Declaration of the external interface between the ng_biopro-software and another external control-program

Uwe Tangen

Version 0.51, 05 September 2008

Uwe Tangen Ruhr-University-Bochum

Abstract

The aim of this specification is to allow external programs to control essential parts of the Omega-machine via an as simple as possible interface. The general idea behind this is to plug the ng_biopro-software into an externally provided optimization and control-system to facilitate a system-integration on a larger scale.

The interface will not encompass all ng_biopro-software functionalities but expects the experimenter to setup the experiment with the ng_biopro-software and after having done so giving control to the external program.

1 General setup

The ng_biopro-software partially works as a client of the Bio@Fox-board, provides the interface to the camera and will retain full control of the Omega-machine. The software will open an external TCP-socket at port 8086 to allow for external control when it is launched with the option '-ext'.

Before it makes sense to control parts of the Omega-machine with the external program the experiment has to be set up already. Everything should be as prepared as a normal experimental session would require.

The external program is launched and it opens the port 8086 at the ng_biopro-software (the IP-address must be the address of the machine running the ng_biopro-software).

Example:

```
ssh liszt -l biopro
cd test
ng_biopro_i686 -ext -srv
```

The server-option is useful if the ng_biopro-software should be used for testing-purposes. If a firewall is between the machine executing the external program and the machine executing the ng_biopro-software, the port 8086 must be tunneled through the SSL-connection, which currently is not working - sorry.

ssh liszt -l biopro -L 8086:liszt:8086

and start the external program with

ng_biopro_external localhost 8086

The external control-program acts as a client to the ng_biopro-software. The external program opens a control-session with specifying a certain name for that session and perhaps additional data in a comment-string. The name of the control-session is always printed out when actions are logged. When finishing the control session, the external-program shall close the session with an appropriate command to allow for a graceful shutdown of the TCP-connection.

The general communication between the external client and the ng_biopro-software is realized via the exchange of ASCII-strings. Binary data is NOT communicated. This saves from the problem of little and big-endian machines.

A communication string starts with the length of the string, given in bytes, and a 1 byte command. Atom-delimiter is the space (blank, 0x20). Integers begin with a non-'0' character and hex-numbers with '0x'. Doubles or floats contain at least a '.' or an 'e' in the set of digits. Upper and lower case DOES matter, 'a' is NOT equal 'A'. Strings with blanks are encapsulated with quotation-marks (0x22).

Sometimes commands require acknowledgements. The simple acknowledgement is given by a **y** (0x79) and the appended session-name, which allows the external program to check the communication channel.

If there are sequences of items, be it atomic-names, numbers or strings, these sequences are prepended with the number of these items. E.g., naming three colors as a list would be written as: 3 black white yellow.

2 The commands specification

In the examples shown below the length of the commands will not be shown. It is determined via the strlen()-function in a lower level of the communication system.

2.1 Quick-list of all commands and return-parameters

2.1.1 Commands which can be sent to the ng_biopro-software

(Caution: all hardware related names are specific to the BioMIP-setup. They might be different in test- or other lab-environments!)

- '**a**' (0x61) - Set AOTF or light-source intensity (2.4.3) {name [biofox_blue, biofox_red, biofox_yellow]} [{intensity [0..63]}]

- ’**B**’ (0x42) - Start (begin) a predefined cylce (2.3.1) {qualifier [**aotf, camera, filter**, **level_one**, **light**, **macro**, **measurement**, **pin, prepos, pump, sensor, sequence, series, temp_cycle, wheel, xyzpos**]} {name} [{start at position *n* (from zero counted)}] /* not yet implemented */
- ’**C**’ (0x43) - Define camera parameters (2.4.1) {name [eval_cam]} [{exposure time in seconds} [{camera gain [0..255]}]].
- ’**c**’ (0x63) - Close a session (2.2.1)
- ’**d**’ (0x64) - List all pins available in design (2.2.2)
- ’**E**’ (0x45) - Stop (end) a predefined running cycle (2.3.2) {qualifier [**aotf, camera, filter**, **level_one**, **light**, **macro**, **measurement**, **pin, prepos, pump, sensor, sequence, series, temp_cycle, wheel, xyzpos**]} {name} [end] /* not yet implemented */
- **’e’** (0x65) - Apply a certain potential to an electrode (2.3.5) {pin-ID [0x80008f0]} {polarity [-, +, z]}
- **’f’** (0x66) - Set a filter-wheel to a specified position (2.3.6) {wheel-name [emission]} {filter-position [1..10]}
- **’G’** (0x47) - Get status of a predefined cycle (2.3.3) {qualifier [**aotf, camera, filter**, **level_one**, **light**, **macro**, **measurement**, **pin, prepos, pump, sensor, sequence, series, temp_cycle, wheel, xyzpos**]} {name} /* not yet implemented */
- **’g’** (0x67) - Get a value of a sensor (2.3.7) {nr. of sensors [1..n]} {sensor-IDs [0x20008f4, ..]}
- **’i’** (0x69) - Get an image from server (2.3.8)
- ’**I**’ (0x49) - Get status from calling a shell-script (2.2.3) {parameters for shell-script}
- ’**l**’ (0x6c) - List a sensors-data (2.2.5) {sensor-ID [0x2000013]}
- ’**L**’ (0x4c) - Extract a list of names (2.2.4) {qualifier [**aotf, camera, filter**, **level_one**, **light**, **macro**, **measurement**, **pin, prepos, pump, sensor, sequence, series, temp_cycle, wheel, xyzpos**]}
- ’**m**’ (0x6d) - List all measurement bodies (2.2.6)
- ’**n**’ (0x6e) - Set norm-duty, active-duty and reference-cycle (2.4.4) {type [**norm**, **active**, **ref**]} [{value [0..]}]
- ’**o**’ (0x6f) - Open a session (2.2.7) {session name} [{comment}]
- ’**p**’ (0x70) - Define new pump-parameters (2.4.5) {pump-ID [1..]} {qualifier [**dia**, **flow**, **unit**, **comment**]} [{value}]

- '**Q**' (0x51) or 'q' (0x71) - Abort a session (2.2.8)
- '**r**' (0x72) - Get a full report on actions taken (2.2.9) {qualifier [after, since]} [{id}] /* not yet implemented */
- '**S**' (0x53) - Synchronize to an existing event or cycle (2.3.4)/* not yet implemented */
- '**s**' (0x73) - set and get the state of the system (2.2.10)
- '**T**' (0x54) - Set temperature (2.4.2) {name [biofox_A, biofox_B, biofox_F]} [{temperature in °C}]
- **'t'** (0x74) - Execute a test program (simulate an intensity-profile) (2.2.11)/* not yet implemented */
- **'u'** (0x75) - Control the pumps (start, stop etc.) (2.3.9) {action [**inject_all**, **withdraw_all**, **stop_all**, **inject**, **withdraw**, **stop]}** [{pump-ID}]
- **'v'** (0x76) - Update design-window (2.2.12)
- **'x'** (0x78) - Move the xy-table to a certain position (2.4.6) {nr. of axes [2]} [{name + position-x name + position-y in micro-meter}]
- **'Y'** (0x59) - Extract a list of all cycles known (2.2.13) /* not yet implemented */
- **'y'** (0x79) - Move the xy-table and z-stage to a predefined position (2.4.7) {name of position}
- **'z'** (0x7a) - Set position of z-stage (2.4.8) {absolute position in micrometer}

2.1.2 Possible return commands received from the ng_biopro-software

- '**c**' (0x63) from **'Y'** (2.2.13),
- '**f**' (0x66) from **'i'** (2.3.8),
- '**g**' (0x67) from **'l'** (2.2.5),
- '**h**' (0x68) from **'i'** (2.3.8),
- **'i'** (0x69) from **'g'** (2.3.7),
- '**l**' (0x6c) from **'i'** (2.3.8),
- **'m'** (0x6d) from **'m'** (2.2.6),
- '**n**' (0x6e) from **'L'** (2.2.4),
- **'p'** (0x70) from **'d'** (2.2.2),

- 'R' (0x52) from 'i' (2.3.8),
- 'r' (0x72) from 'i' (2.3.8),
- 's' (0x73) from 's' (2.2.10), 'G' (2.3.3), 'I' (2.2.3)
- 'S' (0x53) from 'i' (2.3.8),
- 't' (0x74) from 'i' (2.3.8),
- 'x' (0x78) from 'i' (2.3.8),
- 'y' (0x79) from 'o' (2.2.7), 'c' (2.2.1), 'm' (2.2.6), 'd' (2.2.2), 'f' (2.3.6), 'I' (2.2.3), 'L' (2.2.4), 'u' (2.3.9), 'v' (2.2.12), 'r' (2.2.9), 't' (2.2.11), 'B' (2.3.1), 'E' (2.3.2), 's' (2.2.10), 'C' (2.4.1), 'T' (2.4.2), 'a' (2.4.3), 'n' (2.4.4), 'p' (2.4.5), 'x' (2.4.6), 'y' (2.4.7) or 'z' (2.4.8)

2.2 General interface-commands and utility functions

Without any commands given, the server might stop the communication either with _MSG_QUIT_ ('**Q**', 0x51) or with '**q**' (0x71). The numbers in brackets starting with '0x' are the hexadecimal equivalents of the command-letters. Most commands return 'E' plus an explanation string if something fails. The usual return-scheme is not altered though.

2.2.1 'c' (0x63) - Close a session

(back to 2.1.1)

This call has no parameters. The return-value will be **y** (0x79) with the session-name of the just closed session as a control. After having received the correct return the connection to the ng_biopro-software can be savely shut down.

Example:
c
return:
y my_first_test

2.2.2 'd' (0x64) - List all pins available in the design

(back to 2.1.1)

This command returns a list of all known pins with specifics on the single pins. The pinID is then needed for the command **e**, see section 2.3.5. There exist some pins which do not have a net-connected. Though these pins are electrically active, which is due to the special electrodes layout, the according cannot be extracted. Because these pins are sitting at the end of a net and there name is of type *-4 ...*-8 the according valid pins are these with just 4 substracted from the shown numbersi, e.g. B3_WS_X3-6 is identical with B3_WS_X3-2. The position information given are the real coordinates in the design. These are not the coordinates in the camera-window. Only after synchronizing both windows the according camera-coordinates can be calculated.

Example:
d
Return:
p 0x80008f0 B3_WS_O3-2 /B3/N_9428(1) 234.5 323.2
.......pinID..... ..pin-name......net-name........x.......y
.
.
.
y my_first_test

2.2.3 'I' (0x49) - get information on the status of shell-script execution

(back to 2.1.1)

Returns information from an executed shell-script.
I /home/biopro/bin/rob_info_a.sh test upper
...... path of info-script ....list of parameters ...
Return:
I 2 height 30
I 4 width 34 65 78
.
.
y my_first_test

2.2.4 'L' (0x4c) - extract a list of names

(back to 2.1.1)

This command list the known names of certain element categories. A user at the ng_biopro-software can define an abritrary number of names, be it measurement-elements, sensors, temperature cycles, table-movement cycles, electrode pattern and so on. The command allows to specify the following categories or element-types: **aotf, camera, filter**, **level_one**, **light**, **macro**, **measurement**, **pin, prepos, pump, sensor, sequence, series, temp_cycle, wheel, xyzpos**.

The return will be a list of names plus the current intensity and a **y** (0x79) with the session-name appended as last string.

Example:
L light
.. qualifier ...
return:
n 'biofox_blue'
... name ....
n 'biofox_red'
y my_first_test

2.2.5 'l' (0x6c) - list a sensors-data

(back to 2.1.1)

This command gives back data from a sensor. The sensorID can be derived from **m** (see section 2.2.6).

The command needs the sensorID as parameter and returns the name and the geometry.

Example:

l 0x2000013

... sensor-ID ...

Return:

g test_sen 8 23 25 30 30 45 60 56 89

.....name..nr..x0,y0, x1,y1, x2,y2, x3,y3

2.2.6 'm' (0x6d) - list all measurement bodies

(back to 2.1.1)

This command gives back a list of measurement-bodies known to the ng_biopro-software which do have a known size and position in the camera-window. Either these are drawn by hand with the Measurement-button or they steem from the design-window after synchronizing the camera-window with the design-window (Adjust-button). Each item of the list contains the name of this measurement and a list of its sensorIDs in the camera-window.

The command has no parameters and returns a set of strings starting with **m** (0x6d) with the last string being the acknowledgement. A side-effect of this command is the initialization of the sensors just given back. This initialization might take some time, depending on the number of sensors extracted.

Example:

m

Return:

m test_meas 3 0x2000014 0x200000d 0x200000b

........name....nr.............IDs......

m back_sen 1 0x200000c

y my_first_test

2.2.7 'o' (0x6f) - Open a session

(back to 2.1.1)

This call has two parameters: the name of the session, given as a string or atomic sequence of characters (without blank-character) and a comment-string which will also be logged in the session.out file of the ng_biopro-software. The name of the session is being prepended with all actions of the external-program and written in the session.out-file of the ng_biopro-software.

The return value will be **y** (0x79) with the session-name appended.

Example:

o my_first_test "this will be an opportunity"

.... session ..... comment ...

return:
y my_first_test

2.2.8 'q' (0x71) or 'Q' (0x51) - Abort a session

(back to 2.1.1)
This call has no parameters and no return. It should be used if something with the software is wrong.
Example:
q
return:
none

2.2.9 'r' (0x72) - Get a full report on actions taken /* not yet implemented */

(back to 2.1.1)
All actions either via this external interface or interactively at the graphical-user-interface will be stored in a command- or action-list. This list can be fetched with this command. The command has two qualifiers: either the **after** with a number as parameter or **since** with a time (in seconds since 01.01.1970 and micro-seconds as second parameter). If no parameters are provided all available actions are reported. If the ID of the action is omitted then the last action is shown.
Example:
r after [56]
... qualifier ... id ...
return:
c 57 34382322 243234 **u inject** 3
.. nr . seconds .. usec .. command .. qualifiers and parameters
c 58 34682322 23423 **u stop** 2
y my_first_test

2.2.10 's' (0x73) - get the state of the system

(back to 2.1.1)
This command just returns hex-number which flags represent the global state of the ng_biopro-software or it is able to set the corresponding bits. The first possible parameter of the command is a mask which defines, which bits are affected and the second possible parameter are the acutal bits (several bits can be set at the same time, the number have to be given in hex-code, '0xa' and 'a' are equivalent). Please take care on the bits set, they can result in strange program-behavior.
The flags are the following:

```
_S_NON_INT_      0x00000100 /* Non-interactive mode */
_S_GRAFIK_UP_    0x00000400 /* Graphics are up and operational */
```

```
_S_CAMERA_        0x00001000 /* A real camera is attached */
_S_SHUTDOWN_      0x00002000 /* Shutdown is ongoing*/
_S_FOX_OKAY_      0x00004000 /* Foxboard is up and running */
_S_NO_CAM_AQU_    0x00008000 /* Do not aquire images automatically */
_S_TEMP_CONT_     0x00010000 /* Temperature-control enabled */
_S_XY_CONT_       0x00020000 /* XY-table control enabled */
_S_Z_CONT_        0x00040000 /* Z-stage control enabled */
_S_LIGHT_CONT_    0x00080000 /* Light-control enabled */
_S_ENA_CYC_       0x00100000 /* Enable cycling */
_S_CYC_RIGHT_     0x00200000 /* Cycle in right direction */
_S_DIS_CYC_       0x00400000 /* Disable cycling */
_S_PUMPS_DEF_     0x00800000 /* Pumps usage is wanted */
_S_BIO_OKAY_      0x01000000 /* BioPRO-Module is successfully conf*/
```

This state is returned with an '**s**' (0x73) and the session-name appended.
Example:
s [{mask} [{bits}]]
Return:
s my_first_test 0x0081d400

2.2.11 't' (0x74) - Execute a test program /* Not yet implemented */

(back to 2.1.1)

This program is either hardcoded into the software or a script written in NGEN-notation. The parameters semantic is insofar defined that the first parameter has to be the number of tokens or parameters given. Strings with white-space characters inside have to be enclosed via quotation marks. The test-program has to be callable via the alphanumerical interface of the ng_biopro-software. The standard output generated from the test-program is returned as a list of strings. These strings are not interpreted. The last returned value is **y** (0x79) with the session-name appended.

Example:
t t_prog_24 3 jojo 34 940
return:
o “This test was successful”
....
o “we will stop this test”
y my_first_test

2.2.12 'v' (0x76) - Update design-window (let the experimenter staying informed)

(back to 2.1.1)

This call has no parameters. The return-value will be **y** (0x79) with the session-name. The command is there for convenience and control. Because updating the graphical design-window is costly this command lets the user decide when to do it. The functionality of the other commands is not affected.

Example:
v
return:
y my_first_test

2.2.13 'Y' (0x59) - extract a list of known cycles /* not yet implemented */

(back to 2.1.1)

This command list the known names of certain element categories. A user at the ng_biopro-software can define an abritrary number of names, be it measurement-elements, sensors, temperature cycles, table-movement cycles, electrode pattern and so on. The command allows to specify the following categories or element-types: **aotf, camera, filter**, **level_one**, **light**, **macro**, **measurement**, **pin, prepos, pump, sensor, sequence, series, temp_cycle, wheel, xyzpos**.

The return will be a list of names plus the current intensity and a **y** (0x79) with the session-name appended as last string.

Example:
Y
return:
c 't3-35'
.... name of cycle .. *new*
c 'biofox_red_34'
y my_first_test

2.3 Action commands used for controlling the experiments

2.3.1 'B' (0x42) - Start (begin) a predefined cycle /* not yet implemented */

(back to 2.1.1)

This command launches a certain defined cycle. This cycle might be a **aotf, camera, filter**, **level_one**, **light**, **macro**, **measurement**, **pin, prepos, pump, sensor, sequence, series, temp_cycle, wheel, xyzpos** cycle. After the qualifier the according name of the defined cycle has to be written. If the name contains white-space-characters it has to be enclosed in quotation-marks. Depending on the type of cylce further parameters might be defined. With no start position defined only the current cycle position is returned. The acknowledgement return value is **y** (0x79) with the session-name appended and the current cycle position.

Example:
B temp_cycle test_cycle [4]
.... qualifier .... name .. start at position 4 (from zero counted)

return:
y my_first_test 5
.... session.......current position in cycle

2.3.2 'E' (0x45) - Stop (end) a predefined running cycle /* not yet implemented */

(back to 2.1.1)
Stop a certain running cycle. Cycle-qualifiers are: **aotf, camera, filter**, **level_one**, **light**, **macro**, **measurement**, **pin, prepos, pump, sensor, sequence, series, temp_cycle, wheel, xyzpos**. This command is orthogonal to subsection 2.3.1. The acknowledgement return value is **y** (0x79) with the session-name appended.
Example:
E move_cycle grappa end
... qualifier ..... name .. wait till the last element of the cycle before stopping
return:
y my_first_test 34
..... session-id ... position where the cycle stopped .. *new*

2.3.3 'G' (0x47) - Get status of a predefined cycle /* not yet implemented */

(back to 2.1.1)
Get a status of certain cycle. Cycle qualifiers are **aotf, camera, filter**, **level_one**, **light**, **macro**, **measurement**, **pin, prepos, pump, sensor, sequence, series, temp_cycle, wheel, xyzpos**. This command is orthogonal to subsection 2.3.1. The return is '**s**' with cycle-specific informations.
Example:
G temp_cycle test_cycle
.. qualifier .... name ...
return:
s test_cycle 4 26.3 130.4
... name .... cur. pos. ... temperature setpoint ... waiting time in sec.

2.3.4 'S' (0x53) - Synchronize to an existing event or cycle /* not yet implemented */

(back to 2.1.1)
With this command we can wait for certain events to happen before we continue in the program execution. The type of event is specified with a qualifier. The following qualifier are specified: **aotf, camera, filter**, **level_one**, **light**, **macro**, **measurement**, **pin, prepos, pump, sensor, sequence, series, temp_cycle, wheel, xyzpos**. With no further parameter given the synchronization event occurs right after the execution of the current element. When the event occured the command returns with **y** (0x79) and the session-name appended.

Example:

S measurement refractive_index

... qualifier .. name of measurement, after storing the data on disk a signal is raised.

S temp_cycle heatup 5

... qualifier ... name .. after processing this position in the temp-cycle raise signal.

return:

y my_first_test

2.3.5 'e' (0x65) - Apply a certain potential to an electrode

(back to 2.1.1)

This command is the most simple of a list of commands controlling electrodes. The parameters required are the name of the elektrode and the potential to be exerted: '-' equals 0V, '+' equals 3.3V and 'z' equals high-impedance. The electrodes name is actually the pin-name of this electrode which can be read in sessions.out-window when clicking on that specific electrode.

Example:

e 0x80008f0 -

e 0x800032a +

e 0x800032b z

Return:

None (no return to allow fast switching of electrodes)

2.3.6 'f' (0x66) - set a filter-wheel to a certain position

(back to 2.1.1)

With this command a filter can be changed in a filter-wheel. With no filter-number speficied only the current filter-position is returned. The usual acknowledgement is sent on success plus the indication which filter is active.

Example:

f emission [8]

... wheel-name .. filter-position

Return:

y my_first_test 4

......................cur. wheel.. *new*

2.3.7 'g' (0x67) - get a value of a sensor

(back to 2.1.1)

This command is accompanied with the handle of that sensor (availabe via the list-sensors command in sec. 2.2.5. Returned is a list of intensities beginning with **i** (0x69) which reflects exactly the sequence of geometries returned via the list-sensors command. The value given back for each geometry is a real-valued average intensity of the according geometry ($= \frac{\sum pixel\, intensities}{nr.\, of\, pixels}$). The pixel-intensities are the raw-uninterpreted pixel-values retrieved from the camera. This means that e.g. for a 12-bit depth camera the maximum pixel value is 4095.

Example:

g 3 0x20008f1 0x20008f4 0x20008f0

... nr. of sensors ... sensor-IDs ...

Return:

i 3 365.45 738.2 453.6

2.3.8 'i' (0x69) - Fetch a single camera image

(back to 2.1.1)

This command asks the ng_biopro-software to provide a current image. This image contains the full camera view (equivalent to the snap-shot-action in the graphical user-interface). The return of this command is a full image, beginning with a header defining the size in x- and y-coordinates and a number of rows containing the actual image-data. Each row returned has x-entries of short (two bytes, or 16bit) intensity values. There are y rows returned. The upper byte of the intensity-value is given first, the lower byte second. Furthermore, context data for the image is returned. See section 'Information appended to each image' in the ng_biopro User Manual. The image counter '**r**', which is counted relative to the open session, is the last returned entity.

Example:

i

Return:

h 1004 1002

R I0UI0LI1UI1LI2UI2L I1003UI1003L

... upper-byte of intensity at x = 0, lower-byte of intensity at x = 0, ...

t 1178095092 124578

.... seconds ..... usec ..

f 'emission' '1white'

... wheel .... filter ...

l 'biofox_blue' 0

...... name ..... intensity [0..63]

l 'biofox_red' 0

x 'x =' 2450955

.... axis .. um ...

x 'y =' -59685

x 'PI =' 39985

S 'biofox_A' 2645

.... sensor ... temperature in °C * 100 ...

S 'biofox_B' -24645

S 'biofox_F' -355

r 2682

.... image counter ....

2.3.9 'u' (0x75) - Control the pumps (start, stop etc.)

(back to 2.1.1)

This command starts and stops the pumps. The following qualifiers are possible: **inject_all**, **withdraw_all**, **stop_all**, **inject**, **withdraw**, **stop**. The **inject**-, **withdraw**- and **stop**-qualifiers do require a further parameter: the number of the pump which is affected. If, for example, a pump is not able to withdraw (e.g. the MMT-pumps) then the according command is simply ignored.

The return-value will be **y** (0x79) with the session-name.

Example:
u inject [2]
...start pump 2 with injection ..
return:
y my_first_test 23.3 45.6 67.9
......................pumps-rates (zero when pumps are stopped) *new*

2.4 Specifying hardware details and parameters

2.4.1 'C' (0x43) - Define camera parameters

(back to 2.1.1)

Define all parameters of a camera needed to specify its functionality. With no exposure-time specified the current values of the camera are returned. The return-value will be **y** (0x79) with the session-name plus the camera parameters realized.

Example:
C eval_cam [4 [80]]
.... name .. exposure time in seconds .. [camera gain].
return:
y my_first_test 0.045 75
......session......cur. exposure, cur. gain...

2.4.2 'T' (0x54) - Set temperature

(back to 2.1.1)

Set a certain temperature. With no temperature specified only the current temperature is returned. The return-value will be **y** (0x79) with the session-name plus the measured temperature of this sensor.

Example:
T biofox_A [45.0]
... name .... temperature in °C ...
return:
y my_first_test 34.4
.....session......curr. temperature... *new*

2.4.3 'a' (0x61) - Set AOTF or light-source intensity

(back to 2.1.1)

Set a certain light-source intensity. With no intensity specified the current intensity value is returned. The return-value will be **y** (0x79) with the session-name plus the valid or measured intensity.

Example:
a biofox_blue [45]
... name .... intensity [0 .. 63] ...
return:
y my_first_test 66
....session........intensity....

2.4.4 'n' (0x6e) - Set norm-duty, active-duty and reference-cycle

(back to 2.1.1)

Set the general electrode parameters. Three qualifiers (**norm** - norm-duty, **active** - active-duty, **ref** - reference-cycle) define what value is to be specified. With no value given only the current setting is returned. The return-value will be **y** (0x79) with the session-name plus the measured or define setting.

Example:
n norm [80]
.. norm-duty in %
n active [90]
.. active-duty in %
n ref [400]
.. ref. cycle in 1/10 per milli-second
return:
y my_first_test val
.......session......according value....

2.4.5 'p' (0x70) - Define new pump-parameters

(back to 2.1.1)

Define the pump-parameters. Some qualifiers (**dia** - diameter of syringe, **flow** - flow-rate of fluid, **unit** - volume per time, **comment**) define what acutally is specified. If the according qualifier does not make sense for the particular pump hardware (e.g. the flow-unit with the MMT-pumps is always ul/h) it is simply ignored. If no value is given then the current value is returned. The return-value will be **y** (0x79) with the session-name plus the specified or measured value.

Example:
p 1 **dia** [4.61]
.. nr of pump .. qualifier .. diameter of the syringe in mm.
p 2 **unit** [ul/h]
.. nr of pump .. qualifier .. the flow-unit is in this case give as micro-liters per hour

p 2 **flow** [50]
.. nr of pump .. qualifier .. a flow-rate of 50 ul/h specified
p 2 **comment** [“test oil”]
return:
y my_first_test val
.......session.....current value.... *new*

2.4.6 ’x’ (0x78) - Move the xy-table to a certain position

(back to 2.1.1)
Set position of the xy-table is given as a xy-pair of coordinates. The command without any parameters returns the current position of the xy-table. The return-value will be **y** (0x79) with the session-name plus the measured position of the xy-table.
Example:
x 2 x 44394 y 22000
... nr. of axes .. name + position-x name + position-y in micro-meter
return:
y my_first_test pos. x pos. y
......session......coordinate of current table position.

2.4.7 ’y’ (0x79) - Move the xy-table and z-stage to a predefined position

(back to 2.1.1)
This command is in the sense different to the one given in sec. 2.4.6, because now not the absolute position vektor is given, instead a name from one of the predefined positions. These predefined positions are specified with the graphical user-interface of the ng_biopro-software. If the named position is not available the current named position is returned. The return-value will be **y** (0x79) with the session-name plus the active named position.
Example:
y [test_position_a]
.... name ..
return:
y my_first_test test_position_z
......session......actual specified position..

2.4.8 ’z’ (0x7a) - Set position of z-stage

(back to 2.1.1)
Set the height of the zStage to an absolute value. With no absolute value given only the current position is returned. The return-value will be **y** (0x79) with the session-name plus the current position of the zStage.
Example:
y [34]

.. position ..
return:
y my_first_test 35.0
......session......actual position.... *new*

3 Low level communication

3.1 Includes

```
#define _V_MESS_ABORT_ 0x11
#define _MAX_TCP_BUF_  2048
```

3.2 Sending data

```
int ng_senddata(
    ng_ngen *p_ngen,        /* Main NGEN-structure                     */
    int sock,               /* Socket descriptor                       */
    unsigned short desc,    /* Message descripter (only lower byte) */
    int len,                /* Length of message                       */
    char *p_data            /* Pointer to data message                 */
    )
{
    int l;                  /* Length of message                       */
    unsigned char mdesc;    /* A message descriptor is only a byte  */

    mdesc = desc;
    l = htonl(len + 1); /* Do not change len (because */
                        /* of additional write) */
    if (write(sock, &l, sizeof(l)) != sizeof(l))
        printf("Could not write len %d!\n", len);
    else
    {
        if (write(sock, &mdesc, sizeof(mdesc)) != sizeof(mdesc))
            printf("Could not write type 0x%x 0x%x!\n",
                    desc, mdesc);
        else
        {
            if(len > 0)
            {
                /*----- We do send something -----*/
                if (write(sock, (char *) p_data, len) != len)
                    printf("Could not write %d bytes!\n", len);
            }
        }
    }
    return(0);
}
```

3.3 Receiving data

Function read_sock:

```
static int read_sock(
    int sock,  /* From where to read                      */
    char *p_buf, /* Pointer to buffer                     */
    int len    /* Requested length in bytes               */
    )
{
    int ist;   /* Current number of characters read      */
    int rest;  /* Still missing number of characters     */
    int n;     /* Number of characters just read         */

    if(len <= 0)
       printf("Implausible length %d requested!\n", len);
    ist = 0;
    p_buf[0] = 0;
    do
    {
        rest = len - ist;
        n = read(sock, &p_buf[ist], rest);
        if(n == 0)
        {
            perror("Connection in read_sock closed!");
            return(0);  /* Connection has been closed */

        }
        else if(n < 0)
        {
            perror("Read in read_sock failed!");
            return(0);
        }
        ist = ist + n;
    } while (ist < len);

    return(ist);
}
```

and function ng_recdata:

```
int ng_recdata(
    int sock,              /* Socket descriptor                   */
    char **pp_ret          /* Returned data                       */
    )
{
    int l0, len;            /* Length of message                    */
    char *p_buf;            /* Intermediate storage                 */
    int n,ist;              /* Counters                             */

    len = 0;
    p_buf = NULL;
    l0 = 0;
    n = read_sock(sock, (char *) &l0, sizeof(l0));
    if(n == sizeof(l0))
    {
        /*----- We got something -----*/
        len = ntohl(l0);
        if(len < 0 || len >= _MAX_TCP_BUF_ - 1)
            printf("Message l0 = 0x%x length = %d has illegal value!\n",
                   l0, len)

        if(len > 0)
        {
            p_buf = malloc(len + 1);
            ist = read_sock(sock, p_buf, len);
            if (ist != len)
            {
                printf("Got only %d from %d bytes!\n", ist, len);
                len = 1;
                free(p_buf);
                p_buf = malloc(len + 1);
                p_buf[0] = _V_MESS_ABORT_;
                p_buf[1] = 0;
            }
            else
            {
                /*----- A normal message -----*/
                p_buf[ist] = 0;
            }
        }
    }
    else if(n == 0)
    {
        /*----- The other side vanished! -----*/
        len = 1;
        p_buf = malloc(len + 1);
        p_buf[0] = _V_MESS_ABORT_;
        p_buf[1] = 0;
    }
    else
    {
        printf("strange data n = %d l0 = %d 0x%x len = %d!\n",
                n, l0, l0, len):
    }

    *pp_ret = p_buf;
    return(len);
}
```

4 Miscellaneous

4.1 Revision history

0.42:	Changed return of position of xyz-table to letter 'x'. Changed return of temperature sensor to 'S'.
0.43:	Fixed a few bugs, especially in section 2.1.
0.44:	Added comment-qualifier in section 2.4.5, added command **'f'** with section 2.3.6,
0.45:	bug-fix: added reference 'e' in section 2.1, added reference 'y' in section 2.1, changed command syntax of 'x' (2.4.6), removed the command 'z' because it can be replaced with functionality of 'x' (2.4.6)