# CONEX-AGP

## *Agilis-P Controller with Encoder Feedback*

Newport®
Experience | Solutions

# Controller Documentation

Firmware V1.1.x

*For Motion, Think Newport™*

# Table of Contents

**1.0 System Overview........................................................................................ 1**

1.1 General Description..........................................................................................1

1.1.1 Encoder Theory of Operation...........................................................................1

1.1.2 Closed Loop Technology – Deadband Feature...................................................2

1.2 CONEX-AGP ......................................................................................................3

1.2.1 Delivered Items ...........................................................................................3

1.2.2 Specifications ...............................................................................................3

1.2.3 Dimensions...................................................................................................4

1.3 System Environmental Specifications ...............................................................5

1.4 Connector Identification ....................................................................................5

1.5 USB Communication Settings............................................................................5

**2.0 Programming ........................................................................................... 6**

2.1 State Diagram ...................................................................................................6

2.2 Command Syntax...............................................................................................8

2.3 Command Execution Time ..................................................................................8

2.4 Command Set ....................................................................................................8

DB — Set/Get corrector deadband.......................................................................10

HT — Set/Get HOME search type.........................................................................11

ID — Set/Get stage identifier................................................................................12

IF — Set/Get interpolation factor.........................................................................13

KI — Set/Get integral gain....................................................................................14

KP — Set/Get proportional gain ...........................................................................15

LF — Set/Get low pass filter frequency.................................................................16

MM — Enter/Leave DISABLE state .......................................................................17

OR — Execute HOME search.................................................................................19

PA — Move absolute .............................................................................................20

PR — Move relative...............................................................................................21

PW — Enter/Leave CONFIGURATION state..........................................................22

RS — Reset controller............................................................................................23

RS## — Reset controller’s address........................................................................24

SA — Set/Get controller’s RS-485 address ............................................................25

SL — Set/Get negative software limit....................................................................26

SR — Set/Get positive software limit ....................................................................27

ST — Stop motion..................................................................................................28

SU — Set/Get encoder increment value.................................................................29

TB — Get command error string............................................................................30

TE — Get last command error ...............................................................................31

TH — Get target position....................................................................................32
TP — Get current position ..................................................................................33
TS — Get positioner error and controller state ...................................................34
VE — Get controller revision information..........................................................36
ZT — Get all configuration parameters .............................................................37

**3.0 Connector interfaces........................................................................ 38**
3.1 USB (Male mini-USB) ..........................................................................................38

**Service Form ........................................................................................... 39**

# CONEX-AGP
# Agilis- P Controller with Encoder Feedback

## 1.0 System Overview

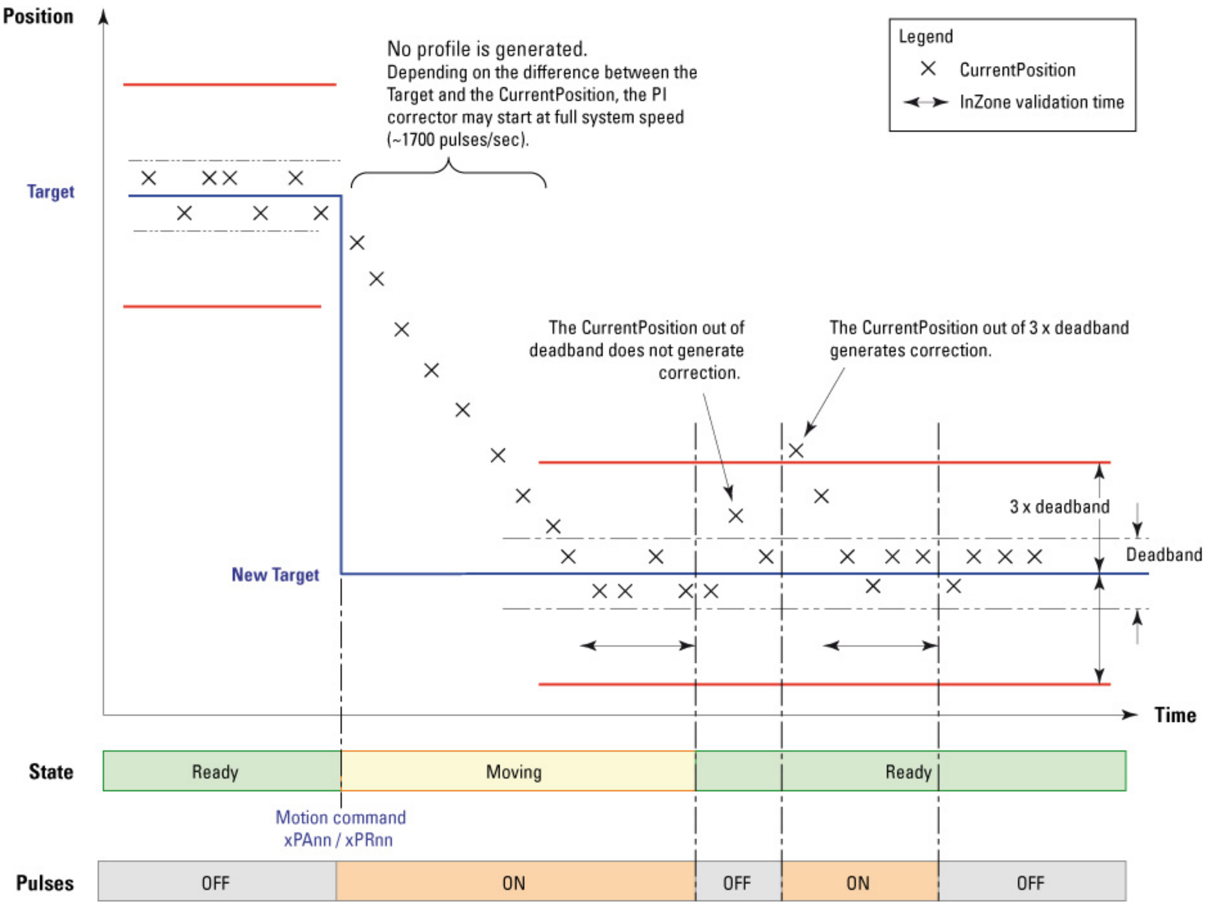
### 1.1 General Description

The CONEX-AGP is a single axis motion controller/driver for piezo actuator with encoder feedback. It provides a very compact and low-cost solution for driving a variety of Newport Agilis-type piezo stages from a PC.

Communication with the CONEX-AGP is achieved via an USB port (requires Windows™ operating system). A Windows™ based software enables basic motion. Advanced application programming is simplified by an ASCII command interface and a set of two letter mnemonic commands.

### 1.1.1 Encoder Theory of Operation

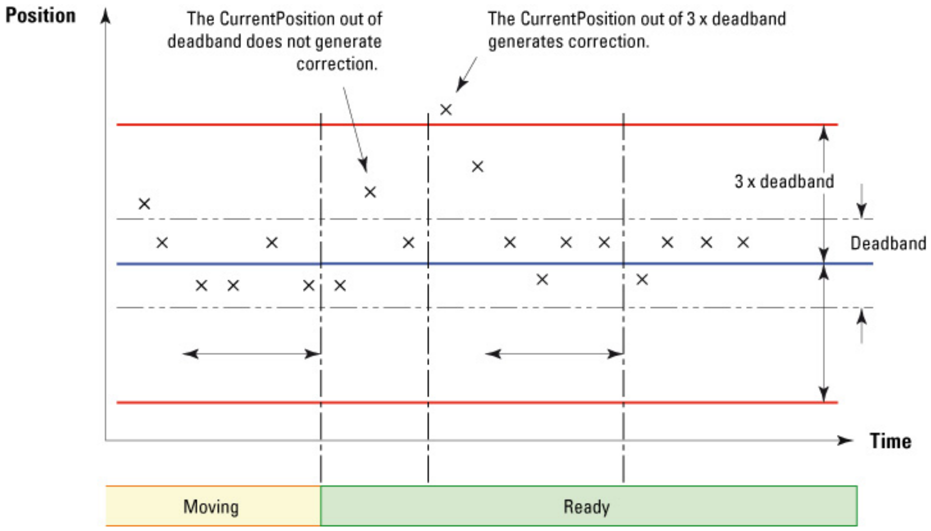
The encoder on the Agilis stage is generated from a proprietary marking technology that creates a pattern of fine lines directly on the stage. From these patterns, sinusoidal signals are generated which are then interpolated by the encoder electronics down to the specified resolution. The accuracy of the stage is very dependent on the quality of the pattern, which also leads to high positioning repeatability.


Position
Target
New Target
No profile is generated.
Depending on the difference between the Target and the CurrentPosition, the PI corrector may start at full system speed (~1700 pulses/sec).
Legend
× CurrentPosition
↔ InZone validation time
The CurrentPosition out of deadband does not generate correction.
The CurrentPosition out of 3 x deadband generates correction.
3 x deadband
Deadband
Time
State
Ready
Moving
Ready
Motion command xPAnn / xPRnn
Pulses
OFF
ON
OFF
ON
OFF

### 1.1.2 Closed Loop Technology – Deadband Feature

The Conex closed loop algorithm for Agilis stages is a simplified version of the typical DC servo loop algorithm. Although a motion profile is not generated in the algorithm, the Conex controller still closes a loop based on the error. The larger the error, the faster the stage is commanded to move to reduce the error. The stage could start at the full system speed of 1700 pulses if the stage is far from the desired position.

To close in on a position, the motion of the stage stops when the stage is within a specified range about the desired position. This range is called the deadband and is set in firmware. The deadband is the allowable deviation from the desired position and if the stage is inside the deadband within 20ms, the stage is considered in position and stops.


Position
The CurrentPosition out of deadband does not generate correction.
The CurrentPosition out of 3 x deadband generates correction.
3 x deadband
Deadband
Time
Moving
Ready

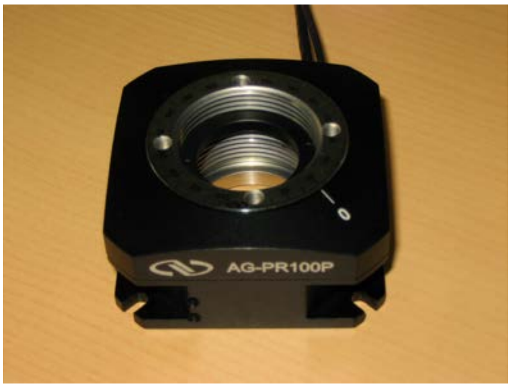
**NOTE**

**The servo loop can be disabled to eliminate any undesired correction of the position when the stage moves outside the deadband due to external sources.**

## 1.2 CONEX-AGP

### 1.2.1 Delivered Items

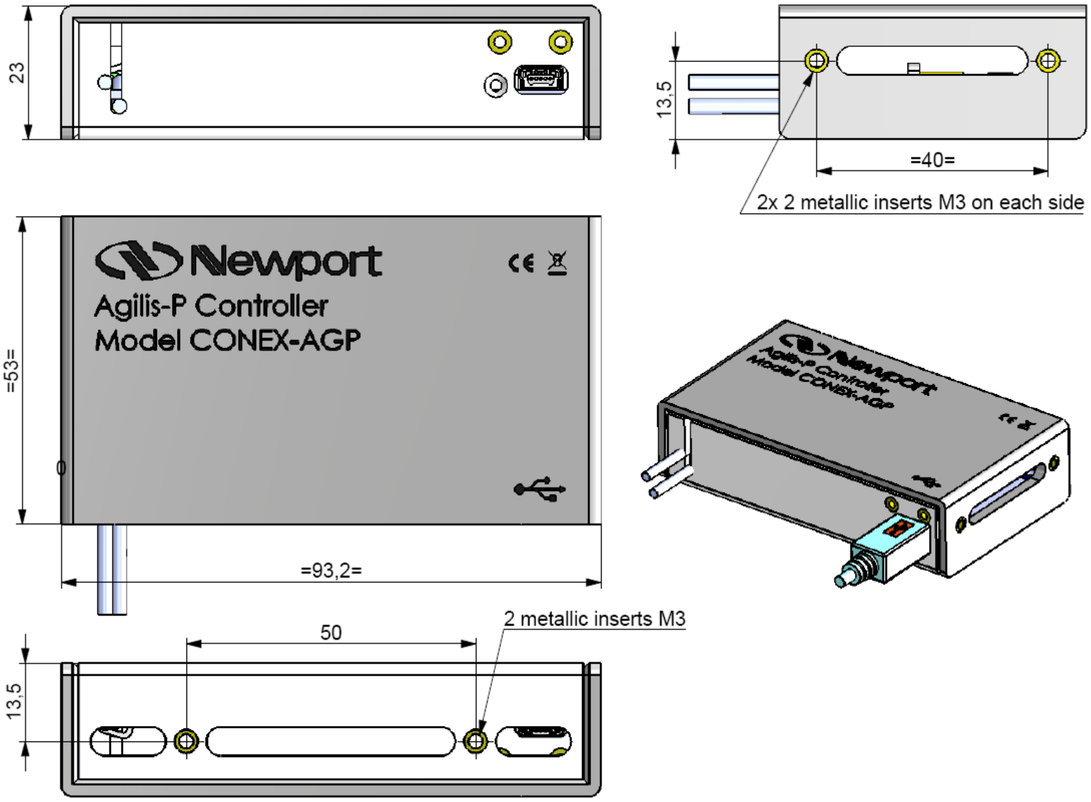
| | |
|---|---|
| • CONEX-xxx | Controller box with stage (cable length: 1.8m) |
| • CONEX-USB | USB cable, 1.8 m length |
| • CONEX-MOTION | CD-Rom |



### 1.2.2 Specifications

| | |
|---|---|
| General Description | Agilis controller with encoder feedback. |
| Control Capability | Piezo motors, closed loop |
| Piezo Output Voltage | 35 Vpeak |
| Control loop | – Digital PI loop<br>– 100 Hz servo rate |
| Motion | Absolute and relative motion. |
| Computer interface | – USB (requires Windows™ operating system) |
| Programming | – 25+ intuitive, 2-letter ASCII commands<br>– Command set includes software limits, user units… |
| Dedicated inputs | – Analog Cosine/Sine signals from encoder. |
| Status display | Two color LED |
| Communication rate | 50 Hz Max. (USB) |
| Internal safety feature | Watchdog timer |
| Consumption | +5V (USB): < 0.5A |

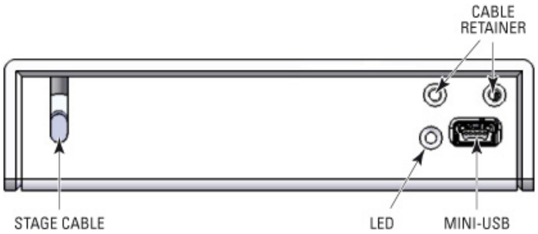
### 1.2.3 Dimensions



## 1.3 System Environmental Specifications

| | |
|---|---|
| Operating temperature | 5 °C to 40 °C |
| Operating humidity | 20% to 85% relative humidity, non-condensing |
| Location | Indoor use only |

## 1.4 Connector Identification



| | |
|---|---|
| USB | mini USB connector |
| LED | Status LED |
| STAGE | Stage entry cable |
| Cable retainer | 2 x M3 threaded hole to attach cable retainer |

## 1.5 USB Communication Settings

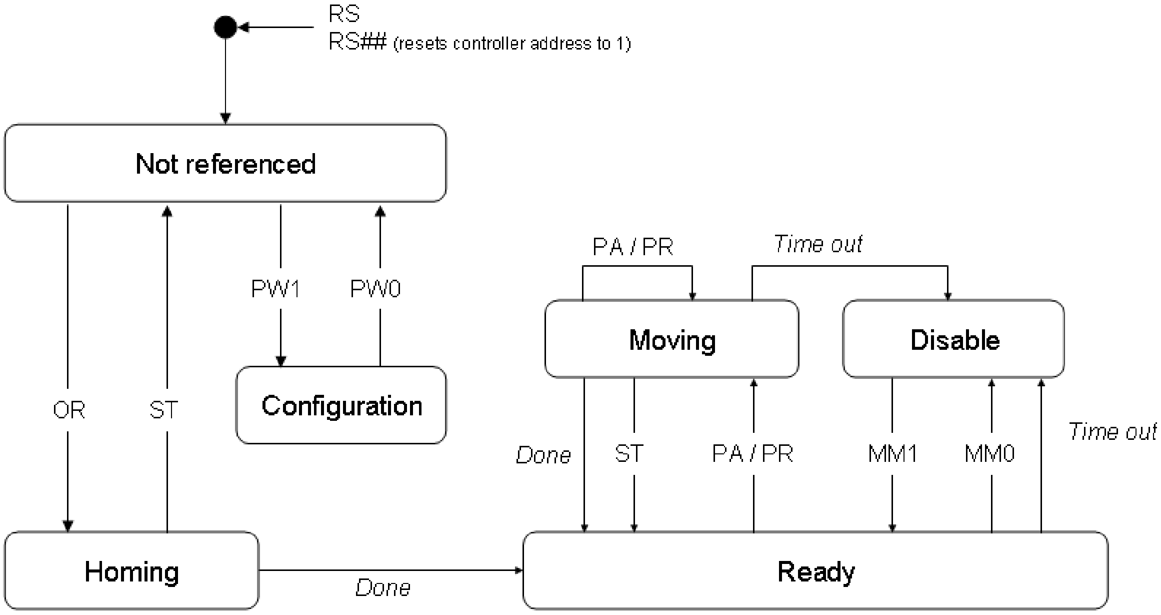
Communication parameters are preset in the CONEX-AGP controller and do not require any configuration:

| | |
|---|---|
| Bits per second | 921,600 |
| Data bits | 8 |
| Parity | None |
| Stop bits | 1 |
| Flow control | Xon/Xoff |
| Terminator | $C_R L_F$ |

# 2.0 Programming

## 2.1 State Diagram

For a safe and consistent operation, the CONEX-AGP uses 6 different operational states: Not referenced, Configuration, Homing, Ready, Disable and Moving. In each state, only specific commands are accepted by the CONEX-AGP. Therefore, it is important to understand the state diagram below and which commands and actions cause transitions between the different states. See section 2.4 for additional command/state information:



**LED display:**

| | |
|---|---|
| NOT REFERENCED: | If everything is OK then SOLID ORANGE. |
| NOT REFERENCED: | If no parameters then SOLID RED. |
| CONFIGURATION: | SLOW BLINKING RED. |
| READY: | SOLID GREEN. |
| DISABLE: | SLOW BLINKING GREEN. |
| HOMING: | FAST BLINKING GREEN. |
| MOVING: | FAST BLINKING GREEN. |

When powering the CONEX-AGP, the controller starts initialization. When the initialization is successful, the controller goes to the NOT REFERENCED state. From the NOT REFERENCED state, the controller can go to the CONFIGURATION state using the PW1 command. In the CONFIGURATION state, the CONEX-AGP allows changes to all configuration parameters, like corrector coefficients or travel limits. The PW0 command saves all changes to the controller’s memory and returns the controller back to the NOT REFERENCED state.

To execute any move commands (PA, PR), the controller must be in the READY or MOVING states. To get from the NOT REFERENCED state to the READY state, the positioner must be homed first with the OR command. During homing (OR command execution), the controller is in HOMING state. When homing is successful, the controller automatically goes to the READY state. The process for homing and the signals utilized during homing can be defined with the HT command.

In the READY state, the control loop is closed. During a move execution (PA/PR), the controller is in the MOVING state and goes automatically back to the READY state when the move is completed. A time out error during a move changes the controller to the DISABLE state.

In the DISABLE state, the control loop is open. But the encoder is still read and the current position gets updated. The DISABLE state can be used to make sure that the control loop will not generate any corrective motion command (due to noise or little drift) while at a given position.To go from the READY state to the DISABLE state and vice versa, use the MM command.

To get from the READY or DISABLE states back to the NOT REFERENCED state, to make some further parameter changes in CONFIGURATION state, for example, you need to reboot the controller with the RS command.

## 2.2 Command Syntax

The CONEX-AGP is a command driven controller. The general format of a command is a two letter ASCII character preceded and followed by parameters specific to the command:

**Command format:**

| | | |
|---|---|---|
| **nn** | **AA** | **xx** |

**nn** — Optional or required controller address.

**AA** — Command name.

**xx** — Optional or required value or "**?**" to query current value.

Both, upper and lower case characters are accepted. Depending on the command, it can have an optional or required prefix (**nn**) for the controller address and/or a suffix (**xx**) value or a "**?**".

**Blank spaces**

Blanks are allowed and ignored in any position, including inside a numerical value. The following two commands are equivalent, but the first example might be confusing and uses more memory:

2P A1.43 6

2PA1.436

**Decimal separator**

A dot (".") is used as decimal separator for all numerical values.

**Command terminator**

Commands are executed as the command terminator $C_RL_F$ (carriage-return line-feed, ASCII 13 and ASCII 10) is received. The controller will analyze the received string. If the command is valid and its parameters are in the specified range, it will be executed. Otherwise it will memorize an error.

After the execution of the command, all remaining characters in the input string, if any, will be ignored. In particular, it is not possible to concatenate several commands on a single string from the PC to the CONEX-AGP

Each command will handle the memorization of related errors that can be accessed with the TE command properly. Please refer to the command set in section 2.4 for details.

## 2.3 Command Execution Time

The CONEX-AGP controller interprets commands continuously as received. The typical execution time for a "tell position command" (nTP?) is about 10 ms. Here, command execution time means the time from sending the command until receipt of the answer.

It is important to note that a move command, that may last for several seconds, will not suspend the controller from further command execution. For an efficient process flow with many move commands, it is recommended to query the controller status (TS command) or the current position (TP command) before any further motion command is sent.

## 2.4 Command Set

This section describes the supported two-letter ASCII commands used to configure and operate the CONEX-AGP. The general command format is:

**Command format:**

| | | |
|---|---|---|
| **nn** | **AA** | **xx** |

**nn** — Optional or required controller address.

**AA** — Command name.

**xx** — Optional or required value or "**?**" to query current value.

Most commands can be used to set a value (in that case the command name is followed by the value "**xx**") or to query the current value (in that case the command name is followed by a "**?**"). When querying a value, the controller responds with the command it received followed by the queried value. For example, a 1LF20 sets the low pass filter frequency of the controller #1 to 20Hz. A 1LF? sends the response 1LF20.

Not every command can be executed in all states of the CONEX-AGP and some commands have different meanings in different states. It is therefore important to understand the state diagram of the controller, see section 2.1.

| | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** | **Description** |
|---|---|---|---|---|---|---|
| **DB** | □ | ○ | □ | – | – | Set/Get corrector deadband |
| **HT** | □ | ○ | – | – | – | Set/Get HOME search type |
| **ID** | □ | ○ | □ | – | – | Set/Get stage identifier |
| **IF** | □ | ○ | □ | – | – | Set/Get interpolation factor |
| **KI** | □ | ○ | □ | – | – | Set/Get integral gain |
| **KP** | □ | ○ | □ | – | – | Set/Get proportional gain |
| **LF** | □ | ○ | □ | – | – | Set/Get low pass filter frequency |
| **MM** | – | – | ● | ● | – | Enter/Leave DISABLE state |
| **OR** | ● | – | – | – | – | Execute HOME search |
| **PA** | – | – | – | ● | ● | Move absolute |
| **PR** | – | – | – | ● | ● | Move relative |
| **PW** | ● | ● | – | – | – | Enter/Leave CONFIGURATION state |
| **RS** | ● | ● | ● | ● | ● | Reset controller |
| **RS##** | ● | ● | ● | ● | ● | Reset controller's address to 1 |
| **SA** | – | ○ | – | – | – | Set/Get controller's RS-485 address |
| **SL** | – | ○ | □ | □ | – | Set/Get negative software limit |
| **SR** | – | ○ | □ | □ | – | Set/Get positive software limit |
| **ST** | – | – | – | – | ● | Stop motion |
| **SU** | – | ○ | – | – | – | Set/Get encoder increment value |
| **TB** | ● | ● | ● | ● | ● | Get command error string |
| **TE** | ● | ● | ● | ● | ● | Get last command error |
| **TH** | ● | ● | ● | ● | ● | Get target position |
| **TP** | ● | ● | ● | ● | ● | Get current position |
| **TS** | ● | ● | ● | ● | ● | Get positioner error and controller state |
| **VE** | ● | ● | ● | ● | ● | Get controller revision information |
| **ZT** | ● | ● | ● | – | – | Get all controller parameters |

| | |
|---|---|
| **Motion:** | Corresponds to HOMING and MOVING state (for details see state diagram, section 2.1). |
| ○ | Changes configuration parameters. Those changes will be stored in the controller's memory with the PW1 command and remain available after switching off the controller. |
| □ | Changes working parameters only. Those changes will get lost when switching off the controller. |
| ● | Accepted command. |
| **–** | Write command not accepted (will return an error). |
| **Command:** | Command passed without preceding controller number applies to all controllers (e.g. MM0 disables all controllers). |

# DB — Set/Get corrector deadband

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | ☐ | ○ | ☐ | – | – |

| | | | |
|---|---|---|---|
| **Syntax** | **xxDBnn** or **xxDB?** | | |
| **Parameters** | | | |
| **Description** | **xx** [int] | — | Controller address. |
| | **nn** [float] | — | Deadband value. |
| **Range** | **xx** | — | **1** to **31** |
| | **nn** | — | ≥ **0** and < **0.05** |
| **Units** | **xx** | — | None |
| | **nn** | — | Preset units |
| **Defaults** | **xx** Missing: | | Error B. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |
| | **nn** Missing: | | Error C. |
| | Out of range: | | Error C. |
| **Description** | The deadband parameter defines an area, around a set position, in which the controller will not try to make any more corrections. This is useful to prevent the corrector from generating unwanted motion, for example because of noise on the encoder. | | |
| **Returns** | If the sign “**?**” takes place of **nn**, this command returns the current programmed value. | | |
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | C | — | Parameter missing or out of range. |
| | D | — | Execution not allowed. |
| | K | — | Execution not allowed in READY state. |
| | L | — | Execution not allowed in HOMING state. |
| | M | — | Execution not allowed in MOVING state. |
| **Rel. Commands** | **KI** | — | Set the integral gain. |
| | **KP** | — | Set the proportional gain. |
| **Example** | 1DB0.000075 | \| | *Set controller #1 deadband to 75nm (in the case of a translation stage).* |

# HT — Set/Get HOME search type

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | ☐ | ○ | – | – | – |

**Syntax** **xxHTnn** or **xxHT?**

**Parameters**

**Description**

**xx** [int] — Controller address.

**nn** [int] — Home type value.

**Range**

**xx** — **1** to **31**

**nn** — **1** use current position as HOME.

**4** use negative end of run as HOME

**5** end of run offset initialization

**Units**

**xx** — None.

**nn** — None.

**Defaults**

**xx** Missing: Error B.

Out of range: Error B.

Floating point: Error A.

**nn** Missing: Error C.

Out of range: Error C.

**Description**

This command sets the type of HOME search used with the OR command.

To improve repeatability of the Home position, the stage does not use the mechanical end of run, but a position a little away from it defined by a precise position of the encoder signal. This position is stored in the controller memory the first time the stage is used in mode 4 (default), without action required from the user.

Mode 5 is a maintenance mode and must be used by Newport service personnel only.

* The maximum number of times that  parameter can be stored to the memory is limited to 100 writes. Unit failure due to excessive use of the command is not covered by warranty.

**Returns**

If the sign “**?**” takes place of **nn**, this command returns the current programmed value.

**Errors**

A — Unknown message code or floating point controller address.

B — Controller address not correct.

C — Parameter missing or out of range.

D — Execution not allowed.

J — Execution not allowed in DISABLE state.

K — Execution not allowed in READY state.

L — Execution not allowed in HOMING state.

M — Execution not allowed in MOVING state.

**Rel. Commands**

**OR** — Execute HOME search.

**Example**

1HT1 | *Set controller #1 HOME sequence to use current position as home.*

# ID — Set/Get stage identifier

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | ☐ | ○ | ☐ | – | – |

| | | | |
|---|---|---|---|
| **Syntax** | **xxIDnn** or **xxID?** | | |
| **Parameters** | | | |
| **Description** | **xx** [int] | — | Controller address. |
| | **nn** [char] | — | Stage model number. |
| **Range** | **xx** | — | **1** to **31** |
| | **nn** | — | **1** to **31** ASCII characters. |
| **Units** | **xx** | — | None |
| | **nn** | — | None |
| **Defaults** | **xx** Missing: | | Error B. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |
| | **nn** Missing: | | Error C. |
| | Out of range: | | Error C. |
| Description | The ID? command return the product name. In CONFIGURATION mode, this command allows changing the controller identifier. | | |
| **Returns** | If the sign "**?**" takes place of **nn**, this command returns the current programmed value. | | |
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | C | — | Parameter missing or out of range. |
| | D | — | Execution not allowed. |
| | H | — | Execution not allowed in NOT REFERENCED state. |
| | J | — | Execution not allowed in DISABLE state. |
| | K | — | Execution not allowed in READY state. |
| | L | — | Execution not allowed in HOMING state. |
| | M | — | Execution not allowed in MOVING state. |
| **Rel. Commands** | **ZT** | — | Get configuration parameters. |
| **Example** | 1ID? | \| | *Get stage identifier for controller #1.* |
| | *1ID CONEX-AGP* | \| | *Controller returns product namer: CONEX-AGP.* |

# IF — Set/Get interpolation factor

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | ☐ | ○ | ☐ | – | – |

| | | | |
|---|---|---|---|
| **Syntax** | **xxIFnn** or **xxIF?** | | |
| **Parameters** | | | |
| **Description** | **xx** [int] | — | Controller address. |
| | **nn** [float] | — | Hysteresis value. |
| **Range** | **xx** | — | **1** to **31** |
| | **nn** | — | > **0** and <= **2000** |
| **Units** | **xx** | — | None |
| | **nn** | — | None |
| **Defaults** | **xx** Missing: | | Error B. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |
| | **nn** Missing: | | Error C. |
| | Out of range: | | Error C. |
| **Description** | The IF command sets the interpolation factor. The interpolation defines the number by which is devided the encoder resolution to define the interpolated resolution. | | |
| **Returns** | If the sign “**?**” takes place of **nn**, this command returns the current programmed value. | | |
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | C | — | Parameter missing or out of range. |
| | D | — | Execution not allowed. |
| | K | — | Execution not allowed in READY state. |
| | L | — | Execution not allowed in HOMING state. |
| | M | — | Execution not allowed in MOVING state. |
| **Rel. Commands** | **SU** | — | Set/Get encoder resolution |
| **Example** | 1IF1000 | \| | *Set controller #1 interpolation factor to 1000.* |

# KI — Set/Get integral gain

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | ☐ | ○ | ☐ | – | – |

| | | | |
|---|---|---|---|
| **Syntax** | **xxKInn** or **xxKI?** | | |
| **Parameters** | | | |
| **Description** | **xx** [int] | — | Controller address. |
| | **nn** [float] | — | Integral gain value. |
| **Range** | **xx** | — | **1** to **31** |
| | **nn** | — | ≥ **0** and <= **3000** |
| **Units** | **xx** | — | None. |
| | **nn** | — | Volt * preset unit/second. |
| **Defaults** | **xx** Missing: | | Error B. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |
| | **nn** Missing: | | Error C. |
| | Out of range: | | Error C. |

**Description**

In CONFIGURATION state, this command sets the integral gain of the PI control loop which can than be saved in the controller's nonvolatile memory using the PW command. It is also the default value that will be used unless a different value is set in DISABLE state.

In DISABLE state, this command allows setting a new working parameter for the integral gain. This value is not saved in the controller's memory and will be lost after reboot.

**Returns** If the sign "**?**" takes place of **nn**, this command returns the current programmed value.

| | | | |
|---|---|---|---|
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | C | — | Parameter missing or out of range. |
| | D | — | Execution not allowed. |
| | K | — | Execution not allowed in READY state. |
| | L | — | Execution not allowed in HOMING state. |
| | M | — | Execution not allowed in MOVING state. |
| **Rel. Commands** | **KP** | — | Set proportional gain. |
| **Example** | 1KI800 | \| | *Set controller #1 integral gain to 800.* |

# KP — Set/Get proportional gain

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | ☐ | ○ | ☐ | – | – |

**Syntax** **xxKPnn** or **xxKP?**

**Parameters**

**Description** **xx** [int] — Controller address.

**nn** [float] — Proportional gain value.

**Range** **xx** — **1** to **31**

**nn** — ≥ **0** and < **3000**

**Units** xx — None.

**nn** — Volt/preset unit

**Defaults** **xx** Missing: Error B.

Out of range: Error B.

Floating point: Error A.

**nn** Missing: Error C.

Out of range: Error C.

**Description** In CONFIGURATION state, this command sets the proportional gain of the PI control loop which can than be saved in the controller's nonvolatile memory using the PW command. It is also the default value that will be used unless a different value is set in DISABLE state.

In DISABLE state, this command allows setting a new working parameter for the proportional gain. This value is not saved in the controller's memory and will be lost after reboot.

**Returns** If the sign "**?**" takes place of **nn**, this command returns the current programmed value.

**Errors** A — Unknown message code or floating point controller address.

B — Controller address not correct.

C — Parameter missing or out of range.

D — Execution not allowed.

K — Execution not allowed in READY state.

L — Execution not allowed in HOMING state.

M — Execution not allowed in MOVING state.

**Rel. Commands** **KI** — Set integral gain.

**Example** 1KP10 | *Set controller #1 proportional gain to 10.*

# LF — Set/Get low pass filter frequency

| Usage | Not Ref. | Config. | Disable | Ready | Motion |
|---|---|---|---|---|---|
| | ☐ | ○ | ☐ | – | – |

**Syntax** **xxLFnn** or **xxLF?**

**Parameters**

| | | | |
|---|---|---|---|
| **Description** | **xx** [int] | — | Controller address. |
| | **nn** [float] | — | Friction compensation value. |
| **Range** | **xx** | — | **1** to **31** |
| | **nn** | — | > **0** and **<= 1000** |
| **Units** | **xx** | — | None. |
| | **nn** | — | Hertz. |
| **Defaults** | **xx** Missing: | | Error B. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |
| | **nn** Missing: | | Error C. |
| | Out of range: | | Error C. |

**Description** This commands sets the low pass filter frequency used on both sine and cosine inputs coming from the optical encoder.

**Returns** If the sign “**?**” takes place of **nn**, this command returns the current programmed value.

| | | | |
|---|---|---|---|
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | C | — | Parameter missing or out of range. |
| | D | — | Execution not allowed. |
| | K | — | Execution not allowed in READY state. |
| | L | — | Execution not allowed in HOMING state. |
| | M | — | Execution not allowed in MOVING state. |
| **Example** | 1LF10 | \| | *Set controller #1 low pass filter to 10Hz.* |

# MM — Enter/Leave DISABLE state

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | – | – | ● | ● | – |

**Syntax** **xxMMnn** or **xxMM?**

**Parameters**

| | | | |
|---|---|---|---|
| **Description** | **xx** [int] | — | Controller address. |
| | **nn** [float] | — | Velocity feed forward value. |
| **Range** | **xx** | — | **0** to **31** |
| | **nn** | — | **0** changes state from READY to DISABLE. |
| | | | **1** changes state from DISABLE to READY. |
| **Units** | **xx** | — | None. |
| | **nn** | — | None. |
| **Defaults** | **xx** Missing: | | Change to 0. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |
| | **nn** Missing: | | Error C. |
| | Out of range: | | Error C. |

**Description** When the MM command is sent without preceding controller number or the controller number is 0, the MM command gets executed on all controllers.

MM0 changes the controller’s state from READY to DISABLE. In DISABLE state the control loop is open. The encoder, though, is still read and the current position gets updated.

MM1 changes the controller’s state from DISABLE to READY. The controller’s set point position is set equal to its current position and the control loop gets closed.

**Returns** If the sign “**?**” takes place of **nn**, this command returns the current controller state (ef).

**Controller states (ef):**

- **0A**: NOT REFERENCED from reset.
- **0B**: NOT REFERENCED from HOMING.
- **0C**: NOT REFERENCED from CONFIGURATION.
- **0D**: NOT REFERENCED from DISABLE.
- **0E**: NOT REFERENCED from READY.
- **0F**: NOT REFERENCED from MOVING.
- **10**: NOT REFERENCED no parameters.
- **14**: CONFIGURATION.
- **1E**: HOMING.
- **28**: MOVING.
- **32**: READY from HOMING.
- **33**: READY from MOVING.
- **34**: READY from DISABLE.
- **3C**: DISABLE from READY.
- **3D**: DISABLE from MOVING.

| | | | |
|---|---|---|---|
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |

C — Parameter missing or out of range.

D — Execution not allowed.

H — Execution not allowed in NOT REFERENCED state.

I — Execution not allowed in CONFIGURATION state.

L — Execution not allowed in HOMING state.

M — Execution not allowed in MOVING state.

**Rel. Commands** **PW** — Enter/leave CONFIGURATION state.

**Example**

```
MM0   |  All controllers go to DISABLE state
MM?
        MM3C
```

# OR — Execute HOME search

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | ● | – | – | – | – |

**Syntax** **xxOR**

**Parameters**

| | | | |
|---|---|---|---|
| **Description** | **xx** [int] | — | Controller address. |
| **Range** | **xx** | — | **1** to **31** |
| **Units** | **xx** | — | None. |
| **Defaults** | **xx** Missing: | | Error B. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |
| | **nn** Missing: | | Error C. |
| | Out of range: | | Error C. |

**Description** This command starts the execution of the HOME search as defined by the HT command.

When in NOT REFERENCED state, for instance after system start, any positioner must first get homed with the OR command before further motion commands can get executed.

The OR command gets accepted only in NOT REFERENCED state.

| | | | |
|---|---|---|---|
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | C | — | Parameter missing or out of range. |
| | D | — | Execution not allowed. |
| | I | — | Execution not allowed in CONFIGURATION state. |
| | J | — | Execution not allowed in DISABLE state. |
| | K | — | Execution not allowed in READY state. |
| | L | — | Execution not allowed in HOMING state. |
| | M | — | Execution not allowed in MOVING state. |
| **Rel. Commands** | **HT** | — | Set HOME search type. |

**Example** 1OR | *Execute HOME search with controller #1.*

# PA — Move absolute

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
|  | – | – | – | ● | ● |

**Syntax** **xxPAnn** or **xxPA?**

**Parameters**

**Description**
**xx** [int] — Controller address.
**nn** [float] — New target position.

**Range**
**xx** — **1** to **31**
**nn** — **> SL** and **< SR**

**Units**
**xx** — None.
**nn** — Preset units.

**Defaults**
**xx** Missing: Error B.
Out of range: Error B.
Floating point: Error A.
**nn** Missing: Error C.
Out of range: Error C.

**Description**

The PA command initiates an absolute move. When received, the positioner will move to the new target position specified by **nn**.

The PA command gets only accepted in READY or MOVING state, AND when the new target position is higher or equal to the negative software limit (SL), AND lower or equal to the positive software limit (SR).

**Returns** If the sign "**?**" takes place of **nn**, this command returns the target position value.

**Errors**
A — Unknown message code or floating point controller address.
B — Controller address not correct.
C — Parameter missing or out of range.
D — Execution not allowed.
G — Target position out of limits.
H — Execution not allowed in NOT REFERENCED state.
I — Execution not allowed in CONFIGURATION state.
J — Execution not allowed in DISABLE state.

**Rel. Commands**
**PR** — Move relative.
**TH** — Get target position.
**TP** — Get current position.
**SU** — Set encoder increment value.

**Example** 1PA2.2 | *Move positioner on controller #1 to absolute position 2.2 units.*

# PR — Move relative

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | – | – | – | ● | ● |

**Syntax** **xxPRnn**

**Parameters**

**Description**
**xx** [int] — Controller address.
**nn** [float] — Displacement.

**Range**
**xx** — **1** to **31**
**nn** — **> SL** and **< SR**

**Units**
**xx** — None.
**nn** — Preset units.

**Defaults**
**xx** Missing: Error B.
Out of range: Error B.
Floating point: Error A.
**nn** Missing: Error C.
Out of range: Error C.

**Description** The PR command initiates a relative move. When received, the positioner will move to a new target position **nn** units away from the current target position.

The PR command gets only accepted in READY or MOVING state, AND when the distance of the positioner to the end of runs is larger than the commanded displacement.

**Returns** If the sign “**?**” takes place of **nn**, this command returns the target position value.

**Errors**
A — Unknown message code or floating point controller address.
B — Controller address not correct.
C — Parameter missing or out of range.
D — Execution not allowed.
G — Displacement out of limits.
H — Execution not allowed in NOT REFERENCED state.
I — Execution not allowed in CONFIGURATION state.
J — Execution not allowed in DISABLE state.

**Rel. Commands**
**PA** — Move absolute.
**TH** — Get target position.
**TP** — Get current position.
**SU** — Set encoder increment value.

**Example** 1PR2.2 | *Move positioner on controller #1 to a new position 2.2 units away from the current target position.*

# PW — Enter/Leave CONFIGURATION state

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | ● | ● | – | – | – |

| | | | |
|---|---|---|---|
| **Syntax** | **xxPWnn** or **xxPW?** | | |
| **Parameters** | | | |
| **Description** | **xx** [int] | — | Controller address. |
| | **nn** [float] | — | Mode. |
| **Range** | **xx** | — | **1** to **31** |
| | **nn** | — | **1**: Go from NOT REFERENCED state to CONFIGURATION state. |
| | | | **0**: Go from CONFIGURATION state to NOT REFERENCED state. |
| **Units** | **xx** | — | None. |
| | **nn** | — | None. |
| **Defaults** | **xx** Missing: | | Error B. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |
| | **nn** Missing: | | Error C. |
| | Out of range: | | Error C. |

**Description**

PW1 changes the controller's state from NOT REFERENCED to CONFIGURATION. In Configuration state all parameter settings are saved in the controller's memory and remain available after switching off the controller. In addition, some settings are only possible in CONFIGURATION state (e.g. set encoder increment value, etc.).

PW0 checks all stage parameters, and if they are acceptable, saves them in the flash memory of the controller. After that, it changes the controller's state from CONFIGURATION to NOT REFERENCED.

The execution of a PW0 command may take up to 10 seconds. During that time the controller will not respond to any other command.

**Returns** If the sign "**?**" takes place of **nn**, this command returns the current state.

| | | | |
|---|---|---|---|
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | C | — | Parameter missing or out of range. |
| | D | — | Execution not allowed. |
| | J | — | Execution not allowed in DISABLE state. |
| | K | — | Execution not allowed in READY state. |
| | L | — | Execution not allowed in HOMING state. |
| | M | — | Execution not allowed in MOVING state. |
| **Rel. Commands** | **MM** | — | Enter/Leave DISABLE state. |
| **Example** | 1PW1 | \| | *Changes controller #1 to CONFIGURATION state.* |

---

**NOTE**

**The PW command is limited to 100 writes. Unit failure due to excessive use of the PW command is not covered by warranty.**

**The PW command is used to change the configuration parameters that are stored in memory, and not parameters that are needed to be changed on the fly.**

---

# RS — Reset controller

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | ● | ● | ● | ● | ● |

| | | | |
|---|---|---|---|
| **Syntax** | **xxRS** | | |
| **Parameters** | | | |
| **Description** | **xx** [int] | — | Controller address. |
| **Range** | **xx** | — | **1** to **31** |
| **Units** | xx | — | None. |
| **Defaults** | **xx** Missing: | | Error B. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |

**Description** The RS command issues a hardware reset of the controller, equivalent to a power-up.

To go from DISABLE or READY state to CONFIGURATION state, it is also needed to first reset the controller with the RS command, and then to change the controller’s state with the PW1 command from NOT REFERENCED to CONFIGURATION.

| | | | |
|---|---|---|---|
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | D | — | Execution not allowed. |
| **Example** | 1RS | \| | *Reset controller #1.* |

# RS## — Reset controller's address

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | ● | ● | ● | ● | ● |

**Syntax** **xxRS## or RS##**

**Parameters**

| | | | |
|---|---|---|---|
| **Description** | **xx** [int] | — | Axis number. |
| **Range** | **xx** | — | **0 to 31** |
| **Units** | **xx** | — | None. |
| **Defaults** | **xx** Missing: | | Change to 0. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |

**Description** The RS## command resets the controller's address to 1. This address needs to be different for each CONEX devices when connected on a RS-485 communication network.

**Returns**

| | | | |
|---|---|---|---|
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | D | — | Execution not allowed. |
| | H | — | Execution not allowed in NOT REFERENCED state. |
| | J | — | Execution not allowed in DISABLE state. |
| | K | — | Execution not allowed in READY state. |
| | L | — | Execution not allowed in HOMING state. |
| | M | — | Execution not allowed in MOVING state. |

**Example** RS## | Res*et controller's address to 1.*

# SA — Set/Get controller’s RS-485 address

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | – | ○ | – | – | – |

**Syntax** **xxSAnn** or **xxSA?**

**Parameters**

| | | | |
|---|---|---|---|
| **Description** | **xx** [int] | — | Axis number. |
| | **nn** [int] | — | Controller’s axis number. |
| **Range** | **xx** | — | **1** |
| | **nn** | — | **2** to **31** |
| **Units** | **xx** | — | None. |
| | **nn** | — | None. |
| **Defaults** | **xx** Missing: | | Error B. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |
| | **nn** Missing: | | Error C. |
| | Out of range: | | Error C. |

**Description** The SA command sets the controller’s RS-485 address. This address is ONLY used when the controller is configured for RS-485 communication.

Newport recommends using the supplied utility software for all controller configurations. The SA command is of practical use only when not using this software.

**Returns** If the sign “**?**” takes place of **nn**, this command returns the current programmed value.

| | | | |
|---|---|---|---|
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | C | — | Parameter missing or out of range. |
| | D | — | Execution not allowed. |
| | H | — | Execution not allowed in NOT REFERENCED state. |
| | J | — | Execution not allowed in DISABLE state. |
| | K | — | Execution not allowed in READY state. |
| | L | — | Execution not allowed in HOMING state. |
| | M | — | Execution not allowed in MOVING state. |

**Example** 1SA3 | *Set controller’s RS-485 address to 3.*

# SL — Set/Get negative software limit

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | – | ○ | ☐ | ☐ | – |

**Syntax** **xxSLnn** or **xxSL?**

**Parameters**

| | | | |
|---|---|---|---|
| **Description** | **xx** [int] | — | Controller address. |
| | **nn** [float] | — | Negative software limit. |
| **Range** | **xx** | — | **1** to **31** |
| | **nn** | — | > **$-10^{12}$** and **≤ 0** |
| **Units** | **xx** | — | None. |
| | **nn** | — | Preset units. |
| **Defaults** | **xx** Missing: | | Error B. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |
| | **nn** Missing: | | Error C. |
| | Out of range: | | Error C. |

**Description**

In CONFIGURATION state, this command sets the negative software limit which can than be saved in the controller’s nonvolatile memory using the PW command. It is also the default value that will be used unless a different value is set in DISABLE or READY state.

In DISABLE or READY state, this command allows setting a new working parameter for the negative software limit. It must be lower or equal to the target position. This value is not saved in the controller’s memory and will be lost after reboot.

The software limits are useful to limit the travel range of a positioner. There is no possibility to disable software limits. For an almost infinite motion, for instance with a rotation stage, set the lowest possible value, which is: -2147000000 * "encoder increment value" (see SU command). For instance if the encoder increment value is 0,0005, this limit is -1073500.

**Returns** If the sign “**?**” takes place of **nn**, this command returns the current programmed value.

| | | | |
|---|---|---|---|
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | C | — | Parameter missing or out of range. |
| | D | — | Execution not allowed. |
| | H | — | Execution not allowed in NOT REFERENCED state. |
| | L | — | Execution not allowed in HOMING state. |
| | M | — | Execution not allowed in MOVING state. |
| **Rel. Commands** | **SR** | — | Set positive software limit. |
| **Example** | 1SL-100 | \| | *Set controller #1 negative software limit to –100 units.* |

# SR — Set/Get positive software limit

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | – | ○ | ☐ | ☐ | – |

**Syntax** **xxSRnn** or **xxSR?**

**Parameters**

| | | | |
|---|---|---|---|
| **Description** | **xx** [int] | — | Controller address. |
| | **nn** [float] | — | Positive software limit. |
| **Range** | **xx** | — | **1** to **31** |
| | **nn** | — | ≥ **0** and < $\mathbf{10^{12}}$ |
| **Units** | **xx** | — | None. |
| | **nn** | — | Preset units. |
| **Defaults** | **xx** Missing: | | Error B. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |
| | **nn** Missing: | | Error C. |
| | Out of range: | | Error C. |

**Description** In CONFIGURATION state, this command sets the positive software limit which can than be saved in the controller’s nonvolatile memory using the PW command. It is also the default value that will be used unless a different value is set in DISABLE or READY state.

In DISABLE or READY state, this command allows setting a new working parameter for the positive software limit. It must be larger or equal to the target position. This value is not saved in the controller’s memory and will be lost after reboot.

The software limits are useful to limit the travel range of a positioner. There is no possibility to disable software limits. For an almost infinite motion, for instance with a rotation stage, set the largest possible value, which is: 2147000000 * "encoder increment value" (see SU command). For instance if the encoder increment value is 0,0005, this limit is 1073500.

**Returns** If the sign “**?**” takes place of **nn**, this command returns the current programmed value.

| | | | |
|---|---|---|---|
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | C | — | Parameter missing or out of range. |
| | D | — | Execution not allowed. |
| | H | — | Execution not allowed in NOT REFERENCED state. |
| | L | — | Execution not allowed in HOMING state. |
| | M | — | Execution not allowed in MOVING state. |
| **Rel. Commands** | **SL** | — | Set negative software limit. |
| **Example** | 1SR100 | \| | *Set controller #1 positive software positive to 100 units.* |

# ST — Stop motion

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | – | – | – | – | ● |

| | | | |
|---|---|---|---|
| **Syntax** | **[xx]ST** | | |
| **Parameters** | | | |
| **Description** | **xx** [int] | — | Controller address. |
| **Range** | **xx** | — | **0** to **31** |
| **Units** | **xx** | — | None. |
| **Defaults** | **xx** | Missing: | Change to 0. |
| | | Out of range: | Error B. |
| | | Floating point: | Error A. |

**Description** The xxST command with preceding controller address stops a move in progress on controller xx. The ST command without preceding controller address stops the moves on ALL controllers.

| | | | |
|---|---|---|---|
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | D | — | Execution not allowed. |
| | H | — | Execution not allowed in NOT REFERENCED state. |
| | I | — | Execution not allowed in CONFIGURATION state. |
| **Example** | ST | \| | *Stop moves on all controllers.* |

# SU — Set/Get encoder increment value

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | – | ○ | – | – | – |

**Syntax** **xxSUnn** or **xxSU?**

**Parameters**

| | | | |
|---|---|---|---|
| **Description** | **xx** [int] | — | Controller address. |
| | **nn** [float] | — | Equivalent units to one encoder count. |
| **Range** | **xx** | — | **1** to **31** |
| | **nn** | — | $> 10^{-6}$ and $< 10^{12}$ |
| **Units** | **xx** | — | None. |
| | **nn** | — | Units. |
| **Defaults** | **xx** Missing: | | Error B. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |
| | **nn** Missing: | | Error C. |
| | Out of range: | | Error C. |

**Description** The SU command sets the value for one encoder count. It defines also the system of units for all other parameters like travel limits, etc. Therefore, it is the first parameter to be defined for any positioner.

Example: For a positioner with an encoder resolution of 1 µm, the command xxSU0.001 sets 1 encoder count = 1 µm = 0.001 unit or 1 unit = 1 mm.

**Returns** If the sign "**?**" takes place of **nn**, this command returns the current programmed value.

| | | | |
|---|---|---|---|
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | C | — | Parameter missing or out of range. |
| | D | — | Execution not allowed. |
| | H | — | Execution not allowed in NOT REFERENCED state. |
| | J | — | Execution not allowed in DISABLE state. |
| | K | — | Execution not allowed in READY state. |
| | L | — | Execution not allowed in HOMING state. |
| | M | — | Execution not allowed in MOVING state. |

**Example** 1SU7.5e-6 | *Set controller #1 encoder increment to* $7.5 * 10^{-6}$ *units.*

# TB — Get command error string

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
|  | ● | ● | ● | ● | ● |

| | | | |
|---|---|---|---|
| **Syntax** | **xxTBnn** | | |
| **Parameters** | | | |
| **Description** | **xx** [int] | — | Controller address. |
| **Range** | **xx** | — | **1** to **31** |
| | **nn** [char] | — | Error code (refer to TE command). |
| **Units** | **xx** | — | None. |
| **Defaults** | **xx** | Missing: | Error B. |
| | | Out of range: | Error B. |
| | | Floating point: | Error A. |
| | **nn** | Missing: | Returns explanation of current error. |
| | | Out of range: | Error C. |
| **Description** | The TB command returns a string that explains the meaning of the error code **nn** (see TE command for complete list). | | |
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | C | — | Parameter missing or out of range. |
| | D | — | Execution not allowed. |
| **Rel. Commands** | **TE** | — | Get error code. |

**Example** 1TB@ | *Get explanation to error code @.*

*1TB@ No error | Controller returns: @ = means no error.*

# TE — Get last command error

| Usage | Not Ref. | Config. | Disable | Ready | Motion |
|---|---|---|---|---|---|
| | ● | ● | ● | ● | ● |

| | | | |
|---|---|---|---|
| **Syntax** | **xxTE** | | |
| **Parameters** | | | |
| **Description** | **xx** [int] | — | Controller address. |
| **Range** | **xx** | — | **1** to **31** |
| **Units** | **xx** | — | None. |
| **Defaults** | **xx** Missing: | | Error B. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |

**Description** The TE command returns the currently memorized error. When a command is not executable, it memorizes an error. This error can be read with the TE command. After the execution of a TE command, the error buffer gets erased and another TE command will return @, means no error. When a new command error is generated before the previous command error is read, the new command error will overwrite the current memorized error.

For a safe program flow it is recommended to always query the command error after each command execution.

| | | | |
|---|---|---|---|
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | D | — | Execution not allowed. |
| **Rel. Commands** | **TB** | — | Get error string. |
| **Example** | 1TE | \| | *Get last error memorized on controller #1.* |
| | | \| | *Controller returns: 1TE@, means no error.* |

List of errors and corresponding strings (see TB command):

| | | |
|---|---|---|
| @ | — | No error. |
| A | — | Unknown message code or floating point controller address. |
| B | — | Controller address not correct. |
| C | — | Parameter missing or out of range. |
| D | — | Command not allowed. |
| E | — | Home sequence already started. |
| G | — | Displacement out of limits. |
| H | — | Command not allowed in NOT REFERENCED state. |
| I | — | Command not allowed in CONFIGURATION state. |
| J | — | Command not allowed in DISABLE state. |
| K | — | Command not allowed in READY state. |
| L | — | Command not allowed in HOMING state. |
| M | — | Command not allowed in MOVING state. |
| N | — | Current position out of software limit. |
| S | — | Communication Time Out. |
| U | — | Error during EEPROM access. |
| V | — | Error during command execution. |

# TH — Get target position

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | ● | ● | ● | ● | ● |

| | | | |
|---|---|---|---|
| **Syntax** | **xxTH** | | |
| **Parameters** | | | |
| **Description** | **xx** [int] | — | Controller address. |
| **Range** | **xx** | — | **1** to **31** |
| **Units** | **xx** | — | None. |
| **Defaults** | **xx** | Missing: | Error B. |
| | | Out of range: | Error B. |
| | | Floating point: | Error A. |
| **Description** | The TH command returns the value of the target position. This is the position where the positioner should be. | | |
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | D | — | Execution not allowed. |
| | H | — | Execution not allowed in NOT REFERENCED state. |
| | I | — | Execution not allowed in CONFIGURATION state. |
| **Rel. Commands** | TP | — | Get current position. |
| **Example** | 1TH | \| | *Get target position of controller #1.* |
| | *1TH0* | \| | *Controller returns: target position = 0 units.* |

# TP — Get current position

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | ● | ● | ● | ● | ● |

| | | | |
|---|---|---|---|
| **Syntax** | **xxTP** | | |
| **Parameters** | | | |
| **Description** | **xx** [int] | — | Controller address. |
| **Range** | **xx** | — | **1** to **31** |
| **Units** | **xx** | — | None. |
| **Defaults** | **xx** | Missing: | Error B. |
| | | Out of range: | Error B. |
| | | Floating point: | Error A. |

**Description** The TP command returns the value of the current position. This is the position where the positioner actually is according to his encoder value. In MOVING state, this value always changes. In READY state, this value should be equal or very close to the target position.

Together with the TS command, the TP command helps evaluating whether a motion is completed.

| | | | |
|---|---|---|---|
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| | D | — | Execution not allowed |
| | H | — | Execution not allowed in NOT REFERENCED state. |
| | I | — | Execution not allowed in CONFIGURATION state. |
| **Rel. Commands** | **TH** | — | Get target position. |
| **Example** | 1TP | \| | *Get current position of controller #1.* |
| | *1TP0* | \| | *Controller returns: actual position = 0 units.* |

# TS — Get positioner error and controller state

| Usage | Not Ref. | Config. | Disable | Ready | Motion |
|---|---|---|---|---|---|
| | ● | ● | ● | ● | ● |

**Syntax** **xxTS**

**Parameters**

**Description** **xx** [int] — Controller address.

**Range** **xx** — **1** to **31**

**Units** **xx** — None.

**nn** — None.

**Defaults** **xx** Missing: Error B.

Out of range: Error B.

Floating point: Error A.

**Description** The TS command returns the positioner error and the current controller state.

**Returns** The TS command returns six characters (1TSabcdef). The first 4 characters (abcd) represent the positioner error in Hexadecimal. The last two characters (ef) represent the controller state.

<u>**Error code (abcd):**</u> Convert each hexadecimal to a binary:

| F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1111 | 1110 | 1101 | 1100 | 1011 | 1010 | 1001 | 1000 | 0111 | 0110 | 0101 | 0100 | 0011 | 0010 | 0001 | 0000 |

F

Each bit represents one possible error:

| A | | | | B | | | | C | | | | D | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| • Not used | • Not used | • Not used | • Not used | • Not used | • Not used | • Not used | • Not used | • No parameters in memory | • Not used | • Motion Time out | • Not used | • Not used | • Not used | • Not used | • Not used |

Examples:

- Error map 0000 = No errors
- Error map 0020 = Motion time out

**Controller states (ef):**

- **0A**: NOT REFERENCED from reset.
- **0B**: NOT REFERENCED from HOMING.
- **0C**: NOT REFERENCED from CONFIGURATION.
- **0D**: NOT REFERENCED from DISABLE.
- **0E**: NOT REFERENCED from READY.
- **0F**: NOT REFERENCED from MOVING.
- **10**: NOT REFERENCED no parameters.
- **14**: CONFIGURATION.
- **1E**: HOMING.
- **28**: MOVING.
- **32**: READY from HOMING.
- **33**: READY from MOVING.
- **34**: READY from DISABLE.
- **3C**: DISABLE from READY.
- **3D**: DISABLE from MOVING.

---

**NOTES**

**THE ERROR BUFFER GETS UPDATED PERIODICALLY, APPROX. EVERY 1 MS.**

**THE TS COMMAND READS THE ERROR BUFFER AND CLEARS THE ERROR BUFFER AT THE SAME TIME (SAME AS FOR COMMANDS TE, TB). SO WHEN LAUNCHING THE TS COMMAND, IT IS IMPORTANT TO PROCESS THE TS FEEDBACK ACCORDINGLY.**

**THE ERROR “NO PARAMETERS” GETS ONLY DETECTED DURING THE BOOTING OF THE CONTROLLER. WHEN READ THE ERROR IS CLEARED.**

**With no errors in the error buffer the color of the LED will change from red to either green or orange depending on the controller state.**

---

| | | | |
|---|---|---|---|
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| **Rel. Commands** | **TE** | — | Get last error. |
| **Example** | 1TS | \| | *Get error and state of controller #1.* |
| | *1TS00000A* | \| | *Controller returns: no errors and NOT REFERENCED from reset.* |

# VE — Get controller revision information

| **Usage** | **Not Ref.** | **Config.** | **Disable** | **Ready** | **Motion** |
|---|---|---|---|---|---|
| | ● | ● | ● | ● | ● |

| | | | |
|---|---|---|---|
| **Syntax** | **xxVE** | | |
| **Parameters** | | | |
| **Description** | **xx** [int] | — | Controller address. |
| | **nn** [string] | — | Action. |
| **Range** | **xx** | — | **1** to **31** |
| **Units** | **xx** | — | None. |
| **Defaults** | **xx** Missing: | | Error B. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |
| **Description** | This command returns the controller's revision information. | | |
| **Errors** | A | — | Unknown message code or floating point controller address. |
| | B | — | Controller address not correct. |
| **Rel. Commands** | **TP** | — | Get current position. |
| **Example** | 1VE | \| | *Get controller #1 revision information.* |

*1VE CONEX-AGP V1.0.0. | Controller returns revision number*

# ZT — Get all configuration parameters

| Usage | Not Ref. | Config. | Disable | Ready | Motion |
|---|---|---|---|---|---|
| | ● | ● | ● | – | – |

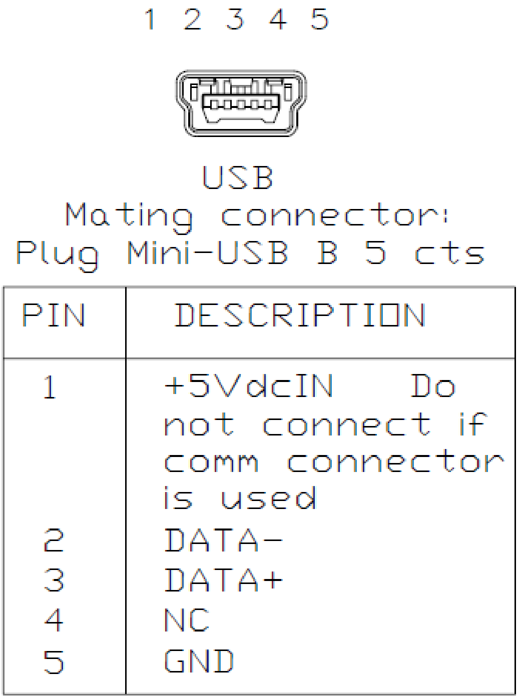
| | | | |
|---|---|---|---|
| **Syntax** | **xxZT** | | |
| **Parameters** | | | |
| **Description** | **xx** [int] | — | Controller address. |
| **Range** | **xx** | — | **1** to **31** |
| **Units** | **xx** | — | None. |
| **Defaults** | **xx** Missing: | | Error B. |
| | Out of range: | | Error B. |
| | Floating point: | | Error A. |
| **Description** | The ZT command returns the list of all current configuration parameters.<br>The ZT command allows a quick review of all current stage parameter and simplifies the configuration of non Newport stages, for instance by using Hyper Terminal file transfer. | | |
| **Errors** | A | — | Unknown message code or floating point controller address |
| | B | — | Controller address not correct |
| **Rel. Commands** | **TE** | — | Get error code. |

**Example**

```
        1ZT   |   Get controller #1 configuration data.
       1PW1
1DB0.000075
      1KP10
          …
       1HT1
       1PW0
```

# 3.0 Connector interfaces

## 3.1 USB (Male mini-USB)



Mating connector:
Plug Mini-USB B 5 cts

| PIN | DESCRIPTION |
|---|---|
| 1 | +5VdcIN Do not connect if comm connector is used |
| 2 | DATA- |
| 3 | DATA+ |
| 4 | NC |
| 5 | GND |

# Service Form

**Your Local Representative**

Tel.: ___________________

Fax:____________________

Name: ____________________________________________________

Company:__________________________________________________

Address:__________________________________________________

Country:__________________________________________________

P.O. Number:______________________________________________

Item(s) Being Returned:___________________________________

Model#: __________________________________________________

Return authorization #: ______________________________________

*(Please obtain prior to return of item)*

Date: ________________________________________________________

Phone Number: ________________________________________________

Fax Number: __________________________________________________

Serial #: ____________________________________________________

Description: _______________________________________________________________________________________

Reasons of return of goods (please list any specific problems): ___________________________________________

Newport®

Experience | Solutions

Visit Newport Online at:
**www.newport.com**

**North America & Asia**
Newport Corporation
1791 Deere Ave.
Irvine, CA 92606, USA

**Sales**
Tel.: (800) 222-6440
e-mail: sales@newport.com

**Technical Support**
Tel.: (800) 222-6440
e-mail: tech@newport.com

**Service, RMAs & Returns**
Tel.: (800) 222-6440
e-mail: rma.service@newport.com

**Europe**
MICRO-CONTROLE Spectra-Physics S.A.S
9, rue du Bois Sauvage
91055 Évry CEDEX
France

**Sales**
Tel.: +33 (0)1.60.91.68.68
e-mail: france@newport-fr.com

**Technical Support**
e-mail: tech_europe@newport.com

**Service & Returns**
Tel.: +33 (0)2.38.40.51.55