The Instruction Set

# of the

TANGO Controller

MÄRZHÄUSER
WETZLAR

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

1. Table of Contents

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.

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 all axis positions
```

Moves:
Move commands are executed as a vector move, so if several axes are involved 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 may 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 initilaization 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 occure 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, too. Please refer to the **"!cts"** command 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 saveguard 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, the speed limit may be increased to up to 100mm/s at own risk. Please refer to the **"secvel"** instruction for further information.

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 velocitiy is motor revolutions per second! Only dim 9 provides the millimeter unit for most parameters[1], 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.

[1] 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

Controller Informations

Instruction		Example	Save	Brief description	Page
(?)	version	version	fix	Read detailed firmware and controller version	15
(?)	det	det	fix	Read detailed configuration information	16
?	readsn	?readsn	fix	Read the controller serial number	16
(?)	ver	ver	fix	Read default version number	17
(?)	iver	iver	fix	Read further version number information	17
(?)	uptime	uptime	-	Read how long the controller is running	17
(?)	temp	temp	-	Read case temperature (available with encoder option)	17

Communication Interface Settings

Instruction		Example	Save	Brief description	Page
? !	baud	!baud 9600	Y	Set RS232 baud rate to 9600 Bd (default=57600)	18
? !	cts	!cts 1	Y	Switch on CTS hardware handshake	18

System Instructions

Instruction		Example	Save	Brief description	Page
(!)	save	save	-	Save parameters to controller nonvolatile memory	19
(!)	restore	restore	-	Reload controller parameters from saved values	19
(!)	reset	reset	-	Reset controller (forces restart, similar to cycle power)	19
!	pa	!pa 1	-	Enable power amplifiers (disable = 0), see ‘axis’ cmd. too	20
? !	ipreter	!ipreter 2	Y	Select optional Venus instruction set	20

Operating Modes

Instruction		Example	Save	Brief description	Page
? !	extmode	!extmode 1	Y	Enable extended controller behavior	21
? !	scanmode	!scanmode 1	Y	Set positioning behavior to scanmode	22
? !	scanvel	!scanvel 20 20	Y	Set scanmode vector velocity to 20mm/s for X and Y	22
? !	modulomode	!modulomode a 1	Y	Set positioning behavior of A axis to turntable mode 1	23

Controller States and Error Messages

Instruction		Example	Save	Brief description	Page
? !	autostatus	!autostatus 0	-	Select autostatus response type 0 (=disabled), range: [0-4]	24
(?)	statusaxis	statusaxis	-	Read axis state [@,M,J,C,S,A,D,-]	25
(?)	status	status	-	Read controller error state	25
(?)	err	err	-	Read error number	26
(?)	help	help	-	Read error number with additional text description	26
(?)	service	service	-	Returns a detailed parameter and state list, for debugging	26
(?)	pci	pci	-	Returns 1 if controller is plugged in a PCI slot (desktop=0)	27
(?)	isvel	?isvel x	-	Query actual velocity of the X axis	27
?	maxpos	?maxpos x	-	Query maximal available position range for X axis	27

General Adjustments

Instruction		Example	Save	Brief description	Page
? !	dim	!dim 1 1 1	Y	Set position units of X Y Z to µm	28
? !	pitch	!pitch 1 1 1	Y	Set spindle pitch of X Y Z to 1 [mm/revolution]	28
? !	gear	!gear 1 1 1	Y	Set gear factor of X Y Z to 1	29
? !	motorsteps	!motorsteps x 200	Y	Set X axis motor has 200 steps per revolution	29
? !	accel	!accel 0.1 0.1 0.1	Y	Set acceleration of X Y Z to 0.1m/s²	30

General Adjustments

Instruction		Example	Save	Brief description	Page
? !	accelfunc	!accelfunc 1 1 0	Y	Set acceleration function X and Y to sin^2, Z to linear	30
? !	stopaccel	!stopaccel 2 2	Y	Set X and Y deceleration during stop condition to 2m/s^2	31
? !	vel	!vel 10 10 10	Y	Adjust speed of X Y Z to 10 [revolutions/s]	31
? !	velfac	!velfac 1 1 1	Y	Set velocity reduction factor for X Y Z to 1 (= no reduction), range is [0.01-1]	32
? !	secvel	!secvel x 20	Y	Set secure speed limit X to 20mm/s (unit is always mm/s)	32
?	maxcur	?maxcur	fix	Show the maximum possible motor currents of all axes	32
? !	cur	!cur 0.5 0.6 1	Y	Set motor current in Ampere: X=0.5 Y=0.6 and Z=1 A	33
? !	reduction	!reduction 0.5 0.5 0.5	Y	Select 50% motor current reduction for X Y Z	33
? !	curdelay	!curdelay 1000	Y	Delay X axis motor current reduction by 1000 [ms]	34
? !	axis	!axis 1 0 -1	Y	Enable X, disable Y and switch off Z axis	34
? !	axisdir	!axisdir 0 1 0	Y	Reverse rotating direction of Y motor (caution!)	35
? !	motortable	!motortable x 2	Y	Select custom motor correction table type 2 for X axis	35
? !	usteps	!usteps 50000	Y	Set dim 0 microsteps to 50000/rev for all axes	36
? !	resolution	!resolution 6	Y	Set position return string resolution to 1 nm	36
? !	backlash	!backlash 12.3 0 0	Y	Set backlash compensation to 12.3µm in X and 0 in Y & Z	37
!?	lock	!lock 2 1	Y	Set write protection for parameter 2 (here: motor current)	38
!?	lockaxis	!lockaxis 0 0 0 0	Y	Remove lock protection from all axes (lock has no effect)	38
?	lockstate	?lockstate x	-	Query extended locked parameters, including internal limitations currently applied to X axis	39
? !	stout	!stout 2	Y	Make Status LED state available at AUX-I/O Pin VR_OUT	39
? !	updelay	!updelay -5000	Y	Wait to a maximum of 5 seconds for valid external power	40

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	41
? !	limctr	!limctr x 1	-	Enable hardware limit switches for X axis, default = 1	41
? !	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	42
? !	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)	42
? !	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	43
? !	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	44
? !	swdir	!swdir x 1	Y	Swap reference- and endswitch assignment for X axis	43
?	readsw	?readsw	-	Read states of all limit switches (1=active and actuated)	45
(?)	swin	swin	-	Read TTL signal level of all limit switch inputs (1=high)	45
(?)	statuslimit	statuslimit	-	Read current limit status „A“ = calibration done „D“ = rm done „L“ = limit switch modified by software „-“ = not yet modified	46

Calibration and Range Measure Instructions

Instruction		Example	Save	Brief description	Page
(!)	cal	cal	-	Perform a calibration move for all enabled axes, see ‘axis’	47
(!)	rm	rm x	-	Perform a range measure move in X	48
? !	calmode	!calmode 2 2	Y	Set calibration/closed loop behavior X, Y to type 2	49
? !	caltimeout	!caltimeout 60 60 10	Y	Set calibration timeout for X and Y to 1 minute, Z to 10s	49
? !	caliboffset	!caliboffset 1 1 1	Y	Set the cal zero-point 1mm aside lower limit switch (dim 2)	50

Calibration and Range Measure Instructions

Instruction		Example	Save	Brief description	Page
? !	rmoffset	!rmoffset 1 1 1	Y	Set rm end-position 1mm aside upper limit switch (dim 2)	50
? !	caldir	!caldir z 1	Y	Calibrate the Z-axis in positive direction	50
? !	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]	51
? !	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]	51
? !	calpos	calpos	-	Read back the encoder position where the calibration switch was released	52
? !	refdir	?refdir y	Y	Read the direction for encoder reference search in Y axis	52
? !	calvel	!calvel x 10 0.5	Y	**Only if extmode = 1:** Set calibration velocities in X	53
? !	rmvel	!rmvel x 10 0.5	Y	**Only if extmode = 1:** Set range measure velocities in X	54
? !	autopitch	!autopitch x 1	Y	Measure pitch after cal move of X axis	54

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)	56
(!)	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	56
(!)	m	m	-	Move relative again (use same parameters as defined by last ‘!mor’ or ‘!distance’ instruction)	57
? !	distance	!distance 1 1 1	-	Set distance for X Y Z ‘m’-move (start with ‘m’ or ‘!m’)	57
(!)	moc	moc x	-	Move X to center position between lower and upper limit switch, or between lower and upper software limits	57
(!)	go	go x 12.5	-	Move X to pos. 12.5, overwritable, for tracking applications	58
? !	speed	!speed 5 5 5 !speed y 0	-	Digital joystick: move X Y Z axis with 5 [revolutions/s] Stop the Y axis speed move	58
(!)	a	a	-	Abort move (Stop)	59
? !	delay	!delay 1000	Y	Delay all consecutive moves by 1000 ms	59
? !	pause	!pause 10	Y	Delay “position reached” autostatus response by 10 ms	59
? !	pos	!pos 0 0 0 !pos z 1.2	-	Set current X Y Z position to 0 Set current Z position to 1.2	60
(!)	zero	!zero z	-	Set Z position and internal counter to 0 (e.g. filter wheel application)	61
(!)	clearpos	!clearpos z	-	Set Z position and internal counter to 0 (e.g. filter wheel application), not executable with measuring system	61

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)	62
? !	joydir	!joydir 1 1 -1	Y	Set motor direction for joystick operation (Z reversed)	63
? !	joywindow	!joywindow 14	Y	Set idle window of the joystick center position, where a joystick deflection has no effect [0..100]	63
? !	joyvel	!joyvel z 1.5	Y	**Only if extmode = 1:** Set joystick velocity for Z to 1.5	64
? (!)	joyspeed	joyspeed 2 25	Y	Set joystick speed for speed button 2 “medium” to 25 rev/s	64
? !	keymode	!keymode 2	Y	Select joystick key mode 2 = high speed preselection	65
? !	keyspeed	!keyspeed x 5 20	Y	Set keymode joystick speed X low=5mm/s, high=20mm/s	66
? (!)	joycurve	!joycurve z 1	Y	Set joystick characteristic for Z ot linear	66
(?)	key	key	-	Read state of all joystick buttons (0=released, 1=pressed)	67
(?)	keyl	keyl	-	Read and clear latched state of all joystick buttons	67
? !	hwfactor	!hwfactor x 1	Y	One handwheel revolution in X is 1mm stage travel	68
? !	hwfactorb	!hwfactorb x 14	Y	One handwheel revolution in X is 14mm stage travel	68

HDI: Joystick, Tackball and Handwheel Instructions

Instruction		Example	Save	Brief description	Page
? !	hwfilter	!hwfilter 0	Y	Switch off handwheel noise reduction	68
? !	tbfactor	!tbfactor 1 1	Y	Set trackball transmission factor in X and Y to default	69
(?)	zwheel	?zwheel	-	Returns 1 if HDI device has a Z-Wheel attached	70
? (!)	zwtravel	!zwtravel 1 0.25	Y	Set default Z-Wheel travel to 2.5 mm/rev	70
? !	zwaxis	!zwaxis a	Y	Assign Z-Wheel to A-axis	71
? !	tvrjoy	!tvrjoy z	Y	Assign AUX-IO pulse&direction joystick to Z axis	71
? !	tvrjoyf	!tvrjoyf 1	Y	Set tvrjoy transmission factor to 1	71
(?)	hdi	hdi	-	Read ID number of the connected HDI device	72
! ?	hdimode	!hdimode 0 1	Y	Set hdimode bit 0 to 1 for ErgoDrive Toggle Mode	73
! ?	configaxsel	! configaxsel	Y	Toggle joystick Z-axis between axes Z and A by F4 key	74

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	75
? !	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	75
(?)	adigin	adigin adigin 2	-	Read all AUX-I/O digital inputs Read logic level of AUX-I/O digital input 2 only	76
? !	adigout	!adigout 3 1 ?adigout	-	Set AUX-I/O digital output 3 to logic level 1 Read back all digital output levels	76
(?)	anain	anain c 2	-	Read input of analogue channel 2	77
? !	anaout	!anaout c 1 17.5	-	Set analogue voltage of channel 1 to 17.5 percent (1.75V)	78
? !	stoppol	!stoppol 1	Y	Set AUX-IO stop input to active high	79
!	stop	!stop 0	-	Release stop condition (in latched stoppol modes 4 or 5)	79
? !	shutter	!shutter 1	-	Set AUX-IO shutter out signal to TTL high	80

Encoder Instructions

Instruction		Example	Save	Brief description	Page
? !	encmask	!encmask 1 1 0	Y	Enable activation of X and Y encoders, disable Z	81
? !	enc	!enc 1 0	-	Manually activate X encoder (caution!), set Y to inactive	82
? !	encperiod	!encperiod 0.1	Y	Set signal period of X encoder to 100 µm	82
? !	encttl	!encttl x 1	Y	X encoder is TTL type (has no analogue sin/cos signal)	84
? !	encdir	!encdir y 1	(Y)	Reverse counting direction for Y encoder	83
? !	encvel	!encvel x 0.5	Y	Set auto-adjust velocity of X encoder to 0.5mm/s	83
? !	encref	!encref 0	Y	Disable usage of X encoder reference signal	84
? !	encnas	!encnas 1 0 0	Y	Enable NAS error signal input encoding for X encoder only	85
(?)	encrefstatus	encrefstatus x	-	Read X encoder reference signal state (1=on reference)	85
(?)	encrefstatusl	encrefstatusl x	-	Read latched X encoder reference signal state	85
(?)	encnasstatus	encnasstatus x	-	Read X encoder NAS signal state (1=NAS error)	86
? !	encerr	!encerr 0	-	Clear encoder error state for X axis (? response is 0 or e)	86
? !	encamp	?encamp x	-	Read X encoder signal amplitude in percent	86
? !	encpos	!encpos 1	-	?pos insruction returns for the encoder positions, if enc=1	87
(?)	hwcount	hwcount	-	Read all encoder positions (TTL counter, no interpolation)	87
(!)	clearhwcount	clearhwcount x	-	Set X axis hwcount to zero	87

MR Encoder Instructions

Instruction		Example	Save	Brief description	Page
? !	mra	?mra x	-	Read amplitude correction factor (sin/cos ratio) of X	88
? !	mro	?mro	-	Read offset correction value for all encoders	88
? !	mrp	!mrp x 0 0 0 0	-	Reset MR-signal peak-to-peak measurement result of X	89

MR Encoder Instructions

Instruction		Example	Save	Brief description	Page
?	mrt	?mrt z 2	-	List two measurement results of the Z input signals	89

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	90
? !	ctrf	!ctrf 2.0	Y	Closed loop factor for X axis is set to 2.0	91
? !	ctrff	!ctrff 2 3.5	Y	Closed loop factors for X axis are set to 2 and 3.5	91
? !	ctrc	!ctrc 3	Y	Closed loop control is called every 3 millisecond	92
? !	ctrd	!ctrd 100	Y	Closed loop in target window for 100 milliseconds	92
? !	ctrt	!ctrt 200	Y	Closed loop control timeout after 200 milliseconds	92
? !	twi	!twi 0.01 0.01 0.01	Y	Set target window for X Y Z to 10µm (assume dim=2)	93
?	ctrstatus	?ctrstatus 1	-	Get Closed Loop active state of all axes	93
?	ctrdiff	?ctrdiff	-	Get Closed Loop position difference of all axes	94

Trigger Signal Configuration[1]

Instruction		Example	Save	Brief description	Page
? !	trig	!trig 1	-	Enable trigger functionality (should be the last command)	95
? !	triga	!triga x	-	Trigger function is related to X axis	95
? !	trigm	!trigm 0	-	Select trigger mode 0	96
(!)	trigger	trigger	-	Manually set trigger output (available in trigm 102, 103)	98
? !	trigs	!trigs 40	-	Set trigger output signal length to 40 microseconds	97
? !	trigd	!trigd 10	-	Set trigger distance to 10 (mm if dim=2)	97
? !	trigf	!trigf 1000	-	Generate periodic trigger pulses with 1kHz	97
? !	trigcount	?trigcount	-	Read number of generated trigger events	98

Snapshot Signal Configuration[1]

Instruction		Example	Save	Brief description	Page
? !	sns	!sns 1	-	Enable snapshot functionality (always 1 after power-up)	100
? !	snsl	!snsl 0	Y	Set snapshot input signal to active low	100
? !	snsf	!snsf 10	Y	Set snapshot signal debounce filter to 10 milliseconds	100
? !	snsm	!snsm 0	Y	Set snapshot mode to 0(=capture, 1=move)	101
? !	snsc	?snsc	-	Read number of snapshot events (=array fill size)	102
? !	snsp	?snsp x	-	Read last captured X position	102
? !	snsa	?snsa 1	-	Read first position entry of snapshot array (all axes)	103
(!)	snse	snse 2	-	Generate SnapShot event F2	103
? !	prehome	!prehome 10 20 1	-	Set prehome positions X Y Z to 10 20 1 (unit depends on dim setting)	104
? !	home	!home 5 5 0	-	Set home positions X Y Z to 5 5 0 (unit depends on dim setting)	104

[1] Function has to be enabled by factory, it is not available per default.

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

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: This instruction returns 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. readsn (Read Serial Number)

Syntax: ?readsn or readsn
Parameter: none

Description: This instruction returns the controller’s serial number.

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
XXX	Index number

Example: ?readsn

7.4. ver (Read default Version Number)

Syntax: ?ver or ver
Parameter: none

Description: This instruction returns the default firmware 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**".

Example: ?ver
Response syntax: Vers:*LSnm.xx.xxx*
(in some cases Vers:*ESnm.xx.xxx*)

"Vers:LS"	Fixed character string
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

7.5. iver (Read internal Version Number)

Syntax: ?iver or iver
Parameter: none

Description: This instruction reads 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 of ?iver response: T04.35.02-0004

7.6. 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

7.7. temp (Read Case Temperature)

Syntax: ?temp or temp
Parameter: none

Description: This instruction reads the temperature inside the controller. Available only with some controllers or the encoder interface.

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

Example of temp response: 28.9

8. Communication Interface Settings

8.1. baud (Baud Rate)

Syntax: !baud or ?baud

Parameter: 9600, 19200, 38400, 57600 or 115200

Description: This instruction sets or reads the baudrate of the serial communication interface. After sending this instruction please make sure that the controlling device's (e.g. a PC) interface has the same settings. Then a **"!save"** instruction may be sent to permanantly store the new baudrate in the controller.
For PCI/PCI-E card versions or Tango-DT with USB interface this instruction has no effect, as they communicate with fixed higher data rates internally. In this case it does not matter which baudrate the virtual COM port is opened with.

Response: Current baud rate of the controller.

Examples:

!baud 57600	The baud rate is set to 57600 [Bd].
?baud	Query controller for current baud rate

8.2. cts (Enable/Disable RS232 Hardware Handshake)

Syntax: ?cts or !cts

Parameter: 0 or 1

Description: Writing a 1 enables additional hardware handshake of the RS232 or USB interface. A 0 disables this function.
For PCI bus communication this instruction has no effect.
Please note that the PC COM port has to be opened in hardware handshake mode, too.

Response: Current state of CTS (0=disabled or 1=enabled)

Examples:

?cts	query controller for current state of CTS
!cts 0	disable CTS handshake
!cts 1	enable 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 or restore
Parameter:	none

Description: The controller reloads the saved parameters from its nonvolatile memory. The current controller parameters get overwritten by the saved defaults. Refer to the "**save**" instruction. The effect is similar to a software "**reset**", but does not affect or restart the entire hardware.

Response:	none
Example:	restore

9.3. reset (Force a Software Reset)

There are two options to reset the controller:

- The power on reset
- The 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. So for knowing if the controller is rebooted and ready, it may be necessary to poll data until it responds again. (E.g. send "**?err**" until the controller responds.)

Response:	none
Example:	reset

9.4. pa (Enable or Disable the Power Amplifiers)

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

Description: This instruction switches all motor amplifiers on(=1) or off(=0). If switched off, no motor current is flowing. To switch off axes individually, please use the **'axis'** instruction.

With amplifiers off it is not ensured that the axis position will be retained. Also if the axis has encoders, the closed loop will be deactivated.

Amplifier switch off can also be caused by a short circuit. Then an internal error (error number 29) is generated and the status LED flashes. ?pa will be read as 0.

1 = Amplifiers on
0 = Amplifiers off

Response: 0 or 1

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

9.5. ipreter (Select Instruction Set)

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

Description: 0 => Prohibited. Register command set is no longer provided.

1 => Default instruction set (Native), as described in this manual

2 => VENUS-1 and VENUS-2

3 => LUDL MAC5000

4 => Use only with Firmware Versions 1.46 and above:
ASI MS-2000

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 => The controller switches 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.

Examples:

!extmode 1	Set controller into extended mode.
?extmode	Query extended mode.

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. In this mode applies a constant vector velocity for automatic moves (moa, mor) which is set by **'scanvel'**.

0 = normal operation (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	Query controller scanmode

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.
As this is a vector mode there is only one velocity parameter.
Please also refer to the **'scanmode'** instruction.

Response: Currently selected velocity in [mm/s]

Examples:

!scanvel 10	Set scanmode vector velocity to 10mm/s
?scanvel	Query scan mode 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.

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 available axes, e.g. 0s)
?modulomode x     => 1     (returns modulo mode for 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:

0 => Sending of any automatic status replies is disabled, except **'save'** instructions will still return either "OK..." or "ERR".

1 => After each automatic move (e.g. moa, mor, cal, rm) the 'position reached' response (a 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 and can not be stored permanently**.

2 => The controller transmits the message 'position reached' plus the **status message** "OK..." or "ERR".

3 => A simple <CR> (0x0d hex) is returned to indicate that position(s) have been reached. 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.

Autostaus can not be saved. After power on or reset it is always set to mode 1.

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**" now has to be polled to find out if the axis is moving).
?autostatus	Read the currently selected autostatus.

11.2. statusaxis (Query State of Axis)

```
Syntax:             ?statusaxis or statusaxis
Parameter:          none

Description:        Statusaxis responds the state of each axis.
                    Similar to the 'autostatus 1' response to 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 command.
                    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.
```

11.3. status (Query 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 error
                    number. Also see 'err' instruction.

Response:           OK... or ERR with error number

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

11.4. err (Query 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 wil be updated or re-set by the next instruction. Additionaly 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 => 0
!err (clear error state if no permanent error)

11.5. help (Query 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.

When called without a parameter:
It returns the controller's error state with description

When 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
(controller state, assumed to be ok here)

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.

Example: service

11.7. pci (Is PCI Bus)

Syntax: ?pci or pci
Parameter: none

Description: The instruction pci returns:

0 = Controller is a desktop version
1 = Controller is a PCI card and plugged in a PCI slot

Response: 0 or 1

Example: pci => 0

11.8. isvel (Query Actual Velocities)

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

Description: Read the actual velocitie(s) with which the axis is currently traveling. Unlike **'?vel'** or **'?speed'** this instruction returns the current real speed of the axes.

Response: Actual motor velocity in [mm/s]

Example:
```
?isvel     => Query actual velocity of all axes
?isvel y   => Query actual velocity of the X axis
```

11.9. maxpos (Maximum Position)

Syntax: ?maxpos
Parameter: none

Description: Query 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 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 (or "dimension") of all input and output parameters related to length, e.g. position or move commands.

The provided units for length (parameters for dim) are:
0 => Micro steps
1 => µm
2 => mm (Tango 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 units in mm/s)

Examples:

!dim 4 1	the selected dimension for X is [revolutions] and for Y is [µm].
!dim z 2	the selected dimension for Z is [mm]
?dim	responds the dimensions for all axes.
?dim a	responds the dimension of the a-axis.
dim 2 2 2 2	set dimension for all axes to [mm]

Response: Current settings

Hint: For dimensions 3 (=360°) and 4 (=revolutions) it is recommended to set the **spindle pitch** to 1mm.

12.2. pitch (Spindle Pitch)

Syntax: !pitch or ?pitch
Parameter: x, y, z, a or none
0.0001 to 68

Description: This instruction sends the spindle pitch (here: travel distance per motor revolution) to the controller. It will be taken for all further calculations.

Response: current spindle pitch

Examples:

!pitch 4.0 1.0	set spindle pitch X=4[mm] and Y=1[mm]
!pitch z 2.0	set spindle pitch Z=2[mm]
?pitch	query all axes for their spindle pitch
?pitch a	query spindle pitch for a-axis

12.3. gear (Gear Ratio)

Syntax: `!gear` or `?gear`

Parameter: x, y, z, a or none
0.001 to 1000

Description: This instruction transmits the gear ratio to and from the controller. The ratio is 1, if the motor is directly mounted on the spindle.

Response: current gear ratio

Examples:

!gear 10	set gear ratio X=1/10
!gear 4.0 1.0	set gear ratio X=1/4 and Y=1/1
!gear z 10.0	set gear ratio Z=1/10
?gear	query all axes for their gear ratio
?gear a	query A-axis for its gear ratio

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 paramerer 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: none, x, y, z or a
0.0001 to 20 [m/s^2]

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

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

Response: Currently used acceleration in m/s^2

Examples:

!accel 0.5	set acceleration X=0.5[m/s^2]. Other axes are not affected.
!accel 1 0.55	set acceleration X=1.0[m/s^2] and Y=0.55[m/s^2]
!accel z 0.2	set acceleration Z=0.2[m/s^2]. Other axes are not affected.
?accel	query all axes for their current acceleration.
?accel z	query Z axis for its acceleration.

Additional information:

?accel returns the acceleration parameter with a fractional resolution of 0.01 m/s^2. If a higher resolution is desired, it can be requested from firmware 1.46 and higher as follows:
In order to read out more decimal places it is possible to add the desired decimal places to the query (valid is 0 to 6).
Examples:

?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, cal, rm).

0 = Linear acceleration/deceleration ramp
1 = $\sin^2$ acceleration/deceleration ramp
2 = reserved, currently same as 1: $\sin^2$

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 $\sin^2$, other axes are not affected
!accelfunc x 1	set accel function X to $\sin^2$, other axes are not affected
!accelfunc 1 1 0	set accel function in X and Y to $\sin^2$, 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^2

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

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

Response: Deceleration for stop conditions

Examples:

!stopaccel 1 1 2	Set the stop deceleration for X and Y to 1 and Z to 2 $[m/s^2]$
!stopaccel x 1.5	Set the X stop deceleration to 1.5$[m/s^2]$
?stopaccel	Returns the currently used stop deceleration for all axes

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 **extmode**=0 (default), the 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 '**velfac**' instruction can be used in addition to 'vel', but is not necessary or recommended.

Response: Currently selected velocity

Examples:

!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	=> 20.000000 0.123456 5.000000
?vel y 4	=> 0.1235

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             query all axes for their current velocity factors
?velfac z           query X axis for its current velocity factor
!velfac x 0.1       set velocity X to 1/10 of current 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
                    1 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.

Response:           Currently used secure velocity [1 to 100 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. maxcur (Read Maximum Motor Current)

```
Syntax:             ?maxcur
Parameter:          x, y, z, a or none

Description:        This instruction reads the maximum possible motor current
                    which the power amplifier is able to provide.
                    Motor current might be limited by the controller type or
                    factory settings in order to protect the motor from overload.

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
```

12.12. cur (Motor Current)

Syntax: !cur or ?cur

Parameter: x, y, z, a or none
0.1 to [maximum current]

Description: This instruction sets or reads the motor current. The maximum current is limited by hardware and may be cheched by the “**maxcur**” instruction.
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 currents of all axes
?cur x	read motor current of X axis only

12.13. reduction (Motor Current Reduction Factor)

Syntax: !reduction or ?reduction

Parameter: x, y, z, a or none
0 to 1.0

Description: This instruction sets or reads the motor current reduction factor. When the axis is idle (stopped), the motor current is reduced by this factor. The floating point values from 0. to 1 represent a current of 0 to 100% of the selected motor current (**cur**). A value of 1 disables the reduction (default).
Motor current reduction can be used to keep the motor temperature low, but as a side effect it may slightly decrease vector throughput performance and position accuracy (waggle while reducing or slight position deviation under load conditions if not in closed loop).
The current reduction can be delayed by the ‘**curdelay**’ instruction.

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

Examples:

!reduction .1 .7	Set idle currents reduction X=0.1*cur[A] and Y=0.7*cur[A]
!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

12.14. curdelay (Delay for Current Reduction)

Syntax: !curdelay or ?curdelay

Parameter: x, y , z, a or none
0 to 10000 [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.

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

Examples:

!curdelay 100 300	set delay for motor current reduction X=100[ms] and Y=300[ms]
!curdelay z 450	set delay for motor current reduction Z=450[ms]
?curdelay	read motor current reduction delay of all axes
?curdelay x	read motor current reduction delay of X axis only

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 ... 819200

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 and applies to all axes that have **dim 0** selected.

Response: Currently used **dim 0** microstepping resolution

Examples:

!usteps 40000	Set microstep resolution to 40000/revolution
?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 for ‘?pos’ and similar position returning instructions for dim 2,9 and 1. 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 µm
1	= 0.1mm	0.1 µm
2	= 0.01mm	0.1 µm
3	= 0.001mm	0.1 µm
4 (default)	= 0.0001mm	0.1 µm
5	= 0.00001mm	0.01 µm
6	= 0.000001mm	0.001 µm

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

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 [µm]

Description: This instruction compensates mechanical backlash for each axis. Unit is µm, independent from dim.

0 = Backlash compensation off

Response: Currently used backlash in µm

Examples:

!backlash 12.7 21.3 0	Set backlash for X to 12.7µm, Y=21.3µm and Z=none
!backlash x 0	Disable backlash compensation for X
!backlash x 5	Compensate a backlash of 5µm 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 (Query 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)

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. 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 for valid motor voltage level
?updelay       => Read the power up delay value
```

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:        This command enables or disables the setting of software
                    limits by the calibration (cal) and range measure (rm)
                    function.
                    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:

This command swaps the cal(E0) and rm(EE) switch assignment.

0 = switches are not swapped
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 independend 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 nor 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 <u>for all axes</u> (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
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 command)
                    Ref = Reference switch
                    EE  = upper limit switch (!rm command)

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” command, swin reflects the TTL
                    input levels. Also disabled switches are represented with
                    their current TTL input signal level. Swin is not affected by
                    the axisdir command (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 (Command a Calibration)

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'.

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@-."

Or the sequence
"!axis 1 0 1"
"!cal"
"!axis 1 1 1"
calibrates the X and Z axis, while Y is not used => "A@A-."

14.2. rm (Command a Range Measure)

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-.”
cal 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. 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: Selected 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

14.4. 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

14.5. 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

14.6. rmoffset (Range Measure Position Offset)

Syntax: !rmoffset or ?rmoffset

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:

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

14.7. caldir (Calibration Direction)

Syntax: !caldir or ?caldir

Parameter: x, y, z or a
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

14.8. 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)

14.9. 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: Set or read the 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.

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)

14.10. calpos (Calibration Position)

Syntax: !calpos or ?calpos

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

Description: Used to measure calibration position accuracy (repeatability of the origin) in combination with an encoder.
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

14.11. 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

14.12. 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)

14.13. 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)

14.14. 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 combinmed 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 slidebar 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 ±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 10.2 ([mm] assume dim=2)
moa 10.2	same as "moa x 10.2"
moa y 34.5	Y-axis travels to position 34.5 ([mm] assume dim=2)

15.2. mor (Move Relative)

Syntax: !mor or mor

Parameter: x, y, z, a or none
distance values within ±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.

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	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 ±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

Description: This instruction commands one or more axes to travel at the specified speed. It can be stopped by setting the speed to zero. The speed instruction is also called "digital Joystick", therefore it only applies when joystick is enabled: **joy,joydir**

Response: Currently executed speed in [rev/s] or [mm/s], when 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.

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. zero (Set Internal Position to Zero)

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

Description:      Unlike the command "!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
```

15.13. 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 enanble or disable the connected HDI device (joystick, trackball, ErgoDrive etc.)
It is recommended to only use the values 0 or 2.

0 disable HDI device
2 enable HDI device (=default)

Remarks: The **'speed'** instruction also requires joy to be enabled (2).

Response: HDI enable state

Examples:
!joy 0 disable the HDI device (joystick, trackball etc.)
!joy 2 enable the HDI device (=default)
?joy read HDI enable state

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 3rd 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 3rd 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. 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

16.4. joyvel (Joystick Velocity)

Syntax: !joyvel or ?joyvel

Parameter: x, y, z, a or none
0.0001 to 100 [revolutions/s] or [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. If so, the vel command 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

16.5. joyspeed (Joystick Speed Presets for BPZ Device)

Syntax: !joyspeed or ?joyspeed

Parameter: 1, 2 or 3 and
0.0001 to 100 [revolutions/s]

Description: Only used by a customer designed external 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

Examples:

?joyspeed 1	Query for "Slow" joystick button speed
!joyspeed 3 30	Set "fast" joystick button speed to 30 [revolutions/s]

16.6. 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)

16.7. keyspeed (Joystick Key Speed Presets)

```
Syntax:             !keyspeed or ?keyspeed
Parameter:          x, y, z, a or none
                    0.0000025 to 100 [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)
```

16.8. 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
```

16.9. 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)
```

16.10. 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 cleard 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)
```

16.11. 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
```

16.12. 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
```

16.13. 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          => Query hwfilter state
```

16.14. 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
```

16.15. 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

16.16. 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'** command 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

16.17. 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)
```

16.18. 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
```

16.19. 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
```

16.20. 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)

16.21. 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 velocitiy XY with F1+F4, Z with F2+F3 1=toggle KeySpeed velocitiy 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)

16.22. 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 (SHUTTER_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
10	**AUX-IO**	**9**	**ANAIN0**
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:

Umot[V] = (5 / 1023) * [anain c 12] * (55.7/4.7)
More accurate:
Umot[V] = (2.5 / [anain c 15]) * [anain c 12] * (55.7/4.7)

Calculating the internal PSE voltage:

Umot[V] = (5 / 1023) * [anain c 11] * (14.7/4.7)
More accurate:
Umot[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:         0,2,4 (= active low modes), 1,3,5 (= active high modes)


Description:       Set or read the operating mode of the AUX-I/O “Stop” input.
                   The stop input has an internal pull-up resistor to +5V.
                   When connecting an NO (normal open) stop switch to GND,
                   any of the active low modes must be selected.
                   For NC (normal closed) switches please select active high.

                   0 = active low , Joystick not affected by stop signal
                   1 = active high, Joystick not affected by stop signal
                   2 = active low , all moves disabled as long as signal applied
                   3 = active high, all moves disabled as long as signal applied
                   4 = active low , all moves disabled until “!stop 0” command
                   5 = active high, all moves disabled until “!stop 0” command

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

Example:
!stoppol 5         => Set the polarity/function of the AUX-I/O stop input to
                      latched active high.
```

17.8. stop (Release or Force Stop Condition)

Syntax: !stop
Parameter: 0, 1

Description: Release or force stop condition in latched ‘**stoppol**’ modes 4 and 5.

```
0 = Release stop condition (if stoppol = 4 or 5 only)
1 = Force stop condition   (if stoppol = 4 or 5 only)
```

Response: -

Example: !stop 0

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
```

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 preformed, which leads to positioning errors.

18.1. encmask (Encoder Mask)

Syntax: !encmask or ?encmask
Parameter: x, y, z, a (or none)
0 or 1

Response: Encoder enable mask

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 command **'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

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	Query encoder mask state for 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
0.00001 to 4.0 [mm]

Description: This instruction reads or sets the encoder signal period. The unit is always [mm].

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

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 tis 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-Ajust 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. encttl (Encoder has TTL Signal)

Syntax: `!encttl or ?encttl`

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 opration (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:

!encttl 0 0 1	Y and Y axis encoders are analog, Z is digital A/B encoder
!encttl z 1	Set Z encoder signal processing to digital
?encttl	Query all axes for their currently used signal type
?encttl 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 for all axes
encrefstatus x      Read REF signal state for 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

Description:        This command reads 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
```

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 acivated)
~~!encpos 1 1 1~~	Invalid!
!encpos x 1	from now on a 'pos' command returns the encoder position for the X -axis (if the encoder is acivated)
?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

Response: Reset encoder hardware counter value(s)

Description: This command resets the hardware counter(s) to zero.

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 command.

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 command.

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

Description:        This instruction reads the corrected sine and cosine A/D
                    converter results of the analog encoder interface as signed 16
                    bit integer.

                    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.

                    Each data line is terminated by a [CR].

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 8 values per line)
?mrt 1              Returns one line  with all axes (up to 8 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
```

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.

Hint:
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)
                    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: 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.

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 intervall

20.5. ctrd (Control Target Window Delay)

Syntax: !ctrd or ?ctrd
Parameter: 0 to 250 [ms]

Description: This instruction reads or sets the control delay.
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 **ctrt** 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. ctrt (Control Timeout)

Syntax: !ctrt or ?ctrt
Parameter: 0 to 10000 [ms]

Description: This instruction reads or sets the control timeout.
It specifies the time the closed loop tries to reach the desired encoder position. If the ctrd condition could not be fulfilled within this ctrt time, it will be aborted.
If ctrd/ctrt is used, the ctrt timeout must be set to a value which is higher than the ctrd, typically 1+ seconds.
Unit is milliseconds. Only one parameter for all axes.

Response: Closed loop control timeout in milliseconds.

Examples:

!ctrt 1000	Closed loop tries to reach the end position for 1 second
?ctrt	Query closed loop timeout

20.7. twi (Target Window)

```
Syntax:             !twi or ?twi
Parameter:          x, y, z, a or none
                    [value corresponding 0.00004 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,ctrt 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. ctrstatus (Control Status)

```
Syntax:             ?ctrstatus
Parameter:          x, y, z, a or none
                    optional parameter: 1

Description:        This instruction returns the one of two possibel closed loop
                    states, depending on if the optional Parameter 1 is used.

                    Sending the ?ctrstatus request with or without specifying an
                    axis returns the internal ctr state which is set when the
                    closed loop gets enabled by the controller, e.g. 0, 1 or 2.

                    Hint: the internal ctr mode is set to the requested ctr mode
                    when the closed loop is acivated. The state may be zero when
                    the closed loop has not been activated yet by cal, calmode,
                    encmask or ctr=0.

                    Sending the ?ctrstatus 1 request with or without specifying an
                    axis returns if the closed loop momentary is active.

                    0 = Closed loop momentary not active (Refer to hint below)
                    1 = Closed loop momentary active in "until target" window mode
                    2 = Closed loop momentary active
                    See "ctr" for more states

                    Hint: The state may be not active when the internal cal mode
                    is zero or e.g. during calibration, invalid encoder signal,
                    low motor current reduction (≤ 30%), axis in limit switch etc.

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. "2 2 0"
?ctrstatus x 1      Returns the Closed Loop active state of the X-axis, e.g. "2"
```

20.9. ctrdiff (Control Position Difference)

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

Description: This instruction returns the momentaryly 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 Signal Configuration (optional)

These commands read or modify the parameters for the trigger output signal. It may be used for synchronization of an external device like e.g. a video camera. The trigger output signal is available on the optional AUX-I/O connector. Access permission to the trigger functionality has to be enabled by factory. 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, 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, set start position

Optional the "trigcount 0" command 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 0	Disable trigger circuit
?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

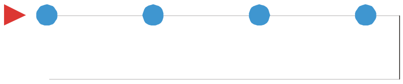
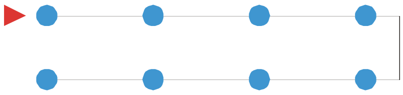
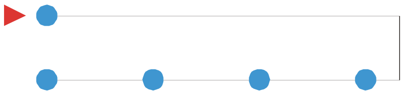
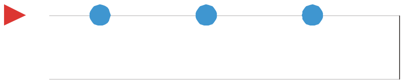
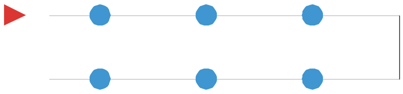
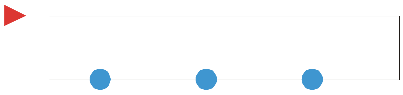
Examples:

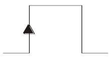
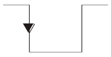
!triga x	Select X-axis as trigger source
!triga y	Select Y-axis as trigger source
?triga	Read current trigger axis

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
1		High active	First pulse when move starts
2		High active	First pulse when move starts
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 trigd/2
7		High active	Triggers shifted by trigd/2
8		High active	Triggers shifted by trigd/2
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	Generates periodic trigger signals with the frequency choosen by the “trigf” parameter.	High active	Does not depend on position
101	-- See Mode 100 --	Low active	Same as 100, signal inverted
102	Allows manual forced trigger signals by the “trigger” command.	High active	Does not depend on position or time
103	-- See Mode 102 --	Low active	Same as 102, signal inverted

104	Trigger on position reached of specified axis	High active 	Comes with @@@ status response
105	-- See Mode 104 --	Low active 	Same as 102, signal inverted

Response: Trigger mode as integer: 0 to 11, 100 to 105

Examples:	!trigm 3	Set Trigger Mode 3
	?trigm	Query current trigger mode

21.4. trigs (Trigger Signal Length)

Syntax: !trigs or ?trigs
Parameter: 0 to 2621400 [µs]

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 (µs)

Examples:

!trigs 40	Set Trigger pulse width to 40 µs
!trigs 2500000	Set Trigger pulse width to 2.5 s
?trigs	Read current trigger pulse width

21.5. 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 whith the selected axis, a trigger signal is generated.

Response: Trigger distance

Examples:

!trigd 3	Set trigger distance to 3mm (if dim of selected axis is 2)
!trigd 0.010	Set trigger distance to 10µm (if dim of selected axis is 2)
?trigd	Read current trigger distance

21.6. trigf (Trigger Frequency)

Syntax: !trigf or ?trigf
Parameter: 0.01 to 12500

Description: This instruction sets the frequency for periodic trigger output mode (trigm 100).
The frequency resolution is 1/40µs.

Response: Trigger frequency

Examples:

!trigf 2500	Periodic trigger pulses have 2.5kHz (signal every 0.4ms)
?trigf	Read trigger frequency

21.7. 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

21.8. 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. The pulse width depends on the "**trigs**" value.
Response:	None

Examples:

trigger	Force trigger pulse now
!trigger	The same as above

1. Snapshot Input Configuration (optional)

Snapshot functionality must be configured by factory.

The **'?det'** instruction may be used to identify if Snapshot is configured.

The snapshot functionality allows capturing of X,Y,Z,A axis positions by either pressing a HDI key (e.g. Joystick F2) or an external TTL signal. Up to 200 values can be stored in an array.

A snapshot event can also command the Tango controller to move to positions stored in the array (move by snapshot event).

The snapshot event can be triggered by either the Joystick key "F2", via the assigned digital Input of the optional AUX I/O connector or by a software instruction.

The position unit depends on the selected dimension **"dim"**. The setting of **"encpos"** defines if the motor position or the truly measured encoder position is written to the array. A maximum of 200 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

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	*invalid*	*invalid*	*invalid*	*invalid*
5	*invalid*	*invalid*	*invalid*	*invalid*
...	...	...	...	...
200	*invalid*	*invalid*	*invalid*	*invalid*

1.1. sns (Snapshot enable/disable)

```
Syntax:             !sns or ?sns
Parameter:          0 or 1

Description:        This instruction globally enables or disables the snapshot
                    functionality.
                    Please note that snapshot is globally enabled by default
                    (power-on), in order to support stand-alone applications.

                    0 = disabled
                    1 = enabled

Response:           Snapshot state

Examples:
!sns 0              Disable snapshot
!sns 1              Enable snapshot
?sns                Read snapshot enable state
```

1.2. snsl (Snapshot Level / Polarity)

```
Syntax:             !snsl or ?snsl
Parameter:          0 (=active low) or 1 (active high)

Description:        This instruction sets the required snapshot signal polarity.

Response:           Currently used snapshot polarity

Examples:
!snsl 0             Set snapshot input to active low
!snsl 1             Set snapshot input to active high
?snsl               Read snapshot input polarity
```

1.3. snsf (Snapshot Filter)

```
Syntax:             !snsf or ?snsf
Parameter:          0 to 100 [ms]

Description:        This instruction reads or modifies the snapshot filter time,
                    which is used to debounce the snapshot input signals.

Response:           Currently used snapshot filter time

Examples:
!snsf 0             Disable input filter
!snsf 10            Set snapshot filter time to 10 ms
?snsf               Query snapshot filter time
```

1.4. snsm (Snapshot Mode)

Syntax: !snsm or ?snsm

Parameter: 0, 1, 2, 3, 4 or 5

Description: This instruction reads or sets the snapshot mode (default=0).

0 = Capture positions with Joystick key "F2"
1 = Automatic mode: Move to Positions with Joystick Key "F2"
2 = Extended move:
~~F1: Step/move through position list forward (pointer+1)~~
(wraps around at the last element)
F2: Step/move through position list backward (pointer-1)
(wraps around at the first element)
~~F3: Move to start of list (first element)~~
F4: Moves to "prehome" position with "vel",
then to "home" position with "secvel"
~~3 = Dissection mode~~
4 = Mode 0 but AUX-I/O SnapShot input is used instead of F2
5 = Mode 1 but AUX-I/O SnapShot input is used instead of F2

Remarks: A snapshot is always executed on all active axes (capture and move as well).

Response: Currently selected snapshot mode

Examples:

!snsm 0	Set snapshot mode to capture
!snsm 1	Set snapshot mode to move
!snsm 2	Set snapshot mode to extended move
?snsm	Query current snapshot mode

1.5. snsc (Snapshot Counter)

Syntax: !snsc or ?snsc
Parameter: --

Description: This instruction reads the snapshot counter, which shows the counted snapshots (= snapshot array entries). This instruction may also be used to reset the counter to zero.

Response: Current snapshot array entries (= number of snapshot events)

Example:

?snsc	Query the number of detected snapshots.
!snsc	Clear snapshot counter

1.6. snsp (Snapshot Position)

Syntax: !snsp or ?snsp
Parameter: x, y, z or a

Description: This instruction reads the most recent snapshot position. The position can also be written to, but has no effect for Firmware versions before 1.57.

Remark: The position data unit depends on selected dimension 'dim'.

Response: Last captured Snapshot position of the specified or all available axes

Examples:

?snsp	Query all axes for their last captured snapshot positions
?snsp z	Query Z axis for its last captured snapshot position
~~!snsp 100 200~~	~~Append snapshot position for X and Y~~
~~!snsp 10 20 30~~	~~Append snapshot position for X, Y and Z axis~~
~~!snsp y 2000~~	~~Append snapshot position to Y = 2000~~

1.7. snsa (Snapshot Array)

Syntax: !snsa or ?snsa
Parameter: x, y, z or a
and entry index from 1 to 200

Description: This command is used to read, modifiy or clear the snapshot position array, which may contain up to 200 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 may be used. The Snapshot counter then will be incremented by 1 automatically.

Remark: The position data unit depends on the selected dimension "**dim**". The setting of "**encpos**" defines if the calculated position or the measured encoder position is written to the array.

Response: Snapshot array position(s)

Examples:

?snsa 1	Returns all axis positions of the 1st snapshot entry
?snsa 33	Returns all axis positions of the 33rd snapshot entry
?snsa z 99	Returns only the Z-axis position of the 99th snapshot entry
?snsa x 199	Returns only the X-axis position of the 199th snapshot entry
!snsa 0	Clear the entire snapshot array
!snsa x 1 20.5	Set X position of first element to 20.5 (e.g. mm if dim=2)
!snsa 2 10 10 10 10	Set all axis positions of the second array entry to 10

1.8. snse (Snapshot Event)

Syntax: !snse or snse
Parameter: 2

Description: The Snapshot event instruction can be used to execute the Snapshot functions via the communication interface.
Instead of pressing a Joystick button or using the AUX-I/O signal, these "**snse**" parameters can be sent 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.

Response: -

Example:

snse 2	Execute F2 Snapshot event (e.g. store current position in the snapshot array and snapshot position)

1.9. prehome (Snapshot PreHome Position)

Syntax: !prehome or ?prehome
Parameter: x, y, z or a

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**”.
See “**snsm**” 2 for more details.

Response: Position value(s)

Examples:

!prehome x 10.2	Set prehome position X-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 X-position
?prehome	Read currently used prehome positions of all axes

1.10. home (Snapshot Home Position)

Syntax: !home or ?home
Parameter: x, y, z or a

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**”.
See “**snsm**” 2 for more details.

Response: Position value(s)

Examples:

!home x 10.2	Set home position X-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 X-position
?home	Read currently used home positions of all axes

2. Document Revision History

No.	Revision	Date	Changes	Remarks
01	A	03. July 2007	New layout, improved and corrected descriptions, added new instructions, re-sorted instructions	Based on Tango firmware revision 1.26
02	B	09. July 2007	Added new instructions	Based on Tango firmware revision 1.26
03	prelim. C	27. July 2007	twi example corrected	Based on Tango firmware revision 1.26
04	C	03. Sept 2007	Added snapshot functions	Based on Tango firmware revision 1.26
05	D	28. Feb. 2008	Added some new instructions of firmware 1.31 and 1.32, Bugfixes in examples and descriptions.	Based on Tango firmware revision 1.32
06	prelim E	17. Jun. 2008	Added new instructions of firmware 1.34	Based on Tango firmware revision 1.34
07	E	07. July 2008	Added new instructions of firmware 1.35	Based on Tango firmware revision 1.35
08	prelim F	23. July 2008	Added encamp instruction	Based on Tango firmware revision 1.35
09	G	14. Aug. 2008	Added instructions keymode, keyspeed, extended help instruction, improved some comments	Based on Tango firmware revision 1.37
10	prelim H	29. Aug. 2008	Added Z-Wheel and extended version instructions	Based on preliminary Tango firmware revision 1.374
11	prelim H	14. Oct. 2008	added backlash instruction, extended lockstate bits	Based on preliminary Tango firmware revision 1.394
12	prelim H	07. Jan. 2009	Improved description of calmode behavior	
13	prelim H	06. Feb. 2009	zwtravel description corrected	
14	H	13. Feb. 2009	Documentation Rev. H released	Based on Tango firmware revision 1.40
12	prelim I	05. May 2009	corrected ?swin- and improved ?readsw command-description, added new MW logo	
13	I	27. May 2009	updated ipreter command description	
14	prelim J	09. July 2009	iver command description	
15	J	26. Aug. 2009	Corrected description of encnasstatus, encrefstatus, encdir. Extended snsm parameter, accel parameter. Improved description of HDI instructions, vel. Added new instructions: go, adigout, adigin, snse, hwfactorb, encrefstatusl, clearpos, maxpos, stout, accelfunc, hdimode, ctrstatus, ctrdiff.	Based on Tango firmware revision 1.46
16	prelim K	31. Aug. 2009	ipreter 1,2 extended up to 5.	Based on Tango firmware revision 1.46
17	prelim L	03. Mar. 2010	description corrected: encpos, refdir, calrefspeed, go resolution example corrected	
18	prelim L	26. Aug. 2010	Trigger modes 104,105 added, Document Title changed from “MST Dokument” to “Tango Instruction Set”	Based on Tango firmware revision 1.515

No.	Revision	Date	Changes	Remarks
19	prelim L	05. Nov. 2010	Improved “hdimode” description, toggle mode and keymode	
20	prelim L	01. Dec. 2010	Updated “?version” description for new controllers	
21	L	27. July 2011	Ipreter options changed	Use only with Tango firmware ≥1.52
22	prelim M	11. Feb. 2011	Changed description for encttl and encref instructions	Based on Tango firmware revision 1.51
23	prelim M	18. Mar. 2011	Added “modulomode”	Only with Tango firmware ≥1.53
24	prelim M	19. May 2011	Revised Introduction and HDI chapters	
25	prelim M	06. July 2011	Encperiod range changed to 0.00001 ~ 4.0 mm Improved description of encoder instructions	Based on Tango firmware revisions ≥1.52
26	prelim M	27. Oct. 2011	Improved Closed Loop description, improved HDI descriptions Added “zwaxis”, “updelay”	Based on Tango firmware revisions ≥1.52
27	prelim M	24. Feb. 2012	Improved SnapShot description	
28	M	26. Mar. 2012	Completely revised and extended documentation	Based on Tango firmware revision 1.52