Contents

Preface
About this documentation ................................................................. 9
Symbols in this documentation ......................................................... 10

Introduction to Venus-2
Venus-2 is an interpreter language .................................................. 12
History ............................................................................................... 12
Command syntax ............................................................................... 13
Table of important ASCII signs for programming .......................... 16
Number format .................................................................................. 17
Data input format for move and speed commands ....................... 18
Data input format for dynamics and pitch settings ...................... 19
Venus-2 Command execution ............................................................ 20

Principles of communication
Pollux / Pollux NT RS-232 communication .................................... 31
Pollux NT CAN communication ...................................................... 32
Communicaton requirements during power up ............................. 37

Hardware identification and command scope
Pollux controller types .................................................................... 39
Pollux controller identification ....................................................... 40

Basic settings
setpitch ......................................................................................... 42
getpitch ........................................................................................ 44
setumotmin .................................................................................... 45
getumotmin .................................................................................... 46
setumotgrad ................................................................................... 47
getumotgrad .................................................................................. 48
setpolepairs .................................................................................... 49
getpolepairs ................................................................................... 50
setphases ....................................................................................... 51
getphases ...................................................................................... 52
setaxis ............................................................................................ 53
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>getaxis</td>
<td>54</td>
</tr>
<tr>
<td>setaxisno</td>
<td>55</td>
</tr>
<tr>
<td>getaxisno</td>
<td>56</td>
</tr>
<tr>
<td>setsw</td>
<td>57</td>
</tr>
<tr>
<td>getsw</td>
<td>58</td>
</tr>
<tr>
<td>setmotiondir</td>
<td>59</td>
</tr>
<tr>
<td>getmotiondir</td>
<td>61</td>
</tr>
<tr>
<td>setnpowerup</td>
<td>62</td>
</tr>
<tr>
<td>getnpowerup</td>
<td>64</td>
</tr>
<tr>
<td><strong>Velocity and acceleration</strong></td>
<td></td>
</tr>
<tr>
<td>setnvel (snv)</td>
<td>66</td>
</tr>
<tr>
<td>getnvel (gnv)</td>
<td>67</td>
</tr>
<tr>
<td>setconfig</td>
<td>68</td>
</tr>
<tr>
<td>getconfig</td>
<td>70</td>
</tr>
<tr>
<td>setnaccel (sna)</td>
<td>71</td>
</tr>
<tr>
<td>getnaccel (gna)</td>
<td>72</td>
</tr>
<tr>
<td>setnstopdecel</td>
<td>73</td>
</tr>
<tr>
<td>getnstopdecel</td>
<td>74</td>
</tr>
<tr>
<td>setncalvel</td>
<td>75</td>
</tr>
<tr>
<td>getncalvel</td>
<td>76</td>
</tr>
<tr>
<td>setnrmvel</td>
<td>77</td>
</tr>
<tr>
<td>getnrmvel</td>
<td>78</td>
</tr>
<tr>
<td>setnrefvel</td>
<td>79</td>
</tr>
<tr>
<td>getnrefvel</td>
<td>80</td>
</tr>
<tr>
<td><strong>Positioning commands</strong></td>
<td></td>
</tr>
<tr>
<td>nmove (nm)</td>
<td>82</td>
</tr>
<tr>
<td>nrmove (nr)</td>
<td>83</td>
</tr>
<tr>
<td>speed</td>
<td>84</td>
</tr>
<tr>
<td>stopspeed</td>
<td>85</td>
</tr>
<tr>
<td>nrandmove</td>
<td>86</td>
</tr>
<tr>
<td><strong>Limit Switch functions</strong></td>
<td></td>
</tr>
<tr>
<td>ncalibrate (ncal)</td>
<td>88</td>
</tr>
<tr>
<td>nrangemeasure (nrm)</td>
<td>89</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------</td>
</tr>
<tr>
<td><code>getswst</code></td>
<td>90</td>
</tr>
<tr>
<td><code>setncalswdist</code></td>
<td>91</td>
</tr>
<tr>
<td><code>getncalswdist</code></td>
<td>92</td>
</tr>
<tr>
<td><code>setnlimit</code></td>
<td>93</td>
</tr>
<tr>
<td><code>getnlimit</code></td>
<td>95</td>
</tr>
<tr>
<td><code>setnlimit</code></td>
<td>96</td>
</tr>
<tr>
<td><code>getnlimit</code></td>
<td>98</td>
</tr>
<tr>
<td><strong>Safety functions</strong></td>
<td></td>
</tr>
<tr>
<td><code>Ctrl-C</code></td>
<td>100</td>
</tr>
<tr>
<td><code>Ctrl-B</code></td>
<td>101</td>
</tr>
<tr>
<td><code>setemergency</code></td>
<td>102</td>
</tr>
<tr>
<td><code>getemergency</code></td>
<td>103</td>
</tr>
<tr>
<td><code>nabort</code></td>
<td>104</td>
</tr>
<tr>
<td><strong>Position / origin / coordinate system</strong></td>
<td></td>
</tr>
<tr>
<td><code>npos (np)</code></td>
<td>106</td>
</tr>
<tr>
<td><code>setnpos</code></td>
<td>107</td>
</tr>
<tr>
<td><code>setselpos</code></td>
<td>108</td>
</tr>
<tr>
<td><code>getselpos</code></td>
<td>109</td>
</tr>
<tr>
<td><strong>Current status inquiries</strong></td>
<td></td>
</tr>
<tr>
<td><code>nstatus (nst)</code></td>
<td>111</td>
</tr>
<tr>
<td><strong>Error messages</strong></td>
<td></td>
</tr>
<tr>
<td><code>getnerror (gne)</code></td>
<td>115</td>
</tr>
<tr>
<td><code>getmerror (gme)</code></td>
<td>117</td>
</tr>
<tr>
<td><strong>System commands</strong></td>
<td></td>
</tr>
<tr>
<td><code>npop</code></td>
<td>119</td>
</tr>
<tr>
<td><code>npush</code></td>
<td>120</td>
</tr>
<tr>
<td><code>getserialno</code></td>
<td>121</td>
</tr>
<tr>
<td><code>nversion</code></td>
<td>122</td>
</tr>
<tr>
<td><code>getnoptions</code></td>
<td>123</td>
</tr>
<tr>
<td><code>nidentify</code></td>
<td>125</td>
</tr>
<tr>
<td><code>getnfpara</code></td>
<td>126</td>
</tr>
</tbody>
</table>
setblcs ........................................................................................................... 165
getblcs ........................................................................................................... 166

CAN configuration
setcanbaseadr .............................................................................................. 168
getcancanbaseadr ....................................................................................... 169
setcananswadr .............................................................................................. 170
getcancananswadr ...................................................................................... 171
setcanbaudrate ............................................................................................. 172
getcancanbaudrate ...................................................................................... 173

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.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Warning" /></td>
<td>Warning. This information must be observed strictly.</td>
</tr>
<tr>
<td><img src="image" alt="Important Information" /></td>
<td>Important information.</td>
</tr>
<tr>
<td><img src="image" alt="Option" /></td>
<td>Indicates that this function can be enabled or upgraded with a release code.</td>
</tr>
<tr>
<td><img src="image" alt="Option" /></td>
<td>This function must be installed by the factory or experts.</td>
</tr>
<tr>
<td><img src="image" alt="Venus-2" /></td>
<td>Venus-2 commands are indicated with this formatting style.</td>
</tr>
<tr>
<td><img src="image" alt="Settings" /></td>
<td>Settings that cannot be stored with command <code>nsave</code> into the flash memory.</td>
</tr>
<tr>
<td><img src="image" alt="Configuration" /></td>
<td>This configuration can be stored with command <code>nsave</code> into the controller flash memory.</td>
</tr>
</tbody>
</table>
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] _

or


_ 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 command, they have to be divided by a blank.

The following numbers and characters are permitted for parameters:

<table>
<thead>
<tr>
<th>Letters</th>
<th>not allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numbers</td>
<td>0-9</td>
</tr>
<tr>
<td>Characters *</td>
<td>+ - ,</td>
</tr>
</tbody>
</table>

*comma is not allowed as a decimal separator.
Axis number

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

Syntax of axis number : Axis mask

<table>
<thead>
<tr>
<th>numbers</th>
<th>letters</th>
<th>characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 .. 16</td>
<td>not allowed</td>
<td>not allowed</td>
</tr>
</tbody>
</table>

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

Syntax axis mask:

<table>
<thead>
<tr>
<th>Numbers</th>
<th>Letters</th>
<th>Characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1 .. - 65535</td>
<td>not allowed</td>
<td>- (minus)</td>
</tr>
</tbody>
</table>

Axis mask calculation:

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

<table>
<thead>
<tr>
<th>Axis</th>
<th>Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>-2</td>
</tr>
<tr>
<td>3</td>
<td>-4</td>
</tr>
<tr>
<td>4</td>
<td>-8</td>
</tr>
<tr>
<td>5</td>
<td>-16</td>
</tr>
<tr>
<td>16</td>
<td>-32767</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>ASCII characters</th>
<th>umlauts</th>
<th>numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>a..z; A..Z</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

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 separated 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

<table>
<thead>
<tr>
<th>ASCII Code</th>
<th>Sign</th>
<th>Dec</th>
<th>HEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR</td>
<td>Ctrl-M</td>
<td>13</td>
<td>0xD</td>
</tr>
<tr>
<td>LF</td>
<td>Ctrl-J</td>
<td>10</td>
<td>0xA</td>
</tr>
<tr>
<td>SP</td>
<td></td>
<td>32</td>
<td>0x20</td>
</tr>
<tr>
<td>ETX</td>
<td>Ctrl-C</td>
<td>3</td>
<td>0x3</td>
</tr>
<tr>
<td>STX</td>
<td>Ctrl-B</td>
<td>2</td>
<td>0x2</td>
</tr>
</tbody>
</table>
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 acceleration:

Inquiry: 1 gna -> Reply: 1000.000

According to the command description, the measurement 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 acceleration 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.

<table>
<thead>
<tr>
<th>Function</th>
<th>Venus-2 command</th>
<th>decimal point used</th>
<th>no decimal point used</th>
</tr>
</thead>
<tbody>
<tr>
<td>acceleration</td>
<td>sna</td>
<td>mm/s²</td>
<td>µm/s²</td>
</tr>
<tr>
<td>velocity</td>
<td>snv</td>
<td>mm/s</td>
<td>nm/s</td>
</tr>
<tr>
<td>pitch</td>
<td>setpitch</td>
<td>mm</td>
<td>0.1 µm*</td>
</tr>
</tbody>
</table>

*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: \textbf{10.123 1 nm}

\begin{verbatim}
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.
\end{verbatim}

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

\begin{verbatim}
10.0 1 // coordinate and axisno
12.0 1 // coordinate and axisno
40.0 1 // coordinate and axisno
80.0 2 // coordinate and axisno *
\end{verbatim}

\begin{verbatim}
nm // move to 80.0 Axis-2
nm // move to 40.0 Axis-2
nm // move to 12.0 Axis-1
nm // move to 10.0 Axis-1
\end{verbatim}
Blocking and non blocking commands

While the controller performs 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 immediately
// 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`.

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.
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:
  - \texttt{nsave}
  - \texttt{nrestore}
  - \texttt{getnpara}

All other system commands are non-blocking.

- Some parameterization commands block the interpreter for operational safety reasons, in particular:
  - \texttt{setnpos}
  - \texttt{setaxisno}
  - \texttt{setloop}
  - \texttt{setcloop}
  - \texttt{setpitch}
  - \texttt{setpolepairs}
  - \texttt{setmotiondir}
  - \texttt{setscaleinterface}
  - \texttt{setncode}
  - \texttt{setaxis}
  - \texttt{setphases}
  - \texttt{setref}
  - \texttt{setblc}
  - \texttt{setbld}
  - \texttt{setblcs}

All other parameterization commands are non-blocking.

- Inquiry commands except for \texttt{gne} do not block the interpreter.

- Emergency shortcuts (\texttt{Ctrl-B}, \texttt{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:

![Diagram]

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 `nsto st 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 significantly 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
```

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[value] absolute/ relative coordinates or speed value</td>
</tr>
<tr>
<td>[axes mask] mask to select the relevant axes*</td>
</tr>
<tr>
<td>[axisno] axis number 1...16</td>
</tr>
</tbody>
</table>
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:**

\[
\begin{align*}
40.0 & \ 3 \ sv \ n \\
200.0 & \ 3 \ sa \\
60.0 & \ 5 \ sv \\
300.0 & \ 5 \ sa
\end{align*}
\]

**Pushing the individual move distances on the stacks:**

\[
\begin{align*}
10.0 & \ 1 \ npush \\
20.0 & \ 3 \ npush \\
30.0 & \ 5 \ npush
\end{align*}
\]

**Mask calculation:**

\[
2^1 + 2^2 + 2^4 = 21
\]

**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 exception 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 coordinate 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 individually. 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 exception 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

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.

<table>
<thead>
<tr>
<th>return string portion</th>
<th>controller type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pollux</td>
<td>Pollux controller</td>
</tr>
<tr>
<td>Pollux2</td>
<td>Pollux NT controller</td>
</tr>
</tbody>
</table>
Basic settings
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 = resulting move distance / 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
```

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

**Syntax:**

```
[axisno] getpitch
```

**Reply:**

```
[pitch]
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1 ... 16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>unit</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[pitch]</td>
<td>mm</td>
</tr>
</tbody>
</table>

**Example:**

```
2 getpitch
```

Reply

```
4.009
```
The command `setumotmin` determines the motor phase voltage at standstill. It can be used to vary the holding torque. The setting also affects 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` |

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[vmin]</code></td>
<td><code>&lt; 4000 *</code></td>
<td><code>mV</code></td>
</tr>
<tr>
<td><code>[axisno]</code></td>
<td><code>1...16</code></td>
<td><code>-</code></td>
</tr>
</tbody>
</table>

* 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**

<table>
<thead>
<tr>
<th>Description:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The command <em>getumotmin</em> returns the setting of the motor phase voltage at standstill.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[axisno] getumotmin</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reply:</th>
</tr>
</thead>
<tbody>
<tr>
<td>[vmin]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 4000 *</td>
<td>mV</td>
</tr>
</tbody>
</table>

* see *setumotmin*

<table>
<thead>
<tr>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 <em>getumotmin</em></td>
</tr>
</tbody>
</table>

Reply
2000
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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[vgrad]</td>
<td>&lt; 250 *</td>
</tr>
<tr>
<td>[axis]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

* 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
```
The command `getumotgrad` returns the setting of umotgrad.

**Syntax:**

```
[axisno] getumotgrad
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Reply:**

```
[vgrad]
```

<table>
<thead>
<tr>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[vgrad]</td>
</tr>
</tbody>
</table>

*see `setumotgrad`*

**Example:**

```
1 getumotgrad
```

Reply:

50
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.

<table>
<thead>
<tr>
<th>[pole pairs]</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>Hybrid Stepper Motor with full step size=1.8°</td>
</tr>
<tr>
<td>100</td>
<td>Hybrid Stepper Motor with full step size= 0.9°</td>
</tr>
</tbody>
</table>

**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
```

**Reply:**
```
[pole pairs] range
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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[phases]</td>
<td>2, 3</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1 ..... 16</td>
</tr>
</tbody>
</table>

**Reply:**

```
[phases]
```

<table>
<thead>
<tr>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[phases] 2, 3</td>
</tr>
</tbody>
</table>

**Example:**

```
1 getphases
```

Reply:

2
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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[enable state]</td>
<td>0, 1, 2</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1... 16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[enable state]</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The axis is disabled for all moves.</td>
</tr>
<tr>
<td>1</td>
<td>The axis is enabled for all moves.</td>
</tr>
<tr>
<td>2</td>
<td>The axis is enabled for all moves but <code>ncal, nrm</code></td>
</tr>
</tbody>
</table>

**Example:**

```
1 3 setaxis
```
**getaxis**

<table>
<thead>
<tr>
<th>Description:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The command <strong>getaxis</strong> returns the axis enable setting.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[axisno] getaxis</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1... 16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reply:</th>
</tr>
</thead>
<tbody>
<tr>
<td>[enable state]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[enable state]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 getaxis</td>
</tr>
</tbody>
</table>
The command *setaxisno* is configurates the individual axis number of the controller.
The axis number is used to address 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
```

<table>
<thead>
<tr>
<th><strong>entry</strong></th>
<th><strong>range</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1,..., 16</td>
</tr>
</tbody>
</table>

**Example:**

```
3 setaxisno
3 nsave
```
The command `getaxisno` returns the axis number.

**Syntax:**
```
getaxisno
```

**Reply:**
```
[axisno]
```

<table>
<thead>
<tr>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno] 1... 16</td>
</tr>
</tbody>
</table>

**Example:**
```
getaxisno
```
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
```

<table>
<thead>
<tr>
<th>[function]</th>
<th>NPN Type *</th>
<th>PNP Type **</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>closer (nc)</td>
<td>opener (no)</td>
</tr>
<tr>
<td>1</td>
<td>opener (no)</td>
<td>closer (nc)</td>
</tr>
<tr>
<td>2</td>
<td>disabled</td>
<td>disabled</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>[index]</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>cal switch indexed</td>
</tr>
<tr>
<td>1</td>
<td>rm switch indexed</td>
</tr>
</tbody>
</table>

### 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.
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
```

### Reply:

```
[function cal] [function rm]
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Setting of cal switch</th>
<th>Setting of rm switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 closer (nc)</td>
<td>opener (no)</td>
<td></td>
</tr>
<tr>
<td>1 opener (no)</td>
<td>closer (nc)</td>
<td></td>
</tr>
<tr>
<td>2 disabled</td>
<td>disabled</td>
<td></td>
</tr>
</tbody>
</table>

### Example:

```
3 getsw
```

Reply:
```
0 0
```
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](image1.png) ![Motor direction is changed with setmotiondir](image2.png)

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 *setciperiod*. Then, before closing the loop again, execute a move command and check for consistency of the effected sensor position variation (amount and direction).
Syntax:

```
[motiondir] [axisno] setmotiondir
```

<table>
<thead>
<tr>
<th>[motiondir]</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Moving upward by nominal position has the effect of clockwise axis rotation (factory setting)</td>
</tr>
<tr>
<td>1</td>
<td>Moving upward by nominal position has the effect of anti clockwise axis rotation</td>
</tr>
</tbody>
</table>

Example:

```
1 1 setmotiondir
```
The command `getmotiondir` indicates the effective motor rotation direction. See `setmotiondir` on information about how the direction is defined.

### Syntax:

```
[axisno] getmotiondir
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

### Reply:

```
[motiondir]
```

<table>
<thead>
<tr>
<th>[motiondir]</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Moving upward by nominal position has the effect of clockwise axis rotation (factory setting)</td>
</tr>
<tr>
<td>1</td>
<td>Moving upward by nominal position has the effect of anti clockwise axis rotation</td>
</tr>
</tbody>
</table>

### Example:

```
1 getmotiondir
```
**setnpowerup**

**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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>meaningful combinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>[powerup]</td>
<td>0, 2, 6, 8, 10, 14</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[powerup] bit (dec)</th>
<th>command</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0 (1)</td>
<td>reserved</td>
<td></td>
</tr>
<tr>
<td>D1 (2)</td>
<td>ncal</td>
<td>The axis is moving to the cal limit switch</td>
</tr>
<tr>
<td>D2 (4)</td>
<td>nrm</td>
<td>The axis is moving to the rm limit switch</td>
</tr>
<tr>
<td>D3 (8)</td>
<td>nrandmove</td>
<td>A random move is performed With limit switches installed, it is recommended to have the controller execute ncal and nrm first. ([powerup] = 14)</td>
</tr>
</tbody>
</table>
Note that the controller will not perform an initial \textit{nrm} without an \textit{ncal} prior to it. So the settings 4 and 12 are invalid and have no effect on power up.

\textbf{Example:}

\texttt{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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1.....16</td>
</tr>
</tbody>
</table>

**Reply:**

```
[powerup]
```

**Example:**

```
8 getnpowerup
```

Reply: 14
Velocity and acceleration
**setnvel (snv)**

**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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[velocity]</td>
<td>0.0001...2000.0 *</td>
<td>mm/s</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1...16</td>
<td>-</td>
</tr>
</tbody>
</table>

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

**Pollux speed grades:**

<table>
<thead>
<tr>
<th>Speed grade</th>
<th>Rotational speed range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>max. 13 rev/s</td>
</tr>
<tr>
<td>2</td>
<td>max. 25 rev/s (Option)</td>
</tr>
<tr>
<td>3</td>
<td>max. 40 rev/s (Option)</td>
</tr>
</tbody>
</table>

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

**Syntax:**

`[axisno] getnvel`

**Reply:**

`[velocity]`

<table>
<thead>
<tr>
<th>unit</th>
<th>[velocity]</th>
</tr>
</thead>
<tbody>
<tr>
<td>mm/s</td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

```
2 gnv
```

Reply:

```
18.000000
```
The command `setconfig` is designed for general controller matching via a configuration register, especially for the purpose of establishing compatibility to other SMC controllers.

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

**Syntax:**

```
[config] [axisno] setconfig
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[config]</td>
<td>0, 1</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[config] bit (dec)</th>
<th>description</th>
<th>available settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0 (1)</td>
<td>sign of coordinate origin offset*</td>
<td>0: positive, 1: negative</td>
</tr>
</tbody>
</table>

*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).

**Related command:**

```
setnpos, getconfig
```

**Example:**

```plaintext
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**

### Description:
Returns the content of the controller configuration register.

### Syntax:

```
[axisno] getconfig
```

### Reply:

```
[config]
```

### Related command:
```
setconfig
```

### Example:

```
8 getconfig
```

Reply: 1
**setnaccel (sna)**

**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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[acceleration]</td>
<td>1.0...2000.0</td>
<td>mm/s²</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1,...16</td>
<td>-</td>
</tr>
</tbody>
</table>

**Related commands:**

`getnaccel`

**Example:**

```
500.0 1 sna
```
getnaccel (gna)

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

**Syntax:**
```
[axisno] getnaccel
```

**Reply:**
```
[acceleration]
```

<table>
<thead>
<tr>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0...2000.0</td>
<td>mm/s²</td>
</tr>
</tbody>
</table>

**Example:**
```
1 gna
```

Reply:
```
200.000
```
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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deceleration</td>
<td>steepness of stop deceleration ramp</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>500.0...2000.0</td>
<td>mm/s²</td>
</tr>
<tr>
<td>axisno</td>
<td>1..16</td>
<td>-</td>
</tr>
</tbody>
</table>

**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**

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

**Syntax:**

```
[axisno] getnstopdecel
```

**Related commands:**
`setnstopdecel, gna`

**Reply:**

```
[deceleration]
```

<table>
<thead>
<tr>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>500.0...2000.0</td>
<td>mm/s²</td>
</tr>
</tbody>
</table>

**Example:**

```
4 getnstopdecel
```
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
```

**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]
```

<table>
<thead>
<tr>
<th>description</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[velocity-1]</td>
<td>0.0001...2000.0</td>
<td>mm/s</td>
</tr>
<tr>
<td>[velocity-2]</td>
<td>0.0001...2000.0</td>
<td>mm/s</td>
</tr>
</tbody>
</table>

**Example:**

```
2 getncalvel
```

Reply:

```
2.000000 0.250000
```
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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[velocity]</td>
<td>0.0001 - 2000.0</td>
<td>mm/s</td>
</tr>
<tr>
<td>[index]</td>
<td>1, 2</td>
<td></td>
</tr>
<tr>
<td>[axisno]</td>
<td>1...16</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[index]</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>velocity towards the limit-switch</td>
</tr>
<tr>
<td>2</td>
<td>velocity out of the limit-switch</td>
</tr>
</tbody>
</table>

**Related commands:**

*getnrmvel, setcalvel*

**Example:**

```
2.0 1 1 setnrmvel
5.0 2 1 setnrmvel
```
getnrmvel

Description:
Command getnrmvel returns the two adjusted \textit{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:

[\text{axisno}] \text{getnrmvel}

Reply:

[\text{velocity-1]} [\text{velocity-2}]

<table>
<thead>
<tr>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[\text{velocity-1]}</td>
</tr>
<tr>
<td>velocity towards switch</td>
</tr>
<tr>
<td>[\text{velocity-2]}</td>
</tr>
<tr>
<td>velocity out of switch</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[\text{velocity-1]}</td>
<td>0.0001...2000.0 mm/s</td>
</tr>
<tr>
<td>[\text{velocity-2]}</td>
<td>0.0001...2000.0 mm/s</td>
</tr>
</tbody>
</table>

Example:

3 \text{getnrmvel}

Reply:
2.000000 0.250000
The command `setnrefvel` configurates both velocities at which the reference move is executed.

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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[velocity]</td>
<td>0.0001 - 2000.0</td>
<td>mm/s</td>
</tr>
<tr>
<td>[index]</td>
<td>1, 2</td>
<td>-</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1...16</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[index]</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>velocity to find the mark (fast)</td>
</tr>
<tr>
<td>2</td>
<td>velocity for the final move to the mark</td>
</tr>
</tbody>
</table>

**Related commands:**

`getnrefvel`

**Example:**

```
20.0 1 1 setnrefvel
0.1 2 1 setnrefvel
```
**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]
```

<table>
<thead>
<tr>
<th>description</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[velocity-1]</td>
<td>velocity to find the mark (fast)</td>
</tr>
<tr>
<td>[velocity-2]</td>
<td>velocity for the final move to the mark</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[velocity-1]</td>
<td>0.0001...2000.0 mm/s</td>
</tr>
<tr>
<td>[velocity-2]</td>
<td>0.0001...2000.0 mm/s</td>
</tr>
</tbody>
</table>

**Example:**
```
2 getnrefvel
```

Reply:
```
10.000000 0.500000
```
Positioning commands
**nmove (nm)**

**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
```

**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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[distance]</td>
<td>+/- 2000.0</td>
<td>mm</td>
</tr>
</tbody>
</table>

**Related commands:**

*nmov*, *speed*

**Example:**

```
0.5 1 nr
```
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 `stop-speed`, `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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[velocity]</td>
<td>+/- 2000.0</td>
<td>mm/s</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1....16</td>
<td>-</td>
</tr>
</tbody>
</table>

* depends on the model and the released speed grade.

### Example:

```
10.0 1 speed
```
The command `stoppseed` terminates the constant velocity move of the target axis with the current acceleration setting (see command `sna`).

**Syntax:**

```
[axisno] stopseed
```

**Example:**

```
1 stopseed
```
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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>axis number</td>
<td>1..16</td>
</tr>
</tbody>
</table>

**Example:**

```
14 nrandmove
```
Limit Switch functions
ncalibrate \((ncal)\)

**Description:**
The command \textit{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 \textit{setnpos}.
The lower limit can be changed using \textit{setnlimit}.

The velocity of the \textit{ncal} movement is adjusted using \textit{setncalvel}.

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

**Syntax:**

\[
\text{[axisno] ncal}
\]

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>axis number</td>
<td>1..16</td>
</tr>
</tbody>
</table>

**Example:**

\[1 ncal\]
The command \textit{nrm} executes a limit-switch movement to the \textit{rm} limit-switch. This procedure determines the upper limit of the selected axis.

The velocity of \textit{nrm} movement is adjusted using the command \textit{setnrmvel}.

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

**Syntax:**

\begin{verbatim}
[axisono] nrm
\end{verbatim}

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisono]</td>
<td>axis number</td>
<td>1..16</td>
</tr>
</tbody>
</table>

**Example:**

\begin{verbatim}
1 nrm
\end{verbatim}
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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[distance]</td>
<td>0.0...1.0</td>
<td>mm</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1....16</td>
<td>-</td>
</tr>
</tbody>
</table>

Related command:

*getncalswdist*

Example:

```
5 1 setncalswdist
```
The command `getncalswdist` returns the calibration switch distance.

**Syntax:**

```
[axisno] getncalswdist
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Reply:**

```
[distance]
```

<table>
<thead>
<tr>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[distance] mm</td>
</tr>
</tbody>
</table>

**Example:**

```
1 getncalswdist
```

Reply:

```
2.000000 0.250000
```
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 targeting 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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[lower limit]</td>
<td>lower limit</td>
<td>0...-1000mm</td>
</tr>
<tr>
<td>[upper limit]</td>
<td>upper limit</td>
<td>0...+1000mm</td>
</tr>
</tbody>
</table>

Related command:

```
setinilimit, setnpos
```

Example:

```
0 100.0 1 setnlimit
```
**getnlimit**

<table>
<thead>
<tr>
<th>Description:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The command <em>getnlimit</em> returns the limit coordinates. If no limits are specified, the maximum limit values (+/-1000mm) are returned.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[axisno] getnlimit</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reply:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[lower limit] [upper limit]</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>1 getnlimit</code></td>
</tr>
</tbody>
</table>

Reply:
0.000000 70.000000
Venus-2 for Pollux 96

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:

<table>
<thead>
<tr>
<th>lower limit</th>
<th>upper limit</th>
<th>entry</th>
<th>description</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[lower limit]</td>
<td>[upper limit]</td>
<td>[axisno]</td>
<td>setnlimit</td>
<td>[lower limit]</td>
</tr>
<tr>
<td>[upper limit]</td>
<td></td>
<td></td>
<td></td>
<td>upper limit</td>
</tr>
</tbody>
</table>
Limit configuration of a 100 mm motion range machine.

0 100. 1 setnlimit
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.
The command `getinilimit` returns the initial motion range limit coordinates. If no limit is specified, the maximum limit value (+/-1000mm) is returned.

**Syntax:**

```plaintext
[axisno] getinilimit
```

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[lower limit]</td>
<td>lower limit</td>
<td>0...-1000mm</td>
</tr>
<tr>
<td>[upper limit]</td>
<td>upper limit</td>
<td>0...+1000mm</td>
</tr>
</tbody>
</table>

**Reply:**

```plaintext
[lower limit] [upper limit]
```

**Example:**

```plaintext
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:**

<table>
<thead>
<tr>
<th>ASCII sign</th>
<th>decimal value</th>
<th>hex value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl-C</td>
<td>3</td>
<td>0x3</td>
</tr>
</tbody>
</table>

**Related command:**
`abort`, `Ctrl-B`

**Example:**

`Ctrl-C`
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:

<table>
<thead>
<tr>
<th>ASCII sign</th>
<th>decimal value</th>
<th>hex value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl-B</td>
<td>2</td>
<td>0x2</td>
</tr>
</tbody>
</table>

### Related commands:

* abort, Ctrl-C, setemergency, setaxis*

### Example:

```
Ctrl-B
1 1 setaxis
1 2 setaxis
```
**setemergency**

**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
```

<table>
<thead>
<tr>
<th>[config]</th>
<th>Ctrl-C</th>
<th>Ctrl-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>blocked</td>
<td>blocked</td>
</tr>
<tr>
<td>1</td>
<td>released</td>
<td>blocked</td>
</tr>
<tr>
<td>2</td>
<td>blocked</td>
<td>released</td>
</tr>
<tr>
<td>3</td>
<td>released</td>
<td>released</td>
</tr>
</tbody>
</table>

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

**Related commands:**
*getemergency*

**Example:**
```
1 1 setemergency
```
The command `getemergency` returns the configuration of the emergency shortcuts.

**Syntax:**

```
[axisno] getemergency
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1..16</td>
</tr>
</tbody>
</table>

**Reply:**

```
[config]
```

<table>
<thead>
<tr>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[config]</td>
</tr>
</tbody>
</table>

**Related commands:**

`setemergency`

**Example:**

```
1 getemergency
```

Reply:

```
3
```
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 preceded 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
```

**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
```

**Reply:**

```
[pos]
```

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>axis number</td>
<td>1..16</td>
</tr>
</tbody>
</table>

*returned with 6 decimal places

**Related commands:**

`setselpos`

**Example:**

```
3 npos
```

Example of a reply: **12.041959**
**setnpos**

**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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[coordinate]</td>
<td>+/-1000 mm</td>
<td>mm</td>
</tr>
</tbody>
</table>

**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**

**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] setselpos
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[source]</td>
<td>0, 1</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1, ..., 16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[source] description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  nominal position</td>
</tr>
<tr>
<td>1  actual position</td>
</tr>
</tbody>
</table>

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

**Syntax:**
```
[axisno] getselpos
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1.... 16</td>
</tr>
</tbody>
</table>

**Reply:**
```
[source]
```

<table>
<thead>
<tr>
<th>[source]</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>nominal position</td>
</tr>
<tr>
<td>1</td>
<td>actual position</td>
</tr>
</tbody>
</table>

**Example:**
```
3 getselpos
```
Current status inquiries
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.

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

* Pollux NT only  ** hardware option

**D0: Motion state**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Move finished</td>
</tr>
<tr>
<td>1</td>
<td>Move in progress</td>
</tr>
</tbody>
</table>

**D1:**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no function</td>
</tr>
<tr>
<td>1</td>
<td>no function</td>
</tr>
</tbody>
</table>

**D2: Machine error state**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No machine error</td>
</tr>
<tr>
<td>1</td>
<td>Machine error occurred</td>
</tr>
</tbody>
</table>
### D3: Speed mode state

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Speed mode active</td>
</tr>
<tr>
<td>1</td>
<td>Speed mode not active</td>
</tr>
</tbody>
</table>

### D5: Closed Loop In-Window state

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Current position out of the target window.</td>
</tr>
<tr>
<td>1</td>
<td>Current position within the target window.</td>
</tr>
</tbody>
</table>

### D6: Motor driver state

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Motor driver enabled from hardware input</td>
</tr>
<tr>
<td>1</td>
<td>Motor driver disabled from hardware input</td>
</tr>
</tbody>
</table>

### D7: Motion disable state

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Motion enabled</td>
</tr>
<tr>
<td>1</td>
<td>Motor driver disable event occurred - motion disabled - reset required to restore full operation</td>
</tr>
</tbody>
</table>

#### Syntax:

```
[axisno] nstatus or nst
```

#### Reply:

```
[status]
```

#### Example:

```
1 nstatus
```

```
Reply:
1
```

Bit pattern: 10000000

Decoded state: Move in progress
Decoding examples:

<table>
<thead>
<tr>
<th>bit value</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>[status] = 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[status] = 1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[status] = 32</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[status] = 192</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>[status] = 128</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

*reserved

- Motor power off / motion disabled
- Motor driver disable input
- In-window
- Speed mode
- Machine error occurred
- Moving

Moving machine error occurred
Speed mode
In-window
Motor driver disable input
Motor power off / motion disabled
Error messages
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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>axis number</td>
<td>1..16</td>
</tr>
</tbody>
</table>

**Reply:**

```
[error code]
```
*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 SMC controllers.

### Valid error codes:

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

*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 SMC controllers.

### Example:

1 gne
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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>axis number</td>
<td>1..16</td>
</tr>
</tbody>
</table>

### Reply:

```
[machine error code]
```

### Error codes

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

### Example:

```
10 gme
```
System commands
The command `npop` removes one element from the parameter stack, according to the LIFO principle. (LIFO = Last in first out)

**Syntax:**

```
[axisno] npop
```

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>axis number</td>
<td>1..16</td>
</tr>
</tbody>
</table>

**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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[value]</td>
<td>absolute or relative coordinates</td>
<td>+/- 2000.0 mm</td>
<td></td>
</tr>
<tr>
<td>[axisno]</td>
<td>axis number</td>
<td>1..16</td>
<td>-</td>
</tr>
</tbody>
</table>

**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
```

<table>
<thead>
<tr>
<th>description</th>
<th>digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>YY</td>
<td>2</td>
</tr>
<tr>
<td>HW</td>
<td>2</td>
</tr>
<tr>
<td>SERIAL</td>
<td>4</td>
</tr>
</tbody>
</table>

**Related command:**

*nidentify, nversion*

**Example:**

```
4 getserialno
```

Reply:
03010113
Description:
Year: 2003
Hardwarerevision 01
Number: 0113
The command `nversion` returns the firmware version of the axis.

**Syntax:**

```
[axisno] nversion
```

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>axis number</td>
<td>1..16</td>
</tr>
</tbody>
</table>

**Reply:**

[firmware version]

**Related command:**

`nidentify`

**Example:**

```
1 nversion
```

Reply:

```
2 1 4
```
**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]
```

<table>
<thead>
<tr>
<th>Bit</th>
<th>Dec.</th>
<th>Released option</th>
<th>Pollux Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0</td>
<td>1</td>
<td>Speed grade 2 = 25 rev/s</td>
<td>Pollux-1 / Pollux-2</td>
</tr>
<tr>
<td>D1</td>
<td>2</td>
<td>Speed grade 3 = 40 rev/s</td>
<td>Pollux-1 / Pollux-2</td>
</tr>
<tr>
<td>D2</td>
<td>4</td>
<td>Closed Loop</td>
<td>Pollux-2</td>
</tr>
<tr>
<td>D3</td>
<td>8</td>
<td>CAN Bus</td>
<td>Pollux-2</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>description</th>
<th>Pollux model (Pollux-1 / Pollux-2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[HW-Rev]</td>
<td>Hardware revision</td>
</tr>
<tr>
<td>[SW-Rev]</td>
<td>Software revision</td>
</tr>
<tr>
<td>[ID]</td>
<td>Internal use</td>
</tr>
<tr>
<td>[ID]</td>
<td>Internal use</td>
</tr>
</tbody>
</table>

**Related command:**

*version*

**Example:**

```
1 nidentify
```

Reply:

```
Pollux2 1 214 1 0
```
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
```

**Example:**

```
2 getnfpara
```
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:

```
[axisno] nsave
```

### Syntax:
```
[axisno] nsave
```

### Related commands:
```
nrestore
```

### Example:
```
3 nsave
```
**nclear**

**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. too 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
```

**Related command:**

`ngsp`

**Example:**

```
2 nclear
```
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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>axis number</td>
<td>1..16</td>
</tr>
</tbody>
</table>

**Example:**

```
4 nreset
```
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 - separate 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 inverse 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 targeted 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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>axis number</td>
<td>1..16</td>
</tr>
</tbody>
</table>

Reply:

```
[number]
```

<table>
<thead>
<tr>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[number]</td>
</tr>
</tbody>
</table>

Related command:

```
n clear
```

Example:

```
1 ngsp
```

Reply:

```
2
```
The command `nrestore` reactivates the parameter set valid at the moment of the last execution of `nsave`.

A reply can be enabled with the following command sequence:

```
nrestore
nstatus
```

Related command:

`getnfpara`

Example:

```
1 nrestore
```
Special functions
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:**

```plaintext
[ user number ] [ register ] [ axisno ] setuv
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[user number]</td>
<td>-2000000000 ...+2000000000</td>
</tr>
<tr>
<td>[register]</td>
<td>0...9</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Example:**

```
12041959 2 1 setuv
```
**getuv**

**Description:**
The command `getuv` returns the specified user variable.

**Syntax:**

```
[register]  [axisno]  getuv
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[register]</td>
<td>0...9</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Reply:**

[user number]

**Example:**

```
2 1 getuv
```

Reply

12041959
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

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[distance]</td>
<td>-2000.0...2000.0</td>
<td>mm</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1....16</td>
<td>-</td>
</tr>
</tbody>
</table>

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

**Syntax:**

```
[axisno] getranddist
```

**Reply:**

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

**Example:**

```
1 getranddist
```

Reply:

```
100.000000
```
Closed loop function
The command `setcloop` is used to determine whether 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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[cloop]</td>
<td>0, 1</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1..16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[cloop]</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Closed Loop disabled (Open Loop mode)</td>
</tr>
<tr>
<td>1</td>
<td>Closed Loop enabled</td>
</tr>
</tbody>
</table>

### Related command:

`getcloop`
Example:

```
1 2 setcloop
```
**getcloop**

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

**Syntax**

```
[axisno] getcloop
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Reply:**

```
[cloop]
```

<table>
<thead>
<tr>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[cloop]   0, 1</td>
</tr>
</tbody>
</table>

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
```
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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>unit</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[window]</td>
<td>&lt;= 1.0</td>
<td>mm *</td>
<td>target window of +/- [window] mm around the target position active</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1...16</td>
<td>-</td>
<td>axis number</td>
</tr>
</tbody>
</table>

* resolution 1 nm

### Related command:

`getclwindow, setclwintime`

### Example:

```
0.001 1 setclwindow
```

The target window is adjusted to +/-1µm, related to the desired position.
The command `getclwindow` returns the setting of the Closed Loop target window.

**Syntax:**

```
[axisno] getclwindow
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Reply:**

```
[window]
```

<table>
<thead>
<tr>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;= 1.0</td>
<td>mm</td>
</tr>
</tbody>
</table>

**Example:**

```
1 getclwindow
```

Reply

0.001
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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>unit</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[time]</td>
<td>0...8191</td>
<td>ms</td>
<td>time on target</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1...16</td>
<td>-</td>
<td>axis number</td>
</tr>
</tbody>
</table>

**Related command:**

`getclwintime`, `setclwindow`

**Example:**

```
20 1 setclwintime
```
getclwintime

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

**Syntax:**

```
[axisono] getclwintime
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisono]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Reply:**

```
[time]
```

<table>
<thead>
<tr>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[time]</td>
<td>0...8191 ms</td>
</tr>
</tbody>
</table>

**Example:**

```
1 getclwintime
```

Reply

20
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 travel range, the move terminates at the given [target position]. The current reference status is returned by the command `getrefst`.

With `Ctrl-C` or `Ctrl-B` the reference move will be aborted.

Syntax:

```
[target position] [axisno] nrefmove
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[target position]</td>
<td>+/-1000</td>
<td>mm</td>
</tr>
</tbody>
</table>

Related commands:

`setnrefvel, getrefst`

Examples:

```
100.0 1 nrefmove
```

The controller will move towards coordinate 100 mm. 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 to coordinate 100 mm before it stops.

```
-7.0 2 nrefmove
```

With this command, axis 2 moves to coordinate -7 mm to find the reference mark.
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
```
The command `setref` defines on which transitions of the reference signal (from an external measurement system) the axis shall react.

**Syntax:**

```
[config] [axisno] setref
```

**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.
The command \texttt{getref} delivers the configurations of the command \texttt{setref}.

Syntax:

\[ \texttt{[axisno]} \texttt{getref} \]

Related commands:

\texttt{setref}
\texttt{getrefst}

Reply:

\[ \texttt{[config]} \]

<table>
<thead>
<tr>
<th>config</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>detect positive signal transition</td>
</tr>
<tr>
<td>1</td>
<td>detect negative signal transition</td>
</tr>
</tbody>
</table>

Example:

\texttt{1 getref}
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
```
- 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 \[\text{distance}\] 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

\[\text{distance} = -0.020\,\text{mm}\]

The value of \text{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:

\[\text{distance} = 10\,\text{mm} / 1000\,\text{periods} = 0.01\]
getclperiod

**Description:**
The command `getclperiod` returns the setting of `setclperiod`.

**Syntax:**

```
[axissno] getclperiod
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axissno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Reply:**

```
[period]
```

<table>
<thead>
<tr>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[period]</td>
<td>-1.0...1.0 mm</td>
</tr>
</tbody>
</table>

**Example:**

```
1 getclperiod
```

Reply:

```
0.020
```
The command *setscaleinterface* adapts the Closed Loop interface to the type of the connected position encoder.

**Syntax:**

```
[type] [axismo] setscaleinterface
```

<table>
<thead>
<tr>
<th>[type]</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MR type, 0.5 mm period length</td>
</tr>
<tr>
<td>1</td>
<td>1 Vpp type, period length adaptable (&lt; 1 mm)</td>
</tr>
<tr>
<td>2</td>
<td>MR type, 1.0 mm period length</td>
</tr>
</tbody>
</table>

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

**Syntax:**

```
[axisno] getscaleinterface
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>axisno</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Reply:**

```
[type]
```

<table>
<thead>
<tr>
<th>[type]</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MR type, 0.5 mm period length</td>
</tr>
<tr>
<td>1</td>
<td>1 Vpp type, period length adaptable (&lt; 1 mm)</td>
</tr>
<tr>
<td>2</td>
<td>MR type, 1.0 mm period length</td>
</tr>
</tbody>
</table>

**Example:**

```
1 getscaleinterface
```

Reply:

```
1
```
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.

`serrMax`

With the absolute value of the `serrMax` (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 `serrMax` 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`. 
Syntax:

```plaintext
[value] [index] [axisno] setsp
```

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>selects 1 of the 10 parameters</td>
<td>1..10</td>
</tr>
<tr>
<td>axisno</td>
<td>axis number</td>
<td>1..16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[value] [index] description</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>integral gain</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>serrMax</td>
<td>max. position error</td>
<td>[-1....1] mm</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Examples:

**1000 2 1 setp**

The I-Controller is parameterized.

**0.459000 7 1 setp**

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 setp**

A running move is stopped if the distance between actual position and nominal position exceeds 0.01 mm, the motor stops and the motor current is switched off. No further moves are possible until the axis is enabled again.
The command *getsp* returns the Closed Loop parameter settings.

**Syntax:**

```
[index] [axisno] getsp
```

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[index]</td>
<td>selects 1 of the 10 parameters</td>
<td>1..10</td>
</tr>
<tr>
<td>[axisno]</td>
<td>axis number</td>
<td>1..16</td>
</tr>
</tbody>
</table>

**Related command:**

*setsp*

**Example:**

```
2 4 getsp
```
The command \texttt{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.

\begin{verbatim}
[axisno] a
\end{verbatim}
<table>
<thead>
<tr>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[sin] sine channel</td>
<td>-32768...32767</td>
</tr>
<tr>
<td>[cos] cosine channel</td>
<td>-32768...32767</td>
</tr>
</tbody>
</table>

Example:

1 a

Reply:
15000 -15000
Backlash compensation
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
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[blc]</td>
<td>0, 1</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Related command:**

`getblc, setblcd, setblcs`

**Example:**

```
1 1 setblc
```
The command `getblc` returns the status of the function "backlash-compensation".

**Syntax:**

```
[axisno] getblc
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Reply:**

```
[blc]
```

<table>
<thead>
<tr>
<th>[blc]</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>compensation disabled</td>
</tr>
<tr>
<td>1</td>
<td>compensation enabled</td>
</tr>
</tbody>
</table>

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
```
**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:**

```
setblcd [blcd] [axisno] `setblcd`
```

<table>
<thead>
<tr>
<th>entry</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[blcd]</td>
<td>distance to be compensated</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[blcd]</td>
<td>0.0...0.1</td>
<td>mm</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1...16</td>
<td>-</td>
</tr>
</tbody>
</table>

**Example:**

```
0.001 1 setblcd
```
The command `getblcd` returns the backlash distance value.

**Syntax:**
```
[axisno] getblcd
```

**Reply:**
```
[blcd]
```

**Example:**
```
1 getblcd
```
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:

\[ t\text{Comp [ms]} = 0.25 \times 2^{\text{blcs}} \]

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

\[ t\text{Comp}_{\text{min}} [\text{ms}] = \text{polepairs} \times \text{blcd} / \text{pitch} \]

With tComp = tComp_{\text{min}}, the compensation distance will be realized by one jump.

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

### Syntax:

\[ \text{[blcs] [axisno]} \text{ setblcs} \]

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[blcs]</td>
<td>1...12</td>
<td>time exponent</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1...16</td>
<td>axis number</td>
</tr>
</tbody>
</table>

### Example:

10 1 setblcs

tComp = 2^{10} \times 250 \mu s = 256 ms
The command `getblcs` returns the currently set value of the backlash time exponent.

**Syntax:**

```
[axisno] getblcs
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Example:**

```
1 getblcs
```
CAN configuration
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 information on appropriate settings, see Chapter "Principles of communication".

**Syntax:**

```
[address][axisno] setcanbaseadr
```

**Entry**

<table>
<thead>
<tr>
<th>entry</th>
<th>range (dec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[address]</td>
<td>0 ... 2000000000</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1 ... 16</td>
</tr>
</tbody>
</table>

**Related command:**

`getcanbaseadr, setcananswadr`

**Example:**

```
1000 1 setcanbaseadr
```
**getcanbaseadr**

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

**Syntax:**
```
[axisno] getcananswadr
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Reply:**
```
[address]
```

<table>
<thead>
<tr>
<th>range (dec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[address]</td>
</tr>
<tr>
<td>0 ... 2000000000</td>
</tr>
</tbody>
</table>

**Example:**
```
2 getcananswadr
```

Reply
400
**setcananswadr**

**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 information on appropriate settings, see Chapter "Principles of communication".

**Syntax:**

```
[address][axisno] setcananswadr
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range (dec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[address]</td>
<td>0 ... 2000000000</td>
</tr>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Related command:**

`setcanbaseadr, getcananswadr`

**Example:**

```
2000 1 setcananswadr
```
**getcananswadr**

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

**Syntax:**
```
[axisno] getcananswadr
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Reply:**
```
[address]
```

<table>
<thead>
<tr>
<th>range (dec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[address]</td>
</tr>
<tr>
<td>0 ... 2000000000</td>
</tr>
</tbody>
</table>

**Example:**
```
2 getcananswadr
```

Reply

500
**setcanbaudrate**

### 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

<table>
<thead>
<tr>
<th>$[select]$</th>
<th><strong>transmission rate</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10 kBaud</td>
</tr>
<tr>
<td>1</td>
<td>20 kBaud</td>
</tr>
<tr>
<td>2</td>
<td>50 kBaud</td>
</tr>
<tr>
<td>3</td>
<td>100 kBaud</td>
</tr>
<tr>
<td>4</td>
<td>125 kBaud</td>
</tr>
<tr>
<td>5</td>
<td>250 kBaud</td>
</tr>
<tr>
<td>6</td>
<td>500 kBaud</td>
</tr>
<tr>
<td>7</td>
<td>800 kBaud</td>
</tr>
<tr>
<td>8</td>
<td>1000 kBaud</td>
</tr>
</tbody>
</table>

### Example:

```
3 1 setcanbaudrate
```
**getcanbaudrate**

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

**Syntax:**

```
[axisno] getcanbaudrate
```

<table>
<thead>
<tr>
<th>entry</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>[axisno]</td>
<td>1...16</td>
</tr>
</tbody>
</table>

**Reply:**

```
[select]
```

<table>
<thead>
<tr>
<th>[select]</th>
<th>transmission rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10 kBaud</td>
</tr>
<tr>
<td>1</td>
<td>20 kBaud</td>
</tr>
<tr>
<td>2</td>
<td>50 kBaud</td>
</tr>
<tr>
<td>3</td>
<td>100 kBaud</td>
</tr>
<tr>
<td>4</td>
<td>125 kBaud</td>
</tr>
<tr>
<td>5</td>
<td>250 kBaud</td>
</tr>
<tr>
<td>6</td>
<td>500 kBaud</td>
</tr>
<tr>
<td>7</td>
<td>800 kBaud</td>
</tr>
<tr>
<td>8</td>
<td>1000 kBaud</td>
</tr>
</tbody>
</table>

**Example:**

```
2 getcanbaudrate
```

Reply

5
## Venus-2 command list

in alphabetic order

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>158</td>
</tr>
<tr>
<td>Ctrl-B</td>
<td>101</td>
</tr>
<tr>
<td>Ctrl-C</td>
<td>100</td>
</tr>
<tr>
<td>getaxis</td>
<td>54</td>
</tr>
<tr>
<td>getaxisno</td>
<td>56</td>
</tr>
<tr>
<td>getblc</td>
<td>162</td>
</tr>
<tr>
<td>getblcd</td>
<td>164</td>
</tr>
<tr>
<td>getblcs</td>
<td>166</td>
</tr>
<tr>
<td>getcananswadr</td>
<td>171</td>
</tr>
<tr>
<td>getcanbaseadr</td>
<td>169</td>
</tr>
<tr>
<td>getcanbaudrate</td>
<td>173</td>
</tr>
<tr>
<td>getcloop</td>
<td>141</td>
</tr>
<tr>
<td>getclperiod</td>
<td>152</td>
</tr>
<tr>
<td>getclwindow</td>
<td>143</td>
</tr>
<tr>
<td>getclwintime</td>
<td>145</td>
</tr>
<tr>
<td>getconfig</td>
<td>70</td>
</tr>
<tr>
<td>getemergency</td>
<td>103</td>
</tr>
<tr>
<td>getinilimit</td>
<td>98</td>
</tr>
<tr>
<td>getmerror (gme)</td>
<td>117</td>
</tr>
<tr>
<td>getmotiondir</td>
<td>61</td>
</tr>
<tr>
<td>getncaccel (gna)</td>
<td>72</td>
</tr>
<tr>
<td>getncalswdist</td>
<td>92</td>
</tr>
<tr>
<td>getncalvel</td>
<td>76</td>
</tr>
<tr>
<td>getnerror (gne)</td>
<td>115</td>
</tr>
<tr>
<td>getnpara</td>
<td>126</td>
</tr>
<tr>
<td>getnlimit</td>
<td>95</td>
</tr>
<tr>
<td>getnoptions</td>
<td>123</td>
</tr>
<tr>
<td>getnpowerup</td>
<td>64</td>
</tr>
<tr>
<td>getnrefvel</td>
<td>80</td>
</tr>
<tr>
<td>getnmvel</td>
<td>78</td>
</tr>
<tr>
<td>getnstdecel</td>
<td>74</td>
</tr>
<tr>
<td>getnvel (gnv)</td>
<td>67</td>
</tr>
<tr>
<td>getphases</td>
<td>52</td>
</tr>
<tr>
<td>getpitch</td>
<td>44</td>
</tr>
<tr>
<td>getpolepairs</td>
<td>50</td>
</tr>
<tr>
<td>getranddist</td>
<td>137</td>
</tr>
<tr>
<td>getref</td>
<td>149</td>
</tr>
<tr>
<td>getrefst</td>
<td>147</td>
</tr>
<tr>
<td>getscaleinterface</td>
<td>154</td>
</tr>
<tr>
<td>getserialno</td>
<td>109</td>
</tr>
<tr>
<td>getselpos</td>
<td>121</td>
</tr>
<tr>
<td>getsp</td>
<td>157</td>
</tr>
</tbody>
</table>
Command List

getsw............................................................................................................. 58
getswst .......................................................................................................... 90
getumotgrad .................................................................................................. 48
getumotmin.................................................................................................... 135
getuv............................................................................................................ 130
nabort ........................................................................................................ 104
ncalibrate (ncal) ........................................................................................... 88
nclear ........................................................................................................... 128
ngsp ............................................................................................................ 130
nidentify ....................................................................................................... 125
nmove (nm) .................................................................................................. 82
npop .............................................................................................................. 119
npos ........................................................................................................... 106
npush .......................................................................................................... 120
nrandmove .................................................................................................. 86
nrangemeasure (nm) ..................................................................................... 89
nrefmove ..................................................................................................... 146
nreset ........................................................................................................... 129
nrestore ...................................................................................................... 132
nrmove (nr) .................................................................................................. 83
nsave ........................................................................................................... 127
nstatus (nst) ............................................................................................... 111
nversion ...................................................................................................... 122
setaxis .......................................................................................................... 53
setaxisno ..................................................................................................... 55
setblc ......................................................................................................... 161
setblcd ......................................................................................................... 163
setblcs ......................................................................................................... 165
setcananswadr ............................................................................................ 170
setcanbaseadr ............................................................................................. 168
setcanbaudrate ........................................................................................... 172
setcirc .......................................................................................................... 139
setclperiod ................................................................................................. 150
setclwindow ............................................................................................... 142
setclwintime ............................................................................................... 144
setconfig ..................................................................................................... 68
setemergency .............................................................................................. 102
setinilimit .................................................................................................... 96
setmotiondir ............................................................................................... 59
setnaccel (sna) ............................................................................................ 71
setncaalwsdist ........................................................................................... 91
setncalvel ................................................................................................... 75
setnlimit ...................................................................................................... 93
setnpos ........................................................................................................ 107
setnpowerup ............................................................................................... 62
setnrefvel ................................................................................................... 79
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>setnrmvel</td>
<td>77</td>
</tr>
<tr>
<td>setnstopdecel</td>
<td>73</td>
</tr>
<tr>
<td>setnvel (snv)</td>
<td>66</td>
</tr>
<tr>
<td>setphases</td>
<td>51</td>
</tr>
<tr>
<td>setpitch</td>
<td>42</td>
</tr>
<tr>
<td>setpolepairs</td>
<td>49</td>
</tr>
<tr>
<td>setranddist</td>
<td>136</td>
</tr>
<tr>
<td>setref</td>
<td>148</td>
</tr>
<tr>
<td>setscaleinterface</td>
<td>153</td>
</tr>
<tr>
<td>setselpos</td>
<td>108</td>
</tr>
<tr>
<td>setsp</td>
<td>155</td>
</tr>
<tr>
<td>setsw</td>
<td>57</td>
</tr>
<tr>
<td>setumotgrad</td>
<td>47</td>
</tr>
<tr>
<td>setumotmin</td>
<td>45</td>
</tr>
<tr>
<td>setuv</td>
<td>134</td>
</tr>
<tr>
<td>speed</td>
<td>84</td>
</tr>
<tr>
<td>stopspeed</td>
<td>85</td>
</tr>
</tbody>
</table>