

The Instruction Set of the TANGO Controller



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



1. Table of Contents

1.	Table of Contents	2
2.	Introduction.....	8
3.	Remarks concerning the controller initialization.....	10
4.	Brief Description of the TANGO Instruction Set	11
5.	Instruction Syntax Description.....	18
6.	Error Numbers and their possible Root Cause	19
7.	Controller Informations.....	20
7.1.	version (Read detailed Version information)	20
7.2.	det (Read detailed Configuration)	21
7.3.	detext (Read Extended detailed Configuration)	21
7.4.	readsn (Read Serial Number)	22
7.5.	ver (Read default Version Number)	22
7.6.	iver (Read internal Version Number)	22
7.7.	uptime (Read Controller Up Time)	23
7.8.	temp (Read Case Temperature)	23
7.9.	maxaxis (Read number of available Axes)	24
7.10.	maxcur (Read Maximum Motor Current)	24
7.11.	etspresent (Read ETS Detect State)	25
7.12.	stagesn (Read Connected Devices Serial Number)	26
7.13.	maxpos (Maximum Position)	27
7.14.	lockpos (TransportLock Position).....	27
7.15.	cmdinfo (List of Available Instructions)	28
8.	Communication Interface Settings.....	29
8.1.	baud (Baud Rate)	29
8.2.	cts (Enable/Disable RS232 Hardware Handshake)	29
9.	System Instructions	30
9.1.	save (Save Parameters)	30
9.2.	restore (Restore Saved Parameters)	30
9.3.	reset (Force a Software Reset).....	31
9.4.	pa (Enable or Disable the Power Amplifiers)	32
9.5.	ipreter (Select Instruction Set)	33
10.	Operating Modes	34
10.1.	autostatus (Set Autostatus Behavior)	34
10.2.	autopreset (Preset for autostatus)	35
10.3.	Extended Mode.....	36
10.3.1	extmode (Switch to Extended Mode)	36
10.4.	Scan Mode	37
10.4.1	scanmode (Scan Mode to change Axis Vector Move Behavior)	38
10.4.2	scanvel (Vector Velocity for Scanmode and Dissection)	39
10.5.	ModuloMode	40
10.5.1	modulomode (Define Linear or Turntable Modes)	40
10.6.	External Display Mode	41
10.6.1	configdisplay (Configure External Display).....	41
11.	Controller States and Error Messages.....	42
11.1.	statusaxis (Read State of Axis).....	42
11.2.	sta (Read Detailed State of Axis).....	43
11.3.	calst (Read Calibration State of Axis)	44



11.4.	corrst (Read Position Correction State)	44
11.5.	status (Read the Controller Error State)	45
11.6.	err (Read Error Number).....	45
11.7.	help (Read Error Number with Description String).....	46
11.8.	cmderr (Command Error List)	46
11.9.	service (Print Service Information to Terminal)	47
11.10.	pci (Is PCI Bus)	47
11.11.	isvel (Read Actual Velocities)	47
11.12.	iscur (Read Actual Motor Current).....	48
12.	General Adjustments.....	49
12.1.	dim (Unit for Positions and Velocities)	49
12.2.	pitch (Spindle Pitch).....	50
12.3.	gear (Gear Ratio).....	50
12.4.	motorsteps (Motor Steps Per Revolution)	51
12.5.	accel (Acceleration)	52
12.6.	accelfunc (Acceleration Ramp Function)	53
12.7.	stopaccel (Emergency Stop Deceleration).....	54
12.8.	vel (Velocity)	55
12.9.	velfac (Velocity Factor)	56
12.10.	sevel (Secure Velocity)	57
12.11.	cur (Motor Current).....	58
12.12.	reduction (Motor Current Reduction Factor)	59
12.13.	curdelay (Delay for Current Reduction)	60
12.14.	ecomove (EcoMove Current Level)	60
12.15.	axis (Enable, Disable, Switch Off Axis).....	61
12.16.	axisdir (Axis Direction).....	61
12.17.	motortable (Motor Correction Table).....	62
12.18.	usteps (Microstep Resolution)	62
12.19.	resolution (Position Number Format).....	63
12.20.	backlash (Mechanical Backlash Compensation)	63
12.21.	lock (Select Parameters to Lock).....	64
12.22.	lockaxis (Apply the Parameter Lock to Axes)	64
12.23.	lockstate (Read all internal Lock States)	65
12.24.	stout (Select Status Signal Output).....	65
12.25.	noled (Force Status LED Off)	66
12.26.	updelay (Power Up Delay).....	66
13.	Limit Switch Instructions (Hardware and Software)	67
13.1.	lim (Software Limits)	67
13.2.	clim (Circular Software Limit)	68
13.3.	limctr (Enable or Disable Limit Control)	69
13.4.	nosetlimit (Do not set Limits by Cal/Rm).....	69
13.5.	limmode (Limit Monitoring Mode).....	70
13.6.	swtyp (Type of Limit Switch)	71
13.7.	swpol (Polarity of Limit Switch)	72
13.8.	swact (Enable or Disable Limit Switches).....	73
13.9.	swdir (Swap Assignment of Cal and Rm Switch)	74
13.10.	readsw (Read Status of Limit Switches)	75
13.11.	swin (Read Limit Switch Input Level).....	75
13.12.	statuslimit (Cal / Rm / Lim Status)	76
14.	Calibration and Range Measure Instructions.....	77
14.1.	cal (Perform Calibration to lower Limit Switch)	77



- 14.2. rm (Perform Range Measure to upper Limit Switch) 78
- 14.3. vrm (Virtual Range Measure) 79
- 14.4. calmode (Closed Loop and Calibration Behavior) 80
- 14.5. calrequired (Calibration Required) 82
- 14.6. caltimeout (Calibration Timeout) 82
- 14.7. caliboffset (Calibration Offset) 83
- 14.8. rmooffset (Range Measure Position Offset) 83
- 14.9. caldir (Calibration Direction) 84
- 14.10. calbspeed (Calibration Speed for Retraction) 85
- 14.11. calrefspeed (Reference Signal Calibration Speed) 85
- 14.12. calpos (Calibration Position read from Encoder) 86
- 14.13. calposmot (Calibration Position read from Motor) 86
- 14.14. calabspos (Calibration Position from Absolute Encoder) 87
- 14.15. calzeropos (Position at CAL) 87
- 14.16. calvel (Calibration Velocities for CAL Instruction) 88
- 14.17. rmvel (Range Measure Velocities for RM Instruction) 89
- 14.18. autopitch (Measure Pitch after CAL Instruction) 89
- 14.19. refdir (Direction for Searching the Reference Switch) 90
- 14.20. posshift (Position Shift for Zero Alignment) 90
- 15. Move Instructions 91**
- 15.1. moa (Move Absolute) 92
- 15.2. mor (Move Relative) 92
- 15.3. m (Move Relative Shortcut) 93
- 15.4. distance (Distance for m) 93
- 15.5. moc (Move to Center) 94
- 15.6. mol (Move to LockPos) 94
- 15.7. go (Go To Position) 95
- 15.8. speed (Speed Move) 96
- 15.9. moe (Move Extended) 97
- 15.10. a (Abort the Current Move) 98
- 15.11. delay (Set the Delay Time for Consecutive Moves) 99
- 15.12. pause (Set the Pause after Position Reached) 99
- 15.13. pos (Read or Set Position) 100
- 15.14. posclr (Clear Position Offset) 100
- 15.15. zero (Set Internal Position to Zero) 101
- 15.16. clearpos (Set Internal Position to Zero) 101
- 16. HDI Instructions (Joystick, Trackball, ERGODRIVE) 102**
- 16.1. joy (Generally Enable/Disable HDI) 102
- 16.2. joydir (Joystick Direction or Assign Joystick per axis) 103
- 16.3. joychangeaxis (Change Joystick X and Y Axis) 103
- 16.4. joywindow (Joystick Window) 104
- 16.5. joyvel (Joystick Velocity) 104
- 16.6. joyspeed (Joystick Speed Presets for BPZ Device) 104
- 16.7. keymode (Joystick Key Mode) 105
- 16.8. keyspeed (Joystick Key Speed Presets) 106
- 16.9. joycurve (Joystick Characteristic) 106
- 16.10. key (Read HDI Device Key State) 107
- 16.11. keyl (Read HDI Device Latched Key State) 107
- 16.12. hwfactor (Coaxial-/ERGODRIVE Transmission Factor) 108
- 16.13. hwfactorb (Alternate Coaxial-/ERGODRIVE Factor) 108
- 16.14. hwfilter (Coaxial-/ERGODRIVE Noise Filter) 108



16.15.	tbfactor (Trackball Factor)	109
16.16.	zwheel (Is Multi-function Wheel Available)	110
16.17.	zwtravel (Multi-function Wheel Travel per Revolution).....	110
16.18.	zwaxis (Multi-function Wheel Axis)	111
16.19.	zwfactor (Multi-function Wheel Factor)	111
16.20.	zwpos (Multi-function Wheel Position Counter)	111
16.21.	tvrjoy (Pulse and Direction Joystick Functionality).....	112
16.22.	tvrjoyf (Pulse and Direction Joystick Factor).....	112
16.23.	hdi (Read HDI ID)	113
16.24.	hdimode (HDI Mode Options).....	114
16.25.	configaxsel (Joystick Axis Select Option)	115
17.	Joystick Function Key Assignments	116
18.	Digital and Analogue I/O	117
18.1.	digin (I/O1 Digital Inputs)	118
18.2.	digout (I/O1 Digital Outputs)	118
18.3.	diginpol (I/O1 Digital Input Polarity)	119
18.4.	digintyp (I/O1 Digital Input Type)	119
18.5.	digoutpreset (I/O1 Digital Output Presets)	120
18.6.	edigin (Multi I/O Digital Inputs).....	121
18.7.	edigout (Multi I/O Digital Outputs).....	121
18.8.	ediginpol (Multi I/O Digital Input Polarity).....	122
18.9.	edigintyp (Multi I/O Digital Input Type).....	122
18.10.	edigoutpreset (Multi I/O Digital Output Presets)	123
18.11.	edigrlly (Multi I/O Relay Option Access)	123
18.12.	adigin (AUX I/O Digital Input)	124
18.13.	adigintyp (AUX I/O, AUX mini Digital Input Type).....	124
18.14.	adigifunc (AUX I/O, AUX mini Digital Input Function)	125
18.15.	adigout (AUX I/O Digital Output)	126
18.16.	adigoutpreset (AUX I/O Digital Output Preset)	126
18.17.	anain (Analogue Input)	127
18.18.	anaout (Analogue Output)	128
18.19.	anamode (Analogue Output Mode)	129
18.20.	stoppol (Mode and Polarity of Stop Input Signal)	131
18.21.	stop (Release, Force or Check Stop Condition)	132
18.22.	stopl (Latched AUX I/O caused Stop Condition).....	132
18.23.	shutter (Shutter Out Signal of AUX I/O).....	132
18.24.	flash (Defined Pulse at AUX I/O Takt Out)	133
18.25.	tvr (Pulse and Direction Input Function)	133
18.26.	brake (Axis Brake Function)	134
18.27.	vbrake (Axis Brake Voltage).....	136
18.28.	brakedelay (Axis Brake Hold Voltage Delay).....	136
18.29.	brakepos (Move to Initial Motor Pole Position)	137
18.30.	drop (Liquid Dispenser – Generate Drops).....	138
18.31.	pump (Liquid Dispenser – Manually Add Air Pressure)	139
18.32.	vbus (+24V Supply Output On/Off).....	140
18.33.	configvbus (+24V Supply Output Preset)	140
18.34.	configcanres (USB Host +5V Supply Output Preset).....	141
18.35.	vusb (USB Host +5V Supply Output On/Off)	141
18.36.	configvusb (USB Host +5V Supply Output Preset).....	141
19.	Encoder Instructions	142
19.1.	encmask (Encoder Mask)	142



- 19.2. enc (Encoder Active) 143
- 19.3. encperiod (Encoder Signal Period) 144
- 19.4. encdir (Encoder Counting Direction) 144
- 19.5. encvel (Encoder Auto-Ajust Velocity) 145
- 19.6. encrefvel (Encoder Ref.-Signal Calibration Velocity) 145
- 19.7. encype (Encoder Type Configuration) 146
- 19.8. encctl (Encoder Configured for TTL Signal) 147
- 19.9. encref (Use Encoder Reference Signal) 147
- 19.10. encnas (Use Encoder NAS Error Signal) 148
- 19.11. encrefstatus (Encoder REF Signal State) 148
- 19.12. encrefstatusl (Latched Encoder REF Signal State) 148
- 19.13. encnasstatus (Encoder NAS Error Signal State) 149
- 19.14. encerr (Encoder Error State) 149
- 19.15. encamp (Encoder Signal Amplitude) 150
- 19.16. encpos (Encoder Position) 151
- 19.17. configencpos (Configure Encoder Position Preset) 152
- 19.18. encsync (Analog Encoder Synchronization Status) 153
- 19.19. hwcount (Hardware Counter) 153
- 19.20. clearhwcount (Clear Hardware Counter) 153
- 20. MR Encoder Instructions 154**
- 20.1. mra (MR Amplitude Correction Factor) 154
- 20.2. mro (MR Offset Correction Value) 154
- 20.3. mrp (MR Signal Peak-To-Peak Measuring Result) 155
- 20.4. mrt (MR Signal Level) 155
- 21. Absolute Encoder Instructions 156**
- 21.1. encform (Absolute Encoder Data Format) 156
- 21.2. encres (Absolute Encoder Data Resolution) 156
- 21.3. abspos (Absolute Encoder RAW Position) 157
- 22. Closed Loop Instructions 158**
- 22.1. Setting Up the Closed Loop 159
- 22.2. ctr (Control Enable) 161
- 22.3. ctrf (Control Factor) 162
- 22.4. ctrff (Extended Control Factor) 162
- 22.5. ctrd (Control Target Window Delay) 163
- 22.6. ctrt (Control Timeout) 163
- 22.7. twi (Target Window) 164
- 22.8. ctrc (Control Call) 164
- 22.9. ctrsm (Control behavior outside Lock-in Range) 165
- 22.10. ctrs (Control Lock-in Range) 165
- 22.11. ctrstatus (Control Status) 166
- 22.12. ctrdiff (Control Position Difference) 167
- 23. Trigger Output Functionality (option) 168**
- 23.1. trig (Trigger) 168
- 23.2. trigm (Trigger Mode) 169
- 23.3. triga (Trigger Axis) 171
- 23.4. trigo (Trigger Output) 171
- 23.5. trigs (Trigger Signal Length) 172
- 23.6. trigd (Trigger Distance) 172
- 23.7. trigcomp (Trigger Compensation) 173
- 23.8. trigenc (Trigger on Encoder) 174
- 23.9. trigf (Trigger Frequency) 174



23.10.	trigbdelay (Precise Trigger Delay for second output).....	175
23.11.	trigbwidth (Precise Signal Width for second output)	175
23.12.	trigbf (Precise Trigger Frequency for second output)	175
23.13.	trigcount (Trigger Counter)	176
23.14.	trigger (Force Trigger Signal)	176
23.15.	trigr (Set Trigger Range).....	177
23.16.	trigp (Trigger Position List Entry)	180
23.17.	trigc (Number of Trigger Position List Entries)	183
23.18.	trigi (Trigger Position List Index)	183
23.19.	trigl (Trigger Level)	184
23.20.	trigsns (Trigger from SnapShot Input)	184
24.	The Second Trigger Signal Output.....	185
24.1.	Introduction.....	185
24.2.	Standard 1:1	186
24.3.	Precise Width.....	187
24.4.	Precise Delay.....	188
24.5.	Precise Frequency.....	189
25.	Snapshot – The Trigger Input Functionality (option)	190
25.1.	sns (Snapshot enable/disable).....	191
25.2.	snspreset (Preset for sns).....	191
25.3.	snsi (Snapshot Level / Polarity)	192
25.4.	snsf (Snapshot Filter).....	192
25.5.	snsm (Snapshot Mode).....	193
25.6.	Snapshot Mode Description and Examples	194
25.7.	snsnc (Snapshot Counter)	196
25.8.	snsni (Snapshot Index)	196
25.9.	snsaxis (Snapshot Axis)	197
25.10.	snsnp (Snapshot Position).....	198
25.11.	snsa (Snapshot Array).....	199
25.12.	snsen (Snapshot Event)	200
25.13.	snsv (Snapshot Voltage)	200
25.14.	snsj (Snapshot Jump).....	201
25.15.	prehome (Snapshot PreHome Position)	202
25.16.	home (Snapshot Home Position).....	202
26.	Appendix A – anain options of different TANGOs	203
27.	Glossary	204
28.	Document Revision History	205

2. Introduction

Communication interface:

All TANGO controllers appear as a serial COM port, independent of the controller type (RS232C, USB, PCI, PCI-E). The default setting to open any TANGO COM Port is 57600,8,2,N. Most USB and PCI TANGOs transform this to much higher baudrates.

Axes:

TANGO controllers are available with up to 4 axes. The axis specifiers used in the TANGO instruction set are the ASCII characters *x*, *y*, *z*, and *a*. Axes can be addressed individually by using the axis specifier or combined if no axis is specified in the instruction.

Instruction syntax:

The instructions and parameters are sent as ASCII strings with a terminating carriage return [CR], which is 0x0d hex. Characters should be lower case, but upper and camel-case are also accepted. The parameters are separated by a space character. This provides easy access to all functions by using a simple terminal program such as HyperTerminal. A typical instruction syntax is as follows:

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

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

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

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

[instruction] Is the instruction word itself.

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

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

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

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

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

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

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

Syntax examples:

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

Moves:

Move instructions are executed as a vector move (except when in **ScanMode**). If several axes are started with one instruction they will reach their destinations at the same time. This means that - depending on velocity, acceleration and travel distance - one leading axis travels at its full velocity while the others follow synchronously. To move axes independently with their individual velocities, they have to be started separately by single axis instructions. Or **scanmode 3** can be set. Please refer to the '**move**' struction descriptions.

Settings:

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

Character limits:

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

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

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

Important: Secure speed limitation

The TANGO controllers have a built in safety 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 prevent the driven axis from 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 perform calibration and range measure after each power on:

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

Important: Measuring units

The measuring unit is set by the '**dim**' instruction, where dim 2 [which is mm] is the default setting.

In all dim settings except of dim 9 or 10, the velocity (vel) is in motor revolutions per second. Dim 9 provides the millimeter unit for most parameters¹, positions and velocities, while dim 10 does the same for μm units (vel in dim 10 is in mm/s not $\mu\text{m/s}$, it provides pitch- and gear-independent velocities, compatible to a 1mm pitch).

Extended mode:

In addition to the improvements when using dim 9 or 10, there is an option to enable "extended mode" behavior. It enables more functionality, like separate calibration, rm and joystick velocities, which else are derived from the axis velocity (vel). Please refer to the '**extmode**' instruction for further details.

Remarks:

** [!,?] Read/write specifier:

Even if not required (optional) with some instructions (e.g. moa, mor, m, go, etc.), the response in **autostatus mode 2** depends on the exclamation mark. If this special autostatus mode is required, it must be taken care of whether or not the [!] is used. Refer to **autostatus** for further information.

¹ Only **calbspeed** and **calrefspeed** are always in 1/100 revolutions/sec, even in dim 9 or 10

3. Remarks concerning the controller initialization

The TANGO controller must be configured to meet the hardware requirements. The configuration can be stored permanently with the **save** instruction. It is recommended to save and reboot the controller after changing the setup parameters (e.g. **!usteps**, **!pitch**, **!gear**) to ensure all changes will be applied.

- The axis units: **!dim**
- The **!extmode** (0 or 1)
- The axis **!pitch** (always in [mm], independent of **dim**)

Dim 9 and Extmode:

Using **dim=9** and **extmode=1** instead of **dim=2** will turn all units (also **vel** and **joyvel**) to [mm] and [mm/s]. **Extmode=1** offers bugfixes, more features (e.g. separate **joyvel**) and flexibility. But it has a slightly different behavior. Please refer to the **Extended Mode** description in this document.

From Firmware 1.73, **dim 10** can be used to replace **dim 1** → velocities in mm/s.

4. Brief Description of the TANGO Instruction Set

Controller Informations				
Instruction	Example	Save	Example description	Page
(?) version	version	-	Read detailed firmware and controller version	20
(?) det	det	-	Read the controller configuration	21
(?) detext	detext	-	Read the controller configuration and descriptive text list	21
(?) readsn	readsn	-	Read the controller serial number	22
(?) ver	ver	-	Read default version number	22
(?) iver	iver	-	Read further version number information	22
(?) uptime	uptime	-	Read how long the controller is running	23
(?) temp	temp	-	Read case temperature (avail. depends on controller type)	23
? maxaxis	?maxaxis	-	Read number of available axes	24
? maxcur	?maxcur	-	Show the maximum possible motor currents of all axes	24
? etspresent	?etspresent	-	Check availability of ETS	25
? stagesn	?stagesn	-	Read the axis serial numbers from ETS	26
? maxpos	?maxpos x	-	Read maximum available position range for X axis	27
? lockpos	?lockpos	-	Read the microscope stage transport lock (screw) position	27

Communication Interface Settings				
Instruction	Example	Save	Example description	Page
? ! baud	!baud 115200	Y	Set RS232 baud rate (here to 115200Bd, default is 57600)	29
? ! cts	!cts 1	Y	Enable CTS hardware handshake	29

System Instructions				
Instruction	Example	Save	Example description	Page
(!) save	save	-	Save parameters to controller nonvolatile memory	30
(!) restore	restore	-	Reload saved controller parameters from n.v. memory	30
(!) reset	reset	-	Reset controller (forces restart, similar to cycle power)	31
? ! pa	!pa 1	-	Power amplifiers ON (OFF=0), also refer to 'axis' instr.	32
? ! ipreter	!ipreter 2	Y	Select optional Venus instruction set	33

Operating Modes				
Instruction	Example	Save	Example description	Page
? ! autostatus	!autostatus 0	-	Select autostatus response type 0 (=disabled), range: [0-4]	34
? ! autopreset	!autopreset 0	Y	Set power-ip default for autostatus to 0	35
? ! extmode	!extmode 1	Y	Enable extended controller behavior	36
? ! scanmode	!scanmode 1	Y	Set positioning behavior to scanmode	38
? ! scanvel	!scanvel 0.5	Y	Set scanmode vector velocity to 0.5 mm/s	39
? ! modulomode	!modulomode a 1	Y	Set positioning behavior of A axis to turntable mode 1	40
? ! configdisplay	!configdisplay 1	Y	Enable PROFILER SCD CL position display on RS232	41

Controller States and Error Messages				
Instruction	Example	Save	Example description	Page
(?) statusaxis	statusaxis (or sa)	-	Read axis state [@,M,J, S,A,D,E,T,-]	42
(?) sta	sta	-	Read detailed axis state as 32bit HEX number	43
(?) calst	calst z	-	Read calibration state of the Z axis	44
(?) corrst	corrst x	-	Read state of position correction	44
(?) status	status	-	Read controller error state	45
(?) err	err	-	Read error number	45
(?) help	help	-	Read error number with additional descriptive text	46

**Controller States and Error Messages**

Instruction	Example	Save	Example description	Page	
(?)	service	service	-	Returns a detailed parameter and state list, for debugging	47
(?)	pci	pci	-	Returns 1 if controller is plugged in a PCI slot (desktop=0)	47
(?)	isvel	?isvel x	-	Read actual velocity of the X axis	47
(?)	iscur	?iscur x	-	Read the momentarily applied motor current of the X axis	48

General Adjustments

Instruction	Example	Save	Example description	Page	
?!	dim	!dim 1 1 1	Y	Set position measuring units of X Y Z to μm	49
?!	pitch	!pitch 1 1 1	Y	Set spindle pitch of X Y Z to 1 [mm/revolution]	50
?!	gear	!gear 1 1 1	Y	Set gear factor of X Y Z to 1	50
?!	motorsteps	!motorsteps x 200	Y	Set motor step type of X to 200 [steps/rev] for a 1.8° motor	51
?!	accel	!accel 0.1 0.1 0.1	Y	Set acceleration of X Y Z to 0.1m/s ²	52
?!	accelfunc	!accelfunc 1 1 0	Y	Set acceleration function X and Y to s-curve, Z to linear	53
?!	stopaccel	!stopaccel 2 2	Y	Set stop deceleration for X and Y to 2m/s ²	54
?!	vel	!vel 10 10 10	Y	Set axis velocity of X Y Z to 10	55
?!	velfac	!velfac 1 1 1	Y	Set velocity reduction factor for X Y Z to 1 (= no reduction), range is [0.01-1], use not recommended	56
?!	sevel	!sevel x 20	Y	Set secure speed limit X to 20mm/s (unit is always mm/s)	57
?!	cur	!cur 0.5 0.6 1	Y	Set motor current in Ampere: X=0.5 Y=0.6 and Z=1 A	58
?!	reduction	!reduction 0.5 0.5 0.5	Y	Set idle motor current reduction for X Y Z to 50%	59
?!	curdelay	!curdelay 1000 1000	Y	Set idle motor current reduction delay for X,Y to 1000 [ms]	60
?!	ecomove	!ecomove 30 30 0	Y	Set motor move eco-level of X and Y to 30% less current	60
?!	axis	!axis 1 0 -1	Y	Enable X, disable Y and switch off Z axis	61
?!	axisdir	!axisdir 0 1 0	Y	Reverse the direction of the Y axis (caution!)	61
?!	motortable	!motortable x 2	Y	Select custom motor correction table to type 2 for X axis	62
?!	usteps	!usteps 50000	Y	Set dim 0 microsteps per rev to 50000 (applies to all axes)	62
?!	resolution	!resolution 6	Y	Set position read resolutions to 1 nanometer	63
?!	backlash	!backlash 12.3 0 0	Y	Set backlash compensation to 12.3 μm in X and 0 in Y & Z	63
?!	lock	!lock 2 1	Y	Set write protection for parameter 2 (here: motor current)	64
?!	lockaxis	!lockaxis 0 0 0 0	Y	Remove lock protection from all axes (lock has no effect)	64
?	lockstate	?lockstate x	-	Read extended locked parameters, including internal limitations currently applied to X axis	65
?!	stout	!stout 2	Y	Make Status-LED state available at AUX I/O Pin VR_OUT	65
?!	noled	!noled 1	Y	Switch Status-LED permanently off	66
?!	updelay	!updelay -5000	Y	Wait to a maximum of 5 seconds for valid external power	66

Limit Switch Instructions (Hardware and Software)

Instruction	Example	Save	Example 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	67
?!	clim	!clim 50 70 10	-	Set circular limit at center X=50, Y=70 with radius 10	68
?!	limctr	!limctr x 0	Y	Disable axis limits for X axis, default = 1	69
?!	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	69
?!	limmode	!limmode 1	Y	Prevent moves that exceed a position limit	70
?!	swtyp	!swtyp 1 0 1 !swtyp y 0 0 0	Y	Set limit switch type for all axes to NPN (pull-up resistor) Set limit switch type for Y to PNP (pull-down resistor)	71
?!	swpol	!swpol 1 0 1 !swpol z 1 0 1	Y	Set limit switch polarity E0, EE for all axes (to active high) Set limit switch polarity E0, EE for Z (1=to active high)	72
?!	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	73
?!	swdir	!swdir x 1	Y	Swap CAL and RM endswitch assignment of X axis	74

**Limit Switch Instructions (Hardware and Software)**

Instruction	Example	Save	Example description	Page
(?) readsw	readsw	-	Read states of all limit switches (1=active and actuated)	75
(?) swin	swin	-	Read TTL signal level of all limit switch inputs (1=high)	75
(?) statuslimit	statuslimit	-	Read momentary limit status „A“ = calibration done „D“ = rm done „L“ = position limit(s) modified by software „-“ = no limits set yet	76

Calibration and Range Measure Instructions

Instruction	Example	Save	Example description	Page
(!) cal	cal	-	Perform a calibration move for all enabled axes, see 'axis'	77
(!) rm	rm x	-	Perform a range measure move in X	78
(!) vrm	vrm 75 50	-	Virtual range measure for a 75x50 microscope stage	79
? ! calmode	!calmode 2 2	Y	Set calibration/closed loop behavior X, Y to mode 2	82
? ! calrequired	!calrequired z 1	Y	Z axis does not move until '!cal' is executed	82
? ! caltimeout	!caltimeout 60 60 10	Y	Set calibration timeout for X and Y to 1 minute, Z to 10s	82
? ! caliboffset	!caliboffset 1 1 1	Y	Set the cal zero-point 1mm aside lower limit switch (dim 2)	83
? ! rmoffset	!rmoffset 1 1 1	Y	Set rm end-position 1mm aside upper limit switch (dim 2)	83
? ! caldir	!caldir z 0	Y	Set Calibration Direction/Type of Z to default (cal→E0)	84
? ! 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]	85
? ! calrefspeed	!calrefspeed 10	Y	Old instruction for globally setting the encoder ref search velocity in [rev/s]. Please use encrfvel instead.	85
? ! calpos	calpos	-	Read encoder position where the cal switch was released	86
? ! calposmot	calposmot	-	Read µstep position where the cal switch was released	86
(?) calabspos	calabspos	-	Read abs.-encoder pos. where cal switch was released	87
? calzeropos	?calzeropos	-	Read the "zero" position after cal (for calmode 1)	87
? ! calvel	!calvel x 10 0.5	Y	Only if extmode = 1: Set calibration velocities in X	88
? ! rmvel	!rmvel x 10 0.5	Y	Only if extmode = 1: Set range measure velocities in X	89
? ! autopitch	!autopitch x 1	Y	Measure pitch after cal move of X axis	89
? ! refdir	?refdir y	Y	(dummy) Read the direction for Y-axis REF switch search	90
? ! posshift	!posshift y 438.1338	Y	Set the Center Reference shift for X to 438.1338 mm	90

Move Instructions

Instruction	Example	Save	Example 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)	92
(!) 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	92
(!) m	m	-	Move relative again (use same parameters as defined by 'distance' or the last relative move (e.g. '!mor') instruction	93
? ! distance	!distance 1 1 1	-	Set distance for X Y Z 'm'-move (start with 'm' or '!m')	93
(!) moe	moe 5 10.2 25.3	-	Move X to 10.2 and Z to 25.3	97
(!)? moc	moc x	-	Move X to center position between lower and upper limit switch, or between lower and upper software limits	94
(!) mol	mol	-	Move all axes to their transportlock positions (fix. screw)	94
(!) go	go x 12.5	-	Move X to pos. 12.5, overwriteable, for tracking applications	95
? ! speed	!speed 5 5 5 !speed y 0	-	Let X Y Z axis travel at 5 [revolutions/s] Stop a running 'speed' or 'go' instruction on the Y axis	96
(!) a	a	-	Abort move (Stop)	98
? ! delay	!delay 1000	Y	Delay the start of move instructions by 1000 ms	99
? ! pause	!pause 10	Y	Delay "position reached" autostatus response by 10 ms	99

**Move Instructions**

Instruction	Example	Save	Example description	Page
? !	pos ?pos z	-	Set current X Y positions to 0 and Z position to 1.5 Read current Z position	100
(!)	posclr x	-	Reset the position offset that was caused by setting "!pos"	100
(!)	zero z	-	Set Z position and internal counter to 0 (e.g. filter wheel application)	101
(!)	clearpos z	-	Set Z position and internal counter to 0 (e.g. filter wheel application), not executable with measuring system	101

HDI Instructions (Joystick, Trackball, ERGODRIVE)

Instruction	Example	Save	Example description	Page
? !	joy !joy 2 !joy 0	Y	Switch joystick ON=2 or OFF**=0 (**Switching off may cause an @@@ autostatus response)	102
? !	joydir !joydir 2 -2 0	Y	Set HDI axis mode to X=ON, Y=Reversed, Z=OFF	103
? !	joychangeaxis !joychangeaxis 1	Y	Change Joystick X and Y axis (X→Y, Y→X)	103
? !	joywindow !joywindow 14	Y	Set idle window of the joystick center position, where an analog joystick deflection has no effect [0..100]	103
? !	joyvel !joyvel z 1.5	Y	Only if extmode = 1: Set joystick velocity for Z to 1.5	104
?(!)	joyspeed joyspeed 2 25	Y	Set joystick speed for speed button 2 "medium" to 25 rev/s	104
? !	keymode !keymode 2	Y	Select joystick key mode 2 = high speed preselection	105
? !	keyspeed !keyspeed x 5 20	Y	Set keymode joystick speed X low=5mm/s, high=20mm/s	106
?(!)	joycurve !joycurve z 1	Y	Set joystick characteristic for Z ot linear	106
(?)	key key	-	Read state of all joystick buttons (0=released, 1=pressed)	107
(?)	keyl keyl	-	Read and clear latched state of all joystick buttons	107
? !	hwfactor !hwfactor x 1	Y	One coaxial drive knob revolution in X is 1mm axis travel	108
? !	hwfactorb !hwfactorb x 14	Y	One coaxial drive knob revolution in X is 14mm axis travel	108
? !	hwfilter !hwfilter 0	Y	Deactivate coaxial drive noise reduction	108
? !	tbfactor !tbfactor 1 1	Y	Set trackball transmission factor in X and Y to default	109
(?)	zwheel ?zwheel	-	Returns 1 if HDI device has a multi-function wheel	110
?(!)	zwtravel !zwtravel 1 0.25	Y	Set default multi-function wheel travel to 2.5 mm/rev	110
? !	zwaxis !zwaxis a	Y	Assign multi-function wheel to A-axis	111
? !	zwfactor !zwfactor 1	Y	Set multi-function wheel factor to 1:1 (default)	111
?(!)	zwpos zwpos	-	Read independent multi-function wheel position counter	111
? !	tvrjoy !tvrjoy z	Y	Assign AUX I/O pulse&direction device to Z axis	112
? !	tvrjoyf !tvrjoyf 1	Y	Set tvrjoy transmission factor to 1	112
(?)	hdi hdi	-	Read ID number of the connected HDI device	113
! ?	hdimode !hdimode 0 1	Y	Set hdimode bit 0 to 1 for ERGODRIVE Toggle Mode	114
! ?	configaxsel !configaxsel 1	Y	Toggle joystick Z-axis between axes Z and A by F4 key	115

Digital and Analogue I/O

Instruction	Example	Save	Example description	Page
(?)	digin digin 8	-	I/O1 Extension module: Read all digital inputs I/O1 Extension module: Read digital input 8	118
? !	digout !digout 5 1 ?digout	-	I/O1 Extension module: Set digital output 5 to logic level 1 I/O1 Extension module: Read back all digital output levels	118
? !	diginpol !diginpol 5 1	Y	I/O1 Extension module: Invert input 5 signal	119
? !	digintyp !digintyp 111111	Y	I/O1 Extension module: Apply pull-up resistor to all inputs	119
? !	digoutpreset !digoutpreset 3 1	Y	I/O1 Extension module: Preset state of output 3 to high	120
(?)	edigin edigin 8	-	Multi I/O Extension: Read all digital inputs Multi I/O Extension: Read digital input 8	121
? !	edigout !edigout 5 1 ?edigout	-	Multi I/O Extension: Set digital output 5 to logic level 1 Multi I/O Extension: Read back all digital output levels	121
? !	ediginpol !ediginpol 5 1	Y	Multi I/O Extension: Invert input 5 signal	122

**Digital and Analogue I/O**

Instruction	Example	Save	Example description	Page	
? !	edigintyp	!edigintyp 111111	Y	Multi I/O Extension: Apply pull-up resistor to input 0-5	122
? !	edigoutpreset	!edigoutpreset 3 1	Y	Multi I/O Extension: Preset state of output 3 to high	123
? !	edigrly	!edigrly 1	-	Multi I/O Extension: Switch optional relay on	123
(?)	adigin	adigin adigin 2	-	Read all AUX I/O digital inputs Read logic level of AUX I/O digital input 2 only	124
! ?	adigintyp	!adigintyp 1111		Apply pull-up resistors to the first 4 adigin inputs	124
! ?	adiginfunc	!adiginfunc 2 0 0 1		Select adigin 0 as snapshot and adigin 3 as stop input	125
? !	adigout	!adigout 3 1	-	Set AUX I/O digital output 3 to high (+5V)	126
? !	adigoutpreset	!adigoutpreset 3 1	Y	TANGO Desktop HE: Set AUX I/O digital output defaults	126
(?)	anain	anain c 10	-	Read input of analogue channel 10 (the analog input)	127
? !	anaout	!anaout c 1 17.5	-	Set analogue output voltage of channel 1 to 17.5 percent	128
? !	anamode	!anamode 0	(Y)	Set analogue output mode to default behavior	129
? !	stoppol	!stoppol 1	Y	Set AUX I/O stop input to active high	131
? !	stop	!stop 0	-	Release stop condition (in latched stoppol modes only)	132
? !	stopl	?stopl	-	Check if an AUX I/O stop condition was present	132
? !	shutter	!shutter 1	-	Set AUX I/O shutter out signal to TTL high	132
(!)	flash	flash 0.1	-	Send a 100µs high pulse to AUX I/O TAKT_OUT (LED)	133
? !	tvr	!tvr z 5	Y	Enable AUX I/O TVR pulse and direction for Z axis	133
? !	brake	!brake 0 0 1	Y	Enable Brake for Z axis on I/O extension output pin OUT0	134
? !	vbrake	!vbrake 24 14	Y	TANGO Desktop HE: Set brake open and hold voltages	136
? !	brakedelay	!brakedelay 50	Y	TANGO Desktop HE: Set open to hold voltage delay [ms]	136
(!)	brakepos	brakepos z	-	Move to initial motorpole position to avoid power-up "jump"	137
? !	drop	!drop 5	-	Liquid Dispenser: generate 5 drops or dispense for 5 sec.	138
? !	pump	!pump 1	-	Liquid Dispenser: Switch on air pressure pump manually	139
? !	vbus	?vbus	-	Read on/off state of the +24V supply output	140
? !	configvbus	!configvbus 1	Y	Set +24V supply output to always on	140
? !	vusb	!vusb1	-	TANGO Desktop HE: Switch on USB-A connector +5V	141
? !	configvusb	!configvusb 1	Y	TANGO Desktop HE: Set USB-A +5V to always on	141
? !	configcanres	!configcanres 1	Y	TANGO Desktop HE: Activate CAN Bus 120Ω termination	141

Encoder Instructions

Instruction	Example	Save	Example description	Page	
? !	encmask	!encmask 1 1 0	Y	Enable the activation of X and Y encoders, disable Z	142
? !	enc	!enc 1 0	-	Manually activate X encoder (caution!), set Y to inactive	143
? !	encperiod	!encperiod 0.1	Y	Set signal period of X encoder to 100 µm	144
? !	enctype	!enctype x 1	Y	Set encoder type of X to TTL (no analogue sin/cos signal)	146
? !	enctl	!enctl x 1	Y	Set encoder type of X to TTL (no analogue sin/cos signal)	147
? !	encdir	!encdir y 1	(Y)	Reverse counting direction for Y encoder (caution!)	144
? !	encvel	!encvel x 0.5	Y	Set auto-adjust velocity of X encoder to 0.5mm/s	145
? !	encrefvel	!encrefvel x 5	Y	Set encoder ref search velocity in X to 5	145
? !	encref	!encref 1 1 0	Y	Enable encoder reference signal for X and Y, disable for Z	147
? !	encnas	!encnas 1 0 0	Y	Enable NAS error signal input encoding for X encoder only	148
(?)	encrefstatus	encrefstatus x	-	Read X encoder reference signal state (1=on reference)	148
(?)	encrefstatusl	encrefstatusl x	-	Read latched X encoder reference signal state	148
(?)	encnasstatus	encnasstatus x	-	Read X encoder NAS signal state (1=NAS error)	149
? !	encerr	!encerr 0	-	Clear encoder error state for X axis (? response is 0 or e)	149
? !	encamp	?encamp x	-	Read X encoder signal amplitude in percent	150
? !	encpos	!encpos 1	-	?pos instruction returns for the encoder positions, if enc=1	151
? !	configencpos	!configencpos 0 0 1	Y	Preset value for encpos at power up (default=0)	152
(?)	encsync	encsync -1	-	Read the state of the encoder signal synchronization	153
(?)	hwcount	hwcount	-	Read all encoder positions (TTL counter, non-interpolated)	153
(!)	clearhwcount	clearhwcount x	-	Set hwcount encoder position counter to zero, here: X axis	153

MR Encoder Instructions

Instruction	Example	Save	Example description	Page
? ! mra	?mra x	-	Read amplitude correction factor (sin/cos ratio) of X	154
? ! mro	?mro	-	Read offset correction value for all encoders	154
? ! mrp	!mrp x 0 0 0 0	-	Reset MR-signal peak-to-peak measurement result of X	155
? mrt	?mrt z 2	-	List two measurement results of the Z encoder sin cos	155

Absolute Encoder Instructions

Instruction	Example	Save	Example description	Page
? ! encform	!encform 24 24	Y	Set the absolute data word size in X+Y to 24 bit	156
? ! encres	!encres y 78.125	Y	Set the absolute data resolution in X to 78.125 nm	156
(?) abspos	abspos	-	Read abs.-encoder pos. at the current position	157
(?) calabspos	calabspos	-	Read abs.-encoder pos. where cal switch was released	87
? ! posshift	!posshift y 438.1338	Y	Set the absolute zero alignment for X to 438.1338 mm	90

Closed Loop Instructions

Instruction	Example	Save	Example description	Page
? ! ctr	!ctr 2 2 2	Y	Set closed loop mode of X Y Z to always active (=default)	161
? ! ctrf	!ctrf 2.0	Y	Closed loop factors for X axis are set to 2.0 (old)	162
? ! ctrff	!ctrff 2 3.5	Y	Closed loop factors for X axis are set to 2 and 3.5	162
? ! ctrd	!ctrd 100	Y	Closed loop in target window for 100 milliseconds	163
? ! ctrt	!ctrt 200	Y	Closed loop control timeout after 200 milliseconds	163
? ! twi	!twi 0.01 0.01 0.01	Y	Set target window for X Y Z to 10µm (assume dim=2)	164
? ! ctrc	!ctrc 3	Y	Closed loop control is called every 3 milliseconds	164
? ! ctrsm	!ctrsm x 1	Y	Set X behavior outside lock-in range to "slow closed loop"	165
? ! ctrs	!ctrs x 0.2	Y	Set lock-in range for X to 0.2mm (assume dim=2)	165
? ctrstatus	?ctrstatus 1	-	Get Closed Loop active state of all axes	166
? ctrdiff	?ctrdiff	-	Get Closed Loop position difference of all axes	167

**Trigger Output Functionality¹**

Instruction	Example	Save	Example description	Page
? ! trig	!trig 1	-	Enable trigger functionality (the setup must be complete)	168
? ! trigm	!trigm 0	Y	Select trigger mode 0	169
? ! triga	!triga x	Y	Trigger function is related to X axis	171
? ! trigo	!trigo 1	Y	Select the default trigger signal output	171
? ! trigs	!trigs 40	Y	Set trigger output signal length to 40 microseconds	172
? ! trigd	!trigd 10	Y	Set trigger distance to 10 (mm if dim=2)	171
? ! trigcomp	!trigcomp 50	Y	Compensate a trigger signal delay of 50µs	173
? ! trigenc	!trigenc 1	Y	Select encoder signal as trigger source (if available)	174
? ! trigf	!trigf 1000	Y	Set periodic trigger pulse frequency to 1kHz	174
? ! trigbdelay	!trigbdelay 15.05	Y	Set Precise secondary trigger out to 15.05µs delayed	175
? ! trigbwidth	!trigbwidth 4.35	Y	Set Precise secondary trigger out signal length to 4.35µs	175
? ! trigbf	!trigbf 66000000	Y	Set Precise secondary trigger out frequency to 66 MHz	175
? ! trigcount	?trigcount	-	Read number of generated trigger events	176
(!) trigger	trigger	-	Manually set trigger output (available in trigm 102, 103)	176
? ! trigr	!trigr 5 10 6	-	Generate 6 trigger signals from 5 to 10→5,6,7,8,9,10 (mm)	177
? ! trigp	!trigp -1 12.5	-	Append a position value to the trigger position list	180
? ! trigc	?trigc	-	Read number of entries in trigp trigger position list	183
? ! trigi	?trigi	-	Read or manipulate the momentary list index of trigp, trigr	183
? ! trigl	!trigl 1	Y	Set trigger signal level for !trigr and !trigp to active high	184
? ! trigsns	!trigsns 1	Y	Enable trigger generation on snapshot events	184

Snapshot - Trigger Input Functionality²

Instruction	Example	Save	Example description	Page
? ! sns	!sns 1	-	Enable snapshot functionality (always 1 after power up)	191
? ! snspreset	!snspreset 0	Y	Set power-up default for sns to 0 (disabled)	191
? ! snsfl	!snsfl 0	Y	Set snapshot input signal to active low	192
? ! snsf	!snsf 10	Y	Set snapshot signal debounce filter to 10 milliseconds	192
? ! snsrm	!snsrm 0	Y	Set snapshot mode to 0 (0=capture pos, 1=move to pos)	193
? ! snsc	?snsc	-	Read number of snapshot events (=array fill size)	196
? ! snsi	!snsi 5	-	Set snapshot index to 6 th entry (snsa 6)	196
? ! snsaxis	!snsaxis 1 1 0 0	-	X and Y axis moves wait for snapshot (in snsm mode 6)	197
? ! snsp	?snsp x	-	Read last captured X position	198
? ! snsa	?snsa 1	-	Read first position entry of snapshot array (all axes)	199
(!) snse	snse 2	-	Generate SnapShot event F2	200
? ! snsv	?snsv 3	-	Read captured ANIN0 voltage in mV of 3 rd snapshot entry	200
? ! snsj	!snsj 1 0.5 0 0	-	Set snapshot jump distances (snapshot mode snsm 9)	201
? ! prehome	!prehome 10 20 1	-	Set prehome positions X Y Z to 10 20 1 (unit depends on dim setting)	202
? ! home	!home 5 5 0	-	Set home positions X Y Z to 5 5 0 (unit depends on dim setting)	202

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 read and write 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	! or ? is missing or not allowed
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 actuated
13	function not executable, because encoder detected
14	error during calibration (limit switch not released)
15	error during calibration (opposing limit switch actuated)
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 amplifiers are disabled (switched OFF)
30	safety circuit out of order
32	move discarded target outside limit
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, deviation or enc. error
75	could not enable axis correction, or axis correction was disabled
76	io extension error (output overload on IO1 or Multi-IO connector)
77	io/xPos internal bus communication error
78	HDI input device error
79	xPos module error
80	internal error: HDI ISR not running
81	internal error: Encoder ISR not running
82	overload on motor connector +5V (PCI-E/DT-E: also on +5V of AUX I/O)
83	overload on AUX I/O +5V supply
84	overload on encoder +5V supply
85	overload on AUX I/O +12V supply or AUX mini +24V supply
86	low brake output voltage
87	overload on motor 4 connector +5V
88	overload on a supply output pin (latched overload state), clear by "!err"
89	not executable while in standby mode
90	temperature error
91	encoder error



7. Controller Informations

The firmware version may be read by sending the instruction '**version**' to the controller. The instruction '**det**' gives 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, 1 or 2

Description: Read the TANGO type and firmware version.

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

From TANGO firmware 1.73, Sending "version 2" returns the type no. and firmware version of a handling system firmware, which is independent from the controller firmware base. Regular TANGO controllers return 0.00.

An exception is the TANGO 3 mini, where "version 2" returns the co-processor firmware version.

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

TANGO-DT-S, Version 1.57, Apr 17 2012 , 12:12:02

TANGO-DT	Desktop version, PCI card based
TANGO-DT-S	Desktop version, PCI-S card based
TANGO-DTe	Desktop version, PCI-E card based
TANGO-PCI	PCI card
TANGO-PCI-S	PCI-S card
TANGO-PCIE	PCI-E card (PCI Express)
TANGO-MINI	TANGO mini
TANGO-C	Motorized stage with integrated controller
TANGO-I	TANGO integrale
TANGO-MINI3	TANGO 3 mini
TANGO-Desktop	TANGO Desktop HE
Version 1.57	Firmware version number
Apr 17 2012	Firmware build date
12:12:02	Firmware build time

Example:

```
?version ==> TANGO-DTe, Version 1.69, Mar 6 2018 , 15:52:19
?version 1 ==> 1.69
?version 2 ==> 1.10 (TANGO 3 mini: Co-Processor Firmware version 1.10)
?version 2 ==> 0.00 (Other TANGOs: Default TANGO Firmware)
?version 2 ==> 1.01 (Other TANGOs: System Firmware of Type 1, Version 01)
```

7.2. det (Read detailed Configuration)

Syntax: ?det or det
Parameter: none

Description: Read detailed information of the controller configuration.

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

```
0x0 0 0 0 0 0 1   Configured encoder interface = 1Vpp
0x0 0 0 0 0 0 2   Configured encoder interface = MR
0x0 0 0 0 0 0 4   Configured encoder interface = TTL
0x0 0 0 0 0 0 8   Configured encoder interface = Universal 1V+MR
0x0 0 0 0 0 3 0   The number of configured axes (0 to 4, here 3)
0x0 0 0 1 0 0 0   Display is configured (PROFILER SCD CL @RS232)
0x0 0 0 2 0 0 0   Speedpoti is configured
0x0 0 0 4 0 0 0   Hand wheel is configured
0x0 0 0 8 0 0 0   Snapshot (Trigger in) is configured
0x0 0 1 0 0 0 0   TVRin is configured
0x0 0 2 0 0 0 0   Trigger out is configured
0x0 0 8 0 0 0 0   TVRout is configured
0x0 1 0 0 0 0 0   Digital extension: I/O1      (24in+8out)
0x0 2 0 0 0 0 0   Digital extension: Multi I/O (12in+8out)
0x0 4 0 0 0 0 0   Trackball is configured
0x0 8 0 0 0 0 0   ETS is connected
0x1 0 0 0 0 0 0   xPos Module (3 axis extension)
0x2 0 0 0 0 0 0   Absolute encoder interface
```

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

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

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

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

7.3. detext (Read Extended detailed Configuration)

Syntax: ?detext or detext
Parameter: none

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

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

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

7.4. readsn (Read Serial Number)

Syntax: ?readsn or readsn
Parameter: none

Description: Reads the serial number of the TANGO controller.

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

YY year of manufacturing
WW week of manufacturing
T controller type identifier (0-9,A-Z)
N in hardware available axes (may differ from ?maxaxis)
XXX index number

Example: ?readsn => 140413045

7.5. ver (Read default Version Number)

Syntax: ?ver or ver
Parameter: none

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

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

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

Example: ?ver => Vers:LS32.00.038 (=3 axis 2.5 A)

7.6. iver (Read internal Version Number)

Syntax: ?iver or iver
Parameter: none

Description: Read the internal version information string. Mostly unused.
To read the TANGO firmware version, please use '**version**'.

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

Example: ?iver => T04.35.02-0004

7.7. uptime (Read Controller Up Time)

Syntax: ?uptime or uptime, !uptime
Parameter: none, 1 or -2, or uptime millisecond offset as integer

Description: Returns the power-on time of the controller since it was switched on or resetted. Depending on the parameter (none, -2 or 1) the readout can be in seconds, in hh,mm,ss or in milliseconds.

The milliseconds can be manipulated. It is possible to to set the milliseconds to zero or a specified positive value. A negative value will clear the milliseconds offset and return to the true internal time. The seconds and hh, mm, ss readout are not affected by the millisecond manipulation. They always return the true uptime.

Remarks: New TANGO Desktop HE controllers support parameter 3, which returns the total operating hours time of the controller as days, hours, minutes and seconds.

Response: Time in seconds
 or in milliseconds with optional time offset
 or in hours, minutes and seconds
 Please refer to the examples below.

Example: ?uptime => 4503 (time in seconds)
 uptime => 4503 (same as ?uptime)

 ?uptime -2 => 01h 15m 03s (time in hh,mm,ss)

 ?uptime -3 => 7d 05h 15m 21s (Desktop HE only)

 ?uptime 1 => 45033821 (time in milliseconds)

 !uptime (sets the milliseconds uptime time to zero)
 ?uptime 1 => 2
 !uptime 1000 (sets the ms uptime time to 1000ms)
 ?uptime 1 => 1002
 !uptime -1 (clear the millisecond offset)
 ?uptime 1 => 45033826

7.8. temp (Read Case Temperature)

Syntax: ?temp or temp
Parameter: none, 1 or 2

Description: Read the ambient temperature inside the controller.
Remarks: Not all TANGO controllers provide a temperature sensor, New controllers provide two (board/ambient and CPU).

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

Example: temp => 28.5
 temp 1 => 28.5 43.8 (Desktop HE only: Ambient + CPU)
 temp 2 => 51.7 75.3 (Desktop HE only: Max since power on)

7.9. maxaxis (Read number of available Axes)

Syntax: ?maxaxis
Parameter: none

Description: Read the number of available (factory configured) axes.
1, 2, 3 or 4 (single axis Z controllers return a 'z').

Response: Number of available axes.

Example: ?maxaxis => 3

7.10. maxcur (Read Maximum Motor Current)

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

Description: Read the maximum possible motor current (amplifier capability) and the maximum current which can be set by **!cur**.
The maximum possible current depends on the capability of the motor amplifier (e.g. 1.00/1.25/2.50/3.75A) and a factory-set limitation to protect the motor from overcurrent.

No parameter: Maximum current the power amplifier can deliver.
Parameter 1: Maximum possible Motor current, including additional limitation (by ETS factory entry).

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

Examples:

```
?maxcur          read maximum amplifier current of all available axes
?maxcur y        read maximum amplifier motor current of Y axis only
?maxcur y 1      read maximum motor current of Y axis only
?maxcur 1        read maximum motor current of all available axes
```


7.11. etspresent (Read ETS Detect State)

Syntax: ?etspresent
Parameter: none

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

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

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

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

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



7.12. stagesn (Read Connected Devices Serial Number)

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

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

Call parameter options:

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

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

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

Example: One ETS connected to the XY stage (at ETS address 0)

```
?stagesn      => 140328015 -----
?stagesn -1   => 140328015      (The ETS at address 0 is the "first" ETS)
?stagesn 0    => 140328015      (The ETS at address 0)
?stagesn 1    => -----
```

One ETS connected to the Z axis (here at ETS address 3)

```
?stagesn      => ----- 140328015
?stagesn -1   => 140328015      (The ETS at address 3 is the "first" ETS)
?stagesn 0    => -----
?stagesn 1    => -----
?stagesn 2    => -----
?stagesn 3    => 140328015      (The ETS at address 3)
```

Two ETS connected to the XY stage and the Z axis (here ETS addresses 0 and 1)

```
?stagesn      => 151125014 160714003 -----
?stagesn -1   => 151125014      (The ETS at address 0 is the "first" ETS)
?stagesn 0    => 151125014      (The ETS at address 0 is the "first" ETS)
?stagesn 1    => 160714003      (The ETS at address 1)
?stagesn 2    => -----
```

7.13. maxpos (Maximum Position)

Syntax: ?maxpos

Parameter: x, y, z, a or none

Description: The maximum position value which the controller can accept. It depends on the configured **pitch**, **gear** and **motorsteps**.

Remarks: A move instruction must not exceed the maxpos distance, the maximum travel range of the axis is within \pm maxpos.

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

Example: ?maxpos => 2600.0000 2600.0000 2600.0000
?maxpos x => 2600.0000 (positioning from -2600mm to +2600mm)

7.14. lockpos (TransportLock Position)

Syntax: ?lockpos

Parameter: x, y, z, a or none

Description: Read the factory set transportlock position, where the axis must be positioned to for the fixation screw/transportation. A value of zero indicates the position is not available.

Remarks: Refer to the **!mol** instruction

Response: Transportlock position

Example:
?lockpos => 50.3815 38.0018 0.0000
?lockpos y => 38.0018 (Y axis transportlock position)
?lockpos z => 0.0000 (Z axis provides no transportlock position)

7.15. cmdinfo (List of Available Instructions)

Syntax: ?cmdinfo or cmdinfo

Parameter: none, 1 or 2
And an optional 1

Availability: Only with TANGO Desktop HE.

Description: Provides the number of instructions
or a multi-line list of available instructions.

The first parameter stands for the interpreter (ipreter),
whoms informations shall be returned.

Parameter1 = 1 or none: TANGO Native instructions
Parameter1 = 2 : Venus instructions

The optional second parameter "1" lists the instructions
of the requested interpreter as multi-line Text, the last
line terminates with "end."

Remarks: The number of instructions might not corrsespond with the
number of listed instructions (might contains less lines).

Response: Number or a multi-line ASCII-Text, last line is "end."

Example: cmdinfo 1 => 500 (amount of TANGO Native instructions)
cmdinfo => 500 (same as cmdinfo 1)

cmdinfo 2 => 241 (amount of Venus-1/-2 instructions)

cmdinfo 1 1 => err (list of TANGO Native instructions)
pos
readsw
...
end.

cmdinfo 2 1 => m (list of Venus instructions)
move
r
rmove
nm
nmove
...
end.

8. Communication Interface Settings

8.1. baud (Baud Rate)

Syntax: !baud or ?baud
Parameter: 1200, 2400, 4800, 9600, 19200, 38400, **57600** or 115200

Description: Set or read the baudrate of the serial COM Port interface(s).
The default baudrate to connect a TANGO is 57600.

When changing the baud rate on a true RS232C interface, the PC COM port also must change its baud rate to maintain communication (close and then re-open the COM Port with the new baud rate).

Then a '**!save**' instruction may be sent to permanently store the new baud rate in the controller.

Some TANGOs like (Desktop HE) provide two RS232 interfaces. When only using the main interface, the baud instruction works identical as with one interface only. The individual baud rates can be set with additional port identifier 1 or 2, refer to examples below.

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

Response: Current baud rate(s) of the controller

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

Examples 2 RS232: !baud 1 57600 Set baud rate of the 1st RS232 interface
 !baud 2 115200 Set baud rate of the 2nd RS232 interface
 !baud 57600 115200 Set baud rate of both RS232 interfaces
 ?baud 1 Read baud rate of the 1st RS232 interface
 ?baud 2 Read baud rate of the 2nd RS232 interface
 ?baud -1 Read baud rate of both RS232 interfaces

8.2. cts (Enable/Disable RS232 Hardware Handshake)

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

Description: Only supported by TANGO PCI-S/DT-S and PCI-E/DT-E controllers.
Enable or disable RTS/CTS hardware.

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

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

Response: Currently selected cts mode 0 or 1

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

9. System Instructions

9.1. save (Save Parameters)

Syntax: !save or save

Parameter: none

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

For Desktop HE it is possible to read the write-cycles of the save instruction on user and configuration memory with ?save.

Remarks: Best practice is to send a save instruction only when all axes are idle. At least for PCI-S/DT-S, PCI-E/DT-E controllers with 4 axes, a save should not be executed while the 4th axis is traveling. And for a TANGO 3 mini, a save instruction should not be executed while any axis is traveling in closed loop.

Response: ASCII string "OK...", or "ERR" with error number

Example: save => OK... (The currently used parameters are stored permanently and used as new power-up defaults)

9.2. restore (Restore Saved Parameters)

Syntax: !restore, restore or ?restore

Parameter: none or -1

Description: Reload or reset the saved parameters.

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

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

?restore returns "OK..." or "ERR" (like the save instruction), Executing restore without a question mark does not reply. ?err then might be sent once after the restore instruction and will reply the err response (e.g. a 0) after restore has completed.

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

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

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

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

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

9.3. reset (Force a Software Reset)

Syntax: !reset or reset
Parameter: none

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

Remarks: New TANGO controllers (Desktop HE) with built-in USB interface will also disconnect the USB connection during reset. Therefore, when not communicating via newer TANGO DLLs (1.399 or later), the virtual COM port must be disconnected and reconnected after sending the reset command (retry reconnect until successful, then continue as usual).

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

Response: none

Example: reset

9.4. pa (Enable or Disable the Power Amplifiers)

Syntax:	<code>!pa</code> or <code>!poweramplifier</code> <code>?pa</code> or <code>?poweramplifier</code>
Parameter:	0, 1, 2 or 3, the options for read are: none or -1
Description:	<p>Switch all motor amplifiers on/off or read amplifier state.</p> <p>If switched off, no motor current is flowing (high impedance).</p> <p>Amplifier switch off can also be caused by a short circuit, the PSE pin functionality or loss of motor supply voltage. Then an internal error (error 29) is generated and the status LED flashes. <code>?pa</code> will return 0. After removing the switch-off cause, the amplifiers can be switched on again, e.g. by <code>!pa 1</code>.</p> <p><u>PARAMETERS FOR THE SET INSTRUCTION (!pa)</u></p> <p>0 : Amplifiers off 1 : Amplifiers on 2 : Special Closed Loop switch on instruction, see remarks 3 : Similar to 1, but keeps the cal and rm limits (caution!)</p> <p><u>PARAMETERS FOR THE READ INSTRUCTION (?pa)</u></p> <p>none = read general amplifier state: 1 = ok, 0 = error or off -1 = read individual amplifiers : 1 = on, 0 = off</p>
Remarks:	<p>To switch off individual axes, use the 'axis -1' instructions.</p> <p>If the amplifiers switched off due to any condition or by the "<code>!pa 0</code>" instruction, and the axes were in active closed loop state before, it is possible to switch them on again and reactivate the closed loop at the current axis positions by sending "<code>!pa 2</code>". A cal or rm sequence is not required and the momentary measuring system positions of the axes are applied. For axes without encoders, "<code>!pa 2</code>" will behave like "<code>!pa 1</code>". "<code>!pa 3</code>" is only for open loop applications and where the axis positions did not change while the amplifiers were off.</p> <p>When using <code>!pa 1</code> (or using <code>!pa 2</code> without active closed loop), the cal and rm sequence must be executed again.</p>
Response:	amplifier state 0 = Amplifiers are off or at least one amplifier switched off (by error, PSE, <code>!pa0</code>) 1 = Amplifiers are on
Example:	<pre>!pa 0 Switch all amplifiers off !pa 1 Switch on all amplifiers, closed loop not recovered CAL/RM has to be repeated due to uncertain position. !pa 2 Switch on all amplifiers, closed loop is recovered. If axes were in closed loop before they switched off, they continue at their momentary position and CAL/RM is not required. !pa 3 Switch on all amplifiers, closed loop not recovered CAL/RM state is recovered (→ useful in open loop) ?pa Read amplifier state (e.g. returns 1) ?pa -1 Read amplifier state of all axes (e.g. returns 1 1 1)</pre>

9.5. ipreter (Select Instruction Set)

Syntax: !ipreter or ?ipreter

Parameter: 1, 2, (3, 4 or 5)

Description: Instruction set of the TANGO controller.
The TANGO controllers provide up to 5 different instruction sets, also called "interpreter". At least interpreter 1 and 2 are implemented.
It is recommended to use the default interpreter 1, as it is the native language and supports the most complex instruction set available.
Other instruction sets may be used for compatibility to existing applications. They are described in separate manuals.

INTERPRETER NR.

```
-----  
0   Not supported! (old register instruction set)  
  
1   TANGO instruction set (default),  
    as described in this manual  
  
2   VENUS-1 and VENUS-2  
  
3   LUDL MAC5000  
  
4   ASI MS-2000 (available with Firmware >= 1.46)  
  
5   PRIOR ProScanII  
-----
```

Remarks: After changing the interpreter, the new setting must be stored by a save instruction in the syntax of the new interpreter.
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

Example:
!ipreter 2 Switch the interpreter to VENUS instruction set
?ipreter (The instruction from this instruction set always returns 1)

10. Operating Modes

10.1. autostatus (Set Autostatus Behavior)

Syntax: !autostatus or ?autostatus

Parameter: 0, 1, 2, 3 or 4

Description: Select auto response behavior for completion of move and calibration instructions. The power-on default is mode 1. Autostatus mode cannot be saved by the **'save'** instruction. Move instructions !moa,!mor,m,!moc,!mol behave identical. Calibrate instructions are slightly different, ('A','D').

Remarks: *The power-up default (1) can be overwritten by **autopreset**.

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

1 : **Default state at power on*. Recommended.**
Position reached response after an automatic move (!moa,!mor,m,!moc,!mol) as a 5 character string with e.g. '@' for each configured axis: "@@@@." !cal returns an 'A' instead of '@' and !rm returns a 'D'.
Disabling the joystick by "!joy 0" also causes a response.
Please refer to **'statusaxis'** for further explanation of the reply.

2 : Instructions with a leading "!" are immediately acknowledged by the **status message** ("OK..." or "ERR"). Move and Cal/Rm instructions will respond additionally like in autostatus 1 (e.g. "AAA-" response).

3 : Similar to the default mode 1, but a simple <CR> (0x0d hex) is returned to indicate that the move is completed. It can be used to improve performance for higher vector throughput, but contains less information e.g. concerning possible errors.

4 : Echoes the instruction that was sent, in case of set or move instruction.

Example: !autostatus 0 disable autostatus (**'statusaxis'** must be polled to identify if the axis is traveling or stopped)
 !autostatus 1 set autostatus to the default behavior
 ?autostatus read the currently selected autostatus

--- autostatus 0 ---

!moa,!mor,m,a,cal,rm ==> [returns nothing, statusaxis (sa) must be polled]

--- autostatus 1 --- (the default mode after power on*)

!moa,!mor,m,!moc,!mol ==> [returns completion, 5 ASCII character string @@@-.]
!cal,!rm ==> [returns completion, 5 ASCII character string AD@-.]

--- autostatus 2 ---

!vel 10 ==> [returns "OK..."]
!vel -10 ==> [returns "ERR 5"]
vel 10 ==> [returns nothing (is executed but no "!" reply)]

--- autostatus 3 ---

!moa,!mor,m,a,cal,rm ==> [returns completion, 0 ASCIIIs only <CR> termination]

--- autostatus 4 ---

!moa 10 5 0 ==> [returns !moa 10 5 0]

10.2. autopreset (Preset for autostatus)

Syntax: !autopreset or ?autopreset

Parameter: 0, 1, 2, 3 or 4

Description: Set or read the predefined power-up setting for **autostatus**. The TANGO will use it as preset value for **autostatus** after power-up or reset. Default = 1.

Remarks: As the **autostatus** setting is volatile and commonly used, e.g. by SwitchBoard or the DLL, it should not be storable. In cases where a certain autostatus mode is required at power-up, autopreset can be used (from Firmware 1.73).

Response: Selected autostatus mode for power-up and reset.

Examples:

```
!autopreset 0    Set power-up preset for autostatus to 0
?autopreset      Read the autostatus preset value
```

10.3. 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 a limit switch 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 Limit switch. The unit is 1/100 rev/s.

Calibration in extmode = 1:

!vel has no influence on the cal / rm move, also '**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 switch (E0)

!rmvel --> Set velocities for moving towards and out of the rm switch (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. 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.

10.3.1 extmode (Switch to Extended Mode)

Syntax: `!extmode` or `?extmode`

Parameter: 0 or 1

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

0 = default, compatible interpreter mode

1 = extended interpreter mode

Response: currently used extmode, 0 or 1

Examples:

`!extmode 1` Set controller behavior to extended mode

`?extmode` Read extended mode setting

10.4. Scan Mode

By default, with ScanMode deactivated, a single axis move (moa or mor x,y,z,a) causes the specified axis to travel at its own velocity and in case several axes are specified (e.g. moa 10 5 3), up to four axes travel as a vector in order to reach their target positions at the same time. The velocity is recalculated in a way that none of the involved axes exceeds its velocity or acceleration limits.

This means the resulting vector velocity of a multi-axis vector move depends on the axis velocities and travel distances.

Scan Modes 1, 2 and 3 change this behavior:

In some cases, it's required that the resulting vector travel velocity is always the same, independent in which direction or angle the vector is pointing. This "continuous path velocity" (parameter '**scanvel**') is applied in ScanMode 1.

ScanMode 2 limits this behavior to multi-axis move instructions only. It causes single axis moves to use their individual velocity (**vel**), which can be used e.g. to drive the Z axis independently while X and Y still travel as a continuous path vector.

ScanMode 3 entirely disables vector moves. All moves, no matter if single axis or multi-axis, are traveling at their own velocity (**vel**) setting and so may reach their target position at different times (they stop individually). This mode is useful if the system configuration consists of individual axes.

ScanMode 3 might also be of advantage in stop-and-go line scans for image stitching, where X or Y is the scan axis and Z is required to follow a focus map = each position consists of an X position with its individual Z value. One axis might have already settled in its target position then (no more oscillating) when the other axis arrives. This might give an advantage in overall speed compared to the default mode, where the X and Z then would arrive (and oscillate and settle) in their target position at the same time.

10.4.1 scanmode (Scan Mode to change Axis Vector Move Behavior)

Syntax: !scanmode or ?scanmode

Parameter: 0, 1, 2 or 3

Description: This instruction switches the TANGO controller into scan mode, which may be used for continuous path control (modes 1,2) or special requirements of move behavior. Modes 1 and 2 apply a constant vector velocity for automatic moves (moa, mor) which is set by '**scanvel**'.

0 = normal operation (the default TANGO mode)

1 = scan mode 1

2 = scan mode 2

3 = no vector moves (else identical to mode 0)

Scan mode 0: Normal operation (TANGO default)

- Single axis moves travel at their individual vel and accel.
- Vector moves are calculated to reach the target position at the same time without exceeding a velocity or acceleration of any of the involved axes. Scanvel is not used.

Scan mode 1: Resulting move velocities are always scanvel

- 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"→scanvel, "!mor 20 20"→x=y=scanvel/sqrt(2)

Scan mode 2: Only vector moves are executed with scanvel

- 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"
- Scanvel only applies to vector moves of 2 or more axes, e.g. "!moa 10 20"

Scan mode 3: No vector moves

- Similar to mode 0 (normal operation, not a mode as 1 or 2).
- When using vector moves (move instruct. with several axes), each axis travels at its individual **vel** and **accel** settings.

Response: Scanmode (automatic move mode) as integer

Examples:

!scanmode 1 Set controller into scanmode 1 (always traveling at scanvel)
?scanmode Read controller scanmode

!vel 20 20 10 (example: vel x,y = 20, vel z = 10)

!scanvel 0.1

!scanmode 2

!moa 50 100 → vector move with scanvel (vector velocity is now 0.1mm/s)

!mor z 1.5 → single axis move executed with vel z (this example 10mm/s)

!scanmode 0 Disable scanmode (=default, normal operation)

!scanmode 3 No scanvel, vector moves now let each axis travel at its own vel x,y,z,a setting towards the target
e.g. !mor 10 10 5 starts each axis at its own vel setting

10.4.2 scanvel (Vector Velocity for Scanmode and Dissection)

Syntax: !scanvel or ?scanvel

Parameter: 0.000001 to 1000 [mm/s]

Description: This instruction sets or reads the '**scanmode**' vector velocity in millimeters per second. It is also used for the snapshot dissection mode (**sns** 3) continuous path velocity. There is only one parameter and the unit is always mm/s.

Remarks: The **secvel** velocity restrictions will be applied to the individual axes as long as they are not calibrated and range measured (when **cal** + **rm** are not executed).

Response: Currently selected scanmode velocity in [mm/s]

Examples:

!scanvel 0.1 Set scanmode vector velocity to 0.1 mm/s

?scanvel Read scanmode velocity (returns e.g. 0.100000)

10.5. ModuloMode

Modulo Mode can be used for turntable- and for endless rotating applications.

10.5.1 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 sets the specified axes from linear into a turntable mode, where the position remains within one revolution, e.g. `[0...360[°` when `dim = 3` or `[0...1[` when the '`dim`' (displayed unit) is set to 4.

The default linear mode and 4 different turntable modes are available:

- `0` : Modulo Mode off (default mode, e.g. for linear axes)
Required for most applicaions.
- `1` : Travel shortest distance to target position
(automatically decides to travel forward or backward)
- `2` : Only travel in positive direction (forward)
If the target position is below the current position, the axis travels once around to reach it (e.g. 359°->358°)
- `3` : Only travel in negative direction (backward)
If the target position is above the current position, the axis travels once around to reach it (e.g. 358°->359°)
- `4` : Do not travel over Zero
e.g. for swiveling axes <360° with limited operation range or as cable tear-off protection (the axis remains within one revolution or the limits, travels forward and backward)

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.

It is possible to switch between the modes (especially 1,2,3) during operation (typ. before a move instruction), to achieve the momentarily required behavior.

Remarks: Endless rotating applications: Modulo Mode 1 in conjunction with the **!speed** instruction can be used to achieve an endless rotating mode (pumps, fans, etc.). (Because in linear mode the rotation will stop at the maximum travel range, e.g. ±2600mm).

Response: Currently selected modulo mode

Examples:

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


10.6. External Display Mode

TANGO controllers with dual interface (USB and RS232) can be configured to support the external position display "PROFILER SCD CL". This display unit connects to the TANGO through RS232.

The internal TANGO position is displayed, independent of closed or open loop operation. If the TANGO uses position correction, the corrected position is shown.

With display enabled (by "**!configdisplay 1**"), the TANGO sends position and status data over its RS232 interface. The RS232 then can't be used for communication with the PC anymore; the PC must be connected via USB.

10.6.1 configdisplay (Configure External Display)

Syntax: `!configdisplay` or `?configdisplay`

Parameter: 0 or 1

Description: This instruction enables or disables the external position display "PROFILER SCD CL".

0: disable external display (normal RS232 operation)

1: enable external display (RS232 occupied by display)

Remarks: Only available if TANGO has two interfaces (USB and RS232, or PCI-E and RS232), e.g. TANGO DT-E/PCI-E and TANGO 3 mini. TANGO firmware 1.67 or higher is required.

Response: Currently selected display mode

Examples:

```
!configdisplay 0
```

```
?configdisplay
```

11. Controller States and Error Messages

11.1. statusaxis (Read State of Axis)

Syntax: ?statusaxis or statusaxis, ?sa or sa

Parameter: x, y, z, a or none

Description: Statusaxis (sa) returns the state of each axis. Similar to the '**autostatus 1**' response for move instructions, but with an additional '-' after the dot (.- instead of .). It can be used for polling move states when in '**autostatus 0**' mode, where no automatic response is generated.

Every response except of 'M' means the axis has stopped for some reason and may be ready for a new move instruction.

It is recommended to check the returned ASCII character for != 'M' (not equal to 'M', 0x4D hex).

Remarks: Statusaxis (sa) reports similar to the **autostatus** reply. It does not report manual moves from the HDI (e.g. joystick), Only move, go, cal/rm or speed instructions cause 'M' replies. This is, because manual HDI moves will be interrupted by those Instructions and no waiting for HDI (joystick) is required.

Hint: The 'sta' instruction contains much more details about the axis states. It is available from Firmware 1.73 and higher.

Response: 6 ASCII characters: [STATUS X][STATUS Y][STATUS Z][STATUS A].- or with specified axis x,y,z,a: 1 ASCII character

J => Axis is ready and may also be controlled manually (joystick enabled)

@ => Axis is not moving and ready, joystick (HDI) is disabled

M => Axis is moving

- => Axis is not available in hardware

For the 'statusaxis'/'sa' instructions, the following states have lower priority and might be superseded when the joystick is globally enabled (?joy is not 0) by the above listed 'J'. Then they do only appear in an **autostatus** reply of a move:

S => Limit switches are actuated and prevent further automatic move

A => ok response after cal instruction

D => ok response after rm instruction

E => error response, move aborted or not executed (e.g. **cal** or **rm** error, or stop input active)

T => Timeout occurred (refer to '**caltimeout**' instruction)

Example: For a 3-axis controller

```
?statusaxis => JJJ-.- (same as "sa"-responses shown below)
sa          => JJJ-.- (all axes idle, no running move, joystick on )
sa          => @@@-.- (all axes idle, no running move, joystick off)
sa          => JMJ-.- (Y axis is traveling , joystick on )
sa          => MMM-.- (all 3 axes are traveling)
sa y       => M      (Y axis is traveling)
```

Example for idle 4-axis controller, joystick enabled: sa => JJJJ.-

Example for idle 3-axis controller, joystick enabled: sa => JJJ-.-

Example for idle 2-axis controller, joystick enabled: sa => JJ--.-

Example for idle 1-axis controller, joystick enabled: sa => J---.-

11.2. sta (Read Detailed State of Axis)

Syntax: ?sta or sta

Parameter: x, y, z, a or none

Description: Sta returns a detailed state of the axes as 32bit HEX values.

Remarks: Available only from TANGO Firmware versions 1.73 and higher.

Response: 32bit HEX numbers 00000000 to FFFFFFFF,
amount (1 to 4) depending on axis count or requested axis.

```
00000001 !axis is set to 0 or 1 (motor current is on)
00000002 !axis is set to 1 (enabled)
00000004 motor power amplifier is on
00000008 motor power amplifier error

00000010 corresponds to statusaxis 'M' state of the axis
00000020 the axis travels due to HDI deflection (e.g.joystick)
00000040 cal is running
00000080 rm is running

00000100 cal already executed ('A' in statuslimit)
00000200 rm already executed ('D' in statuslimit)
00000400 lower limit switch E0 actuated (1 in readsw)
00000800 upper limit switch EE actuated (1 in readsw)

00001000 axis move waits for snapshot signal
00002000 calrequired prevents axis move, no cal/rm yet
00004000 1D position correction active
00008000 2D position correction active (@ Z reply: 2D+z)

00010000 encoder is active (?enc)
00020000 encoder was activated, even if currently disabled
00040000 reserved
00080000 encoder error (encerr)

00100000 closed loop is on
00200000 closed loop is active, regulating
00400000 closed loop is in target window (set by twi)
00800000 closed loop is in lock-in range (set by ctrs)

01000000 stop signal is active
02000000 HDI is enabled for this axis (joy+joydir)
04000000 ... 80000000 reserved
```

```
Example: sta x => 02030307 axis is not traveling, no closed loop, no errors,
                | | | |
                | | | +- axis is on (!axis 1, !pal)
                | | +--- cal and rm are executed
                | +----- encoder is active (and was/is active)
                +----- HDI is enabled (joy+joydir)
```

```
sta => 02030307 02030307 00000007 (reply of a 3 axis TANGO)
sta z => 00000007
```

11.3. calst (Read Calibration State of Axis)

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

Description: Calst returns the calibration state of one or all axes. Similar to the '**statuslimit**' instruction, it contains the information if the cal or rm routines were executed or not. But In a more easy readable way, as a sum of bits.

0 : Neither Cal nor Rm executed yet
1 : Cal executed, Rm not
2 : Rm executed, Cal not
3 : Cal and Rm executed (=1+2)

It is recommended to check the return value bitwise (by &1,&2) to allow addition of further states for future extensions of this instruction.

Remarks: Available form TANGO Firmware versions 1.70 and higher.

Response: Decimal numbers with containing the bit representation of the cal and rm executed states. One value per axis.

Example: calst => 3 3 1 (CAL+RM executed for X and Y, Z only CAL)
 calst z => 1 (CAL executed for Z, RM not)

11.4. corrst (Read Position Correction State)

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

Description: Read the state of position correction. Bits 0 and 1 are for 1D correction (usual per axis correction) While bit 2 is a general information that any correction is currently active. Then bit 5 can be checked if it is 2D or 1D correction. Bits 3, 4 and 5 are for 2D correction available from Firmware 1.73. When 1D correction data is available and requested (Bits 0+1), a cal instruction activates it (Bit 2).

Bit 0 (1) = 1D correction data available
Bit 1 (2) = correction activation requested (by '!corr 1')
Bit 2 (4) = correction is active (1D and/or 2D, 2D+z)
Bit 3 (8) = 2D correction data available (X,Y)
Bit 4 (16)= 2D correction data available (Z) replied by Z axis
Bit 5 (32)= 2D correction is active XY, 2D+z reports on Z axis

Response: Bit coded integer number 0...63 per axis

Example: corrst => 7 3 0 (X: 1D active, Y: 1D available, Z: no data)
 corrst x => 7 (X: Bit 0,1,2 are 1 = 1D correction active)

11.5. status (Read the Controller Error State)

Syntax: ?status or status
Parameter: none

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

Response: OK... or ERR with error number

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

11.6. err (Read Error Number)

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

Description: The 'err' instruction returns the controller error state or 0, if no error occurred. The error state reflects the error of the previously sent instruction or an internal error. The 'err' or '?err' read instruction does not change the error state (the error state remains). The error state can be cleared to zero by sending '!err', except it is a permanent error as e.g. error 29.

Response: Error number as decimal value
 Refer to Chapter 5: **Error Numbers**

Remarks: Errors can be caused by instructions, e.g. if there is a typo or if the parameters are not accepted etc. Those error numbers are typically in the range of 1 to 10. Each instruction overwrites the error state, so 'err' only corresponds to the previously executed instruction. If the previously executed instruction returned no error, the 'err' response is either 0 or if an internal error state is active, the internal error is returned. Those error numbers are typically 20 and above. Also refer to the '**help**' instruction.

Example: err => 0
 ?err => 0 (same as err)

 ?err => 5
 !err (clears the error state to 0 if not a permanent error)
 ?err => 0

11.7. help (Read Error Number with Description String)

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

Description: In addition to '**err**', help returns an describing text about the momentary error state or about a specified error number. It contains the error state with appended error description. By reading help, the error state is not changed or cleared to zero. Please also refer to the '**err**' instruction.

Called without a parameter:

It returns the controller's error state with description

Called with a parameter (error number):

It returns this error with its corresponding description

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

Example: help => ERROR 0,no error
 help => ERROR 5,number outside range
 help 29 => ERROR 29,servo amplifier off

11.8. cmderr (Command Error List)

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

Availability: Only with TANGO Desktop HE.

Description: Provides a multi-line list of up to 15 recent Instructions that produced an error. Each line stands for an error that was returned by the instruction. The last line terminates with "end."

It can be used to check, if an instruction or parameter was not accepted due to availability, syntax, parameters or !/?.

Sending '!cmderr' clears the error list, discards all entries.

The error lines start with the oldest entry and each error line consists of 4 informations:

1. Time past since the error occurred [hh:mm:ss:ms]
2. Error number caused by the instruction [like ?err]
3. The instruction with '!'/ '?' or in "" if not supported/typo
4. Axis x/y/z/a, if available

The parameters sent are not contained in the list.

Response: Multi-line ASCII-Text, last line is "end."

Example: cmderr => 00:01:30.947 err: 4, cmd unknown: "posx"
 00:01:22.619 err: 6, cmd: ?pos x
 00:01:15.455 err: 6, cmd: ?pos
 00:01:08.898 err: 5, cmd: !vel
 end.

!cmderr (Clear the cmderr list)

cmderr => end. (No command error)

11.9. service (Print Service Information to Terminal)

Syntax: ?service or service
Parameter: none

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

Response: Many lines of text including e.g. serial number, parameters, states etc. Each line terminated by a [CR]. From TANGO firmware 1.60C/1.61 (September 02, 2015) and later, the last line sends the string "END_SERVICE_PRINT." indicating the end.

Example: service

11.10. pci (Is PCI Bus)

Syntax: ?pci or pci
Parameter: none

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

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

Response: 0 or 1

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

11.11. isvel (Read Actual Velocities)

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

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

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

Response: Actual axis velocity in [mm/s]
with default 3 or specified 0 - 16 fractional digits

Example:
?isvel Read actual velocity of all axes
?isvel y Read actual velocity of the Y axis (e.g returns 10.000)
?isvel 4 Read actual velocity of all axes with 4 fractional digits
?isvel y 6 Read actual velocity of the Y axis with 6 decimal places
isvel Same as ?isvel



11.12. iscur (Read Actual Motor Current)

Syntax: ?iscur or iscur

Parameter: x, y, z, a or none

Description: Read the momentarily applied motor current, which includes current reduction or switched off states (by **axis -1**).

Remarks: It is not a measured value.

Response: Applied motor current in [A] with 2 fractional digits

Example:

?iscur Read applied motor current of all axes

?iscur y Read applied motor current of the Y axis (e.g returns 1.00)

iscur Same as ?iscur

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 10

Description: The dim instruction sets the unit (here "dimension") of the input and output parameters related to length, e.g. position or move instructions. Dim 9 and dim 10 also sets the velocity parameters to mm/s (e.g. '**vel**', '**speed**'), which else are in motor revolutions per second. Dim 9 and 10 are improvements of dim 2 and 1, which then behave as dim 2 or 1 were at a pitch and gear of 1. As most axes no longer have a 1mm lead screw, using dim 9 or 10 simplifies setting vel and speed to the correct values. Dim 10 is available from Firmware 1.73.

```
0   = Micro steps
1   = µm
2   = mm      (the default: velocities in motor revolutions/s)
3   = 360°
4   = revolutions
5   = cm
6   = m
7   = inch
8   = mil     (1/1000 inch)
9   = mm      (difference to mode 2: all velocity units in mm/s)
10  = mm      (difference to mode 1: all velocity units in mm/s)
```

Remarks: For dim modes 3 (=360°) and 4 (=revolutions) it is recommended to use a **lead screw pitch** of 1mm.

In dim mode 0, the amount of microsteps per revolution can be specified by the '**usteps**' instruction. This provides compatibility to existing software (which e.g. might require 40000 or 54000 steps/rev) as well as giving flexibility in defining own requirements (e.g. 360 steps/rev). As the internal resolution of the TANGO is not affected, the full resolution can still be accessed by using fractional ustep values (e.g. "**!moa** 39000.22" or "**!mor** 178.63"). While reading back the position will not contain fractional digits.

Response: Current dim settings

Examples:

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

12.2. pitch (Spindle Pitch)

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

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

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

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

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

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

12.3. gear (Gear Ratio)

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

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

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

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

Response: currently used gear ratio

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

12.4. motorsteps (Motor Steps Per Revolution)

Syntax: !motorsteps or ?motorsteps
Parameter: x, y, z, a or none
4 to 65532 as multiples of 4

Description: This instruction sets the steps per revolution of the motor, which can be found in the datasheet. Common motors have 200 steps per revolution (1.8° full step). This is the TANGO default value. Other motors may have e.g. 400, 500 or 24 steps per revolution. It is essential for operation to have this parameter set according to the datasheet. The motor steps parameter must be a multiple of 4 in the range of 4 to 65532.

Response: Selected motorsteps of the stepper motor(s)

Examples:

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

12.5. accel (Acceleration)

Syntax: !accel or ?accel
Parameter: x, y, z, a or none
0.0001 to 20 [m/s²]

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

Optional read-resolution: Accel can be read with a higher resolution than the default 2 decimal places. To read accel with more (or less) decimal places, the number [0 to 10] can be specified with "?accel [0...10]". See examples below.

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

Response: Currently used acceleration in m/s²
with default 2 or specified 0 - 10 fractional digits

Examples:

```
!accel 0.5      set acceleration X=0.5[m/s2], other axes are not affected
!accel 1 0.123456 set acceleration X=1.0[m/s2] and Y=0.123456[m/s2]
!accel z 0.2    set acceleration Z=0.2[m/s2], other axes are not affected
?accel         => 1.00 0.12 0.20          (read acceleration of all axes)
?accel y      => 0.12                    (read Z axis acceleration)
?accel 6      => 1.000000 0.123456 0.200000 (read accel with 6 dec. places)
?accel y 4    => 0.1235                  (read Y accel with 4 dec. places)
```

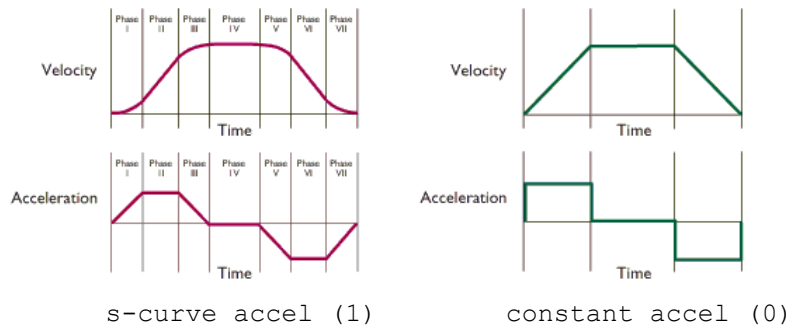
12.6. accelfunc (Acceleration Ramp Function)

Syntax: !accelfunc or ?accelfunc

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

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

0 = constant acceleration and deceleration
1 = s-curve acceleration and deceleration
2 = reserved, currently behaves as 1 (s-curve)



Remarks: For both, constant (0) and s-curve (1) acceleration, the acceleration time at the same accel is identical.

s-curve might be used to introduce less shake in the system, but for smaller step sizes of about <1mm the linear acceleration might be better, as the s-curve has a twice as high acceleration in the ramp center.

The acceleration ramp for **go**, **speed**, **a** and manual control via HDI always remains constant accelerated (0). The deceleration ramp is same as acceleration ramp.

Response: Currently used acceleration type

Examples:

```
!accelfunc 1      set accel type X to s-curve, other axes are not affected
!accelfunc x 1    set accel type X to s-curve, other axes are not affected
!accelfunc 1 1 0  set accel type in X and Y to s-curve, Z to constant
?accelfunc        read acceleration ramp type of all axes
?accelfunc z      read acceleration ramp type 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/s²

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

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

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

Response: Deceleration for stop conditions in [m/s²]
with default 2 or specified 0 - 10 fractional digits

Examples:

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

12.8. vel (Velocity)

Syntax:	!vel or ?vel
Parameter:	x, y, z, a or none 0.000001 to 200 [rev/s] (or up to 3000 [mm/s] at dim 9 and 10)
Description:	Velocity for automatic moves, cal**, rm** and HDI** (Remarks) The velocity unit is motor revolutions per second, except of dim 9 and 10, which provide 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), vel is also used for <ul style="list-style-type: none">• the HDI (joystick) velocity• the cal and rm instructions extmode =1 provides separate parameters (joyvel , calvel , rmvel). Vel can be used to change the travel velocity during a go instruction. In such case, setting vel to zero will also end a running go instruction (e.g. !vel z 0, !vel 0 0 0, etc.) The velfac instruction can be used in addition to vel, but is not necessary or recommended.
Response:	Currently selected velocity in rev/s, or mm/s at dim 9+10 with default 3 or specified 0 - 16 fractional digits
Examples:	<pre>!vel 1.5 15 set velocity X=1.5[revolution/s] and Y=15[revolution/s] !vel y 0.123456 set velocity Z=0.123456 [revolution/s] ?vel read velocity of all axes ?vel x read velocity of X axis only ?vel 6 => 1.500000 0.123456 10.000000 ?vel y 4 => 0.1235</pre>

12.9. velfac (Velocity Factor)

Syntax: !velfac or ?velfac

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

Description: This instruction sets or reads the velocity factor, which is used for all consecutive automatic moves. It is internally multiplied to the velocity (**vel**) and affects the positioning instructions moa, mor, moc, moe, m, cal, rm. It does not affect instructions like go or speed.

Remarks: Velfac is just for backward compatibility and not required anymore, as the **vel** resolution is high enough to achieve the full spectrum of velocities.

Response: Velocity factor (0.01 to 1.00) with 2 decimal places

Examples:

```
?velfac          read velocity factor of all axes, e.g. 1.00 1.00 1.00
?velfac z       read velocity factor of Z axis only, e.g. 1.00
!velfac x 0.1   set velocity factor of X axis to 1/10 of specified velocity
!velfac 1 1 1   set velocity factor of X,Y,Z to specified velocity (default)
```


12.10. secvel (Secure Velocity)

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

Description: The secure speed limitation is intended to prevent mechanical damage when the controller does not know the mechanical limits of the axis, e.g. before cal and rm is executed. This is required because the mechanical space behind the hardware limit switches often is not sufficient to stop the axis under all velocities once they get actuated.

The travel speed is limited to a maximum of secvel as long as the axis is not calibrated and range measured ('cal'+**rm**', refer to Remarks section below). The unit is always mm/s and does not depend on the 'dim' setting.

Setting secvel to higher values may be used at own risk.

The limitation affects all move and speed instructions as well as manual control e.g. by joystick.

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

Remarks: Axes without any limit switches do not apply secvel at all. (E0 and EE switches must be disabled by '**swact**' then.)

 Axes with only one limit switch (E0/cal) release secvel after cal was executed. (EE/rm switch must be disabled by '**swact**'.)

 Axes with both limit switches (or when both limit switches are enabled by '**swact**') require both '**cal**' and '**rm**' to release the secvel.

 A time saving alternate to the **rm** move would be using the "virtual rm" (**vr**m) or having a factory defined axis length stored in the axis.

Response: Currently used secure velocity in [mm/s]
 with default 2 or specified 0 - 16 fractional digits

Examples:

```
!secvel 100 100 100 => Set maximum possible velocity for X Y Z
!secvel y 14.5       => Set maximum possible velocity for Y to 14.5 mm/s
?secvel             => 10.00 10.00 10.00 (X Y Z response of a 3 axis controller)
?secvel x           => 10.00

!secvel y 0.0001
?secvel y           => 0.00       (the default readout is not sufficient here)
?secvel y 5         => 0.00010 (read secvel with additional decimal places)

!secvel 0.123456
?secvel x           => 0.12
?secvel x 5         => 0.12346
?secvel x 6         => 0.123456
```

12.11. cur (Motor Current)

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

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

If the specified motor current exceeds the maximum current, it is automatically limited to this maximum value and the error state **E5** is set.

Remarks: Please check the motor datasheet first in order not to damage the motor by overcurrent/overtemperature. If setting the motor current too low for the application the axis can lose steps. In open loop systems (no encoder feedback) this can cause loss of position; it mostly happens at very low velocities. It can lead to mechanical damage, because the position information is incorrect and with it the reference for the axis position limits. At least for open loop systems it is required to ensure the axis travels correctly under all required velocities and load situations.

If current reduction is active, the momentarily applied motor currents can be checked any time via the **iscur** instruction.

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

Examples:
!cur 1.1 set X motor current to 1.1[A]
!cur 0.7 2.4 set motor current for X=0.7[A] and Y=2.4[A]
!cur z 0.3 set Z motor current to 0.3[A]
?cur read motor current of all axes
?cur x read motor current of X axis only



12.12. reduction (Motor Current Reduction Factor)

Syntax: !reduction or ?reduction

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

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

Recommendations:

Best performance is achieved by keeping reduction at 100%.

0.7 = 70% as a good compromise, heat about 1/2 reduced

0.5 = 50% greatly reduced heat to 1/4

0.3 = 30% lowest recommended**

0 = 0% no torque, in open loop position might be lost

** below 30% (from reduction 0.29), the closed loop stops working (prevented for not stalling the motor due to weak force).
Also, when using a **motor brake**, the brake will be activated each time the reduction falls below 30%.

Remarks: The current reduction can be delayed by the '**curdelay**' Instruction, so it won't apply always and immediately, But only when the axes are standing still for a longer time. Also, without adding a delay, reduction might have an impact on vector throughput, as it takes some extra time to increase the current ahead of each move or joystick deflection. The decrease and increase rate is 1% (0.01) per 160µs. When reduction is active and a move instruction or joystick deflection is executed, the axis current is first ramped up to 100%. This causes a response delay of e.g. 30%-->100% = 70*160µs = 11.2ms.

While reducing, the axis might also show slight wobble, depending on mechanical load.

In systems with encoders: Reducing the current to less than 0.3 (30%) will disable the permanent **closed loop** while reduction is active (axis stopped and **curdelay** expired).

If current reduction is active, the momentarily applied motor currents can be checked any time via the **iscur** instruction.

As reduction reduces torque, it is not recommended for Z-axes.

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

Examples:

!reduction .3 .7 Set idle current reduction X=30% and Y=70% (0.3*cur, 0.7*cur)

!reduction 1 1 1 Disable reduction for axes X,Y,Z (100%)

!reduction z 0.5 Set Z idle current reduction factor to 50% (0.5*cur)

?reduction Read idle current reduction factor of all axes

?reduction x Read idle current reduction factor of X axis only (e.g. 1.00)

12.13. curdelay (Delay for Current Reduction)

Syntax: !curdelay or ?curdelay

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

Description: At the end of each move, the axis enters idle state. If the motor current '**reduction**' is set to a value less than 1, this reduction will take effect after the curdelay time.

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

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

Examples:

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

12.14. ecomove (EcoMove Current Level)

Syntax: !ecomove or ?ecomove

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

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

0 = Full motor current (default)

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

Response: EcoMove level

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

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

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

Description: This instruction enables, disables and switches off axes. A disabled axis still powers the motor with its current, while a switched off axis loses its torque.

1 = axis enabled (default, move possible)
0 = axis disabled (amplifier and motor current remain on)
-1 = axis power stage off, no torque

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: Travel direction of the axes.

0 = Normal axis direction
1 = Reversed axis direction

Remarks: Changing the axis direction should only be used once as a general setup of the hardware direction, and not be used during normal operation. It's recommended to **save** and restart.

The hardware limit switches CAL and RM will automatically be reassigned when switching the axis direction as well as the settings of swact, swpol, swtyp and **readsw**. Only **swin** will not adapt to the new E0/EE assignments, as it is hard-coded.

Closed loop will be deactivated when changing the direction and must be re-enabled by cal or reset/power-on.

If the axis is position corrected by factory (mapped), the axis direction is an essential part of the mapping process and must not be changed by the customer. If the direction of a position corrected axis is changed by the customer, the axis will no longer meet their specifications (position accuracy).

Response: Axis direction

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

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 must 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 vibrations and position error.

0 = No correction (default)

Response: Currently used motortable(s)

Examples:

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

12.18. usteps (Microstep Resolution)

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

Description: This instruction is used in conjunction with the unit '**dim 0**'. As '**!dim 0**' switches the axis unit to microsteps, the "usteps" instruction can be used to select the appropriate number of microsteps that make one revolution of the motor.

Setting usteps to 360 will result in e.g. "**!mor 360**" causing one revolution of the X-axis motor.

One value applies to all axes that have '**dim 0**' selected.

Remarks: The usteps instruction does not change the resolution of the motor. It only allows to select what number will cause one revolution when the axis is set to '**dim 0**' microsteps mode.

The '**dim 0**' is intended for backward-compatibility to existing software packages that might require positioning in microsteps instead of metric or imperial units.

Older software written for e.g. 40000 or 51200 microsteps per revolution can be used to control the TANGO controller.

As the usteps instruction does not change the physical resolution of the motor (typ. 819200 for a 200 steps motor), positioning instructions such as "moa", "mor", "go" can be executed with fractional values also, e.g. "**!mor 12007.3**". But the '**?pos**' instruction will always only return integer When in dim 0.

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

Examples:

```
!usteps 51200          51200 is the count of microsteps for one revolution in dim 0.
?usteps               Read the microstep resolution
```

12.19. resolution (Position Number Format)

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

Description: This instruction sets the resolution of position returning instructions for the metric **dim 1, 2, 9 and 10**. It affects the amount of returned fractional digits, as listed below. One value applies to all axes, the default is 4 (100 nm).

Value	Resolution dim 2+9	Resolution dim 1+10
0	= 1mm	0.1 μ m
1	= 0.1mm	0.1 μ m
2	= 0.01mm	0.1 μ m
3	= 0.001mm	0.1 μ m
4 (default)	= 0.0001mm	0.1 μ m
5	= 0.00001mm	0.01 μ m
6	= 0.000001mm	0.001 μ m

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

Response: Responded fractional digits of 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+9 and 0.00 in dim 1+10.
?resolution        Read the fractional digits resolution for position values
```

12.20. backlash (Mechanical Backlash Compensation)

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

Description: Compensates mechanical backlash of the individual axes. Unit is always micrometer [μ m], independent from dim.

0 = Backlash compensation off

Remarks: Backlash compensation is not applied in **closed loop mode**. Backlash compensation does not affect the axis performance. Backlash compensation is also applied in HDI mode, e.g. when using the joystick. Due to compensation the manual control is greatly improved when using high magnifications.

Backlash Info: Mechanical backlash becomes visible when traveling to the same position from both directions, forward and backward. The backlash value is half the amount of this deviation.

Response: Axis backlash in micrometer [μ m]

Examples:

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

12.21. lock (Select Parameters to Lock)

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

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

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

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

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 '**lock**' lockbits or lockaxis are
zero, nothing will be locked.

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

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

12.23. lockstate (Read all internal Lock States)

Syntax: ?lockstate

Parameter: x, y, z, a or none

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

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

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 ... 4 or 5

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

0	= AUX I/O not used, Status-LED only (default)
1	= AUX I/O Pin 5 (TAKT_OUT) <i>not always supported</i>
2	= AUX I/O Pin 6 (VR_OUT)
3	= AUX I/O Pin 7 (SHÜTTER_OUT)
4	= AUX I/O Pin 8 (TRIGGER_OUT)
5	= PLED Connector (Onboard) <i>TANGO Desktop HE only</i>

Remarks: In order to turn off the original TANGO Status LED(s), the instruction '**noled**' may be used.

Response: Selected status output mode 0 to 5

Example: !stout 0 => Only use TANGO Status LED (default)
?stout => Read status output mode

12.25. noled (Force Status LED Off)

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

Description: Permanently force off the TANGO Status LED.
Forcing the LED off may be required in low light applications where no external light source is wanted.

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

Remarks: Forcing the TANGO Status LED(s) off only affects the TANGO Status LED(s). It does not affect the optional status output signal that can be selected by '**stout**'.

Response: Status LED mode

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

12.26. updelay (Power Up Delay)

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

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

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

Applications:

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

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

Positive values: The controller waits for the specified time.

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

Response: Power up delay time in [ms]

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

13. Limit Switch Instructions (Hardware and Software)

13.1. lim (Software Limits)

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

Description: This instruction sets or reads the software position limits. Software limits can be used to narrow the positioning range of axes, limiting all move instructions and manual control. The software limits must be within the maximum positioning range as set by 'cal', 'rm' (or 'vrm') or, when none of the calibration routines are used, the maximum position range of the axes (e.g. +-2.6 meters). Setting the software limits outside of these limitations will not extend the positioning range and the axes still will stop at those maximum limits. Setting the software limits do not remove the **sevel** velocity limitations. Therefore 'cal', 'rm' or 'vrm' are required. The upper and lower software limits must be sent together in a single "!lim" instruction. The unit depends on 'dim' setting.

Remarks: It is recommended to read the limits axis by axis ('?lim x' etc.), because the '?lim' instruction for all axes has a formatting error of sending an additional [CR] after the X-axis values. If **Extended Mode** is enabled (extmode = 1), the '?lim' instruction returns the limits as a correctly formatted string -> see example below.

By default (**nosetlimit** = 0), the soft limits are overwritten by the 'cal', 'rm' and 'vrm' instructions. Which (re-)set the soft limits to the mechanical limit switch positions.
-> For setting own software limits, use the '!lim' instruction after 'cal' or 'cal'+ 'rm' (or 'cal'+ 'vrm') was executed.

If a software limit is set below the current axis position, it is only possible to travel towards and below this new limit. Any further movement away from it is not possible.

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

Examples:
!lim -2000 2000 -2000 2000 0 50 set the software limits for X, Y and Z axes
!lim 0 200 20 80 set the software limits for X and Y only
!lim z 0 315 set the software limits for Z to 0 ... 315
!lim z 45.37 set the lower software limit of Z to 45.37
?lim y read upper and lower software limits of Y

?lim read all upper and lower software limits
 only recommended in extmode=1, as shown:

?lim response example for 3 axes, without and with **extmode** enabled:
--> extmode=0: -2600 2600, [CR]-2600 2600, -800 800[CR] (mimics faulty behavior)
--> extmode=1: -2600 2600 -2600 2600 -800 800[CR]

13.2. clim (Circular Software Limit)

Syntax: !clim or ?clim
Parameter: center position x,y and radius
 or only radius
 (units depend on '**dim**')

Description: This instruction provides circular movement range limitations. The limit can be specified either with XY center position or just with the radius. Then the current XY axis positions will be used as center. The units depend on '**dim**', x and y must have same dim.

*!clim 0 disables circular limits
?clim 1 queries the state (active=1, inactive=0)*

*!clim [centerPosX] [CenterPosY] [Radius] or
!clim [Radius] set the circular limit.*

Response: Circular limits [X] [Y] [radius] or active state [0,1]

Examples:

!clim 50 70 10 Set clircular limit of radius 10 at center position X=50,Y=70
!clim 9.5 Set clircular limit of radius 9.5 around the current position
!clim 0 Disable circular limits
?clim Read circular limit positions and radius (cenx ceny radius)
?clim 1 Query if circular limits are enabled (1) or disabled (0)

13.3. 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.

0 = disabled
1 = enabled (default from power on)

Attention: Setting limctr to 0 can cause mechanical damage to the axis!
If limit controls are disabled, the controller ignores any limits set by **cal**, **rm** or **lim**.
Actuating the limit switches is still recognized (as long as they are not deactivated by the **swact** instruction).

Remarks: Some TANGO controllers or Firmware versions have limit control enabled by default from power on and do not allow storing the disabled state permanently by the save instruction.

Response: Limit control state

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

13.4. nosetlimit (Do not set Limits by Cal/Rm)

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

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

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

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

13.5. limmode (Limit Monitoring Mode)

Syntax: !limmode or ?limmode

Parameter: 0, 1 or 2

Availability: Only with TANGO Desktop HE controllers.

Description: Behavior when exceeding a position limit (?lim). In case an automatic move moa,mor,m,moc,mol,moe target position exceeds a position limit, it is truncated to the limit position without notice. Only '?pos' can be used to check if the desired Position was reached. This is the default (0).

Limmode offers two additional ways how to handle a move that exceeds a position limit, 1 and 2:

Mode 0 is the regular behavior of a TANGO controller.

Mode 1 will not execute moves and vector moves, if at least one axis would exceed its position limit. The @@@ reply will contain an 'E 'for the not moved axes (discarded move). Also, error state 32 will be set.

Mode 2 maintains the regular behavior, but an 'L' will be sent in the @@@ response for a limited axis.

One parameter applies to all axes.

Response: 0, 1 or 2

Examples: !limmode 0 Set limit mode to default
?limmode => 0 Read limit mode

!limmode 1

```
!lim x 0 50
!moa 75 10 => EE@@.
?err       => 32
?pos x     => 0.0000
```

!limmode 2

```
!lim x 0 50
!moa 75 10 => L@@@.
?err       => 32
?pos x     => 50.0000
```

!limmode 0

```
!lim x 0 50
!moa 75 10 => @@@@. (or L@@@.)
?pos x     => 50.0000
```

13.6. swtyp (Type of Limit Switch)

Syntax: !swtyp or ?swtyp

Parameter: x, y, z or a (specifying an axis is recommended)
0 or 1, 0 or 1, 0 or 1

Description: Set or read the limit switch type, per axis only.
Assigns a pull-up or pull-down resistor to the switch input.
The sequence is always:

```
!swtyp [SWITCH_E0] [SWITCH_REF**] [SWITCH_EE]
```

0 = PNP: applies a pull-down resistor to the switch input

1 = NPN: applies a pull-up resistor (default)

Remarks: ** The REF switch is not used by the TANGO controller.

It is recommended to set each axis individually by using the axis parameter x, y, z or a.

Using no axis parameter will apply the the values to all axes!

Please note that the E0 and EE switches are reassigned by a change of the **axisdir** instruction.

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 E0, EE switch types of Y axis (returns e.g. 1 0 1)
```

```
?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

```
!swtyp 1 0 1
```

set limit switches to NPN type for all axes at once
(not recommended)

13.7. swpol (Polarity of Limit Switch)

Syntax: !swpol or ?swpol

Parameter: x, y, z or a (specifying an axis is recommended)
0 or 1, 0 or 1, 0 or 1

Description: Set or read the active polarity of the limit switches, per axis only. The sequence is always:

```
!swpol [SWITCH_E0] [SWITCH_REF**] [SWITCH_EE]
```

0 = switch has active low signal

1 = switch has active high signal

Remarks: ** The REF switch is not used by the TANGO controller.

It is recommended to set each axis individually by using the axis parameter x, y, z or a. Using no axis parameter will apply the the values to all axes!

Please note that the E0 and EE switches are reassigned by a change of the **axisdir** instruction.

The **swin** instruction is not affected by the polarity setting, as it returns the TTL logic level and not the actuation state.

Response: Polarity of the limit switches

Examples:

```
!swpol y 1 1 1 set polarity of Y limit switches (E0 REF EE) to active high
```

```
!swpol z 0 0 0 set polarity of Y limit switches (E0 REF EE) to active low
```

```
?swpol x read limit switch polarity of the X axis (returns e.g. 0 0 0)
```

```
!swpol 1 0 1 set polarity of limit switches for all axes at once  
(not recommended)
```


13.8. swact (Enable or Disable Limit Switches)

Syntax: !swact or ?swact

Parameter: x, y, z or a (specifying an axis is recommended)
0 or 1, 0 or 1, 0 or 1

Description: Enable or disable the limit switches, per axis only.
The sequence is always:

```
!swact [SWITCH_E0] [SWITCH_REF**] [SWITCH_EE]
```

0 = switch is disabled (actuation state is ignored)
1 = switch is enabled

Remarks: ** The REF switch is not used by the TANGO controller.

It is recommended to set each axis individually by using the axis parameter x, y, z or a.
Using no axis parameter will apply the the values to all axes!

Please note that the E0 and EE switches are reassigned by a change of the **axisdir** instruction.

The **swin** instruction is not affected and can still be used to read the TTL logic level of the inputs.

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

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

Examples:

```
!swact 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 (e.g. 1 0 1)
```

```
!swact 1 0 1 enable cal and rm limit switches for all axes at once  
(not recommended)
```

13.9. swdir (Swap Assignment of Cal and Rm Switch)

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

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

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

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

Attention: swdir should only be used to compensate different wiring of the stage limit switches. **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
```

13.10. readsw (Read Status of Limit Switches)

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

Description: Readsw returns the actuation state of the limit switches. The switch assignment (E0 and EE) depends on **axisdir** and is reassigned automatically when the axis direction is reversed. Disabled switches (**swact** set to 0) are read as inactive (0).

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

The switch state is only valid when **swtyp**, **swpol** are set correctly and the switch is activated by **swact**.

Sequence of the returned 12 ASCII characters is:

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

E0 = lower limit switch (!cal instruction)
Ref = Reference switch (only in center-ref mode, else 0)
EE = upper limit switch (!rm instruction)

Remarks: From Firmware 1.73 it is possible to read single axis states. Order when reading a single axis is: [E0] [EE]

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

Examples: readsw => 000000001000 (read all actuation states)
readsw x => 0 1 (read actuation states of X-axis)

13.11. swin (Read Limit Switch Input Level)

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

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

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

In opposite to the '**readsw**' instruction, swin reflects the TTL input levels. Even disabled switches are represented with their current TTL input signal level. Swin is not affected by the swpol polarity or axisdir setting (doesn't exchange E0 and EE switches if axisdir=1) or. The [Ref] signals are not used.

Sequence of the 8 ASCII 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 (for individual read)
E0 = lower limit switch (cal) for axisdir=0
EE = upper limit switch (rm) for axisdir=0

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)

13.12. statuslimit (Cal / Rm / Lim Status)

Syntax: ?statuslimit or statuslimit
Parameter: none

Description: Read the status of the soft- and hardware limits concerning the **cal**, **rm** and **lim** instructions. It can be used to check if the axes were already calibrated (cal) or range measured (rm) and if a software limit was set.

The status information is arranged in 4 groups.
The ASCII character string positions are:

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

The characters represent the state with 4 possible characters:
- => cal or rm not performed, 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)

```
STRING: "-----" up to max. "AAAADDDDLLLLLLLL"  
AXIS   : xyzaxyzaxyzaxyza  
LIMIT  : CAL RM  LIM-LIM+
```

Remarks: For simpler cal+rm information, refer to '**calst**'.
For a more detailed state information '**sta**' can be used.

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, the hardware position limits, releasing the '**secvel**' velocity limit and enabling the encoders and position correction.

The cal/rm instructions set the limits close to the limit switch positions. An additional position offset for 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 changed by the **caltimeout** instruction. Please also refer to the optional **extmode** enhancements and **calmode** options for calibration. Checking **statuslimit** or **calst** tells if a **!cal** or **!rm** was performed.

14.1. cal (Perform Calibration to lower Limit Switch)

Syntax: `!cal` or `cal`

Parameter: `x, y, z, a` or none (or a bitmask from Firmware ≥ 1.73)

Description: This instruction moves either the specified or all axes towards lower positions until the limit switch E0 is detected. By default, `cal` defines the reference position of the axis and enables encoders, closed loop and axis correction.

From Firmware 1.73 it is possible to easily calibrate any combination of axes by using axis bits (1=X,2=Y,4=Z,8=A).

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.

If the corresponding E0 limit switch is disabled (by **swact**), `cal` will not be performed.

The CAL velocities depend on the setting of '**extmode**':

Extmode=0 (mostly the default setting):

- travels towards switch with the axis velocity **vel**, (**secvel**)
- travels out of switch with '**calbspeed**'

Extmode=1:

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

The movement stops slightly after traveling out of the switch. If required, this gap can be increased by '**caliboffset**'.

Remarks: The calibration state can be read by '**statuslimit**' or '**calst**'. For higher accuracy, Cal can use an encoder reference mark, encoder period or motor period: Refer to '**encref**' options.

Response: Depends on **autostatus** settings, which per default is set to 1. Similar to **!moa**, **!mor** etc., but instead of an '@' it returns a '**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` calibrate all enabled axes => "AAA-."
`cal y` start calibration of the Y axis => "@A@-."
`cal 5` calibrate the X and Z axis (from Firmware 1.73)
Firmware before 1.73 would require disabling axes:
"!axis 1 0 1" / "!cal" / "!axis 1 1 1" => "A@A-."

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

Syntax: !rm or rm

Parameter: x, y, z, a or none (or a bitmask from Firmware >= 1.73)

Description: This instruction moves either the specified or all axes towards higher positions until the limitswitch EE is detected.

From Firmware 1.73 it is possible to easily range measure any combination of axes by using axis bits (1=X,2=Y,4=Z,8=A).

If the corresponding EE limit switch is disabled (by **swact**), rm will not be performed.

The behavior of RM depends on the setting of '**extmode**':

Extmode=0 (mostly the default setting):

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

Extmode=1:

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

The movement stops slightly after traveling out of the switch. If required, this gap can be increased by '**rmoffset**'.

if '**nosetlimit**' is set to 0 (default), rm also sets the rm-position as the new upper software limit.

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

Response: Depends on **autostatus** settings, which per default is set to 1. Similar to **!moa**, **!mor** etc., but instead of an '@' it returns a 'D' after a successful rangemeasure or 'E' if an error occurred (rm was unsuccessful) 'T' if a timeout occurred (rm was unsuccessful) '-' if the axis is not present (e.g. 4th axis)

Examples: rm range measure all enabled axes => "DDD-."
rm y start range measure of the Y axis => "@D@-."
rm 5 range measure the X and Z axis (from Firmware 1.73)
Firmware before 1.73 would require disabling axes:
"!axis 1 0 1" / "!rm" / "!axis 1 1 1" => "D@D-."
or sending "cal x" and "cal z" at once.

14.3. vrm (Virtual Range Measure)

Syntax: !vrm or vrm

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

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

Same as with the **rm** instruction the

- **secvel** is released
- corresponding **statuslimit** 'D' entry is set
- hardware limit is set (the internal axis end position)

Vrm requires that **!cal** was performed on the axis.

Vrm does not return any **autostatus** reply (like the @@@-), It returns immediately.

Warning: Specifying a wrong axis length can cause mechanical damage.

Remarks: Check **?err** response or **statuslimit** for 'D' entries to make sure the parameters were accepted.

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

Axes with MR encoders (nanoScale) should always perform a true **rm** for high accuracy measurement.

Vrm can also be used before executing an **rm** move in order to travel fast (without **secvel** limitation) near the **rm** limit switch → then **rm** can be executed (caution: velocity might be high now) to get the real maximum hardware travel range and/or to allow the above mentioned encoder autoadjust which is performed during **rm**. [Example #3]

Response: None

Examples:

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

Sequence example1:
cal x
cal y
cal z
?statuslimit => "AAA-----"
vrm 300 300 20
?statuslimit => "AAA-DDD-----"

Sequence example2:
cal x
cal y
!moa 10 10
!pos 0 0 (the lower limit is now at position -10 -10)
vrm 300 300 (the upper limit is set to 290 290)

Sequence example3:
cal x (calibrate the X axis)
!vel 40 (set X velocity to e.g. 40mm/s)
vrm 300 (virtually set X **rm** for e.g. a 300mm axis)
!moa 300 (move to end of X travel range without **secvel**)
!vel 10 (slow down for executing **rm**)
!rm x (perform true hardware **rm** from here =time saving)

14.4. calmode (Closed Loop and Calibration Behavior)

Syntax: !calmode or ?calmode

Parameter: x, y, z, a or none
0, 1, 2, 3, 4 or 5

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

- 0 : **cal** instruction sets the zero position (**default**) and the Closed loop is enabled after cal
- 1 : the zero position is set at power-up and remains, a cal instruction doesn't set or modify the zero position*
Closed loop is enabled from power-up
- 2 : **cal** instruction sets the zero position as mode 0, but the Closed loop is enabled from power-up
This is especially useful or required, if the axis is driven by e.g. a bowden wire or Uhing® drive.
Depending on the encperiod size, the axis will more or less wiggle for activating closed loop (MR: about 1mm).
- 3 : the axis is used to readout a measuring system only.
Requirements are:
 - the encoder must be a TTL or 1Vpp type, no analog MR
 - encmask must be enabled
 - encpos must be enabled (for reading ?pos)
 - encdir must be set to the required counting direction
 - axis amplifier should be switched off (e.g. !axis z -1)
(it might be possible to use the motor independently, but at least a cal will set both, the motor and encoder positions to zero. The pos at encpos=0 belongs to the motor, the pos at encpos=1 to the encoder)
- 4 : performs an automatic **cal** move after power-up.
Caution! The axes will move without notice immediately after reset or after switching on the TANGO controller. It is not possible to communicate with the TANGO while the cal move is running. It is also not possible to stop the axes by the abort instruction.

Mode 4 is useful in standalone applications (without a PC), where the zero position must be set automatically At power up and/or the position correction of the axis.

The cal is executed axis by axis, starting with X.

The TANGO will not respond to any instruction while the cal moves are executed. If no axis is connected, the cal will consume all the caltimeout of the corresponding axis/axes.

The Status LED can be used as an indicator for a running cal instruction: The LED goes shortly dark about every 2.5 seconds.

The cal is executed at the vel or secvel velocity, depending on which is set to the lowest velocity.
In extmode=1, the calvel will be used instead of vel.

The cal will not release secvel, the secvel velocity limit will only be released if an axlen is specified in the ETS. Else a rm instruction would be required afterwards.

5 : restores the position at power-up.

The TANGO will remember the axis positions at which the power was switched off and restore them at power-up. Work can continue at the previous position without having to calibrate the axes.

If cal/rm was executed, the TANGO will restore those too, release the **sevel** limits and apply position correction.

Only open loop is supported, closed loop or encoders will not be restored.

Soft limits (set by !lim) are not restored.

Caution! The position relies on the last known position where the TANGO was switched off. If the axes are displaced while the TANGO is switched off, the position information is invalid and so might be the limit switches. Which in worst case can cause damage.

So please ensure that the axes remain at their positions during power off when using this calmode 5 functionality.

If the TANGO is switched off while the axes are traveling, the restored positions might not be correct.

Remarks: For activating the closed loop, '**encmask**' of the corresponding axes must be set to 1. If encmask is not set or no encoder is present, the calmode only affects the axis zero position behavior.

Calmode 0 is the default setting for most applications, calmode 2 is useful for axes with encoders, to have closed loop active from power on without requiring a cal for it.

* In calmode 1, the position offset between the CAL switch and the axis position (which is set and remains from the power-up position) can be read out by using '**?calzeropos**', after a **cal** move was executed.

Response: Calmode of the axes (0...5)

Examples:

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

14.5. calrequired (Calibration Required)

Syntax: !calrequired or ?calrequired

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

Description: The calibration required mode offers a mechanism to prevent axis travel until the **!cal** instruction is executed. It can be used to e.g. prevent collisions until the axis is in a known position.

0 = The axis can travel even without !cal (default)
1 = The axis is halted until a !cal is executed

Remarks: An internal calrequired state can be applied automatically in certain caldir modes and absolute encoder settings. Then, the calrequired setting is internally overwritten by it. The internal cal required state, set by this instruction or the certain modes, can be read in an '**?sta**' bit.

Response: Calrequired setting of the axes (0 or 1)

Examples:

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

14.6. caltimeout (Calibration Timeout)

Syntax: !caltimeout or ?caltimeout

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

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

Remarks: For long axes (over 350mm) the default timeout of 40...60s becomes insufficient and must be increased, as the typical cal, rm travel velocity is 10mm/s and the cal, rm routines require some additional time around the limit switch.

Response: Calibration+RangeMeasure timeout in seconds

Examples:

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

14.7. caliboffset (Calibration Offset)

Syntax: !caliboffset or ?caliboffset

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

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

Response: Calibration Cal switch position offset

Examples:

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

14.8. rmosffset (Range Measure Position Offset)

Syntax: !rmoffset or ?rmoffset

Parameter: x, y, z, a or none
Position (0.0 ... 100mm, unit 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
```

14.9. caldir (Calibration Direction)

Syntax: !caldir or ?caldir

Parameter: x, y, z, a or none
0, 1, ... 7

Description: Behavior of the calibration '**cal**' function, default=0.
From TANGO Firmware 1.73 and higher, axes with Center Reference are supported (Modes 2-7)**.

0 = Default behavior, cal calibrates to lower limit (E0)
(1 = NOT SUPPORTED! cal calibrates positive to upper limit)

2 = Center Reference, signal active at lower positions
3 = Center Reference, signal active at upper positions
--> cal stops directly at the reference position

4 = Center Reference, signal active at lower positions
5 = Center Reference, signal active at upper positions
--> cal stops at the exact center position

6 = Center Reference, signal active at lower positions
7 = Center Reference, signal active at upper positions
--> cal stops at the lower position of the axis, pos 0

Remarks: ** Center Reference requires additional configurations which must be set by factory. Wrong configuration will lead to damage of the axis.
In Center Reference mode, an E0 signal is expected that masks about the upper or lower half of the travel range.

For Center Reference, the position shift from the axis zero position to the position where the center ref cal function completes must be specified by **posshift** or in the ETS.

Response: Current behavior of the cal function.

Examples:

```
?caldir          read caldir of all axes
?caldir x        read caldir of X axis only
!caldir y 0      set Y caldir to default calibration behavior (lower limit)
!caldir 0 0 1    set Z caldir to positive direction, X and Y to negative
```

14.10. calbspeed (Calibration Speed for Retraction)

Syntax: !calbspeed or ?calbspeed

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

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

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

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

Examples:

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

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

14.11. calrefspeed (Reference Signal Calibration Speed)

Syntax: !calrefspeed or ?calrefspeed

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

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

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

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

Examples:

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

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

14.12. calpos (Calibration Position read from Encoder)

Syntax: !calpos or ?calpos
Parameter: x, y, z, a or none
 none or 0 (0 ... 100mm possible, unit depends on dim)

Availability: Only for axes with encoders.

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

Remarks: In case there is no encoder, or the encoder period is smaller than the deviation range, calposmot possibly can be used.

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
```

14.13. calposmot (Calibration Position read from Motor)

Syntax: !calposmot or ?calposmot
Parameter: x, y, z, a or none
 none or 0 (0 ... 16383 possible)

Availability: From TANGO Firmware 1.73.

Description: Like calpos, calposmot is used to measure calibration position accuracy (repeatability of the origin), but without the need of an encoder or in case of open loop applications.
 During calibration the motor current signal period, where the limit switch E0 was released and the origin (pos=0) was set, is stored. This position value within an motor current sine period can be read with ?calposmot.
 The position may also be set to another value (usually to 0).

Remarks: The motor current signal period range depends on lead screw pitch, gear setting and the motorsteps.
 It can be calculated as $(\text{pitch}/\text{gear}) * (4/\text{motorsteps})$ [mm]
 E.g. a 4mm pitch and 200 steps 1.8° motor have 80µm period.

Response: A position within the range of one motor current signal period
 0 ... 16383

Examples:

```
?calposmot y            read cal position of Y axis motor period (e.g. returns 5193)
?calposmot             read cal position of all axes
!calposmot 0 0 0       set cal positions to zero (X, Y and Z)
!calposmot y 0         set cal position of Y axis to zero
```

14.14. calabspos (Calibration Position from Absolute Encoder)

Syntax: !calabspos or ?calabspos

Parameter: x, y, z, a or none

Availability: TANGO Desktop HE only, for axes with absolute encoder.

Description: Returns the absolute position of the encoder where the cal routine has completed (and usually defines the zero position). If the axis provides a lower limit switch (E0/CAL), the '!cal' instruction can be used to once determine the posshift value for zero alignment (posshift = -calabspos).

Remarks: If no E0/Cal switch is available, the absolute position can be determined by manually shifting the axis to the lowest position (maintain some 1/10mm gap to the mechanical limit) and read the local '?abspos' there.

Response: RAW position of the absolute encoder at the cal position (0). The unit depends on dim.

Examples:

```
cal 3          => AA@@.
calabspos     => 438.1338 35.7331 0 0
calabspos y   => 35.7331
```

14.15. calzeropos (Position at CAL)

Syntax: ?calzeropos

Parameter: x, y, z, a or none

Description: In **calmode 1**, the axis position is not set to zero by the **cal** instruction. In order to determine the "zero" position after **cal**, calzeropos can be read.

Remarks: CalModes other than 1 (0,2) set the position to zero after executing cal. So the returned position is zero in this cases.

Response: Position at which the **cal** instruction completed. The unit depends on '**dim**'.

Examples:

Assuming calmode set to "!calmode 2 1 0 0" and cal already executed on all axes:

```
?calzeropos y   -153.1433
?calzeropos     0.0000 -153.1433 0.0000 0.0000
?calzeropos x   0.0000
```

Assuming all axes traveled +10 mm after the cal instruction:

```
?calzeropos y   -153.1433 (this axis is in calmode 1)
?pos y          -143.1433 => -143.1433 - (-153.1433) = 10.0000
```

```
?calzeropos x   0.0000 (this axis is in calmode 2)
?pos x          10.0000 => 10.0000 - 0.0000 = 10.0000
```

```
?calzeropos z   0.0000 (this axis is in calmode 0)
?pos z          10.0000 => 10.0000 - 0.0000 = 10.0000
```

14.16. calvel (Calibration Velocities for CAL Instruction)

Syntax: !calvel or ?calvel
Parameter: x, y, z, a or none
one or two velocities (depending on dim)

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

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

Velocity unit: [motor rev/s] for dim = 0 ... 8
[mm/s] for dim = 9 and 10

The travel speed towards the limit switch should not be faster than 10mm/s to prevent mechanical damage (axis must be able to stop within a short distance after a limit switch event).
The travel speed out of the limit switch should be low for achieving high position accuracy, e.g. 0.5mm/s

Remarks: calvel can be set per axis only, with x,y,z,a specified

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

Examples:
!calvel x 10 0.5 !cal in X travels towards E0 limit switch at a velocity of 10
and out of the limit switch at a velocity of 0.5
!calvel x 10 set the velocity towards the switch only
?calvel read cal velocities of all axes (xvel1 xvel2 yvel1 yvel2 etc.)
?calvel y read cal velocities of Y axis only (e.g. returns 10.000 0.500)
!calvel 10 0.5 NOT SUPPORTED! The set instruction is per axis only

14.17. rmvel (Range Measure Velocities for RM Instruction)

Syntax: !rmvel or ?rmvel
Parameter: x, y, z, a or none
one or two velocities (depending on dim)

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

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

Velocity unit: [motor rev/s] for dim = 0 ... 8
[mm/s] for dim = 9 and 10

The travel speed towards the limit switch should not be faster than 10mm/s to prevent mechanical damage (axis must be able to stop within a short distance after a limit switch event).
The travel speed out of the limit switch should be low for achieving high position accuracy, e.g. 0.5mm/s

Remarks: rmvel can be set per axis only, with x,y,z,a specified

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

Examples:
!rmvel x 10 0.5 !rm in X travels towards EE limit switch at a velocity of 10
and out of the limit switch at a velocity of 0.5
!rmvel x 10 set the velocity towards the switch only
?rmvel read cal velocities of all axes (xvel1 xvel2 yvel1 yvel2 etc.)
?rmvel y read cal velocities of Y axis only (e.g. returns 10.000 0.500)
!rmvel 10 0.5 NOT SUPPORTED! The set instruction is per axis only

14.18. 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 axis pitch each time when executing a cal instruction. Used for drives with unknown pitch values.

Remarks: Only works if encoders are present.
Not recommended for drives with known pitch, e.g. lead screw.
TANGO Desktop HE does not save the autopatch setting anymore.

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 adjusting sequence for X: !autopitch x 1
!cal x
[wait for reply]
!autopitch x 0
?pitch x
!save

14.19. refdir (Direction for Searching the Reference Switch)

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

Description: DUMMY INSTRUCTION. The additional REF switch is not supported by TANGO controllers. Only CAL and RM switches are used.

Specifies or reads the direction in which the !ref instruction searches the [REF] reference switch. The !ref instruction is not available.

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

14.20. posshift (Position Shift for Zero Alignment)

Syntax: !posshift or ?posshift
Parameter: x, y, z, a or none
 position values (depending on dim within \pm maxpos)

Availability: From TANGO Firmware 1.73 and higher.

Description: Position shift to align the zero position of the axis. Used for Center Reference (**caldir**) Modes and for axes with absolute encoders, where the position must be shifted to become zero at the axes zero position.

Remarks: Posshift must or should be stored in the ETS by factory:

Only for Center Reference Modes, this internal posshift value can be used (but is recommended to write it to the ETS). Then it defines the distance from 0 to the center reference, e.g. 37.9833 (it is always a positive value).

In case of Absolute Encoders, it is mandatory to have the posshift in the ETS, so it is ensured safely coupled to a certain axis configuration. It is the absolute position at the axis zero position.

Response: Shift position for Center Reference or Absolute Encoders. The unit depends on '**dim**'.

Examples: !posshift 438.1338 35.7331 0 0
 !posshift y 35.7331
 ?posshift => 438.1338 35.7331 0.0000 0.0000
 ?posshift x => 438.1338

15. Move Instructions

Move instructions cause 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 user-defined **!pos**, while **mor** travels relative to the current position.

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

A special case of positioning is available with the **go** instruction. It does not move as a vector, here each axis travels at its own velocity and accel settings. Also, it only provides linear acceleration (accelfunc does not apply). The advantage of the **go** instruction is that it can be overwritten at any time with no need to abort the currently executed move. It will smoothly change directions. Target applications are e.g. focusing via a slider or tracking a mouse cursor position.

The third option for axis travel is a **speed** move. Here velocities are specified instead of positions. The addressed axes travel at those velocities until stopped, aborted or reaching a position limit. As with **go**, the speed instruction is also not executed as a vector and does not use the accelfunc. Speed requires the joystick to be enabled (**joy** and **joydir**).

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**).

Autostatus 1 reply after an automatic move has completed:

In default **autostatus=1**, there is an ASCII-reply when all moves have completed. It differs a bit from what **?statusaxis** returns (when polling the move state).

The auto-reply is triggered when all move instructions or **cal**, **rm** have ended. The reply is also sent if an axis is still traveling due to a speed instruction. The information about the 4 axes in sequence XYZA and a terminating dot ".". A reply for a 4 axis TANGO might look like this: @@@@.
The 4 axis characters might reply as follows:

'-' = this axis is not configured or available → e.g. @@--. for a 2 axis TANGO
'@' = this axis is at the desired position and idle
'M' = this axis is not idle (possibly still traveling due to speed instruction)
'A' = this axis has successfully completed a **cal** instruction (reply of **!cal**)
'D' = this axis has successfully completed a **rm** instruction (reply of **!rm**)
'S' = this axis was stopped by a hardware limit switch E0 or EE
'E' = this axis was stopped by an error, a stop signal, abort or in **limmode 1**
'L' = this axis move was cut by a position limit (in **limmode 2** only)

15.1. moa (Move Absolute)

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

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

Response: 5 ASCII characters, representing axes 1-4 and terminating '.'.
Example: "@@@-."
Depends on **autostatus** settings, which per default is set to 1. Each axis responses a '@' after successfully completing the move, or e.g. an 'E' if an error occurred, 'S' being stopped by a limit switch.
TANGO Desktop HE (in **limmode** 2) might also reply an 'L' when the target position was truncated by a software limit. For further information on possible response characters, please refer to **move instructions**, **autostatus** and **statusaxis**.

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

15.2. mor (Move Relative)

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

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

Response: 5 ASCII characters, representing axes 1-4 and terminating '.'.
Example: "@@@-."
Depends on **autostatus** settings, which per default is set to 1. Each axis responses a '@' after successfully completing the move, or e.g. an 'E' if an error occurred or an 'S' for being stopped by a limit switch.
TANGO Desktop HE (in **limmode** 2) might also reply an 'L' when the target position was truncated by a software limit. For further information on possible response characters, please refer to **move instructions**, **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: Similar to **mor**, the **m** command can be used to move one or more axes relative to their current position. It is useful to speed up communication especially for consecutive identical vectors. With the **m** command, the positions are defined by a preceding **distance** or **mor** command. **m** will move all enabled axes if their **distance** is not set to zero.

Response: 5 ASCII characters, representing axes 1-4 and terminating '.'.
Example: "@@@-."
Depends on **autostatus** settings, which per default is set to 1. Each axis responses a '@' or 'J' after successfully completing the move, or e.g. an 'E' if an error occurred, 'S' for switch. For further information on possible response characters, please refer to **move instructions**, **autostatus** and **statusaxis**.

To maximize performance, **autostatus** mode 3 can be set, where the position reached response string consists of only a [CR] ([CR] = 0x0d hex, or '\r').

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	m continues until:
!distance 1 2 0 0		(specify m distance)
m	will move to 4 6 5 6	
m	will move to 5 8 5 6	
m	will move to 6 10 5 6	
m	will move to 7 12 5 6	
.		
.		
.		

15.4. distance (Distance for m)

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

Description: 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 or by a relative move of the 'moe' instruction.

Distances must be defined for all axes, axes that shall not move have to be set to distance = 0.

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

Examples:

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

15.5. moc (Move to Center)

Syntax: !moc, moc or ?moc
Parameter: x, y, z, a or none (or a bitmask from Firmware >= 1.73)

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.

Remarks: From Firmware 1.73 it is possible to
- directly read the moc center position(s) by '?moc'
- execute a !moc with a bitmask: specified axes XYZA= 1,2,4,8

Response: 5 ASCII characters, representing axes 1-4 and terminating '.'
Example: "@@@-." Depends on **autostatus** settings (default=1). Each axis responses an '@' after successfully completing the move, or e.g. an 'E' if an error occurred, 'S' for switch. For further information on possible response characters, please refer to **move instructions**, **autostatus** and **statusaxis**.

Examples:
moc all axes travel to their center position
moc y Y-axis travels to the center position
moc 5 X(1) and Z(4) travel to their center positions (Firmw. 1.73)
?moc x => 25.0000 read center position of the X-axis (Firmware 1.73 or higher)

15.6. mol (Move to LockPos)

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

Description: The specified or all enabled axes travel to the transportlock position. This special move instruction may be used to position the microscope stage or axes to their transport position, where the fixation screw is inserted.

Requirements: 1) The axes must be calibrated (by **!cal**)
2) The lock position must be stored in the ETS or virtual ETS

Remarks: The factory defined lock positions can be read by **lockpos**. If the position value is greater 0, a lock position is available.

Response: 5 ASCII characters, representing axes 1-4 and terminating '.'
Example: "@@@-."
Depends on **autostatus** settings, which per default is set to 1. Each axis responses a '@' or 'J' after successfully completing the move, or e.g. an 'E' if an error occurred, 'S' for switch. For further information on possible response characters, please refer to **move instructions**, **autostatus** and **statusaxis**.

Examples:
mol all axes travel to their transportlock position (if available)
mol y Y-axis travels to to the transportlock position (if available)

15.7. go (Go To Position)

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

Description: Intended for tracking a position that changes randomly while the axes are still traveling, e.g. by a mouse or touchscreen. Similar to 'moa', it executes a move to an absolute position. For non-interrupted positioning, 'moa' should always be used.

The differences to a regular 'moa' instruction are:

- Go can be overwritten any time by another go position, even while traveling
- The velocity can be changed while traveling (**vel**)
- The move can be ended (at the normal acceleration) by sending a go instruction without position value, (or by setting vel=0 or **speed**=0)
"go" stops all axes, "go y" only the Y axis etc.
- Go is not a vector move, each axis travels at its own velocity (**vel**) and acceleration (**accel**)
- It only supports linear acceleration (no s-curve)
- It might introduce more shake or backlash than moa
- No "@@@" **autostatus** reply on completion (see Remarks)

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

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

Use TANGO Firmware 1.68/1.60S3 from March 2017 or higher. Snapshot Mode **snsn** 6 is ignored by the go instruction.

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)
go (or !speed 0 0 0) stops the go instruction for X, Y and Z axis (using accel)
go z (or !speed z 0) stops the go instruction for Z axis only
```

```
go 10 100
go 2 11.5 => change in direction while axis is already traveling
```

Sequence examples:

```
!vel z 10
go z 100 => starts an absolute move in Z at a velocity of 10
!vel z 1 => changes the velocity of Z to 1 while still traveling
!vel z 0.1 => changes the velocity of Z to 1 while still traveling
go 0.5 5.2 100.15 => start absolute move for X, Y and Z axis
!vel y 40 => changes the velocity of Y to 40 while still traveling
!vel 20 15 40 => changes velocities of X, Y and Z while still traveling
go 10 100 => starts an absolute move in two axes (X, Y), now...
"go x" or "!vel x 0" or "!vel 0" will end the X move at the default acceleration
"go y" or "!vel y 0" will end the Y move at the default acceleration
"go" or "!vel 0 0" will end the two axis move
```

15.8. speed (Speed Move)

Syntax: !speed or ?speed

Parameter: x, y, z, a or none
+-100 [rev/s], or as [mm/s] in dim 9 and 10 only

Description: Lets axes travel (or motors spin) at the specified velocities. A speed move is stopped by setting the speed to zero, by abort (**a**) and when reaching a software- (**lim**) or hardware-limit. Speed uses the acceleration set by **accel** and only provides linear acceleration.

Remarks: **Joydir** must be enabled for the corresponding axes to allow execution of the speed instructions, while the **joy** setting doesn't affect speed.

For endless rotation please refer to **modulomode=1** or **!zero**.

Setting speed to zero also stops a running **go** instruction.

Unless **dim 9** or **10** is used, the unit of speed is always in motor revolutions per second.

Example: If dim is < 9 at a 4mm pitch, to achieve 1mm/s, the !speed must be set to 0.25

Response: Currently executed speed in [rev/s], or [mm/s] at dim=9+10 with default 3 or specified 0 - 16 fractional digits
Speed does not generate an **autostatus** reply.

Examples:

```
!speed 29.3 0.01 start speed move for X=29.3[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 (or stops a go instruction)
!speed 0 stop speed move for X-axis (or stops a go instruction in X)
!speed z 0 stop speed move for Z-axis (or stops a go instruction in Z)
?speed read the currently executed speed of all axes
?speed z read the currently executed speed of Z-axis only, e.g. 0.000
?speed 5 read the currently executed speeds with 5 fractional digits
?speed z 6 read the currently executed Z speed with 6 fractional digits
```




15.9. moe (Move Extended)

Syntax: !moe or moe
Parameter: Bit-mask followed by the there specified amount of target positions and distances in order X->A

Availability: Only with TANGO Desktop HE controllers.

Description: Flexible move instruction, combines absolute and relative moves on any combination of axes. It is possible to move e.g. only X and A in one instruction (and as a vector) or Y and Z and as well A relative, etc. All sorts of variations on up to 4 axes. The only restriction is that each axis (bit) can only be set as either abs or rel.

The moe instruction relies on the flag bits sent with moe:

Bits 0...3 = Axes for absolute move: 1=X, 2=Y, 4=Z, 8=A
Bits 4...7 = Axes for relative move: 16=X, 32=Y, 64=Z, 128=A

With this mask, 'moe' then is executed as:

"moe [Sum of Bits] [position(s) or distance(s)]" or
"moe x[Bits as hex] [position(s) or distance(s)]"

The two options (integer and hex) exist, as a hex number might be easier to understand than an integer sum.

Some examples of abs/rel combinations:

RELATIVE (axes for mor)				ABSOLUTE (axes for moa)					
A	Z	Y	X	A	Z	Y	X	Flag	bit sum for
128	64	32	16	8	4	2	1		
x80	x40	x20	x10	x08	x04	x02	x01		
0	0	0	0	0	0	0	0	= 0	no axis
0	0	0	0	0	0	0	1	= 1	absolute X
0	0	0	0	0	0	1	0	= 2	absolute Y
0	0	0	0	0	0	1	1	= 3	absolute X + Y
0	0	0	0	0	1	0	1	= 5	absolute X + Z
0	0	0	0	0	1	1	0	= 6	absolute Y + Z
0	0	0	0	1	1	0	0	= 12	absolute Z + A
0	0	0	0	1	1	1	0	= 13	absolute Y,Z,A
0	1	0	0	0	0	0	0	= 64	relative Z
0	1	0	0	0	0	0	1	= 65	relative Z, absolute X

Response: For further information on possible response characters, please refer to **move instructions**, **autostatus** and **statusaxis**.

Examples: moe 5 3.3 7.5 (1+ 4 : X and Z absolute to 3.3 and 7.5)
moe 65 10 0.5 (1+64 : X absolute to 10, Z relative by 0.5)
moe x41 10 0.5 (x01|x40: X absolute to 10, Z relative by 0.5)
moe 48 0.5 1.5 (16+32 : X and Y relative by 0.5 and 1.5)

15.10. a (Abort the Current Move)

Syntax: !a or a or [hex 0x03]

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

Description: Aborts automatic moves (moa,mor,moc,go,...), cal/rm and speed instructions of all axes or of a specified axis and sets them into position reached state. The stop is performed with high acceleration of **stopaccel**. PCI-E and DT-E from Firmware 1.66, others from 1.68, support the optional parameter "-1", where the axis stops at regular **accel** acceleration (see examples).

Most TANGO controllers also provide the "Ctrl+C" (hex 0x03) special character stop, which bypasses the parser (faster response, independent of running instructions) to stop all axes and clear the input buffer.

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

HDI (joystick etc.) movement cannot be stopped by abort. The HDI must be disabled e.g. by setting **joy** or **joydir** to 0.

Accelfunc is not used, abort always stops with constant accel.

Response: Depends on the instruction being executed (moa,speed,go etc.) and **autostatus**. If a move is aborted in default **autostatus=1** and all axes stopped, the aborted axes auto response is 'E'.

The '**?err**' error state is not changed by the 'a' instruction and remains in the previous state.

Example: a (abort move of all axes , with **stopaccel**)
a y (abort move of Y axis only, with **stopaccel**)
a -1 (abort move of all axes , with **accel**)
a z -1 (abort move of Z axis only, with **accel**)

15.11. delay (Set the Delay Time for Consecutive Moves)

Syntax: ?delay or !delay

Parameter: 0 to 10000 [ms]

Description: This instruction inserts a delay time before executing a move (delayed start). One value applies to all axes. Applies to: moa,mor,moc,mol,moe,m and snapshot mode 7, not to the go instruction. Also refer to "**!pause**".

Response: Delay time in [ms]

Examples:

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

?delay Read the delay time

15.12. 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,moe,m. One value applies to 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

15.13. pos (Read or Set Position)

Syntax:	!pos or ?pos
Parameter:	x, y, z, a or none Position (within $\pm\text{maxpos}$)
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). From Firmware 1.73, 'pos' can also be sent without '?'. Even axes with encoders return the motor position by default. Returning the encoder position can be achieved by setting the corresponding ' encpos ' to 1. The effect of manipulating positions with '!pos' can be removed by the instruction ' posclr '. Also, the difference (offset) from the original zero position can be read by it.
Response:	Axis position(s) (depends on dim , also enc and encpos state)
Examples:	<pre>?pos Read all axis positions ?pos z Read Z axis position only !pos 100 200 Set the current X and Y axis positions !pos x 0 Set the current X position to zero !pos -0.1 Set the current X position to -0.1 (unit depends on dim) !pos y 2000 Set the current Y position to 2000 (unit depends on dim)</pre>

15.14. posclr (Clear Position Offset)

Syntax:	!posclr or ?posclr
Parameter:	x, y, z, a or none
Description:	Remove or read the position offset to the axis origin, which was added by a !pos instruction. The axis position can be redefined by '!pos'. In order to return to the original absolute position, posclr offers reading or clearing the changes made by !pos.
Response:	Position offsets (depending on dim)
Examples:	<pre>!posclr x (reset X to original position) !posclr (reset all positions to original position) ?posclr (read position offset of all axes) ?pos => 1.0000 2.0000 3.0000 !pos y 8 (here a position offset of 8-2=6 is added to Y) ?pos => 1.0000 8.0000 3.0000 ?posclr => 0.0000 6.0000 0.0000 !posclr (here the !pos set position offsets are removed) ?pos => 1.0000 2.0000 0.0000</pre>

15.15. zero (Set Internal Position to Zero)

Syntax: `!zero` or `zero`

Parameter: `x`, `y`, `z`, `a` or none

Description: Unlike "`!pos 0`", the "`!zero`" instruction also resets the internal position counter to zero. It can be used in applications where axes exceed the position limits, e.g. filter wheels (in such case a "`!pos 0`" instruction is not sufficient, as internal limits will apply). In order to ensure the reference point remains at the same position, the zero instruction should be executed after completing one or several complete revolutions.

Remarks: For rotational axes '`modulomode`' can be used. In these modes, setting the axes to zero is not required.

Response: none.

Examples:

`!zero` Set all internal positions to zero

`!zero z` Set Z axis position to zero

15.16. 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 Instructions (Joystick, Trackball, ERGODRIVE)

The HDI (human device interface) provides manual control of the axes and also supports special functions.

The HDI interface accepts hot plugging of the devices. It is possible to unplug, plug in, or exchange the input devices during operation of the TANGO controller.

The HDI velocities are limited to 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 software limits (defined by **lim** instruction).

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

The **keymode** functionality enables selection of different **keyspeed** velocities by pressing the function keys F1-F4 of the joystick. Please refer to **keymode** for further information.

The optional **multi-function wheel**, found on several HDI devices, can be assigned to any axis (instruction **zaxis**) and the LED100 brightness (via **hdimode**).

16.1. joy (Generally Enable/Disable HDI)

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

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

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

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

Response: HDI enable state

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

Behavior examples / sequences:

```
-----  
?joy     ==> 2           (HDI is enabled)  
!joy 0   ==> @@-..       (response when switched enable→disable with autostatus on)  
!joy 0                   (no response when joy already is disabled)  
!joy 2                   (no response when joy is enabled)  
-----
```

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

```
!joydir 0 0 0 0         (workaround #2 is to use 'joydir' instead)  
!joydir 2 2 2 2         (make sure the direction is correct, 2 or -2)  
-----
```

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

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

Description: In addition to the '**joy**' instruction, joydir can be used to enable/disable individual HDI axes and set their directions. For compatibility, a value of 1 or -1 has the same effect as 2 or -2. 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: To activate the joystick, please also make sure that the joystick is globally enabled by the '**joy**' instruction.

When using a 4 axis controller with a 3 axis joystick, its 3rd axis will be used to control axes 3 and 4. The selection which axis (3 or 4) is controlled must be selected by enabling or disabling the corresponding joydir. By default, the joydir of the 4th axis is disabled (0). Refer to examples below.

This instruction also enables or disables (0) the '**speed**' move for the corresponding axes, but does not affect the speed move directions (joydir 2 or -2 enables **speed**).

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. joychangeaxis (Change Joystick X and Y Axis)

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

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

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

Remarks: Only for Joystick devices.

Response: Joystick X-Y axis change state

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

16.4. joywindow (Joystick Window)

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

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

Response: joywindow [in digits]

Examples: ?joywindow read joystick window
!joywindow 14 set joystick window to ± 14

16.5. joyvel (Joystick Velocity)

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

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

Response: Currently used joystick velocities

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

16.6. joyspeed (Joystick Speed Presets for BPZ Device)

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

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

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

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

16.7. keymode (Joystick Key Mode)

Syntax: !keymode or ?keymode

Parameter: 0, 1 or 2

Description: Instead of the usual **vel** (or **joyvel** in extmode 1), keymode assigns the independent **keyspeed** velocities to the joystick. Each axis has its two dedicated **keyspeeds**, which can be selected (toggled) by the joystick function keys. The way how the speeds can be toggled is defined by '**hdimode**', bit 1.

The different keymodes 1 and 2 can be used to select the preferred speeds which are active on startup.

Keymode can be set to:

0 = keyspeed disabled, vel or joyvel is used (default)

1 = keyspeed for X,Y and Z velocity, preset is **keyspeed 1**

2 = same as 1, but preset is **keyspeed 2**

Depending on the selected toggle mode (by '**hdimode**' bit 1), the behavior is

A) **hdimode** bit 1 = 0 (no joystick toggle mode):

F4 : selects keyspeed 1 values of X and Y axis,

F1 : selects keyspeed 2 values of X and Y axis.

F3 : selects keyspeed 1 value of the Z axis,

F2 : selects keyspeed 2 value of the Z axis.

The F-key does not have to be pressed all time, as it immediately switches to the desired keyspeed.

B) **hdimode** bit 1 = 1 (joystick toggle mode):

F1 : toggles XY between the keyspeed values 1 and 2

F4 : toggles Z keyspeeds 1,2 (firmware 1.56 or higher)

Remarks: Please note that other special functions which require the same joystick buttons (e.g. **snapshot modes**) might interfere with keymode (keymode will still behave as required, but other functions might then be executed as well).

For joysticks with the optional multi-function wheel, three wheel velocities are available by pressing F1, F4 or no key. Please refer to the '**zwtravel**' description.

Response: keymode as decimal number

Examples:

!keymode 1 keyspeed 1 is active on startup

?keymode => 1 (keymode 1 is selected)

16.8. keyspeed (Joystick Key Speed Presets)

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

Description: Two Joystick velocities can be set for each axis individually. The first parameter is called the slow value and the second parameter is fast. Unit is always mm/s, independent from **dim**. The keyspeeds become available through **keymodes** 1 and 2.

Remarks: In **Keymode**, the X and Y keyspeeds do toggle together by F4/F1, while the Z keyspeeds are toggled separately by F3/F2. A different toggle mode can be set by **hdimode** bit 1, where F1 toggles between the keyspeeds for X+Y instead of F4/F1 and F4 toggles between the keyspeeds for Z instead of F3/F2.

The keyspeeds are limited to '**secvel**' as long as no **cal+rm** sequence is executed.

Hints: Setting one keyspeed to zero could be used to have a startup setting where movement by the joystick is in fact disabled and the user has to press a joystick function key in order to use the joystick (toggle it to a speed different from zero).

Response: Two floating point values per axis (1="slow" and 2="fast")
one axis : [keyspeed1] [keyspeed 2]
more axes: [k.spd.1_x] [k.spd.2_x] [k.spd.1_y] [k.spd.2_y] ...

Examples:
?keyspeed x => 1.00 10.00 (Read X Joystick keyspeed)
?keyspeed => 2.00 20.00 2.00 20.00 0.10 1.00 (Reply of a 3 axes TANGO)
!keyspeed z 0.1 1 (Set keyspeed1 to 0.1 and keyspeed2 to 1 [mm/s] for Z)
!keyspeed 5 20 2 10 0.2 2 (Set keyspeed of 3 axes at once)

16.9. joycurve (Joystick Characteristic)

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

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

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

Remarks: The default (logarithmic) setting is the most useful as it allows very fine positioning at lower deflections and high velocities towards the full deflection.

Response: Currently used characteristic

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

16.10. key (Read HDI Device Key State)

Syntax: ?key or key

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

Description: This instruction reads the state of all 4 or the specified HDI device key(s).

0 = key is currently released or not available

1 = key is currently pressed

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

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

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

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

16.11. keyl (Read HDI Device Latched Key State)

Syntax: keyl, ?keyl or !keyl

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

Description: The ?keyl or keyl instruction reads the latched state of the specified or all 4 HDI device keys. The latched state of the requested key(s) is cleared after reading.

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

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

1 = key is/was pressed since last keyl 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 2 => clear latch state of key 2 only (to zero)

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

16.12. hwfactor (Coaxial-/ERGODRIVE Transmission Factor)

Syntax: !hwfactor or ?hwfactor

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

Description: Aaxis travel distance in millimeter per coaxial drive revolution. Negative factors reverse the travel direction. (The hardware provides about 100000 steps per revolution.)

Remarks: Some HDIs provide a switch to change between two different factors. Please refer to '**hwfactorb**'.

Response: Currently used factor(s)
As floating point number(s) between -200.0 and +200.0

Examples:

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

16.13. hwfactorb (Alternate Coaxial-/ERGODRIVE Factor)

Syntax: !hwfactorb or ?hwfactorb

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

Description: Alternate (second) parameter for travel distance per coaxial drive revolution '**hwfactor**'. Available with e.g. ERGODRIVE and Pilot stage. Negative factors reverse the travel direction.

Response: Currently used alternate coaxial drive factor(s)
As floating point number(s) between -200.0 and +200.0

Examples:

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

16.14. hwfilter (Coaxial-/ERGODRIVE Noise Filter)

Syntax: !hwfilter or ?hwfilter

Parameter: 0 or 1

Description: Coaxial drive noise filter.

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

The filter is activated/deactivated for X and Y axes at once.
** Disabling the filter can result in a finer resolution, but it also causes position inaccuracy e.g. between automatic moves or when the axis is not moving: Its signal noise will cause a permanent slight position jitter.

Response: State of the coaxial drive noise filter

Examples:

```
!hwfilter 0      Disable noise filter
?hwfilter        Read hwfilter state
```



16.15. **tbfactor (Trackball Factor)**

Syntax: `!tbfactor` or `?tbfactor`

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

Description: Set or read 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 (similar to 'joydir'). The default factor is 1.

Response: Currently used trackball factor(s)

Examples:

```
!tbfactor x 100 Set X axis 100 times more sensitive than the default  
!tbfactor y 12.5 Set Y axis 12.5 times more sensitive than the default  
!tbfactor 0.5 0.5 Set X+Y axis to half the default sensitivity  
?tbfactor Read sensitivity factor of all axes
```

16.16. zwheel (Is Multi-function Wheel Available)

Syntax: ?zwheel or zwheel
Parameter: none

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

 0 = HDI device has no multi-function wheel
 1 = HDI device has a multi-function wheel

Remarks: To identify the HDI device, use '**hdi**' instruction

Response: 0 or 1

Example: ?zwheel => 0

16.17. zwtravel (Multi-function Wheel Travel per Revolution)

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

Description: Set or read the travel distances for one revolution of the multi-function wheel, available with several HDI devices, e.g. ERGODRIVE and Joystick.

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

 Presets for travel distance are
 1: 0.1 mm/rev (default factor)
 2: 0.01 mm/rev (alternate factor, factory preset to slow)
 3: 1.0 mm/rev (alternate factor, factory preset to fast)

Remarks: ERGODRIVE and Pilot stage only offer switching between two travel distances. In this case distance parameter 1 remains the default; parameter 3 is used as alternate second factor.

 '**secvel**' and '**vel**' (or '**joyvel**' in **extmode** 1) may prevent faster traveling when turning the wheel.

 It is possible to set negative values and by this offering direction change via HDI key.

 The multi-function wheel can also be assigned to other axes with the '**zwaxis**' instruction. By default, it is set to Z.

Hints: For safety reasons, the default travel can be set to zero. So the axis will move only when a key is pressed (F1, F4).

Response: Travel distance(s) of the multi-function wheel

Examples:

```
?zwtravel            Read all 3 travel distances: [1:default] [2:slow] [3:fast]
?zwtravel 1         Read "default" travel distance
?zwtravel 2         Read "slow" travel distance
!zwtravel 3 2.5     Set "fast" travel distance to 2.5 [mm/revolution]
!zwtravel 1 0       Set "default" parameter to zero (inactive without keypress)
```

16.18. zwaxis (Multi-function Wheel Axis)

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

Description: Assign multi-function wheel to an axis (default: z)

Response: x, y, z or a

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

16.19. zwfactor (Multi-function Wheel Factor)

Syntax: !zwfactor or ?zwfactor
Parameter: 0, 1, 2 ... to 20

Description: For custom designed applications only.
 Increase Wheel transmission multiplier, default=1.

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

Remarks: Useful in custom designs where the application requires a different rotary encoder to provide the wheel functionality. If the chosen encoder type provides less resolution than the Joystick multi-function wheel, zwfactor can be used to adapt the behavior (here: to achieve correct **zwtravel** distances). The multi-function wheel has a resolution of 480 counts/rev. In example if the encoder has 128 counts/rev, zwfactor can be set to 4.

Response: Multiplier

Example: !zwfactor 1 (set default multiplier, as for TANGO HDIs)
 !zwfactor 5 (set multiplier for lower res. encoder wheel)
 ?zwfactor (read the currently used multiplier)

16.20. zwpos (Multi-function Wheel Position Counter)

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

Description: Independent position counter of the Multi-function Wheel.

Remarks: Always counts 480 counts/rev, **zwfactor** has no influence. Only counts on **zwaxis** axes that have HDI enabled by **joy** and **joydir** and are available on the controller.

Response: Counter value as 32 Bit signed integer

Example: zwpos (read the position counter, same as ?zwpos)
 !zwpos 0 (set position counter to zero)
 !zwpos 2000 (set position counter to 2000)

16.21. tvrjoy (Pulse and Direction Joystick Functionality)

Syntax: !tvrjoy or ?tvrjoy

Parameter: 0, z, a

Description: Enables and assigns the AUX IO pulse&direction inputs TAKT_IN, V/R_IN to an axis, providing a pulse and direction interface. The behavior is similar to the trackball, which is available as HDI device.

Remarks: The function does not provide absolute positioning accuracy. It can be compared to a HDI device behavior.

This function is also available without connecting a HDI device, by entering **tvr mode 5** for the corresponding axis

0 = Disabled (default)

z = Assigned to Z-axis

a = Assigned to A-axis

Response: Currently assigned axis

Examples:

!tvrjoy 0 Disable AUX I/O tvr joystick function

!tvrjoy z Assign AUX I/O tvr joystick function to Z-axis

?tvrjoy Query assigned axis

16.22. tvrjoyf (Pulse and Direction Joystick Factor)

Syntax: !tvrjoyf or ?tvrjoyf

Parameter: -200.0 to +200.0

Description: Set or read the AUX I/O tvrjoy transmission factor as floating point number between -200.0 and +200.0. A sign change can 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

16.23. hdi (Read HDI ID)

Syntax: ?hdi or hdi
Parameter: none or -1

Description: Read 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.

The additional ?hdi -1 option is available from firmware 1.68. It returns the ID and % match plus a descriptive text, which also includes the optional multi-function wheel.

ID numbers: 0, 1, 2, ... 15, (16 or -1 = no device connected)
ID match : 0 (poor) ... 100 (good)

ID	DEVICE
0	(Reserved for special devices)
1	Coaxial drive
2	Custom designed console
3	ERGODRIVE
4	SmartMove
5	Custom device
6	Custom device
7	Stand alone Jogwheel/Multi-Function Wheel
8	-
9	-
10	2x 2-Axis Joystick or 4-Axis Jogbox
11	Trackball with 2-Axis Joystick
12	Joystick 2-Axis
13	Trackball with 3-Axis Joystick
14	Trackball
15	Joystick 3-Axes
16	No device connected
17	(Device identification in progress)
18	(Device initialization in progress)
-1	No device connected (Digital HDIs)

Remarks: The instruction may be used to identify the connected HDI device. Additionally, '**zwheel**' can be used to identify if the device provides an additional multi-function wheel.

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

Example: ?hdi => 16 99 (ID ≥16 = no hdi connected)
 ?hdi => 12 97 (meaning: HDI ID = device nr. 12, 97% match)
 ?hdi -1 => 12 97 [Joystick 2 axis] (device description text)
 ?hdi -1 => 15 93 [Joystick 3 axis with MF-Wheel]

16.24. hdimode (HDI Mode Options)

Syntax: ?hdimode or !hdimode
Parameter: a string of 0s and 1s (to set LSB or more bits at once)
or two numbers, 0 to 15 and 0 or 1 (to set a single bit)

Description: Select 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) and can be truncated at any length (not all 16 bits required). Setting a bit to 1 enables the functionality, 0 disables it.

<u>Bit</u>	<u>Function</u>
------------	-----------------

- | | |
|-----|---|
| 0: | Toggle Mode for ERGODRIVE XY and Z Keys (0=off, 1=on) |
| 1: | Toggle Mode for Joystick (in KeyMode 1 or 2)
0=select KeySpeed velocity XY with F1+F4, Z with F2+F3
1=toggle KeySpeed velocity XY by just pressing F1
F4 is used to toggle Z (firmware 1.56 and higher) |
| 2: | LED100 brightness control via Joystick or ERGODRIVE (1=enabled). The AUX I/O TAKT_OUT pin is controlled automatically then (LED digitally switched of at 0%)
Remarks: If enabled, it can interfere with the second Trigger Output respectively adigout 0 . |
| 3: | LED100 fine manual brightness control,
16x finer with Multifunction Wheel
4x finer when controlled via Joystick Y deflection
(1=fine mode enabled, only in conjunction with Bit 2) |
| 4: | LED100 as main function of the Multifunction Wheel.
The function key (Joystick: F2, ERGODRIVE: F1) which must be pressed to control the LED is now used to drive the assigned axis instead. |
| 5: | - reserved - |
| 6: | Joystick Z knob drives the A axis (axis 4 instead of 3) |
| 7: | Diagonal mode (HDI X applied to X and Y simultaneously) |
| 8: | Trackball Y axis drives the Z axis (X then is disabled) |
| 9: | Joystick Y drives Z axis (and Z drives Y, if 3 axis JS) |
| 10: | Joystick Z knob is auto-disabled while X or Y deflected |
| 11: | - reserved - |
| 12: | - reserved - |
| 13: | - reserved - |
| 14: | - reserved - |
| 15: | - reserved - |

Response: Single mode bit or all 16 mode bits in one ASCII string

Examples:

```
!hdimode 2 1 Set mode bit 2 to 1 (on) = LED100 brightness control via HDI
!hdimode 3 0 Set mode bit 3 to 0 (off)= Coarse LED100 brightness control
!hdimode 1 Set mode bit 0 to 1 (on) (equal to "!hdimode 0 1")
!hdimode 100010 Set mode bits 0 and 4 to "on", bits 1,2,3,5 to "off". Bits
6...15 are left unchanged.
?hdimode 1000100000000000 Read the state of all 16 mode bits, LSB left
?hdimode 0 Read the state of mode bit 0 (ERGODRIVE toggle mode)
```

16.25. configaxsel (Joystick Axis Select Option)

1 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 (by joydir a/see remarks), the Z knob of the joystick either drives Z (F4 key released) or A (F4 key pressed).

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 (F4 applies Z→A with 3 axes joystick)
!configaxsel 0 Axis select disabled (default)
?configaxsel Read the axis select configuration (returns 0 or 1)

17. Joystick Function Key Assignments

The Joystick provides 4 function keys, F1-F4. The key states can be read by the **key** and **key1** instructions. Several operating modes of the TANGO controller also assign special functions to the F-keys. The chart shows the key assignments for the different modes:

Mode / Key		F1	F2	F3	F4
SnapShot Mode	0	-	set new point	-	-
	1	-	next point	-	-
	2	previous point	next point	prehome & first point	prehome & home
	3	-	start dissection	-	-
	7	Move sequence prehome & home	autoinc from 1st point	pause/continue	pause & previous point
	9	Relative Jump back	Relative Jump forward	-	-
Axis Select Mode ²⁾		-	-	-	Joystick Z-Axis controls A-Axis ¹⁾
KeyMode		Select X,Y KeySpeed2	Select Z KeySpeed2	Select Z KeySpeed1	Select X,Y KeySpeed1
KeyMode+Toggle		Toggle X,Y KeySpeed	Toggle Z KeySpeed	-	-
Joystick has wheel		zwtravel3 (fast) ¹⁾	-	Wheel "Joystick" ¹⁾	zwtravel2 (slow) ¹⁾
LED Mode no wheel		-	-	Y-axis controls LED brightness ¹⁾	-
				F3+F4: Store LED brightness	
LED Mode w. wheel		zwtravel3 (fast) ¹⁾	Wheel controls LED brightness ¹⁾	Wheel "Joystick" ¹⁾	zwtravel2 (slow) ¹⁾
				F3+F4: Store LED brightness	

1) Function selected only as long as key pressed.
In all other cases the function is selected or executed by pressing the key.

2) F4 can be configured as Axis Select (Z<->A). Please refer to '**configaxsel**'.

Remarks: When selecting more than one mode, function keys may become assigned to several functions at once.

Joysticks with multi-function wheel behave different in case of LED control than josticks without a wheel (refer to chart above).

18. Digital and Analogue I/O

TANGO Desktop, Desktop-S/-E/ HE, PCI/PCI-S/PCI-E and TANGO 3 mini controllers provide several I/O options which become available with

1) the (optional) auxiliary I/O port "AUX I/O"

Which provides 5V digital I/O, analogue output(s), analogue input and more. In case of Desktop HE, the AUX I/O is always present, and the TANGO 3 mini has a slightly different "AUX mini". Other controllers can be ordered with AUX I/O. For command options, refer to **adigin**, **adigout**, **anain**, **anaout**, **anamode** and in case of TANGO 3 mini or Desktop HE also **adigintyp** and **adiginfunc**.

2) optional I/O extension port modules (Multi I/O preferred)

TANGO PCI-E or HE based controllers with optional **I/O1** or **Multi I/O** port module provide additional 24 or 12 digital inputs and 8 digital outputs via **digin/digout**, **edigin/edigout** etc. '**det**' may be used to check if a module is installed. A preset value can be assigned to the outputs which defines the initial state after power up.

Remarks: In case of I/O1 and Multi I/O modules, the **brake** functionality can interfere with the digital outputs. If a motor brake is activated, it occupies the specified output pin(s) which are controlled by the brake and can't be set by **digout**, **edigout**.

18.1. digin (I/O1 Digital Inputs)

Syntax: ?digin or digin
Parameter: none or 0 to 23

Remarks: Only available with TANGO PCI-E/DT-E extension module I/O1.
For Multi I/O modules, please refer to 'edigin'.

Description: This instruction reads the logic state of one or all digital inputs of the optional digital I/O1 extension.
If called without parameter, all inputs are returned as a string of 24 characters. If called with parameter (input number), only the state of the specified input is returned.

Response: logic state of digital input(s)
ASCII string 0 or 1, LSB (IN0) is the first/leftmost character
0 = low, 1 = high (depends on the polarity setting **diginpol**)

Examples:
?digin read all 24 digital inputs (e.g. 000000000000000000000000)
?digin 8 read logic level of input 8 (response e.g. 1)

18.2. digout (I/O1 Digital Outputs)

Syntax: !digout or ?digout
Parameter: string of up to 8 characters 0 and 1,
or single bit access with output number 0 to 7 and state 0, 1

Remarks: Only available with TANGO PCI-E/DT-E extension module I/O1.
For Multi I/O modules, please refer to 'edigout'.

Description: This instruction sets or reads back the logic level of one or all digital outputs of the optional digital I/O1 extension.
Outputs may be set either by a string of levels (up to eight 0s and 1s) or by output number and signal level.
The string is LSB first (output 0 is the leftmost).

Response: current output state(s)

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'.
!digout 100 The digital output 0 is set to logic '1' and the outputs 1 and 2 are set to logic '0'. Outputs 3 to 7 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 5 read the state of output 5

18.3. diginpol (I/O1 Digital Input Polarity)

Syntax: !diginpol or ?diginpol
Parameter: string of up to 24 characters 0 and 1,
or single bit with two numbers 0 to 23 and 0 or 1

Remarks: Only available with TANGO PCI-E/DT-E extension module I/O1.
For Multi I/O modules, please refer to 'ediginpol'.

Description: This instruction sets or reads back if the I/O1 input signal inverters are activated or not. Each of the 24 inputs can be inverted individually.
The inverter may be set either by a string of levels (up to 24 0s and 1s) or by specifying the input number and inverter state. The string is LSB first (input 0 is the leftmost).

Response: current inverter setting(s)

Examples:
!diginpol 010000000000000000000000 Set all inverter states (IN1 to inverted)
!diginpol 11000 The digital inputs IN0 and IN1 are set inverted, IN2, 3 and 4 are set to non inverted, input IN5 to IN23 settings are left unchanged
!diginpol 5 1 activate inverter of digital input IN5 only
!diginpol 17 0 disable inverter of digital input IN17 only
?diginpol read the inverter setting of all 24 inputs
?diginpol 5 read the inverter setting of input 5 only

18.4. digintyp (I/O1 Digital Input Type)

Syntax: !digintyp or ?digintyp
Parameter: string of up to 6 characters 0 and 1,
or single bit access with block number 0 to 5 and state 0, 1

Remarks: Only available with TANGO PCI-E/DT-E extension module I/O1.
24 Volt configured I/O1 modules only work with pull-down (0).
For Multi I/O modules, please refer to 'edigintyp'.

Description: This instruction sets or reads back if the I/O1 pull-up / pull-down resistor settings for the 24 inputs.
The resistors are arranged in 6 blocks. Therefore inputs can only be accessed blocks of 4: IN0-3, IN4-7, ... IN20-IN23

0 = pull down
1 = pull up

The pull up/down may be set either by a string of levels (up to 6 0s and 1s) or by specifying the block number and level.
The string is LSB first (block 0 is the leftmost).

Response: current pull up/down setting(s)

Examples:
!digintyp 110000 Set all pull up/downs (IN0-IN7 to pull up, rest to pull down)
!digintyp 001 Set pull up/down of IN0-IN7 to pull down, IN8-IN11 to pull up the settings for IN12 to IN23 are left unchanged
!digintyp 4 1 set block 4 (IN16-IN19) to pull up
!digintyp 2 0 set block 2 (IN8-IN11) to pull down
?diginpol read the pull up/down setting of all 6 blocks
?diginpol 5 read the pull up/down setting of input 5 only

18.5. digoutpreset (I/O1 Digital Output Presets)

Syntax: !digoutpreset or ?digoutpreset

Parameter: string of up to 8 characters 0 and 1,
or single bit access with output number 0 to 7 and state 0, 1

Remarks: Only available with TANGO PCI-E/DT-E extension module I/O1.
For Multi I/O modules, please refer to 'edigoutpreset'.

Description: This instruction sets or reads back the logic levels of one or all digital outputs of the optional digital I/O1 extension, which are applied after power up of the TANGO. Output preset levels may be set either by a string of levels (up to eight 0s and 1s) or by output number and signal level. The string is LSB first (output 0 is the leftmost).

Response: output preset value(s)

Examples:

```
!digoutpreset 11110000 After power on, the digital outputs 0,1,2,3 are set to
                        logic '1' and the outputs 4,5,6,7 are set to logic '0'
!digoutpreset 100 After power on, the digital output 0 is set to logic '1' and
                  the outputs 1 and 2 are set to logic '0'. Outputs 3 to 7 are
                  left unchanged.
!digoutpreset 5 1 set preset value of output 5 to logic 1 (high)
!digoutpreset 7 0 set preset value of output 7 to 0 (low)
?digoutpreset read the state of all outputs
?digoutpreset 5 read the state of output 5
```


18.6. edigin (Multi I/O Digital Inputs)

Syntax: ?edigin or edigin
Parameter: none or 0 to 11

Remarks: Only available with the Multi I/O port of TANGO PCI-E/DT-E.

Description: This instruction reads the logic state of one or all digital inputs of the optional Multi I/O digital extension port. If called without parameter, all inputs are returned as a string of 12 characters. If called with parameter (input number), only the state of the specified input is returned.

Response: logic state of digital input(s)
 ASCII string 0 or 1, LSB (IN0) is the first/leftmost character
 0 = low, 1 = high (depends on the polarity setting **ediginpol**)

Examples:
?edigin read all 12 digital inputs (e.g. 000000000000)
?edigin 8 read logic level of input 8 (response e.g. 1)

18.7. edigout (Multi I/O Digital Outputs)

Syntax: !edigout or ?edigout
Parameter: string of up to 8 characters 0 and 1,
 or single bit access with output number 0 to 7 and state 0, 1

Remarks: Only available with the Multi I/O port of TANGO PCI-E/DT-E.

Description: This instruction sets or reads back the logic level of one or all digital outputs of the optional Multi I/O extension port. Outputs may be set either by a string of levels (up to eight 0s and 1s) or by output number and signal level. The string is LSB first (output 0 is the leftmost).

Response: current output state(s)

Examples:
!edigout 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'.
!edigout 100 The digital output 0 is set to logic '1' and the outputs 1 and
 2 are set to logic '0'. Outputs 3 to 7 are left unchanged.
!edigout 5 1 set digital output 5 to logic 1 (high)
!edigout 7 0 set output 7 to 0 (low)
?edigout read the state of all outputs
?edigout 5 read the state of output 5

18.8. ediginpol (Multi I/O Digital Input Polarity)

Syntax: !ediginpol or ?ediginpol
Parameter: string of up to 12 characters 0 and 1,
or single bit access with input number 0 to 11 and state 0, 1

Remarks: Only available with the Multi I/O port of TANGO PCI-E/DT-E.

Description: This instruction sets or reads back if the Multi I/O input signal inverters are activated or not. Each of the 12 inputs can be inverted individually.
The inverter may be set either by a string of levels (up to 12 0s and 1s) or by specifying the input number and inverter state. The string is LSB first (input 0 is the leftmost).

Response: current inverter setting(s)

Examples:
!diginpol 010000000000 Set all inverter states (IN1 to inverted)
!diginpol 11000 The digital inputs IN0 and IN1 are set inverted, IN2, 3 and 4 are set to non inverted, input IN5 to IN11 settings are left unchanged
!diginpol 5 1 activate inverter of digital input IN5 only
!diginpol 7 0 disable inverter of digital input IN7 only
?diginpol read the inverter setting of all 12 inputs
?diginpol 5 read the inverter setting of input 5 only

18.9. edigintyp (Multi I/O Digital Input Type)

Syntax: !edigintyp or ?edigintyp
Parameter: string of up to 12 characters 0 and 1,
or single bit access with input number 0 to 11 and state 0, 1

Remarks: Only available with the Multi I/O port of TANGO PCI-E/DT-E.
24 Volt configured Multi I/O only works with pull-down (0).

Description: This instruction sets or reads back if the Multi I/O pull-up / pull-down resistor settings for the 12 inputs. Each of the 12 resistors can be set individually.

0 = pull down
1 = pull up

The pull up/down may be set either by a string of levels (up to 12 0s and 1s) or by specifying the input number and level. The string is LSB first (input 0 is the leftmost).

Response: current pull up/down setting(s)

Examples:
!edigintyp 110000000000 Set all pull up/downs (IN0,IN1 to pull up, rest down)
!edigintyp 001 Set IN0+IN1 to pull down, IN1 to pull up, rest left unchanged
!edigintyp 4 1 set IN4 to pull up
!edigintyp 2 0 set IN2 to pull down
?ediginpol read the pull up/down settings of all 12 inputs
?ediginpol 5 read the pull up/down setting of input 5 only

18.10. edigoutpreset (Multi I/O Digital Output Presets)

Syntax: !edigoutpreset or ?edigoutpreset
Parameter: string of up to 8 characters 0 and 1,
 or single bit access with output number 0 to 7 and state 0, 1

Remarks: Only available with the Multi I/O port of TANGO PCI-E/DT-E.

Description: This instruction sets or reads back the logic levels of one or all digital outputs of the optional Multi I/O extension port, which are applied after power up of the TANGO. Output preset levels may be set either by a string of levels (up to eight 0s and 1s) or by output number and signal level. The string is LSB first (output 0 is the leftmost).

Response: output preset value(s)

Examples:

```
!digoutpreset 11110000  After power on, the digital outputs 0,1,2,3 are set to
                        logic '1' and the outputs 4,5,6,7 are set to logic '0'
!digoutpreset 100      After power on, the digital output 0 is set to logic '1' and
                        the outputs 1 and 2 are set to logic '0'. Outputs 3 to 7 are
                        left unchanged.
!digoutpreset 5 1      set preset value of output 5 to logic 1 (high)
!digoutpreset 7 0      set preset value of output 7 to 0 (low)
?digoutpreset          read the state of all outputs
?digoutpreset 5        read the state of output 5
```

18.11. edigrly (Multi I/O Relay Option Access)

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

Remarks: Only available with the Multi I/O port of TANGO PCI-E/DT-E and installed relay option.

Description: Switch the optional relay to

 1 = ON
 0 = OFF

Response: Relay ON/OFF state

Examples:

```
!edigrly 1              Switch relay to ON
?edigrly                read relay state (e.g returns 0)
```

18.12. adigin (AUX I/O Digital Input)

Syntax: ?adigin or adigin
Parameter: none or 0 to 3 or 4

Description: Read Digital inputs of the AUX I/O or AUX mini connector. Returns the logic state of one or all digital inputs on the AUX I/O or AUX mini connector. If no parameter is used, all inputs are returned as an ASCII character string of 0 or 1.

0 = Bit 0: AUX I/O Pin 1 (Takt In) **
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)
And TANGO Desktop HE adds:
4 = Bit 4: Motor Connector Pin 23 (TRIN)

Remarks **: Bit 0 (Takt In) is not available with TANGO PCI-S and DT-S.

TANGO 3 mini: The TANGO 3 mini controller has a different bit assignment:

0 = Bit 0: AUX mini Pin 1 (Takt In)
1 = Bit 1: Motor Connector Pin 7 (TrIn)
2 = Bit 2: not available (Dummy)
3 = Bit 3: AUX mini Pin 2 (SnapShot)

Response: Logic state of the digital input(s), LSB first (LSB->MSB)

Examples:

?adigin => 0111 read all AUX I/O digital inputs (here: "Takt In" is 0)
?adigin 3 => 1 read AUX I/O digital input 3 ("SnapShot2")

18.13. adigintyp (AUX I/O, AUX mini Digital Input Type)

Syntax: !adigintyp or ?adigintyp
Parameter: input bit: none or 0 to 3 or 4
and type : 0 or 1

Availability: With TANGO 3 mini and TANGO Desktop HE only.

Description: Select a pull-up or pull-down resistor for the digital inputs: 1 = pull-up (default), 0 = pull-down.

TANGO 3 mini: 0 = Bit 0 = AUX mini Pin 1 (Takt In)
1 = Bit 1 = Motor Connector Pin 7 (TrIn)
2 = Bit 2 = not available (Dummy)
3 = Bit 3 = AUX mini Pin 2 (SnapShot)

TANGO Desktop HE: 0 = Bit 0: AUX I/O Pin 1 (Takt In)
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)
4 = Bit 4: Motor Connector Pin 23 (TRIN)

Response: State(s) of the digital input pull-up/pull-down resistor(s)

Examples:

!adigintyp 0111 Set input 0 to pull-down, inputs 1,2 and 3 to pull-up
!adigintyp 1 0 Set input 1 to pull-down, other inputs remain unchanged
?adigintyp Read all digital input pull-up/downs (response e.g. 0011)
?adigintyp 3 Read pull-up/down setting of digital input 3 (e.g. 1)

18.14. adiginfunc (AUX I/O, AUX mini Digital Input Function)

Syntax: !adiginfunc or ?adiginfunc
Parameter: input bit : none or 0 to 3 or 4
and function: 0, 1 or 2

Availability: With TANGO 3 mini and TANGO Desktop HE only.

Description: Assign a function to the digital inputs.

Input bits:

TANGO 3 mini:	0 = Bit 0 = AUX mini	Pin 1 (Takt In)	→default=0
	1 = Bit 1 = Motor Connector	Pin 7 (TrIn)	→default=0
	2 = Bit 2 = not available	(Dummy)	→default=0
	3 = Bit 3 = AUX mini	Pin 2 (SnapShot)	→default=2
TANGO Desktop HE:	0 = Bit 0: AUX I/O	Pin 1 (Takt In)	→default=0
	1 = Bit 1: AUX I/O	Pin 2 (V/R In)	→default=0
	2 = Bit 2: AUX I/O	Pin 3 (Stop)	→default=1
	3 = Bit 3: AUX I/O	Pin 4 (SnapShot2)	→default=2
	4 = Bit 4: Motor Connector	Pin 23 (TRIN)	→default=0

And function:

0 = input pin for **adigin** readout only
1 = stop function (also refer to **stoppol** and **adigintyp**)
2 = snapshot function (also refer to **sns1** and **adigintyp**)

The stop input function is a software stop. It can be set to a variety of behaviors as described in **stoppol**.

It is possible to assign the same function to several inputs.

By default, SnapShot is assigned to the SnapShot input and stop to the stop input (or for TANGO 3 mini: not assigned).

Response: Currently selected input pin function(s)

Remarks: If selecting a stop or snapshot function for one or several inputs, please ensure that the correct pull-up/pull-down resistor is set by **adiginfunc** and the correct polarity is chosen by **stoppol** or **sns1**.

When connecting the MW liquid dispenser to the AUX mini port, please ensure that the SnapShot function is assigned to the correct pin, else the **drop** counter will not work correctly.

Examples:

```
!adiginfunc 1 2      Set a single input by specifying bit and function, here:
                    Assign snapshot function (2) to input 1
!adiginfunc 0 1 0 2  Set all 4 input functions: input 0 as normal input (0),
                    input 1 as stop input (1), input 2 as input
                    and input 3 to snapshot function (2).
!adiginfunc -1      Reset all inputs to their default function
?adiginfunc → 0 0 1 2 Returns the function of all digital inputs
?adiginfunc 3 → 2   Returns the function of input 3 (here: SnapShot)
```

18.15. adigout (AUX I/O Digital Output)

Syntax: !adigout or ?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 bit number and signal level:

 0 = Bit 0: AUX I/O Pin 5 (TAKT_OUT, default LED100 on/off pin)
 1 = Bit 1: AUX I/O Pin 6 (VR_OUT)
 2 = Bit 2: AUX I/O Pin 7 (SHUTTER_OUT)
 3 = Bit 3: AUX I/O Pin 8 (TRIGGER_OUT)

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

TANGO 3 mini: 0 = Bit 0: AUX mini Pin 6 (TAKT_OUT, def. LED100 on/off pin)
 1 = Bit 1: AUX mini Pin 7 (VR_OUT)
 2 = Bit 2: AUX mini Pin 8 (SHUTTER_OUT)
 3 = Bit 3: AUX mini Pin 9 (TRIGGER_OUT)

Remarks: Some outputs might be occupied when the **trigger** is activated.
 Shutter out can be controlled by the **shutter** instruction also.
 In manual LED control mode (**hdimode** bit 2), TAKT_OUT will be
 set high/low automatically depending on brightness (0%=high).

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: output 0 to high, 1 to low,
                  outputs 2 and 3 are left unchanged
!adigout 0      set digital output 0 to logic 0 (e.g. LED100 on)
!adigout 1      set digital output 0 to logic 1 (e.g. LED100 off)
!adigout 1 0    set digital output 1 to logic 0
!adigout 2 1    set digital output 2 to logic 1
?adigout        read back the level of all outputs (e.g. returns 0000)
?adigout 3      read back the level of output 3     (e.g. returns 0)
```

18.16. adigoutpreset (AUX I/O Digital Output Preset)

Syntax: !adigoutpreset or ?adigoutpreset
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

Availability: Only with TANGO Desktop HE.

Description: With a syntax same as the adigout instruction, adigoutpreset
 allows to specify the default power up state of the AUX I/O
 digital output pins of the TANGO Desktop HE.

 0 = Bit 0: AUX I/O Pin 5 (TAKT_OUT, default LED100 on/off pin)
 1 = Bit 1: AUX I/O Pin 6 (VR_OUT)
 2 = Bit 2: AUX I/O Pin 7 (SHUTTER_OUT)
 3 = Bit 3: AUX I/O Pin 8 (TRIGGER_OUT)

Response: Output state(s), 0 or 1

Examples: !adigoutpreset 1011

18.17. anain (Analogue Input)

Syntax: ?anain or anain
Parameter: c or v
 Channel number, refer to **Appendix A**

Description: Read analog signal values. The channels and options depend on hardware and are different for each TANGO controller type. Channel 10 provides the AUX I/O analogue input signal value. Most channel values are returned in 10-bit digits, 0 to 1024, while others offer monitoring of supply voltages, currents, temperatures or HDI values such as e.g. joystick deflection. Using "anain v" instead of "anain c" returns two additional fractional digits that increase the resolution if the TANGO provides a higher resolution ADC than 10 bits, e.g. 12. If not available, "anain v" returns the same as "anain c".

Remarks: A complete anain channel list for each TANGO controller can be found in **Appendix A – anain options of different TANGOs**.

For monitoring the supply Voltage, Firmware 1.73 and higher offer direct readout of the supply Voltage ("motor voltage") through channel 28. In case of older Firmware versions, it can be calculated to Volts as follows:

- PCI-E, DT-E, TANGO mini, Pilot, TANGO 3 mini:
 $U_{\text{mot}}[\text{V}] = [\text{anain c } 12] * 0.05792$
- TANGO Integrale:
 $U_{\text{mot}}[\text{V}] = [\text{anain c } 12] * 0.03545$
- TANGO PCI, PCI-S, DT, DT-S: (PCI-E, DT-E is compatible)
 $U_{\text{mot}}[\text{V}] = [\text{anain c } 12] * 29.63 / [\text{anain c } 15]$
- To calculate the internal PSE voltage:
 $U_{\text{pse}}[\text{V}] = [\text{anain c } 11] * 7.819 / [\text{anain c } 15]$

Example:
anain c 10 => 510 Read channel 10 (AUX I/O analogue input pin 9) as 10 bit
anain v 10 => 509.75 Read value with higher resolution, e.g. 12 bit



18.18. anaout (Analogue Output)

Syntax: !anaout or ?anaout
 Parameter: 0 to 100 in percent (for anaout values)
 c or p (c = single channel keyword, p for preset)
 0, 1 or 2 (single channel number, when using c or p)

Description: Sets or reads back the AUX I/O analog output signal level in percent. It can be accessed either directly or by specifying an individual channel with the 'c' keyword (refer to examples)

Power-on presets can be specified to provide a certain output voltage after switching on the controller. The functionality can be accessed by the 'p' keyword and stored permanently by using **'save'**.

The signal resolution is 14 bit (100%/16384) with PCI-E and PCI-S based TANGO controllers and the TANGO 3 mini. Fractional numbers can be used for higher resolution.

100% corresponds to 10 Volts (TANGO 3 mini: 5 Volts).

Channel	Connector	Signal Name	TANGO AUX I/O Pin	TANGO 3 mini AUX mini Pin
0	AUX I/O	ANOUT0	10	11
1	AUX I/O	ANOUT1	11	-
2	reserved	-	-	-

Instructions: !anaout [level of anaout0] [optional also level of anaout1]
 !anaout c [channel no.] [level of specified anaout channel]
 !anaout p [channel no.] [preset value of specified channel]

Remarks: Channel 0 is used for brightness control of the LED100 illumination. In order to entirely switch off the LED, use **'adigout'**. Also refer to manual LED control via **hdimode**.

In case of TANGO 3 mini the output voltage is 0~5 Volts and only one output is available: ANOUT0 at AUX mini pin 11. For compatibility, ANOUT1 is kept as a dummy value which remains at 0.00%.

Response: Analogue output signal level(s) in percent

Examples:

!anaout 100 50.08 Set channel 0 = 100% (10V) and channel 1 = 50.8% (5.008V)

!anaout 75.4 Set channel 0 = 75.4% (7.54V)

!anaout c 1 25.3 Set channel 1 to 25.3% (2.53V)

!anaout p 0 10 Set channel 0 power-on preset value to 10% (1 Volt)
 save

?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)

18.19. anamode (Analogue Output Mode)

Syntax: !anamode or ?anamode
Parameter: 0 to 5

Description: Defines the behavior of the optional analog outputs, available with the AUX I/O connector. Older TANGO Firmware versions before 1.69 do not support storing this parameter. When using those older versions, anamodes different from 0 must be set each time after power up or reset if required.

```
0 : default mode - output controlled by '!anaout'  
    or by HDI devices in LED100 mode (hdimode)  
1 : ANOUT0 controlled by anain          (0~5Vin --> 0~10Vout)  
2 : ANOUT1 controlled by anain          (0~5Vin --> 0~10Vout)  
3 : ANOUT1 controlled by Z-axis position (0~10V out, 14bit)  
4 : ANOUT1 controlled by A-axis position (0~10V out, 14bit)  
5 : Z-axis is controlled by ANIN and TAKT_IN ('OLAF mode')
```

Remarks: Anamode is available with PCI-S/DT-S and PCI-E/DT-E TANGOs. The optional AUX I/O connector is required (output pin 11).

TANGO 3 mini: TANGO 3 mini supports anamodes 0 and 3 only. The 14bit output voltage range is 0~5 Volts.

Response: Selected mode as integer

Examples: !anamode 3 (set mode 3 to drive analog piezo stage with Z)
?anamode ==> 3

Anamode 3 or 4 - Controlling a Piezo Z-Stage

In anamodes 3 or 4, the analog output ANOUT1 follows the Z or A axis position. So it is possible to control a piezo stage through the 0-10 Volts output signal on the AUX I/O port, instead of having a stepper motor on the motor connector.

The 0-10 Volts output ANOUT1 then corresponds to the absolute piezo position.

The spindle **pitch** must be set to achieve the required travel of the piezo axis.

Example of a 0...10V piezo stage which travels 0.3mm at 10 Volts input signal:

```
!dim z 2  
!pitch z [stage travel in mm @ 10V]*12.5      (e.g. 0.3mm*12.5 = !pitch z 3.75)  
!lim z 0 [stage travel in mm @ 10V]          (e.g.                !lim z 0 0.3 )  
!anamode 3
```

Now the 10 Volts output corresponds to pos z = 0.3mm and the axis can travel between the absolute positions 0.0 and 0.3 mm (**moa**, **mor** etc. can be used).

Velocity and acceleration can be adjusted if required.

Dim 1 also works, but then lim must be specified in μm (here e.g. !lim z 0 300).

For anamode 4, replace z of the example above by a and set !anamode 4.

TANGO 3 mini has a 0-5V analog output signal.

This signal either has to be amplified externally to 0-10V or the travel range will be limited to 50% of the usual 0-10V range. In this case, the pitch for z must be doubled in order to achieve the correct travel distance within 0-5V.



Anamode 5 is a special mode that works in conjunction with a laser autofocus.

Available from Firmware 1.73, anamode 5 can not be saved. It must be enabled by "!anamode 5" each time to ensure to start from a know state (the laser is switched on, connected, and ready for operation).

The laser autofocus here delivers an analog signal of [0.0 ... 2.5 ... 5.0]V, where 2.5V means "in focus".

The Z-axis will travel in a constant loop to keep the input voltage at 2.5V as long as the digital "signal valid" information on the AUX I/O "TAKT IN" is at a logic 1 (high).

18.20. stoppol (Mode and Polarity of Stop Input Signal)

Syntax: !stoppol or ?stoppol

Parameter: 0 to 5 or 8 to 13

Description: Operating mode of the AUX I/O "Stop" input.
Signal polarity and behavior of the TANGO can be set:

**0,1 Stop only as long as stop signal is applied
HDI (joystick) remains active!**
0=active low, 1=active high

**2,3 Stop only as long as stop signal is applied
HDI (joystick) is also disabled**
2=active low, 3=active high

**4,5 Stop signal is latched (sticky) released by "!stop 0" only
HDI (joystick) is also disabled**
4=active low, 5=active high

6,7 Not available

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

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

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

Modes 16 and 17 by TANGO Desktop HE only:

16,17 Only disable HDI (joystick) while stop signal is present
16=active low, 17=active high

Requirements: A stop signal must be applied for at least 50µs.

***) Stoppol modes 8-13 (all, including the latched modes):
In order to stop, the stop signal must remain active until
all axes have completed their currently running move
instruction. If the stop signal is removed while an axis
is still traveling, no stop will be performed.**

TANGO 3 mini: Has no dedicated stop input. It can be assigned by **adiginfunc**.

Remarks: Usually the stop input has an internal pull-up resistor
to +5V, while TANGO 3 mini and Desktop HE offer additional
adigintyp and **adiginfunc** options.

If closed loop 'ctrsm' mode is set to 4, an error condition
of the closed loop will modify the stoppol to a latched mode
by internally oring the current stoppol with 4 (0-->4 etc.).

Sending "**!stop 1**" in latched stop modes immediately applies a
stop, even in modes 10-13.

TANGOs without a stop input may use the latched stoppol modes
in conjunction with manually sending "!stop 1" and "!stop 0".

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

Example:

```
!stoppol 5      Set the AUX I/O stop input to latched stop active high.
!stoppol 13     Same as above, but a currently running move is completed
                first and only the following moves will be suppressed.
?stoppol => 0   Current stoppol mode is mode 0 (default)
```

18.21. stop (Release, Force or Check Stop Condition)

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

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

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

Remarks: in `stoppol` modes 12 and 13, a forced stop by sending "`!stop 1`" stops the axes immediately (same as in `stoppol` modes 4,5).

Response: Internal stop state of the controller (0, 1)

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

18.22. stopl (Latched AUX I/O caused Stop Condition)

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

Availability: Only with TANGO Desktop HE.

Description: Latched AUX I/O stop condition (not if caused by "`!stop`"). To check if a stop condition was active due to the stop input. Can be used if e.g. a move ended or instant closed loop did not switch on because there was a (temporary) external stop condition which might not be active anymore.

The latched state can be cleared by "`!stopl 0`".

Response: Latched stop state from the AUX I/O stop input signal.

Example: !`stopl 0` clear the stop latch to 0
 !`stopl 1` set the stop latch (makes no sense)
 ?`stopl` read if a stop condition was caused from AUX I/O

18.23. shutter (Shutter Out Signal of AUX I/O)

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

Description: Set the AUX I/O 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

18.24. flash (Defined Pulse at AUX I/O Takt Out)

Syntax: flash or !flash

Parameter: +-0.00001 ... 32500 [ms]

Availability: TANGO PCI-S/DT-S, PCI-E/DT-E TANGO 3 mini and Desktop HE.

Description: Sends a pulse of defined length to the AUX I/O TAKT_OUT pin.
Used e.g. for LED strobes.

Floatingpoint numbers in [ms].
Range 0.00001 (10ns) to 32500 (32.5s).
Resolution is 1/132µs.

Pulse Polarity depends on sign:

- Positive numbers generate an active high pulse
- negative numbers generate an active low pulse

For safe operation it is recommended to once send one dummy pulse when initializing in order have the correct polarity.

Remarks: Might interfere with secondary trigger output(refer to **trigo**).

Response: None

Example:

```
flash 0.001      (high pulse with duration of 1µs)
flash -0.01     (low pulse with duration of 10µs)
```

18.25. tvr (Pulse and Direction Input Function)

Syntax: !tvr or ?tvr

Parameter: x, y, z, a or none
0, 5

Availability: TANGO PCI-E, DT-E and Desktop HE only.

Description: Set the TVR Mode, used to drive an axis through an externally applied pulse and direction signal.

0 = disabled
(1) = ~~enabled without tvrf factor~~
(2) = ~~enabled with tvrf factor~~
(3) = ~~enabled without tvrf factor, requires ext. start/stop~~
(4) = ~~enabled with tvrf factor, requires ext. start/stop~~
5 = enabled with tvrjoyf factor

Response: Currently selected tvr mode(s)

Example:

```
!tvr 0 0 0 0    (disable tvr on all 4 axes)
!tvr z 5        (enable tvr mode 5 for Z axis)
?tvr z          (read tvr mode of Z axis)
?tvr            (read tvr mode of all axes)
```

Example for TVR via AUX I/O standalone operation:

```
!tvrjoyf 1.0
!tvr z 5
!tvrjoy z
```

18.26. brake (Axis Brake Function)

Syntax: !brake or ?brake
Parameter: x, y, z, a or none
0,1...8 or -255...0 (for IO Brake) and 0 or 1 (Onboard Brake)

Availability: TANGO PCI-E/Desktop-E, TANGO Desktop HE, TANGO 3 mini.
The availability of the brake function depends on order.
A) The **TANGO PCI-E/Desktop-E** provides a brake function via optional IO1 or Multi-IO extension module outputs OUT0-7.
B) The **TANGO 3 mini** has one brake pin on its motor connector (only for 3 mini: always available). The instruction syntax is identical, except only having 0 or 1 ("OUT0").
C) The **TANGO Desktop HE** allows both, either a sophisticated onboard brake over a motor connector pin or a brake via IO1 and Multi-IO extension modules, for compatibility to existing PCI-E/Desktop-E applications or several brakes.

Description: Function for electrically released brakes, intended to hold axes if their motor loses its torque (due to switched off axes, amplifiers, undervoltage, PSE or current reduction).

Motor torque is on : Output = high (24V to release brake)
Motor torque is off: Output = low (0V, brake closes)

From TANGO Firmware 1.70, a motor **current reduction** to <30% also leads to activating the brake. Also, when ramping up a current after power-on, the brake is released from >=30%. Setting a low '**cur**' value does not activate the brake.

As output "low" is the safe brake state, when sharing outputs with several axes the output state zero is dominant.

IO Brake: Individual, combined, or shared IO outputs can be selected for each axis.

Each axis can be applied to an output by integer values 1~8. Negative values can be used to apply several output pins or combine/share pins - even with other axes. They represent a bit mask of 8 bit (0x00~0xFF) as integer values 0 to -255.

I/O:	[OUT7]	[OUT6]	[OUT5]	[OUT4]	[OUT3]	[OUT2]	[OUT1]	[OUT0]
PIN:	8	7	6	5	4	3	2	1
BIT:	-128	-64	-32	-16	-8	-4	-2	-1

0 = brake function disabled (default)
1 = apply brake to 1st IO pin (OUT0) *PIN specified*
2 = apply brake to 2nd IO pin (OUT1) *PIN specified*
3,4,5,6,7 ... (OUT2,3,4,5,6) ...
8 = apply brake to 8th IO pin (OUT7) *PIN specified*

-1 = apply brake to 1st IO pin (OUT0) *BIT specified*
-2 = apply brake to 2nd IO pin (OUT1) *BIT specified*
etc. ...
-17 = apply brake to 1st and 5th pin (OUT0 + OUT4) *BITs*

If the TANGO provides both IO1 and Multi-IO extension modules, the Multi-IO is used for the brake functionality.

The assigned pins cannot be accessed by IO write instructions !**digout**, !**edigout**. But reading the digital output state with ?**digout**, ?**edigout** does return the state of the brake output.



Onboard Brake: Only one output. One or multiple axes can be selected to activate the brake pin on the 3-axis motor connector. TANGO Desktop HE provides a brake module, where the brake voltage for activating and holding the brake and the delay between it can be configured via 'vbrake' and 'brakedelay'.

"?brake 1" reads the state of the onboard brake output.
Here: 1=brake active (closed), 0=brake inactive (open).

Response: Integer value(s), 0 or -255 to 8 (Onboard Brake: 0s or 1s)

Examples:

```
!brake z 1          Enable brake for Z axis on OUT0 or Onboard Brake
!brake 0 0 1       Enable brake for Z axis on OUT0 or Onboard Brake

!brake 1 3 8 0     Enable brakes for XYZ axes on different IO-pins, disable A
!brake 4           Enable brake for X axis on 4th IO-pin (OUT3)
!brake -17 -18 -20 Enable brake for XYZ on IO-pins 123, create XYZ common pin 5

?brake            Read brake setting for all axes (e.g. returns 0 0 0)
?brake z         Read brake setting for Z axis only (e.g. returns 0)

?brake 1 => 0     Read the state of the Onboard Brake (here: inactive/open)
!brake z         Assign the Onboard Brake to only Z (others will be removed)
```

18.27. vbrake (Axis Brake Voltage)

Syntax: !vbrake or ?vbrake
Parameter: 4.0...24.0 Volt (to open)
4.0...24.0 Volt (to hold open, optional 2nd parameter)

Availability: TANGO Desktop HE with Onboard Brake only.

Description: Set the voltages for opening and for holding the brake open. The default setting is 24V for both, opening and holding.

When using one parameter, it applies to both, open and hold.

The time delay between opening and holding voltage can be adjusted by !brakedelay.

As the usual brake state is "open" (powered), the brake introduces heat into the system. The heating can be reduced by lowering the brake holding voltage. Possibly also the opening voltage can be reduced or has to be in case not 24V. Similar to motor current reduction, lowering the brake hold voltage to 70% would reduce power and heat to 1/2 (50%). Please refer to the brake data sheet to ensure safe function.

Remarks: The minimum voltages (of an individual brake) can be found by

- setting "!vbrake 4" (minimum voltages)
- activating the brake, e.g. by "!brake x" (so it would open)
- increase "!vbrake 5", 6, ... until the brake opens (e.g. 16)
- reduce !vbrake again until the brake closes (e.g. 10)
- now the absolute minimum voltages are known (here 16 and 10)
- probably decide to set "!vbrake 22 14"

Response: Currently selected vbrake voltages (for opening and holding)

Example: !vbrake 24
?vbrake => 24.0 24.0
!vbrake 22 13.5
?vbrake => 22.0 13.5

18.28. brakedelay (Axis Brake Hold Voltage Delay)

Syntax: !brakedelay or ?brakedelay
Parameter: 1 ... 10000 [ms]

Availability: TANGO Desktop HE with Onboard Brake only.

Description: Time delay before applying the hold voltage to the brake. Allows the brake to safely open with the opening voltage before switching to the (lower) hold voltage (see **vbrake**). The default value of 100ms should be sufficient for most brakes, please refer to the brake data sheet.

Response: Delay time etween opening and reducing to hold voltage in [ms]

Example: !brakedelay 30
?brakedelay => 30

18.29. brakepos (Move to Initial Motor Pole Position)

Syntax: !brakepos or brakepos

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

Availability: TANGO PCI-E/Desktop-E, TANGO Desktop HE, TANGO 3 mini running firmware 1.70 or greater.

Description: Drives the motor to the nearest motor pole, which also is the initial position after a power-on or reset of the TANGO.

If brakepos is executed before power down or before activating the motor brake, it avoids a "jump" of the motor at power up. This jump can be within ± 2 motor steps, depending on where the axis was positioned before.

While this jump usually is not an issue, it can cause a problem on Z-axes with heavy load: When the **brake** is released, this jump, under certain conditions, can cause the motor to stall and the axis then runs down until its mechanical limit.

So for heavy loaded Z-axes with attached motor **brake**, the brakepos instruction can be used to prevent the axis from running down at power-up (when the **brake** is released).

The brakepos instruction can only be executed for individual axes, so the axis specifier **x,y,z or a is required**.

There also is an **optional parameter** 1, -1 or none, which can be used to specify the travel direction in which brakepos is executed:

[none] = positions the motor to the nearest pole
positive or negative rotation **

1 = positions the motor to the next pole
in positive direction **

-1 = positions the motor to the next pole
in negative direction **

** The parameter can be used to ensure the axis is traveling forward (or backward) only, to avoid possible collisions.

If the axis is standing within a limit switch, the direction is set to "out of the switch" automatically, independent of the direction parameter.

The normal axis **velocity** and **acceleration** is used.
For a typical motor with 200 steps, the travelled distance is less than $4/200 = 0.02$ revolutions or 7.2 degrees.

Response: none (but brakepos blocks all following instructions until the move is completed, usually within milliseconds)

Examples:

brakepos z => Travel to the nearest motor pole in Z

brakepos z 1 => Travel forward to the next motor pole in Z...
?err => and wait for executipon by using the ?err reply

brakepos z -1 => Travel backward to the next motor pole in Z...
!pa0 => then turn off the amplifiers without waiting, as brakepos blocks the !pa 0 instruction until it has completed.

18.30. drop (Liquid Dispenser – Generate Drops)

Syntax: !drop or ?drop
Parameter: 0, 1, ... 6000

Description: Used with the Märzhäuser Liquid Dispenser.
Generates drops or reads back the amount of generated drops.

The instruction supports two dispenser types:

1) For upright microscopes

Drops are generated and counted.

2) For inverted microscopes

Liquid is dispensed for the specified amount of seconds or 1/10 seconds (depends on dispenser setup). Time is counted.

!drop 0 resets the counter to zero

!drop N generates *N* drops (or *N* seconds), *N*=1...6000

?drop 0 reads amount of drops still to be generated by the recent !drop instruction, this countdown can be used to identify the drop instruction has finished (=0). Here it's not required to reset the counter.

?drop reads amount of drops/time since counter was resetted

Remarks: **Only available with TANGO PCI-E, TANGO DT-E and TANGO 3 mini.**
Might interfere with trigger and snapshot functionality.

TANGO 3 mini: The drop counter is connected to the AUX I/O SnapShot input.
In case of TANGO 3 mini, the snapshot pin assigned by **adiginfunc** is used.

Response: Amount of counted drops or time, depending on the device.
The drop counter counts to a maximum of 65535 then rolls over.

Example:
!drop 10 generate 10 drops, or dispense liquid for 10 seconds,
 (depends on hardware)

?drop 0 => 7 (still 7 drops or seconds to generate, not finished yet)
?drop 0 => 0 (all drops have been generated, ready)

!drop 0 reset the drop counter for ?drop
?drop read the drop counter (counting since it has been resetted)

18.31. pump (Liquid Dispenser – Manually Add Air Pressure)

Syntax: !pump or ?pump

Parameter: 0, 1

Description: Used with the Märzhäuser Liquid Dispenser. Manually overwrites the air pressure pump. Can be used to ensure sufficient pressure before dispensing liquids in inverted applications (time interval mode). The dispenser switches the pump on and off automatically, but in case of the time interval dispenser it might be safer to ensure sufficient pressure before dispensing by this instruction. For generating drops, it might not be required to ensure pressure because they are counted. It then just may take longer.

1 = Air pressure pump on

0 = Air pressure pump off

It is ensured by design that the maximum pressure won't be exceeded.

Remarks: **Only available with TANGO PCI-E, TANGO DT-E and TANGO 3 mini.** Might interfere with trigger and snapshot functionality.

Response: 0 (pump is off), 1 (pump is on)

Example:

!pump 1 switch pump on

!pump 0 switch pump off

?pump => 0 (pump is off)

18.32. vbus (+24V Supply Output On/Off)

Syntax: !vbus or ?vbus
Parameter: 0, 1

Availability: Only with TANGO 3 mini and TANGO Desktop HE.

Description: On/off state of the +24V power output.
A 24V supply can be provided for external components.
For TANGO 3 mini, the +24V is output on the AUX mini port,
For TANGO Desktop HE it is output from the CAN Bus connector.

1 = +24V on
0 = +24V off

Remarks: After power up or reset, the +24V is not provided on the AUX mini connector by default. It must be switched on by !vbus or configured to always on by 'configvbus'.

Response: 0 (+24V is off) or 1 (+24V is on)

Example:
!vbus 1 switch +24V on
!vbus 0 switch +24V off
?vbus => 0 Read on/off state (here: +24V is off)

18.33. configvbus (+24V Supply Output Preset)

Syntax: !configvbus or ?configvbus
Parameter: 0, 1

Availability: Only with TANGO 3 mini and TANGO Desktop HE.

Description: Sets the default state of the +24V supply output after power up or reset.
A 24V supply can be provided for external components.
For TANGO 3 mini, the +24V is output on the AUX mini port,
For TANGO Desktop HE it is output from the CAN Bus connector.

1 = +24V on after power up
0 = +24V off after power up (default)

Remarks: The +24V pin might not powered by default. To change this behavior, configvbus can be used. The +24V can also be switched on and off temporarily during operation by the 'vbus' instruction.
The momentary on/off state can be checked by '?vbus'.

Warning: It is not recommended to plug or unplug the AUX mini connector when the 24V is on. It could lead to damage of 5V electronics or of the RS232-Tx driver.

Response: 0 or 1

Example:
!configvbus 1 +24V on after power up (and +24V is immediately switched on)
!configvbus 0 +24V off after power up (+24V is not immediately switched off)
?configvbus => 0 (+24V is configured to off)

18.34. configcanres (USB Host +5V Supply Output Preset)

Syntax: !configcanres or ?configcanres
Parameter: 0, 1

Availability: Only with TANGO Desktop HE.

Description: Optional activation of the onboard CAN terminating resistor.
0 = 120Ω termination off (default)
1 = 120Ω termination on

Response: 0 or 1

Example:

```
!configcanres 1          Enable onboard CAN Bus termination (120Ω load)
!configcanres 0          Disable onboard CAN Bus termination
?configcanres            => 0  Read CAN termination state (here: 0 = off/disabled)
```

18.35. vusb (USB Host +5V Supply Output On/Off)

Syntax: !vusb or ?vusb
Parameter: 0, 1

Availability: Only with TANGO Desktop HE.

Description: On/off state of the 5V power output of the USB Host interface.
1 = +5V on (default)
0 = +5V off

Remarks: The 5V is of the USB Host (USB-A) switched on by default.
The default behavior can be changed by '**configvusb**'.

Response: 0 (+5V is off) or 1 (+5V is on)

Example:

```
!vusb 1                  switch +5V on
!vusb 0                  switch +5V off
?vusb                    => 0  Read on/off state (here: +5V is off)
```

18.36. configvusb (USB Host +5V Supply Output Preset)

Syntax: !configvusb or ?configvusb
Parameter: 0, 1

Availability: Only with TANGO Desktop HE.

Description: On/off state of the 5V power output of the USB Host interface after power up or reset.
1 = +5V on (default)
0 = +5V off

Response: 0 (+5V is off after power up) or 1 (+5V is on after power up)

Example:

```
!configvusb 1           Set power up default for the USB Host +5V to ON
?configvusb             => 1  Read power up default for the USB Host (here: +5V ON)
```

19. Encoder Instructions

The encoder interface supports incremental 1Vpp, 5Vpp MR and RS422-TTL encoders. The encoder type can be selected by the **enctype** instruction. Most TANGO encoder interfaces support reference marks (except TANGO mini 2-axis). TANGO Desktop HE controllers additionally support **absolute encoders** with BiSS-C or SSI interface and optional 1Vpp analog signal.

(Please also refer to the encoder interface description of the TANGO controller for further information: TANGO mini 2-axis and TANGO integrale have limitations in travel velocity when using 1Vpp or RS422 TTL encoder signals. For RS422 TTL, those even require a hardware change. Single ended TTL encoders always require additional circuitry.)

To enable encoder functionality, the encoder mask **encmask** must be set for the corresponding axes. If **encmask** is set, the encoders are activated after calibration **cal** or at power-up, depending on the selected **calmode**. This also enables the selected Closed Loop mode (set by **ctr**).

calmode also offers mode 3, which only reads out measuring systems.

The **enc** state is internally set to 1 when the encoders are activated. Manually setting the encoders **enc** state to 1 is not recommended, as it might cause unpredictable behavior in closed loop mode (due to counting direction, position alignment, signal validation the TANGO controller applies internally). Also, in case of analog MR encoders, the signal correction will not be applied, which leads to positioning errors.

Reference marks can be activated by **encref**, the search velocity sets **encrefvel**.

Encoders that provide an active low error signal can be supported by **encnas**.

19.1. encmask (Encoder Mask)

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

Description: Reads or sets the encoder globally enable mask, which is required to activate the encoders (**enc-->1**).

The encoders then will be detected and activated after

- A) a successful calibration instruction '**cal**'
 or
- B) after power up, when **calmode** 2 or 1 is selected

0 = clear enable mask (encoder will be ignored, not activated)
1 = set enable mask (TANGO will try to activate the encoder)

Response: Encoder enable mask as 0s and 1s

Example:
!encmask 1 1 0 Globally enable encoders for X, Y and disable Z-axis
!encmask z 0 Globally disable encoder for Z-axis
?encmask Read encoder mask state of all axes (e.g. 1 1 0)

19.2. enc (Encoder Active)

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

Description: Query if the encoders are active (successfully activated by a **cal** instruction or at power up in **calmode** 2, 1 or 4). **enc** is activated by the TANGO through **cal** or the calmodes. It is not recommended to manually activate the encoders by sending a "!enc 1" instruction, because the counting direction might be wrong and because some encoders (as MR 5Vpp) require a dedicated calibration procedure or signal check. 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**. Please refer to the above-mentioned instructions and to the remarks below.

0 = Encoder is inactive (not used)
1 = Encoder is active (used)

Response: Encoder active state

Example:

```
?enc           Read encoder active state of all axes (e.g. 1 1 0 for 3 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 the X-axis encoder)                   **
```

Remarks: ** Manual activation of the encoders (by !enc 1) is not recommended. In general, this is not recommended if the axis runs in closed loop. In case of MR encoders a special calibration procedure has to be performed, which is only available by !cal or instant closed loop (**calmode** 2 or 1). An exception can be if the axis is only used for measuring purposes and a TTL or 1Vpp encoder is attached (no MR).

If the application requires to disable and enable the encoders during operation (e.g. !enc 0 0 ... !enc 1 1), it is possible to check if the encoder once was successfully activated by cal or instant closed loop - even if it is now set to zero: ?enc 1 / ?enc x 1 will return a 1 if the encoders once were activated by cal or instant closed loop and no error occurred. This option is available since firmware 1.71.

Examples:

```
[power on or reset]
?enc   => 0 0     (the encoders are off, not activated)
?enc 1 => 0 0     (encoders never were successfully activated)
```

```
[power on or reset]
!cal x => A@--.   (here: the X-encoder gets activated by cal)
?enc   => 1 0     (the X-encoder is on)
?enc 1 => 1 0     (the encoder has been activated (by cal))
!enc 0 0         (the application forces the encoders off)
?enc   => 0 0     (the encoders are off)
?enc 1 => 1 0     (the X-encoder once was activated (by cal))
!enc x 1         (it is okay to manually switch it on again)
?enc   => 1 0     (the X-encoder is on again)
```

19.3. encperiod (Encoder Signal Period)

Syntax: !encperiod or ?encperiod

Parameter: x, y, z, a or none
0.000002 to 4.0 [mm]

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

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

Response: Encoder signal period(s)

Example:

```
!encperiod 0.5 0.5 0.001    Set encoder period for X and Y to 500µm, Z to 1µm
!encperiod z 0.02          Set encoder period of Z-axis to 20µm
!encperiod 0.00001960784   Set encoder period of X-axis
?encperiod                 Read encoder period of all axes
?encperiod z               Read encoder period of Z-axis
?encperiod 12              Read period of all axes with 12 fractional digits
?encperiod z 9             Read period of Z-axis with 9 fractional digits
```

19.4. encdir (Encoder Counting Direction)

Syntax: !encdir or ?encdir

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

Remarks: Setting the encoder direction is not required and should never be changed by this instruction, as the TANGO identifies the correct counting direction itself.
The only exception might be if the connected encoder is used as a truly independent measuring system, and never by the TANGO for closed loop operation.

Description: Set or read the encoder counting direction.
Do not set this parameter when the TANGO is in closed loop!
The encoder direction is set by the TANGO automatically (e.g. after calibration **cal** or instant closed loop during power-on).

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
```


a

19.5. 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
```

19.6. encrefvel (Encoder Ref.-Signal Calibration Velocity)

Syntax: !encrefvel or ?encrefvel
Parameter: x, y, z, a or none
0.000001 to 3000 [rev/s] (or [mm/s] if dim = 9 or 10)

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

Response: Currently used encrefvel in [rev/s] or [mm/s] in dim 9+10

Examples:

```
!encrefvel 5 5 5 Set encoder reference mark search velocity of X, Y, Z to 5
!encrefvel z 0.5 Set encoder reference mark search velocity for Z axis to 0.5
?encrefvel Read ref mark search velocities of all axes
?encrefvel y Read ref mark search velocity of Y axis only
```

19.7. `enctype` (Encoder Type Configuration)

Syntax: `!enctype` or `?enctype`

Parameter: `x, y, z, a` or `none`
0 to 2 or 6 (depends on hardware)

Remarks: Most TANGO controllers, from Firmware 1.60 or higher, provide and support a Universal Encoder Interface. This interface can be configured by software to support 5Vpp MR, 1Vpp or digital RS422 incremental encoders. In order to provide the new features, the `encttl` instruction was replaced by `enctype` (`encttl` should not be used anymore).

TANGO Desktop HE provides an Absolute Encoder Interface, which extends the functionality of the Universal Encoder Interface by options for BiSS-C or SSI absolute encoders with or without additional analog 1Vpp signal.

Description: The instruction reads or sets the encoder signal type.

0 = MR 5Vpp analog sin/cos interpolation
1 = TTL RS422 A/B digital incremental signal
2 = 1Vpp analog sin/cos interpolation

TANGO Desktop HE extends the options by absolute interfaces:

3 = Absolute BiSS-C
4 = Absolute BiSS-C with 1Vpp sin/cos analog
5 = Absolute SSI
6 = Absolute SSI with 1Vpp sin/cos analog

The read instruction provides two options:
By using an additional parameter "1" with the read instruction, the effectively by hardware applied encoder type is returned. This might happen when e.g the encoder option was not purchased and the TANGO degrades the interface to TTL counting only. When sent without the parameter 1, the instruction returns the selected encoder type.

Remarks: If digital encoders (A/B-TTL, RS422) are used and the interface by mistake is programmed to an analog signal mode, this can cause a sporadic malfunction (encoders become deactivated) due to analog signal monitoring.

Absolute encoders require additional parameters:
The number of bits (`encform`) and resolution (`encres`).
If the absolute encoders provide an additional 1Vpp signal, the `encperiod` must be set, else not required.

Response: 0 to 6 (or -1 = not available, invalid configuration)

Example:

```
!enctype 1 0 2      Set X encoder to A/B-TTL, Y to MR and Z to 1Vpp
!enctype x 1       Set X encoder to A/B-TTL, leave others unchanged
?enctype           => 1 0 2   Read the requested encoder types of all 3 axes
?enctype 1        => 1 1 1   Read the applied encoder types (e.g. TTL only)
?enctype z        => 2       Read the requested encoder type of the Z axis
?enctype z        => 2       Read the applied encoder type of the Z axis
?enctype a        => 0       Read the requested encoder type of the A axis
?enctype a 1     => -1       Read the applied encoder type of the A axis
```

19.8. encctl (Encoder Configured for TTL Signal)

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

Remarks: In newer versions of the TANGO Controller - with Universal Encoder Interface and Firmware 1.60 or higher - this instruction is replaced by '**enctype**' and therefore encctl should no longer be used, except for compatibility.

Description: This instruction reads or sets the type of encoder signal. If digital encoders (A/B-TTL, RS422) are used with an analog encoder interface (configured for 1Vpp or 5Vpp MR), the corresponding encoder has to be set to TTL mode. Else the TTL signal will be found as invalid (due to signal level) and not be used or deactivated during operation (sporadic malfunction).

0 = Analog sin/cos encoder (1Vpp or MR is defined by hardware)
1 = Digital quadrature A/B encoder (e.g. RS422)

Response: Currently selected encoder signal type(s)

Example:

```
!encctl 0 0 1 Set X and Y axis encoders to analog, Z to digital A/B-TTL
!encctl z 1 Set Z encoder type to digital
?encctl Read encoder signal type of all axes
?encctl x Read encoder signal type of X-axis
```

19.9. encref (Use Encoder Reference Signal)

Syntax: !encref or ?encref
Parameter: x, y, z, a or none
0, 1, 2 or 3

Description: Enable or disable the use of encoder reference marks. If enabled, the '**cal**' instruction will - after reaching the lower limit switch - travel to the reference mark and set the axis position to zero.

In case of MR encoders, there is an option to store the sin/cos signal constellation outside the limit switch (done by factory through 'callrn') which greatly improves accuracy and repeatability of the origin without having a reference mark. Therefore, encref mode 2 was introduced.

0 = Encoder reference signal not used
1 = Encoder reference signal used for calibration
2 = MR signal period used as reference (set by callrn)
3 = Motor step pos used as reference (set by callrn)

Remarks: The velocity towards the reference mark is set by **encrefvel**.

Response: 0 ... 3

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 (e.g. 1 1 0)
```

19.10. 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, an encoder NAS error generates an internal '**encerr**' error state. The NAS input is active low.

0 = NAS encoder input state is ignored (default)
1 = NAS encoder input signal is used for error detection

Remarks: The NAS signal state can be read by '**encnasstatus**'.

Response: Encoder NAS signal used / not used for error detection

Example:
!encnas 1 1 0 Use encoder NAS signal for X and Y-axis, ignore for Z
!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

19.11. encrefstatus (Encoder REF Signal State)

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

Description: Returns the REF signal input state.

0 = REF signal is inactive
1 = REF signal is active (encoder is on a reference mark)

Response: Encoder reference signal state

Example:
encrefstatus Read REF signal state of all axes
encrefstatus x Read REF signal state of X-axis only

19.12. 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 encrefstatusl, a 1 is returned. The corresponding latch state(s) are cleared after reading.

0 = REF signal is and was inactive since last read
1 = REF signal is/was active (encoder is or 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

19.13. 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. (This signal is usually ignored, but can be enabled by **encnas** as error source)

0 = NAS signal is inactive (High=the encoder reports no error)

1 = NAS signal is active (Low =the encoder reports an error)

Response: Encoder NAS error signal state (=inverted NAS input pin level)

Example:

encnasstatus Read NAS signal (error) state of all axes

encnasstatus x Read NAS signal (error) state of X-axis only

19.14. 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, e.g. a low **encamp** signal amplitude or active NAS error signal from the encoder '**encnas**'+'**encnasstatus**', the encoder signal is invalid and the closed loop for the corresponding axis is switched off.

0 = No error, normal function

e = Encoder error

Response: Encoder error state, 0 or e

Example:

!encerr 0 Reset encoder error

?encerr Read encoder error states of all axes, e.g. "0 0 e" for 3 axes

?encerr x Read encoder error state of X-axis only

19.15. encamp (Encoder Signal Amplitude)

Syntax: ?encamp or encamp

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

Description: Read the encoder signal amplitude.
100 (percent) represents the maximum undistorted signal amplitude, for 1Vpp this is 1.2V.

Remarks: Differential TTL encoders return about 140%.
Single ended TTL encoders, due to the necessary input circuit modifications, might return around 0%.

In case of absolute encoders without an additional 1Vpp signal, the amplitude is fixed and reflects the error and warning state of the data:

No error, no warning = 67 [%] (the desired state)

Warning = 40 [%] (still working)

Error and warning = 0 [%] (error)

Response : Encoder signal amplitude in percent
as integer or with 1 fractional digit

Example:

?encamp Read all encoder signal amplitudes (returns e.g. 57 74 0)

?encamp x Read X encoder signal amplitude

?encamp 1 Read all amplitudes with 1 fractional digit (57.3 73.8 0.5)

?encamp x 1 Read X encoder signal amplitude with 1 fractional digit (57.3)

19.16. encpos (Encoder Position)

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

Description: The position "source" for the **?pos** instruction.
 Only affects the readout position values of pos and sns. If encpos is set to 1 and the encoder '**enc**' is activated, **pos** returns the encoder position, else the motor position.

Remarks: For compatibility, sending a single 0 or 1 without specifying an axis applies the setting to all axes.
enc is activated by the TANGO through 'cal' or the calmodes, not by the user.

The setting of encpos is volatile, but from Firmware 1.72, a power-on preset setting can be stored by '**configencpos**'.

Response: Position source
 0 = pos instruction returns motor position (default)
 1 = pos instruction returns encoder position (if enc. Active)

Example:
 !encpos 1 a 'pos' instruction will return the encoder position for all axes (if the axes encoders are active)
 !encpos 0 0 1 encoder position is enabled for Z and disabled for X and Y
 !encpos x 1 'pos' will return the encoder position for the X-axis
 ?encpos Use of ?encpos is not recommended. Just for compatibility, this instruction reads the "ored" position source of all axes (it returns just one 0 or 1, it returns a 1 if at least one axis has encpos enabled)
 ?encpos x Read position source of the X-axis
 !encpos 0 1 0 Set individual axes
 → ?encpos -1 => 0 1 0
 → ?encpos => 1 Compatible readout (?encpos only for backw. compatib.)
 → ?encpos z => 0

!encpos 1 One sets all
 → ?encpos -1 => 1 1 1

Sequence example of a 3 axis TANGO:

```
?calmode => 0 0 0
!encpos 1 1 0 (pos should return the encoder position in X and Y)
?enc => 0 0 0
?pos => [motor pos] [motor pos] [motor pos]
!cal x => @@@-.
?enc => 1 0 0
?pos => [encoder pos] [motor pos] [motor pos]
!cal y => @@@-.
?enc => 1 1 0
?pos => [encoder pos] [encoder pos] [motor pos]
```

19.17. configencpos (Configure Encoder Position Preset)

Syntax : !configencpos or ?configencpos

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

Description: Set or read the predefined power-up setting for **encpos**. The TANGO will use it as preset value for **encpos** after power-up or reset.

Remarks: As the **encpos** setting is volatile and commonly used, e.g. by SwitchBoard, it should not be storable. In cases where encpos must be activated at power-up, e.g. standalone applications without a PC, configencpos can be used. The function is available from Firmware 1.72.

Setting "!configencpos" also sets the current encpos.

Response: Position source
0 = pos instruction returns motor position (default)
1 = pos instruction returns encoder position (if enc. Active)

Example:

```
!configencpos 0 0 1    Set power-up preset for encpos to X,Y=motor, Z=encoder
!configencpos 1        Set power-up preset for encpos to X=encoder
!configencpos z 0      Set power-up preset for encpos to Z=motor
?encpos                Read the encpos preset values of all axes
?encpos y              Read the encpos preset values of the y axis
```

Sequence example of a 3 axis TANGO:

```
?encpos -1            => 0 0 0    after power-on, the encpos is 0
!configencpos 0 0 1    configuring the !configencpos...
?configencpos         => 0 0 1
?encpos -1            => 0 0 1    ...also changes the encpos...
!save                 ...and if saved...
!reset
?encpos -1            => 0 0 1    ...works as a preset for encpos.
```


19.18. encsync (Analog Encoder Synchronization Status)

Syntax: ?encsync or encsync
Parameter: x, y, z, a or none
-1 or none

Description: Returns the synchronization state of the encoder signal and the digital hardware counter.
Called without parameter, it returns if the encoder(s) are synchronized. When called with "-1", the quadrants of the analog and digital signal paths are shown and the applied correction (deviation) of the digital quadrant.
The deviation should never exceed ± 1 . This instruction is available from TANGO PCI-E/DT-E firmware 1.71 and above.

Response: none: Encoder synchronized = 1, not synchronized = 0
-1 : encoder signal analog, digital quadrants and deviation.
Quadrants are 0,1,2,3, the deviation is 0 or +1.
Please refer to the examples below.

Example:
encsync x → 1 (1 = X encoder synchronized, 0 = not synchronized)
encsync → 1 1 0 (3 axis TANGO, X and Y are synchronized, Z not)
encsync x -1 → 2 3 -1 (X encoder an.quad=2, dig.quad=3,quad.comp.d->a=-1)
encsync -1 → 2 3 -1 2 2 0 0 0 (response from a 3 axis TANGO X,Y,Z)

19.19. hwcount (Hardware Counter)

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

Description: Returns the positions of the independent quadrature encoder counter. It counts the signal edges without further signal interpolation, meaning one signal period corresponds to a counter increment of 4, half a period to 2 etc.

Remarks: Refer to '**clearhwcount**' for setting the counter(s) to zero.

Response: Encoder hardware position-counter

Example:
hwcount Returns the position counter of all axes
hwcount x Returns the position counter of X-axis only

19.20. clearhwcount (Clear Hardware Counter)

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

Description: Set the encoder quadrature hardware-counter positions to zero.
Remarks: Only affects the position returned by ?**hwcount** only.

Response: none.

Example:
clearhwcount Reset hwcount position of all axes to zero
clearhwcount x Reset hwcount position of X-axis to zero

20. MR Encoder Instructions

20.1. mra (MR Amplitude Correction Factor)

Syntax: !mra or ?mra
Parameter: x, y, z, a or none
0.8 to 1.2

Description: This instruction reads or sets the cosine amplification correction factor of the analogue encoder signal (here: sin/cos amplitude ratio). This factor is calculated automatically on each calibration move '**cal**' and should not be changed. If the axis is manually controlled and only used for relative measurement, so that no '**cal**' is possible, the user may determine the ratio itself and then write it into mra for more accurate results. Please also refer to the '**mro**' instruction.

Response: Currently used correction factor(s)

Example:
?mra Read MR signal correction factor of all axes
?mra x Read MR signal correction factor of X-axis only
!mra x 1.0095 Amplify the X cosine signal by *1.0095 compared to the sine

20.2. mro (MR Offset Correction Value)

Syntax: !mro or ?mro
Parameter: x, y, z, a or none
-2048 to +2048

Description: This instruction reads or sets the sine and/or cosine offset compensation value as 16bit signed digits. This factor is calculated automatically on each calibration move '**cal**' and should not be changed. If the axis is manually controlled and only used for relative measurement, so that no '**cal**' is possible, the user may determine the offset itself and then write it into mro for more accurate results. Please also refer to the '**mra**' instruction.

Response: Currently used correction values

Example:
?mro Read MR signal offset value sine and cosine for all axes
?mro x Read MR signal offset value sine and cosine for X-axis only
!mro 48 -100 0 0 0 0 0 Set X offset to sin=48digit, cos=-100digit, Y, Z = 0
!mro y 16 -28 Set Y offset to sin=16digit, cos=-28digit
!mro y 16 Set only sine offset of Y encoder

20.3. 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

20.4. mrt (MR Signal Level)

Syntax: ?mrt
Parameter: x, y, z, a or none
1 to 32767 and -1 to -32767

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

In case of MR encoders, specifying a negative number returns the corrected analog value (offset, amplitude).

The number of data samples (lines) to read should be specified, e.g. "?mrt 1" for returning one sample. If there is no number specified, the instruction returns 10 sampling results per default.

Response: [sine] [cosine] results as signed 16 bit values
or [x_s] [x_c] [y_s] [y_c] [z_s] [z_c] for all
Each data line is terminated by a [CR].

Example:
?mrt Returns 10 lines with all axes (up to 6 values per line)
?mrt 1 Returns one line with all axes (up to 6 values)

?mrt x Returns 10 lines with [x_sin] [x_cos] signal digits
?mrt x 1 Returns one line with [x_sin] [x_cos] signal digits
?mrt y 2 Returns two lines with [y_sin] [y_cos] signal digits
?mrt y 1000 Returns 1000 lines with [y_sin] [y_cos] signal digits

?mrt -1 Returns one line, all axes with MR correction applied
?mrt x -10 Returns 10 lines, X axis with MR correction applied

21. Absolute Encoder Instructions

The TANGO Desktop HE currently supports Renishaw® Atom™ and Numerik Jena LAK absolute encoders. **Encres** and **encform** must be specified.

If the encoder provides an additional 1Vpp signal (as the LAK encoder does), the **encperiod** must also be specified.

Also, a **posshift** parameter is required to align the absolute position to the axis position (shift value to the zero position). The **posshift** parameter is also used for Center Referencing Modes (**caldir**) without absolute encoder.

The **posshift** can be determined by **abspos** or, via an E0/CAL switch, **calabspos**.

21.1. encform (Absolute Encoder Data Format)

Syntax: !encform or ?encform

Parameter: x, y, z, a or none
18 to 32 [bit]

Availability: Only with TANGO Desktop HE.

Description: Length of the position data word in bits. Only the counting data bit size, excluding error, warn and CRC bits.

Response: Absolute encoder position data word size in bit

Example:

```
!encform 32 32 24          Set the word size to 32 in X and Y, to 24 in Z
!encform z 24             Set the word size to 24 in Z
?encform                  => 32 32 24   Read the word size of all axes
?encform z                => 24       Read the word size of the Z-axis
```

21.2. encres (Absolute Encoder Data Resolution)

Syntax: !encres or ?encres

Parameter: x, y, z, a or none
1.000 to 10000 [nm]

Availability: Only with TANGO Desktop HE.

Description: Position data resolution of the absolute encoder interface. In nanometers per counting step (1 bit). Similar to what the encoder period is to analog signals, **encres** translates the counter value into a position.

Optional read-resolution: When reading, the number of requested fractional digits can be sent. Default = 3.

Response: Absolute encoder resolution in nm (nm per count) with 3 or requested amount of fractional digits

Example:

```
!encres 1 1 78.125        Set resolutions X,Y=1nm, Z=78.125nm
!encres z 78.125         Set resolutions to 78.125nm in Z
?encres                  => 1.000 1.000 78.125  Read the resolutions of all axes
?encres z                => 78.125          Read the resolution of the Z-axis
?encres 6                 => 1.000000 1.000000 78.125000
?encres z 6              => 78.125000
```

```
!encres x 1.2345
?encres z                => 1.235
?encres z 6              => 1.234500
```

21.3. abspos (Absolute Encoder RAW Position)

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

Availability: TANGO Desktop HE only, for axes with absolute encoder.

Description: Returns the absolute encoder position at the current position.
The unit is always [mm].

Optional read-resolution: The amount of fractional digits for the reply can be specified (0 to 16).

Remarks: If the axis provides no lower limit switch (E0/CAL), the instruction can be used to once determine the posshift value for zero alignment (posshift = -abspos).

If no E0/CAL switch is available, the absolute position can be determined by manually shifting the axis to the lowest position (maintain some 1/10mm gap to the mechanical limit) and read the local '?abspos' there.

Response: RAW position of the absolute encoder at the current position.
The unit depends on dim.

Examples:
abspos y => 456.2099
abspos y 6 => 456.209874
abspos => 9.7055 456.2099 0.0000 0.0000
abspos 6 => 9.705519 456.209874 0.000000 0.000000

22. Closed Loop Instructions

The closed loop control pulls the axis towards the measuring system position, compensating the inaccuracies of the drive.

The closed loop mode is set by the **ctr** instruction. It also requires setting **encmask** for the individual axes (encmask requests enabling of the encoders, which is a precondition for enabling closed loop). Closed loop is finally activated by either executing a calibration (**cal**) or after power-up by selecting the power-up modes (**calmode** 2, 1 or 4).

The **ctrstatus** instructions may be used to check if the closed loop is activated.

Remarks:

Activating closed loop fails if the **pitch**, **gear** or **encperiod** settings are incorrect. The error tolerance of these parameters is about a factor of 2.

The closed loop target window (**twi**, in combination with **ctrd**, **ctrtrt**) is the condition to identify if the axis has reached its target position. In the default closed loop mode (**ctr** = 2) the axis will (continue to) travel precisely to the target position, even if the target window is already reached.

When the optional motor current **reduction** is set below 0.3 (30%), closed loop will be disabled during reduction (axis has stopped and **curdelay** has expired).

The closed loop behavior can be set to different behaviors for reliability or safety, when exceeding a specified deviation. The instructions **ctrsm** and **ctrs** may be used.

22.1. Setting Up the Closed Loop

The closed loop circuit provides several instructions to adjust, optimize and customize its behavior:

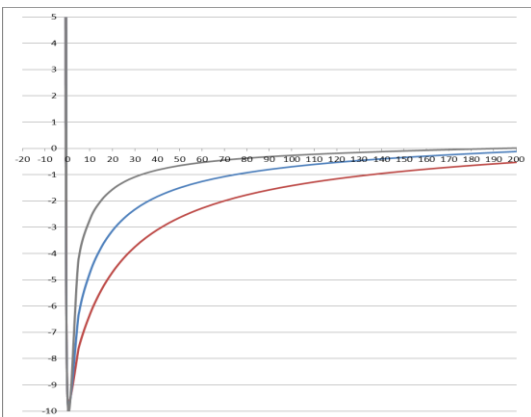
- **ctr** - the closed loop mode
- **ctrff** - the closed loop amplification multipliers
- **twi** - the "position reached" deviation criteria
- **ctrd** - the "position reached" time criteria
- **ctrtrt** - the maximum waiting time for the criteria (timeout)
- **ctrsm** - the behavior at greater deviation
- **ctrs** - the definition of greater deviation

Positioning with Closed Loop - twi and ctrd

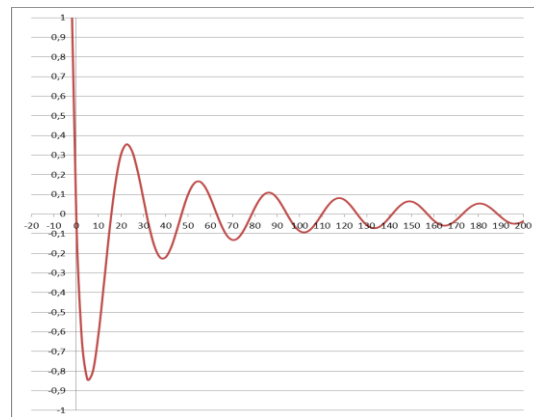
In closed loop mode the position reached reply is delayed until the specified criteria are fulfilled. The criteria consist of a deviation limit and a time, during which the deviation limit must be kept.

The **twi** and **ctrd** instructions define a window with the height of $\pm twi$ and the width of **ctrd**. If the deviation exceeds $\pm twi$, the delay time begins again. So the $(\pm twi * ctrd)$ window travels along, until the criteria are fulfilled or until the timeout (**ctrtrt**) is reached. In such case the position reached reply will be sent even without fulfilling the criteria.

The window definition is required, as position deviation of the axis isn't just constantly decreasing, there is also oscillation of the axis position:



P1 Deviation at the end of a move compensated by the closed loop



P2 Oscillation at the end of a move (sinusoidal decay)

So without defining a **ctrd** delay time criterion, the position reached reply would be sent as soon as the deviation once went below the **twi** value. But as **P2** shows, this might be just temporary and the deviation will build up again (oscillation). To be safe, **ctrd** time must be at least the oscillation period, which makes sure both, minima and maxima of the deviation are checked within the criteria window.

Example: **twi** is $\pm 0.2\mu m$ and **ctrd** is 30ms. In **P2**, the **twi** limit is exceeded at 1ms and remains exceeded until about 14ms. During this time, the **ctrd** time countdown of 30ms is constantly restarted. At 18ms the **twi** limit is exceeded once again until about 29ms, during this time the **ctrd** time is restarted again. After 29ms, the deviation remains within $\pm twi$. So **ctrd** keeps counting down its 30ms and replies the position reached event $29+30 = 59ms$ after the original move has completed.

Closed Loop Setup

The closed loop factor **ctrff** can be used in order to reach the **twi** and **ctrd** criteria more quickly.

In general, its default values of 2 and 2 provide a good base to start from. Ctrff has two parameters per axis. The first one is the factor applied when the axis is traveling, the second is applied when the axis stopped traveling and is idle. Picture *P1* shows how increasing the second factor influences compensation of position error at the end of a move. Greater values cause the axis to compensate faster, but from a certain point might begin to be unstable.

Closed Loop behavior

Ctr sets the general behavior of the closed loop, which might be off, always on or on until position is reached.

In addition, a behavior can be set when the position deviation exceeds a certain limit. Please refer to the **ctrsm** and **ctrs** instructions.

Examples

1: enable closed loop for X and Y axes only

```
!ctr 2 2 0
!encmask 1 1 0
!cal
```

2: enable closed loop individually for Z axis

```
!ctr z 2
!encmask z 1
!cal z
```

3: enable closed loop for X and Y from power-on

```
!ctr 2 2
!encmask 1 1
!calmode 2 2
save
(then cycle power or send reset instruction)
```


22.2. ctr (Control Enable)

Syntax: !ctr or ?ctr
Parameter: x, y, z, a or none
0,1,2 (,3,4)

Description: Sets, reads or changes the closed loop mode.

0 = Closed Loop OFF
1 = Closed Loop inactive when position is reached
2 = Closed Loop always active (default, recommended)
3 = (not supported, behaves like mode 1)
4 = (not supported, behaves like mode 2)

Closed loop mode is activated by either executing **cal** or after a power up or reset (refer to **calmode**).

If encoders were activated at power on (**encmask** = 1), closed loop can be switched on (0->1, 0->2) afterwards. Closed loop mode can also be changed during operation.

Preconditions for successfully entering closed loop are having **ctr** and **encmask** set.

Pros and Cons: Mode 1: (+) Axis is stopped after reaching the target window
(-) Position is scattered within $\pm twi$ range
(-) Position deviation may occur due to external influence (drive concept, gravity, etc.)
(-) Is only active when moving
(-) Can cause a position jump at the beginning of a move, if a deviation exists (see influences above)

Mode 2: (+) Always stays closest possible to target position
(+) Compensates external influences (required with several odif concepts or with Z axes)
(-) Might still travel after position is reached (after the "@@@" reply of the target window) as it always travels towards zero deviation
(-) Might not absolutely stand still after reaching a position. Depending on drive concept, slip stick effects can cause the axis to slightly bounce between positions within e.g. ± 50 nanometers. similar might happen with TTL encoders of coarse resolution. Then mode 1 might be preferred.

Remarks: Using closed loop mode 2 is recommended for most applications.

Mode 1 is recommended only if the application does not stand still sufficiently in the end position (e.g. long exposures). In this case, the HDI - especially an ERGODRIVE - should be disabled, too (by e.g. **!joydir** 0 0 0 0 or **!joy** 0 instruction).

Response: Closed loop mode(s)

Examples:

!ctr 0 0 0 0 Closed loop off for all axes (here: 4 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

22.3. ctrf (Control Factor)

Syntax: !ctrf or ?ctrf
Parameter: x, y, z, a or none
0.0 to 25.0

Description: CTRF IS FOR COMPATIBILITY ONLY - PLEASE USE **CTRFF**.
This instruction reads or sets the closed loop factors.
Higher values result in more stiffness and faster settling.
Above a critical value this may lead to oscillation.
The default factor of 2.0 mostly results in a good behavior.
Hint: Using the ctrff instruction instead offers more options.

Remarks: Even if setting the parameter supports floating point, the return value is integer (for compatibility).
When using ctrf, the one parameter applies to both, idle and move. It is recommended to use ctrff instead of ctrf.

Response: Closed loop factors as integers (rounded)

Examples:
!ctrf 2 2 2 Set closed loop factor to 2 for all axes
!ctrf z 3.5 Set closed loop factor for Z axis to 3.5
?ctrf Read closed loop factors of all axes (as integer)
?ctrf z → 4 Read closed loop factor of Z axis (as integer, 3.5 → 4)

22.4. 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 settling.
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 specified)

Parameter1: Is used while axis is traveling
Parameter2: Is used when axis is idle or while trying to settle within the target window (**twi**)

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) as fractional numbers

Examples:
~~!ctrff 2 2 2 2~~ Not supported!
!ctrff x 2 4.5 Set closed loop factors for X axis 2(moving) and 4.5(reached)
!ctrff x 0 0 Disable closed loop in X, while loop calculations keep running
?ctrff Read closed loop factors of all axes (2 parameters per axis)
?ctrff y Read closed loop factors of Y axis only (2 parameters)
?ctrff → 2.0 4.5 2.0 2.0 2.0 2.0 (response of a 3 axis TANGO)
?ctrff y → 2.0 2.0

22.5. ctrd (Control Target Window Delay)

Syntax: !ctrd or ?ctrd
Parameter: 0 to 1000 [ms]
optional axis x,y,z,a

Description: This instruction reads or sets the target window delay that is used as a position reached criteria in closed loop mode. The position deviation at the end of a move must remain inside the target window (**twi**) for this amount of time, then the "position reached" state is set. If the target window is left before the delay time is over, the delay starts counting again. Please also refer to the **ctrt** timeout, which aborts the waiting for **twi+ctrd** after a certain amount of time. Either one delay for all axes can be set or axes can be set individually.

Remarks: Setting the delay time to zero forces the axis to immediately reply without waiting for the axis to match the target window.

In all other cases, **ctrd** delay times must be set higher than the **ctrc** interval (>5ms).

While the default target delay of 100ms works with most drive concepts, it may be optimized for performance individually. In order to find the optimal setting, the sinusoidal decay of the axis must be measured. The **ctrd** delay time should be at least half the signal period of the mechanical oscillation.

Response: Closed loop control delay in milliseconds.

Examples:
~~!ctrd 50 50 50~~ Not supported!
!ctrd 50 Closed loop target window delay to 50 ms for ALL AXES
!ctrd x 25 Closed loop target window delay to 25 ms for X axis
?ctrd Read closed loop target window delay (for backw.compatibility)
?ctrd y Read closed loop target window delay of Y axis
?ctrd -1 Read closed loop target window delay of all axes

22.6. crtt (Control Timeout)

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

Description: This instruction reads or sets the control timeout. It specifies the maximum time the closed loop tries to reach the desired **target window**. If the **ctrd+twi** condition could not be fulfilled within this **crtt** time, it will be aborted. The **crtt** timeout should be set to a value higher than **ctrd**, usually to the default of 1 second. The unit is milliseconds. Only one parameter for all axes.

Remarks: Setting the timeout to zero forces all axes to immediately reply without waiting to match their target windows.

Response: Closed loop control timeout in milliseconds.

Examples:
!crtt 1000 Closed loop tries to reach the target window for 1 second
?crtt Read closed loop timeout

22.7. twi (Target Window)

Syntax: !twi or ?twi
Parameter: x, y, z, a or none
±window size, unit depends on **dim**, range = 0.00001 to 1 mm

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 closed loop timeouts (higher ctrd, ctrt values required). The unit depends on **dim**.

From Firmware 1.73 when reading, an optional read-resolution of 0 to 6 (1mm to 1nm) can be specified. Without specifying, the resolution depends on the setting of '**resolution**'.

Remarks: The target windows minimum size should not be set smaller than the encoder resolution and noise. Please make sure the measuring system is able to deliver the required signal quality. For analog encoders the limit might be between 1/1000 to 1/5000 of the signal period, which in case of MR encoders about 0.5µm and for a 4µm optical scale 4nm.

If the target window is smaller than the '**resolution**', e.g. 50nm, either the resolution must be increased or the twi must be read with higher resolution by specifying it.

Response: Closed loop target window. The unit depends on **dim**.

Examples:

```
!twi 0.001 0.001 Closed loop target window ±1.0µm (if dim=2) for X and Y-axis
!twi y 0.002 Closed loop target window ±0.2µm (if dim=2) for Y-axis
?twi Read target window of all axes
?twi z Read target window of Z-axis
?twi 6 Read target window of all axes with 1nm resolution
?twi z 6 Read target window of Z-axis with 1nm resolution
!twi z 0.00005 Example: Set twi in Z to 50nm (here at "mm" dim 2 or 9)
?twi z => 0.0000 Then reading will return 0 at the default resolution (4)
?twi z 6 => 0.000050 Firmware 1.73 and higher provide individual resolution
```

22.8. ctrc (Control Call)

Syntax: !ctrc or ?ctrc
Parameter: 1 to 100 [ms] **DO NOT CHANGE!**

Description: This instruction reads or sets the controller call interval. It specifies the time interval in which the closed loop circuit checks the position deviation.

The unit is milliseconds. There is only one parameter which applies to all axes. **THE DEFAULT FACTORY SETTING (3 or 5 ms) depends on the TANGO controller and SHOULD NOT BE CHANGED.** Values of less than 3 [ms] are not recommended.

Response: Closed loop control call interval in milliseconds.

Examples:

```
!ctrc 5 Closed loop control is executed every 5 milliseconds
?ctrc Read closed loop call interval
```

22.9. ctrsm (Control behavior outside Lock-in Range)

Syntax: !ctrsm or ?ctrsm
Parameter: x, y, z, a or none
0 to 5

Description: Behavior of the closed loop circuit when outside the lock-in range (**ctrs**). When outside the ctrs lock-in range, which is regarded to be an error, the TANGO controller can treat this condition as follows:

0 = Continue as usual, motor might possibly stall (default)
1 = Limit the closed loop velocity (avoid stalling the motor)
2 = Disable closed loop until returning to lock-in range
3 = Disable closed loop permanently
4 = Axis is set to latched stop condition (read remarks)**
5 = Switch off all power stages (like sending "!pa 0")

Remarks: ** **stoppol** is manipulated by this function in order to achieve a latched stop (stoppol value is or'ed by 4).
The stop condition must be released by sending "**!stop 0**"

ctrsm 1 assures to return from large deviations without stalling the motor. If outside the **ctrs** lock-in range, a slow return velocity is applied. In most cases it is the better, reliable alternate to the default ctrsm 0.

Ctrsm modes 3 to 5 may be used for safety.

Response: Selected Behavior of the axes (as integer value)

Examples:
!ctrsm z 1 Select slow mode in Z (to avoid stalling of the motor)
!ctrsm 5 5 5 Select switch off mode for safety (e.g. collision detection)
?ctrsm Read all behaviors
?ctrsm y Read behavior of Y-axis only

22.10. ctrs (Control Lock-in Range)

Syntax: !ctrs or ?ctrs
Parameter: x, y, z, a or none
0.01 [mm] to [**maxpos**]

Description: Lock-in range of the closed loop circuit.
When the closed loop position difference exceeds this limit, the behavior defined with **ctrsm** is applied to the axis/axes. The unit depends on **dim**.

Remarks: From Firmware 1.73 when reading, an optional read resolution of 0 to 6 (1mm to 1nm) can be specified. Without specifying, the resolution depends on the setting of '**resolution**'.

Response: Closed loop lock-in range(s), unit depends on **dim**.

Examples:
!ctrs 0.5 0.5 0.2 Set lock-in range of X=0.5mm, Y=0.5mm, Z=0.2mm (if dim=2 or 9)
!ctrs z 0.1 Set lock-in range of Z=0.1mm (if dim = 2 or 9)
?ctrs Read lock-in range of all axes
?ctrs z Read lock-in range of Z-axis only

22.11. ctrstatus (Control Status)

Syntax: ?ctrstatus or ctrstatus

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

Description: Read the internal closed loop states of the specified or all axes. Options and responses are:

A) Called without parameter:

Returns the internally applied **ctr** state which is set when the closed loop gets enabled by the controller (after !cal or when in calmode=1 or 2).

0 = Closed loop permanently off (or not activated yet)

1 = Closed loop only active while axis is traveling

2 = Closed loop always on (the default closed loop mode)

(3 = Closed loop only active while axis is traveling)

(4 = Closed loop always on)

B) Called with parameter "1":

Check if the closed loop is currently active.

0 = Closed loop not active

(e.g. ctr=0, ctr=1, !cal running, encerr, limit swich)

1 = Closed loop active

C) Called with parameter "2":

Check if closed loop is in target window.

0 = Position outside target window

1 = Position in target window

D) Called with parameter "3":

Check if closed loop is in lock-in range.

0 = Position outside lock-in range

1 = Position in lock-in range

E) Called with parameter "0": (All in one request)

Returns the internally applied **ctr** state, like **A)** does and a second hex parameter representing all state bits of the calling parameter 1,2,3 like cases **B,C,D** above:

Bit 0 (0x1): Closed Loop Active

Bit 1 (0x2): Closed Loop in Target Window

Bit 2 (0x4): Closed Loop in Lock-In Range

Response: Closed loop state, 1 or 2 decimal numbers. See "Description".

Examples:

?ctrstatus Returns the internally running ctr mode of all axes
?ctrstatus y Returns the internally running ctr mode of the Y-axis

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

?ctrstatus 2 Returns if Closed Loop is in target window, e.g. "1 1 0"
?ctrstatus z 2 Returns if Closed Loop of the Z-axis is in target window

?ctrstatus 3 Returns if Closed Loop is in lock-in range, e.g. "1 1 0"
?ctrstatus z 3 Returns if Closed Loop of the Z-axis is in lock-in range

ctrstatus x 0 → 2 7 (ctr mode 2 is applied, is active/in window/in range)
ctrstatus 0 → 2 7 2 7 0 0 (response of 3 axes X X Y Y Z Z)

22.12. ctrdiff (Control Position Difference)

Syntax: ?ctrdiff or ctrdiff
Parameter: x, y, z, a or none
 None, 1, 2, 3 or 4

Description: This instruction returns the momentary measured closed loop position difference between the motor- and encoder position.

From TANGO Firmware 1.69 and above, ctrdiff can be used without a leading '?'. Newer firmware versions might have additional options. Please refer to the examples below.

From TANGO Firmware 1.72 and above, when called with an axis (x, y, z or a), ctrdiff returns at least 5 fractional digits (10nm resolution) and a second parameter which indicates if the motor is running (=1) or idle (=0) when called with none, 1, or 2 parameters. Refer to examples below.

The unit of the returned value depends on the **dim** settings. For higher resolutions the **resolution** value can be increased.

Remarks: Difference is only calculated when closed loop is activated. To measure position deviations without closed loop influence, the closed loop **ctrff** parameters can temporarily be set to 0 in order to suppress regulation.

Response: Momentary position difference, refer to examples.

Examples:
?ctrdiff Returns the position difference of all axes
?ctrdiff y Returns the position difference of the Y-axis, e.g. "0.0015"

From Firmware 1.69:

```
?ctrdiff                 Returns the position difference of all axes
ctrdiff                 same as ?ctrdiff, question mark not required
ctrdiff z               Returns the position difference of the Z-axis
ctrdiff x 1              Returns the internal closed loop circuit output signal of X
ctrdiff 1                Function not available, behavior is same as ctrdiff
```

From Firmware 1.71:

```
ctrdiff                 same as ?ctrdiff, question mark not required
ctrdiff z               Returns the position difference of the Z-axis
ctrdiff x 1              Returns the internal closed loop circuit output signal of X
ctrdiff 1                Returns the internal closed loop output signal of all axes
ctrdiff 2                Returns the internal motor position shift of all axes
ctrdiff x 2              Returns the internal motor position shift of the X axis
ctrdiff y 3              Returns position difference AND output signal of Y
ctrdiff 3                Not supported
ctrdiff y 4              Returns position difference AND motor position shift of Y
ctrdiff 4                Not supported
```

Examples for Firmware ≥ 1.72:

```
ctrdiff z               => -0.13852 0 (5 fractional digits and "motor idle")
ctrdiff z               =>  0.06331 1 (5 fractional digits and "motor running")
ctrdiff x 1              => 17.88353 1 (5 fractional digits and "motor running")
ctrdiff x 2              => 22.06331 1 (5 fractional digits and "motor running")
ctrdiff x 3              (as with earlier firmware)
ctrdiff y 4              (as with earlier firmware)
```

23. Trigger Output Functionality (option)

Trigger functionality must be configured by factory.

To identify if the Trigger functionality is configured, use `'?det'` or `'detext'`.

The trigger output generates TTL signals dependent either on axis positions or as a constant frequency. It can be used to synchronize external devices like e.g. a camera. TANGO Desktop, PCI/PCI-E and TANGO 3 mini provide up to 2 trigger outputs via the optional AUX I/O connector. See **mode** and **output description**.

A special trigger mode - ideal for on the fly scanning applications - is provided by the `!trigr` instruction. This mode achieves the highest accuracy. A more sophisticated mode is provided by `!trigp`, an individual position list.

The trigger can be based on the motor position or on the more accurate encoder position, if the axis provides position encoders.

Axis positions and analog inputs can be captured on trigger by **SnapShot Mode 8**.

The LED100 illumination is active low and typically wired to OUT2 (TAKT_OUT). Remarks: If **hdimode** controls the LED100, it can interfere with TAKT_OUT.

The trigger signals are processed in a 40-microsecond interval.

Before enabling the trigger function by `!trig 1`, please ensure that all trigger settings have been made.

```
Example1:  !trig 0[CR]           Disable trigger globally
           !trigm 0[CR]        Select trigger mode 0
           !triga x[CR]        Set X axis as trigger source
           !trigd 0.100[CR]    Set trigger distance to 100µm (if dim = 2)
           !trigs 400[CR]      Set trigger pulse width to 0.4ms
           !trig 1[CR]        Enable trigger and set start position
```

```
Example2:  !trig 0[CR]           Disable trigger globally
           !trigs 120[CR]        Set trigger pulse width to 120µs
           !trigf 2500[CR]      Set pulse frequency to 2.5kHz
           !trigm 100[CR]       Select trigger mode 100 (periodic signal)
           !trig 1[CR]        Enable trigger
```

Optional: `!trigcount 0` instruction may be executed to reset the event counter.

23.1. trig (Trigger)

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

Description: This instruction enables or disables the trigger circuit. The position at which `!trig 1` is executed also defines the start position for trigger modes 0 to 11 (→current pos).

0 = Trigger function globally disabled
1 = Trigger function globally enabled

Response: 0 or 1

Examples:
`!trig 1` Enable trigger circuit (and define the start position = here)
`?trig` Read enable state of trigger circuit

23.2. trigm (Trigger Mode)

Syntax: !trigm or ?trigm
 Parameter: 0 to 11, 100 to 105

Description: This instruction selects the required trigger mode.

Trigger Mode	Trigger Generation	Trigger Signal	Remarks
0		High active	First pulse when move starts <i>Direction forward</i>
1		High active	First pulse when move starts <i>Bidirectional</i>
2		High active	First pulse when move starts <i>Direction backward</i>
3	-- See Mode 0 --	Low active	Same as 0, signal inverted
4	-- See Mode 1 --	Low active	Same as 1, signal inverted
5	-- See Mode 2 --	Low active	Same as 2, signal inverted
6		High active	Triggers shifted by $\text{trigd}/2$ <i>Direction forward</i>
7		High active	Triggers shifted by $\text{trigd}/2$ <i>Bidirectional</i>
8		High active	Triggers shifted by $\text{trigd}/2$ <i>Direction backward</i>
9	-- See Mode 6 --	Low active	Same as 6, signal inverted
10	-- See Mode 7 --	Low active	Same as 7, signal inverted
11	-- See Mode 8 --	Low active	Same as 8, signal inverted
100	Periodic trigger signal the frequency can be set by "trigf" instruction	High active	Does not depend on position
101		Low active	
102	Manually forced trigger signals by the "trigger" instruction which releases one or several pulses	High active	Does not depend on position or time
103		Low active	
104	Position reached trigger signal when all moves including the specified axis "triga" have completed	High active	Comes with the "@@@" Response, → 'autostatus' must be on (not 0)
105		Low active	



Remarks: The start position is defined by the position where the trigger was globally enabled (by "**!trig 1**" instruction).

 Trigger modes 20 and 21 are applied internally by the **trigr** and **trigg** instructions. In those two cases trigger mode 20 is for increasing positions (pos. direction), trigger mode 21 is for decreasing positions (neg. direction).

Response: Trigger mode as integer: 0 to 11, (20,21), 100 to 105

Examples: !trigm 0 Select trigger mode 0
 ?trigm Read current trigger mode (returns e.g. 0)

23.3. triga (Trigger Axis)

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

Description: This instruction selects the axis on which to trigger.

Response: x, y, z or a

Examples:
 !triga y Select Y-axis as trigger source
 ?triga Read current trigger axis (returns x,y,z or a as lower case)

23.4. trigo (Trigger Output)

Syntax: !trigo or ?trigo
 Parameter: 0 to 15

Description: This instruction selects the trigger outputs.

The secondary output TAKT_OUT provides extended functionality:
 2, 3: 1:1 mode, generating the same signal as output 1
 6, 7: precise width
 10,11: high precision delay
 14,15: precise (and optionally even higher) frequency

TAKT_OUT is also the default output to control the LED100.

Output / Mode	STANDARD	PREC.WIDTH2	PREC.DELAY2	PREC.FREQUENCY2
No output No signal out	0	(4)	(8 PCI-E)	(12)
Primary TRIGGER OUT	1	(5)	(9 PCI-E)	(13)
Secondary ** TAKT OUT	2	6	10 (PCI-E)	14
Both, P&S ** TRIGGER+TAKT	3	7	11 (PCI-E)	15

A combined delay and width of 40µs resolution is available in standard modes 1,2,3 by the **trigs** instruction.

****** For further information on the second trigger output, please refer to **trigs**, **trigwidth**, **trigbdelay**, **trigbf** and the description of the **second trigger signal output**.

Remarks: Options depend on hardware: TANGO PCI-E/DT-E and TANGO 3 mini controllers with AUX I/O connector offer all features. PCI-S based controllers do not provide the precision delay function (only a precise edge is generated there) and TANGO mini or integrale provide none or one trigger output.

Response: Selected trigger outputs

Examples:
 !trigo 0 No output signal
 !trigo 1 Default trigger output (TRIGGER_OUT)
 !trigo 2 Secondary trigger output (AUX I/O TAKT_OUT, LED100)
 !trigo 3 Both trigger outputs 1:1 (TRIGGER_OUT and TAKT_OUT)
 ?trigo Read back the selected trigger output mode

23.5. trigs (Trigger Signal Length)

Syntax: !trigs or ?trigs
Parameter: 0 to 2500000 [μ s]
or optional 3x 0 to 2500000 with secondary trigger option

Description: This instruction sets the trigger pulse width in the range of 40 microseconds to 2.5 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 nearest multiple: e.g. 90 \rightarrow 80, 100 \rightarrow 120.
When read back, the corrected (nearest) value is returned.

Remarks: **Secondary Trigger Option:** TANGO controllers with a secondary trigger output (AUX I/O) offer a **1:1 mode**. When !trigs is called with 3 parameters, OUT2 provides the option of a delay and individual width. Both parameters must be specified and may be set from 0 to 2500000 (μ s) in increments of 40 (μ s), as mentioned above.

The 2nd trigger signal must be active within the range of the 1st signal. At the end of signal 1, signal 2 is set back to inactive, too. For further information, refer to the trigger description of the **Standard 1:1** output mode.

Example: !trigs 120 40 80

Generates 120 μ s on OUT1 and a 40 μ s delayed 80 μ s long pulse on OUT2. The 2nd and 3rd parameter cannot be read back by ?trigs, it only returns the signal length of output OUT1.

Response: 0 to 2500000 (μ s) trigger signal length

Examples:

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

!trigs 200 40 80 Set Trigger output 1 pulse width, delay, output 2 pulse width
The two additional parameters cannot be read back with ?trigs

23.6. trigd (Trigger Distance)

Syntax: !trigd or ?trigd
Parameter: >0.0 to 5000000 (unit depends on **dim** of the selected axis)

Description: This instruction sets or reads the trigger distance.
Equidistant trigger signals are generated in this position interval.

Remarks: The trigger axis (**triga**) should be defined before setting trigd.

Response: Trigger distance (the unit is defined by **dim**)

Examples:

!trigd 3.01 Set trigger distance to 3.01mm (if dim of selected axis is 2)
?trigd Read the trigger distance

23.7. trigcomp (Trigger Compensation)

Syntax: !trigcomp or ?trigcomp
Parameter: -10000 ... +10000 [μ s] (smallest step is 10 μ s)

Description: Time delay compensation for position trigger (look ahead).
It releases the trigger at an earlier (+) or later (-) time
in order to be executed at the required position.

It can be used to compensate time delays in the signal chain
or to center a trigger pulse around the target position.

Application 1:

To avoid stitching mismatch with bidirectional scans
(and so increase performance by not requiring unidirectional)

At high velocity on the fly scans, the delay of the trigger
signal chain has an effect on where the sample is taken.
When scanning in both directions, this effect becomes visible
typically by a comb-like appearance of the stitched image.
This can be greatly improved or entirely removed by the
trigcomp delay compensation. The default setting is 0.

Example: If a camera has a shutter release delay of 100 μ s and
the axis travels at 10mm/s, an uncompensated bidirectional
scan will cause a 1 μ m shift in each direction, causing a 2 μ m
shift between forward and backward direction. When this delay
is compensated by "!trigcomp 100", the trigger is released at
an 100 μ m earlier position (which the TANGO calculates
internally, based on the current velocity) and the camera now
will take the picture without any position shift.

Remarks:

The described comb effect can also be caused by a mechanical
backlash of the scan axis. On open loop axes without encoder
feedback, the effect can possibly be minimized by using the
backlash compensation.

Application 2:

To generate a symmetrical trigger pulse, which is active for a
certain time before and after the trigger position.

Example: The pulse is 200 μ s long and should be active 100 μ s
before and after the trigger position

→ **trigs** 200, **trigcomp** 100

Response: Compensated delay in [μ s]

Examples:
!trigcomp 130 Compensate trigger signal chain delay of 130 μ s
?trigcomp Read the compensation delay (e.g. returns 0)

23.8. trigenc (Trigger on Encoder)

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

Description: Trigger position source select, encoder or motor position.

0 = Trigger position from motor position (default)
1 = Trigger position from encoder (true position)

Remarks: Triggering on encoder signals is designed for analogue 1Vpp or MR encoders. When using A/B-TTL encoders, the available position range might be extremely limited due to the low encoder period. The valid trigger position range is ± 32000 encoder periods
Examples:

- MR @ $500\mu\text{m} * \pm 32000 = \pm 16000 \text{ mm}$
- 1Vpp @ $20\mu\text{m} * \pm 32000 = \pm 640 \text{ mm}$
- TTL @ $1\mu\text{m} * \pm 32000 = \pm 32 \text{ mm}$

In order to use trigenc=1, the encoder must be activated. Without active encoder (?enc = 0), the motor position is used.

Response: Currently selected trigger position source

Examples:
!trigenc 1 Trigger based on the encoder signal, if available
?trigenc Read the trigger position source, e.g. 0

23.9. trigf (Trigger Frequency)

Syntax: !trigf or ?trigf
Parameter: 0.01 to 25000

Description: This instruction sets the frequency for periodic trigger output modes **trigm** 100 and 101. The internal frequency resolution is in steps of $1/40\mu\text{s}$.

Response: Trigger frequency

Remarks: As the internal resolution has $40\mu\text{s}$ steps, the resulting frequency might not always match the requested frequency. The higher the frequency, the more deviation may occur (e.g. 2500 exactly meets the frequency, 2600 does not). In order to identify the resulting frequency, ?trigf can be used.

For highly accurate frequencies with fine resolution, the second trigger output can be used (refer to **trigbf**).

From TANGO Firmware 1.60C / 1.61 it is also possible to send a fixed number of trigger pulses at the specified trigf by using the manual trigger modes 102, 103 and calling the manual **!trigger** instruction with the number of pulses as parameter. Refer to **trigger** description.

Examples:
!trigf 2500 Periodic trigger pulses at 2.5kHz (signal every 0.4ms)
?trigf Read trigger signal frequency (the true value)

23.10. trigbdelay (Precise Trigger Delay for second output)

Syntax: !trigbdelay or ?trigbdelay
Parameter: 0.00 to 32500000 [µs]

Description: Precise delay for secondary trigger output signal (TAKT_OUT). Applies to trigger output settings !trigo 10 and 11. Unit in microseconds [µs], resolution is 1/132µs.

Remarks: Secondary trigger can either have precise width or delay. **Only available with TANGO PCI-E, Desktop-E and TANGO 3 mini. TANGO PCI-S and Desktop-S only provide a delayed signal edge (rising or falling), while the signal state before and after the trigger pulse remains in an active state.**

Response: Delay time in µs

Examples:
!trigbdelay 0.35 Delay the secondary trigger signal by 350ns (to TRIGGER_OUT)
?trigbdelay Read the secondary trigger delay (e.g. returns 0.00)

23.11. trigbwidth (Precise Signal Width for second output)

Syntax: !trigbwidth or ?trigbwidth
Parameter: 0.00 to 32500000 [µs]

Description: Precise width for secondary trigger output signal (TAKT_OUT). Applies to trigger output settings !trigo 6 and 7. Unit in microseconds [µs], resolution is 1/132µs.

Remarks: Secondary trigger can either have precise width or delay. **Only available with TANGO PCI-S, PCI-E, Desktop-S, Desktop-E and TANGO 3 mini.**

Response: Signal width in µs

Examples:
!trigbwidth 5.01 Set secondary trigger signal width to 5.01µs
?trigbwidth Read the secondary trigger signal length (e.g. returns 40.00)

23.12. trigbf (Precise Trigger Frequency for second output)

Syntax: !trigbf or ?trigbf
Parameter: 0.010 ... 66000000 [Hz]

Description: Precise frequency for secondary trigger output (TAKT_OUT). Applies to trigger output settings !trigo 14 and 15. Unit is microseconds [µs], resolution is 1/132µs.

Remarks: Trigcount does not count the precise trigger events. **Only available with TANGO PCI-S, PCI-E, Desktop-S, Desktop-E and TANGO 3 mini.**

Response: Output frequency [Hz] for precise frequency output mode.

Examples:
!trigbf 0.01 Set secondary trigger frequency to 0.01Hz
!trigbf 50000005 Set secondary trigger frequency to 50.000005 MHz
?trigbf Read the secondary trigger frequency (e.g. returns 1000.000)

23.13. trigcount (Trigger Counter)

Syntax: !trigcount or ?trigcount
Parameter: 0 to 2147483647

Description: Read or set the trigger event counter.
Trigcount increments on every executed trigger pulse.

Response: Number of executed triggers

Examples:

```
?trigcount      Read trigger counter
!trigcount 0    Clear trigger counter
!trigcount 110  Set trigger counter to 110 counted events
```

23.14. trigger (Force Trigger Signal)

Syntax: !trigger or trigger
Parameter: None or 0 to 127

Description: This instruction generates one trigger output pulse or a fixed amount of pulses, refer to description below. Manual trigger is available in manual **trigger modes** 102 and 103 only. The pulse width depends on **trigs** setting.

From TANGO Firmware 1.60C / 1.61 it is also possible to manually generate a fixed number of 1 to 127 trigger pulses by calling the instruction with an additional count parameter. The pulse frequency depends on the **trigf** setting.

Remarks: When executing multiple triggers (trigger was called with parameter), the function does not return until all trigger pulses are executed (the command interpreter is blocked). To identify if the instruction completed - and that the TANGO controller is able to receive new instructions - it is recommended to send an ?err request and wait for it to return. → Refer to Example 2(b).

Response: None

Example1: trigger (Force one trigger pulse now)

Example2: (requirement is at least TANGO Firmware 1.60C or 1.61)

```
!trig 0          (Disable trigger)
!trigf 1000      (Set frequency of trigger signal in Hz)
!trigs 400       (Set duration of trigger pulse in µs)
!trigm 102      (Set trigger mode to manual trigger 102 or 103)
!trig 1          (Enable trigger)
```

```
trigger 5        (Manually force 5 trigger pulses at 1000 Hz)
```

Example2(b): err → 0 (Optional wait for trigger function to complete by waiting for err response, e.g. zero)

23.15. trigr (Set Trigger Range)

Syntax: !trigr or ?trigr

Parameter: x, y, z, a or none
start position
end position
number of equidistant trigger signals within a position range

Description: Provides the *range trigger mode*, which
- begins and ends at defined positions
- generates a defined number of equidistant, unidirectional trigger signals (10000 max.)

Setting a trigger mode is not required, !trigr sets it to mode 20 (up) or 21 (down) automatically.

If no axis is specified, the trigger axis is used (?triga).
If an axis is specified (x,y,z,a) the trigger axis is set to this axis and remains set until changed.

Trigger signals are generated within a position range.
The scan direction is set by the start,end positions:
- start < end = trigger in positive travel direction
- start > end = trigger in negative travel direction

Start and end position units are defined by **dim**, e.g. mm.

The number of trigger signals defines their distance:
The first trigger is generated at the start position and the last trigger is generated at the end position (n+1).
e.g. to achieve 1mm distance from 0 to 10 mm = 11 signals.

If only one trigger position is specified (start = end), the trigger direction is determined from the current position:
- If the trigger position is greater than the axis position, the trigger direction is set to positive (up, forward).
- If the trigger position is lower than the axis position, the trigger direction is set to negative (down).

Once set, the trigger range remains and is reactivated automatically when the axis travels back behind the start position. If required, !trigr can set new parameters ahead of every move instruction.

For bidirectional scans, !trigr must be set to the required travel direction before each move (from pos -> to pos).
The move must start from below the start position and travel past the end position.

Remarks: The signal polarity for this special trigger mode can be set by the **!trigl** instruction.
In order to apply the signal polarity immediately, the trigger mode can be set to **!trigm = 20** or **21** before setting **trigl**.

The trigr instruction clears and overwrites possibly existing **triggp** position lists of the specified axis and vice versa.
(!trigr internally writes its positions to the trigr list.)

Response: Currently applied values: [startpos] [endpos] [num of trigger]

Examples for trigr:

--- Preparation ---

- Globally disable the trigger:

```
!trig 0
```

- Select trigger polarity:

```
!trigm 20 (mode 20 or 21 to apply trigr immediately)
!trigl 1 (set trigger signal level to active high)
```

- Specify the axis:

```
( !trigenc 1 ) (optional if the axis provides an encoder)
!triga x (can also be specified or changed by !trigr)
```

--- Trigger ---

- A) Trigger in positive direction:

```
!moa 9 (move X to position below the trigger range)
!trigr 10 20 11 (one trigger every mm, 10,11,12,...20)
!trig 1 (globally enable the trigger**)
!moa 21 (move X to position past the trigger range)
```

- B) Trigger in negative direction:

```
!moa 21 (move X to position below the trigger range)
!trigr 20 10 11 (one trigger every mm, 20,19,18,...10)
!trig 1 (globally enable the trigger**)
!moa 9 (move X to position past the trigger range)
```

- C) Single trigger position:

```
!moa 19 (move X to position below the trigger range)
!trigr 20 20 1 (one trigger at 20mm)
!trig 1 (globally enable the trigger**)
!moa 21 (move X to position past the trigger range)
```

```
!trigr y 11.5 16 10 (select y axis for trigger from 11.5 to 16)
```

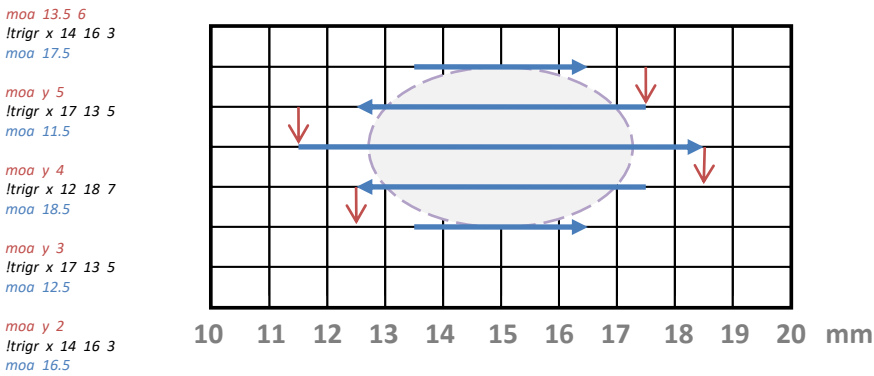
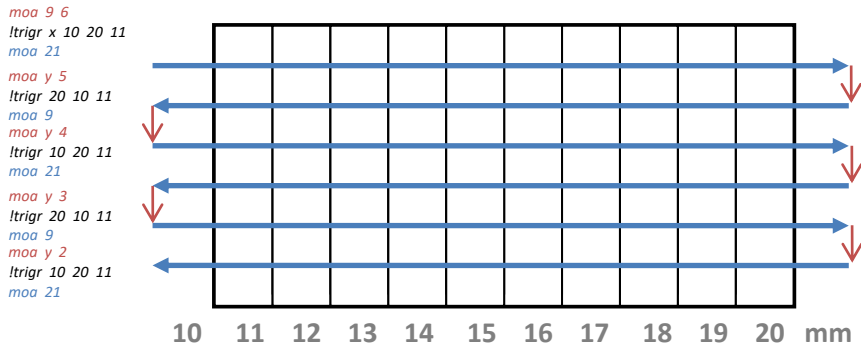
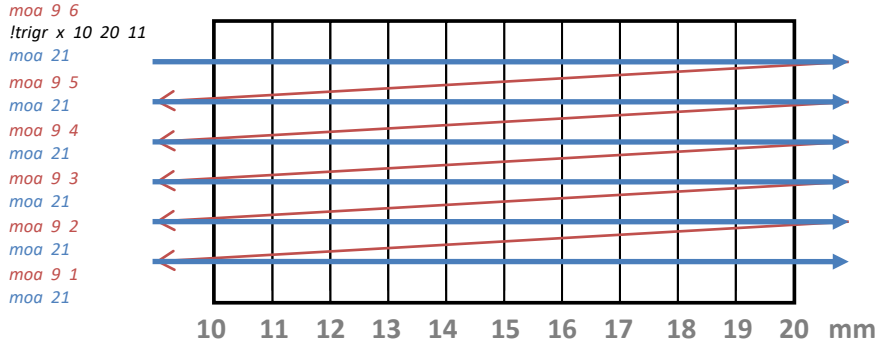
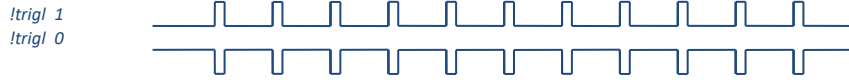
```
!trigr 11.5 16 10 (use recently set axis for trigger)
```

```
?trigr → 11.5000 16.5000 11 (read trigger settings)
```

```
?triga → y
```

** Trigger should be globally disabled for the initial configuration and may remain enabled afterwards, even if the !trigr parameters are modified.

Trigger Range Functionality (!trigr)



23.16. `trigp` (Trigger Position List Entry)

Syntax: `!trigp` or `?trigp`
Parameter: none or `x, y, z, a`
List index and position value

Description: Provides a list of individual, unidirectional trigger positions. The position unit depends on **dim** (mm, μ m, ...). Up to 10000 list entries are possible.

Setting a trigger mode is not required. `!trigp` sets **trigm** to mode 20 (up) or 21 (down) automatically.

The entries are only valid for the specified axis. If the axis has to be changed, all entries must be rewritten to the list.

If no axis is specified, the trigger axis is used (**triga**). If an axis is specified (`x,y,z,a`) with the first entry or when clearing the list, the trigger axis (**triga**) is set to this axis and remains set until changed. Later attempts of change are ignored and an error will be returned.

Trigger signals are generated at the specified position(s). The entries must be either of increasing or decreasing value, as the scan direction is identified by the first two entries. In case of only one list entry, the direction is determined from the current axis position here or when `!trig 1` is set:

- If the trigger position is greater than the axis position, the trigger direction is set to positive (up $\hat{=}$ **trigm** 20).
- If the trigger position is lower than the axis position, the trigger direction is set to negative (down $\hat{=}$ **trigm** 21).

Once set, the position list remains and is reactivated automatically when the axis travels back behind the first trigger position in the list. If required, the `!trigp` list can be loaded with new parameters ahead of every move instruction.

For bidirectional scans, `!trigp` must be filled in the required travel direction before each move (from pos \rightarrow to pos). The move must start from before the first trigger position.

```
!trigp 0      Discard the entire position list  
!trigp x 0   Sets triga to X axis and discards the X list  
!trigp -1 2.5 Append a position to the list (here e.g. 2.5 mm)  
!trigp x -1 9 Only possible if the list is empty (trigc=0):  
                Set the first position and select trigger axis  
!trigp 1 5.3 Set or modify the first list entry (here 5.3 mm)  
!trigp 5 9.1 If entries  $\geq$  5: Replace the 5th list entry  
                If entries = 4: Append entry (like -1 does)  
?trigp -1    Read back the last entry on the list  
?trigp 7     Read back the 7th list entry
```

Remarks: The signal polarity for this special trigger mode can be set by the `!trigl` instruction.

The number of entries can be read by the `trigc` instruction.

In order to apply the signal polarity immediately, the trigger mode can be set to **trigm** = 20 or 21 before setting **trigl**.

Response: Requested trigger position list entry in the current **dim**

Examples for trigp:

--- Preparation ---

- Globally disable the trigger:

```
!trig 0
```

- Select trigger polarity:

```
!trigm 20 (mode 20 or 21 to apply trigl immediately)
!trigl 1 (set trigger signal level to active high)
```

- Specify the axis:

```
( !trigenc 1 ) (optional if the axis provides an encoder)
!triga x (can also be specified or changed by !trigp)
```

- Empty the position list

```
!trigp 0 (if required, appending is also possible)
(remove all existing entries from the list)
```

--- Trigger ---

- A) Trigger in positive direction:

```
!moa 105 (move X to position below the trigger range)
!trigp -1 110.5 (append a position value, here: first entry)
!trigp -1 125.7 (append a position value)
!trigp -1 200.3 (append a position value)
!trig 1 (globally enable the trigger**)
!moa 205 (here: move X past the last entry)
```

- B) Trigger in negative direction:

```
!moa 205 (move X to position above the 1st entry pos)
!trigp -1 200.3 (append a position value, here: first entry)
!trigp -1 125.7 (append a position value)
!trigp -1 110.5 (append a position value)
!trig 1 (globally enable the trigger**)
!moa 105 (here: move X past the last entry)
```

- C) Single trigger position:

```
!moa 105 (move X before the position)
!trigp -1 110.5 (set only one trigger position)
!trig 1 (globally enable the trigger**)
!moa 115 (move X to position past the trigger range)
```

```
!trigp y 11.5 (set y as trigger axis, overwrites !triga)
!trigp -1 11.5 (use recently set axis for trigger)
?trigc (read the number of list entries)
?trigp 1 (read the first trigger position entry)
?triga y (read the trigger axis)
```

** The trigger should be globally disabled for the initial configuration and while filling or modifying the trigger position list.

During a scan it is not necessary to disable/enable the trigger, as the trigger re-enables itself when the axis moves before the first position.



More examples for trigp:

```
!trig 0
!trigm 20
!trigl 1
!trigenc 1
moa 5
```

Globally disable the trigger
mode 20 or 21 to apply trigl immediately
set trigger signal level to active high
optional, set the encoder as trigger source

position the x axis below the 1st triggerpos

EXAMPLE1 (strict)

EXAMPLE2 (alternate)

```
!triga x
!trigg 0
!trigg 1 10
!trigg 2 10.5
!trigg 3 11
!trigg 4 11.5
!trigg 5 12
1)
!trigg 6 20
!trigg 7 20.5
!trigg 8 21
!trigg 9 21.5
!trigg 10 22
1)
!trigg 11 30
!trigg 12 30.5
!trigg 13 31
!trigg 14 31.5
!trigg 15 32
```

```
!trigg x 0
!trigg -1 10
!trigg -1 10.5
!trigg -1 11
!trigg -1 11.5
!trigg -1 12
!trigg -1 20
!trigg -1 20.5
!trigg -1 21
!trigg -1 21.5
!trigg -1 22
!trigg -1 30
!trigg -1 30.5
!trigg -1 31
!trigg -1 31.5
!trigg -1 32
```

Select trigger axis and erase position list

Insert 1st trigger position

...

...

Insert last trigger position

The following instructions would now deliver:

```
?trigc => 15
?trigi => 1
?trigg 3 => 11.0000
?trigg x 3 => 11.0000
?trigg -1 => 32.0000
```

read the amount of entries
read the list index (here: at 1st entry)
read 3rd entry of trigger axis (here: X)
read 3rd entry of X (redundant information)
read last list position

It is possible to overwrite entries, e.g. !trigg 11 30.25

It is possible to shorten the list, e.g. !trigg 10 (=positions 30...32 mm deleted)

It is possible to append positions, e.g. !trigg -1 40.225

Which is the same as ?trigc → 15 !trigg 16 40.225

```
!trig 1
moa 37
```

Globally enable the trigger

Travel x past the last trigger position

Now 15 triggers are generated (examples here: 3 regions of interest with individual gaps inbetween).

1)

Important note: Sending a !trigg list to the TANGO cannot be done in one block. As with all instruction sequences, it must be ensured that the 256 byte input buffer does not run over. Good practice might be to read back "err" or ?trigg after each !trigg position, at least every few !trigg positions. This will ensure no data gets lost during transmission.

23.17. **trigc (Number of Trigger Position List Entries)**

Syntax: !trigc or ?trigc

Parameter: none or [0 to trigc]

Description: Read the number of position list entries, which are made by **!trigp** or **!trigr**. Or reduce the position list size by specifying a lower amount of entries.

Remarks: Only for special trigger functionality of **!trigp** and **!trigr**.

Response: Amount of trigger position list entries.

Examples: ?trigc (read the amount of trigger position list entries)
!trigc 0 (clear the trigp position list)
!trigc 5 (reduce the entries to 5, only possible if greater)

23.18. **trigi (Trigger Position List Index)**

Syntax: !trigi or ?trigi

Parameter: none or 1 to 10000 (less or equal to the current **trigc** count)

Description: This instruction reads or sets the trigger position list pointer of the selected trigger axis (**triga**). The behavior is similar to **snsi** of the snapshot array, except the index here is consistent from 1 to N (**snsi** is 0...N-1).

In trigger modes 20 and 21 this pointer access can be used to read back where in the position list the trigger unit currently is. Or it can be used to manipulate the list index (position pointer) of the entries made by **trigp** or **trigr**.

The value must not exceed the amount of **trigc** list entries.

Response: Current trigger position list pointer

Example:

?trigi Read the current trigger position list pointer (e.g returns 1)
!trigi 1 Set position list pointer to the first element
!trigi Same as !trigi 1
!trigi 21 Set position list pointer to the 21st element

23.19. `trigl` (Trigger Level)

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

Description: Select the trigger signal polarity for trigger modes 20 and 21 (**!trigr**, **!trigg** functionality).

0 = Trigger signal is active low
1 = Trigger signal is active high

Remarks: Only for special trigger functionality of **!trigg** and **!trigr**. For all other cases, the polarity is defined by their **trigger mode**.

Response: Trigger signal polarity for trigger modes 20 and 21.

Examples: `!trigl 0` (set to active low)
`!trigl 1` (set to active high)
`?trigl` (read trigger level, e.g. returns 1 = active high)

23.20. `trigsns` (Trigger from SnapShot Input)

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

Availability: TANGO PCI-E/Desktop-E and Desktop HE, from Firmware 1.73.

Description: Generate a Trigger output signal when the AUX I/O SnapShot input changes from inactive to active. The input polarity can be set by **sns1**.

0 = Trigger on SnapShot disabled (default)
1 = Trigger on SnapShot enabled

Remarks: The `trigsns` function is available in each trigger mode. It adds trigger signals when a SnapShot event is detected. The SnapShot functionality does not have to be enabled. Only the default, dedicated AUX I/O SnapShot input can be Used. Other inputs assigned by **adiginfunc** are not used.

The SnapShot input is updated in a 40µs interval. The input signal must be high for >40µs or low for >40µs in order to be safely detected. The trigger signal will be generated a few µs after Detection.

Response: Enable State of the SnapShot→Trigger function (0 or 1).

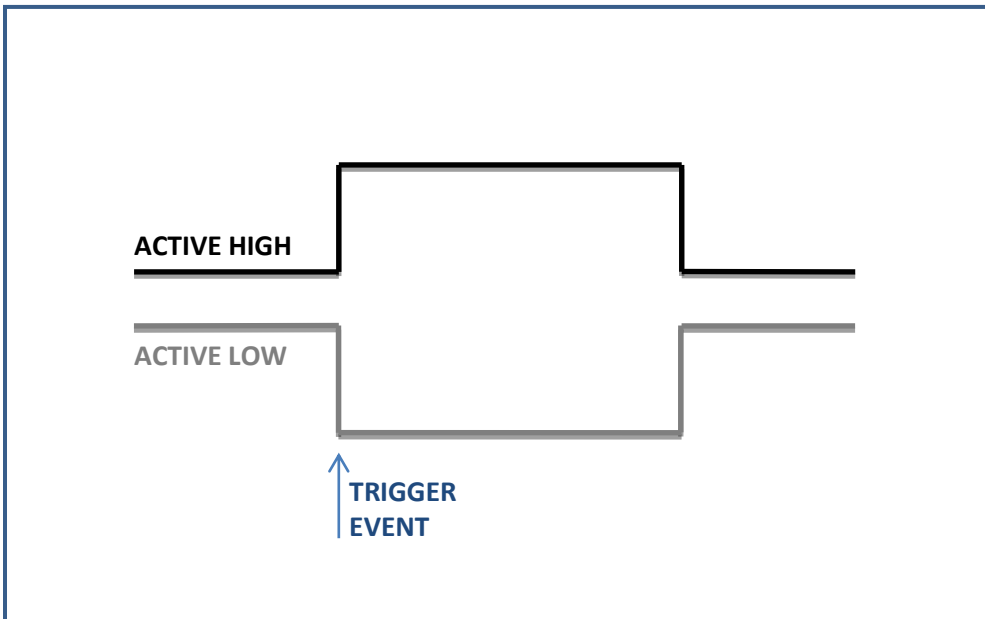
Examples: `!trigsns 1` (enable)
`!trigsns 0` (disable)
`?trigsns ==> 0` (read the current state, here: disabled)

24. The Second Trigger Signal Output

24.1. Introduction

TANGO PCI/PCI-S/PCI-E TANGOS, Desktop TANGOs and the TANGO 3 mini provide two trigger output signals on the AUX I/O connector (TRIGGER_OUT, TAKT_OUT). Only the PCI-E, Desktop-E and TANGO 3 mini support the full functionality as described in this chapter.

Depending on the **trigger mode**, the output signals are active high or active low. This description shows the trigger signals in active high mode.



Up to two output signals are available and can be selected with the `'trigo'` instruction:

- TRIGGER_OUT (here called OUT1)
- TAKT_OUT (here called OUT2)

OUT2 is provided by PCI/PCI-E or Desktop TANGOs and TANGO 3 mini only.

Both outputs share the same polarity setting and trigger source.

The optional LED100 illumination uses OUT2.

The minimum OUT1 width of 40µs results in 12.5 kHz max. signal frequency. If the OUT1 signal width is set to zero, a short spike is still generated. This allows edge trigger events of up to 25 kHz.

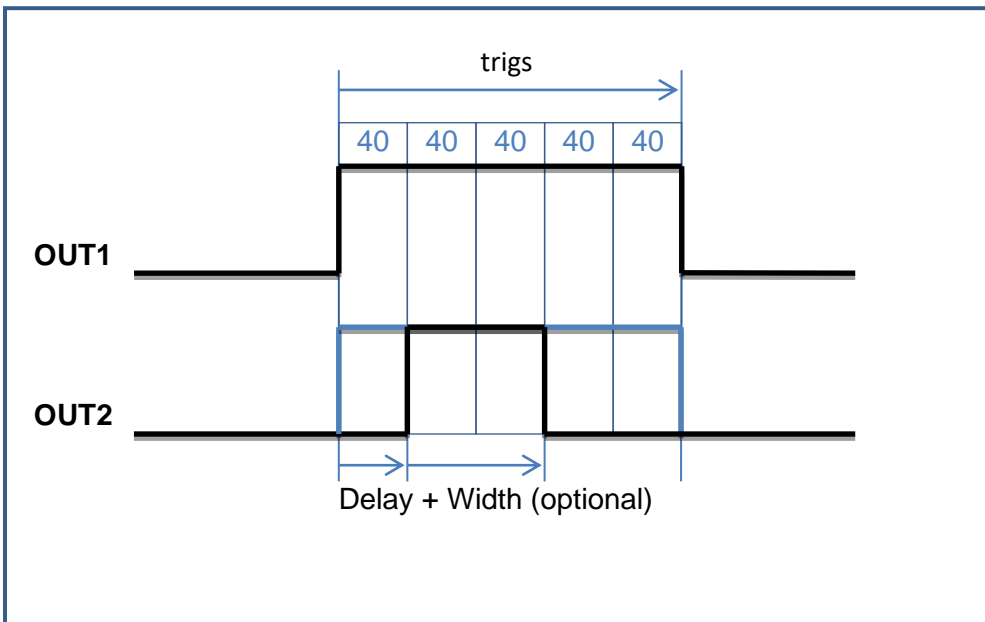
24.2. Standard 1:1

Signal: OUT2 provides the same signal as OUT1 plus optional delay and width

Options: When the **trigs** signal width instruction is sent with 3 parameters, OUT2 provides the option of a delay and individual width. OUT2 becomes inactive latest when OUT1 returns to inactive state. OUT2 must be within the active range of OUT1.

Remarks: Can be used with any **trigger mode**.
 Parameter in increments of 40µs (0; 40; 80; ... 2,500,000)
 Values inbetween the 40µs get rounded to nearest, e.g. <20=0, ≥20=40

Application: Camera shutter and LED flash



OPTIONS

- `!trigo 1` → OUT1 only, OUT2 set to inactive level
- `!trigo 2` → OUT2 only, OUT1 set to inactive level
- `!trigo 3` → OUT1 and OUT2

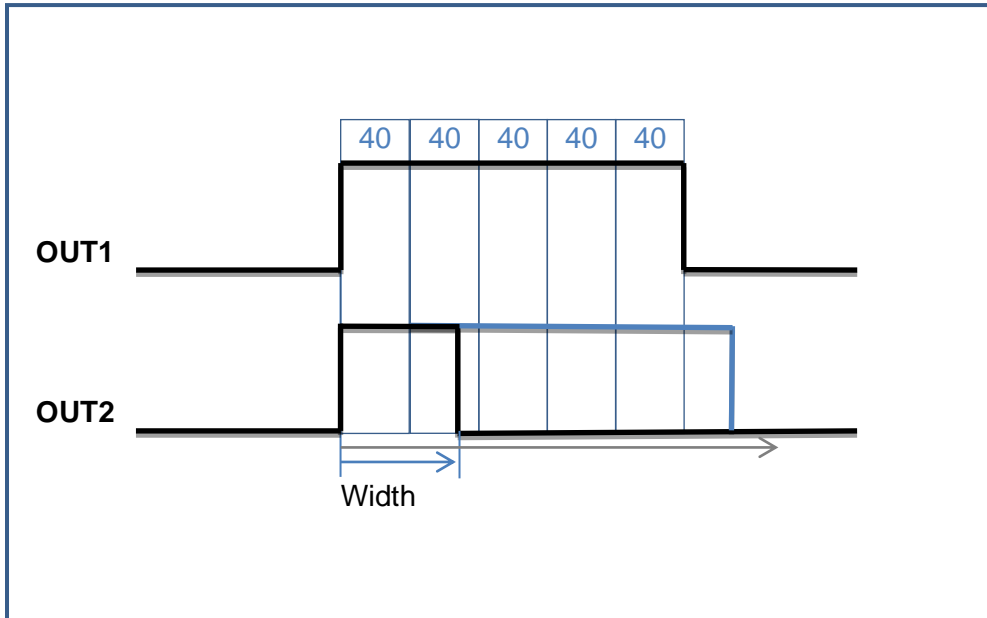
EXAMPLE

- `!trigo 3`
- `!trigs 200 40 80` → Set OUT1 signal width & maximum duration (200µs), Delay (40µs) and Width (80µs) for OUT2
- or `!trigs 200` → Set OUT1 and OUT2 signal width to 200µs

24.3. Precise Width

Signal: OUT2 starts with OUT1 but has individual, high resolution width.

Remarks: Can be used with any **trigger mode** of OUT1.
OUT2 resolution is in 7.6ns steps, up to 32.5 seconds.
OUT2 width can be longer than OUT1.



OPTIONS

`!trigo 5` → OUT1 only, OUT2 set to inactive level
`!trigo 6` → OUT2 only, OUT1 set to inactive level
`!trigo 7` → OUT1 and OUT2

EXAMPLE

`!trigo 7`
`!trigs 200`
`!trigbwidth 67.4` or `!trigbwidth 230`

24.4. Precise Delay

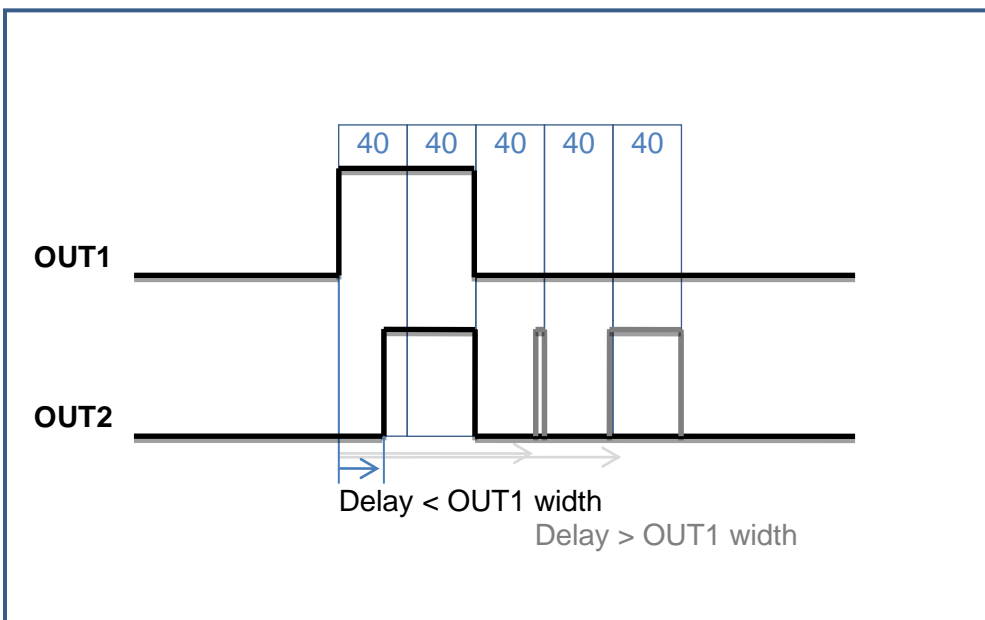
Signal: OUT2 provides a precise edge delay to OUT1.

Remarks: Can be used with any trigger mode of OUT1.
Very low jitter between the two edges OUT1 → OUT2 < 10ns.
OUT2 resolution is in 7.6ns steps, up to 32.5 seconds.
OUT2 delay can be longer than (past) the OUT1 signal width.

If the OUT2 pulse starts at last 1µs before the OUT1 pulse ends, then the OUT2 pulse is resetted with the OUT1 pulse.

Else the OUT2 pulse is resetted with the next 40µs interval.
In this case, OUT2 signal widths of 100ns to 40µs may occur which makes only sense in edge triggered applications. Refer to the third OUT2 pulse "`!trigbdelay 159.7`" in the figure below.

Application: Precisely delayed trigger edges for transducer and receiver



OPTIONS

`!trigo 9` → OUT1 only, OUT2 set to inactive level
`!trigo 10` → OUT2 only, OUT1 set to inactive level
`!trigo 11` → OUT1 and OUT2

EXAMPLE

`!trigo 11`
`!trigs 80`
`!trigbdelay 25.5` or `!trigbdelay 112.03` or `!trigbdelay 159.7`

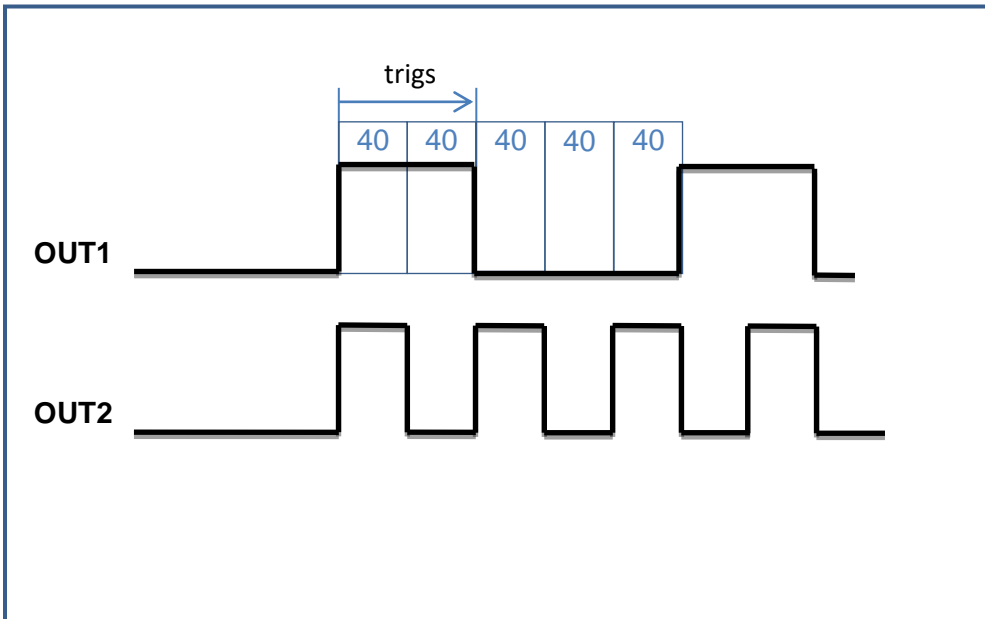
REMARKS

The TANGO PCI-S and Desktop-S controllers provide limited delay functionality. Here only the leading edge (active high = rising, active low = falling edge) of OUT2 is valid. The signal level inbetween is on active level (not inactive as shown above). When using those controllers, the delay function only makes sense with edge sensitive devices and not with level sensitive ones.

24.5. Precise Frequency

Signal: OUT1 and OUT2 provide individual output frequencies.
OUT2 can be used for higher frequencies and high resolution.

Remarks: OUT1 can run in an individual trigger mode (position dependent, etc.)
OUT1 width can be specified by `'trigs'`
OUT1 frequency is limited to 0.01 to 25000 Hz and 1/40µs resolution
OUT2 provides 0.010...66,000,000 Hz high resolution and low jitter
OUT2 has fixed width of 50%
OUT1 and OUT2 are in phase only if the frequencies match (multiples)



OPTIONS

```
!trigo 13 → OUT1 only, OUT2 set to inactive level
!trigo 14 → OUT2 only, OUT1 set to inactive level
!trigo 15 → OUT1 and OUT2
!trigm can be set to an independent behavior, must not be frequency output
```

EXAMPLE

```
!trigo 15
!trigs 80
!trigf 5000
!trigbf 12500
!trigm 100
```

25. Snapshot – The Trigger Input Functionality (option)

Snapshot functionality must be configured by factory.

To identify if the Snapshot functionality is configured, use `?det` or `detext`.

The snapshot functionality allows capturing of X,Y,Z,A axis positions by either pressing a HDI key (e.g. Joystick F2), an external signal or a trigger-out event. Hardware dependent, analog input signals from the AUX I/O connector are captured as well. The values are appended to the snapshot array. The snapshot array can also be filled, extended or manipulated by `snsa`, `!snsp` instructions. Please refer to `snsm` for available snapshot modes.

In `snsm 1` or `5` a snapshot event can command the TANGO controller to move to positions stored in the snapshot array. Every snapshot event goes to the next array entry and wraps around at the end of the array. It can i.e. be used to move to positions (points of interest) captured in snapshot mode `snsm 0` or `4`.

In `snsm 6`, `moa`, `mor`, `speed` and `go` instructions wait for a snapshot event on the specified `snsaxis`.

The snapshot event can be triggered by either

- A) the Joystick key F2
- B) via the SnapShot signal input (depends on hardware, AUX I/O connector option)
- C) by a software instruction (`snse`)

The position unit depends on the selected dimension (`dim`). The setting of `encpos` defines if the motor position or the measured encoder position is read from the array. A maximum of 1024 snapshot positions can be captured.

For changing the snapshot configuration please disable the snapshot (`!sns 0`) and then re-enable it (`!sns 1`) after all settings have been made.

Remarks:

Most of the snapshot settings cannot 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).

As the snapshot signal is sampled every 160µs, it should have an active and inactive time of at least 200µs. Else the signal may not be recognized by the controller. Also refer to '`snsf`' for the debounce filter time.

Example: Three snapshot positions are captured with a 3 axis TANGO

Index	Position X	Position Y	Position Z	Position A
1	1.0000	1.2345	1.2345	0
2	2.1200	1.3520	0.9343	0
3	3.5900	1.9000	0.8341	0
4	<i>invalid</i>	<i>invalid</i>	<i>invalid</i>	<i>invalid</i>
5	<i>invalid</i>	<i>invalid</i>	<i>invalid</i>	<i>invalid</i>
...
1024	<i>invalid</i>	<i>invalid</i>	<i>invalid</i>	<i>invalid</i>

Here: `?snsc → 3 ?snsa 2 → 2.1200 1.3520 0.9343 ?snsa y 1 → 1.2345`

25.1. sns (Snapshot enable/disable)

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

Description: Globally enables or disables the snapshot functionality. Firmware versions until 1.72, snapshot is globally enabled by default (sns=1 at power-on). From Firmware 1.73, the default is off and can be configured by snspreset. Latched key pressed events or the **snsm=6** waiting state of a move is cleared when enabling/disabling the Snapshot (sns0,1).

0 = disabled
1 = enabled

Response: Snapshot state

Examples:
!sns 0 Disable snapshot
?sns Read snapshot enable state

25.2. snspreset (Preset for sns)

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

Description: Set or read the predefined power-up setting for **sns**. The TANGO will use it as preset value for sns after power-up or reset. Default = 0.

Remarks: Firmware Versions before 1.73 use sns = 1 as default, meaning the snapshot is globally enabled at power-up. From Firmware 1.73 it is possible to define the value for power-up/reset with snspreset. New Default = 0.

Response: Selected sns enable state for power-up.

Examples:
!snspreset 0 Set power-up preset for sns to 0 (globally disabled)
?snspreset Read the sns preset value

25.3. sns1 (Snapshot Level / Polarity)

Syntax: !sns1 or ?sns1
Parameter: 0 or 1

Description: This instruction sets the snapshot input signal polarity.
 0 = active low (event detected at falling edge)
 1 = active high (event detected at rising edge)

Response: Currently used snapshot polarity

Examples:
!sns1 0 Set snapshot input to active low (default)
?sns1 Read current snapshot input polarity

25.4. snsf (Snapshot Filter)

Syntax: !snsf or ?snsf
Parameter: 0 to 255 [ms]

Description: Reads or set the snapshot filter time in ms, which is used
 to debounce the snapshot signal (HDI F2 button or snapshot
 input). Default is 10ms. The filter time does not delay the
 reaction to the signal, it only delays the next detection
 of a signal change.

Requirements: The minimum high and low times of the input signal must be
 greater or equal to 200µs to be recognized.

Remarks: Debouncing is recommended for push-buttons, switches, etc.
 For digitally generated signals, the filter time can be 0.

Response: Snapshot filter time

Examples:
!snsf 0 Disable input filter
!snsf 10 Set snapshot filter time to 10 ms
?snsf Read snapshot filter time in [ms]

25.5. snsrm (Snapshot Mode)

Syntax: !snsrm or ?snsrm

Parameter: 0, 1, ... 12

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

- 0 = Capture positions to list with Joystick key F2
- 1 = Move forward through position list with Joystick key F2 (wraps around at the last element)
- 2 = Extended move with Joystick keys:
 - F1: Move backward through position list (wraps around)
 - F2: Move forward through position list (wraps around)
 - F3: Move to '**prehome**' then start of list (first element)
 - F4: Move to '**prehome**' with '**vel**' then '**home**' with '**sevel**'
- 3 = Simple continuous path control in conjunction with pre-loaded **snsa/snsp** positions (dissection mode). The path travel velocity is set by '**scanvel**'. Start with Joystick key F2 or corresponding '**snsr 2**'. HDI must be enabled by **joy** and **joydir** = 2.
- 4 = Like mode 0, but snapshot I/O input is used instead of F2
- 5 = Like mode 1, but snapshot I/O input is used instead of F2
- 6 = Triggered start: Move and Speed instructions will wait for the I/O connector snapshot event to start moving. Axes must be assigned by the '**snsaxis**' instruction.
- 7 = Custom HDI mode with Prehome, Home and auto increment in a time-interval set by the '**delay**' instruction
- 8 = A Trigger-OUT event (not a snapshot event) captures axis positions and the ANIN0 analogue signal (of the optional AUX I/O connector, available with DT and PCI/PCI-E TANGOs)
- 9 = Move relative '**snsj**' Jump distances with Joystick keys F2 = jump, F1 = jump in reversed direction (back)
- 10= Like mode 9, but snapshot I/O input is used instead of F2
- 11= Z axis follows an X-Z position list (e.g. focus for scan)
- 12= A axis follows an X-Z position list (e.g. focus for scan)

Remarks: F1-F4 key events can also be generated by '**snsr**' 1 to 4.

Position capture modes (0,4,8) also capture the analog voltage at the AUX I/O ANIN0 input pin. It can be read by '**?snsv**'.

A snapshot capture is executed on all active axes. If the snapshot array is filled by **snsa**, it is possible to fill in individual axes x, y, z, a and let only those travel.

When setting snsrm Snapshot mode, latched key events (**key1**) or a pending snapshot mode 6 waiting state is cleared.

Response: Currently selected snapshot mode

Examples: !snsrm 0 (Set snapshot mode to capture with joystick key F2)
!snsrm 1 (Set snapshot mode to move with joystick key F2)
!snsrm 8 (Set snapshot mode to capture at trigger out events)
?snsrm (Read current snapshot mode)

25.6. Snapshot Mode Description and Examples

Snapshot Mode 1,2,3,7,9 move functions, accessible via Joystick F1-F4 keys or !snse 1-4 instruction:

Key	Mode function				
	1	2	3 (DISSECTION)	7 ***	9
F1 =	-	previous point**	-	prehome* & home	jump back
F2 =	next point	next point**	start dissection	auto inc from 1 st point	jump fwd
F3 =	-	prehome*& first point	-	pause/continue	-
F4 =	-	prehome*& home	-	pause & previous point	-

* Remarks: A move to the "prehome" positions always travels at **sevel**.

Prehome & home means that the axes move to prehome and from there to home. This is used to avoid possible collisions on the way to the home position.
Prehome & first point means that the axes move to prehome and after reaching directly to the first position in the position list. This sequence is also for collision prevention.

If prehome is not required, it can be set to the same position as home.
 For graphical explanation of home & prehome positions, refer to Figure 1 below.

** SnapShot mode 2:

F3 function must be executed in order to enable the F1 and F2 functionality
 F4 disables the F1 and F2 functionality, so pressing F3 is required again

Instruction sequence example for snsm 2:

```
!cal
!rm
!sns 0
!prehome [Xpos] [Ypos] [Zpos] [Apos] (specify a redirection to avoid obstacles)
!home [Xpos] [Ypos] [Zpos] [Apos] (specify home, e.g. load, unload position)
!snsa 0 (clear the snapshot array / position list)
!snsa 1 [Xpos] [Ypos] [Zpos] [Apos] (load position list, at least one entry)
!snsa 2 ... (load position list) ...
... !snsa N ...
!snsm 2 (enter snapshot mode 2 for F1-F4 functionality)
!sns 1 (activate snapshot function globally)
```

*** SnapShot Mode 7:

1. Execute a !cal and !rm instruction to define the repeatable, absolute position (origin).
2. Define the !prehome and the !home positions.
3. Define a position list of points using the !snsa instruction.
4. Specify a user delay between the positions by using the !delay instruction.
5. Enter snapshot mode 7 by sending the !snsm 7 instruction.

All information is lost when the TANGO controller is switched off or resetted.
 The delay also applies to all move instructions. So when leaving the snapshot mode 7 it is recommended to reset the delay value back to zero (!delay 0).
 Snapshot mode 7 uses the joystick keys F1 to F4 according this specification:

- F1 = Go to load/unload position (as defined with the !home and !prehome instructions)
- F2 = Start Auto Movement from Point 1 (through position list, defined by the !snsa instruction and using a delay as defined by the !delay instruction)
- F3 = Pause/Continue from current position (halt the automatic move)
- F4 = Move back one position in the list (when in pause, else it is paused after reaching the next position and then it moves back.)

Instruction sequence example for snsm 7:

```
!cal
!rm
!sns 0
!delay 3000 (delay between moves, e.g. 3 seconds)
!prehome [Xpos] [Ypos] [Zpos] [Apos] (specify a redirection to avoid obstacles)
!home [Xpos] [Ypos] [Zpos] [Apos] (specify home, e.g. load, unload position)
!snsa 0 (clear the snapshot array / position list)
!snsa 1 [Xpos] [Ypos] [Zpos] [Apos] (load position list)
!snsa 2 ... (load position list) ...
... !snsa N ...
!snsm 7 (enter snapshot mode 7 for F1-F4 functionality)
!sns 1 (activate snapshot function globally)
```

Autoincrement can be aborted by the **abort (a)** instruction.

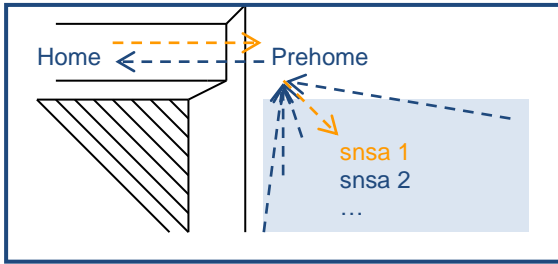


Figure 1: home & prehome (in modes 2 and 7)

SnapShot Mode 3 (dissection, simple continuous path functionality):

Typical applications are laser dissection or evaporation, or adhesive dosers/dispensers.

Remarks: All axes must be idle and all axis positions must be loaded with valid entries.

Joy and joydir must be enabled (e.g. set to 2) for all required axes (as with !speed).

The continuous path functionality has a continuous increase in position deviation

which will build up during the sequence. The interpreter is blocked during execution.

```

!scanvel [mm/s]                (set the path/vector velocity)
!sns 0
!snsa 0                          (clear the snapshot array / position list)
!snsa -1 [Xpos] [Ypos] [Zpos] [Apos] (write first entry to position list, here: append)
?err                             (read back if accepted, also prevents buf.overflow)
!snsa -1 [Xpos] [Ypos] [Zpos] [Apos] (continue as above for all entries ...)
?err
...
!sns 3                          (enter snapshot dissection mode 3)
!sns 1                          (activate snapshot function globally)
!snse 2                         (start the dissection sequence, or press F2 key)
?err                             (send a request and wait for reply → completed)

```

SnapShot Mode 11 (X-Z focus list):

In snapshot mode 11, Z follows a position list dependent on the X axis position.

Typically used for focus applications under high magnification.

Best performance with piezo Z-stages, which are able to follow more precisely and at higher velocities.

The snapshot array can be loaded with X-Z position pairs of increasing X value.

The X distances can be random, no equal distance or match with e.g. trigger distances required.

When snsm 11 is activated, Z travels at interpolated positions between the X points.

X positions below the first snapshot array entry will cause Z to remain at the first Z position.

X positions higher than the last entry will cause Z to remain at the last Z position.

The Z positioning will begin as soon as the snapshot mode is activated.

Z then is controlled by the snapshot array position list.

The snapshot array positions of Y and A axes are ignored in snsm 11.

```

!sns 0
!snsa 0                          (clear the snapshot array / position list)
!snsa 1 [Xpos] [don't care] [Zpos] (load X-Z position list first entry)
!snsa 2 ...                      (load X-Z position list) ...
... !snsa N ...
!sns 11                          (enter X-Z focus mode)
!sns 1                          (activate snapshot function globally)
?err                             (send a request and wait for reply → completed)

```

```

Example: !snsa 1 5.1 0 2.33
         !snsa 2 7.1 0 5
         !snsa 3 11.7 0 1.65

```

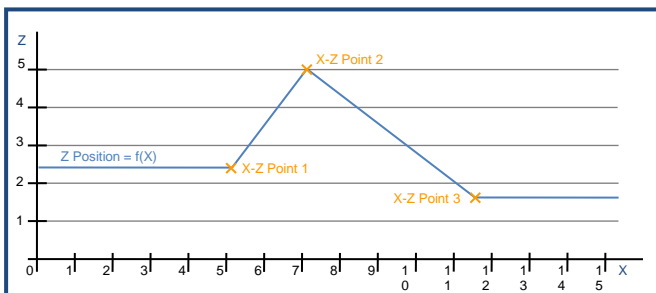


Figure 2: X-Z focus application (snapshot mode 11)

25.7. snsc (Snapshot Counter)

Syntax: !snsc or ?snsc
Parameter: none, 0 or 1 to 1024 (not greater than the current snsc count)

Description: This instruction reads or sets the snapshot counter, which shows the snapshot array entries (counted snapshot events). This instruction may also be used to reset the counter to zero or reduce the size of an existing array.

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

Example:
?snsc Read the number array entries (e.g. captured snapshots)
!snsc Clear snapshot counter to zero elements
!snsc 0 Clear snapshot counter (same as !snsc without parameter)
!snsc 50 Reduce snapshot array entries to 50 (only if the current snsc value is equal or higher than the new value)

25.8. snsi (Snapshot Index)

Syntax: !snsi or ?snsi
Parameter: none or 0 to 1023 (less than the current snsc count)

Description: This instruction reads or sets the snapshot array pointer.

In snapshot modes 1 and 5 this pointer access can be used to manipulate the array index where move target positions are read from.

Opposed to other snapshot instructions the index range here is [0...N-1] instead of [1...N]. The value must be less than the amount of **snsc** array entries.

Remarks: The pointer is only used for the snapshot move. It does not define the array index where positions are written to. Captured positions are always appended as the last element of the array. For array manipulations by software please refer to '**snsa**' and '**snsr**'.

Response: Current snapshot array pointer

Example:
?snsi Read the current array pointer position
!snsi 0 Set array pointer to the first element
!snsi Same as !snsi 0
!snsi 20 Set array pointer to the 21st element

25.9. snsaxis (Snapshot Axis)

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

Description: Defines which axes should wait in Snapshot mode 6.
 Please refer to '**sns**' for further information.
 When sending a moa, mor or speed instruction with sns 6,
 axes can be specified that should not execute the move until
 triggered by a snapshot-in event. This can be used in cases
 where precise start timing has to be controlled by an external
 signal (maybe also set the '**trigf**' value to zero then).

1 = move or speed of this axis will wait for snapshot signal
0 = axis does not wait for snapshot signal (default operation)

Remarks: If not all axes are set to waiting mode, a vector move may
 start some of the axes while the others are waiting.
 Such problem does not exist when using single axis move/speed
 instructions.

Response: Wait mode setting of the axes

Example:

```
!snsaxis 1 0 1     Set axis X, Z to waiting mode, Y does not have to wait
!snsaxis y 1      Set axis Y to waiting mode
?snsaxis          Read waiting mode of all available axes (e.g. returns "0 0 0")
?snsaxis z        Read waiting mode of Z axis only (e.g. returns "1")
```

25.10. snsp (Snapshot Position)

Syntax:	!snsp or ?snsp
Parameter:	x, y, z, a or none Positions (unit depends on ' dim ')
Description:	Read the last captured Snapshot Position or append a new snapshot array position. The snapshot counter ' sns ' is incremented automatically.
Remarks:	Only motor positions can be read. For reading a captured encoder position, please refer to ' ?snsa -1 '.
Response:	Last captured Snapshot position(s) of the specified axis or of all available axes. The unit depends on ' dim '.
Examples:	
?snsp	Read last captured snapshot position of all axes
?snsp z	Read last captured snapshot position of Z axis only
!snsp 8.5 50.2	Append a new X and Y entry (then only X and Y axes will be used for positioning, Z and A will remain at their positions)
!snsp y 20.5	Append a new position entry with Y-axis position only

25.11. sns (Snapshot Array)

Syntax: !snsa or ?snsa
Parameter: x, y, z, a or none
-1, 0 or 1 to 1024 and up to 4 positions in the current '**dim**'

Description: Read, fill, modify or clear the snapshot position array, which may contain up to 1024 entries.
For reading a valid entry, the index must have a value between 1 and the snapshot counter value '**?snsc**'.

<u>Index</u>	<u>Function</u>
-1	read the last entry or append a new entry
0	clear the entire array (also sets snsc to 0)
1...1024	r/w access to the corresponding array entry
[snsc +1]	like -1, an index of the number of entries +1 can be used to append a new entry

Remarks: Captured positions can be read as motor- or encoder positions, depending on the '**encpos**' setting. Axes without encoders or array entries written by software (via !snsa or !snsp) only return the motor position.
When the snapshot array is used to position axes on snapshot events (e.g. by **snsm 1**), axes will travel as follows:
If the array was filled by capturing positions (e.g. **snsm 0**), Then all axes will travel to the captured position.
If the array was filled by software, via !snsa or !snse, then only the specified axes will travel, others are not affected. Refer to examples below.

Response: Snapshot array position(s) in '**dim**' units.

Examples:
!snsa 0 Clear the entire snapshot array
!snsa -1 8.5 50.2 Append a new entry X+Y → for this entry, only X+Y axes will travel
!snsa z -1 20.5 Append a new entry for Z → here only Z will travel

!snsa 2 50 50 2.8 7 Set or overwrite all 4 axis positions of the 2nd array entry

?snsa 1 Read all axis positions of the first snapshot entry
?snsa -1 Read all axis positions of the last snapshot entry
?snsa z 99 Read the Z-axis position of the 99th snapshot entry
?snsa y -1 Read the Y-axis position of the last snapshot entry

25.12. snse (Snapshot Event)

Syntax: !snse or snse
Parameter: 1, 2, 3 or 4

Description: Executes HDI F-key Snapshot functions via the communication interface. Instead of pressing a Joystick button or using the AUX I/O signal, the functions can be triggered by software:

1 = Function normally executed by pressing Joystick F1 key
2 = Function normally executed by pressing Joystick F2 key
or using the AUX I/O SnapShot input
3 = Function normally executed by pressing Joystick F3 key
4 = Function normally executed by pressing Joystick F4 key

Remark: Behavior is the same as with the function keys. It depends on the snapshot mode settings and only works when Snapshot is enabled.

Response: -

Example:
snse 2 Execute F2 Snapshot event (e.g. capture current position to the snapshot position array)

25.13. sns v (Snapshot Voltage)

Syntax: !sns v or ?sns v
Parameter: 1 to 1024 or -1 to -1024

Description: Read the AUX I/O ANIN0 input voltage (0...5V), captured at the specified position array index. The value can also be written to (i.e. as temporary data storage)
Index > 0: read Voltage in [mV]
Index < 0: read raw ADC data

Remarks: The specified index must be in the range of ± 1 to $\pm ?snsc$. The [mV] values are always calculated with the internal voltage reference, while the raw data is the direct ADC sampling result which may be less accurate (only PCI-E based TANGOs have an accurate ADC sampling result). The raw data range is always 12bit, but with PCI-S based controllers the 2 LSBs are always zero (10bit resolution).

Response: Voltage in [mV] or in [12bit ADC raw data]

Example:
?sns v 2 Read captured Voltage [mV] of 2nd snapshot array entry
?sns v -2 Read captured Voltage [raw] of 2nd snapshot array entry
!sns v 5 1234 Store the value 1234 in 5th snapshot array voltage element

25.14. snsj (Snapshot Jump)

Syntax: !snsj or ?snsj
Parameter: x, y, z, a or none
 Positions (unit depends on 'dim')

Description: Sets the jump distance for snapshot-initiated relative moves. The unit of the position depends on the 'dim' setting. Depending on the snapshot mode '**snsm**', the jump is executed either by HDI (e.g. Joystick) function keys or via the AUX I/O snapshot input:

snapshot mode **snsm** 9:

- HDI F2 key = jump (move relative) in the specified direction
- HDI F1 key = jumps in the opposite direction (backwards)
- The **snse** 1 and **snse** 2 instructions can be used also

snapshot mode **snsm** 10:

- AUX I/O snapshot input jumps in the specified direction only

Response: Position value(s)

Examples:

```
!snsj z 1.5               Set the jump distance for Z axis to 1.5 (mm if dim= 2 or 9)
!snsj 1 -10 0 0.2        Set the jump distance for all axes (here Z does not move)
?snsj y                  Read the jump distance of the Y axis
?snsj                    Read the jump distance of all axes
```

25.15. prehome (Snapshot PreHome Position)

Syntax: !prehome or ?prehome

Parameter: x, y, z, a or none

Description: This instruction sets the prehome position used by the snapshot extended move. The unit of the position depends on the setting of '**dim**'.

Remarks: Used in snapshot modes (**sns**m) 2 and 7.

Response: Position value(s)

Examples:

```
!prehome y 10.2 Set prehome position Y-value to 10.2 (e.g. [mm] when dim=2)
!prehome 10 0 20 Set prehome position X,Y,Z
?prehome x Read currently used prehome position of X axis only
?prehome Read currently used prehome positions of all axes
```

25.16. home (Snapshot Home Position)

Syntax: !home or ?home

Parameter: x, y, z, a or none

Description: This instruction sets the home position used by the snapshot extended move. The unit of the position depends on the setting of '**dim**'.

Remarks: Used in snapshot modes (**sns**m) 2 and 7.

Response: Position value(s)

Examples:

```
!home y 10.2 Set home position Y-value to 10.2 (e.g. [mm] when dim=2)
!home 10 0 20 Set home position X,Y,Z
?home x Read currently used home position of X axis only
?home Read currently used home positions of all axes
```



26. Appendix A – anain options of different TANGOs

Ch.	Signal Name/Controller	Desktop HE	PCI-E DT-E	PCI-S DT-S	3 mini	Integrale	TANGO-C Pilot
0	HDI IN1A (Joystick X)	-	X	X	-	-	X
1	HDI IN2A (Joystick Y)	-	X	X	-	-	X
2	HDI IN3A (Joystick Z)	-	X	X	-	-	X
3	HDI IN4A (Joystick A)	-	X	X	-	-	X
4	HDI Speedpoti	-	X	X	-	-	X
5	HDI IN1B	-	X	X	-	-	X
6	HDI IN2B	-	X	X	-	-	X
7	HDI IN3B	-	X	X	-	-	X
8	HDI IN4B	-	X	X	-	-	X
9	HDI-ID	-	X	X	-	-	X
10	AUX I/O ANIN0	-	X	X	-	EXT 5V	X
11	U-HIP PSE	-	X	X	(X)	U-HDI AN	X
12	V-MOT	-	X	X	(X)	U-HIP	X
13	X-ID0	-	X	X	-	VCC3	X
14	X-ID1 / PCI-S: Temp.	-	X	X	-	V-DSP	X
15	REF (2.5V =512 digit)	-	X	X	(X)	V-ENC	X
16	5VEXT1 HDI	-	X	-	(X)	V-MOT	-
17	AUX I/O PSE ON 1/0	X	-	-	X	-	-
18	5V USB (from PC) 1/0	X	-	-	X	-	-
19	Powersupply Good 1/0	X	-	-	X	-	-
20	5V Encoder ok 1/0	X	-	-	X	-	-
21	5V external 1 (V)	X	-	-	X	-	-
22	5V external 2 (V)	X	-	-	X	-	-
23	5V HDI (V)	X	-	-	X	-	-
24	Brake Voltage (V)	X	-	-	X	-	-
25	V-BUS 24V (V)	X	-	-	X	-	-
26	VCC5 (V)	X	-	-	X	-	-
27	U-PSE (V)	X	-	-	X	-	-
28	V-MOT (Supply) (V)	X	X	-	X	X	X
29	Input current (A)	X	-	-	X	-	-
30	Input power (W)	X	-	-	X	-	-
31	Input I-peak (A)	X	-	-	X	-	-
32	Coax Drive Sin Cos X	X	X	X	X	X	X
33	Coax Drive Sin Cos Y	X	X	X	X	X	X
34	Coax Offs. Sin Cos X	-	X	X	-	-	X
35	Coax Offs. Sin Cos Y	-	X	X	-	-	X
36	Coax Fact. Sin Cos X	-	X	X	-	-	X
37	Coax Fact. Sin Cos Y	-	X	X	-	-	X
40	Joystick deflection X	X	X	X	X	X	X
41	Joystick deflection Y	X	X	X	X	X	X
42	Joystick deflection Z	X	X	X	X	X	X
43	Joystick deflection A	X	X	X	X	X	X
50	I/O Module Supply	X	X	-	-	-	-
51	I/O Module Supply (V)	X	X	-	-	-	-
52	5V INT (V)	X	-	-	-	-	-
53	5V USB from PC (V)	X	-	-	-	-	-
54	5V USB HOST (V)	X	-	-	-	-	-
55	I-CAN24 Peak (A)	X	-	-	-	-	-
56	Input I-peak since on	X	-	-	-	-	-
60	ANAIN0 (V)	X	-	-	-	-	-

27. Glossary

TANGO	Motion controller for stepper motors, also called « controller ».
PCI-S,DT-S	TANGO controller type for PCI slots. Name is TANGO DT-S if used in TANGO Desktop (TANGO Desktop-S).
PCI-E,DT-E	TANGO controller type for PCI-E slots. Offers greater functionality compared to the PCI-S. Name is TANGO DT-E if used in TANGO Desktop (TANGO Desktop-E).
TANGO mini	A small 2 axis / 1.0 ampere controller with RS232 connector.
TANGO 3 mini	Same size as the TANGO mini, but with up to 3 axes / 1.25A and much more features, performance and functionality than TANGO mini.
Desktop HE	Next generation TANGO Desktop 2HE and TANGO Desktop 3HE controller, replaces TANGO Desktop-E with more options and functionality.
AUX I/O	Optional extension connector for PCI-S and PCI-E based TANGOs and TANGO Desktop. Provides analog and digital I/O, safety and trigger functionality. For TANGO 3 mini please refer to AUX mini.
AUX mini	AUX I/O of the TANGO 3 mini controller.
HDI	Human Device Interface, manual control of the axes, e.g. a Joystick.
LED100	Optional LED illumination unit for microscopes
Multi-IO IO1	Optional I/O connector for DT-E,PCI-E and HE: 12xIN/8xOUT (24/12/5V) Optional I/O connector for DT-E,PCI-E and HE: 24xIN/8xOUT (24/12/5V)
Cal	Causes axes to travel towards the lower limit switch (E0). Usually the zero position is also set by this instruction.
Rm	Causes axes to travel towards the upper limit switch (EE). Usually executed directly after cal, it then disables the secvel velocity limitation.
Secvel	Secure velocity which limits the axis travel velocity as long as no cal and rm has been executed. Ment to protect the axes from damage when traveling too fast into (yet unknown) axis hardware limits.
Snapshot	Trigger input functionality, also available with joystick key F2.
Closed Loop	The axis position is controlled according to a measuring system.
Automatic move	Positioning instructions moa,mor,m,moc,mol,moe (also cal,rm) which cause an autostatus reply. Speed and go behave different.



28. Document Revision History

No.	Revision	Date	Changes	Remarks
01	A	27. March 2012	Newly revised and extended document version	Based on TANGO firmware revision 1.57
02	B	16. April 2012	Added instructions: Ecomove, noled, calrequired, joychangeaxis, ctrsm, ctrs, trigo, trigcomp, trigenc, snsi, snsaxis, sns, zwfactor, detext, posclr, maxaxis, trigbwidth, trigbdelay, trigbf, encrefvel, vrm, flash, etspresent, stagesn Added stop and stoppol modes !joy instruction autostatus behavior	Based on TANGO firmware revision 1.57
03	C	03. May 2012	caldir marked as dummy, naming conventions, version instruction	
04	D	18. July 2012	Remarks concerning closed loop behavior and (current) reduction	
05	-	30. July 2012	digin, digout, diginpol, digintyp, digoutpreset	Based on TANGO firmware revision 1.57
06	Prelim E	06. Sept. 2012	Added: Glossary of common terms, Joystick Key Assignment chart Second Trigger Output description Multi I/O instructions edigin, edigout, ediginpol, edigintyp, edigoutpreset Adapted to naming conventions Improved description of HDI	Based on TANGO firmware revision 1.58
07	E	18. Feb. 2013	Added: anamode, trigr, corrst	Based on TANGO firmware revision 1.60
08	F	25. June 2013	Added I/O communication error 77 ?ver instruction "Ver:" to "Vers:" Improved vrm description	
09	G	23. Aug. 2013	Improved go, speed, vel description	Final documentation of firmware 1.60
10	Prelim H	23. Aug. 2013	Extended functionality of "go" and "vel" instructions	Based on TANGO firmware revision 1.60C
11	H	04. Sept. 2013	Added "!mol" and "lockpos"	Based on TANGO firmware revision 1.60C
12	I	16. Oct. 2013	Added "lpa 2"	Based on TANGO firmware revision 1.58
13	Prelim J	02. Dec. 2013	Added multiple manual trigger pulses description	Based on TANGO firmware revision 1.60C
14	Prelim J	06. Dec. 2013	Added scanmode 3	Based on TANGO firmware revision 1.63
15	J	20. Oct. 2014	Corrected "limctr" description Added new ctrstatus option, Added hdimode options for LED100 Added "zwpos", "enctype", "brake", "trigl" instructions Improved descriptions Added snapshot mode "snsm 9" (jump mode) and "snsj" Added calvel, rmvel with one parameter Mentioned "trigenc" encoder trigger position limitations Added sns10 Extended description of "trigs" Revised closed loop descriptions Added "calzeropos" and "tvr" instructions	Final documentation of firmware 1.65



No.	Revision	Date	Changes	Remarks
16	Prelim K	02. April 2015	Corrected snsm 10 description, added liquid dispenser instructions "drop" and "pump", extended "a" description, added LED100 example for adigout	Based on TANGO firmware revision 1.66
17	Prelim K	21. Sept. 2015	Added "clim" instruction, added hdimode bit 8	Based on TANGO firmware revision 1.66
18	Prelim K	21. Oct. 2015	Extended trigr description	Based on firmware 1.66 and 1.60H/1.60S
19	Prelim K	30. Oct. 2015	Extended speed and velfac description	
20	K	10. Nov. 2015	Document released for TANGO Firmware 1.66	Based on TANGO firmware revision 1.66
21	-	01. Dec. 2015	Corrected trigenc description for TTL Added lock and lockstate bits 14, 15	
22	L	22. Dec. 2015	Revised release for TANGO Firmware 1.66	Based on TANGO firmware revision 1.66
23	M	13. Jan. 2016	Improved trigger, "trigs" and 1:1 trigger output mode description	
24		20. Jan. 2016	Improved descriptions of LED100	
25		27. Jan. 2016	Improved and corrected encoder descriptions (encerr = e) etc.	
26	N	16. Feb 2016	Added "trigp", "trigc", "trigi", "iscur", "snsm 11", Improved trigger description, Improved and extended snapshot mode "snsm" description	Final documentation of firmware 1.67
27	O	11. July 2016	Improved descriptions, added "?hdi -1", corrected response description for cal/rm and move instructions.	Based on TANGO firmware revision 1.68
28		20. July 2016	Corrected "refdir" description, Improved descriptions	
29		31. Aug. 2016	Added "edigrly", Added "anamode" setup example Corrected "?anain" motor voltage calculation	
30		06. Sept. 2016	Added "hdimode" bit 9: Swap Joystick Y and Z axes	Based on TANGO firmware revision 1.68
31		18.Oct. 2016	Added "configdisplay", improved "noled" and "accelfunc" descriptions	Based on TANGO firmware revision 1.67
32		20. Dec. 2016	Added adigintyp, adiginfunc Added TANGO 3 mini sections to anain, anaout, anamode, adigin, adigout, brake, stoppol and drop Added new Error Numbers Improved descriptions	Based on TANGO 3 mini firmware 1.68
33		01. March 2017	Added "ctrd" parameter -1, Improved ctrd+ctrc description	Based on TANGO firmware 1.68
34		29. March 2017	Improved "scanmode" and "modulomode" description	
35	P	15. May 2017	Removed formatting error (20 was shown instead of the parameters throughout the document) Improved trigger description, added TANGO 3 mini	Based on TANGO firmware 1.68
36	Q	12. Jan. 2018	Improved I/O description and influence of the brake function Improved "anaout", "keymode" and "keyspeed" descriptions	



No.	Revision	Date	Changes	Remarks
37		09. March 2018	Added hdimode bit 10 description Added ctrdiff 1 option Changed Firmware from 1.68 to 1.69 Extended anamode description	Based on TANGO firmware 1.69
38	R	26. March 2018	Added decription of cal, rm behavior and nosetlimit to the llim instruction	
39		25. May 2018	Moved autostatus description from chapter "Controller States and Error Messages" to "Operating Modes"	
40		01. Aug. 2018	Added "calst" instruction Improved introduction, improved descriptions of closed loop and several instructions	Based on TANGO firmware 1.70
41	S	12. Sept. 2018	Added "brakepos" instruction	Based on TANGO firmware 1.70
42	T	22. Nov. 2018	-	Release for TANGO firmware 1.70
43		09. May 2019	Updated "ctrdiff" documentation Added "?enc 1" option	Based on TANGO firmware 1.71
44		22. May 2019	Added "encsync" instruction	
45	U	04. June 2019	Improved "lim" and "sevel" description	Release for TANGO firmware 1.71
46		25. June 2019	Improved "usteps" description	
47		18. July 2019	Improved description of the second trigger output and the snsj instruction	Based on TANGO firmware 1.72
48		28. Aug. 2019	Extended and updated "encpos" description Added "configencpos" description Added "calmode" 3, 4, 5 description Extended and updated "uptime" description	Based on TANGO firmware 1.72
49	V	06. Nov. 2019	Added "!pa 3"	Final documentation of firmware 1.72
		09. March 2020	"adigintyp" and "adiginfunc" changed from AUX mini to TANGO 3 mini	
50		13. March 2020	Improved description of "go"	
51		30. Oct. 2020	Added "encref" parameter 3	Based on TANGO firmware 1.73
52		10. June 2021	Improved "statusaxis" description, added "sta" description	Based on TANGO firmware 1.73
53		01. July 2021	Added "autopreset" and "snspreset"	Based on TANGO firmware 1.73
54		12. July 2021	Extended descriptions	Based on TANGO firmware 1.73
55		16. July 2021	Added Desktop HE instructions "cmderr", "cmdlist", "moe", "limmode", "vusb", "configvusb", "configcanres", "adigoutpreset", "vbrake", "brakedelay", "stopl" Added Desktop HE descriptions to "vbus", "configvbus", "adigin", "adigintyp", "adiginfunc", "brake", "enctype" Extended and improved descriptions	Based on TANGO Desktop HE
56		16. July 2021	Added "calposmot" description, "anamode 5" description	Based on TANGO firmware 1.73
57		19. July 2021	Added absolute encoder description to "encamp" Added "encres", "encform", "abspos", "calabspos", "posshift"	Based on TANGO Desktop HE
58	W	21. July 2021	Added "trigsns" instruction Improved "zwpos" description Corrected errors in the entire document	Final documentation of firmware 1.73 and TANGO Desktop HE

