Examples:  
?calpos y               read calibration position of Y axis (e.g. returns 0.0000)  
?calpos                 read calibration position of all axes  
!calpos 0 0 0           set calibration positions to zero (X, Y and Z)  
!calpos y 0             set calibration position of Y axis to zero



## 14.15. calvel (Calibration Velocities for CAL Instruction)

Syntax: !calvel or ?calvel  
Parameter: x, y, z, a or none  
two velocities >0.0 (in [motor rev/s] or [mm/s] if dim=9)

Description: **This instruction is accessible in EXTMODE=1 only.**  
If 'extmode' is set to 1, this instruction replaces the 'calbspeed' and 'vel' parameters for the calibration (!cal) function:

Parameter 1 = speed towards cal limit switch (find)  
Parameter 2 = speed out of cal limit switch (release)

The unit is [motor rev/s] for 'dim' settings 0 to 8, and [mm/s] fro dim=9.

The travel speed towards the limit switch should not be more than 10mm/s to prevent mechanical damage (axis must stop after sudden limit switch event).

The travel speed out of (releasing) the limit switch should be low for achieving high position accuracy, e.g. 0.5mm/s

Response: Two velocities (towards and out of endswitch) per axis

Examples:

!calvel x 10 0.5 Cal in X moves towards endswitch with velocity 10 [rev/s] or [mm/s], depending on dim and out of the endswitch with velocity 0.5  
?calvel read cal velocities of all axes  
?calvel y read cal velocities of Y axis only (e.g. returns 10.000 0.500)

## 14.16. rmvel (Range Measure Velocities for RM Instruction)

Syntax: !rmvel or ?rmvel  
Parameter: x, y, z, a or none  
two velocities >0.0 (in [motor rev/s] or [mm/s] if dim=9)

Description: **This instruction is accessible in EXTMODE=1 only.**  
If 'extmode' is set to 1, this instruction replaces the 'calbspeed' and 'vel' parameters for the range measure (!rm) function:

Parameter 1 = speed towards rm limit switch (find)  
Parameter 2 = speed out of rm limit switch (release)

The unit is [motor rev/s] for 'dim' settings 0 to 8, and [mm/s] fro dim=9.

The travel speed towards the limit switch should not be more than 10mm/s to prevent mechanical damage (axis must stop after sudden limit switch event).

The travel speed out of (releasing) the limit switch should be low for achieving high position accuracy, e.g. 0.5mm/s

Response: Two velocities (towards and out of endswitch) per axis

Examples:  
!calvel x 10 0.5 Rm in X moves towards endswitch with velocity 10 [rev/s] or [mm/s], depending on dim and out of the endswitch with velocity 0.5  
?calvel read rm velocities of all axes  
?calvel y read rm velocities of Y axis only (e.g. returns 10.000 0.500)

## 14.17. autopitch (Measure Pitch after CAL Instruction)

Syntax: !autopitch or ?autopitch  
Parameter: x, y, z, a or none  
0 or 1

Description: Measures and sets the spindle pitch each time when executing a cal instruction.

Remarks: Only works if encoders are present.  
Not recommended for spindles and precision drives.

Response: Autopitch enabled (1) or disabled (0, default)

Examples:  
!autopitch 1 1 0 Measure and readjust pitch after each cal instruction X and Y  
!autopitch y 1 Measure and readjust pitch after each cal instruction in Y  
?autopitch read autopitch setting of all axes  
?autopitch x read autopitch setting of X axis

Pitch measuring sequence example:  
"!autopitch x 1"  
"!cal x"  
[wait for reply]  
"!autopitch x 0"  
"?pitch x"  
"!save"

## 15. Move Instructions

Move instructions command the Tango to move axes to certain positions or to travel at a constant, specified velocity. Positioning can be executed for individual axes or combined as a vector move.

Positioning (moa, mor, m, moc) is based on the velocity (vel) and acceleration (accel, accelfunc) settings. If executed as a vector of 2 or more axes, the Tango automatically selects the "leading" axis and adjusts the other axes in a way that none of their velocity and acceleration parameters is exceeded.

Moa and mor are similar instructions. Moa travels based on absolute coordinates, defined by the axis origin or the overwritten "pos", while mor travels relative to the current position.

The "m" instruction can be used to achieve high vector throughput with little communication overhead (often combined with autostatus=3). The relative distances have to be preset - either the last "mor" or a "distance" instruction.

A special case of positioning is available with the !go instruction. It does not move as a vector, here each axis travels at its own velocity and accel settings. Also it only provides linear acceleration (accelfunc does not apply).

The advantage of the !go instruction is that it can be overwritten at any time with no need to abort the currently executed move. It will smoothly change directions. Target applications are e.g. focusing via a slider or tracking a mouse cursor position.

The third option for axis travel is "speed move". Here not positions but velocities are specified. The addressed axes travel at this velocities until stopped, aborted or reaching a position limit. Like !go, the speed instruction is also not executed as a vector and does not use the accelfunc. Speed requires the "joystick" to be enabled.

### Remarks on relative positioning:

Due to internal resolution a sequence of many consecutive relative moves may lead to (minor) absolute position deviation. Executing an absolute move at times is recommended.

Also, if HDI is enabled, minor changes in position may occur due to the connected device (Joystick, ErgoDrive). Which can also accumulate position error when only using relative moves. It is recommended to deactivate the HDI when using relative moves (by instructions **joy** or **joydir**).

## 15.1. moa (Move Absolute)

Syntax: !moa or moa  
Parameter: x, y, z, a or none  
position values within  $\pm$ maxpos

Description: This instruction commands one or more axes to the specified position(s). The position unit depends on **dim** settings.

Response: Depends on **autostatus** settings, which per default is set to 1. Each commanded (and enabled) axis responses either '@' after successfully completing the move, or 'E' if an error occurred. For further information on response options, please refer to **autostatus** and **statusaxis**.

Examples:  
moa 10 0 20 axes X,Y,Z travel to the specified positions (vector move)  
moa 10 0.5 axes X and Y travel to the specified positions (vector move)  
moa x 10.2 X-axis travels to position (e.g. 10.2mm if dim=2)  
moa 10.2 same as "moa x 10.2"  
moa y 34.5 Y-axis travels to position (e.g. 34.5mm if dim=2)

## 15.2. mor (Move Relative)

Syntax: !mor or mor  
Parameter: x, y, z, a or none  
distance values within  $\pm$ maxpos

Description: This instruction commands one or more axes relative to the current position. The position unit depends on **dim** settings.

Response: Depends on **autostatus** settings, which per default is set to 1. Each commanded (and enabled) axis responses either '@' after successfully completing the move, or 'E' if an error occurred. For further information on response options, please refer to **autostatus** and **statusaxis**.

Examples:  
mor 10 0 -20 axes X,Y,Z travel the specified distances (vector move)  
mor 10 0.5 axes X and Y travel the specified distances (vector move)  
mor x 10.2 X-axis travels the specified distance (e.g. 10.2mm if dim=2)  
mor 10.2 same as "mor x 10.2"  
mor y -34.5 Y-axis travels e.g. (if dim=2) 34.5mm backwards

### 15.3. m (Move Relative Shortcut)

Syntax: !m or m  
Parameter: none

Description: The instruction is a shortcut (abbreviation) of mor. It is useful to speed up communication especially for consecutive identical vectors. The vector is taken from the preceding !mor or !distance instruction. The instruction will move all enabled axes if their distance is not zero.

Response: Depends on state of autostatus, recommended is mode 3.

Example: Positioning sequence involving moa,mor,m

!moa 1 2 3 4	will move to 1 2 3 4	
!mor 1 1 1 1	will move to 2 3 4 5	(mor sets distance)
m	will move to 3 4 5 6	
!distance 0 2 0 0		(specify distance)
m	will move to 3 6 5 6	
m	will move to 3 8 5 6	

### 15.4. distance (Distance for m)

Syntax: !distance or ?distance  
Parameter: x, y, z, a or none  
Distance (within +-maxpos)

Description: This instruction sets the travel distance for !m instructions. The unit depends on the 'dim' settings.

Remarks: The distance value is also set by each '!mor' instruction. Distances must be defined for all axes, axes that shall not move have to be set to distance = 0.

Response: Currently used value for distance (unit depends on 'dim')

Examples:

?distance	read distance values of all axes
?distance z	read distance value of Z axis only
!distance 10 20 0	set X and Y distance
!distance 1 2 0.5	set X, Y and Z distance
!distance y 20.2	set Y distance only, other axes keep their distance values

### 15.5. moc (Move to Center)

Syntax: !moc or moc  
Parameter: x, y, z, a or none

Description: The specified or all enabled axes travel to the center position between their lower and upper software limit. It is recommended to first execute the !cal and !rm instructions.

Response: Depending on 'autostatus' settings, each successful centered axis responds with "@".

Examples:

moc	all axes travel to their center position
moc y	Y-axis travels to the center position

## 15.6. go (Go To Position)

Syntax: !go or go  
Parameter: x, y, z, a or none  
position values within  $\pm$ maxpos

Description: Used for position tracking applications.  
Similar to the 'moa' instruction, 'go' executes a move of one or more axes to an absolute position.

The differences to absolut move instructions are:

- Go can be overwritten anytime, also when moving, by another go position (without the need to abort or stop it first)
- Go is no vector move, each axis moves at its own velocity '**vel**' and acceleration
- It supports only linear acceleration
- No autostatus reply on completion (polling of '**statusaxis**' required)

The unit of the position values depends on **dim**.

Remarks: In order to check for a completed go move, please poll the '**statusaxis**' state, which should not be 'M' then.

Response: None.

Examples:  
go 10.7 14 axes X and Y travel to the specified positions (no vector)  
go x 10.2 the X-axis travels to position 10.2 ([mm] assume dim=2)  
go 10.2 same as "go x 10.2"  
go 10.1 -0.5 0 axes X, Y, Z travel to the specified positions (no vector)

## 15.7. speed (Speed Move)

Syntax: !speed or ?speed  
Parameter: x, y, z, a or none  
+-100.0 [rev/s] or [mm/s] in dim=9 only

Description: Start axes to travel at the specified velocities. Travel can be stopped by setting the speed to zero. Or when reaching a soft- or hardware limit.

The speed instruction is also called "digital Joystick", therefore it only applies when joystick is enabled: **joy, joydir**

Remarks: For endless rotation please refer to '**!zero**' and **modulomode=1**.

Response: Currently executed speed in [rev/s], or [mm/s] if dim=9  
There is no autostatus reply.

Examples:  
!speed 33 0.01 start speed move for X= 33[rev/s] and Y= 0.01[rev/s]  
!speed 10 start speed move for X-axis to 10[revolutions/s]  
!speed y 0.001 start speed move for Y-axis  
!speed 0 0 0 0 stop speed move for all axes  
!speed 0 stop speed move for X-axis  
!speed z 0 stop speed move for Z-axis  
?speed read the currently executed speed of all axes  
?speed z read the currently executed speed of Z-axis only

## 15.8. a (Abort the Current Move)

Syntax: !a or a

Parameter: x, y, z, a or none

Description: This instruction stops either all axes or the specified axis and sets them into position reached state. Sending a "Ctrl+C" (hex 0x03) will stop all axes as well.

Remarks: Abort might fail in special cases of closed loop errors. In such case closed loop has to be deactivated as well.

Response: Depends on the instruction being executed (moa,speed,go etc.) and **autostatus**. If a move was aborted in autostatus=1 (default) and all axes stopped, each axis responds an ,@'.

Example: a (abort move of all axes)  
a y (abort move of Y axis only)

## 15.9. delay (Set the Delay Time for Consecutive Moves)

Syntax: ?delay or !delay

Parameter: 0 to 10000 [ms]

Description: This instruction will insert a delay time before executing a move (delayed start). There is only one value for all axes. Applies to: moa,mor,moc,m

Response: Delay time in [ms]

Examples:

!delay 500 Delay the start of a move instruction by 0.5 seconds

?delay Read the delay time

## 15.10. pause (Set the Pause after Position Reached)

Syntax: ?pause or !pause

Parameter: 0 to 10000 [ms]

Description: Complementary to "delay", this instruction adds a pause time after the axes have reached their target positions. In autostatus=1 mode the "@@@" response is delayed by this time. It may be used to insert an automatic settling time after moa,mor,moc or m. There is only one value for all axes.

Response: Pause time in [ms]

Examples:

!pause 10 Delay the autostatus response of a move by 10 milliseconds

?pause Read the pause time

## 15.11. pos (Read or Set Position)

Syntax:           !pos or ?pos  
Parameter:       x, y, z, a or none  
                  Position (within +-maxpos)

Description:      This instruction either reads or sets the axis position.  
                  If set, this defines a new absolute position of the axis.  
                  The unit depends on the selected dimension (**dim**).

Remarks:         For axes with encoders, the encoder position can be returned  
                  by setting its **'encpos'** to 1.

                  The effect of manipulating positions with "!pos" can be  
                  removed by the instruction "posclr".

Response:        Axis position(s) (depends on **dim** and enc+encpos state)

Examples:         ?pos                Read all axis positions  
                  ?pos z               Read Z axis position only  
                  !pos 100 200        Set the current X and Y axis positions  
                  !pos -0.1           Set the current X position to -0.1 (unit depends on dim)  
                  !pos y 2000         Set the current Y position to be 2000 (unit depends on dim)

## 15.12. posclr (Clear Position Offset)

Syntax:           !posclr or ?posclr  
Parameter:        x, y, z, a or none

Description:      Reset or read the position offset to the axis origin, which  
                  was added by a !pos instruction.

                  The absolute position can be redefined by "!pos". In order to  
                  return to the original absolute position, clearpos offers  
                  reading or clearing of the changes made by !pos.

Response:        Position offsets (depending on **dim**)

Examples:         !posclr x           (reset X to original position)  
                  !posclr             (reset all positions to original position)  
                  ?posclr             (read position offset of all axes)

                  ?pos                => 1.0000 **2.0000** 3.0000  
                  !pos y 8            (*here a position offset of 8-2=6 is added to Y*)  
                  ?pos                => 1.0000 **8.0000** 3.0000  
                  ?posclr            => 0.0000 **6.0000** 0.0000  
                  !posclr            (*here the !pos set position offsets are removed*)  
                  ?pos                => 1.0000 **2.0000** 0.0000



## 15.13. zero (Set Internal Position to Zero)

Syntax:               !zero or zero  
Parameter:           x, y, z, a or none

Description:         Unlike "!pos 0", the "!zero" instruction also resets the internal position counter to zero. It has to be used in applications where axes exceed the position limits, e.g. filter wheels (in such case a "!pos 0" instruction is not sufficient). The zero instruction should be executed after completing one or several complete revolutions, before reaching the software limits. So the reference point remains at the same position.

Response:            none.

Examples:  
!zero                 Set all internal positions to zero  
!zero z              Set Z axis position to zero

## 15.14. clearpos (Set Internal Position to Zero)

Syntax:               !clearpos or clearpos  
Parameter:            x, y, z, a or none

Description:         For compatibility with LStep controllers. Functionality is almost the same as with the 'zero' instruction. The only difference is that the clearpos instruction is not executed when in closed loop.

Response:            none.

Examples:  
!clearpos            Set all internal positions to zero  
!clearpos x          Set X axis position to zero

## 16. HDI: Joystick, Tackball and Handwheel Instructions

The HDI (human device interface) provides manual control of the axes.

The velocities are limited by the **secvel** velocity as long as no **cal** and **rm** sequence has been executed. The axis travel will stop at either the hardware limit switches or the adjustable software limits (**lim**).

The velocities are either taken from the current axis velocity **vel** or, if '**extmode**' is enabled as an independent **joyvel**.

The HDI interface tolerates hot plugging of the devices. It is possible to unplug, plug or change the input devices during operation of the controller.

The **keymode** functionality enables selection of different **keyspeed** or **zwtravel** wheel velocities by pressing the function keys of the joystick. Please refer to **keymode** for further informations.

The optional **z-wheel**, found on several devices, can be assigned to any axis. The default assignment is axis 3 (Z) and can be changed by **zwaxis**.

### 16.1. joy (Generally Enable/Disable HDI)

Syntax:               !joy or ?joy  
Parameter:            0, 1, 2, 3, 4 or 5

Description:           Generally enable or disable the connected HDI device (joystick, trackball, ErgoDrive etc.)  
It is recommended to only use the values 0 or 2. For compatibility, a value of 1 has the same effect as 2.

0   OFF - disable HDI device  
2   ON  - enable HDI device (default setting)

If joy is switched from an ON state to OFF (0), an automatic status message like "@@@" is generated. If this is not wanted a workaround is using '**joydir**' to disable the HDI or temporarily disabling autostatus.

Remarks:             This also enables or disables the '**speed**' instruction.

Response:            HDI enable state

Examples:

!joy 0                disable the HDI device (e.g. joystick) and speed instruction  
!joy 2                enable the HDI device (default) and speed instruction  
?joy                  read HDI enable state

Behavior examples / sequences:

?joy ==> 2           (HDI is enabled)  
!joy 0 ==> @@@-     (response when switched to disable, default autostatus on)  
!joy 0               (no response when joy already is disabled)  
!joy 2               (no response when joy is enabled)

!autostatus 0        (workaround #1 to avoid response when disabling)  
!joy 0  
!autostatus 1

!joydir 0 0 0 0      (workaround #2 by instead using joydir)  
!joydir 2 2 2 2      (make sure the direction is correct, 2 or -2)

## 16.2. joydir (Joystick Direction or Assign Joystick per axis)

Syntax: !joydir or ?joydir  
Parameter: x, y, z, a or none  
and 0, 1, 2, -1, -2

Description: In addition to the '**joy**' instruction, joydir can be used to enable/disable individual HDI axes and set their directions. It is recommended to use the values 2, 0 or -2 only. The options are:

0 = Disable HDI axis (e.g. joystick deflection is ignored)  
1 = Enable HDI axis, no motor current reduction  
2 = Enable HDI axis, current reduction supported (default)  
-1 = Same as 1, direction reversed  
-2 = Same as 2, direction reversed

Remarks: Please also make sure that the joystick function is globally enabled by the '**joy**' instruction.

When using a 4 axis controller with a 3 axis HDI device, the 3<sup>rd</sup> axis must be assigned to axis 3(=default setting), 4 or both (3 and 4) by enabling/disabling their joydir!

This instruction also enables or disables the '**speed**' move for individual axes, but does not change the speed directions.

Response: HDI directions of the axes or specified axis

Examples:  
!joydir -2 enable HDI X-axis in reversed direction  
!joydir z 0 disable HDI Z-axis  
!joydir 2 2 0 2 set positive direction, allow current reduction, assign the joysticks 3<sup>rd</sup> axis to the controller A axis instead of Z  
?joydir read HDI enable/direction setting of all axes  
?joydir y read HDI enable/direction setting of Y axis only

## 16.3. joychangeaxis (Change Joystick X and Y Axis)

Syntax: !joychangeaxis or ?joychangeaxis  
Parameter: 0 or 1

Description: Change the assignment of the Joystick X and Y axes.

0 = no change (default)  
1 = Joystick X and Y axes changed (X=Y, Y=X)

Remarks: Only for Joystick devices.

Response: Joystick X-Y change setting

Examples:  
!joychangeaxis 1 change X and Y axis of the Joystick  
?joychangeaxis read Joystick X,Y change state

## 16.4. joywindow (Joystick Window)

Syntax: !joywindow or ?joywindow  
Parameter: 0 to 100

Description: This instruction sets the center position threshold of the Joystick in digits. A deflection, as long as it is in this window, has no effect. There is only one value for all axes. The default value of +/- 14 should not be reduced, as this may result in slow unwanted creeping of axes even when the joystick is apparently not deflected. Increasing the value will reduce the velocity resolution (available steps).

Response: joywindow [in digits]

Examples:  
?joywindow read joystick window  
!joywindow 14 set joystick window to +-14

## 16.5. joyvel (Joystick Velocity)

Syntax: !joyvel or ?joyvel  
Parameter: x, y, z, a or none  
0.000001 to 200 [revolutions/s] or equiv. [mm/s] if dim=9

Description: **This instruction is accessible in extmode 1 only!**  
In extmode=1 this instruction must be used to set the joystick velocities. As the vel instruction then has no influence to the joystick velocity. In normal mode (extmode=0) the joystick velocities are derived from the axis vel settings.

Response: Currently used joystick velocities

Examples:  
!joyvel 12.5 20 0.4 Set joystick velocities for 3 axes  
!joyspeed z 1 Set joystick velocities for z to 1 [rev/s], (e.g. dim=2)  
or [mm/s] if dim=9  
?joyvel x Read joystick velocity of X-axis

## 16.6. joyspeed (Joystick Speed Presets for BPZ Device)

Syntax: !joyspeed or ?joyspeed  
Parameter: 1, 2 or 3 and  
0.000001 to 200 [revolutions/s]

Description: **Only used by a customer designed HDI device** (called BPZ), this instruction sets the joystick speeds for the three speed buttons (Slow, Medium, Fast). Unit is in motor revolutions per second (like 'vel' instruction). While the velocity applies to all axes, each speed button has to be set individually:  
1 = Slow Button speed, one parameter for all axes  
2 = Medium Button speed, one parameter for all axes  
3 = Fast Button speed, one parameter for all axes

Response: Speed currently assigned to the specified button in [rev/s]

Examples:  
?joyspeed 1 Read "Slow" joystick button speed  
!joyspeed 3 30 Set "fast" joystick button speed to 30 [revolutions/s]

## 16.7. keymode (Joystick Key Mode)

Syntax: !keymode or ?keymode

Parameter: 0, 1 or 2

Description: Assign keyspeed values to the Joystick buttons. The Joystick can be used with two different velocity settings slow/fast. In keymodes 1,2 'vel' or 'joyvel' instructions have no effect. In such case refer to '**keyspeed**', which will be used then. Please note that other special functions which require Joystick buttons (e.g. **snapshot modes**) should not be used at the same time as keymode.

A) If Joystick toggle mode ('**hdimode**') is selected:

--> F1 toggles XY between the keyspeed values

--> F4 toggles Z keyspeeds (available from firmware 1.56)

B) If '**hdimode**' is not set to toggle mode, the behavior is:

Pressing F1 selects the fast keyspeed values of X and Y axis, while F4 selects the slow keyspeed values of X and Y axis.

Pressing F2 selects the fast keyspeed value of the Z axis, while F3 selects the slow keyspeed value of the Z axis.

In case of joysticks with the optional '**z-wheel**', the wheel velocities can be selected by pressing F1, F4, or no key. Please refer to the '**zwtravel**' description.

Recommended Joystick settings are:

2/3 axis Joystick: hdimode 0 or 1 to either toggle velocities by pressing F2/F3, F1/F4 (0) or to toggle by pressing F1 (1).

Joystick with wheel: hdimode 0 to toggle XY by F2,F3 and the wheel with pressing none/F1/F2.

Parameter description:

0 = Normal key functions

1 = X/Y and Z Joystick velocity, initial value: slow [F4,F3]

2 = X/Y and Z Joystick velocity, initial value: fast [F2,F1]

Response: keymode as decimal number

Examples:

!keymode 1 slow preset keymode

?keymode => 0 (in case keymode is disabled)

## 16.8. keyspeed (Joystick Key Speed Presets)

Syntax: !keyspeed or ?keyspeed

Parameter: x, y, z, a or none  
0.000001 to 3000 [mm/s]

Description: Two Joystick velocities can be set for each axis individually. The first parameter is the slow value and the second parameter is the fast. Unit is always mm/s, independent from 'dim'. In keymode 1 or 2 the X and Y values (slow/fast) are assigned to F4 and F1, while the Z values are assigned to F3/F2. Please also refer to 'keymode'.

Response: Two floating point values per axis (slow fast)  
single axis : => [slow] [fast]  
multiple axes: => [slow\_x] [fast\_x] [slow\_y] [fast\_y] ...

Examples:

```
?keyspeed x      => 1.00 10.00 (Read X Joystick button velocity)
?keyspeed        => 1.00 10.00 1.00 10.00 0.10 1.00 (Reply of 3 ax. controller)
!keyspeed z 0.1 1 (Set fast Joystick button speed to 0.1 and fast to 1 [mm/s])
!keyspeed 5 20 2 10 0.2 2 (Set 3 axes at once)
```

## 16.9. joycurve (Joystick Characteristic)

Syntax: !joycurve or ?joycurve

Parameter: x, y, z, a or none  
0, 1, 2

Description: The speed characteristic of Joystick deflection can be independently defined for each axis.

0 = Logarithmic (default)  
1 = Linear  
2 = Quadratic

Response: Currently used characteristic

Examples: !joycurve 0 0 0 => set X,Y,Z axes to logarithmic  
!joycurve z 1 => set Z axis to linear  
?joycurve => read characteristic of all axes

## 16.10. key (Read HDI Device Key State)

Syntax: ?key or key

Parameter: none or key number 1, 2, 3, 4

Description: This instruction reads the state of up to 4 HDI device keys.

0 = key is currently released or not available

1 = key is currently pressed

Response: 1 or 4 Key states, each either 0 or 1

Examples: key => query all keys, returns 4 numbers, e.g. 0 0 0 0

key 1 => query only key 1 (e.g. F1 Joystick button)

key 3 => query only key 3 (e.g. F3 Joystick button)

## 16.11. keyl (Read HDI Device Latched Key State)

Syntax: ?keyl or !keyl

Parameter: none or key number 1, 2, 3, 4

Description: The ?keyl or keyl instruction read the latched state of the specified or all 4 HDI device keys and clears their latched state. The latch state is cleared after reading.

The Instruction !keyl clears the latched state of the specified or all keys to zero (0) without reading.

0 = key is is/was released since last key or keyl instruction

1 = key is is/was pressed since last key or keyl instruction

Response: 1 or 4 Latched key states, each either 0 or 1

Examples: keyl => read+clear all 4 keys, returns e.g. 0 1 0 0

keyl 1 => read+clear only key 1 (e.g. F1 Joystick button)

?keyl 1 => same as "keyl 1"

!keyl 1 => clear latch state of key 1 only (to zero)

!keyl => clear latch state of all 4 keys (to zero)

## 16.12. hwfactor (Handwheel Transmission Factor)

Syntax: !hwfactor or ?hwfactor  
Parameter: x, y, z, a or none  
and -200.0 to 200.0

Description: The handwheel transmission factor defines the axis travel distance in millimeter per handwheel knob revolution, which is a floating point number between -200.0 and +200.0. Negative factors reverse the travel direction (as joydir -2). Higher factors result in a more coarse resolution, a typical handwheel provides about 100000 steps per revolution.

Response: Currently used handwheel factor(s)

Examples:  
!hwfactor 14 14 => One knob revolution in X or Y results in 14mm axis travel  
!hwfactor x 100 => One knob revolution in X results in 100mm travel  
?hwfactor => Read transmission factor of all axes

## 16.13. hwfactorb (Alternate Handwheel Factor)

Syntax: !hwfactorb or ?hwfactorb  
Parameter: x, y, z, a or none  
and -200.0 to 200.0

Description: Similar to '**hwfactor**', this instruction gives access to the alternate (second) parameter for travel distance per knob revolution. Which is available with the ErgoDrive. Negative factors reverse the travel direction.

Response: Currently used alternate handwheel factor(s)

Examples:  
!hwfactorb 26.6 26.6 => One knob revolution in X or Y results in 26.6mm travel  
?hwfactorb y => Read alternate transmission factor of Y axis only

## 16.14. hwfilter (Handwheel Noise Filter)

Syntax: !hwfilter or ?hwfilter  
Parameter: 0 or 1

Description: This instruction sets or reads the handwheel noise filter state.

1 = Noise filter is active (recommended, default)  
0 = Noise filter is deactivated (finer step resolution)

The filter can only be activated/deactivated for all axes. Disabling the filter may result in a finer resolution. But it also may result in some inaccuracy between automatic moves, as its signal noise will cause a permanent slight position jitter.

Response: Current state of handwheel filter

Examples:  
!hwfilter 0 => No noise filter for handwheel, increased finer resolution  
?hwfilter => Read hwfilter state





## 16.15. **tbfactor (Trackball Factor)**

Syntax: `!tbfactor` or `?tbfactor`

Parameter: `x, y, z, a` or none  
and `-200.0` to `200.0`

Description: This instruction sets or reads the trackball sensitivity (transmission factor), which is a floating point number between `-200.0` and `+200.0`. A negative value can be used to change direction (works like `'joydir'`). The default factor is 1.

Response: Currently used trackball factor(s)

Examples:

```
!tbfactor x 100 => X axis is 100 times more sensitive than the default setting  
!tbfactor y 12.5 => X axis is 12.5 times more sensitive than the default  
!tbfactor 0.5 0.5 => X and Y axis set to half the default sensitivity  
?tbfactor => Read sensitivity factor of all axes
```

## 16.16. zwheel (Is Z-Wheel Available)

Syntax: ?zwheel or zwheel  
Parameter: none

Description: Identify if the connected HDI device provides a Z-Wheel.

0 = HDI device has no Z-Wheel  
1 = HDI device has a Z-Wheel

Response: 0 or 1

Example: ?zwheel => 0

## 16.17. zwtravel (Z-Wheel Travel per wheel revolution)

Syntax: !zwtravel or ?zwtravel  
Parameter: 1, 2 or 3 and  
-50.0 to 50.0 [mm/revolution]

Description: This instruction sets the travel distances for one revolution of the "Z-Wheel" knob, found on several HDI devices, e.g. ErgoDrive and some Joysticks. Please check that 'secvel' and 'vel' or 'joyvel' do not prevent from faster traveling when turning the Z-Wheel.

1 = Default (used when no HDI function key is pressed)  
2 = Used while Joystick F4 button is pressed (suggested slow)  
3 = used while Joystick F1 button is pressed (suggested fast)

Presets for travel distance are  
1: 0.1 mm/rev  
2: 0.01 mm/rev  
3: 1.0 mm/rev

Remarks: If necessary, the default travel may be set to zero (0) for security reasons. So the axis will move only when a key is pressed (F1, F4).

It is also possible to set negative values for direction change. While the 'joydir' instruction changes the direction in general.

The Z-Wheel can also be assigned to other axes, anyway it is named Z because this is the default axis. Default, slow and fast is the intended use of the 3 available velocities, but might be assigned as required (slow can be faster than fast etc.)

Response: The travel distance(s) currently assigned to the specified function.

Examples:  
?zwtravel Read all 3 travel distances: [default] [slow] [fast]  
?zwtravel 1 Read "default" Z-Wheel travel distance  
?zwtravel 2 Read "slow" Z-Wheel travel distance  
!zwtravel 3 2.5 Set F1 "fast" Z-Wheel travel to 2.5 [mm/revolution]  
!zwtravel 1 0 Set "default" velocity to zero

## 16.18. zwaxis (Z-Wheel Axis)

Syntax: !zwaxis or ?zwaxis

Parameter: x, y, z or a

Description: Assign Z-Wheel to axis (default: z)

Response: x, y, z or a

Example: !zwaxis a (assign z-wheel to axis 4)  
!zwaxis x (assign z-wheel to axis 1)  
?zwaxis => z (z-wheel is currently assigned to axis 3)

## 16.19. zwfactor (Z-Wheel Factor)

Syntax: !zwfactor or ?zwfactor

Parameter: 0, 1, 2 ... to 20

Description: Increase Z-Wheel transmission multiplier, default=1.  
May be used for Encoder wheels of customer devices  
which provide less resolution (pulses per revolution).

0 = Z-Wheel has no effect  
1 = Z-Wheel default (1:1)  
...  
20 = Z-Wheel travels 20 times more distance

Response: Multiplier

Example: !zwfactor 1 (set default multiplier, as for Tango HDIs)  
!zwfactor 5 (set multiplier for lower res. Encoder wheel)  
?zwfactor (Read the currently used multiplier)

## 16.20. tvrjoy (Pulse and Direction Joystick Functionality)

Syntax: !tvrjoy or ?tvrjoy  
Parameter: 0, z, a

Description: This instruction enables and assigns the AUX-IO pulse and direction input to an axis for simple joystick functionality. The behavior is similar to the trackball, which is available as HDI device.  
Important: This option must not be used for absolute positioning of axes by an external controller. Please use the tvr functionality for this applications.

0 = Disabled (default)  
z = Assigned to Z-axis  
a = Assigned to A-axis

Response: Currently assigned axis

Examples:  
!tvrjoy 0           Disable AUX-IO tvr joystick function  
!tvrjoy z           Assign AUX-IO tvr joystick function to Z-axis  
?tvrjoy             Query assigned axis

## 16.21. tvrjoyf (Pulse and Direction Joystick Factor)

Syntax: !tvrjoyf or ?tvrjoyf  
Parameter: -200.0 to +200.0

Description: This instruction sets or reads the tvrjoy transmission factor, which is a floating point number between -200.0 and +200.0. A sign change may be used to change direction. The default setting is 1.

Response: Currently used tvr factor

Examples:  
!tvrjoyf 10         Axis is 10 times more sensitive than the default setting  
?tvrjoyf            Read tvrjoy transmission factor

## 16.22. hdi (Read HDI ID)

Syntax: ?hdi or hdi  
Parameter: none

Description: This instruction reads the ID number of the connected hdi device.  
The second value shows how good the hardware ID code matches the theoretical ID value [in %]. This value should be more than 30 for secure device identification.

ID range = 0,1,2, ... 16 (=no device connected)  
ID match = 0 (poor) ... 100 (good)

ID	DEVICE
0	Special device (reserved)
1	Coaxial drive (handwheel)
2	Application specific
3	ErgoDrive
4	SmartMove device
10	2x2 Axis Joystick or 4 axis jogbox
11	Trackball + 2 Axis Joystick
12	Joystick 2 Axis
13	Trackball + 3 Axis Joystick
14	Trackball
15	Joystick 3 Axes
16	No device connected
17	(Identification in progress)
18	(Initialization in progress)

Remarks: The instruction may be used to identify the connected HDI device. In addition '?zwheel' can be checked to identify if the device also provides a Z-Wheel.

Response: HDI ID number and the hardware coded ID match in percent.

Example: ?hdi => 12 97 (hdi device nr. 12, 97% match)



## 16.23. hdimode (HDI Mode Options)

Syntax: ?hdimode or !hdimode

Parameter: Set LSB or more bits at once: string of 0s and 1s,  
or single bit with two numbers: 0 to 15 and 0 or 1

Description: This instruction provides access to extended HDI device options.

Options may either be set by a string of bits (0s and 1s) or by specifying bit number and logic state (on/off = 1/0). The string is LSB first (bit 0 is the first and leftmost).

Bit	Function
0:	Toggle Mode for ErgoDrive (0=off, 1=on)
1:	Toggle Mode for Joystick (in KeyMode 1 or 2) 0=select KeySpeed velocity XY with F1+F4, Z with F2+F3 1=toggle KeySpeed velocity XY by just pressing F1 from firmware 1.56 button F4 toggles Z
2:	- reserved -
3:	- reserved -
4:	- reserved -
5:	- reserved -
6:	- reserved -
7:	- reserved -
8:	- reserved -
9:	- reserved -
10:	- reserved -
11:	- reserved -
12:	- reserved -
13:	- reserved -
14:	- reserved -
15:	- reserved -

Response: Single mode bit or all 16 mode bits as ASCII string

### Examples:

```
!hdimode 100010 Set mode bits 0 and 4 to "on", bits 1,2,3,5 to "off". Bits
6...15 are left unchanged.
!hdimode 0 1 Set mode bit 0 to 1 (on) = ErgoDrive Toggle Mode selected
!hdimode 5 1 Set mode bit 5 to 1 (on)
!hdimode 3 0 Set mode bit 5 to 0 (off)
!hdimode 1111 Set mode bits 0,1,2,3 to 1 (on)
?hdimode Read the current state of all mode bits (returns 16 digits)
?hdimode 0 Read the current state of mode bit 0 (ErgoDrive toggle mode)
```



## 16.24. configaxsel (Joystick Axis Select Option)

Syntax: !configaxsel or ?configaxsel

Parameter: 0 or 1

Description: Used in Tango 4 axes systems.  
Enable the axis select functionality when the joystick Z-axis should drive both, the controller Z and A axes.  
If the A axis joystick is enabled (see remarks), A can be driven instead of Z by pressing F4 key of the joystick.

1 = axis select enabled (pressing F4 key → A-axis used)  
0 = axis select disabled (default, joystick z always z axis)

Remarks: Please make sure that the joystick function for A axis is enabled (`'joydir a'` instruction)

Response: Axis select configuration

Examples:

!configaxsel 1 Axis select enabled (Z<->A with 3 axes joystick)

!configaxsel 0 Axis select disabled (default)

?configaxsel Read the axis select configuration (returns 0 or 1)

## 17. Digital and Analogue I/O

The Tango provides several digital I/O, two analogue outputs (channel 0 and 1) and one analogue input. These are available on the optional auxiliary I/O port. Furthermore the HDI Interface analogue inputs can be read, making some of them optional analog 0-5V inputs if no HDI-device is connected.

### 17.1. digin (Digital Input)

Syntax:           ?digin or digin  
Parameter:       none or 0 to 15

Description:       Only available with I/O extension board.  
This instruction reads the logic state of one or all digital inputs. If no parameter is used all inputs are returned as a string of 16 characters, ASCII 0 or 1, LSB (channel #0) first.

Response:         logic state of digital inputs

Examples:  
?digin             read all 16 digital inputs (response e.g. 0000000000000000)  
?digin 8           read logic level of digital input 8 (response e.g. 1)

### 17.2. digout (Digital Output)

Syntax:           !digout oder ?digout  
Parameter:       Set LSB or more bits at once:   string of 0s and 1s,  
                  or single bit with two numbers: 0 to 15 and 0 or 1

Description:       Only available with I/O extension board.  
This instruction sets or reads back the logic level of the optional digital outputs.  
Outputs may be set either by a string of levels (0s and 1s) or by channel number and signal level.  
The string is LSB first (channel 0 is the leftmost).

Response:         current output state

Examples:  
!digout 11110000   The digital outputs 0,1,2,3 are set to logic ,1' and the outputs 4,5,6,7 are set to logic ,0'. Outputs 8...15 are left unchanged.  
!digout 5 1         set digital output 5 to logic 1 (high)  
!digout 7 0         set output 7 to 0 (low)  
?digout            read the state of all outputs  
?digout 8           read the state of output 8



## 17.3. adigin (AUX-I/O Digital Input)

Syntax:               ?adigin or adigin  
Parameter:           none or 0 to 3

Description:         Available with the AUX-I/O connector.  
This instruction returns the logic state of one or all digital inputs on the optional AUX-I/O connector. If no parameter is used all inputs are returned as a string of 4 characters, ASCII 0 or 1, LSB first:

~~0 = Bit 0 = AUX-I/O Pin 1 (Takt In) may not be available!~~  
1 = Bit 1 = AUX-I/O Pin 2 (V/R In)  
2 = Bit 2 = AUX-I/O Pin 3 (Stop)  
3 = Bit 3 = AUX-I/O Pin 4 (SnapShot2)

Response:            logic state of digital inputs

Examples:

?adigin               read all (4) AUX-I/O digital inputs (response e.g. 1111)  
?adigin 3             read AUX-I/O digital input 3 ("SnapShot2", response e.g. 1)

## 17.4. adigout (AUX-I/O Digital Output)

Syntax:               !adigout oder ?adigout  
Parameter:           Set LSB or more bits at once:   string of 0s and 1s,  
  or single bit with two numbers: 0 to 3 and 0 or 1

Description:         Available with the AUX-I/O connector.  
This instruction sets or reads back the logic level of the AUX-I/O digital outputs.  
Outputs may be set either by a string of levels (0s and 1s) or by individual channel number and signal level:

0 = BIT0 = AUX-I/O Pin 5 (TAKT\_OUT) may not be available!  
1 = BIT1 = AUX-I/O Pin 6 (VR\_OUT)  
2 = BIT2 = AUX-I/O Pin 7 (SHÜTTER\_OUT)  
3 = BIT3 = AUX-I/O Pin 8 (TRIGGER\_OUT)

The string is LSB first (channel 0 is the leftmost).

Some outputs might not be available here when trigger modes are activated.

Response:            Output state(s), 0 or 1

Examples:

!adigout 1011         Digital outputs 0,2,3 are set to high, output 1 is set to low  
!adigout 10           Digital outputs 0 and 1 are set to logic 1(BIT0) and 0(BIT1),  
  outputs 2 and 3 are left unchanged  
!adigout 1 0          set digital output 1 to logic 0  
!adigout 2 1          set digital output 2 to logic 1  
?adigout              read the level of all outputs (e.g. returns 0000)  
?adigout 3            read the level of output 3       (e.g. returns 0)

## 17.5. anain (Analogue Input)

Syntax: ?anain  
Parameter: c (c = channel)  
0 to 15 (channel number)

Description: This instruction reads the current value of one analogue input channel. The range is decimal from 0 (=0V) to 1023 (=5V).

Channel No	Connector	Pin	Signal Name
0	HDI	1	Joystick X
1	HDI	2	Joystick Y
2	HDI	3	Joystick Z
3	HDI	4	
4	HDI	5	Speedpoti
5	HDI	6	
6	HDI	7	
7	HDI	8	
8	HDI	9	
9	HDI	10	HDI-ID
<b>10</b>	<b>AUX-IO</b>	<b>9</b>	<b>ANAINO</b>
11	internal	-	(PSE)
12	internal	-	V-MOT
13	EXT	20	X-ID0
14	EXT	18	X-ID1 / temp
15	internal	-	REF (2.5V)

Calculating the internal motor voltage:

$U_{mot}[V] = (5 / 1023) * [anain\ c\ 12] * (55.7/4.7)$   
More accurate:  
 $U_{mot}[V] = (2.5 / [anain\ c\ 15]) * [anain\ c\ 12] * (55.7/4.7)$

Calculating the internal PSE voltage:

$U_{mot}[V] = (5 / 1023) * [anain\ c\ 11] * (14.7/4.7)$   
More accurate:  
 $U_{mot}[V] = (2.5 / [anain\ c\ 15]) * [anain\ c\ 11] * (14.7/4.7)$

Calculating the case temperature (if available):

$T[°C] = (250 / [anain\ c\ 11]) * [anain\ c\ 14]$

Example:  
?anain c 10      Read value of channel 10 (analogue input of AUX-IO connector)



## 17.6. anaout (Analogue Output)

Syntax:                   !anaout or ?anaout  
 Parameter:               0 to 100 in percent   (100% = 10V)  
                           c                                       (c = single channel keyword)  
                           0, 1 or 2                           (single channel number)

Description:            This instruction sets and reads the analog output signal levels in percent. There are two ways to access, with or without the 'c' keyword (see examples below). This allows to access either a single channel by using the 'c' or channel 0 up to all channels by directly sending the percent values. The signal resolution is typically 14 bit, or 12 bit on older devices (Tango PCI-4).

Fractional numbers may be used, 100% corresponds to 10 Volts.

Channel No.	Connector	Pin	Signal Name
0	AUX-IO	10	ANOUT0
1	AUX-IO	11	ANOUT1
2	reserved	-	-

Response:               Analogue output signal level in percent

Examples:  
 !anaout 100 50.1   Set channel 0 = 100% (10V) and channel 1 = 50.1% (5.01V)  
 !anaout 75        Set channel 0 = 75% (7.5V)  
 !anaout c 1 25.3   Set channel 1 to 25.3% (2.53V)  
 ?anaout            Read output level of all channels   (e.g. 0.00 0.00 0.00)  
 ?anaout c 0        Read output level of channel 0 only (e.g. returns 100.00)

## 17.7. stoppol (Mode and Polarity of Stop Input Signal)

Syntax: !stoppol or ?stoppol

Parameter: 0 to 5 and 8 to 13

Description: Operating mode of the AUX-I/O "Stop" input.  
The modes offer selection of high and low polarities as well as different behaviors like: stop on signal, sticky stop, HDI remaining on or also stopped, immediatly abort running move or complete move first then stop.

**0,1 Stop only as long as stop signal is applied**

**HDI (joystick) remains active!**

0 : active low

1 : active high

**2,3 Stop only as long as stop signal is applied**

**HDI (joystick) is also disabled**

2 : active low

3 : active high

**4,5 Stop signal is latched (sticky), must be released by**

sending a "!stop 0" instruction

HDI (joystick) is also disabled

4 : active low

5 : active high

*6,7 Not available*

**8,9 Same as 0,1 but a running move will be completed first**

**10,11 Same as 2,3 but a running move will be completed first**

**12,13 Same as 4,5 but a running move will be completed first**

Requirements: In order to be detected, A stop signal must be applied for at least 50µs.

Remarks: Usually the stop input has an internal pull-up resistor to +5V, please refer to the corresponding Tango operating manual.

Response: Operating mode of AUX-I/O stop signal input

Example:

!stoppol 5 Set the function of the AUX-I/O stop input to latched stop active high.

## 17.8. stop (Release, Force or Check Stop Condition)

Syntax:            !`stop` or `?stop`  
Parameter:        0, 1

Description:       Release or force stop condition in latched '`stoppol`' modes 4, 5, 12 and 13. Or read if stop is active.

0 = Release stop condition (for `stoppol` 4,5,12,13 only)  
1 = Force stop condition (for `stoppol` 4,5,12,13 only)

Response:         -

Example:           !`stop` 0     release a latched stop  
                  !`stop` 1     force a stop, in latched `stoppol` modes only  
                  ?`stop`       read if stop is currently active (=1)

## 17.9. shutter (Shutter Out Signal of AUX-IO)

Syntax:           !`shutter` or `?shutter`  
Parameter:        0, 1

Description:       Set the AUX-IO shutter out signal to the desired TTL level:

0 signal = low  
1 signal = high

Response:         Output level of shutter signal

Example:           !`shutter` 1     Set the shutter out signal to TTL high state

## 17.10. flash (Defined Pulse at AUX-IO Takt Out)

Syntax:           !`flash` or `?flash`  
Parameter:        +`-0.00001 ... 32500` [ms]

Description:       Sends a pulse of defined length to the AUX-IO TAKT\_OUT pin.  
Used e.g. for LED strobes.

Floatingpoint numbers in [ms].  
Range 0.00001 (10ns) to 32500 (32.5s).  
Resolution is 1/132µs.

Pulse Polarity depends on sign:

- Positive numbers generate an active high pulse
- negative numbers generate an active low pulse

For safe operation it is recommended to once send one dummy pulse when initializing in order have the correct polarity.

Remarks:         **Only available with PCI/PCI-E based controllers and Tango DT.**  
Might interfere with secondary trigger output.

Response:         None

Example:           !`flash` 0.001     (high pulse with duration of 1µs)  
                  !`flash` -0.01     (low pulse with duration of 10µs)

## 18. Encoder Instructions

The encoder interface supports incremental encoders with or without a reference mark. The type of encoder (analog 1Vpp, analog MR or RS422/TTL) should be configured by factory, as it might require a different hardware. For Tango Desktop and PCI/PCI-E it is possible to switch from any analog (1Vpp or MR) interface to a digital RS422 interface by setting the `encttl` parameter to 1.

To enable encoder functionality, first the **encoder mask** has to be set for the corresponding axes. After that, depending on **calmode**, the encoders (`enc=1`) or also the closed loop will either be activated by calibration of the axis or by power-on.

Manually setting the encoders 'enc' state to 1 is not recommended, as it possibly causes unpredictable in closed loop mode. Also in case of analog MR encoders the signal correction will not be performed, which leads to positioning errors.

### 18.1. encmask (Encoder Mask)

Syntax:                    !`encmask` or ?`encmask`  
Parameter:                x, y, z, a or none  
                            0 or 1

Description:             The instruction reads or sets the encoder globally enable mask, which is required to activate the encoders.  
                            The encoders then will be detected and activated after a successful calibration instruction '`cal`' or in automatic Closed Loop activation modes (**calmode**) after power up.

                            0 = clear enable mask (encoder will not be checked/activated)  
                            1 = set enable mask

Response:                Encoder enable mask

Example:  
!`encmask` 1 1 0        Globally enable encoders for X, Y and disable Z-axis  
!`encmask` z 0         Globally disable encoder for Z-axis  
?`encmask`             Read encoder mask state of all axes

## 18.2. enc (Encoder Active)

Syntax:               ?enc (or !enc)  
Parameter:           x, y, z, a or none  
                      0 or 1

Description:        This instruction may be used to query if the encoders are active (e.g. successfully activated by a '**cal**' instruction). It is not recommended to manually activate the encoders by sending any of the "!enc 1" instructions. For error free Closed Loop behavior and best measuring accuracy, encoders must be activated by the Tango controller. This depends on '**calmode**', '**cal**' and '**encmask**'. For further details please refer to the above mentioned instruction descriptions.

0 = Encoder is inactive (not used)  
1 = Encoder is active (used)

Response:           Encoder active state

Example:  
?enc                 Read encoder active state of all axes  
?enc y              Read encoder active state of Y-axis  
!enc z 0            Disable encoder of Z-axis  
!enc 1 1 0          Manually activate encoders of X, Y and disable Z-axis  
!enc x 1            Manually activate (not recommended!) the X-axis encoder

## 18.3. encperiod (Encoder Signal Period)

Syntax:               !encperiod or ?encperiod  
Parameter:           x, y, z, a or none  
                      0.000002 to 4.0 [mm]

Description:        This instruction reads or sets the encoder signal period. The unit is always [mm].

Optional read-resolution: As an option to read the parameter with higher precision, the number of required decimal places can be specified with the query "?encperiod [0...16 decimal places]". If no precision is defined, the default resolution is 4 decimal places.

Response:           Encoder signal period(s)

Example:  
!encperiod 0.5 0.5 0.001     Set encoder period for X and Y to 500µm, Z to 1µm  
!encperiod z 0.02            Set encoder period of Z-axis to 20µm  
!encperiod 0.00001960784     Set encoder period of X-axis  
?encperiod                    Read encoder period of all axes  
?encperiod z                  Read encoder period of Z-axis  
?encperiod 12                 Read period of all axes with 12 fractional digits  
?encperiod z 9                Read period of Z-axis with 9 fractional digits

## 18.4. encdir (Encoder Counting Direction)

Syntax:               !encdir or ?encdir  
Parameter:           x, y, z, a or none  
                      0 or 1

Description:         Encoder counting direction.  
Do not set this parameter when closed loop is active!  
The encoder direction is set automatically by the Tango controller before activating the closed loop (e.g. after calibration 'cal' or power-on).  
Only if the axis is not used for closed loop (e.g. only for relative distance measuring) the encdir may be set manually.

0 = Encoder counting direction default  
1 = Encoder counting direction reversed

Response:            Encoder counting direction

Example:  
!encdir 1 1 1         Reverse encoder counting direction for all axes  
!encdir x 1          Reverse encoder counting direction for X-axis only  
?encdir              Read encoder counting direction of all axes  
?encdir y            Read encoder counting direction of Y-axis only

## 18.5. encvel (Encoder Auto-Adjust Velocity)

Syntax:               !encvel or ?encvel  
Parameter:            x, y, z, a or none  
                      0.01 ... 20.0 [mm/s]

Description:         The velocity for encoder auto-calibration can be set or read by this instruction. It is recommended to keep the default setting. The unit is always [mm/s].

Response:            Velocity used for Encoder detection and calibration in [mm/s]

Example:  
!encvel 0.5 0.5 0.5   Set encoder auto-adjust velocity for all axes  
!encvel 0.5            Set encoder auto-adjust velocity for X-axis only  
!encvel z 0.5         Set encoder auto-adjust velocity for Z-axis only  
?encvel                Read encoder auto-adjust velocity of all axes  
?encvel y              Read encoder auto-adjust velocity of Y-axis only



## 18.6. encctl (Encoder has TTL Signal)

Syntax: !encctl or ?encctl  
Parameter: x, y, z, a or none  
0 or 1

Description: This instruction reads or sets the type of encoder signal. If digital encoders (A/B-TTL,RS422) are used with an analog encoder interface (configured for 1Vpp or 5Vpp MR), the corresponding encoder has to be set to TTL mode. Else the TTL signal will be found as invalid (due to signal level) and not be used or deactivated during operation (sporadic malfunction).

0 = Encoder has analog sin/cos signals  
1 = Encoder has digital quadrature A/B signals (e.g. RS422)

Response: Currently selected encoder signal type(s)

Example:  
!encctl 0 0 1 Y and Y axis encoders are analog, Z is digital A/B encoder  
!encctl z 1 Set Z encoder signal processing to digital  
?encctl Query all axes for their currently used signal type  
?encctl x Query X-axis for its currently used signal type

## 18.7. encref (Use Encoder Reference Signal)

Syntax: !encref or ?encref  
Parameter: x, y, z, a or none  
0 or 1

Description: Use encoder reference mark. If enabled the 'cal' instruction will, after reaching the lower limit switch, travel to the reference mark and set the axis position to zero.

0 = Encoder reference signal not used  
1 = Encoder reference signal used for calibration

Response: Encoder reference signal used, not used

Example:  
!encref 1 1 0 Use encoder reference signal as origin for X and Y-axis  
!encref y 1 Use encoder reference signal as origin for Y-axis  
?encref Read Encoder reference signal usage of all axes  
?encref z Read Encoder reference signal usage of Z-axis

## 18.8. encnas (Use Encoder NAS Error Signal)

Syntax: !encnas or ?encnas  
Parameter: x, y, z, a or none  
0 or 1

Description: Before enabling this functionality please make sure that the connected encoder provides a NAS error signal. If enabled, a encoder NAS error also generates an internal 'err' error state. The NAS input signals an encoder error state by a TTL low level.

0 = NAS encoder input state is ignored (default)  
1 = NAS encoder input signal is used for extended error detection

Response: Encoder NAS signal used / not used for error detection

Example:  
!encnas 1 1 0 Use encoder NAS signal for X and Y-axis  
!encnas x 1 Use encoder NAS signal for Y-axis  
?encnas Read encoder NAS signal use state of all axes  
?encnas x Read encoder NAS signal use state of X-axis

## 18.9. encrefstatus (Encoder REF Signal State)

Syntax: ?encrefstatus or encrefstatus  
Parameter: x, y, z, a or none

Description: Returns the REF signal input state.

0 = REF signal is inactive  
1 = REF signal is active (encoder is on a reference mark)

Response: Encoder reference signal state

Example:  
encrefstatus Read REF signal state of all axes  
encrefstatus x Read REF signal state of X-axis only

## 18.10. encrefstatusl (Latched Encoder REF Signal State)

Syntax: ?encrefstatusl or encrefstatusl  
Parameter: x, y, z, a or none

Description: Returns the latched REF signal input state. If the REF signal was active since last reading of the encrefstatusl, a 1 is returned.

0 = REF signal is/was inactive  
1 = REF signal is/was active (encoder is/was on a reference mark)

Response: Latched encoder reference signal state

Example:  
encrefstatusl Read+clear latched REF signal state of all axes  
encrefstatusl x Read+clear latched REF signal state of X-axis only

## 18.11. encnasstatus (Encoder NAS Error Signal State)

Syntax: ?encnasstatus or encnasstatus  
Parameter: x, y, z, a or none

Description: Returns the NAS error signal input state.

0 = NAS signal is inactive (encoder signals 'no error')  
1 = NAS signal is active (error flag is set by encoder)

Response: Encoder NAS error signal state

Example:

encnasstatus Read NAS signal (error) state of all axes  
encnasstatus x Read NAS signal (error) state of X-axis only

## 18.12. encerr (Encoder Error State)

Syntax: !encerr or ?encerr  
Parameter: x, y, z, a or none  
0

Description: This instruction reads or resets the encoder error state. On error (low signal amplitude '**encamp**', or NAS error flag) the encoder signal is invalid and the closed loop for the corresponding axis is switched off.

0 = No error, normal function  
1 = Encoder error

Response: Encoder error state

Example:

!encerr 0 Reset encoder error  
?encerr Read encoder error states of all axes  
?encerr x Read encoder error states of X-axis only

## 18.13. encamp (Encoder Signal Amplitude)

Syntax: ?encamp  
Parameter: x, y, z, a or none  
Optional parameter 1

Description: Read the encoder signal amplitude. 100 (percent) represents the maximum undistorted signal amplitude.

Remarks: In case of single ended TTL encoders the amplitude might be returned as 0.

Response: Encoder signal amplitude in percent as integer

Example:

?encamp Read all encoder signal amplitudes (returns e.g. 57 74 0)  
?encamp x Read X encoder signal amplitude  
?encamp 1 Read all amplitudes with 1 fractional digit (57.3 73.8 0.5)  
?encamp x 1 Read X encoder signal amplitude with 1 fractional digit (57.3)

## 18.14. encpos (Encoder Position)

Syntax: !encpos or ?encpos  
Parameter: x, y, z, a or none  
0 or 1

Description: Set or read the position source for a **?pos** instruction. If set to 1 and the encoder is active (corresponding **'enc'=1**), **?pos** returns the encoder position, else the motor position. Refer to the **'pos'** and **'enc'** instructions for further information.

Remarks: For compatibility, sending a 0 or 1 without specifying an axis applies the setting to all axes.

Response: Position output source  
0 = pos instruction returns motor position (default)  
1 = pos instruction returns encoder position (if enc. active)

Example:  
!encpos 1 from now on a **'?pos'** instruction returns the encoder position for all axes (if the corresponding encoders are activated)  
~~!encpos 1 1 1~~ Invalid!  
!encpos x 1 from now on a **'pos'** instruction returns the encoder position for the X-axis (if the encoder is activated)  
?encpos Use not recommended: Due to compatibility this instruction reads the "ored" position output source of all axes (returns just one 0 or 1)  
?encpos x Read position output source of X-axis

## 18.15. hwcount (Hardware Counter)

Syntax: ?hwcount or hwcount  
Parameter: x, y, z, a or none

Description: Hwcount returns the position(s) of the independent TTL encoder counter. It is a digital counter that counts the signal slopes (4 per period) and does not provide signal interpolation. So one signal period corresponds to a counter reading of 4. Also refer to the **'clearhwcount'** instruction.

Response: Encoder hardware counter value(s)

Example:  
hwcount Returns the position counter of all axes  
hwcount x Returns the position counter of X-axis only

## 18.16. clearhwcount (Clear Hardware Counter)

Syntax: !clearhwcount or clearhwcount  
Parameter: x, y, z, a or none

Description: Reset the hardware counter(s) to zero.

Response: Reset encoder hardware counter value(s)

Example:  
clearhwcount Reset hwcount position of all axes to zero  
clearhwcount x Reset hwcount position of X-axis to zero

## 19. MR Encoder Instructions

### 19.1. mra (MR Amplitude Correction Factor)

Syntax: !mra or ?mra  
Parameter: x, y, z, a or none  
0.8 to 1.2

Description: This instruction reads or sets the cosine amplification correction factor of the analogue encoder signal (here: sin/cos amplitude ratio). This factor is calculated automatically on each calibration move 'cal' and should not be changed. If the axis is manually controlled and only used for relative measurement, so that no 'cal' is possible, the user may determine the ratio itself and then write it into mra for more accurate results. Please also refer to the 'mro' instruction.

Response: Currently used correction factor(s)

Example:  
?mra Read MR signal correction factor of all axes  
?mra x Read MR signal correction factor of X-axis only  
!mra x 1.0095 Amplify the X cosine signal by \*1.0095 compared to the sine

### 19.2. mro (MR Offset Correction Value)

Syntax: !mro or ?mro  
Parameter: x, y, z, a or none  
-2048 to +2048

Description: This instruction reads or sets the sine and/or cosine offset compensation value as 16bit signed digits. This factor is calculated automatically on each calibration move 'cal' and should not be changed. If the axis is manually controlled and only used for relative measurement, so that no 'cal' is possible, the user may determine the offset itself and then write it into mro for more accurate results. Please also refer to the 'mra' instruction.

Response: Currently used correction values

Example:  
?mro Read MR signal offset value sine and cosine for all axes  
?mro x Read MR signal offset value sine and cosine for X-axis only  
!mro 48 -100 0 0 0 0 0 Set X offset to sin=48digit, cos=-100digit, Y, Z = 0  
!mro y 16 -28 Set Y offset to sin=16digit, cos=-28digit  
!mro y 16 Set only sine offset of Y encoder

### 19.3. mrp (MR Signal Peak-To-Peak Measuring Result)

Syntax: !mrp or ?mrp  
Parameter: x, y, z, a or none  
-2048 to +2048

Description: This instruction reads or sets the sine and/or cosine peak values, measured since they were reset the last time. It is just a measurement and has no effect to the signal processing itself. The returned values are signed 16bit digits.

Response: [sine max] [sine min] [cosine max] [cosine min] result(s)

Example:  
?mrp x Returns [x\_sin max] [x\_sin min] [x\_cos max] [x\_cos min]  
?mrp Returns the above, but for all axes (up to 16 values)  
!mrp x 0 0 0 0 Reset the peak-to-peak measurement for x  
!mrp x 0 0 Reset only the X sine min, max values  
!mrp 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Reset measurement for all 4 axes

### 19.4. mrt (MR Signal Level)

Syntax: ?mrt  
Parameter: x, y, z, a or none  
1 to 32767 and -1 to -32767

Description: This instruction reads the corrected sine and cosine A/D converter results of the analog encoder interface as signed 16 bit integer.

in case of MR encoders, specifying a negative number returns the corrected analog value (offset, amplitude).

The number of data samples (lines) to read should be specified, e.g. '?mrt 1' for returning one sample. If there is no number specified, the instruction returns 10 sampling results per default.

Response: [sine] [cosine] results as signed 16 bit values  
or [x\_s] [x\_c] [y\_s] [y\_c] [z\_s] [z\_c] for all  
Each data line is terminated by a [CR].

Example:  
?mrt Returns 10 lines with all axes (up to 6 values per line)  
?mrt 1 Returns one line with all axes (up to 6 values)  
  
?mrt x Returns 10 lines with [x\_sin] [x\_cos] signal digits  
?mrt x 1 Returns one line with [x\_sin] [x\_cos] signal digits  
?mrt y 2 Returns two lines with [y\_sin] [y\_cos] signal digits  
?mrt y 1000 Returns 1000 lines with [y\_sin] [y\_cos] signal digits  
  
?mrt -1 Returns one line, all axes with MR correction applied  
?mrt x -10 Returns 10 lines, X axis with MR correction applied

## 20. Closed Loop Instructions

The closed loop control positions the axis to the measuring system position. So the inaccuracy of the drive is compensated. The closed loop control circuit is activated by the "**ctr**" instruction. Also the encoder(s) require the enable request "**encmask**" to be set. To enter the closed loop, either a calibration "**cal**" has to be executed or the power up modes have to be set to **calmode** 1 or 2.

### Remarks:

For successfully enabling the closed loop (use the "**ctrstatus**" instructions to check if it is running) it is necessary to have the correct **pitch**, **gear** and **encperiod** settings. The detection tolerance of these parameters is about a factor of 2, else the closed loop will not be activated.

### 20.1. ctr (Control Enable)

Syntax:                   !ctr or ?ctr  
Parameter:               x, y, z, a or none  
                          0,1,2,3,4

Description:             This instruction activates the closed loop circuit.  
                          0 = Closed Loop OFF  
                          1 = Closed Loop Auto OFF each time position is reached  
                          2 = Closed Loop always ON (default, recommended)  
                          3 = (currently not supported!)  
                          4 = (currently not supported!)

Response:                Closed loop state(s)

Examples:  
!ctr 0 0 0 0             Closed loop off for all axes  
!ctr 2 2                Closed loop for X- and Y-axis permanently on  
!ctr z 1                Closed loop for Z-axis switches off after position reached  
?ctr                    Read closed loop states of all axes  
?ctr x                  Read closed loop state of X axis

## 20.2. ctrf (Control Factor)

Syntax:               !ctrf or ?ctrf  
Parameter:            x, y, z, a or none  
                          0.0 to 25.0

Description:           CTRF IS FOR COMPATIBILITY ONLY - PLEASE USE CTRFF.  
                          This instruction reads or sets the closed loop factor.  
                          Higher values result in more stiffness and faster settlement.  
                          Above a critical value this may lead to oscillation.  
                          The default factor of 2.0 mostly results in a good behavior.  
                          Hint: Using the ctrff instruction instead offers more options.

Remarks:             Even if setting the parameter supports floating point, the  
                          return value is integer (for compatibility).  
                          It is recommended to use ctrff instead of ctrf.

Response:             Closed loop factors as integers (rounded)

Examples:  
!ctrf 2 2 2            Set closed loop factor to 2 for all axes  
!ctrf x 3             Set closed loop factor for X axis to 3  
?ctrf                 Read closed loop factors of all axes (as integer)  
?ctrf y               Read closed loop factor of Y axis (as integer)

## 20.3. ctrff (Extended Control Factor)

Syntax:               !ctrff or ?ctrff  
Parameter:            x, y, z or a  
                          0.0 to 25.0  
                          0.0 to 25.0

Description:           This instruction reads or sets 2 closed loop factors per axis.  
                          Higher values result in more stiffness and faster settlement.  
                          Above a critical value this may lead to oscillation.  
                          The default factor of 2.0 mostly results in a good behavior.  
                          Important: Can only be set per axis (with x,y,z,a parameter)!

Parameter1: Is used for regulation while axis is moving  
Parameter2: Is used for regulation when axis is stopped

Parameter 2 can be set to higher values than Parameter1  
to achieve smoother axis travel while still having the  
stiffness and faster settling times at the end of a move.  
(E.g.: "!ctrff x 2 4".)

Response:             Closed loop factors (2 per axis)

Examples:  
~~!ctrff 2 2 2 2~~       Not supported!  
!ctrff x 2 4           Set closed loop factors for X axis 2(moving) and 4(reached)  
?ctrff                 Read closed loop factors of all axes (2 parameters per axis)  
?ctrff y               Read closed loop factors of Y axis only (2 parameters)



## 20.4. ctrc (Control Call)

Syntax: !ctrc or ?ctrc  
Parameter: 1 to 100 [ms]

Description: This instruction reads or sets the controller call interval. Unit is milliseconds. Only one parameter for all axes. The default interval of 5 [ms] in most cases leads to the best results. Values of less than 3 [ms] are not recommended.

Response: Closed loop control call interval in milliseconds.

Examples:

!ctrc 5 Closed loop control is executed every 5 milliseconds  
?ctrc Read closed loop call interval

## 20.5. ctrd (Control Target Window Delay)

Syntax: !ctrd or ?ctrd  
Parameter: 0 to 1000 [ms]

Description: This instruction reads or sets the control delay, which is used as a position reached criteria when in closed loop mode. The closed loop has to remain inside the target window (**twi**) for this time, until the position reached state is set. If the target window is left before the delay time is over, the time starts counting again. Please also refer to the **ctrtr** timeout, which aborts the waiting for **twi+ctrd** after a certain amount of time.

The unit is milliseconds. Only one parameter for all axes.

Response: Closed loop control delay in milliseconds.

Examples:

!ctrd 100 Closed loop controller must be in target window for 100 ms  
?ctrd Read closed loop target window delay

## 20.6. ctrtr (Control Timeout)

Syntax: !ctrtr or ?ctrtr  
Parameter: 0 to 10000 [ms]

Description: This instruction reads or sets the control timeout. It specifies the maximum time the closed loop tries to reach the desired encoder position. If the **ctrd+twi** condition could not be fulfilled within this **ctrtr** time, it will be aborted. If **ctrd/ctrtr** is used, the **ctrtr** timeout must be set to a value which is higher than the **ctrd**, typically 1 or more seconds. Unit is milliseconds. Only one parameter for all axes.

Response: Closed loop control timeout in milliseconds.

Examples:

!ctrtr 1000 Closed loop tries to reach the end position for 1 second  
?ctrtr Read closed loop timeout

## 20.7. twi (Target Window)

Syntax:                   !twi or ?twi  
Parameter:                x, y, z, a or none  
                            [value corresponding 0.00001 to 1 mm] in dim units

Description:             This instruction reads or sets the closed loop control target window width (+-). While increasing this value leads to position variance, setting a too narrow window may result in oscillation and closed loop timeouts (higher ctrd,ctrtr values necessary).  
                            The unit depends on 'dim'.

Response:                Closed loop target window.

Examples:  
!twi 0.001 0.001    Closed loop target window +-1µm (if dim=2) for X and Y-axis  
!twi y 0.005        Closed loop target window +-5µm (if dim=2) for Y-axis  
?twi                 Read target window of all axes  
?twi z               Read target window of Z-axis only

## 20.8. ctrsm (Control behavior outside Lock-in Range)

Syntax:                   !ctrsm or ?ctrsm  
Parameter:                x, y, z or none

Description:             Behavior of the closed loop circuit when outside the lock-in range (ctrs).  
                            When outside defined ctrs lock-in range, which is regarded to be an error, the Tango controller can treat this condition as follows:

0 = Do nothing (default)  
1 = Limit the closed loop velocity (avoid stalling the motor)  
2 = Disable closed loop until returning to lock-in range  
3 = Disable closed loop permanently  
4 = Axis is set to sticky stop condition (stoppol 4,5, !stop)  
5 = Switch off all power stages (like !pa 0)

Response:                Selected Behavior of the axes

Examples:  
!ctrsm z 1            Select slow mode in Z (to avoid stalling of the motor)  
!ctrsm 5 5 5         Select switch off mode for safety (e.g. collision detection)  
?ctrsm                Read all behaviors  
?ctrsm y              Read behavior of Y-axis only

## 20.9. ctrs (Control Lock-in Range)

Syntax:               !ctrs or ?ctrs  
Parameter:            x, y, z or none  
                          0.01 to [maxpos]

Description:           Lock-in range of the closed loop circuit.  
                          When the closed loop position difference exceeds this limit,  
                          the behavior defined with ctrsm is applied to the axis/axes.  
                          The unit depends on 'dim'.

Response:             Closed loop lock-in range.

### Examples:

```
!ctrs 0.5 0.5 0.2 Set lock-in range of X=0.5mm, Y=0.5mm, Z=0.2mm (if dim=2 or 9)
!ctrs z 0.1       Set lock-in range of Z=0.1mm (if dim= 2 or 9)
?ctrs            Read lock-in range of all axes
?ctrs z         Read lock-in range of Z-axis only
```

## 20.10. ctrstatus (Control Status)

Syntax: ?ctrstatus  
Parameter: x, y, z, a or none  
optional parameter: 1, 2 or 3

Description: Read the internal closed loop states of the specified or all axes. Options and responses are:

**A) Called without parameter:**

Returns the internally applied **ctr** state which is set when the closed loop gets enabled by the controller (!cal or calmode=1,2).

- 0 Closed loop permanently off
- 1 Closed loop only active while moving
- 2 Closed loop always on (default closed loop operation)
- (3 Closed loop only active while moving)
- (4 Closed loop always on)

**B) Called with parameter "1":**

Check if the closed loop is currently active.

- 0 = Closed loop not active  
(e.g. ctr=0, !cal running, encerr, limit swich, etc.)
- 1 = Closed loop active

**C) Called with parameter "2":**

Check if closed loop is in target window.

- 0 = Position outside target window
- 1 = Position in target window

**A) Called with parameter "3":**

Check if closed loop is in lock-in range.

- 0 = Position outside lock-in range
- 1 = Position in lock-in range

Response: Closed loop state.

Examples:

```
?ctrstatus Returns the internally running ctr mode of all axes
?ctrstatus y Returns the internally running ctr mode of the Y-axis

?ctrstatus 1 Returns the Closed Loop active state of all axes, e.g. "1 1 0"
?ctrstatus x 1 Returns the Closed Loop active state of the X-axis, e.g. "1"

?ctrstatus 2 Returns if Closed Loop is in target window, e.g. "1 1 0"
?ctrstatus z 2 Returns if Closed Loop of the Z-axis is in target window

?ctrstatus 3 Returns if Closed Loop is in lock-in range, e.g. "1 1 0"
?ctrstatus z 3 Returns if Closed Loop of the Z-axis is in lock-in range
```



## 20.11. ctrdiff (Control Position Difference)

Syntax: ?ctrdiff

Parameter: x, y, z, a or none

Description: This instruction returns the momentarily measured closed loop position difference between the motor and encoder position (mot.pos - enc.pos). The unit depends on **dim** settings.

Response: Momentary position difference.

Examples:

?ctrdiff Returns the position difference of all axes

?ctrdiff y Returns the position difference of the Y-axis e.g. "0.0015"

## 21. Trigger Output Functionality (option)

Trigger functionality must be configured by factory.

To find out if the Trigger functionality is accessible, use `'?det'` or `'detext'`.

The optional trigger output generates TTL output signals dependend on axis positions or as a constant frequency. It may be used for synchronization of external devices like e.g. a video camera. For Tango Desktop and PCI/PCI-E controllers the trigger output is available with the optional AUX-I/O connector. Positions or analog values can be captured on a trigger event (SnapShot Mode 8).

Before enabling the trigger function (by  `'!trig 1'`), please make sure that all trigger settings have been made.

```
Example1:  !trig 0[CR]           Disable trigger
           !trigm 0[CR]        Choose trigger mode 0
           !triga x[CR]        Choose X axis as trigger source
           !trigd 0.100[CR]     Set trigger distance to 100µm (if dim = 2)
           !trigs 400[CR]      Set trigger pulse width to 0.4ms
           !trig 1[CR]        Enable trigger and set start position
```

```
Example2:  !trig 0[CR]           Disable trigger
           !trigs 120[CR]        Set trigger pulse width to 120µs
           !trigf 2500[CR]       Set pulse frequency to 2.5kHz
           !trigm 100[CR]        Choose trigger mode 100 (periodic signal)
           !trig 1[CR]        Enable trigger
```

Optional `"trigcount 0"` instruction may be executed to reset the event counter.

### 21.1. trig (Trigger)

```
Syntax:      !trig or ?trig
Parameter:   0 or 1
```

Description: This instruction enables or disables the trigger circuit. `"!trig 1"` also sets the trigger start position.

0 = Trigger function globally disabled  
1 = Trigger function globally enabled

```
Response:   0 or 1
```

```
Examples:
!trig 1      Enable trigger circuit (also defines the start position)
?trig       Read enable state of trigger circuit
```

### 21.2. triga (Trigger Axis)

```
Syntax:      !triga or ?triga
Parameter:   x, y, z or a
```

Description: This instruction selects the axis on which to trigger.

```
Response:   x, y, z or a
```

```
Examples:
!triga y     Select Y-axis as trigger source
?triga       Read current trigger axis
```

## 21.3. trigm (Trigger Mode)

Syntax: !trigm or ?trigm  
 Parameter: 0 to 11, 100 to 105

Description: This instruction selects the required trigger mode.

Trigger Mode	Trigger Generation	Trigger Signal	Remarks
0		High active	First pulse when move starts <i>Direction forward</i>
1		High active	First pulse when move starts <i>Bidirectional</i>
2		High active	First pulse when move starts <i>Direction backward</i>
3	-- See Mode 0 --	Low active	Same as 0, signal inverted
4	-- See Mode 1 --	Low active	Same as 1, signal inverted
5	-- See Mode 2 --	Low active	Same as 2, signal inverted
6		High active	Triggers shifted by $\text{trigd}/2$ <i>Direction forward</i>
7		High active	Triggers shifted by $\text{trigd}/2$ <i>Bidirectional</i>
8		High active	Triggers shifted by $\text{trigd}/2$ <i>Direction backward</i>
9	-- See Mode 6 --	Low active	Same as 6, signal inverted
10	-- See Mode 7 --	Low active	Same as 7, signal inverted
11	-- See Mode 8 --	Low active	Same as 8, signal inverted
100	<b>Periodic trigger</b> signal the frequency can be set by "trigf" instruction	High active	Does not depend on position
101		Low active	
102	<b>Manually forced</b> trigger signals by the "trigger" instruction	High active	Does not depend on position or time
103		Low active	
104	<b>Position reached</b> trigger signal if all moves including the specified axis "triga" have completed	High active	Comes with the "@@@" autostatus response
105		Low active	



Response: Trigger mode as integer: 0 to 11, 100 to 105

Examples: !trigm 0 Set Trigger Mode 0
?trigm Read current trigger mode

21.4. trigo (Trigger Output)

Syntax: !trigo or ?trigo
Parameter: 0 to 15

Description: This instruction selects the trigger outputs.

With PCI-E based devices the secondary output provides precision options (refer to the trigb Instructions).

Values greater 6 use the secondary (TAKT\_OUT) output with the "trigb..." high precision options (precise puls width or pulse delay, precise frequency. Combining precise delay and width is not possible.

Table with 5 columns: Output / Mode, STANDARD, PREC. WIDTH2, PREC. DELAY2, PREC. FREQUENCY2. Rows include No output, Primary TRIGGER OUT, Secondary TAKT OUT, and Both TRIGGER+TAKT.

Remarks: Options depend on hardware - Tango PCI-E / DT-E based controllers with AUX-IO connector offer all features. PCI-S and DT controllers do not provide the precision Delay function and Tango mini provides none or one output.

Response: Selected trigger outputs

Examples: !trigo 0 No output signal
!trigo 1 Default trigger output (TRIGGER\_OUT)
!trigo 2 Secondary trigger output (AUX-IO TAKT\_OUT)
!trigo 3 Both trigger outputs 1:1 (TRIGGER\_OUT and TAKT\_OUT)
?trigo Read selected trigger output



## 21.5. trigs (Trigger Signal Length)

Syntax: !trigs or ?trigs  
Parameter: 0 to 2621400 [ $\mu$ s]

Description: This instruction is used to adjust the trigger pulse width from 40 microseconds to 2.6214 seconds in increments of 40. (0 = shortest trigger signal width, narrow pulse)  
If the parameter is not a multiple of 40 it will be rounded to the next lower multiple (e.g. 100 --> 80). When read back, the corrected value is returned (here: 80).

Response: 0 to 2621400 ( $\mu$ s)

Examples:  
!trigs 40 Set Trigger pulse width to 40  $\mu$ s  
!trigs 2500000 Set Trigger pulse width to 2.5 s  
?trigs Read current trigger pulse width

## 21.6. trigd (Trigger Distance)

Syntax: !trigd or ?trigd  
Parameter: >0.0 to 5000000 (unit depends on **dim** of the selected axis)

Description: This instruction sets the required trigger distance. After passing this position interval of trigd with the selected axis, a trigger signal is generated.

Response: Trigger distance

Examples:  
!trigd 3.01 Set trigger distance to 3.01mm (if dim of selected axis is 2)  
?trigd Read current trigger distance

## 21.7. trigcomp (Trigger Compensation)

Syntax: !trigcomp or ?trigcomp  
Parameter: -10000 ... +10000 [ $\mu$ s] (smallest step is 10 $\mu$ s)

Description: Delay compensation for position trigger (look ahead) to avoid stitching mismatch with bidirectional scan applications.

At high velocity "on the fly" scans, the delay of the trigger signal chain has an effect on where the sample was taken. When scanning in both directions this effect becomes visible (e.g. comb-like appearance of the stitched image). This can be greatly improved or entirely removed by increasing the delay compensation. The delay added by the Tango already is internally compensated, so the default setting is 0.

Remarks: Also check if the scan axis has a mechanical backlash, which can be compensated for open loop applications by using the backlash instruction.

Response: Compensated delay in [ $\mu$ s]

Examples:  
!trigcomp 130 Compensate trigger signal chain delay of 130 $\mu$ s  
?trigcomp Read the compensation delay (e.g. returns 0)

## 21.8. trigenc (Trigger on Encoder)

Syntax:           !trigenc or ?trigenc  
Parameter:        0 or 1

Description:       Trigger position source select, encoder or motor position.

0 = Trigger position from motor position (default)  
1 = Trigger position from encoder (true position)

Remarks:         Encoder position is only used when encoders are active.

Response:         Currently selected trigger position source

Examples:  
!trigenc 1         Trigger based on the encoder signal  
?trigenc           Read the trigger position source

## 21.9. trigf (Trigger Frequency)

Syntax:           !trigf or ?trigf  
Parameter:        0.01 to 25000

Description:       This instruction sets the frequency for periodic trigger  
output mode (trigm 100).  
The frequency resolution is 1/40 $\mu$ s.

Response:         Trigger frequency

Examples:  
!trigf 2500        Periodic trigger pulses have 2.5kHz (signal every 0.4ms)  
?trigf             Read trigger frequency

## 21.10. trigbdelay (Precise Trigger Delay for second output)

Syntax: !trigbdelay or ?trigbdelay  
Parameter: 0.00 to 32500000 [µs]

Description: Precise delay for secondary trigger output signal (TAKT\_OUT). Applies to trigger output settings !trigo 10 and 11. Unit in microseconds [µs], resolution is 1/132µs.

Remarks: Secondary trigger can either have precise width or delay.  
**Only available with Tango PCI-E based controllers.**

Response: Delay time in µs

Examples:  
!trigbdelay 0.35 Delay the secondary trigger signal by 350ns (to TRIGGER\_OUT)  
?trigbdelay Read the secondary trigger delay (e.g. returns 0.00)

## 21.11. trigbwidth (Precise Signal Width for second output)

Syntax: !trigbwidth or ?trigbwidth  
Parameter: 0.00 to 32500000 [µs]

Description: Precise width for secondary trigger output signal (TAKT\_OUT). Applies to trigger output settings !trigo 6 and 7. Unit in microseconds [µs], resolution is 1/132µs.

Remarks: Secondary trigger can either have precise width or delay.  
**Only available with Tango PCI/PCI-E based controllers.**

Response: Signal width in µs

Examples:  
!trigbwidth 5.01 Set secondary trigger signal width to 5.01µs  
?trigbwidth Read the secondary trigger signal length (e.g. returns 40.00)

## 21.12. trigbf (Precise Trigger Frequency for second output)

Syntax: !trigbf or ?trigbf  
Parameter: 0.010 ... 66000000 [Hz]

Description: Precise frequency for secondary trigger output (TAKT\_OUT). Applies to trigger output settings !trigo 14 and 15. Unit in microseconds [µs], resolution is 1/132µs.

Remarks: **Only available with Tango PCI/PCI-E based controllers.**  
Trigcount does not count the precise trigger events.

Response: Output frequency [Hz] for precise frequency output mode.

Examples:  
!trigbf 0.01 Set secondary trigger frequency to 0.01Hz  
!trigbf 50000005 Set secondary trigger frequency to 50.000005 MHz  
?trigbf Read the secondary trigger frequency (e.g. returns 1000.000)



## 21.13. trigcount (Trigger Counter)

Syntax: !trigcount or ?trigcount  
Parameter: 0 to 2147483647

Description: This instruction reads or sets the trigger event counter.

Response: Number of executed triggers

Examples:

?trigcount Read trigger count  
!trigcount 0 Clear trigger counter  
!trigcount 110 Set trigger counter to 110 counted events

## 21.14. trigger (Force Trigger Signal)

Syntax: !trigger or trigger  
Parameter: None

Description: This instruction generates a trigger output pulse. It is available in manual forced trigger modes 102 and 103 only. The pulse width depends on the "**trigs**" setting.

Response: None

Examples: trigger (Force trigger pulse now)

## 22. Snapshot -Trigger Input Functionality (option)

Snapshot functionality must be configured by factory.

To identify if the Snapshot functionality is configured, use `?det` or `detext`.

The snapshot functionality allows capturing of X,Y,Z,A axis positions by either pressing a HDI key (e.g. Joystick F2), an external signal or a trigger-out event. Hardware dependent, analog input signals from the AUX I/O connector are captured as well. The values are appended to the snapshot array. The snapshot array can also be filled, extended or manipulated by `snsa`, `!snsp` instructions.

A snapshot event can command the Tango controller to move to positions stored in the snapshot array (`snsm 1,5`). Every snapshot event moves to the next array entry and wraps around at the end. It can i.e. be used to review memory positions (points of interest) stored in `snsm 0,4` before.

Snapshot can also start a move (`moa`, `mor`, `speed`) instruction, refer to `snsm 6`.

The snapshot event can be triggered by either

- A) the Joystick key F2
- B) via the assigned SnapShot input of the optional AUX I/O connector or
- C) by a software instruction (`snse`)

Please refer to `snsm` for available snapshot modes.

The position unit depends on the selected dimension `dim`. The setting of `encpos` defines if the motor position or the measured encoder position is read from the array. A maximum of 1024 snapshot positions can be captured.

For changing the snapshot configuration please disable the snapshot (`!sns 0'`) and then re-enable it (`'!sns 1'`) after all settings have been made.

### Remarks:

Most of the snapshot settings can not be stored permanently to the controller.

### Requirements:

The Tango controller must be ordered with this function enabled, as it is not available by default or might require additional hardware (e.g. connector).

The snapshot signal must have an active and inactive time of at least 200µs each. Else the signal may not be recognized by the controller.

**Example: Three snapshot positions are captured with a 3 axis Tango**

Index	Position X	Position Y	Position Z	Position A
1	1.0000	1.2345	1.2345	0
2	2.1200	1.3520	0.9343	0
3	3.5900	1.9000	0.8341	0
4	<i>invalid</i>	<i>invalid</i>	<i>invalid</i>	<i>invalid</i>
5	<i>invalid</i>	<i>invalid</i>	<i>invalid</i>	<i>invalid</i>

...                    ...                    ...                    ...                    ...

1024	<i>invalid</i>	<i>invalid</i>	<i>invalid</i>	<i>invalid</i>
------	----------------	----------------	----------------	----------------

Here: `?snc --> 3 ?nsa 2 --> 2.1200 1.3520 0.9343 ?nsa y 1 --> 1.2345`

## 22.1. sns (Snapshot enable/disable)

Syntax:               !sns or ?sns  
Parameter:            0 or 1

Description:           This instruction globally enables or disables the snapshot functionality. Snapshot is globally enabled by default (power-on) in order to support stand-alone applications. Latched key pressed events or the sns=6 waiting state of a move is cleared when enabling/disabling the Snapshot (sns0,1).

0 = disabled  
1 = enabled

Response:            Snapshot state

Examples:  
!sns 0                Disable snapshot  
!sns 1                Enable snapshot  
?sns                  Read snapshot enable state

## 22.2. sns1 (Snapshot Level / Polarity)

Syntax:               !sns1 or ?sns1  
Parameter:            0 or 1

Description:           This instruction sets the snapshot input signal polarity.  
0 = active low (falling edge)  
1 = active high (rising edge)

Response:            Currently used snapshot polarity

Examples:  
!sns1 0              Set snapshot input to active low  
!sns1 1              Set snapshot input to active high  
?sns1                Read current snapshot input polarity

## 22.3. snsf (Snapshot Filter)

Syntax:               !snsf or ?snsf  
Parameter:            0 to 255 [ms]

Description:           This instruction reads or sets the snapshot filter time which is used to debounce the snapshot input signals. The value is in milliseconds, 0 to 255. Debouncing should be used for push-buttons, switches, light barriers etc. A time of 0 disables the filter, which might be necessary for faster, digitally generated signals. Please note that the minimum high and low times of the input signal must be greater or equal to 200µs each in order to be recognized.

Response:            Currently used snapshot filter time

Examples:  
!snsf 0              Disable input filter  
!snsf 10             Set snapshot filter time to 10 ms  
?snsf                Read snapshot filter time

## 22.4. snsrm (Snapshot Mode)

Syntax:               !snsrm or ?snsrm  
Parameter:           0, 1, ... 8

Description:        This instruction reads or sets the snapshot mode (default=0).

- 0 = Capture positions to list                               with Joystick key F2
- 1 = Move forward through position list with Joystick key F2  
(wraps around at the last element)
- 2 = Extended move with Joystick keys:
  - F1: Move backward through position list (wraps around)
  - F2: Move forward through position list (wraps around)
  - F3: Move to '**prehome**' then start of list (first element)
  - F4: Move to '**prehome**' with '**vel**' then '**home**' with '**sevel**'
- 3 = Simple continuous path control in conjunction with  
preloaded **snsa/snse** positions (dissection mode).  
The path travel velocity is set by scanvel.  
Start with Joystick key F2 or corresponding "snse 2".  
HDI must be enabled (by joy, joydir) to provide this  
functionality.
- 4 = Like mode 0, but Snapshot I/O input is used instead of F2
- 5 = Like mode 1, but Snapshot I/O input is used instead of F2
- 6 = Triggered start: Move and Speed instructions will wait  
for the I/O connector snapshot event to start moving.  
Axes must be assigned by the '**snsaxis**' instruction.
- 7 = Custom HDI mode with Prehome, Home and auto increment  
in a time interval set by the '**delay**' instruction.
- 8 = A Trigger-OUT event (not a Snapshot event) captures  
positions and the ANIN0 analogue signal (of the optional  
AUX-I/O connector, available with DT and PCI/PCI-E Tangos)

Position capture also captures the analog voltage at the AUX-I/O ANIN0 input pin. For controllers that provide this input the captured voltage can be read by sending '**?snsrv**'.

Latched key pressed events or the snsrm=6 waiting state of a move is cleared when setting the Snapshot mode.

Remarks:           F1-F4 key events can also be generated by '**snse**' 1 to 4.

A snapshot is always executed on all active axes. Capture and move. For snapshot move the position list must be filled with valid positions for all axes.

Response:           Currently selected snapshot mode

Examples:

```
!snsrm 0       Set snapshot mode to capture (with joystick key F2)
!snsrm 1       Set snapshot mode to move       (with joystick key F2)
!snsrm 2       Set snapshot mode to extended move
!snsrm 8       Set snapshot mode to capture at trigger out events
?snsrm         Read current snapshot mode
```

## 22.5. Snapshot Mode Description and Examples

Snapshot Mode 1,2,3,7 move functions, accessible via Joystick F1-F4 keys or !snse 1-4 instruction:

Key	Mode function			
	1	2	3 (DISSECTION)	7 ***
F1 =	-	previous point**	-	prehome & home
F2 =	next point	next point**	start dissection	auto inc from 1st point
F3 =	-	prehome & first point	-	pause/continue
F4 =	-	prehome & home	-	pause & previous point

Remarks: A move to the "prehome" positions always travels at secvel.

**Prehome & home** means that the axes move to prehome and after reaching directly to home. This is used to avoid possible collisions on the way to the home position.

**Prehome & first point** means that the axes move to prehome and after reaching directly to the first position in the position list. This sequence is also for collision prevention.

### \*\* SnapShot mode 2:

F3 function must be executed in order to enable the F1 and F2 functionality  
F4 disables the F1 and F2 functionality, so pressing F3 is required again

### \*\*\* SnapShot Mode 7:

1. Execute a !cal and !rm instruction to define the repeatable, absolute position (origin).
2. Define the !prehome and the !home positions.
3. Define a position list of points using the !snsa instruction.
4. Specify a user delay between the positions by using the !delay instruction.
5. Enter snapshot mode 7 by sending the !snsm 7 instruction.

All information is lost when the Tango controller is switched off or reset.

The delay also applies to all move instructions. So when leaving the snapshot mode 7 it is recommended to reset the delay value back to zero (!delay 0).

Snapshot mode 7 uses the joystick keys F1 to F4 according this specification:

- F1 = Go to load/unload position (as defined with the !home and !prehome instructions)
- F2 = Start Auto Stage Movement from Point 1 (through position list, defined by the !snsa instruction and using a delay as defined by the !delay instruction)
- F3 = Pause/Continue from current position (halt the automatic move)
- F4 = Move back one position in the list (when in pause, else it is paused after reaching the next position and then it moves back.)

instruction sequence example for snsm 7:

```
!cal
!rm
!sns 0
!delay 1000 (delay between moves e.g. 1000ms)
!prehome [Xpos] [Ypos] [Zpos] [Apos] (specify load position)
!home [Xpos] [Ypos] [Zpos] [Apos] (specify unload position)
!snsa 0 (clear the snapshot array / position list)
!snsa 1 [Xpos] [Ypos] [Zpos] [Apos] (load position list)
!snsa 2 ... (load position list) ...
... !snsa N ...
!snsm 7 (enter snapshot mode 7 for F1-F4 functionality)
!sns 1 (activate snapshot function globally)
```

(From Tango firmware 1.57 autoincrement can be aborted by the abort "a" instruction)

### SnapShot Mode 3 (dissection, simple continuous path functionality):

Typical applications are laser dissection or adhesive dosers/dispensers.

Remarks: All axes must be idle and all axis positions must be loaded with valid entries.

The simple continuous path functionality has a continuous increase in position deviation that will build up during the sequence (about 10µm). Interpreter is blocked.

```
!scanvel [mm/s] (set the path/vector velocity)
!sns 0
!snsa 0 (clear the snapshot array / position list)
!snsa 1 [Xpos] [Ypos] [Zpos] [Apos] (load position list)
!snsa 2 ... (load position list) ...
... !snsa N ...
!snsm 3 (enter snapshot dissection mode 3)
!sns 1 (activate snapshot function globally)
!snse 2 (start the dissection sequence, or press F2 key)
?err (send a request and wait for reply --> completed)
```



## 22.6. snsc (Snapshot Counter)

Syntax:                   !snsc or ?snsc  
Parameter:               none, 0 or 1 to 1024 (not greater than the current snsc count)

Description:             This instruction reads or sets the snapshot counter, which shows the snapshot array entries (counted snapshot events). This instruction may also be used to reset the counter to zero or reduce the size of an existing array.

Response:                Current snapshot array entries (number of snapshot events)

Example:  
?snsc                    Read the number array entries (e.g. captured snapshots)  
!snsc                    Clear snapshot counter to zero elements  
!snsc 0                  Clear snapshot counter (same as without parameter)  
!snsc 50                 Reduce snapshot array entries to 50 (only if the current snsc value is equal or higher than the new value)

## 22.7. snsi (Snapshot Index)

Syntax:                   !snsi or ?snsi  
Parameter:                0 to 1023 (less than the current snsc count)

Description:             This instruction reads or sets the snapshot array pointer.

In snapshot modes 1 and 5 this pointer access can be used to manipulate the array index where move target positions are read from.

Opposed to other snapshot instructions the index range here is [0...N-1] instead of [1...N]. The value must be less than the amount of **snsc** array entries.

Remarks:                The pointer is only used for the snapshot move. It does not define the array index where positions are written to. Captured positions are always appended as the last element of the array. For array manipulations by software please refer to '**snsa**' and '**snsr**'.

Response:                Current snapshot array pointer

Example:  
?snsi                    Read the current array pointer position  
!snsi 0                  Set array pointer to the first element  
!snsi 20                 Set array pointer to the 21st element

## 22.8. snsaxis (Snapshot Axis)

Syntax:                   !snsaxis or ?snsaxis  
Parameter:                x, y, z, a or none  
                            0 or 1

Description:             Defines which axes should wait in Snapshot mode 6.  
                            Please refer to '**sns**' for further information.  
                            When sending a moa, mor or speed instruction with sns 6,  
                            axes can be specified that should not execute the move until  
                            triggered by a snapshot-in event. This can be used in cases  
                            where precise start timing has to be controlled by an external  
                            signal (maybe also set the '**trigf**' value to zero then).

**1** = move or speed of this axis will wait for snapshot signal  
**0** = axis does not wait for snapshot signal (default operation)

Remarks:                If not all axes are set to waiting mode, a vector move may  
                            start some of the axes while the others are waiting.  
                            Such problem does not exist when using single axis move/speed  
                            instructions.

Response:                Wait mode setting of the axes

### Example:

```
!snsaxis 1 0 1     Set axis X, Z to waiting mode, Y does not have to wait
!snsaxis y 1       Set axis Y to waiting mode
?snsaxis           Read waiting mode of all available axes (e.g. returns "0 0 0")
?snsaxis z         Read waiting mode of Z axis only (e.g. returns "1")
```

## 22.9. snsp (Snapshot Position)

Syntax:	!snsp or ?snsp
Parameter:	x, y, z, a or none Positions (unit depends on "dim")
Description:	This instruction reads the most recently captured Snapshot Position. It can also be used to append new positions to the Snapshot position array. The snapshot counter 'snsc' will be incremented automatically. Please also refer to 'snsa'.
Remarks:	The unit of the position data depends on selected 'dim' value. Only motor positions can be read. In order to read captured encoder positions, please refer to '?snsa'.
Response:	Last captured Snapshot position of the specified axis or of all available axes
Examples:	
?snsp	Read last captured snapshot position of all axes
?snsp z	Read last captured snapshot position of Z axis only
!snsp 10 20 30	Append new motor position entry for X, Y and Z axis
!snsp y 2000	Append new position entry with Y-axis only (not recommended)

## 22.10. snsas (Snapshot Array)

Syntax:	!snsas or ?snsas
Parameter:	x, y, z, a or none -1, 0 or 1 to 1024
Description:	Read, fill, modify or clear the snapshot position array, which may contain up to 1024 elements. For reading a valid element, the index must have a value between 1 and the maximum of the snapshot counter value. Please also refer to '?snsc'. To append new position data to the array, an index of [snsc+1] or index=-1 may be used. The Snapshot counter then will be incremented by 1 automatically. Parameter: -1 = last/append, 0 = clear, 1...1024 = r/w access.
Remarks:	The unit of the position data depends on selected 'dim' value. The read position values are either motor positions or encoder positions, depending on active encoders and 'encpos' settings. Encoder position is only available with captured positions, not when the position entry was written by software (!snsp or !snsas). In this case the encoder position is read as 0 and 'encpos' should be set to 0 for reading this position.
Response:	Snapshot array position(s) in 'dim' units.
Examples:	
?snsas 1	Read all axis positions of the first snapshot entry
?snsas 33	Read all axis positions of the 33rd snapshot entry
?snsas z 99	Read the Z-axis position of the 99th snapshot entry
?snsas y -1	Read the Y-axis position of the last snapshot entry
?snsas -1	Read all axis positions of the last snapshot entry
!snsas 0	Clear the entire snapshot array
!snsas x 1 20.5	Set X position of first element to 20.5 (e.g. mm if dim=2)
!snsas 2 10 10 10 10	Set all axis positions of the second array entry to 10
!snsas -1 10 10 10 10	Append a new entry to the position array (snsc is incremented)

## 22.11. snse (Snapshot Event)

Syntax: !snse or snse  
Parameter: 1, 2, 3 or 4

Description: Executes the F-key Snapshot functions via the communication interface. Instead of pressing a Joystick button or using the AUX-I/O signal, the functions can be triggered by software:

1 = Function normally executed by pressing Joystick F1 key  
2 = Function normally executed by pressing Joystick F2 key  
or using the AUX-I/O SnapShot input  
3 = Function normally executed by pressing Joystick F3 key  
4 = Function normally executed by pressing Joystick F4 key

Remark: Behavior is the same as with the function keys. It depends on the snapshot mode settings and only works when Snapshot is enabled. Does not work with Snapshot Mode 8, as this mode requires trigger out signals being generated by the Tango.

Response: -

Example:  
snse 2 Execute F2 Snapshot event (e.g. capture current position to the snapshot position array)

## 22.12. snsV (Snapshot Voltage)

Syntax: !snsv or ?snsv  
Parameter: 1 to 1024 or -1 to -1024

Description: Read the AUX-I/O ANIN0 input voltage (0...5V), captured at the specified position array index. The value can also be written to (i.e. as temporary data storage)  
Index > 0: read Voltage in [mV]  
Index < 0: read raw ADC data

Remarks: The specified index must be in the range of 1 to +- '**snsC**'. The [mV] values are always calculated with the internal voltage reference, while the raw data is the direct ADC sampling result which may be less accurate (only PCI-E based Tangos have an accurate ADC sampling result). The raw data range is always 12bit, but with PCI-S based controllers the 2 LSBs are always zero (10bit resolution).

Response: Voltage in [mV] or in [12bit ADC raw data]

Example:  
?snsv 2 Read captured Voltage [mV] of 2<sup>nd</sup> snapshot array entry  
?snsv -2 Read captured Voltage [raw] of 2<sup>nd</sup> snapshot array entry  
!snsv 5 1234 Store the value 1234 in 5<sup>th</sup> snapshot array voltage element

## 22.13. prehome (Snapshot PreHome Position)

Syntax: !prehome or ?prehome  
Parameter: x, y, z, a or none

Description: This instruction sets the prehome position used by the snapshot extended move. The unit of the input position depends on the setting of "dim". Please refer to "snsm" 2 for more details.

Response: Position value(s)

Examples:

```
!prehome y 10.2 Set prehome position Y-value to 10.2 (e.g. [mm] when dim=2)
!prehome 10 0 20 Set prehome position X,Y,Z
?prehome x Read currently used prehome position of X axis only
?prehome Read currently used prehome positions of all axes
```

## 22.14. home (Snapshot Home Position)

Syntax: !home or ?home  
Parameter: x, y, z, a or none

Description: This instruction sets the home position used by the snapshot extended move. The unit of the input position depends on the setting of "dim". Please refer to "snsm" 2 for more details.

Response: Position value(s)

Examples:

```
!home y 10.2 Set home position Y-value to 10.2 (e.g. [mm] when dim=2)
!home 10 0 20 Set home position X,Y,Z
?home x Read currently used home position of X axis only
?home Read currently used home positions of all axes
```



## 23. Document Revision History

No.	Revision	Date	Changes	Remarks
01	A	27. March 2012	Newly revised and extended document version	Based on Tango firmware revision 1.57
02	B	16. April 2012	Added instructions: Ecomove, noled, calrequired, joychangeaxis, ctrsm, ctrs, trigo, trigcomp, trigenc, snsi, snsaxis, snsv, zwfactor, detext, posclr, maxaxis, trigbwidth, trigbdelay, trigbf, encrefvel, vrm, flash, etspresent, stagesn Added stop and stoppol modes !joy instruction autostatus behavior	Based on Tango firmware revision 1.57