Corvus PCI

high resolution positioning controller

Venus-1 Command Language

About this documentation

This handbook provides detailed information on the Venus-1 command language. Their function and syntax is explained.

The grouping of the commands in function groups improves the overview.

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.

As far as it is necessary to comprehend the correlations, hardware specific peculiarities of the controller are also explained. These attributes are described in greater detail in the manual of the controller.

Contents

Velocity and acceleration

Positioning commands

Abortion commands

Workspace and point of origin

Current status inquiries

Input / Output Functions

Joystick / Handwheel

System commands

Configuration examples

Introduction in Venus-1

Venus-1 is an interpreter language

Venus-1 commands consist of ASCII-signs which are inter-
preted 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 or Ethernet interface.
In the simplest way the commands are directly transmitted to
the controller via an ASCII terminal.

History

Venus-1 for Corvus has been developed on the basis of the
interpreter language Venus-1 for the controllers mc-compact,
smc-compact, MC-2000 and MC-3000.
The fundamental command construction is identical.
All basic functions are compatible to the former version.

Command syntax for parametrisation

The parameterisation commands are assembled following this scheme:

[parameter] _ [axis index] _ [command] _

_ blank, (space) or (SP)

Parameter

The parameter transmits a value without any unit.
If several parameters are prescribed for one command, they have to be divided by a blank (SP).

The following numbers and characters are permitted for parameters:

Letters	not allowed
Numbers	0-9
Characters	+ - .

-1 Parameter

Most of the get-commands allow the the combination -1 to read out the setting of all axes.

For example:

With the command *2 getpitch* the spindle pitch of axis-2 is asked.
The command *-1 getpitch* reads out the pitch setting of all axes.

Axes index

With the axes index the target axis is addressed. The number of the index is equal with the labeling at the motor connector.

Axis label	Axis index
Axis-1	1
Axis-2	2
Axis-3	3

Commands

For the parametrisation the commands are named with get or set. It consists of several ASCII characters, capitalization is distinguished.

The following letters are allowed for commands:

ASCII-Characters	a-z A-Z
Umlauts	not allowed
Numbers	not allowed

Command syntax for positioning commands

The positioning commands are assembled following this scheme:

[Axis-1] _ [Axis-2] _ [Axis-3 _ [command] _

_ blank, (space) or (SP)
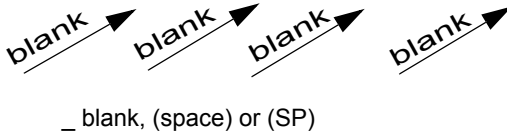
Axis-1, Axis-2, Axis-3

For positioning, the absolut or relative coordinates are trans-fered to the controller.
The values must be divided by a blank (SP).

The number of position values to be transfered depends on the setting of *setdim.*

setdim	Axis that must be trans-fered
1 setdim	Axis-1
2 setdim	Axis-1_Axis-2
3 setdim	Axis-1_Axis-2_Axis-3

If insufficient coordinates are transfered, the move command will not be executed.
Useless coodinates will remain on the stack

The following letters are allowed for position coordinates:

Letters	no
Numbers	0 - 9
Characters	+ -.

Command ending character while transmitting

In the **host mode** data which are transmitted have to be completed with a blank
[parameter] _ [axes index] _ [command] _

In the **terminal mode** the command ending is executed by CR (carriage return).
[parameter] _ [axes index] _ [command] CR

Command ending character while receiving

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

Data which are delivered by the controller are always completed with ASCII (CR) and (LF). Different data requests deliver several parameters in several lines. In these cases each line is completed with (CR) and (LF).
How many lines a request delivers is mentioned in the command description.

Table of important ASCII signs for programming

ASCII Code	Sign	Dez	HEX
CR	Ctrl-M	13	0xD
LF	Ctrl-J	10	0xA
SP		32	0x20
ETX	Ctrl-C	3	0x3

Command execution

For the correct programming it is important to know the internal courses during the execution of the interpreter commands.

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

- **data input memory**
- **scanner and stack**
- **interpreter**

Data input memory

The ASCII Data´s from the communication interface are
 transferd to the data input memory and remain there until
they are processed. The memory possesses a FIFO structure
The data input memory is able to accept up to 256 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 not to much data should be transmitted too
fast to the controller.
The FIFO is deleted after switching off the controller.

Scanner ->interpreter ->stack

The data in the FIFO is read by the scanner and during this checked for parameters and commands.
The parameters are transmitted to a stack which can accept up to 99 values.
Commands are passed on to the interpreter as soon as it is free. The interpreter takes the assigned parameters from the stack and executes the command.
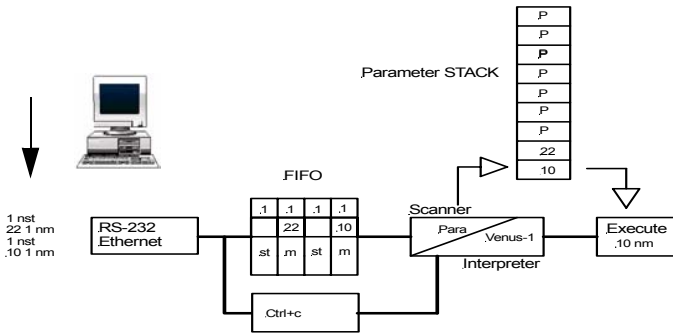
Blocking and non blocking commands

During the interpreter executes a positioning it is able to exe-
cute several other commands parallel to it. These commands
are called non blocking commands.

On the other side there are commands that will be only exe-
cuted until the positioning is finished.
If such a command is stored in the FIFO all other commands
behind it will be blocked until the command is executed and
removed from the FIFO itself.
These commands are also called blocking command.

Examples of blocking and non blocking commands

The interpreter is able to execute several commands at the
same time.
Below mentioned, the interpreter executes the instruction
p and is also free to process 3x *st*.
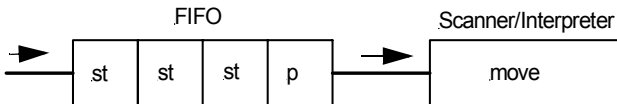
The interpreter has been blocked with the command *ge*.

The interpreter is executes the command *move*. The FIFO contains the command *ge*, this blocks the interpreter for the execution of further commands until *move* is completed After *ge* is executed the commands *st* are processed.

FIFO   Scanner/Interpreter

| st | st | st | **ge** | ✕ | move |
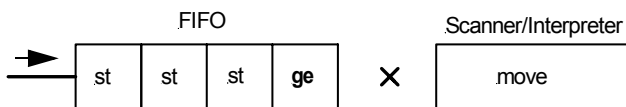
List of non blocking commands

The commands below do not block the interpreter. These commands are also executed if the interpreter processes a *move* command.

Command	Description
st	Status
p	Current position
getin	Read digital Input
setout	Write digital Output
abort	Aborts the current command Attention: This command has to pass through the FIFO it's execution could be delayed with a blocking command.
Ctrl+c	Aborts the current command This commands has **not** to pass the FIFO and could not be delayed with a blocking command.

Unlock interpreter

A lasting blockage of the interpreter by blocking commands is not possible, because all commands are finally processed. To accelerate the unlocking, the command *Ctrl+c* can be used.

This command always aborts the current command and by that it is possible to execute the blocking commands in the FIFO faster.

Terminate command execution

The current executed command is aborted immediately with *Ctrl+c*.

This command must not pass through the FIFO and has a direct access to the interpreter.

The FIFO will not be erased.
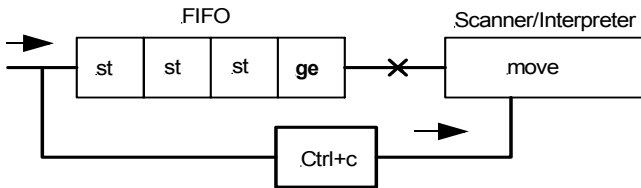
Producing an automatic status reply message

With the following sequence of instructions the blocking effect of the interpreter can be used to generate an automatic status reply.

10 10 2 move
0 0 1 r
st
ge

Effect:
An automatic status feedback is produced, after the instruction *10 10 2 move* is processed.

Description:
The instruction *0 0 0 r* has only the function to block the interpreter and prevent the execution of *st*.
After *move* is executed, *0 0 0 r* is processed with no effect, because it is a relative positioning with 0mm.
Afterwards the command *st* produces the desired status feedback.
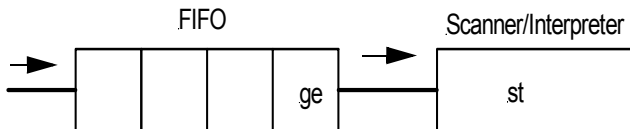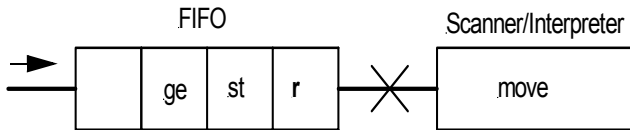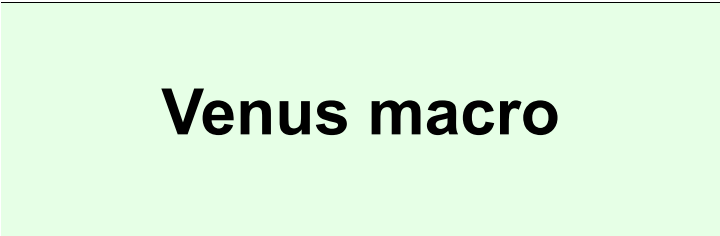
Corvus communication concept

Ethernet and RS-232

RS-232 is the standard communication interface of Corvus.
Optionally the Ethernet interface can be released.
Both interfaces are always ready to receive data.
The data feedback is automatically send to the interface from where the data inquiry comes.
Terminal and host mode are supported from both interfaces.
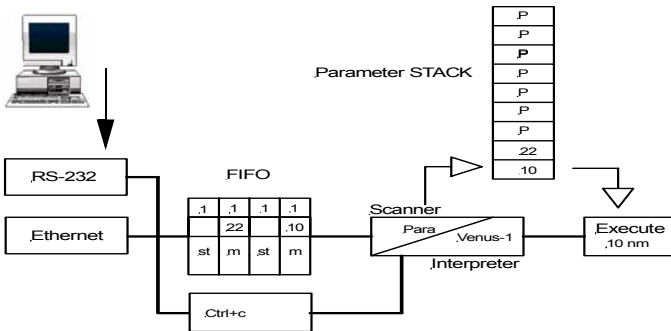
Venus macro

What is a Venus macro?

A Venus macro is in principle a script file consist of arbitrary Venus-1 commands. This file must be downloaded via the serial line one time and can be executed limitless with a macro start command until the controller is switched off.

The greatest benefit of the macro feature in the actual implementation is to reduce the communication overhead, to give the real-time functions "trigger out" much more performance.

Macro Syntax

beginmakro [Venus-1 commands] endmakro

Example to built a Venus macro with the trigger out com-
mand *wpot.*

beginmakro
0.08 m
0.0 1 1 1 1 1 wpot
0.01 1 1 1 1 1 wpot
0.02 1 1 1 1 1 wpot
0.03 1 1 1 1 1 wpot
0.04 1 1 1 1 1 wpot
0.05 1 1 1 1 1 wpot
0.06 1 1 1 1 1 wpot
0m
0.06 0 1 1 1 1 wpot
0.05 0 1 1 1 1 wpot
0.04 0 1 1 1 1 wpot
0.03 0 1 1 1 1 wpot
0.02 0 1 1 1 1 wpot
0.01 0 1 1 1 1 wpot
0.0 0 1 1 1 1 wpot
endmakro

Macro commands

- *listmakro*

This macro command must be send via the RS-232 line to the
controller to ask the number of symbols that are included in
the downloaded macro.

- *startmakro*

This macro command must be send via RS-232 line to start a
downloaded macro.

Communication

mode

storable

With the command *mode* it is possible to switch the communication into terminal or host mode.

In **terminal mode** a terminal screen mask is automatically transmitted from the controller to all communication interfaces. The mask delivers an input line for Venus-1 commands as well as a position display.

In **host mode** communication data are only transmitted from the controller if the host ask for them.

Alternatively it is possible to switch the two modes by DIP-Switch 6 directly at the controller.

See **Corvus Manual**.

Syntax:

[Index] *mode*

[Index]	Description
0	Host Mode
1	Terminal Mode

Example:

1 mode

Corvus is switched to Terminal Mode

Basic settings

getfpara

Description:

The command *getfpara* activates the factory configuration.

Attention:
Doing this all current parameters are overwritten but not stored.

Syntax:

getfpara

Reply:

Reply can be indirectly controlled with the command sequence *getfpara status*

Example:

getfpara

setdim

Description:

The command **setdim** defines how many parameters the controller expects or replies if a dimension dependent command is used.

With the command **setdim** an axis can not be switched off.

Only the following combinations are possible.

setdim	expected or replied axes parameters
1 setdim	Axis-1
2 setdim	Axis-1_Axis-2
3 setdim	Axis-1_Axis-2_Axis-3

Examples:

Positioning commands:
Depending on the setting of **setdim** one, two or three coordinates must be transferred with a positioning command **rmove** or **move**.

Positioning replies:
Depending on the setting of **setdim** one, two, or three positions values are read back with the command **pos.**

Syntax:

[Dimension] **setdim**

	Range
[Dimension]	1, 2, 3

Related command:

getdim

Example:

1 setdim

getdim

Description:

The command *getdim* replies the setting of *setdim*.

Syntax:

getdim

Reply:

[Dimension]

	Range
[Dimension]	1, 2, 3

Example:

getdim

setunit

Description:

The command *setunit* defines the units of the input and output values.
The unit of each axis can be individually assigned.
With index number 0 (virtual 0-axis) the unit of speed and acceleration is specified.

The speed settings *setcalvel* and *setrmvel* are using the fixed unit **rev/s** and are not affected from *setunit*

To be compatible with the older controllers the unit Microstep is emulated from Corvus.
With this unit the resolution of the controller is reduced.
 1 Microstep = 1 Revolution / 40000

This unit should not be used for new product developement.

Syntax:

[Index] [Axis] *setunit*

	Range
[Index]	0, 1, 2, 3, 4, 5, 6
[axis]	0, 1, 2, 3

[Index]	unit
0	Microstep
1	µm
2	mm
3	cm
4	m
5	inch
6	mil (1/ 1000 inch)

Related command:

getunit

Example:

2 0 setunit

The unit of the virtual 0-axis is set to **mm**.
As a result the output values of speed and acceleration are
assigned to the units mm/s resp. mm/s².
The input values are also interpreted as mm/s resp. mm/s²

1 1 setunit
The unit of axis-1 is set to **µm**.
All axis specific inputs and outputs are assigned to this unit.

getunit

Description:

The command *getunit* replies the unit settings of the controller

Syntax:

[axis] *getunit*

	Range
[Axis]	-1, 0, 1, 2, 3

Reply:

[Index]

	Range
[Index]	0, 1, 2, 3, 4, 5, 6

Example:

1 getunit   *-1 getunit*

Reply: Reply:
1   2 1 1 1

setumotmin

Description:

storable

The command ***setumotmin*** is setting a value for the motor phase current. This setting influence the holding torque and the torque in the lower speed range
The value of umotmin does not correspond to the real motor-current it is only an index.

A greater index value, will increase the motor current and the power consumption of the motor drivers.

Syntax:

[Index] [Axis] ***setumotmin***

	Index range
[Index]	0 - 3000
[Axis]	1, 2, 3

Related command:

getumotmin, setumotgrad

Example:

2000 1 setumotmin

getumotmin

Description:

The command *getumotmin* reads out the setting of umotmin.

Syntax:

[axis] *getumotmin*

	range
[Axis]	-1, 1, 2, 3

Reply:

[Index]

	range
[Index]	0 - 3000

Example:

1 getumotmin	*-1 getumotmin*
Reply:	Reply:
2000	2000
	1000
	750

setumotgrad

Description:

storable

The command **setumotgrad** is setting a value for the motor phase current. This setting influence torque in the middle and upper speed range

The value of umotgrad does not correspond to the real motor current it is only an index.

A greater Index value, will increase the motor current and the power consumption of the motor drivers.

Syntax:

[Index] [Axis] **setumotgrad**

	Range
[Index]	0 - 300
[Axis]	1, 2, 3

Related commands:

getumotgrad, setumotmin

Example:

70 1 setumotgrad

getumotgrad

Description:

The command *getumotgrad* reads out the setting of umotgrad.

Syntax:

[Axis] *getumotgrad*

	Range
[Axis]	-1, 1, 2, 3

Reply:

[Index]

	Range
[Index]	0 - 300

Example:

1 getumotgrad   *-1 getumotgrad*

Reply: Reply:
50   50
                                         40
                                         100

setaxis

storable

The command *setaxis* enables or disables the specified axis for the positioning or the endswitch move procedure (cal/rm).

setaxis also has an effect for the commands *pos*, *setpos cal, rm* and the hardware limits

The settings are significant for the programmable and manual move.

Syntax:

[Index] [Axis] *setaxis*

	Range
[Index]	0, 1, 2, 3, 4
[Axis]	1, 2, 3

[Index] = 0:

The axis is consequently disabled for all movements.
The commands *cal*, *rm* and *0 0 0 setpos* will clear the actual position but will not change the hardware limits of the axis.

[Index] = 1:

The axis is consequently enabled for all movements.
The commands *cal*, *rm* and *0 0 0 setpos* will clear the actual position and also clear the hardware limits of the axis.

[Index] = 2:

The axis is restricted enabled.
The endswitch movement procedure cal / rm
will not be processed.
The commands *cal*, *rm* and *0 0 0 setpos*
will clear the actual position but will not change
the hardware limits of the axis.

[Index] = 3:

The axis is consequently disabled for all
movements.
The commands *cal*, *rm* and *0 0 0 setpos*
will **not** clear the actual position and **not**
change the hardware limits of the axis.

[Index] = 4:

The axis is restricted enabled.
The endswitch movement procedure cal / rm
will not be processed.
The commands *cal*, *rm* and *0 0 0 setpos*
will **not** clear the actual position and **not**
change the hardware limits of the axis.

Related command:

getaxis

Example:

1 3 setaxis

getaxis

Description:

The command *getaxis* replies the setting of *setaxis*

Syntax:

[Axis] *getaxis*

	Range
[Axis]	-1, 1, 2, 3

Reply:

[Index]

	Range
[Index]	0, 1, 2, 3, 4

Example:

2 getaxis   *-1 getaxis*

Reply: Reply:
2   1 2 2

setcalvel

Description:

The command *setcalvel* defines two velocities for the cal endswitch movement. The setting is significant for all axes.

storable

1. Velocitiy moving to the endswitch
2. Velocitiy moving out from the endswitch

To be compatible with the older controllers the unit of velocity is revolutions/s .
The resulting velocity in mm/s depends on the pitch value of the virtuell 0-axis. (see command *setpitch*)

Syntax:

[Velocity] [Index] *setcalvel*

[Index]	Description
1	Velocity to the endswitch
2	Velocity out of the endswitch

	Range	Unit
[Velocity]	0 - 45	revolution/s
[Index]	1, 2	-

Related commands:

getcalvel, setrmvel

Example:

2 0 setpitch (virtuell 0-axis)
2 1 setcalvel
1 2 setcalvel
The controller is moving with 2 U/s (4 mm/s) to the cal endswitch and with 1 U/s (2 mm/s) out of the cal endswitch.

getcalvel

Description:

The command *getcalvel* reads back the two adjusted cal movement velocities.

Syntax:

getcalvel

Reply:

[velocity 1]
[velocity 2]

	Range	Unit
[velocity 1]	0 - 45	rev. /s
[velocity 2]	0 - 45	rev. /s

Example:

getcalvel

Reply:

2.000000
0.250000

setrmvel

Description:

The command ***setrmvel*** defines two velocities for the rm endwitch movement. The setting is significant for all axes.

storable

1. Velocitiy moving to the endswitch
2. Velocity moving out from the endswitch

To be compatible with the older controllers the unit of velocity is rev. /s .
The resulting velocity in mm/s depends on the pitch value of the virtuell 0-axis.

Syntax:

[Velocity] [Index] ***setcalvel***

[Index]	Description
1	Velocity to the endswitch
2	Velocitiy out of the endswitch

	Range	Unit
[Velocity]	0 - 45	rev. / s
[Index]	1, 2	-

Related commands:

getcalvel, setrmvel

Example:

2 0 setpitch (virtuell 0-axis)
2 1 setrmvel
1 2 setrmvel
The controller is moving with 2 U/s (4 mm/s) to the rm endswitch and with 1 U/s (2 mm/s) out of the rm endswitch.

getrmvel

Description:

The command *getrmvel* reads back the two adjusted rm movement velocities.

Syntax:

getrmvel

Reply:

[velocity 1]
[velocity 2]

	Range	Unit
[velocity 1]	0 - 45	rev. /s
[velocity 2]	0 - 45	rev. /s

Example:

getrmvel

Reply:

2.000000
0.250000

setaccelfunc

Description:

The command *setaccelfunc* defines with which acceleration function the axes are positioned.

The following acceleration functions are possible:
- Linear accelereation (trapezioal)
- Sin² acceleration (S-curve)
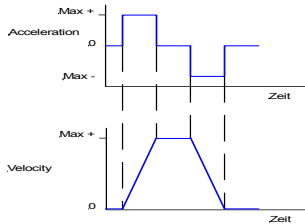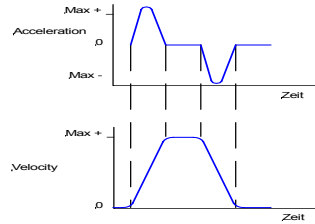
Linear acceleration   sin² acceleration

Syntax:

[Index] *setaccelfunc*

[Index]	Description
0	Linear acceleration
1	Sin² acceleration

Related command:

getaccelfunc

Example:

1 setaccelfunc

getaccelfunc

Description:

The command *getaccelfunc* reads the adjusted accelera-
tion function of the axes.

Syntax:

getaccelfunc

Reply:

[Index]

	Range
[Index]	0, 1

Example:

getaccelfunc

setpitch

With the command *setpitch* the axis is adapted to the ratio of motor revolution and the resulting movement.
The value of setpitch is equivalent to the resulting movement of one motor revolution.

storable

$$\text{Pitch} = \frac{\text{unit of movement}}{\text{Motor revolution}}$$

Syntax:

[Pitch] [Axis] *setpitch*

	Range	Unit
[Pitch]	0.0001 bis 4095	mm
[Axis]	1, 2, 3	

Related command:

getpitch

Example:

4.0009 1 setpitch

One motor rev. of Axis-1 results in a movement of 4.009mm

More examples in the following page.

Examples::

Drive mechanism with ball screw at axis-1 :

Ball screw pitch = 2mm
One motor rev. follows a movement of 2mm

Pitch = 2mm / 1 rev. = 2

Venus-1 command : *2 1 setpitch*

Dirive mechanism with ball screw and gear at axis-3

Ball screw pitch = 4mm
Gear = 120:1
(120 motor rev. gives 1 rev. after the gear = 4mm)

Pitch = 4mm / 120 rev. = 0.0333

Venus-1 command : *0.033 3 setpitch*

Rotation table at axis-2, unit = degrees

Pitch = 360° / 1 rev. = 360

Venus-1 command : *360 2 setpitch*

Rotation table with gear 120:1 at axis-2, unit = degress

120 motor rev. give 360°.

Pitch = 360° / 120 Motorumdrehungen = 3

Venus-1 command : *360 2 setpitch*

getpitch

Description:

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

Syntax:

[Axis] *getpitch*

	Range
[Axis]	-1, 1, 2, 3

Reply:

[value]

	Unit
[value]	mm

Example:

2 getpitch   *-1 getpitch*

Reply: Reply:
4.000900   4.000900
                    2.000000
                    2.000000

setsw

Description:

The command *setsw* adapts the specified endswitch inputs to the switch type of the cal/rm-endswitches.
Additionally the inputs can set to be ignored

Following settings are possible:
• Opener
• Closer
• Endswitch ignored

If an endswitch input is set to be ignored it can not accomplish a safety function.

Syntax:

[Function] [Endswitch] [Axis] *setsw*

NPN Type is switching to GND
PNP Type is switching to VCC

[Function]	Description NPN	Description PNP
0	Closer to GND	Opener to VCC
1	Opener to GND	Closer to VCC
2	Ignore	Ignore

[Endswitch]	Description
0	cal-input
1	rm-input

Examples:

0 0 1 setsw
2 1 2 setsw

getsw

Description:

The command *getsw* replies the setting of the endswitch in-puts.

Syntax:

[Axis] *getsw*

	Range
[Axis]	-1, 1, 2, 3

Reply:

[Function cal-input] [Function rm-input]

Example:

3 getsw   *-1 getsw*

Reply: Reply:
0 0   0 0 0 1 2 2

                                         [Axis-1] [Axis-2] [Axis-3]
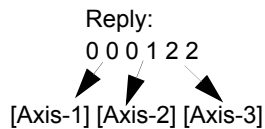
setclperiod

Description:

With command *setclperiod* the axis is adapted to closed-loop mode if incremental linear encoders are used.

The value for *setclperiod* is the resulting movement distance within one signal periode of the encoder.
With the algebraic sign -/+ it is possible to adapt the count direction to the motor direction.

The transmission ratio caused from the spindle or gear has to be considered in some cases to calculate the *setclperiode* value (see examples).

With rotary encoders the same setting is performed with the command *setclfactor.*

To support analog encoders Corvus must be equipped with the so called SIN/COS Module for each closed-loop axis.
To adapt this module to different encoder types the Venus-1 command *setscaleinterface* is used.

Syntax:

[Count direction] [Distance / periode] [Axis] *setclperiod*

	Range	Unit
[Direction]	+ / -	
[Distance]	0.0000001-1.999999	mm
[Axis]	1, 2, 3	

Related command:

getclperiod, setscaleinterface

Examples:

- 0.002 3 setclperiod

Example: Linear stage with linear encoder

Linear stage with following specifications:
Linear encoder with signal periode = 20µm
Pitch = 10mm

setclperiod = 0.020mm

The value of *setclperiod* depends **not** on the pitch
because the encoder is directly in contact with the movement
of the stage.

If this construction is equipped with a gear 5:1, the spindle
pitch must be set to 10/5 = 2mm.
The value for *setclperiod* doesn´t vary because the resul-
ting movement distance for each signal periode is the same.

Example: Linear stage with rotary encoder

Specifications:
Rotary encoder with sinusoidal outputs,
1000 signal periods / rev., mounted on the motor shaft.
Stage pitch = 10mm

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

setclperiod = 10 mm / 1000 periodes = 0.01

getclperiod

Description:

Command *getclperiod* replies the setting of *setclperiode*.

Syntax:

[Axis] *getclperiod*

Parameter	Range
[Axis]	1, 2, 3

Reply:

[Value]

	Range	Unit
[Value]	0.0000001-1.999999	mm

Example:

1 getclperiod

Reply:

0.000000100

setcloop

Description:

Command *setcloop* enables closed-loop mode.

storable

For proper closed-loop settings see also following commands:
setscaletype, setscaleinterface, setclperiod, setclpara, setnselpos

Syntax:

[Index] [Axis] *setcloop*

	Range
[Index]	0, 1
[Axis]	1, 2, 3

[Index]	Description
0	Closed-Loop disabled
1	Closed-Loop enabled

Related command:

getcloop

Example:

1 2 setcloop
0 3 setcloop

Axis-1 closed-loop mode enabled
Axis-3 closed-loop mode disabled

getcloop

Description:

Command *getcloop* replies the closed-loop status of the controller.

Syntax

[Axis] *getcloop*

	Range
[Axis]	-1, 1, 2, 3

Reply:

[Index]

	Range
[Index]	0, 1

Example:

1 getcloop

Reply:
1

setclfactor

Description:

With command *setclfactor* the axis is adapted to closed-loop mode if a digital rotary encoder with RS-422 Output signals is used.
The value for *setclfactor* is equivalent to the line counts of the encoder.
With the algebraic sign -/+ it is possible to adapt the count direction to the motor direction.

Configuration with *setclfactor* only if
setscaletype = 0

Syntax:

[Count direction] [Line counts] [Axis] *setclfactor*

Parameter	Range	Unit
+ / -		
Line counts	1-50000	Imp/Rev
Axis	1, 2, 3	

Related command:

getclfactor, setscaletype

Example:

- 500 3 setclfactor

getclfactor

Description:

Command *getclfactor* replies the setting of *setclfactor*

Syntax:

[Axis] *getclfactor*

Parameter	Range
Axis	-1, 1, 2, 3

Reply:

Value

	Range	Unit
Value	0-50000	Line count / rev

Example:

1 getclfactor

Reply:

500

Velocity and acceleration

setvel (sv)

Description:

The command **setvel** defines the velocity **v**x with which the controller shall execute a programmed positioning.
The programmed movement starts all axes simultaneously
They also will reach the target positions at the same time.

The value of **setvel** relates therefore to the axis which must travel the longest distance (see diagram).
The maximum velocitiy (**V**y) of the other axes depends on it´s ratio to the longest travel axis.

$$v_y = \frac{s_y}{s_x} \cdot v_x$$
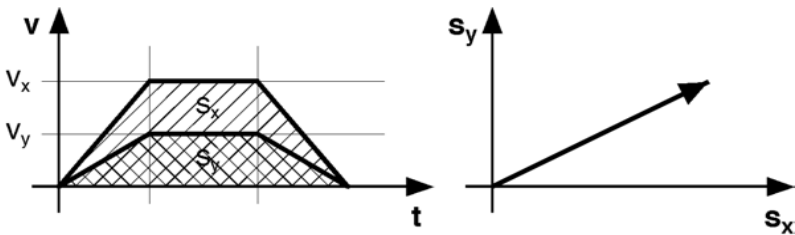
Syntax:

[Velocity] **setvel**

	Range
minimum velocity	15,26 nm/s
maximum velocity	180mm /s = 45 rev/s pitch =4mm

	Unit
[velocity]	selected unit

Other velocitiy commands:

setcalvel / setrefvel / setjoystickspeed

Example:

100 sv

getvel (gv)

Description:

The command *getvel (gv)* replies the setting of *setvel*.

Syntax:

getvel

Reply:

[velocity]

	Unit
[velocity]	selected unit

Example:

gv

Reply:
180.000000

setaccel (sa)

Description:

The command *setaccel (sa)* defines the acceleration with which the controller shall execute a programmed positioning. For a programmed positioning all enabled axes are starting simultaneously and reach it´s position at the same time. The value of *setaccel* relates therefore to the axis which must travel the longest distance (see diagram).

The maximum acceleration of the other axes depends on it´s ratio to the longest travel axis.

Acceleration and deceleration ramp are identical.

Syntax:

[Acceleration] *setaccel*

	Range with unit = mm	Unit
[Acceleration]	0 - 2400	unit/s²

Related commands:

getaccel / setmanaccel

Example:

500 sa

getaccel (ga)

Description:

The command *getaccel (ga)* replies the setting of *setaccel.*

Syntax:

getaccel

Reply:

[Acceleration]

	Range with unit = mm	unit
[Acceleration]	0 - 2400	unit/s²

Example:

ga

Reply:
2400000.000000 (if unit = µm)

setmanaccel

Description:

The command *setmanaccel* defines the acceleration of
the axis for the manual operation mode with Joystick
or handwheel.
The unit depends on the unit setting of the 0-axis.

Syntax:

[Acceleration] *setmanaccel*

	Range	Unit
[Acceleration]		unit 0-axis /s²

Related commands

getmanaccel / setaccel

Example:

100 setmanaccel

getmanaccel

Description:

The command *getmanaccel* replies the setting of *setmanaccel.*

Syntax:

getmanaccel

Reply:

[Value]

	Range	Unit
[Value]	0 - 2400	mm/s²

Example:

getmanaccel

Reply:
2400.000000

Positioning commands

move (m)

Description:

The command **move** positions all active axes to the given coordinates.
In consideration of velocitiy, acceleration and the given hard or software range limits, the controller calculates the moving profile of all axes.

The programmed movement starts all axes simultaneously and also they will reach the target positions at the same time.

The origin is either defined by the cal-endswitch movement or by the command **setpos**.

The command **status** replies the actual state of the move.

With the command **Ctrl+c** or **abort** it is possible terminate the move.

Syntax:

[Axis-1] [Axis-2] [Axis-3] **move**

	Range if unit = m m	unit
[Axis -1]	+/- 16383	unit
[Axis -2]	+/- 16383	unit
[Axis -3]	+/- 16383	unit

The number of expected coordinates depends on the setting of the command *setdim*.

setdim	Expected nummer of axis coordinates
1 setdim	Axis-1
2 setdim	Axis-1_Axis-2
3 setdim	Axis-1_Axis-2_Axis-3

Related commands:

rmove, speed

Examples:

Dimension = 3
12.5 20 0.0001 m
Absolute positioning of all three Axes.

Dimension = 1
12.5 m
Absolute positioning of Axis-1

Dimension = 2
12.5 20 m
Absolute positioning of Axis-1 and Axis-2

rmove (r)

Description:

The command *rmove* positions all active axes relative to the current coordinates.

In consideration of velocitiy, acceleration and the given hard or software range limits, the controller calculates the moving profile of all axes.

The programmed movement starts all axes simultaneously and will also reach the target positions at the same time.

The command *status* replies the actual state of the move.

With the command *Ctrl+c* or *abort* it is possible abort the move.

Syntax:

[Axis-1] [Axis-2] [Axis-3] *rmove*

Parameter	Wertebereich bei unit = m m	Einheit
Achse -1	+/- 16383	unit
Achse -2	+/- 16383	unit
Achse -3	+/- 16383	unit

The number of expected coordinates depends on the setting of the command *setdim*.

setdim	Expected nummer of axis coordinates
1 setdim	Axis-1
2 setdim	Axis-1_Axis-2
3 setdim	Axis-1_Axis-2_Axis-3

Related commands:

move, speed

Examples:

Dimension = 3
0.5 20 0.0001 r
Relative positioning of Axis-1, Axis-2, Axis-3

Dimension = 1
12.5 rmove
Relative positioning of Axis-1

Dimension = 2
12.5 20 r
Relative positioning of Axis-1 and Axis-2

calibrate (cal)

Description:

The command ***cal*** triggers the limit switch movement to the cal-endswitches. Doing this, all axes are simultaniously positioned in negative direction until the cal-switches are active. After that the controller moves in direction to positive positioning values as long as the endswitches are again not active. At this coordinates the position of all axis is set to zero (depends on ***setaxis***) and temporary stored.
After that it is not possible to move the axes to positioning values less than zero.
If the controller is switched off, the stored limits and the origin are lost.

With the command ***setcalswdist*** a additionally distance to the endswitches can be defined.

The command ***setaxis*** has an effect for the limits (***getlimit***) and the actual position (***pos)***.

With ***Ctrl+c*** the cal-endswitch movement is immediatelly aborted and the origin and limit is set at this point.

The origin and the limits are not stored. After power off. This values must defined again.
The command ***cal*** is a blocking command. Thus no further commands can be processed until the cal procedure is finished.
The proceeding to generate an automatic status reply after completion of the limit switch movement, is described in then Venus-1 introduction.

Syntax:

calibrate or ***cal***

Example:

cal

rangemeasure (rm)

Description:

The command *rm* triggers the limit switch movement to the rm-endswitches. Doing this, the active axes are simultaniously positioned in positive direction until all rm-switches are active.
After that the controller moves all axes in direction to negative positioning values as long as the endswitches are again not active.
At this coordintates the upper range limits are temporary stored (depends on *setaxis*).
After that it is not possible to move the axes beyond the upper limits. If the controller is switched off, the stored limits are lost. With the command *setcalswdist* a additionally distance to the endswitches can be defined.

The command *setaxis* influences the effect to the limits (*getlimit*) and the actual position (*pos*).

With *Ctrl+c* the rm-endswitch movement is immediatelly aborted and the upper limits are set at this point.

For a accurate determination of the maximum moving range of all axes, the cal-endswitch movement has to be performed first.
The origin and the limits are not stored. After power off this values must defined again.
The command *rm* is a blocking command. Thus no further commands can be processed until the cal procedure is finished.
The proceeding to generate an automatic status reply after completion of the limit switch movement, is described in then Venus-1 introduction

Syntax:

rangemeasure or *rm*

Example:

rm

Abortion commands

Ctrl+c

The command *Ctrl+c* aborts the actual executed command. All moves will be stopped immediatelly with the current acceleration settings. The data input FIFO will not be cleared.

Ctrl+c has **not** to pass the FIFO and could not be delayed with a blocking command.

Ctrl+c can also be used to abort the cal- and rm-endswitch procedure to determine the lower und upper range limit.

It is forbidden to use the command in a fast sequence of stopping and starting movements.
For this procedure we recommend the commands
abort and *startspeed / stopspeed*

Syntax:

Ctrl+c (ASCII 3)

Related command:

abort

Example:

Ctrl+c

abort

Description:

The command *abort* cancels the momentary executed command.
All moves will be stopped immediatelly with the current acceleration settings. The data input FIFO will not be cleared.

Ctrl+c has to pass the FIFO and can be delayed with a blocking command.

Example: *abort* is blocked with the command *ge*

1. 100 0 0 move (momentary executed)
2. ge
3. abort

Syntax:

abort

Related command:

Ctrl+c

Example:

abort

Workspace and point of origin

setpos

Description:

The command *setpos* defines the point of origin of all ena-
bled axes.
This point could be set at arbitrary points within the limits.
If the coordinates of the limits are defined before, they will be
recalculated with the new point of origin.
For special cases the zero point can be defined with an rela-
tive offset.

Syntax:

[Axis-1] [Axis-2] [Axis-3] *setpos*

	Range if units = mm	unit
[Axis-1]	+/-16383	unit
[Axis-2]	+/-16383	unit
[Axis-2]	+/-16383	unit

Example:

0 0 0 setpos
The current coordinate is defined as the origin.

10 10 10 setpos   / unit = mm
The current coordinate is defined as the origin with an relative
offset 10 mm each axis.
The command *pos* will reply the position value -10 -10 -10 if
the previous coordinate was 0 0 0.

setlimit

Description:

The command ***setlimit*** defines the so-called software limits of the controller.
The hardware limits have to be determined before by the commands ***cal*** and ***rm***

If a limit is determined the controller is not able to move beyond it. Alle movements are slowed down and will stop exactly on the limit border.

Conditions for setting software limits:

• The maximum working range has to be defined with the commands ***cal*** and ***rm*** before setting the soft limits.

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

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

Syntax:

[-A1] [-A2] [-A3] [A1+] [A2+] [A3+] *setlimit*

The number of axis parameters depends on the setting of *setdim*

	Description
[-A1]	lower Limit Axis-1
[-A2]	lower Limit Axis-2
[-A3]	lower Limit Axis-3

	Description
[A1+]	upper Limit Axis-1
[A2+]	upper Limit Axis-2
[A3+]	upper Limit Axis-3

	Range	Unit
[-A1] [-A2] [-A3]	-16383	unit
[A1+] [A2+] [A3+]	+16383	unit

Related command:

getlimit

Example:

getdim = 3

0 0 0 12 25 30 setlimit

This command will result in following limits
Axis-1: 0 to 12
Axis-2: 0 to 25
Axis-3: 0 to 30

getlimit

The command *getlimit* replies the limits of the current working range of all axis.
Dependent on the setting of *setdim* the controller replies the limits of 1, 2 or 3 axes.
If a limit is not determined the following values are replied:

16383.000000   16383.000000

l

Syntax:

getlimit

Reply:

		Unit	Axis
[lower Limit]	[upper Limit]	unit	1
[lower Limit]	[upper Limit]	unit	2
[lower Limit]	[upper Limit]	unit	3

* Depending on the setting of *setdim*

Example:

getdim = 3
getlimit   (unit = mm)

0.000000   7.723750
0.000000   7.723750
-16383.000000   16383.000000

Current status inquiries

geterror (ge)

Description:

With the command *getnerror* the axis can be checked for general errors. The last occurred error of the axis is always indicated.
The error message is deleted after the command is executed.
The occurrence of these errors is **not** shown in the *status*. reply.

Syntax:

geterror

Reply:

[Error code]

Error codes	Description
1....4	Internal error
1001	Wrong parameter
1002	Not enough parameter on the stack
1003	Range of parameter is exceeded
1004	Move stopped working range should run over
1008	Not enough parameter on the stack (same as 1003)
1009	Not enough space on the stack
1010	Not enough space on parameter memory
1015	Parameters outside the working range
2000	Unknown command

Example:

ge

status (st)

Description:

With the command *status* the momentarily condition of the controller is asked.
The replied value reflects the operating state of the controller in a binary coded decimal representation.

To decode the states correctly it is neccessary to use a bit-mask.

Binary	Decimal	Function
D0	1	Status command execution
D1	2	Status Joystick or Handwheel
D2	4	Status Button A
D3	8	Machine error
D4	16	Status speed mode
D5	32	Status In-Window
D6	64	Status setinfunc
D7	128	Status motor enable, safety device

D0:

0	Interpreter is ready to execute a command
1	Interpreter is busy

D1: Command: *joystick*

0	Manual mode not active
1	Manual mode is active

D2:Switch A at the front panel

0	Button A not pressed
1	Button A pressed

D3:

0	no function
1	no function

D4: Command _speed_

0	Speed mode acitve.
1	Speed mode not active

D5: Command _setclwindow_

0	Position out of the target window.
1	Position within the target window.

D6: Command _setinfunc_

0	no input signal from external device
1	input signal from external device detected

D7:

0	Motordriver enabled
1	Motordriver disabled from external device

Syntax:

status or _st_

Reply:

[bitcodet decimal value]

Example:

status
Reply:
 2
Interpreter is idle, manual mode active

More examples on the following page.

Command execution

Manual mode

Button A

Speed mode

in-window

setinfunc mode

Motor driver status
Overcurrent
or motor disabled
from external device

Binary coded status reply

	1	2	4	8	16	32	64	128	
	D0	D1	D2	D3	D4	D5	D6	D7	
	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	2
	0	1	0	0	0	0	1	0	66
	0	1	1	0	0	0	0	0	6
	0	1	1	0	0	0	0	1	134
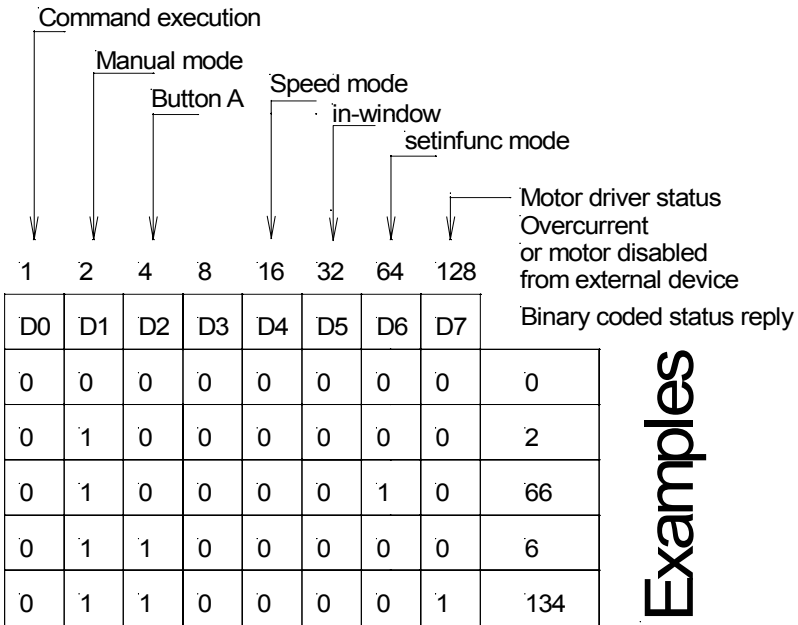
Examples

pos (p)

Description:

The instruction *pos* returns the current position of the axes.
The value relates to the origin which is set with the command
cal or *setpos*.
The number of replied axis coordinates depends on the set-
ting of *setdim*.

Alternatively the actual position from an external measure-
ment system or the calculated position is supplied (see com-
mand *setselpos*).

Syntax:

pos or *p*

Reply:

[Pos Axis-1] [Pos Axis-2] [Pos Axis-3]

	Description	unit
[Pos Axis-1]	Position of Axis-1	unit
[Pos Axis-2]	Position of Axis-2	unit
[Pos Axis-3]	Position of Axis-3	unit

.

Example:

getdim = 2

pos

Reply:

1.00000   19.00000

identify

Description:

The command *identify* replies following hardware informations of the controller.

- Controller Type
- Firmware Revision
- Hardware Revision
- On Board Switch
- Hex coded Jumper setting

Syntax:

identify

Reply:

[Model] [HW-Rev]   [SW-Rev] [Board-SW]   [JA4/5]

Value	Description
[Modell]	Controller Type
[HW-Rev]	Hardware Revision
[SW-Rev]	Software Revison
[Board-SW]	On Board Switch
[JA-4/5]	Hex coded Jumper setting of JA4 and JA5

Related command:

version

Example:

identify

Reply:

Corvus 1 312 1 10F

version

Description:

The command *version* replies the version number of the internal firmware of the controller.

Syntax:

version

Reply:

[Version number]

Example:

version

Reply:

3.23

getswst

Description:

The command *getswst* reads back the current activitiy of the endswitch inputs.

Syntax:

[Axis] *getswst*

	Range
[Axis]	-1, 1, 2, 3

Reply:

[cal-Input]] [rm-Input]

cal-Input = 0	cal-endswitch not switched
cal-Input = 1	cal-endswitch switched

rm-Input = 0	rm-endswitch not switched
rm-Input = 1	rm-endswitch switched

Example:

3 getswst   *-1 getswst*
Reply: Reply:

0 0   0 0 1 0 0 0
Axis-3   All axes:
Nno endswitch switched   cal-endswitch of Axis-2
                                         is switched

Input / Output Functions

getin

Description:

The command *getin* replies the current status of the digital Inputs Din-1, Din-2, Din-3. The replied value is bit coded.
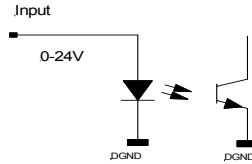
Syntax:

getin

Reply:

[bit coded value]

	Range
[bit code]	0-7

Example:

getin

[bit code]	Input Din-1	Input Din-2	Input Din-3
0*	0	0	0
1**	1	0	0
2	0	1	0
4	0	0	1

*0 : Input voltage 0-2V

**1: Input voltage 3-24V

setout

Description:

Command *setout* is switching the digital output signals Dout1 - Dout3.
The outputs are open collector circuits. In ON state the output transistor is switched to DGND.
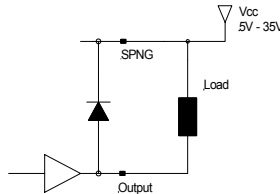
Syntax:

[value] *setout*

Value	Dout 1 (9)	Dout 2 (4)	Dout 3 (8)
0	OFF	OFF	OFF
1	ON	OFF	OFF
2	OFF	ON	OFF
3	ON	ON	OFF
4	OFF	OFF	ON
5	ON	OFF	ON
6	OFF	ON	ON
7	ON	ON	ON

Related command:

getout

Example:

1 setout

getout

Description:

Command *getout* replies the output value which is set with *setout*

Syntax:

getout

Reply:

[value]

Range: 0 - 7

Example:

getout

outtrig (ot)

Description:

Command *ot* generates a trigger output signal via a specified I/O interface output.

Syntax:

[time] [pol] [output] *ot*

	Description
[time]	Trigger out pulse width
[pol]	Trigger out polarity
[output]	I/O-Interface output

	Range	Unit	Function
[time]	0-1000	ms	
[pol]	0, 1		
[output]	1, 2, 3		Equivalent I/O interface output

Example:

100   1   1   *ot*

I/O interface output

Trigger out pulse polarity

Trigger out pulse width
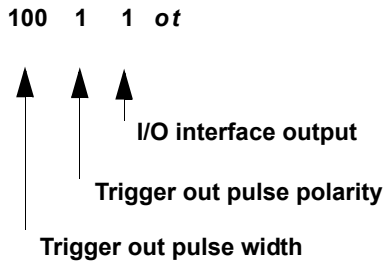
getticks (gt)

Description:

Command *gt* (**g**et_**t**icks) replies the number of processor cycles (250 μs) since the controllers has started.

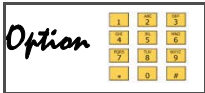After 298 hours this counter will overflow and start with zero.

.

Syntax:

gt

Example:

gt

Replies: 10922835

waitposot (wpot)

Description:

Command **wpot** (**w**ait_**p**os_**o**ut_**t**rigger) configures the position synchronized output function.

This function enables the possibility to perform Trigger output signals at arbitrary stage positions within the main controller Maximum output frequency is 4Khz.

.

Syntax:

[pos] [dir] [axis] [time] [pol out] [output] **wpot**

	Description
[pos]	Desired Trigger output position
[dir]	Move direction where the trigger out is enabled
[axis]	Specified axis
[time]	Trigger output pulse width
[pol_out]	Trigger out polarity, active low, active high
[output]	I/O Interface output

	Range	Unit	Function
[pos]	+/-16383	unit	
[dir]	0, 1		0 = negative direction 1 = positive direction
[axis]	1, 2, 3		Equivalent controller axes
[time]	0-1000	ms	
[pol_out]	0, 1		0 = active low 1 = active high
[ouput]	1, 2, 3		Equivalent I/O interface outputs

Example:

unit = 2 (mm)

**12.54   1   1   10   0   1   *wpot*

I/O interface output

Trigger out pulse polarity

Trigger out pulse width

Desired trigger axis

Move direction of trigger axis

Trigger out position of desired axis
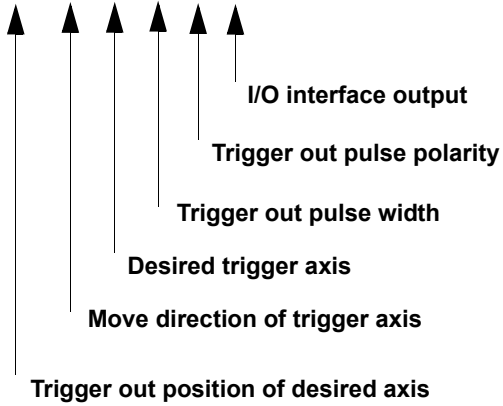
waitintrigot (witot)

Command *witot* (wait_in_trigger out trigger) is a fast combination of the commands *waitintrig* and *outtrig*.

Syntax:

[pol_in] [input] [time] [pol_out] [output] *witot*

	Description
[pol_in]	Trigger input polarity
[input]	I/O-interface input
[time]	Trigger output pulse width
[pol_out]	Trigger output pulse polarity
[output]	I/O-interface output

	Range	Unit	Function
[pol_in]	0, 1	unit	
[input]	1, 2, 3		Equivalent I/O interface input
[time]	0-1000	ms	
[pol_out]	0, 1		
[output]	1, 2, 3		Equivalent I/O interface output

Example:

0   1   10   1   1   *witot*

I/O interface output

Trigger output pulse polarity

Trigger output pulse width

I/O interface input

Trigger input polarity

waittimeot (wtot)

Description:

Option

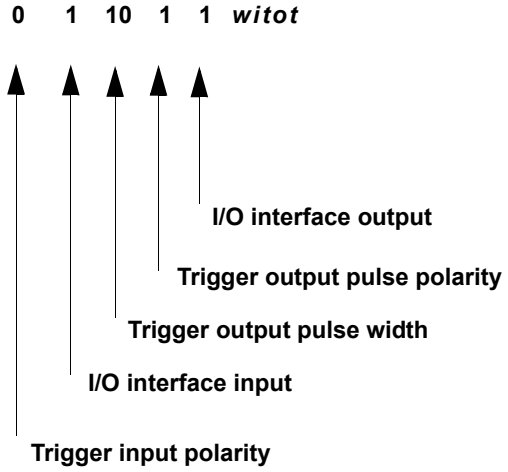Command *wtot* (wait_time out trigger) is a fast combination of the commands *waittime* and *outtrig*.
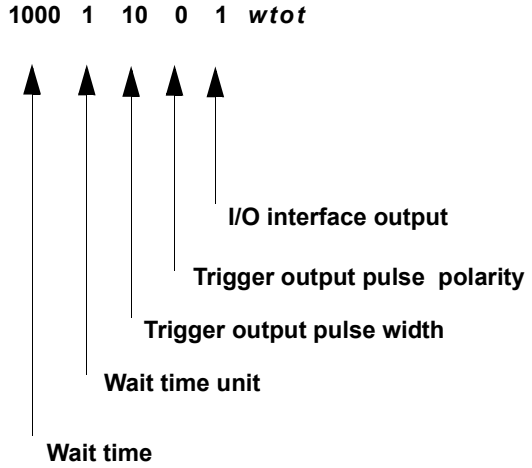
Syntax:

[time_wait] [wait_unit] [time_trigger] [pol_out] [output] *wtot*

	Description
[time_wait]	Waiting time
[wait_unit]	wait time unit
[time_trigger]	I/O-interface input
[pol_out]	Trigger output pulse width
[output]	I/O-interface output

	Range	Unit	Function
[time_wait]	Integer value	unit	
[wait_unit]	0, 1		0 = ticks (250µs each tick) 1 = seconds (s)
[time_trigger]	0-1000	ms	
[pol_out]	0, 1		
[output]	1, 2, 3		

Example:

1000   1   10   0   1   *wtot*

I/O interface output

Trigger output pulse polarity

Trigger output pulse width

Wait time unit

Wait time

waitintrig (wit)

Description:

Command *wit* (**w**ait_**i**n_**t**rigger) configures the controller to interrupt the command execution until a specified input signal is active (level triggered).
The command does not interrupt momentary executed commands.
The *wit* command allways awaits the end of a momentary move command, therefore it has no influence to the manual move.
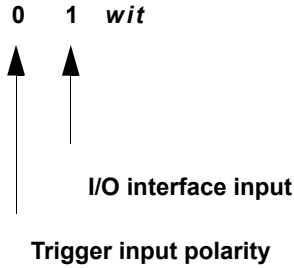
Syntax:

[pol_in] [input] *wit*

	Description
[pol_in]	Trigger input polarity
[input]	I/O-interface input

	Range	Unit	Function
[pol_in]	0, 1	unit	
[input]	1, 2, 3		Equivalent I/O interface input

Example:

<div align="center">

0   1   *wit*

</div>

I/O interface input

Trigger input polarity

Command sequence:

0 1 wit
ge

The *wit* command configures the controller to reply the *ge* command if trigger level is active.

waittime (wt)

Description:

Option

Command **wt** (**wait_time**) configures the controller to interrupt the command execution a specified time.
The command does not interrupt momentary executed commands.
The **wt** command disables also the manual move.

Syntax:

[time_wait] [wait_unit] **wt**

	Description
[time_wait]	Waiting time
[wait_unit]	waiting time unit

	Range	Unit	Function
[time_wait]	Integer value	unit	
[wait_unit]	0, 1		0 = ticks (250µs each tick) 1 = seconds (s)

Example:

<div align="center">

1000 1 *wt*

</div>

Wait time unit

Wait time
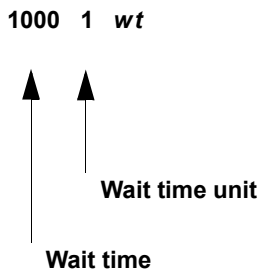
Command sequence:

1000 0 wt
ge

The *wt* command configures the controller to reply the *ge* command after 1000 ticks = 0.25s

1 1 wt
ge

The *wt* command configures the controller to reply the *ge* command after 1s

waitpos (wp)

Description:

Command *wp* (**w**ait_**p**os) configures the controller to inter-
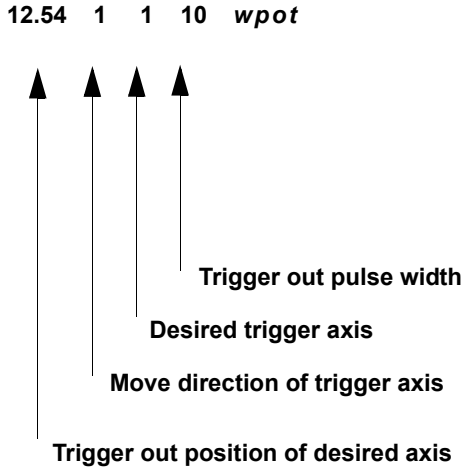rupt the command execution until a specified axis position is
reached.

.

Syntax:

[pos] [dir] [axis] *wp*

	Description
[pos]	Desired Trigger output position
[dir]	Move direction where position is reached
[axis]	Specified axis

	Range	Unit	Function
[pos]	+/-16383	unit	
[dir]	0, 1		0 = negativ direction 1 = positiv direction
[axis]	1, 2, 3		Equivalent controller axes

Example:

unit = 2 (mm)

12.54   1   1   10   *wpot*

Trigger out pulse width

Desired trigger axis

Move direction of trigger axis

Trigger out position of desired axis

unit = 2 (mm)
Command sequence:
100 10 m
12.54 1 1 10 wp
ge

The *wp* command configures the controller to reply the *ge* command if moving in positiv direction at position 12.54mm of Axis-1.

Joystick / Handwheel

joystick (j)

Description:

storable

The command **joystick** enables oder disables the manual mode.
The activity of this mode is indicated in status bit D1 and also displayed at the front panel of the controller.

Special function in Joystick mode

After power up the zero setting of the joystick is checked.
A tolerance of +/- 10% is acceptable.
With greater deviations the appropriate axis can not be enabled for the joystick mode.

Syntax:

[Index] **joystick**

	Range
[Index]	0, 1

	Function
0	Manual mode disabled
1	Manual mode enabled

Example:

1 j

Enables manual mode.

setjoyspeed (js)

Description:

storable

 The command *setjoyspeed* defines the maximum velocitiy with which an axis can be moved in the manual mode.

Syntax:

[velocity] *setjoyspeed*

	Unit
[velocity]	units of the 0-axis

	Range
min. velocity	15.62 nm/s
max. velocity	180 mm/s

Related command:

getjoyspeed

Example:

20 setjoyspeed

unit = mm

The joystick velocity is 20 mm/s

getjoyspeed

Description:

The command *getjoyspeed* reads the adjusted maximum velocitiy for manual moves.

Syntax:

getjoyspeed

Reply:

[velocity]

	Unit
[velocity]	unit 0-axis

Example:

getjoyspeed

Reply:

20.000000

setjoy<u>b</u>speed

storable

The command *setnjoybspeed* can fix a second joystick velocity which gets active by pressing the joystick switch.

Syntax:

[velocity] *setjoybspeed*

	Unit
[velocity]	units 0-axis

	Range
min. velocity	15.62 nm/s
max. velocity	180 mm/s

Related command:

getjoybspeed

Example:

0.01 setjoybspeed

unit = mm

As long as the joystick button is pressed, the maximum joystick velocity is 0.01 mm/s.

getjoy<u>b</u>speed

Description:

The command *getnjoybspeed* reads the configuration of the second joystick velocity.

Syntax:

getjoybspeed

Reply:

[velocity]

	Unit
[velocity]	unit 0-axis

Example:

getjoybspeed

Reply:

0.010000

System commands

save

Description:

The command *save* saves all active parameters in a non volatile memory.

Allways the last saved settings are reactivated after power on.

Programmable moves are aborted if the *save* command is executed. Manual moves are only interrupted during time of saving.

The end of saving is indicated in terminal mode with the characters OK

In host mode an automatic reply can be prepared with the command sequence *save* and *status.*

Parameters which can be saved are declared with following symbol in this handbook

storable

Syntax:

save

Reply:

In Terminal mode the characters *OK* are replied if the save is finished.

Example:

save

restore

Description:

The command *restore* reactivates the last saved parameters.
With the command sequence *restore save* the controller will be prepared to reply the status information after the restore is finished.

Syntax:

restore

Reply:

Reply can be indirectly controlled with the command sequence
restore status

Related command:

getfpara

Example:

restore

reset

Description:

The command *reset* performes an initialisation of the controller.
The effect is comparable with a power off / on situation.

During the *reset* the front panel LED´s will be a short time switched off.
The proper state of the controller is indicated with a beep.

Syntax:

reset

Example:

reset

clear

Description:

The command *clear* deletes the contents of the parameter stack.

The parameter stack can altogether accept 99 parameters. The execution deletes the data when the appropriate command uses them.
Normally there are only the parameters for the next command on the stack.

By means of error inputs it can happen that more data remains on the stack than Venus-1 commands use.
This can lead to an overflow of the stack.

With the command *gsp* the number of elements on the stack-can be read out.

Syntax:

clear

Related command:

gsp

Example:

clear

gsp

Description:

The command *g s p* reads out the number of elements on the stack.

Syntax:

g s p

Reply:

[Number]

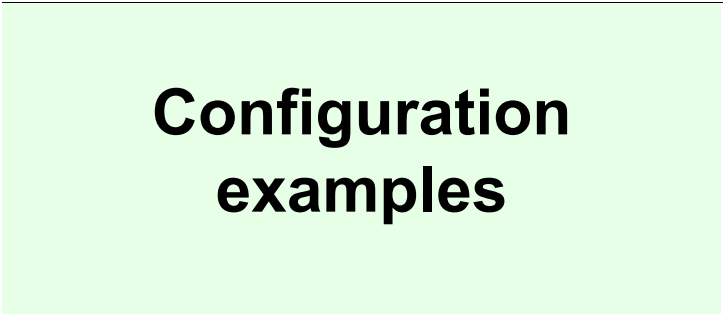|	Range
[Number]	0 -99

Related command:

clear

Example:

g s p

Reply:
2

Configuration examples

Introduction

For proper function it is essential to adapt the controller correctly to the connected devices.
The following chapter describes:

- **Pitch adaptation to the stage**
- **Usage of the command *setunit***

To configure the controller via RS-232 interface we recommend our Windows programming tool **WinPos**, that is included in the controller delivery.
Of course any other terminal program is also usable.

Adaptation to the stage

pitch settings

High resolution stages employ a precision ground ball screw spindle to translate the motor rotation to a linear motion. The resulting linear motion depends on the pitch of the screw.

For example: A motor driven spindle with a pitch of 4mm produces a linear motion of 4mm with each motor revolution. Thus the controller has to know with which spindle the table is equipped to calculate the motion distance correctly.

The following effects are indicated if the pitch setting is wrong:

- **The controller is moving a distance much longer than expected:**

- **Cause: Controller pitch setting undersized**

- **The controller is moving a distance much fewer than expected:**

- **Cause: Controller pitch setting oversized**

Procedure to check the pitch of the stage:

An easy procedure to evaluate the spindle pitch of a stage, is to rotate the motor one revolution and measuring the resulting linear distance from the starting point to the endpoint.

For example:
If one motor revolution cause a linear distance of 1mm then the spindle pitch is also 1mm.

Examples:

To configure the pitch setting the Venus-1 command
setpitch is used.
The unit of the pitch value depends on the setting of the controller axis unit.
Check these settings with the command: *getunit*

Example:

1 getunit
2 getunit
3 getunit

Given:
Two axes stage, equipped with a spindle (pitch = 2mm).
Connected at the controller motor output Axis-1 and Axis-2.
All axes units = mm

Configuration commands
2 1 setpitch
2 2 setpitch

Same example with axis unit = µm (1)

2000 1 setpitch
2000 2 setpitch

Given:
Three axes configuration, stage with focus drive.
Axis-1 and Axis-2 are connected to the stage.
Axis-3 is connected to the focus drive.
Stage spindle pitch = 4mm
The Focus drive spindle pitch = 0.5mm
All axes units = mm

Configuration commands:
4 1 setpitch
4 2 setpitch
0.5 3 setpitch

Units

The controller is able to deal with different physical units.
The assignment is made via the command *setunit*.
It is necessary to distinguish between three command groups
to understand the influence of the setunit command.

- **Global units, valid for all axes**
- **Axis specific units, assigned to an axis**
- **Unchangeable unit, fixed to unit: round/second.**

Within the command *setunit* the first two groups are
adressed with an axis index.
The unchangeable unit **round/second** is directly assigned to
the commands *setcalvel, setrmvel, setrefvel*.

The following table shows the interrelationship of the axis in-
dex, unit groups and influenced commands.

Axis index	Unit groupe	Influenced commands
0	Global units (Valid for all axes)	setvel, setaccel, setmanaccel
1, 2, 3	Axis specific units (Directly assigned to an axis)	move, rmove, pos setpos, setpitch,get-limit, setlimit setcl-period, setcalswdist,

Examples:

Given:

Global unit = µm
Axis specific unit, Axis-1 = mm
Axis specific unit, Axis-2 = µm

Commands:
1 0 setunit
2 1 setunit
1 2 setunit

Desired settings:

Velocity: 20mm/s,
Acceleration: 100mm/s²
Manual acceleration: 50mm/s²

Commands:
20000 setvelocity
100000 setacceleration
50000 setmanaccel

Command to move Axis-1, 100mm relative
100 1 rmove

Command to move Axis-2, 100mm relative
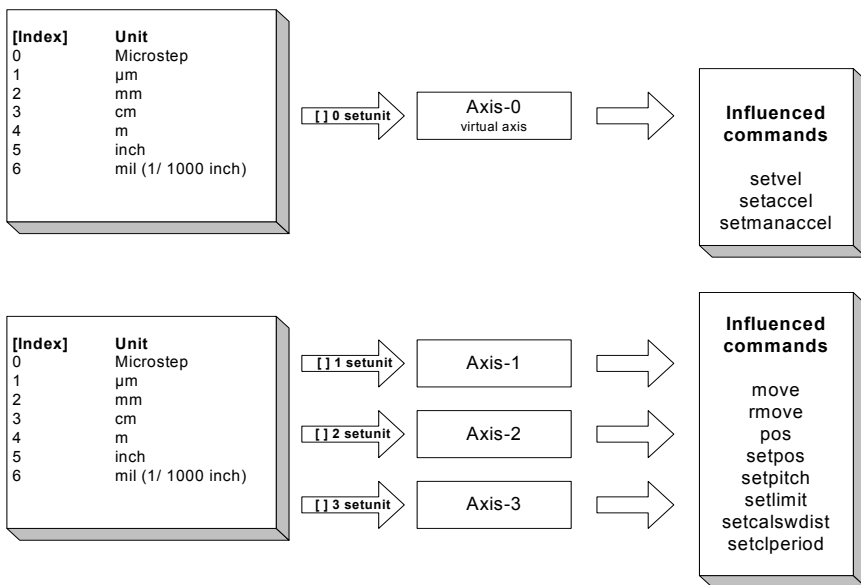*100000 2 rmove*

Command to reply position:

pos

Replies:
100 100000

The following diagram demonstrates the mode of operation of the global units and the axis specific units.

[Index]	Unit
0	Microstep
1	μm
2	mm
3	cm
4	m
5	inch
6	mil (1/ 1000 inch)

[ ] 0 setunit → Axis-0 (virtual axis) →

Influenced commands

setvel
setaccel
setmanaccel

[Index]	Unit
0	Microstep
1	μm
2	mm
3	cm
4	m
5	inch
6	mil (1/ 1000 inch)

[ ] 1 setunit → Axis-1 →

[ ] 2 setunit → Axis-2 →

[ ] 3 setunit → Axis-3 →

Influenced commands

move
rmove
pos
setpos
setpitch
setlimit
setcalswdist
setclperiod

Fixed unit commands

The commands *setcalvel, setrmvel*
and *setrefvel* are using the unit round/second.
This unit assignment unchangeable.
The resulting speeds of the calibration move
range measure move and refmove are depending on the set-
ting of the virtual 0-Axis pitch. See diagram.
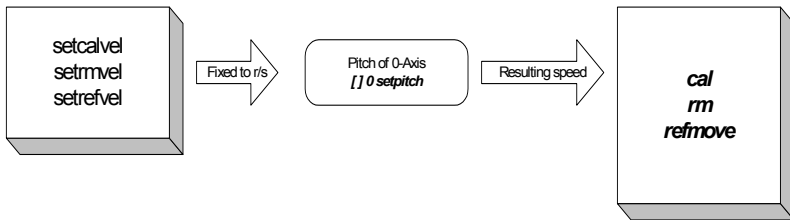
Examples

Given:

2 0 setpitch   // 2mm
5 1 setcalvel   // 5mm
0.5 2 setcalvel // 0.5mm

Resulting cal move velocities = 0-Axis pitch x setcalvel

= 10 mm/s to endswitch
= 1mm out of the endswitch