Power-Supply	60 W
Model	PSAA-60 W
Input	100-240 VAC
Output	24 VDC, 2.5A
Baudrate	19200
Version	Type I Drive
Driver	☑ 1.2A
Serial Nr. (Controller)	1005-1411 – 1005-1420
Version (Firmware)	4.20
Axisnumber (address)	1
Configured for	VT-21
max.speed	6.5mm /s

Actual settings:

Settings for Axis.: #% Current axis-No.: 1
Serialnumber..... : 10051411
Version.....: 4 2 0
Options.....: 0
Status..... : 0
Position..... : 13.000000
Switch-Status.....: 0 0
Limits.....: 0.000000 26.744102
Error.....: 0
Stack.....: 0
1 % setaxis
0.5000 % setpitch
5.000000 % snv
100.000 % sna
200.000 % setnstopdecel
3.000000 1 % setncalvel
0.100000 2 % setncalvel
3.000000 1 % setnrmvel
0.100000 2 % setnrmvel
0.000000 % setncalswdist
6500 % setumotmin
45 % setumotgrad
1 0 % setsw
1 1 % setsw
0 % setmotiondir
-1000.000000 1000.000000 % setinilimit

Shortform SMC *pollux / SMC pollux NT*

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.

Command syntax

The commands are assembled following this scheme:
[parameter] _ [axis index] _ [command] _ blank, (space) or (SP) blank

Command ending character while transmitting

The Venus- 2 command must be terminated with a blank (SP). [Parameter] SP [Axes index] SP [Venus- 2 command] SP

Command ending character while receiving

[1st parameter] SP [2nd parameter] SP [n- parameter] CR LF
Data which are delivered by the controller are always terminated with ASCII (CR) and (LF).

RS-232 Interface Configuration

Data bits	8
Stop bits	1
Parity	no
Handshake	no
Baudrate	19200

Error numbers:

1.... 4	Internal error
1001	Wrong parameter type
1002	parameter stack underrun
1003	parameter out of range
1004	Movement range should be exceeded caused by limit switch
1008	See 1002
1010	RS-232 input buffer lacking space (<30 para left)
1015	Parameter out of the movement area (softlimit)
1100	Both Limit Switches are active
2000	Unknown command

First steps:

Normaly the controller is configured for the delived stage. If delivery contains more then one controller, the controllers are labeled with an axisnumber (the address of the controller)

The actual settings are documented in a *.txt file, which is downloadable with our demo-application SMC-Pollux xxx.exe, also documented as pdf-file.

For the first step, hyperterminal, any other terminal-program, or the program smc-pollux xxx.exe could be a good choice.

Due to the fact that the controller is ready configured, the main commands for customers use are the basic move commands, homing and position and status query:

If a communication is established just type following commands: (axis 1 must be connected)

```
1 np          ; controller returns the actual position, after reset always 0.00000
1 ncal        ; axis searches the limit reverse the release point is the physical zero-position
              of the system
2.0 1 nr      ; axis moves 2 unit relative (usually 2 mm or 2 degree)
1 np          ; returns the actual position, now 2.00000
4.0 1 nm      ; axis moves to absolute 4.0 units
1 nst         ; if stage is just moving return value =1
              ; if stage not moving, means at target, return value = 0
```

Multiple Axis Application:

Please verify that all daisy-chained controller do have different axis-addresses. If no address labled you should assume that the address is **'1'.**

Assigning new controller address:

Assigning a new controller address is very easy, connect **only one** controller to the RS-232, connect with program *SMC-pollux xxx.exe* and change the address with dialog *Controller|Assign Address*. After this save the new settings with Button *'nsave'* .

Command-Overview:

Command	Description	Parameters	R/W	Range	Example
nrmove (nr)	move relative, without query status	*relpos axisid*	w	-1000.0 ..+1000.0	1.0 1 nr
nmove (nm)	move absolute without query status	*abspos axisid*	w	-1000.0 ..+1000.0	5.1 1 nm
npos (np)	returns actual position	*axisid*	r		1 np
npush	loads targets on stack (for synchronized start)	*position axisid*	w	-1000.0 ..+1000.0	10.0 1 npush
npop	removes values from stack	*axisid*	w		1 npop
setnpos	redefines the actual position	*abspos axisid*	w	-1000.0 ..+1000.0	0.0 1 setnpos
nstatus (nst)	returns actual status	*axisid*	r		1 getaxis
getnerror (gne)	returns actual error number	*axisid*	r		1 gne
getmerror (gme)	returns machine error number	*axisid*	r		1 gme
nabort	stops a move	*axisid*	w		1 nabort
<CtrlC>	stops move of all connected axes		w		<CTRL-C> hex 3
speed	starts a constant velocity move	*+/-speed axisid*	w		2.5 1 speed
stopspeed	stops constant velocity move	*axisid*	w		1 stopspeed
setnpowerup	defines startup behaviour	*value axisid*	w		0 1 setnpowerup
getnpowerup	returns startup behaviour	*axisid*			1 getnpowerup
ncal	homing (search limit reverse)	*axisid*	w		1 ncal
nrm	rangemeasure (search limit forward)	*axisid*	w		1 nrm
nversion	returns the firmware-version	*axisid*	r		1 nversion
nidentify	returns the controller identification	*axisid*	r		1 nidentify
getnserialno getserialno	returns the serial-number	*axisid*	r		1 getnserialno 1 getserialno
getnoptions	returns the options-code	*axisid*	r		1 getnoptions
getaxis	returns the wether axis is active or not	*axisid*	r		1 getaxis
setaxis	defines if axis active or not	*status axisid*	w	0..2	0 1 setaxis
getswst	returns the status of limit-inputs	*axisid*	r		1 getswst
getsw	returns the setting of limit-inputs	*axisid*	r		1 getsw
setsw	defines the limit-switch-status	*status 0 axisid* *status 1 axisid*	w	0..2	1 0 1 setsw 1 1 1 setsw
getmotiondir	returns the setting of the direction of motion	*axisid*	r		1 getmotiondir
setmotiondir	defines the direction of motion	*value axisid*	w	0..1	0 1 setmotiondir
getncalswdist	returns the calswitch-distance	*axisid*	r		1 getncalswdist
setncalswdist	defines the calswitch-distance	*distance axisid*	w	0..1.0	0.5 1 setncalswdist
getpitch	returns the pitch of the stage	*axisid*	r		1 getpitch
setpitch	defines the pitch of the stage	*pitch axisid*	w	0.1..50	1.0 1 setpitch
getnvel (gnv)	returns the velocity for move	*vel axisid*	r		1 gnv

setnvel (snv)	defines the velocity for move	*axisid*	w	0.0001...2000.0	12.0 1 snv
getnaccel (gna)	returns the acceleration for move	*axisid*	r		1 gna
setnaccel (sna)	defines the acceleration for move	*acc axisid*	w	1..2000	120.0 1 sna
getnstopdecel	returns the acceleration for a commanded stop or limit-switch activation	*axisid*	r		1 getnstopdecel
setnstopdecel	defines the acceleration for a commanded stop or limit-switch activation	*acc axisid*	w	1..2000	400.0 1 setnstopdecel

getncalvel	returns the speed for cal-move	*axisid*	r		1 getncalvel
setncalvel	defines the speed for cal-move	*value 1 axisid* *value 2 axisid*	w		5.0 1 1 setncalvel 0.1 2 1 setncalvel
getnrmvel	returns the speed for rm-move	*axisid*	r		
setnrmvel	defines the speed for rm-move	*value 1 axisid* *value 2 axisid*	w		50 1 1 setnrmvel 0.1 2 1 setnrmvel
getumotmin	returns the motor-umotmin	*axisid*	r		1 getumotmin
setumotmin	defines the motor-umotmin (*)	*value axisid*	w	see table	500 1 setumotmin
getumotgrad	returns the motor-umotgrad	*axisid*	r		1 getumotgrad
setumotgrad	defines the motor-umotgrad (*)	*value axisid*	w	see table	20 1 setumotgrad
getnlimit	returns the travel-limits	*axisid*	r		1 getnlimit
setnlimit	defines the travel-limits	*low high axisid*	w	-1000.0 ..1000.0	0.0 100.0 1 setnlimit
nsave	save all parameters in flash-memory	*axisid*	w		1 nsave
nrestore	restores the last saved parameters	*axisid*	w		1 nrestore
ngsp	returns the stack-counter	*axisid*	r		1 ngsp
nclear	clear controllers internal stack	*axisid*	w		1 nclear
setaxisno	define address of the controller	*axisid*	w	1..16	2 setaxisno
getaxisno	returns address of the controller	*axisid*	r	1..16	2 getaxisno
nreset	resets the controller	*axisid*	w		1 nreset

setuv	stores uservalue (32-bit int) to memory	*value Id axisid*	w		12222 1 2 setuv
getuv	load uservalue (32-bit int) from memory	*id axisid*	r		1 2 getuv

Pollux NT (closed-loop) specific commands:

getcloop	returns if closed loop active or not	*axisId*	r		1 getcloop
setcloop	defines if closed loop active or not	*status axisId*	w	0..1	1 1 setcloop
getclwindow	returns the defined closed-loop in-target window	*axisId*	r		1 getclwindow
setclwindow	defines the closed-loop in-target window	*size axisId*	w	0.0-1.0	0.001 1 setclwindow
getclwintime	returns the defined closed-loop in- window time [ms]	*axisId*	r		1 getclwintime
setclwintime	defines the defined closed-loop in- window time [ms]	*time axisId*	w	0-8191	10 1 setclwintime
getnrefvel	returns the speed for refmove (index search)	*axisId*	r		1 getnrefvel
setnrefvel	defines the speed for refmove (index search)	*value 1 axisid* *value 2 axisid*	w		1.0 1 1 setnrefvel 2.0 2 1 setnrefvel
nrefmove	starts a refmove (index-search)	*abstarget axisId*	w		5.0 1 nrefmove
getrefst	returns the status of refmove (index-search)	*axisId*	r		1 getrefst
getref	returns the transition of the index mark	*axisId*	r		1 getref
setref	defines the transition of the index mark	*transition axisId*	w		0 1 setref
getemergency	returns the configuration of emergency shortcuts	*axisId*	r		1 getemergency
setemergency	defines the configuration of emergency shortcuts	*config axisId*	w	0-3	3 1 setemergency
getscaleinterface	returns the type of encoder	*axisId*	r		1 getscaleinterface
setscaleinterface	defines the type of encoder	*type axisId*	w	0-2	1 1 setscaleinterface

Some motorsettings for motors used by MICOS:

The values could vary, dependent on the desired load and application!

Motor	Nominal Current [Amp]	Coil resistance [Ohms]	commonly used with stages	*umotmin*	umotgrad
Pollux Motor I and II	1.2		VT80,DT80	*2000..2300*	110
Pollux Motor III	1.2		VT80,DT80	*2000*	110
4H4018	1.7	1.7	VT80, DT80 , HT90	*3200*	90
ST-4018 L1206 halfcoil	1.2	3.3	LS110, PLS85, DT65N, ES65, MA35	*3500*	140
PK-245-01B fullcoil	0.85	6.6		*4000..5000*	400
PK-245-01B halfcoil	1.2	3.3		*3000..3500*	150
PK-244-01B halfcoil	1.2	3.3	ES-100	*3500*	140
PK-244M-01B fullcoill	0.85	6.6	DT-50R DT-80R	*4000-5000*	140
ZSS-43-200-1.2-E parallel	1.2	2.6	LS110, PLS85, PRS110, HPS170, MS8,DT65N, WT90	*4300*	150
ZSS-42-200-1.2-E parallel	1.2	1.6	NPE200, MA35	*3500*	140
ZSS-52-500-2.5E parallel	2.5	0.6	LS180, UPM160		
AM1524-A0.25	0.25	12.5	MP20S MP20L, MT55, MT60,MT40, ASS5/ADS5	*2500 –3500*	140
ZSS-25-200-1.2 parallel	1.2	0.95	WT85, WT100, MT40,MTS-70	*2000*	24
ZSS-32-200-1.2 parallel	1.2	1.3	MT60	*3300*	60
PK266-E2.0 parallel	2	0.9	DT120, UPL-160, WT120	*3000..3200*	160..250
CTP11-13	1.3	3.3		*3500*	140
ST-2818S1006	0.95	3.4		*4000*	100
LIN-208-17-1	0.8	5.4		*3800-5400*	20
LIN-211-18-02	1.3	1.3	VT-40 MP-20	*2400*	60
MICOS 2Ph-018	0.24	20.4	VT-21, MP-21, ES-50	*6500*	45

Nominal Current: motors rated continues current, not the real current with the documented settings
Coil resistance: motor single phase resistance (varies depending on wiring type, fullcoil, halfcoil , serial or parallel)

Please note: Without damper motors get stuck in the resonance area of the motor, which is mainly in the range of 4 rev/sec (200 fullstep motor). A damper (oriental) eliminates perfectly!

Read the actual parameters: axisId getumotmin and axisId getumotgrad

Write new parameters: value axisId setumotmin and value axisId setumotgrad

If parameters ok, save flash-memory axisId nsave

Power-Connector:

manufacturer	Binder Connector
type	Kabeldose gerade Serie 719 3pol
Art.Nr (manufacturer)	09 9748 70 03

Binder 3 pin	Function
1	+24 V
2	-
3	GND

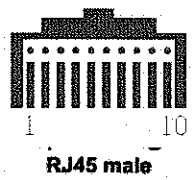
Motor Interface: DSub9 standard

DSub9emale	Function
1	Phase A+
2	Phase A-
3	Phase B+
4	Phase B-
5	Gnd limit-common
6	cal-switch (limit reverse)
7	rm-switch (limit forward)
8	+ 5V for active sensors
9	nc.

Interface-Cable RS-232:

DSub9f	function	color	RJ45 male 10 pin
2	RxD	yellow	5
3	TxD	green	6
5	GND	brown	8

Caution: PC-side : connect pin 1+4+6
connect pin 7+8



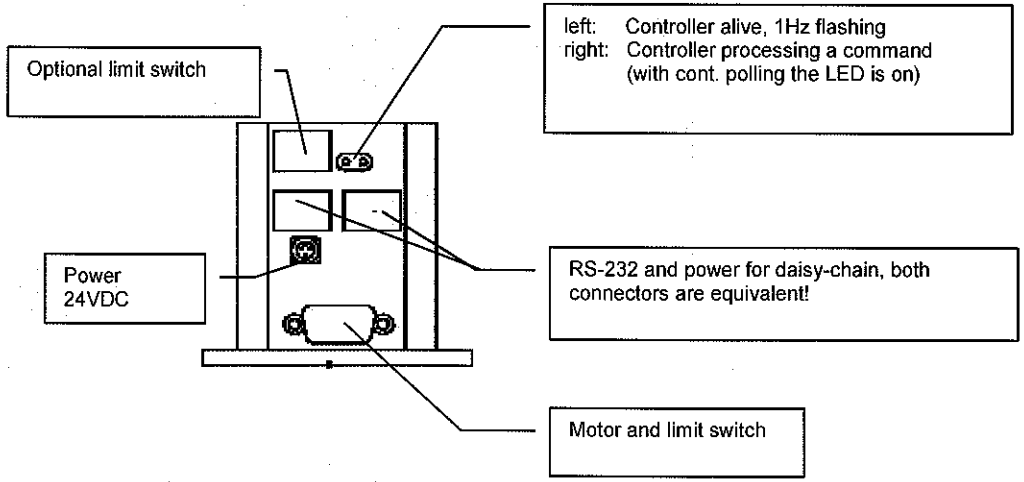
RJ45 10 pin
front view to contacts

Pollux-Motor Limit Switch Connection

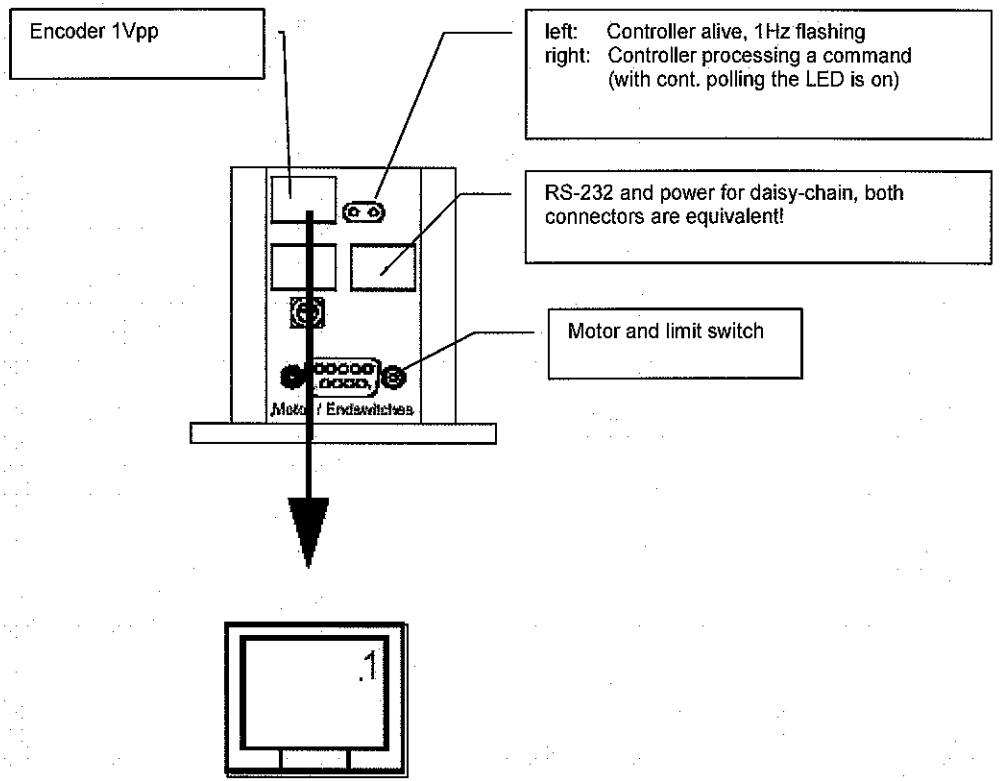
Open-leads to connect directly to the switches (active and passive)

lead-color	Function
white	+ 5V for active sensors
yellow	cal-switch (limit reverse)
green	rm-switch (limit forward)
brown	Gnd limit-common

Pollux:

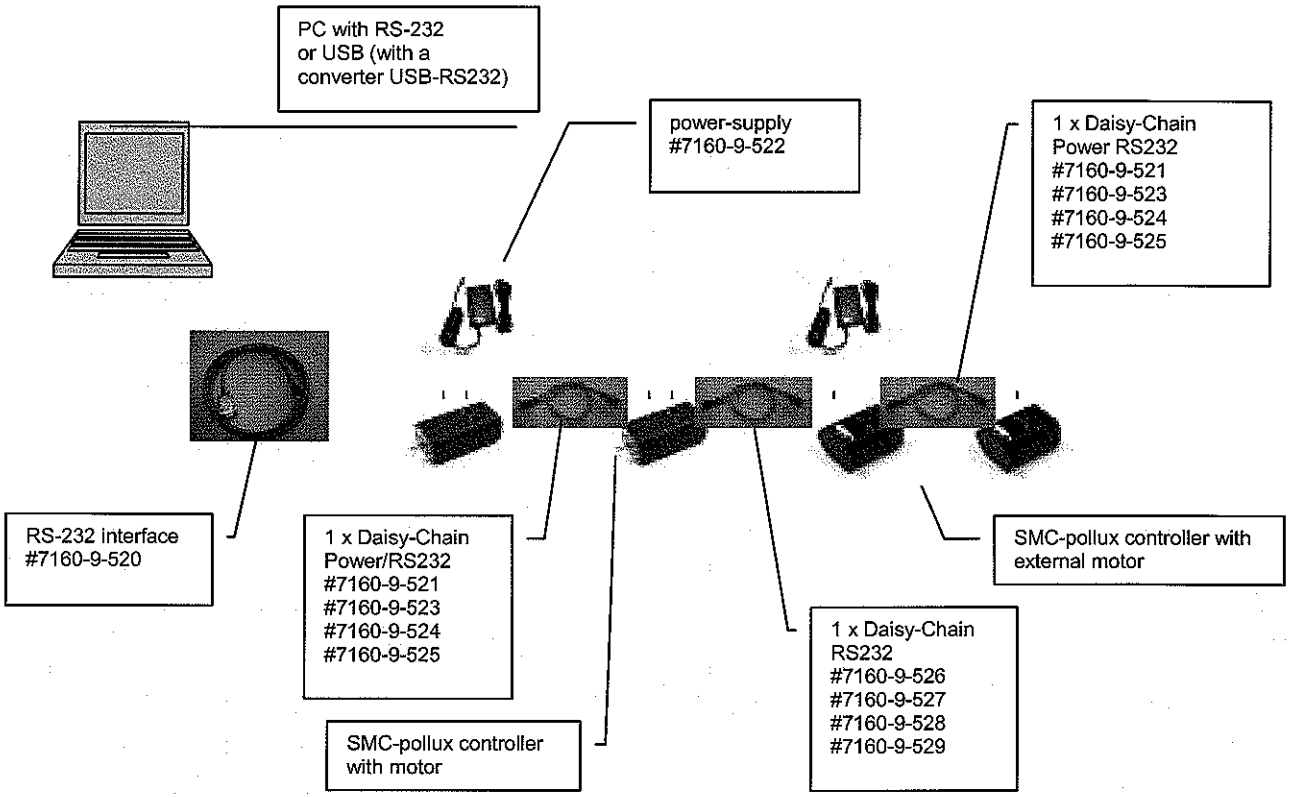


Pollux NT (closed-loop):

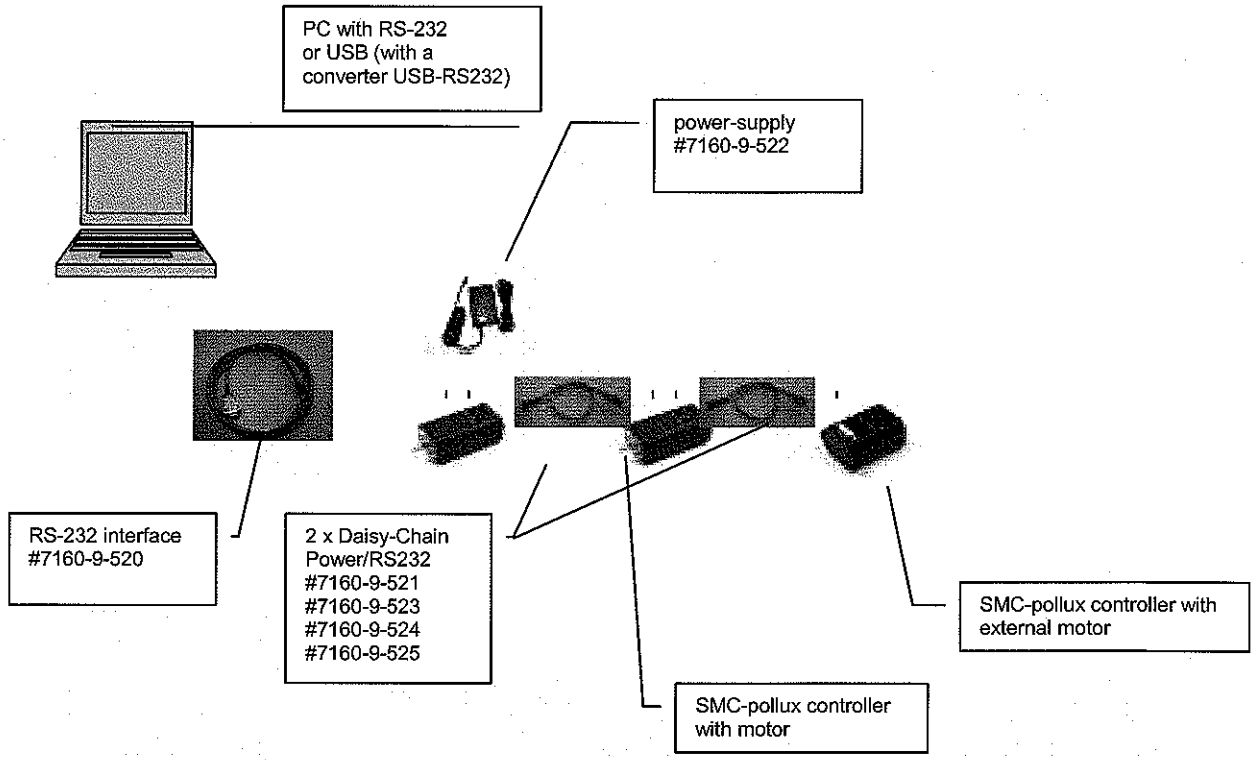


RJ-45 10 pin	Function 1Vpp Encoder
1	5V
2	
3	Sin + (A+)
4	Sin – (A-)
5	Cos + (B+)
6	Cos – (B-)
7	Ref + (Index+
8	Ref- (Index-)
9	Gnd
10	

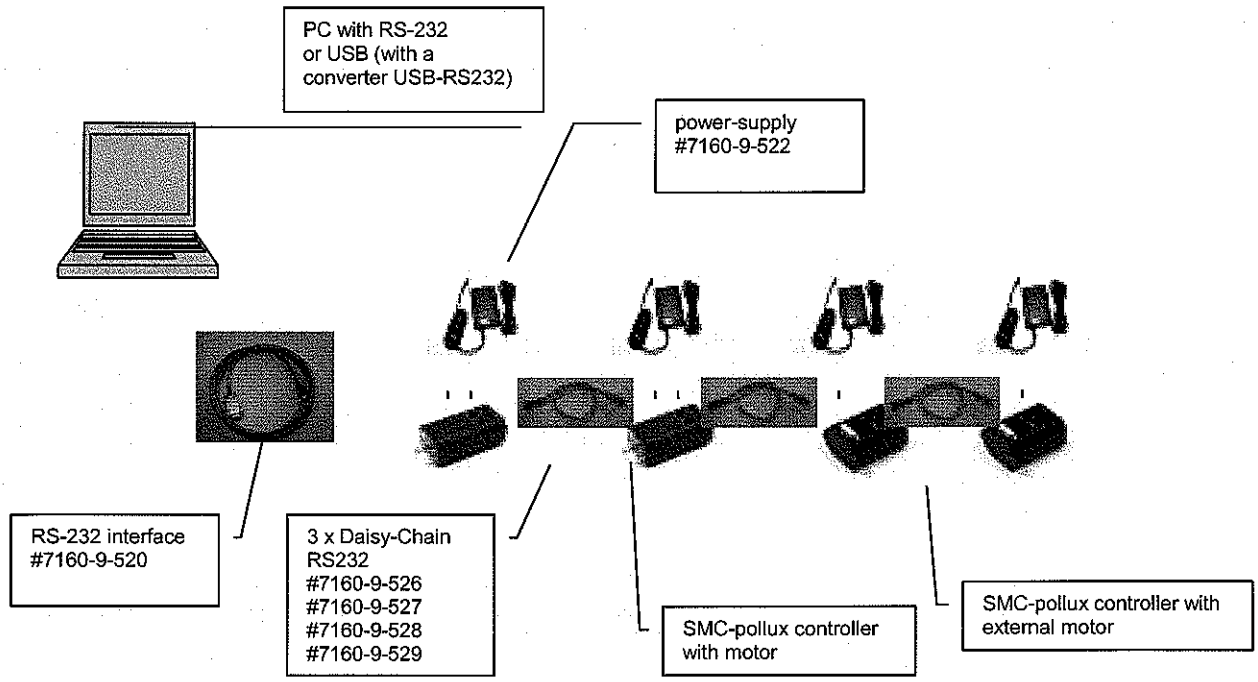
Multiaxis configuration with 2 power-supplies



Multiaxis configuration with one power-supply



Multiaxis configuration with separated power-supply



Content

1 General

These VIs have been created using LabVIEW 6i and will not run with older versions of LabVIEW.

The communication with the controller SMC is done by sending and receiving ASCII strings, this command language is called VENUS language. Depending on the controller, VENUS 1 or VENUS 2 is used. For details about the VENUS language please read the corresponding documentation.

These VIs encapsulate the most important VENUS commands like movement commands, position and status interrogation, velocity, or finding limit switches and index marks. The more low level VENUS commands for configuration of the controller are not implemented because the controller already is configured for your mechanics. If you have problems, please contact MICOS GmbH.

The SMC VIs itself use another VI called SMC_Manager, this SMC_Manager stores information like the number of axes. It also stores the last acquired position and status, so if you need these information you can get it directly from the SMC_Manager. It also allows a simulation mode which is useful for writing software without a controller connected.

The SMC VIs also use a VI named Communication Manager, which itself does all the communication with the various communication channels (RS232, GPIB, TCPIP, VISA).

2 SMC VIs

2.1 Initialization

2.1.1 SMC_Init

Initialize the communication with the controller and prepare the SMC_Manager.

Input Parameters:
- Comm Mode: RS232, TCPIP, GPIB or VISA
- Com Port: use 1 for COM1 and so on
- Baud Rate: use 9600 for the SMC Basic or the SMC Compact, 19200 for the SMC Pollux, SMC Pegasus or SMC Taurus, or 57600 for the SMC Corvus or SMC PCI in the default configuration
- IPAddress: IP address of the SMC controller (only if available)
- Remote Port: port for TCPIP, usually 23
- GPIB Address: GPIB address of the SMC controller (only if available)
- VISA Resource Name: enter here a valid VISA name (e.g. GPIB0::4::INSTR)
- VISA Serial: if the VISA connection uses RS232, then this must be set to true
- VISA Baud Rate: if the VISA connection uses RS232, then specify here the baud rate
- Number of Axis: use 2 or 3 depending on your order option
- VENUS language: select VENUS 1 or VENUS 2 depending on the controller
- Controller Nr: address of the controller (only for VENUS 2)
- Simulation: use true if you want to work without a connected SMC controller
- Error

Output Parameters:
- Error

Remark:
This VI calls the VIs Communication Manager to open the communication, the SMC_Manager, and SMC_SetDim.

2.1.2 SMC_Close

Closes the communication with the controller.

Remark:
This VI calls the VIs Communication Manager to close the communication with the controller.

2.2 Interrogation

2.2.1 SMC_GetDim

Get the number of axes (dimension) which the SMC controller uses by sending the VENUS command getdim. For VENUS 2 the result is always 1.

Input Parameters:
- Error

Output Parameter:
- Error
- Dimension of the SMC controller (number of axis)

2.2.2 SMC_GetPos

Get the actual position by sending the VENUS command pos or np.

Input Parameters:
- Error

Output Parameter:
- Error
- Answer string from the SMC

Remark:
This VI calls the VI SMC_Manager with the mode SetPostion so that the SMC_Manager stores the actual position (as array of double).

2.2.3 SMC_GetSt

Get the actual status of the controller by sending the VENUS command st or nst.

Input Parameters:
- Error

Output Parameter:
- Error
- Answer string from the SMC

Remark:
This VI calls the VI SMC_Manager with the mode SetStatus so that the SMC_Manager stores the actual status and moving mode.

2.2.4 SMC_GetSwSt

Get the condition of the limit switches (touched or not touched) by sending the VENUS command getswst.

Input Parameters:
- Error

Output Parameter:
- Error
- Switches min: array of boolean with the information about the backward limit switches
- Switches max: array of boolean with the information about the forward limit switches

2.2.5 SMC_GetVel

Get the velocity setting by sending the VENUS command gv or gnv.

Input Parameters:
- Error

Output Parameter:
- Error
- velocity

2.2.6 SMC_GetAcc

Get the acceleration setting by sending the VENUS command ga or gna.

Input Parameters:
- Error

Output Parameter:
- Error
- acceleration

2.2.7 SMC_GetLimit

Get the possible travel range by sending the VENUS command getlimit or getnlimit.

Input Parameters:
- Error

Output Parameter:
- Error
- MinValues: array of double with the minimum travel ranges
- MaxValues: array of double with the maximum travel ranges

2.2.8 SMC_GetError

Get the last occurred error of the SMC controller by sending the VENUS command ge or gne. 0 means no error.

Input Parameters:
- Error

Output Parameter:
- Error
- Error Number
- Error String

Remark:
The Error String is available in English and in German, the language you can select in the block diagram.

2.2.9 SMC_GetAI

Get the value of the analog input.

Input Parameters:
- Error

Output Parameter:
- Error
- Value from the analog input
- Average value

Remark:
The analog input is a special option and not available for all controllers. Also included is an averaging of the measurement value.

2.3 Finding switches and index marks

2.3.1 SMC_Calibrate

Find the backward limit switches and set the position to zero by sending the VENUS command cal or ncal.

Input Parameters:
- Error

Output Parameter:
- Error

Remark:
For some controllers (e.g. SMC Corvus) this VI exits only after the movement has been finished. For other controllers (e.g. SMC Pollux) it returns immediately.

2.3.2 SMC_RangeMeasure

Find the forward limit switches by sending the VENUS command rm or nrm.

Input Parameters:
- Error

Output Parameter:
- Error

Remark:
For some controllers this VI exits only after the movement has been finished.

2.3.3 SMC_RefMove

Find the index marks of the encoders by sending the VENUS command refmove or nrefmove.

Input Parameters:
- Error
- Number of revolutions

Output Parameter:
- Error

Remark:
This is only possible with the corresponding order option.
This VI exits only after the movement has been finished.

2.4 Movement

2.4.1 SMC_AxisAbs

Move one axis absolute by sending the VENUS command m or nm.

Input Parameters:
- Error
- Axis number
- New position
- WaitUntilReady

Output Parameter:
- Error

Remark:
Before starting the movement, the moving mode in the SMC_Manager is set to true.
Depending on the input WaitUntilReady, the VI returns immediately or waits until the end of the movement. The input Axis number is used only for VENUS 1, it is ignored for VENUS 2.

2.4.2 SMC_AxisRel

Move one axis relative by sending the VENUS command r or nr.

Input Parameters:
- Error
- Axis number
- Distance
- WaitUntilReady

Output Parameter:
- Error

Remark:
Before starting the movement, the moving mode in the SMC_Manager is set to true.
Depending on the input WaitUntilReady, the VI returns immediately or waits until the end of the movement. The input Axis number is used only for VENUS 1, it is ignored for VENUS 2.

2.4.3 SMC_MoveAbs

Move all axes absolute by sending the VENUS command m or nm.

Input Parameters:
- Error
- Array of double with the new positions
- WaitUntilReady

Output Parameter:
- Error

Remark:
Before starting the movement, the moving mode in the SMC_Manager is set to true.
Depending on the input WaitUntilReady, the VI returns immediately or waits until the end of the movement.

2.4.4 SMC_MoveRel

Move all axes relative by sending the VENUS command r or nr.

Input Parameters:
- Error
- Array of double with the distances
- WaitUntilReady

Output Parameter:
- Error

Remark:
Before starting the movement, the moving mode in the SMC_Manager is set to true.
Depending on the input WaitUntilReady, the VI returns immediately or waits until the end of the movement.

2.4.5 SMC_Stop

Stop all movements by sending CTRL C (ASCII #3).

Input Parameters:
- Error

Output Parameter:
- Error

2.5 Configuration

2.5.1 SMC_SetDim

Sets the dimension (number of axes) the controller uses by sending the VENUS command setdim. For VENUS 2, this VI does not send a command to the controller.

Input Parameters:
- Error
- Number of dimensions

Output Parameter:
- Error

Remark:
You can configure a three axis controller to act like a two axis controller, but you cannot use a two axis controller as a three axis controller. This is an extra order option.

2.5.2 SMC_SetVel

Sets the velocity by sending the VENUS command sv or snv.

Input Parameters:
- Error
- Velocity value

Output Parameter:
- Error

2.5.3 SMC_SetAcc

Sets the acceleration by sending the VENUS command sa or sna.

Input Parameters:
- Error
- Acceleration value

Output Parameter:
- Error

2.6 Miscellaneous

2.6.1 SMC_SetZero

Sets the coordinates to zero by sending the VENUS command setpos or setnpos.

Input Parameters:
- Error

Output Parameter:
- Error

2.6.2 SMC_SendCommand

Sends directly a VENUS command to the controller.

Input Parameters:
- Error
- VENUS command

Output Parameter:
- Error
- Answer string

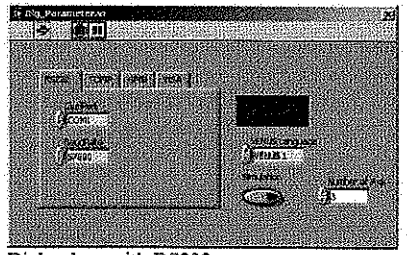
Remark:
This VI expects one line as answer. To avoid a timeout error if you send a command which gives no answer back, a " st" or " nst" command is automatically added. If the answer consists of more than one line, only the first one is given back, the other lines are ignored.

If you want to send a command which gives back an answer consisting of more than one line, you should do it in a way like in the SMC_GetLimit VI.

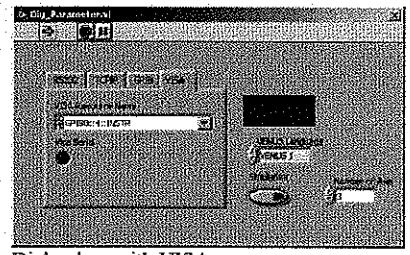
3 TestProgram

TestProgram is a simple demo program which shows the use of the SMC VIs.

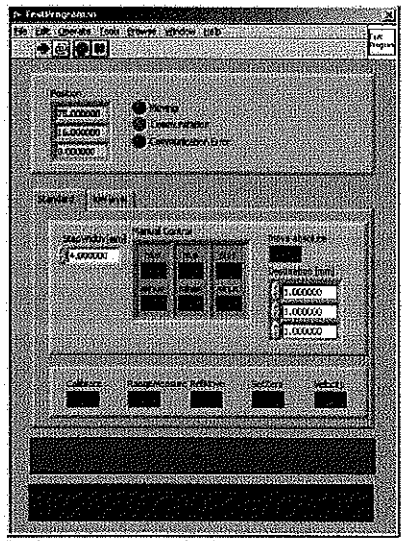
In the first frame in the block diagram you can enter the communication parameters and the number of axis manually, or you can use a dialog box at every start of the program.



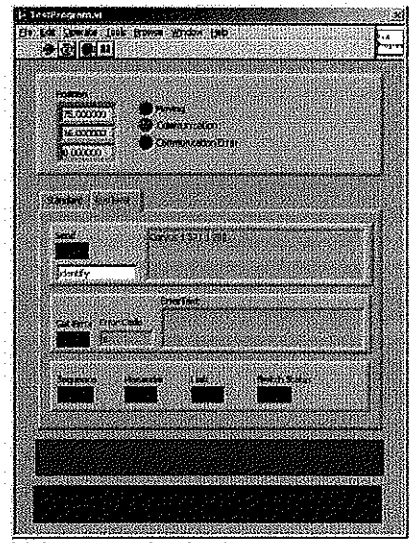
Dialog box with RS232 parameters



Dialog box with VISA parameters

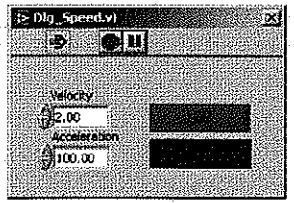


Main program, standard page



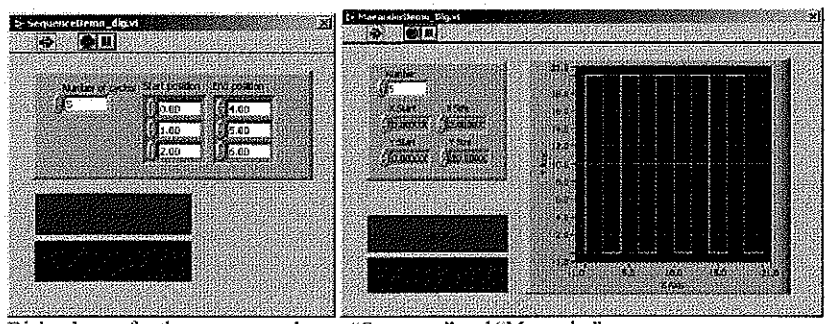
Main program, low level page

In the main program, you can start relative movements with the corresponding "plus" and "minus" buttons and absolute movements with the "Move Absolute" button.
"Calibrate" looks for the backward limit switches, "RangeMeasure" looks for the forward limit switches. If available, "RefMove" looks for the index mark of an optional encoder.
"Velocity" opens a dialog box which shows and sets the movement velocity and acceleration.



Dialog box for setting the movement velocity and acceleration

In the low level page, you can directly send VENUS commands like "pos" or "identify". Please be careful, here you have full access to all parameters of the controller. Also in the low level page, you can start some movement demos.


Dialog boxes for the movement demos "Sequence" and "Maeander"

The second frame in the block diagram consists of two loops. The upper one reacts on the buttons, or if you don't press a button, it queries the actual position and status which are stored in the SMC_Manager and also written in two global variables.
The lower loop runs always and has two jobs: it checks the Stop button and updates the position and moving display from the global variables.

The global variables are used for the display only (which is updated in the lower loop of the test program). Since the three VIs SMC_Calibrate, SMC_RangeMeasure and SMC_RefMove do return only after the movement has been finished, the global variable Moving is set manually in the test program.

The advantage of the concept of the SMC_Manager who stores the actual position and moving status is that if you need these informations in your application, you get these informations much faster from the SMC_Manager than from the controller itself.