Venus-2
command language

for
Pollux
high resolution positioning controller

Pollux
Pollux NT

User Manual

Contents

Velocity and acceleration

Positioning commands

Limit Switch functions

Safety functions

Position / origin / coordinate system

Current status inquiries

Error messages

System commands

Special functions

Closed loop function

Backlash compensation

CAN configuration

Venus-2 command list

Preface

About this documentation

The interpreter language Venus-2 of the controller Pollux is described in this manual. Its function and syntax is explained. The grouping of the commands in function groups improves the overview, an index table and alphabetical command list gives additional assistance.

The mechanisms of the command execution are commented in an introduction. To study this chapter is important as the way of procedure for the correct programming is explained there.

Symbols in this documentation

To clarify the content following symbols are used.

Symbol	Description
	Warning. This information must be observed strictly.
	Important information.
Option	Indicates that this function can be enabled or upgraded with a release code.
Option	This function must be installed by the factory or experts.
Venus-2	Venus-2 commands are indicated with this formatting style.
	Settings that cannot be stored with command *nsave* into the flash memory.
	This configuration can be stored with command *nsave* into the controller flash memory.

Introduction to Venus-2

Venus-2 is an interpreter language

Venus-2 commands consist of ASCII-signs which are interpreted in the controller and immediately executed.
A software development surrounding to produce the control programs is not needed.
The commands can be produced by any host and whatever programming language you are using, on condition that there is an access to the RS-232 interface.
In the simplest way the commands are directly transmitted to the controller via an ASCII terminal.

History

Venus-2 has been developed on the basis of the interpreter language Venus-1. The fundamental command construction is identical.
The expansion was necessary as the fundamental structures of Venus-1 are designed for controller with at most three linear interpolated axes; but the controller Pollux supports any number of independent axes (n). Venus-1 commands whose structure is designed for operating n-axes are taken over in Venus-2 without any syntax alteration.
The other commands are expanded with the addition "n".
For example: The Venus-1 command *cal* which has at the same time an effect on three axes has become the axis specific command *ncal* in Venus-2, *move* has become *nmove* etc.

Command syntax

The commands are assembled following this scheme:

[parameter] _ [axes index] _ [command] _

blank ▶   blank ▶   blank ▶

or

[parameter] _ [axes mask] _ [command] _

blank ▶   blank ▶   blank ▶

_ blank, (space) or (SP)
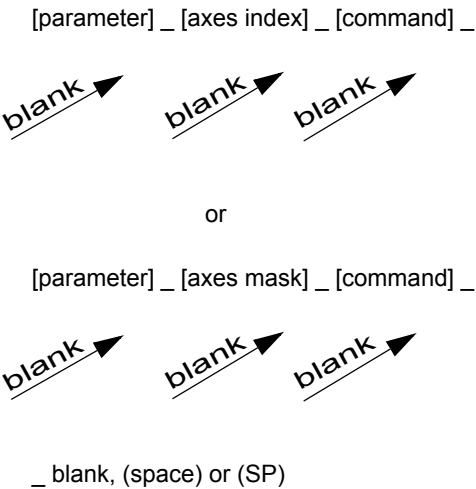
Parameter

The parameter transmits a value without any unit.
For positioning commands *e.g.* this value is the target
coordinate or the value for the relative move.
For parameterizing a decimal number or a switch with 0 or 1
is possible. If several parameters are prescribed for one com-
mand, they have to be divided by a blank.

The following numbers and characters are permitted for
parameters:

Letters	not allowed
Numbers	0-9
Characters *	+ - .

*comma is not allowed as a decimal separator.

Axis number

Each Pollux axis has an unique address in order to communicate with the host.
The axis number is assigned with command *setaxisno*.

Syntax of axis number :Axis mask

numbers	letters	characters
1 .. 16	not allowed	not allowed

To trigger synchronous moves or parameterizing several axes, the axis mask is used.

Syntax axis mask:

Numbers	Letters	Characters
-1 .. - 65535	not allowed	- (minus)

Axis mask calculation:

Axis mask [n][m] = - (2^{n-1}) - (2^{m-1})

Axis	Mask
1	-1
2	-2
3	-4
4	-8
5	-16
.	.
16	-32767

Example:

The velocity 5.2 mm/s is to be configured for
axis-1 and axis-5.

Axis mask calculation:
Axis mask [1] [5] = - 2^{1-1} - 2^{5-1} = -17

Command:
5.2 -17 snv

Command

The command name is intended to describe the function as
clear as possible. It consists of several ASCII signs,
capitalization is distinguished.

Following letters are used for commands:

ASCII characters	umlauts	numbers
a..z; A...Z	no	no

Command ending characters

Transmitting

The Venus-2 command must be terminated with a space (*sp*).

[Parameter] *sp* [Axes index] *sp* [*Venus-2 command*] *sp*

Receiving

Data which are replied from the controller are seperated with
a space and completed with ASCII [CR][LF].

[1st parameter] sp [2nd parameter] sp [n-parameter] [CR] [LF]

Table of important ASCII signs for programming

ASCII Code	Sign	Dec	HEX
CR	Ctrl-M	13	0xD
LF	Ctrl-J	10	0xA
SP		32	0x20
ETX	Ctrl-C	3	0x3
STX	Ctrl-B	2	0x2

Number format

One thing that has to be observed strictly especially with Pollux controllers is to enter each number with the particular format it requires.

- Values that are integers by nature, such as bit-coded numbers, indexes, mode selectors etc., have to be entered as integers, i.e. with no decimal point.

- Values that are rational by nature, such as position, velocity, acceleration, time, etc., should be entered as fixed point values, utilizing a decimal point. The integer part of the number is always based on a fixed measurement unit then, such as mm, mm/s, mm/s², ms, etc. Unless otherwise noted, this is the unit given in the respective parameter/command description.

- Due to a technical limitation of the controller, the processing of each number, whatever the format be, is based on an indivisible atomic unit which defines the available resolution. It is, for instance, nm (nanometers) with position values. If a rational number is entered as an integer, it is assumed to be based on that unit.

- The atomic unit or resolution, respectively, can, as well as the value itself, be determined by parameter inquiry. By way of example we take the accelaration:

Inquiry: 1 gna -> Reply: 1000.000

According to the command description, the measurent unit is mm/s². Thus the current setting at axis 1 is 1000 * 1 mm/s² = 1 m/s²; the last decimal place following the decimal point represents the atomic unit, which is 1 µm/s² in this case. So

1000. 1 sna will effect **1 m/s²** accleration at axis 1, whereas **1000** 1 sna will effect 1000 * 1 µm/s² = **1 mm/s²**.

Data input format for move and speed commands

The unit of the target coordinate or speed value is interpreted in **mm** or **mm/s**.
If no decimal point is used the value would be identified as **nm** or **nm/s**.

It is recommend to enter the positioning values with a decimal point.

Examples:

Decimal point is used

*100.00_ 1_ nm_*
*1._ 2_ nr_*
 // Axis 1 is moving to the absolute coordinate 100 **mm**
 // Axis 2 is moving relative 1 **mm**

Without decimal point

*100_1_nm_*
*1_2_nr_*

 // Axis 1 is moving to the abs. coordinate 100 **nano meter**
 // Axis 2 is moving relative 1 **nano meter**

 _ = space

Data input format for dynamics and pitch settings

The parameter for velocity, acceleration and spindle pitch should also be terminated with a decimal point to make sure that these values are identified as mm/s, mm/s² or mm. Following table shows the effect if a decimal point is used or not.

Function	Venus-2 command	decimal point used	no decimal point used
acceleration	sna	mm/s²	µm/s²
velocity	snv	mm/s	nm/s
pitch	setpitch	mm	0.1 µm*

*since Pollux-1 firmware version 4.2.0 and Pollux-2 firmware version 2.3.0; 1 µm before

It is recommend to enter the values of above settings with a decimal point.

Example

*100.00_1 sna_*

// Acceleration setting for Axis-1 is 100mm/s²

_ = space

Venus-2 Command execution

The ASCII data, transmitted by a host run through the following areas of the controller:

- **Data input FIFO**
- **Scanner / Stack**
- **Interpreter**

Data input FIFO

The commands from the host are sent simultaneously to the data input FIFOs of all Pollux controllers installed.
The data input memory of each controller is able to accept up to 100 signs.
There is no data flow control during the transmission of the data, i.e. an overflow of the FIFO would not be recognized. For that reason the character transmission rate is limited. Error code 1010 (->command *gne*) is set if once the FIFO holds more than 70 characters. The FIFO is cleared with command *nreset* or if the power is disconnected.

Scanner / Interpreter / Parameter Stack

The data input FIFO is read by the scanner and checked for parameters, commands and axes numbers.
The parameters are pushed to the parameter stack which can accept up to 99 values.
If the scanner recognizes a valid command, it is passed to the interpreter. The interpreter will remove the assigned parameters from the stack during execution of the command.

If the axes number does not correspond with the axis, the command is deleted by the scanner and the parameters are removed from the stack.
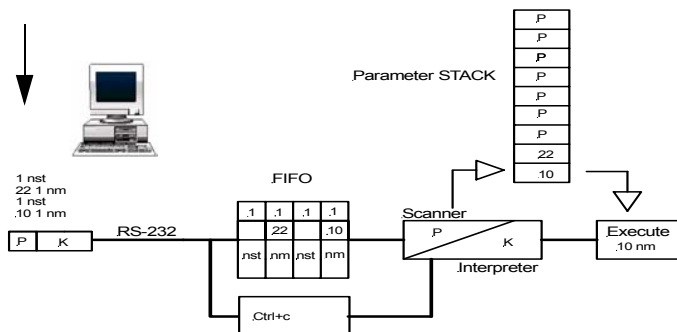
Note that when a Pollux controller without any more free parameter stack space left is still receiving data, it will automatically clear the stack and block the move processing, i.e. discard any further move commands. The *getaxis* command will return 0 then. This is likely to occur if the host continuously transmits command lines with more parameters than necessary and should be avoided since data loss is probable. To restore full operation, stop data transmission, then utilize the *setaxis* command. Error code 1009 (->command *gne*) is set if once the stack holds more than 90 parameters.

Example to clarify the stack principle:

The following command is executed: *10.123 1 nm*

1 ngsp -> 0   // no elements on the stack
10.123 _ 1 _   // parameters are send to the controller
1 ngsp -> 2 // Two elements are pushed on the stack
nm   // Venus-2 command send to the controller
1 ngsp -> 0 // all elements on the stack are removed.

Pushing up to 98 coordinates and corresponding axis numbers on the stack is theoretically possible. Each of these coordinates can be executed with a *nmove* command.

10.0 1 // coordinate and axisno
12.0 1 // coordinate and axisno
40.0 1 // coordinate and axisno
80.0 2 // coordinate and axisno *

nm // move to 80.0 Axis-2
nm // move to 40.0 Axis-1
nm // move to 12.0 Axis-1
nm // move to 10.0 Axis-1

Blocking and non blocking commands

While the controller performes a move, it is able to execute some Venus commands such as *nst*, *np,* and *Ctrl-C* in parallel to it. Those commands are called non-blocking commands because they are executed immediately and are not blocking the data input FIFO.

On the other hand, there are blocking commands whose execution will be delayed until the ongoing move is finished. Such pending commands will block the data input FIFO for all subsequent commands.

In other words, if a blocking command is transmitted to an axis while a move is performed, e.g. "1 gne", it is stored in the data input FIFO and will block all following commands until execution of the blocking command has been completed.

Example: status inquiry unblocked

```
100.0 1 nm              // move performed
1 nst                   // 1 nst will immediatelly
                        // executed
```

Example: status inquiry is blocked with 1 gne

```
100.0 1 nm              // move performed
1 gne                   // asks for error code
1 nst                   // 1 nst is replied if commands
                        //  100.0 1 nm and 1 gne
                        //  are executed
```

Examples of blocking and non blocking commands

The interpreter is able to execute non-move commands while a move is in progress.

Below mentioned, the interpreter executes the instruction *nmove* and is also free to process 3x *nst* and *np*.

FIFO					Scanner/Interpreter
.nst	.nst	.nst	.np	→	.nmove
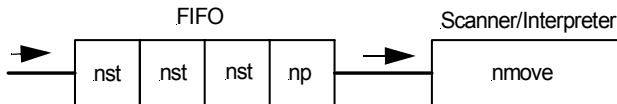
The interpreter has been blocked by the command *gne*.

The interpreter is executes the command *nmove*. The FIFO contains the command *gne*, this blocks the interpreter for the execution of further commands until *nmove* is completed After *gne* has been executed, the *nst* commands are processed.

.FIFO					Scanner/Interpreter
.nst	.nst	.nst	**gne**	✕	.nmove
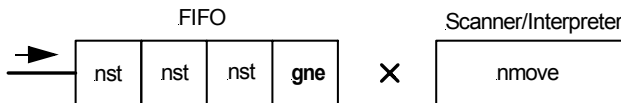
Which commands do block the interpreter?

- All move commands **block** the interpreter.

- Storage and restorage commands **block** the interpreter for operational safety reasons, in particular:

> - *nsave*
> - *nrestore*
> - *getnfpara*

All other system commands are **non-blocking**.

- Some parameterization commands **block** the interpreter for operational safety reasons, in particular:

> - *setnpos*
> - *setaxisno*
> - *setcloop*
> - *setclperiod*
> - *setpitch*
> - *setpolepairs*
> - *setmotiondir*
> - *setscaleinterface*
> - *setncode*
> - *setaxis*
> - *setphases*
> - *setref*
> - *setblc*
> - *setblcd*
> - *setblcs*

All other parameterization commands are **non-blocking**.

- Inquiry commands except for *gne* **do not block** the interpreter.

- Emergency shortcuts (*Ctrl-B, Ctrl-C*) **do not block** the interpreter.

Immediate move termination and command FIFO unlocking

The *Ctrl-C* and *Ctrl-B* shortcuts are designed for immediate termination of currently running moves. As opposed to the *nabort* command, they bypass the command FIFO. A *Ctrl-C* or *Ctrl-B* terminate the currently running move and, if applicable, unlock a blocking FIFO, thereby enabling pending commands in the FIFO to be executed. Consult the command reference (safety functions) for further shortcut explanation.

Example:
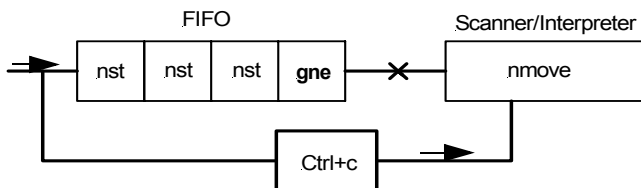
The currently executed command *nmove* is terminated by *Ctrl-B.* The interpreter is unlocked to execute the following command *gne*.

Ctrl-C and *Ctrl-B* are broadcast commands, affecting all axes simultaneously.

Performing an automatic status reply message

With the following sequence of instructions the blocking effect of some commands can be used to generate an automatic status reply (applies here to the 1. axis):

10 1 nmove
0 1 nr
1 nst

Effect:

An automatic status feedback is produced, after the instruction ***10 1 nmove*** is processed.

Description:

The instruction ***0 nr*** blocks the interpreter and prevents the execution of nst. After ***nmove*** is executed, immediately ***0 nr*** is processed with no effect, because it is a relative positioning with 0mm.
Afterwards ***1 nst*** produces the desired status feedback.
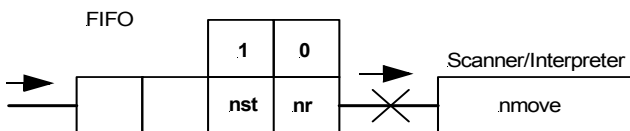
0 nr is only used to block ***1 nst*** until the ***nmove*** is finished.

Quasi-simultaneous move start in a network

The overall move start delay in a Pollux network can be reduced signifantly by first pushing the target position coordinates on the concerned controllers' parameter stacks and then "broadcast" one move command separately, utilizing the axes mask feature. Then the overall move start delay offset among all axes will usually not exceed 250 µs. The maximum delay depends, however, on the composition of the network and is therefore not guaranteed.

The feature is useful for starting moves quasi-simultaneously, triggered by the commands *nm, nr*, or *speed*.

Procedure to establish a quasi-simultaneous move start:

1. Push relative or absolute coordinates on relevant stacks
2. Calculate axes mask to select the relevant axes
3. Start move with a move command

Pushing the stacks:
[value] [axisno] *npush*
[value] [axisno] *npush*
[value] [axisno] *npush*

Triggering an absolute move
[-axes mask] *nm*

Triggering a relative move
[-axes mask] *nr*

Triggering a speed move
[-axes mask] *speed*

	Description
[value]	absolute/ relative coordinates or speed value
[-axes mask]	mask to select the relevant axes*
[axisno]	axis number 1...16

For axis mask calculation, see chapter "Introduction to Venus-2".

Example task:

Axis-1, Axis-3 and Axis-5 are to be moved quasi-synchronously relative to their current coordinates. Axis-1 velocity and acceleration settings are assumed to be fixed at 20 mm/s and 100 mm/s², respectively.

Axis-1 move distance: 10mm
Axis-3 move distance: 20mm
Axis-5 move distance: 30mm

Matching Axis-2 and Axis-3 velocity and acceleration:
40.0 3 *snv*
200.0 3 *sna*
60.0 5 *snv*
300.0 5 *sna*

Pushing the individual move distances on the stacks:
10.0 1 *npush*
20.0 3 *npush*
30.0 5 *npush*

Mask calculation:
$2^0 + 2^2 + 2^4 = 21$

Trigger command:
-21 nr

Axis-1, Axis-3, and Axis-5 will now start moving quasi-simultaneously. The target coordinates will be reached nearly simultaneously after 0,7 seconds. However, since there is no synchronisation, the overall delay in reaching the target coordinates is not defined.

Principles of
communication

Pollux / Pollux NT RS-232 communication

Pollux can operate as a single controller or can be daisy chained to a multiple of maximum 16 controllers.

The RxD Input of the controller is always enabled to listen for command messages. The TxD output is tri-stated and only active if the controller replies.

For command validation the axis index is used.

The index number can be configured and stored in the controller. (See command *setaxisno*) Default index = 1

If several Pollux controllers are working at the same RS-232 line via daisy chaining, each controller has to be provided with an individual index number.

As the data bus connects all axis modules with equal rights, there is the hazard that several Pollux modules place at the same time data on the bus which might cause a data collision. Because of that it is strictly recommended that requests are only transmitted to the controller if the reply message of the preceding request has been received.

The user programming has to guarantee that it may not lead to any data collision.
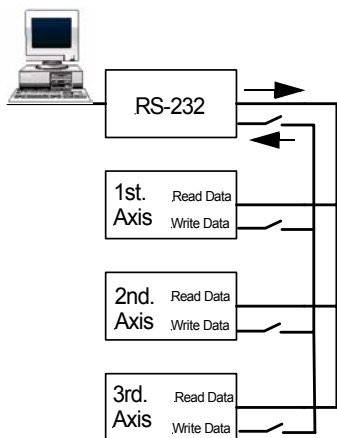
Pollux NT CAN communication

An upgrade option available with Pollux NT controllers is the activation of the Controller Area Network (CAN). With this interface, just as with RS-232 communication, data and command strings are assembled by the Venus-2 interpreter.

The CAN reception address can be configured utilizing *setcanbaseadr*. The controller - in return - will send data to the CAN target address specified by *setcananswadr* (answer address) if needed. The CAN baud rate is set utilizing *setcanbaudrate*.

With the individual reception addresses set to be equal at all Pollux NT nodes included in a network, and the individual answer addresses set to a common host address, communication is virtually the same as with RS-232 usage.
By contrast, if all controllers feature different reception and answer addresses, this enables the host to set up a separate communication channel to each individual controller. This can be quite useful with some applications.

Example:

Two Pollux NT controllers (Controllers 1 and 2) have to be configured to operate in a CAN at 100 kBaud. Axis numbers are assumed to be set to 1 at Controller 1 and to 2 at Controller 2. The host offer addresses 300 ff. (decimal) for receiving command replies. Available target controller addresses be 200 ff. For CAN configuration, the controllers are put into an RS-232 daisy chain at any one time.

Setting of the 100 kBaud transmission rate is as follows:

```
3 1 setcanbaudrate
3 2 setcanbaudrate
save
reset
1 getcanbaudrate -> 3
2 getcanbaudrate -> 3
```

1. Common address operation

Setting of the reception address and corresponding answer address is as follows:

200 1 setcanbaseadr
200 2 setcanbaseadr
300 1 setcananswadr
300 2 setcananswadr
save
reset
1 getcanbaseadr -> 200
2 getcanbaseadr -> 200
1 getcananswadr -> 300
2 getcananswadr -> 300

Both controllers will now receive and process everything that comes in at CAN address 200 (decimal) at 100 kBit. If needed, commands will be answered at CAN address 300 (decimal).

The communication procedure is shown in the following example. Axes 1 and 2 (now in the CAN) shall be moved to coordinates 10 mm and 20 mm, respectively. Correct processing is to be verified utilizing error and position checking.

Host output at address 200:

(1) 10 1 nm 1 gne 1 np <CR>

If there is no exception during the requested move, reply at address 300 will be:

0   (gne answer)
10.000000   (np answer)

after move termination. Axis 1 slide will now be at position coordinate 10 mm.

Host output at address 200:

(2) 20 2 nm 2 gne 2 np <CR>

If there is no exeception during the requested move, reply at address 300 will be:

0   (gne answer)
20.000000   (np answer)

after move termination. Axis 2 slide will now be at position co-ordinate 20 mm.

Note that with this mode of operation (as would be with RS-232), termination of move and axis reply of line (1) have to be awaited before the network is ready for transmission of line (2) because otherwise unique assignment of the axes' replies would be impossible.

2. Individual address operation

Now the controllers' CAN interfaces shall be configured indi-vidually. Instead of the common addresses 200 and 300 above, reception addresses are set to (200 + axis number), answer addresses to (300 + axis number):

201 1 setcanbaseadr
202 2 setcanbaseadr
301 1 setcananswadr
302 2 setcananswadr
save
reset
1 getcanbaseadr -> 201
2 getcanbaseadr -> 202
1 getcananswadr -> 301
2 getcananswadr -> 302

As opposed to the common address operation described above, commands transmitted by host at address 201 will now exclusively be processed by Controller 1 and (if needed) replied to at address 301, and commands transmitted by host at address 202 will exclusively be processed by Controller 2 and replied to at address 302.

The same positioning task as described above can now be dealt with as follows.

Host output at address 201:

(1) 10 1 nm 1 gne 1 np <CR>

Host output at address 202:

(2) 20 2 nm 2 gne 2 np <CR>

If there is no exeception during the requested moves, replies will be:

0	(gne answer)
10.000000	(np answer)

at address 301 and

0	(gne answer)
20.000000	(np answer)

at address 302 after move termination at respective axis. Axis slides will be at the corresponding position coordinates.

Note that with this mode of operation, lines (1) and (2) can be transmitted back-to-back or even simultaneously because reply assignment is provided by answer address. Also note that - for compatibility reasons - the target axis number, though now redundant, is still to be delivered correctly with each command transmission.

In the long run, mixed communication, i.e. permanently using the RS-232 and CAN interfaces alternately or simultaneously, will cause internal data corruption. Therefore it is not recommended. With CAN operation, the RS-232 interface should be considered a mere auxiliary for CAN configuration.

Communicaton requirements during power up

After power up the controller requires a total of 4 seconds to be in a proper state.This time is divided in two parts:

- Wait time if a firmware update is performed
- Booting time till the motor is energised

Time polling for update   Time booting device

Power Up

0s   2s   4s
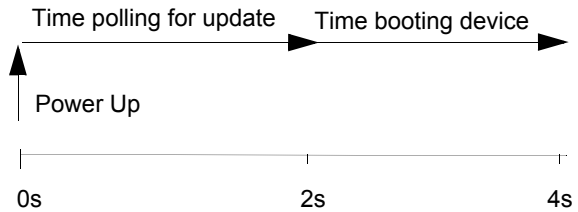
Apart from an update, it is strictly recommended to disable the communication between host and controller until the controller has finished the first two seconds after power up. If any data are transmitted in this time the controller boot procedure will be delayed.

Hardware identification and command scope

Pollux controller types

Regarding controller hardware and firmware, there exist two different types of Pollux controllers. In this documentation, they are referred to as "Pollux" on one hand and "Pollux NT" on the other. The Pollux NT controller is more powerful, where the main difference is the optional activation of the following features:

- position measurement support / position control
- Control Area Network (CAN) interface

Therefore, some of the commands described in the following are applicable to Pollux NT controllers only. You can tell the validity extent of each command from the respective command description header.

Pollux controller identification

To pinpoint the hardware type of a Pollux controller, use the *nidentify* command. You can tell the controller type from the first portion of the return string.

return string portion	controller type
Pollux	Pollux controller
Pollux2	Pollux NT controller

Basic settings

setpitch

Description:

The command *setpitch* adapts the controller to the transmission ratio of the drive train. The entered value is assembled with a step width of 0.1 µm (1 µm up to Pollux-1 Rev. 4.1.4 and Pollux-2 Rev. 2.2.0). Any further decimals will be discarded (value round-off).

$$pitch = \frac{\text{resulting move distance}}{\text{number of motor revolutions}}$$

The change of pitch during normal operation may lead to a position leap if the axis is not located at its initial position. The amount of the leap depends on the current position.

Syntax:

[pitch] [axisno] *setpitch*

entry	range	unit
[pitch]	0.1 (1*) to 50	mm
[axisno]	1...16	-

* see description above

Related command:

getpitch

Example:

4.009 1 setpitch

Further examples see in the following page

Examples:

Drive mechanism with ball screw at Axis-1:

Lead screw with pitch = 2mm
Each motor revolution produces a move distance of 2mm
Pitch = 2mm / 1 rev. = 2

Setting : *2.0 1 setpitch*

Drive mechanism with lead screw and gear box

Lead screw pitch = 4mm
Gear = 20:1
20 motor revolutions produce a move distance of 4mm
Pitch = 4mm / 20 rev. = 0.2

Setting : *0.2 [axis] setpitch*

getpitch

Description:

The command *getpitch* returns the pitch setting of the axis.

Syntax:

[axisno] *getpitch*

entry	range
[axisno]	1 ... 16

Reply:

[pitch]

	unit
[pitch]	mm

Example:

2 getpitch
Reply
4.009

setumotmin

Description:

The command *setumotmin* determines the motor phase voltage at standstill. It can be used to vary the holding torque. The setting also effects the torque during motion.

CAUTION: To be handled with great care! A higher [vmin] value will increase the motor voltage and implies an increased motor phase current and will produce more overall torque. It also increases power consumption at the motor and motor driver which may sustain damage with a mismatched setting. Settings from 2000 through 4000 have emerged to be practicable in most cases. Regard there is no limitation by firmware. Do not raise the setting beyond what is necessary for proper operation.

Syntax:

[vmin] [axisno] *setumotmin*

entry	range	unit
[vmin]	< 4000 *	mV
[axisno]	1...16	-

* works for most cases; no limitation by firmware though - can be raised up to 24000 if needed

Related command:

getumotmin, setumotgrad

Example:

1800 1 setumotmin

getumotmin Pollux | Pollux NT

Description:

The command *getumotmin* returns the setting of the motor phase voltage at standstill.

Syntax:

[axisno] *getumotmin*

entry	range
[axisno]	1...16

Reply:

[vmin]

	range	unit
[vmin]	< 4000 *	mV

* see *setumotmin*

Example:

1 getumotmin

Reply
2000

setumotgrad

Description:

The command **setumotgrad** determines the motor voltage in the middle and upper speed range. It can be used to vary the torque reached during motion.

CAUTION: To be handled with great care! A higher [vgrad] value will increase the motor voltage and implies an increased motor phase current and torque during motion. It also increases power consumption at the motor and motor driver which may sustain damage with a mismatched setting. Settings from 50 through 250 have emerged to be practicable in most cases. Regard there is no limitation by firmware. Do not raise the setting beyond what is necessary for proper operation.

Syntax:

[vgrad] [axis] **setumotgrad**

entry	range
[vgrad]	< 250 *
[axis]	1...16

* works for most cases; no limitation by firmware though - can be raised up to 32767 if needed

Related commands:

getumotgrad, setumotmin

Example:

70 1 setumotgrad

getumotgrad

Description:

The command *getumotgrad* returns the setting of umotgrad.

Syntax:

[axisno] *getumotgrad*

entry	range
[axisno]	1...16

Reply:

[vgrad]

	range
[vgrad]	< 250 *

*see **setumotgrad**

Example:

1 getumotgrad

Reply:
50

setpolepairs

Description:

The command *setpolepairs* adapts the controller to the number of the stepper motor pole-pairs.

The relationship between motor type and pole-pairs is shown in the following table
.

[pole pairs]	description
50	Hybrid Stepper Motor with full step size=1.8°
100	Hybrid Stepper Motor with full step size= 0.9°

Syntax:

[pole pairs] [axisno] *setpolepairs*

entry	range
[pole pairs]	50, 100
[axisno]	1...16

Related command:

getpolepairs

Example:

50 1 setpolepairs

Axis-1 is configured to drive a Stepper Motor with 100 poles or 50 pole-pairs (full step size 1.8°).

getpolepairs

Description:

The command *getpolepairs* returns the configured number of pole-pairs.

Syntax:

[axisno] *getpolepairs*

entry	range
[axisno]	1 16

Reply:

[value]

	range
[pole pairs]	50, 100

Example:

1 getpolepairs

Reply:
50

setphases

Description:

With the command *setphases* the adaption to the number of phases of the connected motor is done. 2-and 3-phase motors are supported. The number of phases has to be entered as an integer value

The standard Pollux controller supports 2-phase motors only.

Syntax:

[phases] [axisno] *setphases*

entry	range
[phases]	2, 3
[axisno]	1...16

Related command:

getphases

Example:

2 2 setphases

The 2nd axis is configured for 2-phase motors.

getphases

Description:

The command **getphases** returns the configured number of motor phases.

Syntax:

[axisno] *getphases*

entry	range
[axisno]	1 16

Reply:

[phases]

	range
[phases]	2, 3

Example:

1 getphases

Reply:
2

setaxis

Description:

The command *setaxis* enables or disables the specified axis for positioning tasks.
This function has no effect to the motor driver. The motor keeps its holding torque also if the axis is disabled.

Note that the axis enable setting can - on some exceptional error conditions - autonomously be cleared by the controller itself. In this case, use the *setaxis* command to restore full operation.

Syntax:

[enable state] [axisno] *setaxis*

entry	range
[enable state]	0, 1, 2
[axisno]	1... 16

[enable state]	description
0	The axis is disabled for all moves.
1	The axis is enabled for all moves.
2	The axis is enabled for all moves but *ncal*, *nrm*

Example:

1 3 setaxis

getaxis

Description:

The command *getaxis* returns the axis enable setting*.*

Syntax:

[axisno] *getaxis*

entry	range
[axisno]	1... 16

Reply:

[enable state]

	range
[enable state]	0, 1, 2

Example:

2 getaxis

setaxisno

Description:

The command *setaxisno* is configurates the individual axis number of the controller.
The axis number is used to adress the controllers in the network configuration.

Procedure to assign the axis number:

1. Make sure only one Pollux controller is connected to the RS-232 interface
2. Assign the axis number with the command *axisno*
3. Store the number with [axis number] *nsave*

Note that each Pollux controller in a network has to be provided with a unique axis number in order to avoid transmission data from being corrupted.

Syntax:

[axisno] *setaxisno*

entry	range
[axisno]	1.... 16

Example:

3 setaxisno
3 nsave

getaxisno

Description:

The command *getaxisno* returns the axis number.

Syntax:

getaxisno

Reply:

[axisno]

	range
[axisno]	1.... 16

Example:

getaxisno

setsw

Description:

The command *setsw* adapts the specified limit-switch input to the connected switch type of the ncal/nrm-switch.

Following settings are possible:
• normally open (no)
• normally closed (nc)
• limit-switch disabled

If a limit-switch input is disabled, it can not accomplish a safety function.

Syntax:

[function] [index] [axisno] *setsw*

[function]	NPN Type *	PNP Type **
0	closer (nc)	opener (no)
1	opener (no)	closer (nc)
2	disabled	disabled

* NPN-Switch is switched to GND
** PNP-Switch is switched to VCC

[index]	description
0	cal switch indexed
1	rm switch indexed

Examples:

0 0 1 setsw
cal limit-switch input of Axis-1 is prepared for a NPN-closed or PNP-open switch.

2 1 2 setsw
rm limit-switch input of Axis-2 is disabled.

getsw

Description:

The command *getsw* returns the setting of the limit-switch inputs. Unlike with the setting command *setsw*, the particular switch inputs do not have to be selected by an index; they are returned in one line.

Syntax:

[axisno] *getsw*

entry	range
[axisno]	1.... 16

Reply:

[function cal] [function rm]

	description
[function cal]	[function] of cal switch
[function rm]	[function] of rm switch

[function]	NPN-Switch *	PNP-Switch **
0	closer (nc)	opener (no)
1	opener (no)	closer (nc)
2	disabled	disabled

Example:

3 getsw

Reply:
0 0

setmotiondir

Description:

With the command *setmotiondir* the factory-assigned motor rotation direction can be reversed.

In case the rotation direction is reversed, the functions of the limit switch inputs will be swapped.
This means that during the calibration move ncal, the controller expects the limit switch activity at the nrm input and during the nrm move at the ncal input.

Motor direction if a move to positive coordinates is executed.
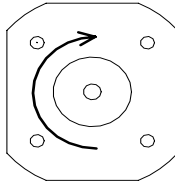
Factory setting
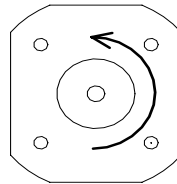
Motor direction is changed with *setmotiondir*

Never change rotation direction with a Pollux-2 controller while it is operating in closed loop mode. First open the position control loop using *setcloop*. Then, after applying *setmotiondir* as described, it is mandatory to also reverse the count direction of the sensor position using *setclperiod*. Then, before closing the loop again, execute a move command and check for consistency of the effected sensor position variation (amount and direction).

[motiondir] [axisno] *setmotiondir*

[motiondir]	description
0	Moving upward by nominal position has the effect of clockwise axis rotation (factory setting)
1	Moving upward by nominal position has the effect of anti clockwise axis rotation

Example:

1 1 setmotiondir

getmotiondir

Description:

The command *getmotiondir* indicates the effective motor rotation direction. See *setmotiondir* on information about how the direction is defined.

Syntax:

[axisno] *getmotiondir*

entry	range
[axisno]	1...16

Reply:

[motiondir]

[motiondir]	description
0	Moving upward by nominal position has the effect of clockwise axis rotation (factory setting)
1	Moving upward by nominal position has the effect of anti clockwise axis rotation

Example:

1 getmotiondir

setnpowerup Pollux | Pollux NT

Description:

The command *setnpowerup* initialises the controller to execute fixed commands automatically after power up.

Each power up command is decoded from a binary state (bit D0-D3). The respective command will be executed if the corresponding bit is set. To combine several power up commands to a command sequence, their binary states must be added.

Syntax:

[powerup] [axisno] *setnpowerup*

entry	meaningful combinations
[powerup]	0, 2, 6, 8, 10, 14
[axisno]	1...16

[powerup] bit (dec)	command	description
D0 (1)		reserved
D1 (2)	*ncal*	The axis is moving to the cal limit switch
D2 (4)	*nrm*	The axis is moving to the rm limit switch
D3 (8)	*nrandmove*	A random move is performed With limit switches installed, it is recommended to have the controller execute ncal and nrm first. ([powerup] = 14)

Note that the controller will not perform an initial *nrm* without an *ncal* prior to it. So the settings 4 and 12 are invalid and have no effect on power up.

14 1 setnpowerup
After power up, the controller first determines the limits, then the axis is moved to randomized coordinates in an infinite move sequence.

getnpowerup

Description:

The command *getnpowerup* returns the powerup command settings of the controller.

Syntax:

[axisno] *getnpowerup*

entry	range
[axisno]	1.....16

Reply:

[powerup]

Example:

8 getnpowerup

Reply: 14

Velocity and acceleration

setnvel (snv) Pollux | Pollux NT

Description:

The command *setnvel* defines the max. velocity with which the specified axis moves the actuator (not the motor).
The motor rotation speed is determined from the *setnvel* value and the *setpitch* value.

Syntax:

[velocity] [axisno] *setnvel*

entry	range	unit
[velocity]	0.0001...2000.0 *	mm/s
[axisno]	1...16	-

* maximum is overall; also depends on the actual Pollux rotary speed range and machine geometry (pitch, polepairs)

Pollux speed grades:

speed grade *	rotational speed range
1	max. 13 rev/s
2	max. 25 rev/s (Option)
3	max. 40 rev/s (Option)

* the speed grade setting at the time of delivery (upgrade optional) is indicated by the "Type" entry on the controller name plate

Example:

10.0 1 snv

getnvel (gnv)

Description:

The command *getnvel (gnv)* returns the setting of *setnvel*.

Syntax:

[axisno] *getnvel*

Reply:

[velocity]

	unit
[velocity]	mm/s

Example:

2 gnv

Reply:
18.000000

setconfig   Pollux | Pollux NT

Description:

The command *setconfig* is designed for general controller matching via a configuration register, especially for the purpose of establishing compatibility to other ITK controllers.

The value is bit-coded. The sign of the user coordinate offset is the only option at the time.

Syntax:

[config] [axisno] *setconfig*

entry	range
[config]	0, 1
[axisno]	1...16

[config] bit (dec)	description	available settings
D0 (1)	sign of coordinate origin offset*	0: positive 1: negative

*If

    [value] [axisno] setnpos

is entered, the effected controller position will be

    - [value] if the offset sign setting is positive (0)
    + [value] if the offset sign setting is negative (1)

The alteration of bit D0 in the configuration register does not automatically realign the current coordinate. The configuration setting is exclusively regarded every time *setnpos* is applied (see example below).

Example:

1 1 setconfig
30. 1 setnpos
1 np -> 30.000000
0 1 setconfig
1 np -> 30.000000
30. 1 setnpos
1 np -> -30.000000

getconfig | Pollux | Pollux NT

Description:

Returns the content of the controller configuration register.

Syntax:

[axisno] *getconfig*

entry	range
[axisno]	1.....16

Reply:

[config]

Related command:

setconfig

Example:

8 getconfig

Reply: 1

setnaccel (sna)

Pollux | Pollux NT

Description:

The command *setnaccel (sna)* defines the acceleration ramp at which the controller executes the programmed move.

Acceleration and deceleration ramp are identical.

Syntax:

[acceleration] [axisno] *setnaccel*

entry	range	unit
[acceleration]	1.0...2000.0	mm/s²
[axisno]	1....16	-

Related commands:

getnaccel

Example:

500.0 1 sna

getnaccel (gna)

Pollux	Pollux NT

Description:

The command *getnaccel (gna)* returns the setting of the acceleration ramp.

Syntax:

[axisno] *getnaccel*

Reply:

[acceleration]

	range	unit
[acceleration]	1.0...2000.0	mm/s²

Example:

1 gna

Reply:
200.000

setnstopdecel

<div align="right">Pollux | Pollux NT</div>

Description:

The command *setnstopdecel* defines the deceleration ramp in case of an emergency stop. A stop can be caused by the limit-switches or by the command breaks *Ctrl-B*, *Ctrl-C* or *nabort*.

Syntax:

[deceleration] [axisno] *setnstopdecel*

entry	description
deceleration	steepness of stop deceleration ramp

entry	range	unit
value	500.0...2000.0	mm/s²
axisno	1..16	-

Related commands:

getnstopdecel, gna, sna

Example:

1000.0 3 setnstopdecel

In case of an emergency stop or motion abort, the motor will stop at a deceleration ramp of 1 m/s².

getnstopdecel | Pollux | Pollux NT

Description:

The command *getnstopdecel* returns the setting of the emergency stop deceleration ramp.

Syntax:

[axisno] *getnstopdecel*

Related commands:

setnstopdecel, gna

Reply:

[deceleration]

	range	unit
[deceleration]	500.0...2000.0	mm/s²

Example:

4 getnstopdecel

setncalvel

Description:

The command *setncalvel* defines both velocities for the *ncal* limit-switch move.

1. Velocity of move in negative direction / towards the switch
2. Velocity of move in positive direction / away from the switch

Syntax:

[velocity] [index] [axisno] *setncalvel*

entry	range	unit
[velocity]	0.0001...2000.0	mm/s
[index]	1, 2	-
[axisno]	1...16	-

[index]	description
1	velocity towards the limit-switch
2	velocity out of the limit-switch

Related commands:

getncalvel, setnrmvel

Example:

2.0 1 2 setncalvel
0.1 2 2 setncalvel

getncalvel

Description:

The command *getncalvel* returns the *ncal* limit-switch move velocities. Unlike with the setting command *setncalvel*, the particular velocities do not have to be selected by an index; they are returned in one line.

Syntax:

[axisno] *getncalvel*

Reply:

[velocity-1] [velocity-2]

	description
[velocity-1]	velocity towards switch
[velocity-2]	velocity out of switch

	range	unit
[velocity-1]	0.0001...2000.0	mm/s
[velocity-2]	0.0001...2000.0	mm/s

Example:

2 getncalvel

Reply:

2.000000 0.250000

setnrmvel

Description:

The command *setnrmvel* defines both velocities for the nrm limit-switch move.

1. Velocity of move in positive direction / towards the switch
2. Velocity of move in negative direction / away from the switch

Syntax:

[velocity] [index] [axisno] *setcalvel*

entry	range	unit
[velocity]	0.0001 - 2000.0	mm/s
[index]	1, 2	-
[axisno]	1...16	-

[index]	description
1	velocity towards the limit-switch
2	velocity out of the limit-switch

Related commands:

getnrmvel, setncalvel

Example:

2.0 1 1 setnrmvel
5.0 2 1 setnrmvel

getnrmvel

Description:

Command *getnrmvel* returns the two adjusted *nrm* move velocities. Unlike with the setting command *setnrmvel*, the particular velocities do not have to be selected by an index; they are returned in one line.

Syntax:

[axisno] *getnrmvel*

Reply:

[velocity-1] [velocity-2]

	description
[velocity-1]	velocity towards switch
[velocity-2]	velocity out of switch

	range	unit
[velocity-1]	0.0001...2000.0	mm/s
[velocity-2]	0.0001...2000.0	mm/s

Example:

3 getnrmvel

Reply:

2.000000 0.250000

setnrefvel

Description:

The command *setnrefvel* configurates both velocities at which the reference move is exectued.

1. velocity of first move by which the reference mark is found
2. velocity of second move by which the slide is exactly driven back to the reference mark position.

The reference move is started with the command **nrefmove**.

Syntax:

[velocity] [index] [axisno] *setnrefvel*

entry	range	unit
[velocity]	0.0001 - 2000.0	mm/s
[index]	1, 2	-
[axisno]	1...16	-

[index]	description
1	velocity to find the mark (fast)
2	velocity for the final move to the mark

Related commands:

getnrefvel

Example:

20.0 1 1 setnrefvel
0.1 2 1 setnrefvel

getnrefvel

Pollux NT

Description:

The command *getnrefvel* returns the reference move velocities. Unlike with the setting command *setnrefvel*, the particular velocities do not have to be selected by an index; they are returned in one line.

Syntax:

[axisno] *getnrefvel*

Related commands:

setnrefvel

Reply:

[velocity-1] [velocity-2]

	description
[velocity-1]	velocity to find the mark (fast)
[velocity-2]	velocity for the final move to the mark

	range	unit
[velocity-1]	0.0001...2000.0	mm/s
[velocity-2]	0.0001...2000.0	mm/s

Example:

2 getnrefvel

Reply:

10.000000 0.500000

Positioning commands

nmove (nm) Pollux | Pollux NT

Description:

The command *nmove* executes point to point positioning tasks to absolute coordinates based on the point of origin.

The command *nstatus* returns the current state of the move procedure.

Synchronous move:

Instead of an individual axis index a so-called axes mask can be used to start several axes at the same time to different coordinates. See chapter "Introduction to Venus-2".

Ctrl-B, *Ctrl-C* or *nabort* terminate the actual move.

Syntax:

[coordinate] [axisno] *nmove*

entry	range	unit
[coordinate]	+/- 1000.0	mm

Related commands:

nrmove, speed

Examples:

12.5 1 nm

nrmove (nr)

Description:

The command *nrmove* executes point to point positioning tasks relative to the current position.

The command *nstatus* returns the actual state of the move procedure.

Ctrl-B, Ctrl-C or *nabort* terminate the actual move.

Syntax:

[distance] [axisno] *nrmove*

entry	range	unit
[distance]	+/- 2000.0	mm

Related commands:

nmove, speed

Example:

0.5 1 nr

speed

Description:

The command *speed* starts a constant velocity move.

The move profile is calculated in respect to the velocity/ acceleration setup and the given motion range limits. The motion direction is defined by the sign of the [velocity].

Speed and direction can be changed on the fly by reapplying *speed* with a different [velocity] value.

The commands *stopspeed, Ctrl-B, Ctrl-C* or *nabort* terminate the actual move.

The speed mode is indicated in the *nstatus* reply (see command *nstatus*).

Syntax:

[velocity] [axisno] *speed*

entry	range	unit
[velocity]	+/- 2000.0	mm/s
[axisno]	1....16	-

* depends on the model and the released speed grade.

Example:

10.0 1 speed

stopspeed

Pollux | Pollux NT

Description:

The command *stopspeed* terminates the constant velocity move of the target axis with the current acceleration setting (see command *sna*).

Syntax:

[axisno] *stopspeed*

Example:

1 stopspeed

nrandmove

Description

The command *nrandmove* moves the axis to randomized coordinates with an also randomized velocity setup. Move distances and velocities are linked in a way that a single move will not exceed a time duration of about 8 seconds (somewhat longer, depending on the current acceleration setting because acceleration/deceleration ramps are not regarded in duration calculation).

Ctrl-B, Ctrl-C and *nabort* terminate the procedure.

In a configuration with limit switches it is recommended to execute *ncal* and *nrm* in advance. Otherwise the procedure is likely to be terminated inadvertently by one of the limit switches.

Syntax:

[axisno] *nrandmove*

entry	description	range
[axisno]	axis number	1..16

Example:

14 nrandmove

Limit Switch functions

ncalibrate (ncal)

Description:

The command *ncal* executes a limit-switch move to the cal limit-switch. This procedure determines the origin and lower limit of the selected axis.

The origin can be changed using *setnpos*.
The lower limit can be changed using *setnlimit*.

The velocity of the *ncal* movement is adjusted using *setncalvel*.

With *Ctrl-B* *or* *Ctrl-C* the *ncal* limit-switch move is immediately aborted and the origin and lower motion range limit are set at the current position.

Syntax:

[axisno] *ncal*

entry	description	range
[axisno]	axis number	1..16

Example:

1 ncal

nrangemeasure (nrm) Pollux | Pollux NT

Description:

The command *nrm* executes a limit-switch movement to the rm limit-switch. This procedure determines the upper limit of the selected axis.

The velocity of *nrm* movement is adjusted using the command *setnrmvel*.

With *Ctrl-C* or *Ctrl-B* the *nrm* limit-switch move is immediately aborted and the origin and lower limit are set at the current position.

Syntax:

[axisno] *nrm*

entry	description	range
[axisno]	axis number	1..16

Example:

1 nrm

getswst Pollux | Pollux NT

Description:

The command *getswst* returns the current activity of the limit-switch inputs (cal/rm).

Syntax:

[axisno] *getswst*

entry	description	range
[axisno]	axis number	1..16

Reply:

[cal-input]] [rm-input]

[cal-input]	description
0	cal limit-switch is in OFF state
1	cal limit-switch is in ON state

[rm-input]	description
0	rm limit-switch is in OFF state
1	rm limit-switch is in ON state

Related commands:

setsw

Example:

3 getswst
Reply: 0 0
The cal and rm limit-switches of Axis-3 are in OFF state.

setncalswdist

Description:

The command *setncalswdist* is used for defining a distance between the release point of either limit-switch and the corresponding motion range limit (calibration switch distance).
This setting reduces the working area at both ends.

With this setting, the limit-switch move procedure works as follows:
1. Move to the limit-switch until it is engaged (ON state)
2. Move out of the limit-switch until it is released (OFF state)
3. Move out an additional distance defined by *setncalswdist*

Syntax:

[distance] [axisno] *setncalswdist*

entry	range	unit
[distance]	0.0...1.0	mm
[axisno]	1....16	-

Related command:

getncalswdist

Example:

5 1 setncalswdist

getncalswdist

Description:

The command *getncalswdist* returns the calibration switch distance.

Syntax:

[axisno] *getncalswdist*

entry	range
[axisno]	1...16

Reply:

[distance]

	unit
[distance]	mm

Example:

1 getncalswdist
Reply:

2.000000 0.250000

---

setnlimit

Description:

With the command *setnlimit*, the software limits are defined.

Once the limits are determined, the axis is not able to move beyond them. A running move targetting an invalid position will be stopped at the limit border. However, this does not apply to the *ncal* and *nrm* moves.
This scenario will produce error Code 1004 or 1015
(see command *gne*)

Requirements to define software limits:

• If the hard limits are specified with the cal/rm limit switch moves, the software limits must be located between these limits.

• The value of the lower limit has to be less than the value of the upper limit.

• The current position has to be located these limits, otherwise the command is not executed.

• The limit-switch moves *ncal* and *nrm* overwrite the software limits.

• With the command *reset*, all limits are lost and set to their maximum value +/-1000 mm

Syntax:

[lower limit] [upper limit] [axisno] *setnlimit*

entry	description	range
[lower limit]	lower limit	0...-1000mm
[upper limit]	upper limit	0...+1000mm

Related command:

setinilimit, setnpos

Example:

0 100.0 1 setnlimit

getnlimit Pollux | Pollux NT

Description:

The command *getnlimit* returns the limit coordinates.

If no limits are specified, the maximum limit values (+/-1000mm) are returned.

Syntax:

[axisno] *getnlimit*

Reply:

[lower limit] [upper limit]

Example:

1 getnlimit

Reply:
0.000000 70.000000

setinilimit

Description:

Used to set up an initial motion range limitation which is storable and effective as default hardware limits after powerup. This is especially useful for installations with one range limit (cal) switch only (see example below). The default range is +/- 1m.

The limits are effective:

- with the setting of the upper limit by touching the limit switch during *ncal*
- as range limits with *nm, nr, nrandmove* after powerup or software reset, respectively

They are particularly *not* effective as:

- range limits with *setnlimit, setnpos, setblcd, setranddist*
- range limits with *speed*, *ncal*, *nrm*
- with the setting of the upper limit by touching the limit switch during *nrm*

Typical application is that the controller powers up with the initial upper limit set to the distance coverage of the machine. Execution of *ncal* will set the current position and lower limit to 0 after termination, while the upper limit persists. In this way it is possible to have the motion range limited at both ends without imperatively having to install a second limit switch.

Syntax:

[lower limit] [upper limit] [axisno] *setnlimit*

entry	description	range
[lower limit]	lower limit	0...-1000mm
[upper limit]	upper limit	0...+1000mm

ncal, *nrm*, *setnlimit*, *getinilimit*

Limit configuration of a 100 mm motion range machine.

0 100. 1 setinilimit
1 nsave
1 nreset
1 getnlimit -> 0.000000 100.000000
1 ncal 1 gne -> 0
1 np -> 0.000000
1 getnlimit -> 0.000000 100.000000

The slide is now located at the lower limit switch and the motor will not be driven further than the machine permits the slide to move.

getinilimit

Description:

The command *getinilimit* returns the initial motion range limit coordinates.

If no limit is specified, the maximum limit value (+/-1000mm) is returned.

Syntax:

[axisno] *getinilimit*

entry	description	range
[lower limit]	lower limit	0...-1000mm
[upper limit]	upper limit	0...+1000mm

Reply:

[lower limit] [upper limit]

Example:

1 getnlimit

Reply:
0.000000 70.000000

Safety functions

Ctrl-C

Description:

The key shortcut *Ctrl-C* immediately terminates the currently running move.

The move is stopped immediately with the acceleration setup defined with the command *setnstopdecel*.

Pending commands from the RS-232 input buffer will be executed subsequently.

Ctrl-C is a broadcast command; this means all axes in a Pollux-1 network will be stopped.

With the command *setemergency* each axis can be parameterized to ignore the **Ctrl-C** shortcut (valid only for Pollux-2).

If an **ncal** or **nrm** limit-switch move is interrupted by **Ctrl-C**, the current axis position is taken over as the controller lower limit or maximum range limit.

The *Ctrl-C* command does not have to pass the command FIFO, therefore it can not be delayed with a blocking command.

Syntax:

ASCII sign	decimal value	hex value
Ctrl-C	3	0x3

Related command:

abort, Ctrl-B

Example:

Ctrl-C

Ctrl-B

Description:

With the command *Ctrl-B,* the running move at each axis in a Pollux network is immediately terminated (same as with *Ctrl-C*). Moreover, the position control loop is opened and the axis set to move-blocking state.

The communication with the controller is further on possible. Full operation can be restored by executing *setaxis*.

With the command *setemergency* each axis can be parametrized to ignore the *Ctrl-B* command.

Syntax:

ASCII sign	decimal value	hex value
Ctrl-B	2	0x2

Related commands:

abort, *Ctrl-C*, *setemergency, setaxis*

Example:

Ctrl-B
1 1 setaxis
1 2 setaxis

setemergency Pollux NT

Description:

With the command *setemergency* the axis can be released or blocked for the emergency stop shortcuts *Ctrl-C* and *Ctrl-B*.

Syntax:

[config] [axisno] *setemergency*

[config]	Ctrl-C	Ctrl-B
0	blocked	blocked
1	released	blocked
2	blocked	released
3	released	released

entry	range
[config]	0, 1, 2, 3
[axisno]	1...16

Related commands:

getemergency

Example:

1 1 setemergency

getemergency Pollux NT

Description:

The command *getemergency* returns the configuration of the emergency shortcuts.

Syntax:

[axisno] *getemergency*

entry	range
[axisno]	1..16

Reply:

[config]

	range
[config]	0, 1, 2, 3

Related commands:

setemergency

Example:

1 getemergency

Reply:
3

nabort

Description:

The command *nabort* terminates the current running command of the axis at the moment of execution. A move will be stopped immediately with the acceleration setup defined by the command *setnstopdecel.*

The commands in the command FIFO are not cleared.

With *nabort* having to pass the data input FIFO, it will be delayed if preceeded by a move and a blocking command.

For example: *nabort* is blocked with the command *gne*

1. *100 1 nm (currently executed)*
2. *1 gne*
3. *1 nabort*

Syntax:

[axisno] *nabort*

entry	description	range
[axisno]	axis number	1..16

Related command:

Ctrl-C

Example:

2 nabort

Position / origin / coordinate system

npos (np)

Description:

The command *npos* returns the current axis coordinate.
The position value relates to the origin which is defined by
the commands *ncal* or *setnpos*.

Syntax:

[axisno] *npos* or *np*

entry	description	range
[axisno]	axis number	1..16

Reply:

[pos]

	description	unit
[pos] *	axis coordinate	mm

*returned with 6 decimal places

Related commands:

setselpos

.

Example:

3 npos

Example of a reply: *12.041959*

---

setnpos   Pollux | Pollux NT

Description:

With the command *setnpos* the position origin of the axis can be defined. The origin position is entered as a distance, relative to the current axis location. If the origin is shifted, the coordinates of the limits will be recalculated accordingly.

Syntax:

[coordinate] [axisno] *setnpos*

entry	range	unit
[coordinate]	+/-1000 mm	mm

Example:

0 1 setnpos

The current coordinate is defined as the new origin. The nominal position will be 0 afterwards.

-30 1 setnpos

The position -30 mm relative to the current coordinate is defined as the new origin. The nominal position will be 30 mm afterwards.

setselpos   Pollux NT

Description:

The command *setselpos* specifies if the internal calculated positioning data (nominal value) or the measured value of an external measuring system (actual value) are delivered upon application of *npos*.

Syntax:

[source][axisno] *setsw*

entry	range
[source]	0, 1
[axisno]	1.... 16

[source]	description
0	nominal position
1	actual position

Related commands:

getselpos, *npos*

Examples:

0 3 setselpos
1 1 setselpos

The 3rd axis delivers the desired position.
The 1st axis delivers the measured position.

getselpos

Description:

The command *getselpos* delivers the position source selected by *setselpos*..

Syntax:

[axisno] *getselpos*

entry	range
[axisno]	1.... 16

Reply:

[source]

[source]	description
0	nominal position
1	actual position

Example:

3 getselpos

Current status inquiries

nstatus (nst)

<div style="text-align: right">Pollux | Pollux NT</div>

Description:

The command *nstatus* returns the current state of the axis.

Each state is assigned to a binary digit from D0 to D8.
If several states are active, the decimal values of the digits are added.
To decode the replied status, it is necessary to convert the decimal value into a binary pattern and mask the bits.

Binary	Decimal	Function
D0	1	Move in progress / finished
D1	2	no function
D2	4	Machine error occurred *
D3	8	no function
D4	16	Speed mode on/off
D5	32	Closed Loop In-Window function *
D6	64	Motor driver state**
D7	128	Motion enable state**

* Pollux NT only   ** hardware option

D0: Motion state

0	Move finished
1	Move in progress

D1:

0	no function
1	no function

D2: Machine error state

0	No machine error
1	Machine error occurred

D3:

0	no function
1	no function

D4: Speed mode state

0	Speed mode active.
1	Speed mode not active

D5: Closed Loop In-Window state

0	Current position out of the target window.
1	Current position within the target window.

D6: Motor driver state

0	Motor driver enabled from hardware input
1	Motor driver disabled from hardware input

D7: Motion disable state

0	Motion enabled
1	Motor driver disable event occurred - motion disabled - reset required to restore full operation

Syntax:

[axisno] *nstatus* or *nst*

Reply:

[status]

Example:

1 nstatus
Reply:
1
Bit pattern: 10000000

Decoded state: Move in progress

Decoding examples:

motor power off / motion disabled

motor driver disable input

in-window

speed mode

machine error occurred

moving

bit	0	1	2	3	4	5	6	7
bit value	1	0*	4	0*	16	32	64	128
[status] = 0	0	0	0	0	0	0	0	0
[status] = 1	1	0	0	0	0	0	0	0
[status] = 32	0	0	0	0	0	1	0	0
[status] = 192	0	0	0	0	0	0	1	1
[status] = 128	0	0	0	0	0	0	0	1

*reserved

Error messages

getnerror (gne)   Pollux | Pollux NT

With the command *getnerror* the last occurred system error is returned. Afterwards the error code memory is cleared.
The occurrence of a system error is not reflected in the status reply.

Syntax:

[axisno] *getnerror*

entry	description	range
[axisno]	axis number	1..16

Reply:

[error code]

Valid error codes:

Error code	Description
1002	Parameter stack underrun
1003	Parameter out of range
1004	Position range exceeded *
1009	Para stack lacking space (< 10 para. left)
1010	RS-232 input buffer lacking space (< 30 char. left)
1015	Limit setting inconsistent *
1100	Limits switches states inconsistent / both active
2000	Unknown command

*If a move is terminated by either limit switch in spite of valid motion range limits, this will be indicated by code **1004**. (This could, for instance, occur subsequent to loss of motor steps in open loop operation).

If a manual alteration of the motion range by *setnlimit* would as a consequence invalidate the current axis coordinate, the controller will discard the setting, leave the limits unchanged and set code **1015**.

When a move is meant to target a position outside the currently valid motion range, the axis controller will target the respective position limit instead. Pollux-1 indicates this by code **1004**, Pollux-2 by code **1015** for compliance with other ITK controllers.

Example:

1 gne

getmerror (gme) | Pollux | Pollux NT

Description:

The command *getmerror* indicates the hardware error status.

Pollux-1 will always return the code of the last error that occurred. With Pollux-2, pending machine errors are put on a memory FIFO which can store a maximum of 10 error codes, so repetitive execution of *gme* will return the error codes in the order of their appearance.
The occurrence of a machine error is indicated in the status reply (Bit 2).

As long as no error indication is pending, *gme* replies 0.

Syntax:

[axisno] *getmerror* or *gme*

entry	description	range
[axisno]	axis number	1..16

Reply:

[machine error code]

Error codes	Description	Related Pollux
13	Position deviation out of range (following error)	Pollux-2
20	Position sensor amplitude too low	Pollux-2
22	Position sensor offset /amplitude correction failed	Pollux-2
30	Parameter stack overflow	Pollux-1 / Pollux-2

Example:

10 gme

System commands

npop

Description:

The command *npop* removes one element from the parameter stack, according to the LIFO principle.

(LIFO = Last in first out)

Syntax:

[axisno] *npop*

entry	description	range
[axisno]	axis number	1..16

Related commands:

npush

Example:

1 npop

npush

Description:

With the command *npush*, it is possible to push parameters to a selective controller in a Pollux network environment.
With each parameter the stack of the axis counts up. The state of the stack counter is replied with command *ngsp*.
To remove stack elements, use *npop*.

Syntax:

[value] [axisno] *npush*

entry	description	range	unit
[value]	absolute or relative coordinates	+/- 2000.0	mm
[axisno]	axis number	1..16	-

Related commands:

npop, *ngsp*

Example:

28.1 1 npush
12.4 2 npush
23.4 3 npush
18.4 5 npush

getserialno

Description:

The command *getserialno* returns the serial number of the axis.

Syntax:

[axisno] *getserialno*

Reply:

YY HW SERI

	description	digits
YY	Year	2
HW	Hardware Revision	2
SERIAL	Consecutive number	4

Related command:

nidentify, *nversion*

Example:

4 getserialno

Reply:
03010113
Description:
Year: 2003
Hardwarerevision 01
Number: 0113

nversion

Description:

The command *nversion* returns the firmware version of the axis.

Syntax:

[axisno] *nversion*

entry	description	range
[axisno]	axis number	1..16

Reply:

[firmware version]

Related command:

nidentify

Example:

1 nversion

Reply:

2 1 4

getnoptions

Pollux | Pollux NT

Description:

The command *getnoptions* returns a decimal number that indicates the enabled options.

Each option is assigned to a binary digit from D0 to D9.
If several options are enabled, the decimal values of the digits are added.
To get each enabled option separately, it is necessary to convert the decimal value into a binary value and mask the bit pattern.

Syntax:

[axisno] *getnoptions*

Reply:

[value]

	range
[value]	0 - 15

Bit	Dec.	Released option	Pollux Type
D0	1	Speed grade 2 = 25 rev./s	Pollux-1 / Pollux-2
D1	2	Speed grade 3 = 40 rev./s	Pollux-1 / Pollux-2
D2	4	Closed Loop	Pollux-2
D3	8	CAN Bus	Pollux-2

Example:

1 getnoptions

Reply: 6

Speed grade 3 and Closed Loop is released

nidentify

Description:

The command *nidentify* returns the controller hardware and software revision.

Syntax:

[axisno] *nidentify*

Reply:

[Model] [HW-Rev] [SW-Rev] [Board-Sw] [ID]

	description
[Model]	Pollux model (Pollux-1 / Pollux-2)
[HW-Rev]	Hardware revision
[SW-Rev]	Software revision
[ID]	Internal use
[ID]	Internal use

Related command:

nversion

Example:

1 nidentify
Reply:
Pollux2 1 214 1 0

---

getnfpara

Pollux | Pollux NT

Description:

The command *getnfpara* activates the factory configuration.

Attention:
The currently stored parameter set is being overwritten but can be restored with the command *restore*.

Syntax:

[axisno] *getnfpara*

Reply:

Example:

2 getnfpara

nsave

Description:

The command ***nsave*** stores a great deal of the currently valid controller parameter set in a non-volatile memory. The stored parameter set is automatically restored during power up.

These non-volatile parameters are declared with the following symbol:

Syntax:

[axisno] ***nsave***

entry	description	range
[axisno]	axis number	1..16

Related commands:

nrestore

Example:

3 nsave

nclear

Pollux | Pollux NT

Description:

The command *nclear* deletes the content of the parameter stack.

The related command **ngsp** displays the current number of parameters on the stack.

In an accurate operation the number of parameters on the stack will always go to zero when there are no commands pending.

A permanent remainder of elements on the stack indicates an inaccurate use of the Venus-2 language (i.e. to many parameters, wrong syntax). In a worst case the stack will overflow if more than 99 elements are put on the stack and cause a malfunction of the controller. However, Pollux features a stack overflow event handling to minimize this risk. For more information, see chapter "Introduction to Venus-2", section "Interpreter/Scanner/Parameter stack".

Syntax:

[axisno] *nclear*

entry	description	range
[axisno]	axis number	1..16

Related command:

ngsp

Example:

2 nclear

nreset

Description:

The command *nreset* performs a device reset, which is - regarding firmware - equal to temporarily disconnecting the device from the power.
The proper state of the controller after a reset is indicated with LED (1s heart beat).

Syntax:

[axisno] *nreset*

entry	description	range
[axisno]	axis number	1..16

Example:

4 nreset

ngsp

Description:

The command *ngsp* returns the number of elements on the parameter stack.

Information about the stack.
Pollux has a parameter stack (LIFO structure) which holds the received parameters - seperate from the received command IDs. The stack can contain a maximum of 99 parameters.
On data reception, parameters are put (pushed) on the stack. On each Venus-2 command execution, data are - in inversed order - removed (popped) from the stack as needed. So basically there is no direct line-based assignment from commands to their data. The assignment is determined by the temporal order of command and parameter transmissions.

Following command is executed: *10.123 1 nm*
1 ngsp -> 0 // no elements on the stack
10.123 _ 1 _ // parameters are sent to the controller
1 ngsp -> 2 // 2 elements have been pushed on the stack
nm   // Venus-2 command is sent to the controller
1 ngsp -> 0 // all elements on the stack are removed.

It is theoretically possible to push up to 98 parameters/ coordinates and corresponding axis numbers on the stack in a row. These coordinates can, for instance, in inverse order be targetted by a matching number of subsequent move commands.

Example:
10.0 1
12.0 1
40.0 1
nm // move to 40.0 mm
nm // move to 12.0 mm
nm // move to 10.0 mm

Syntax:

[axisno] *ngsp*

entry	description	range
[axisno]	axis number	1..16

Reply:

[number]

	range
[number]	0 -99

Related command:

nclear

Example:

1 ngsp

Reply:
2

nrestore

Description:

The command *nrestore* reactivates the parameter set valid at the moment of the last execution of *nsave*.

Syntax:

[axisno] *nrestore*

entry	description	range
[axisno]	axis number	1..16

Reply:

A reply can be enabled with the following command sequence
nrestore
nstatus

Related command:

getnfpara

Example:

1 nrestore

Special functions

setuv

Description:

The command *setuv* allows to store a maximum of 10 user specific parameters in the flash memory.
The controller provides 10 memory registers. Each register is able to store one 32 Bit integer value.
The parameters are stored non-volatile with the command *nsave*.

With *nrestore* the last saved parameters are restored.

Syntax:

[user number] [register] [axisno] *setuv*

entry	range
[user number]	-2147483648 ...+2147483647
[register]	0....9
[axisno]	1...16

Example:

12041959 2 1 setuv

getuv

| | Pollux | Pollux NT |

Description:

The command *getuv* returns the specified user variable.

Syntax:

[register] [axisno] *getuv*

entry	range
[register]	0....9
[axisno]	1...16

Reply:

[user number]

Example:

2 1 getuv

Reply
12041959

setranddist

Description:

By utilization of the *setranddist* command, the axis can be induced to alternate ("shake") between two fixed positions on execution of *nrandmove*. One of the alternating positions is located where the move starts. The parameter to deliver with *setranddist* is the distance from the starting position to the alternative position. All other *nrandmove* application conditions are unchanged. Setting the distance to 0 disables the "shake" function.

Syntax:

[distance] [axisno] *setranddist*

entry	range	unit
[distance]	-2000.0...2000.0	mm
[axisno]	1....16	-

Related commands:

nrandmove

Example:

1 np -> 200.000000
25. 1 setranddist
1 nrandmove

The axis will alternate between position coordinates 200.0 and 225.0.

getranddist

Description:

The command *getranddist* returns the setting of the "shake" distance.

Syntax:

[axisno] *getranddist*

entry	range	unit
[axisno]	1....16	-

Reply:

[distance]

	range	unit
[distance]	-2000.0...2000.0	mm

Example:

1 getranddist

Reply:
100.000000

Closed loop function

setcloop Pollux NT

Description:

The command *setcloop* is used to determine whether the the controller operates in Open Loop or Closed Loop mode. In Closed Loop mode, the position control loop is active. The closed loop function requires a properly configured and working external analog measurement system of a matching type.

For running the controller in Closed Loop mode, the corresponding option has to be enabled via activation code.

For proper Closed Loop configuration see also the following commands:
setsp, setclperiod, setclpara, setnselpos, setscaleinterface, getnoptions

Syntax:

[cloop] [axisno] *setcloop*

entry	range
[cloop]	0, 1
[axisno]	1..16

[cloop]	description
0	Closed Loop disabled (Open Loop mode)
1	Closed Loop enabled

Related command:

getcloop

Example:

1 2 setcloop

getcloop   Pollux NT

Description:

The command *getcloop* returns the Closed Loop setting of the controller.

Syntax

[axisno] *getcloop*

entry	range
[axisno]	1...16

Reply:

[cloop]

	range
[cloop]	0, 1

The reply indicates the user setting, not the actual loop state. It will, for instance, not be altered by an error condition that causes the Pollux controller to force the position control loop open autonomously, like an amplitude or following error.

Example:

1 getcloop

Reply:
1

setclwindow Pollux NT

Description:

The command ***setclwindow*** defines a symmetrical window (target window) centered at the current nominal position for the closed loop function. The [window] parameter defines half the width of the window.

The window is designed to indicate the settling of the slide or rotor after a move or other displacement. If once the measured position enters the window and then steadily remains therein for a defined time (see ***setclwintime***), the "in-window" status is on, reflected by status register bit D5 (see ***nstatus***) going active. If once the measured position leaves the window (or the window is displaced by a move), the "in-window" status is off, indicated by D5 being reset.

Syntax:

[window] [axisno] ***setclwindow***

entry	range	unit	description
[window]	<= 1.0	mm *	target window of +/- [window] mm around the target position active
[axisno]	1...16	-	axis number

* resolution 1 nm

Related command:

getclwindow, setclwintime

Example:

0.001 1 setclwindow

The target window is adjusted to +/-1µm, related to the desired position.

getclwindow Pollux NT

Description:

The command *getclwindow* returns the setting of the Closed Loop target window.

Syntax:

[axisno] *getclwindow*

entry	range
[axisno]	1...16

Reply:

[window]

	range	unit
[window]	<= 1.0	mm

Example:

1 getclwindow

Reply

0.001

setclwintime

Description:

The command *setclwintime* specifies the period of time during which the currently measured position must steadily be located within the target window before status register Bit D5 gets active (time on target). If set to 0, the time on target is virtually disabled, meaning that the "in-window" indication bit will go active anytime the actual position is within the defined window. See *setclwindow* for further information on the target window feature.

Syntax:

[time] [axisno] *setclwintime*

entry	range	unit	description
[time]	0...8191	ms	time on target
[axisno]	1...16	-	axis number

Related command:

getclwintime, setclwindow

Example:

20 1 setclwintime

getclwintime Pollux NT

Description:

The command *getclwintime* returns the setting of the time on target.

Syntax:

[axisno] *getclwintime*

entry	range
[axisno]	1...16

Reply:

[time]

	range	unit
[time]	0...8191	ms

Example:

1 getclwintime

Reply

20

nrefmove Pollux NT

Description:

The command *nrefmove* moves the axis to the reference mark of the measurement system at the velocity defined by *setnrefvel*. If no reference mark is found within the specified [distance], the move terminates. The sign of [distance] determines the direction the axis moves in. The current reference status is returned by the command *getrefst*.
With *Ctrl-C* or *Ctrl-B* the reference move will be aborted.

Syntax:

[distance] [axisno] *nrefmove*

entry	range	unit
[distance]	+/-1000	mm

Related commands:

setnrefvel, getrefst

Examples:

100.0 1 nrefmove

The controller moves axis 1 in positive direction. If a reference mark is found, the controller stops at the defined deceleration (command *sna*) and moves the axis back to the mark. If no reference mark is found, the axis will move exactly 100 mm before it stops.

-7.0 2 nrefmove
With this command, axis 2 moves in negative direction by 7 mm to find the reference mark.

getrefst Pollux NT

Description:

The command *getrefst* returns the current detection state of the measurement system reference mark. The status is 1 if the rotor or slide currently covers the reference mark, otherwise 0.

Syntax:

[axisno] *getrefst*

entry	range
[axisno]	1...16

Reply:

[status]

[status]	description
0	Reference mark not active
1	Reference mark active

Example:

2 getrefst

setref

Description:

The command *setref* defines on which transitions of the reference signal (from an external measurement system) the axis shall react.

Syntax:

[config] [axisno] *setref*

[config]	description
0	detect positive signal transition
1	detect negative signal transition

Related commands:

getref
getrefst

Example:

0 1 setref

The reference input of the 1st axis reacts on the positive signal transition of the reference signal.

getref

Description:

The command *getref* delivers the configurations of the command *setref*.

Syntax:

[axisno] *getref*

Related commands:

setref
getrefst

Reply:

[config]

[config]	description
0	detect positive signal transition
1	detect negative signal transition

Example:

1 getref

setclperiod

Description:

The command *setclperiod* adapts the Closed Loop interface to the period length and count direction of the connected position encoder.

The value of [period] is the move distance effecting one signal period of the encoder.

The transmission ratio caused from the spindle or gear has to be considered to calculate the *setclperiod* value (see examples).
With the algebraic sign -/+ it is possible to adapt the count direction to the motor direction.

Note that the clperiod setting is only effective with 1 Vpp type encoders.

Syntax:

[period] [axisno] *setclperiod*

entry	description
[period]	period length and counting direction

entry	range	unit
[period]	-1.0...1.0	mm
[axisno]	1...16	-

Related command:

getclperiod, setscaleinterface

- 0.002 3 setclperiod

Example: Linear stage with linear encoder

Specifications:
Linear stage with following specifications:
Encoder with signal period = 20 µm, Spindle pitch = 10 mm
The count direction of the encoder and the axis' effective positioning direction are assumed to be diametrical.

The value of [distance] is the resulting move distance within a one signal period of the encoder.The count direction of the encoder and the axis' effective positioning direction are assumed to be diametrical. Therefore

[distance] = *- 0.020mm*

The value of *distance* does **not** depend on the pitch because the resulting move distance is the same as the signal period of the encoder.

Example: Linear stage with rotary encoder

Specifications:
Rotary encoder with 1 Vpp output signal,
1000 signal periods / rev., mounted on the motor shaft.
Spindle pitch = 10 mm

In this combination the movement distance within one signal period depends on the pitch.
The value for setclperiod will be calculated as follows:

[distance] = 10 mm / 1000 periods = 0.01

getclperiod

Description:

The command *getclperiod* returns the setting of *setclperiod*.

Syntax:

[axisno] *getclperiod*

entry	range
[axisno]	1...16

Reply:

[period]

	range	unit
[period]	-1.0...1.0	mm

Example:

1 getclperiod

Reply:

0.020

setscaleinterface

Description:

The command *setscaleinterface* adapts the Closed Loop interface to the type of the connected position encoder.

Syntax:

[type] [axisno] *setscaleinterface*

[type]	description
0	MR type, 0.5 mm period length
1	1 Vpp type, period length adaptable (< 1 mm)
2	MR type, 1.0 mm period length

Related command:

setclperiod, getscaleinterface

Examples:

1 2 setscaleinterface

Adapts the controller to a 1 Vpp position measurement signal. The length of the signal period can be adapted using *setclperiod*.

getscaleinterface

Description:

The command *getscaleinterface* returns the setting of the position measurement system type.

Syntax:

[axisno] *getscaleinterface*

entry	range
[axisno]	1...16

Reply:

[type]

[type]	description
0	MR type, 0.5 mm period length
1	1 Vpp type, period length adaptable (< 1 mm)
2	MR type, 1.0 mm period length

Example:

1 getscaleinterface

Reply:

1

setsp

Description:

With the command *setsp,* the Closed Loop parameters are adjusted. The position controller is a pure integral controller.

Note for parameterizing the Pollux-2 controller:
(A correct Closed Loop setup assumed)

1. Check if the positioning works in open loop mode.
2. Activate Closed Loop using *setcloop*.
3. Check if the default integral gain (index = 2) setting yields good positioning results.
4. If not, adjust the integral gain to eliminate the position error. If the axis is self-oscillating, the setting must be lowered. If the axis takes too long to settle after displacement, the setting should be raised. The optimum will be a compromise of control speed and position overshoot.

s_{errMax}

With the absolute value of the s_{errMax} (maximum position error) parameter, the max. aberration between the actual position (supplied by the measurement system) and the nominal (calculated) position is determined.

If this limit is exceeded, the nominal position is forced to the actual position. As a consequence, the motor stops moving at once. The position controller remains active.

The sign of s_{errMax} defines if the controller - in addition to the handling above - switches off the motor current (switched off if negative).

There are no further moves possible until the axis is reenabled using the command *setaxis*.

[value] [index] [axisno] *setsp*

entry	description	range
[index]	selects 1 of the 10 parameters	1..10
[axisno]	axis number	1..16

[value]	[index]	description	range	unit
	1	-		-
I	2	integral gain		-
	3	-		-
	4	-		
	5	-		
	6	-		
s_errMax	7	max. position error	[-1....1]	mm
	8	-		
	9	-		
	10	-		

Examples:

1000 2 1 setp
The **-I-**Controller is parameterized

0.459000 7 1 setsp
A running move is stopped at the current position if the distance between actual and nominal position exceeds 0,459 mm.
The nominal position is readjusted to the actual position while the axis is still under PID control. No further moves are possible until the axis is enabled again.

-0.01 7 1 setsp
A running move is stopped if the distance between actual position and nominal position exceeds 0,01mm, the motor stops and the motor current is switched off. No further moves are possible until the axis is enabled again.

getsp

Description:

The command *getsp* returns the Closed Loop parameter settings.

Syntax:

[index] [axisno] *getsp*

entry	description	range
[index]	selects 1 of the 10 parameters	1..10
[axisno]	axis number	1..16

Related command:

setsp

Reply:

[value]

Example:

2 4 getsp

Description:

The command *a* is a means for checking the analog position measurement input signals. This is a basic check for error tracking if the position controller fails to work and the delivered measurement position seems corrupted for no obvious reason.

The controller does not return the raw signal levels, but offset and amplitude corrected values as a signed 16 Bit integer representation.

Both signals have to follow an approx. sinusoidal course over equidistant changes in sensor position, where each displacement by once the measurement system period length effects one complete sine cycle, and the signals follow each other with an absolute displacement of a quarter period length and roughly equal amplitudes.

If at least one of the signal levels stays around 0 while the sensor is moving, or changes erratically, the sensor is deficient or improperly connected.
If the signal amplitude is very low, the sensor is mismatched or misaligned. This may cause the controller to switch over to emergency state when in closed loop mode.
If the signal level reaches the maximum or minimum values at any position, the respective analog signal input is being overdriven, which is an indication for mismatch or misalignment, too. This will lead to periodic positioning errors in closed loop mode.

Syntax:

[axisno] *a*

[sin] [cos]

	description	range
[sin]	sine channel	-32768...32767
[cos]	cosine channel	-32768...32767

Example:

1 a

Reply:
15000 -15000

Backlash compensation

setblc

Description:

The command *setblc* enables or disables the backlash compensation.
With this function backlash errors caused by the reversal of travel direction are compensated.

Due to the fact that the backlash compensation is counterproductive when the position control is active, it is automatically disabled when the controller works in Closed Loop mode.

Syntax:

[blc] [axisno] *setblc*

[blc]	description
0	compensation disabled
1	compensation enabled

entry	range
[blc]	0, 1
[axisno]	1...16

Related command:

getblc, setblcd, setblcs

Example:

1 1 setblc

getblc   Pollux | Pollux NT

Description:

The command *getblc* returns the status of the function "backlash-compensation".

Syntax:

[axisno] *getblc*

entry	range
[axisno]	1...16

Reply:

[blc]

[blc]	description
0	compensation disabled
1	compensation enabled

The reply indicates the user setting, not the actual enable state. It will not be altered by the controller working in Closed Loop mode (see *setblc*).

Example:

1 getblc

setblcd

Description:

The command *setblcd* allows to define the distance value that is compensated by the backlash compensation function. The compensation is realized after each change of motion direction in either direction. Dependent on the compensation time setting (see *setblcs)* the compensation is realized in one jump or slowly within a specified time.
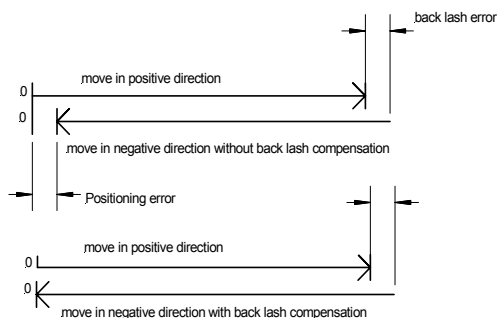
Syntax:

[blcd] [axisno] *setblcd*

entry	description
[blcd]	distance to be compensated

entry	range	unit
[blcd]	0.0...0.1	mm
[axisno]	1...16	-

Example:

0.001 1 setblcd

getblcd Pollux | Pollux NT

Description:

The command *getblcd* returns the backlash distance value.

Syntax:

[axisno] *getblcd*

entry	range
[axisno]	1...16

Reply:

[blcd]

Example:

1 getblcd

setblcs

Pollux | Pollux NT

Description:

The command ***setblcs*** defines the time (tComp) during which the compensation distance is realized in the related positioning task. The time can be set in geometrical steps of 2 as follows:

tComp [ms] = .25 * 2 ^{blcs}

For proper function, tComp must be adjusted considering the backlash distance (see ***setblcd***). The greater the distance, the greater the required tComp minimum setting:

tComp $_{min}$ [ms] = polepairs * blcd / pitch

With tComp = tComp $_{min}$, the compensation distance will be realized by one jump.

Factory settings:
blc = 0, blcd = 0.0, blcs = 7 (tComp = 32 ms)

Syntax:

[blcs] [axisno] ***setblcs***

entry	range	description
[blcs]	1....12	time exponent
[axisno]	1...16	axis number

Example:

10 1 setblcs

tComp = 2 ^{10} * 250 µs = 256 ms

getblcs

Description:

The command *getblcs* returns the currently set value of the backlash time exponent.

Syntax:

[axisno] *getblcs*

entry	range
[axisno]	1...16

Reply:

[blcs]

Example:

1 getblcs

CAN configuration

setcanbaseadr Pollux NT

Description:

The command *setcanbaseadr* sets the valid data reception address at the controller's Controller Area Network interface. The controller will exclusively receive and process data incoming at this address. For more infomation on appropriate settings, see Chapter "Principles of communication".

Syntax:

[address][axisno] *setcanbaseadr*

entry	range (dec.)
[address]	0 ... 2147483647
[axisno]	1...16

Related command:

getcanbaseadr, setcananswadr

Example:

1000 1 setcanbaseadr

---

getcanbaseadr Pollux NT

Description:

The command *getcananswadr* returns the controller's currently valid Controller Area Network reception address.

Syntax:

[axisno] *getcananswadr*

entry	range
[axisno]	1...16

Reply:

[address]

	range (dec.)
[address]	0 ... 2147483647

Example:

2 getcananswadr

Reply
400

setcananswadr Pollux NT

Description:

The command *setcananswadr* sets the valid command answer address at the controller's Controller Area Network interface. The controller will exclusively transmit command replies at this address. For more infomation on appropriate settings, see Chapter "Principles of communication".

Syntax:

[address][axisno] *setcananswadr*

entry	range (dec.)
[address]	0 ... 2147483647
[axisno]	1...16

Related command:

setcanbaseadr, getcananswadr

Example:

2000 1 setcananswadr

getcananswadr Pollux NT

Description:

The command *getcananswadr* returns the controller's currently valid Controller Area Network answer address.

Syntax:

[axisno] *getcananswadr*

entry	range
[axisno]	1...16

Reply:

[address]

	range (dec.)
[address]	0 ... 2147483647

Example:

2 getcananswadr

Reply
500

setcanbaudrate   Pollux NT

Description:
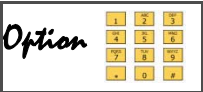
The command *setcanbaudrate* is used to set the transmission rate at the controller's Controller Area Network interface.

Syntax:

[select][axisno] *setcanbaudrate*

entry	range
[select]	0...8
[axisno]	1...16

[select]	transmission rate
0	10 kBaud
1	20 kBaud
2	50 kBaud
3	100 kBaud
4	125 kBaud
5	250 kBaud
6	500 kBaud
7	800 kBaud
8	1000 kBaud

Example:

3 1 setcanbaudrate

getcanbaudrate

Description:

The command *getcanbaudrate* returns the controller's Controller Area Network transmission rate setting.

Syntax:

[axisno] *getcanbaudrate*

entry	range
[axisno]	1...16

Reply:

[select]

[select]	transmission rate
0	10 kBaud
1	20 kBaud
2	50 kBaud
3	100 kBaud
4	125 kBaud
5	250 kBaud
6	500 kBaud
7	800 kBaud
8	1000 kBaud

Example:

2 getcanbaudrate

Reply
5

Venus-2 command list

in alphabetic order