User's-Manual of BioPRO-Software

Uwe Tangen

August 15, 2008

Ruhr-University Bochum c/o BMZ Otto-Hahn-Str. 15 44227 Dortmund

Version 0.3

Contents

Chapter 1

Introduction

This manual is meant for normal users and software-developers at the same time using the electronically controlled microfluidic chips - BioModules.

The overall goal for this software is to facilitate and operate these microfluidic chips with hundred or more electrodes and many different variants of fluidic designs. It has to be admitted that this task is difficult and that the average reader should not expect to utilize the full power of the software and these special chips in a few minutes of training. To illustrate the complexity of the software: about two hundred thousands lines of code and several man-years of development are behind what is described in this manual.

The software, so far, is running on Linux, MacOS-X and potentially every other Unix-like operating system. Binaries can be provided on short notice (given that the according computer plus compiler are available).

Essentially three parts are provided with this software, a Perl-script which operates the ng_biopro-software, several design-files for fluidics, electronics and FPGA (field programmable gate array).

1.1 Installation instructions

Currently, May 2007, the installation of the software is not as easy as a typical Windows-user would expect. An experienced Unix-manager should install this software.

Two environment variables are required to be set in advance: HOME which points to the home-directory of the user running this software and NGEN_DESIGN pointing to the design-tree with all binaries and design-files, see section 1.1.2. Furthermore it is expected that the user has a directory in his executable search path which is owned by himself, which typically is $HOME/bin.

1.1.1 Operating system requirements

The operating system requirements for Linux (Version 9.0 and higher) are available in most of the distributions. The software uses X11, GTK-+2 (pango, atk, freetype, fontconfig ...), readline and Perl-TK. With MacOS things are more complicated, because the programming environment by default usually only supports X11. Perl-TK and GTK-+2 have to be compiled and installed separately. This can become quite tedious and cumbersome and is usually only be manageable by an experienced system-manager, the Xcode utilities and the X11-development-environment from Apple have to be installed prior, see 6.1.

1.1.2 Directory trees and installation

A default installation would look like this:

$HOME an environment-variable containing the path of the user's-data area, $HOME/bin, a directory which is in the search path for executables (the name 'bin' is not mandatory, every other name would suffice also), $HOME/sessions, a directory which will be created during installation and which serves as a root for sessions using this software (the name 'sessions' is hard coded in the starting script 'startbio') and last but not least the design-environment, $HOME/bioenv, which contains all binaries and design-files usable by the software (the name 'bioenv' is not mandatory but has to be defined in the environment variable NGEN_DESIGN). Updates of the software will change or add files below this directory.

Further directories which are provided during the installation-procedure are $HOME/sessions/session_template (a directory which contains native design-files used to start the software), $NGEN_DESIGN/bin, the directory tree containing all available binaries of that revision (e.g. ng_biopro_i686), $NGEN_DESIGN/design, a directory tree containing all design-files available (you can add further design files at all times if you obey certain naming conventions, see section 3.9) and $NGEN_DESIGN/icons, the full collections of icons used by the software. The current version of the software is available on line in the members-section of the PACE-website at:

PACE : Working_Area : Movement_simulator : BioPRO_Software : bioenv

Three container-files are provided: bioenv_shipped.tar.gz, sessions_shipped.tar.gz, bioenv_fluidic.tar.gz. All three of them should be unpacked in the same directory.

A typical installation comprises the following steps:

1. unpack the tar-archive in the home-directory or where ever the user has write access to.

2. let the environment variable NGEN_DESIGN point to the newly created sub-directory 'bioenv'. Verify that HOME points to the users home-directory. With bourne-shells the command looks like: 'export NGEN_DESIGN=/home/myuser/bioenv/'.

3. create a link 'startbio' in a directory which is in the user's search path for executables (e.g. ln -s $NGEN_DESIGN/bin/startbio $HOME/bin/startbio). The file 'startbio' at the web-page is usually the most recent one. It can be copied into 'bioenv/bin' and replace the already available 'startbio'. Please be aware that site-specific changes are made in 'startbio' and have to be updated with a new 'startbio'-file installed.

4. Run the script 'startbio' and provide the missing libraries which are mentioned as error-messages at several levels of the startup-procedure (this step probably has to be accompanied by someone who has root-access (super-user) and knows the concept of libraries).

1.2 Running the software

The usual mode of operation is executing the script 'startbio'. This should be done once per experiments-day. It is not important at which specific place this script is executed. All file-creations and changes are done below $HOME/sessions. Calling 'startbio' ensures that the user is able to choose from the designs available, change further runtime-parameters and gets a full log of all operations during the session. If 'startbio' is called the first time in a day it creates a new session_{date}-folder and all logs and data-files are stored in this directory.

At start-up of 'startbio' a Perl-TK window opens and lets the user choose parameters. Take your time and play with these parameters. Essentially the user is asked about the hardware-setup, whether and how many pumps are attached to the computer, what the interface between computer and the BioModule is, whether a new sessions-directory is to be created (if it is already available) and whether the BioModul should be configured or not. See section 2.1 for a detailed description of all the features of this interface.

With the button 'Start' the real ng_biopro-software is launched. With the button 'Exit' the Perl-TK script 'startbio' finishes and gives the control back to the user. When ng_biopro is launched the 'startbio'-scripts goes suspend, meaning, it is no longer reacting to user-events.

With launching ng_biopro a big Xterm-window is opened executing the ng_biopro-software with the attached runtime-parameters. Through this Xterm-window the ng_biopro-software can be controlled on a console-level, a simple prompt using the readline-library, is provided, see section 3.7 for an explanation of all possible commands and features. During the boot process of the ng_biopro-software specified design-files are read-in, contact is established to the defined hardware-resources and an X-windows based graphical user-interface is opened, see chapter 2 for further details.

A multitude of error-checks are undertaken and it might very well be that a lot of error-messages do occur if something with the hardware or the designs fail. When the fluidic- and the electronic design is visible in one window and the current camera image in another window as well the software is operational.

1.3 Feedback-loops and level-one elements

The major objective of the integrated system environment 'ng_biopro' is to provide the ability on using electronics to directly control the biochemistry happening in the micro-fluidic channels. It is the central feature of the PACE project concerning its embedding in the IST-framework.

This objective requires full access to the images produced by the camera and as much control as possible on all actuators available in the system. With the Bio@Fox-box many of these actuators can be accessed.

The direct electronic control of the biochemistry immediately allows the creation of a multitude of feedback-loops in the system. The fluorescence images from the camera are evaluated at certain regions of interest (ROI), intensity-values are calculated and due to several already defined or yet to be defined regulatory elements (all written in software) commands for the actuators (electrodes, xy-table, aotf, filters etc.) are derived.

Currently, six different simple-feedback-loop elements are defined: CATCHER, BARRIER, NEURON, TRAP, TRIGGER and SPREADER. You can find specific descriptions of these 'level-one-elements' in chapter 4.

Chapter 2

User-Interface

The user-interfaces comprises several levels of interaction with the system. As a preparation on what the software should do a Perl-TK script ('startbio') asks for the hardware to be used during the session.

A low-level interface (console based) gives the user access to special features and provides debugging facilities in case of failures. This low-level interface also provides communication with the program 'ng_biopro' in case of a missing graphical environment.

The third user-interface level is the graphical user-interface with click-able buttons and design-data presented. Further interfaces exist which allow the program to communicate with other programs, or to setup a client-server-operation scheme. All these interfaces are described in the sequel.

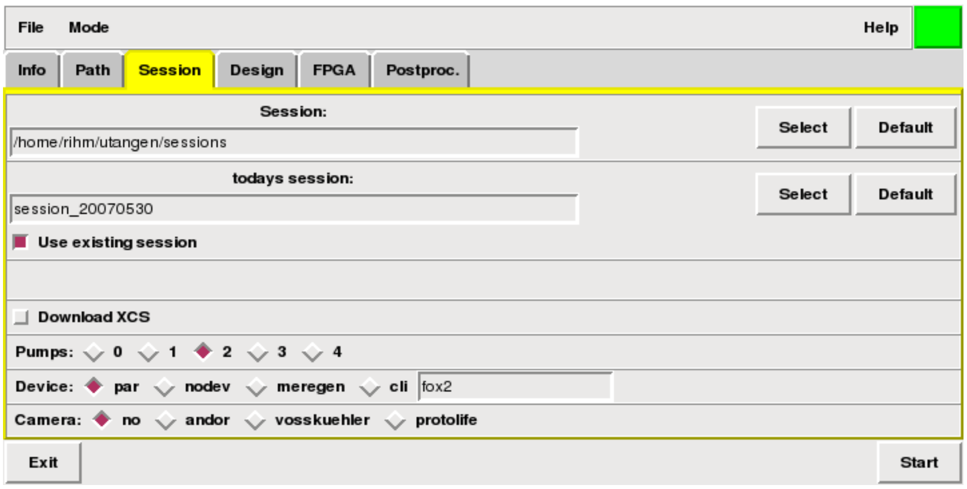
2.1 Perl-TK user-interface ('startbio')

The Perl-TK user-interface serves two purposes. The first one is to provide an easy to use navigation tool allowing the user to choose between different design variants and to tell the software which hardware configuration is to be used. In addition, path-information is defined including a simple session-management and logging. The second purpose is to provide a user-editable script which allows customization to the users' need – of course a person able to handle Perl-scripts is needed for this feature.

The ng_biopro-software uses several different input-interfaces. Two of them are utilized by this 'startbio'-script. For runtime parameters of the program, see section 3.6 and for the scripting interface of the ng_biopro-software, see section 3.7.

When starting the 'startbio'-script the 'Session-tab' is displayed.

2.1.1 Session-tab



Session: /home/rihm/utangen/sessions

todays session: session_20070530

[✓ Use existing session]

Download XCS

Pumps: 0 1 [✓ 2] 3 4

Device: [✓ par] nodev meregen cli fox2

Camera: [✓ no] andor vosskuehler protolife

This is the first window seen after the 'startbio'-script has been launched. The user is allowed to define a directory in which all the logging and further definitions of the software behaviors are saved. The two paths in the upper half of the window are preset with default values which might be changed via the buttons 'Select' on the right side.

The directory chosen in the first path-entry has to exist already, it is thought as the general container of all experiments done with this software. The second will be created, if not already existent. By switching off the check-box 'Use existing session' a new directory is being created each time the ng_biopro-software is started anew.

A further check-box 'Download XCS' is provided to let the user decide whether a plugged-in BioModule should be configured or not. If no BioModule is attached this check-box avoids lots of error-messages and the waiting on hardware to respond correctly. This check box is working only if the BioModule is configured from the PC starting the 'startbio'-script. In all other cases, especially when using the Bio@Fox-box, the chip will always be configured.

The ng_biopro-software supports an arbitrary number of pumps to be addressed. This 'startbio'-scripts lets the user to choose between zero and four different pumps. The control window of these selected pumps is only visible on request, see section 2.5.1 for a further explanation.

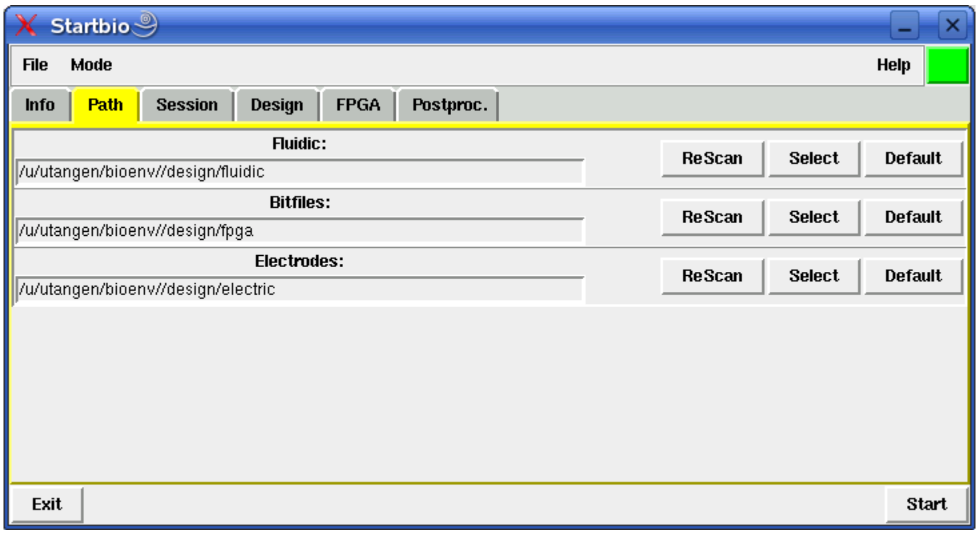
Due to the multitude of different hardware types, and many of them only with simple serial-interfaces, a real plug-and-play philosophy cannot be provided. Instead, the user is required to choose the hardware components by clicking at the appropriate check-boxs given as Device: and Camera: area. In the case shown above, 'par' means use of the parallel-port of the computer.

This is for users connecting the BioModule directly to the parallel-port. The parallel-port must be configured in IEEE1284_MODE_COMPAT mode, see section 3.1.7 for further specifications.

If the software is operated with a MereGen-board, which is controlling large parts of the setup, then the user is required to check the 'meregen' button. With no BioModule attached 'nodev' should be checked. Checking the button 'cli' (client) and providing the appropriate name of the Bio@Fox-box (in principle IP-addresses might be sufficient as well, but some peculiar socket-issues will result sometimes in error-messages if not given a real name) opens the connection to the according Bio@Fox-controller.

In principle, the ng_biopro-software supports several different types of cameras. Check the button 'no' if no camera is currently available. If an Andor-camera is provided check the button 'andor' and have a look into section 3.1.1.2 for further information on the software-development-kit used in this implementation. Even though firewire (IEEE 1394) is a generally defined camera-bus each firewire-camera requires special control-flags to be set. The current ng_biopro-software supports the Vosskuehler camera. With a Vosskuehler camera attached, check the button 'vosskuehler'. With 'protolife' the driver for a Cascade-II camera from Princeton Instruments is selected.

2.1.2 Path-tab



Fluidic:
/u/utangen/bioenv//design/fluidic

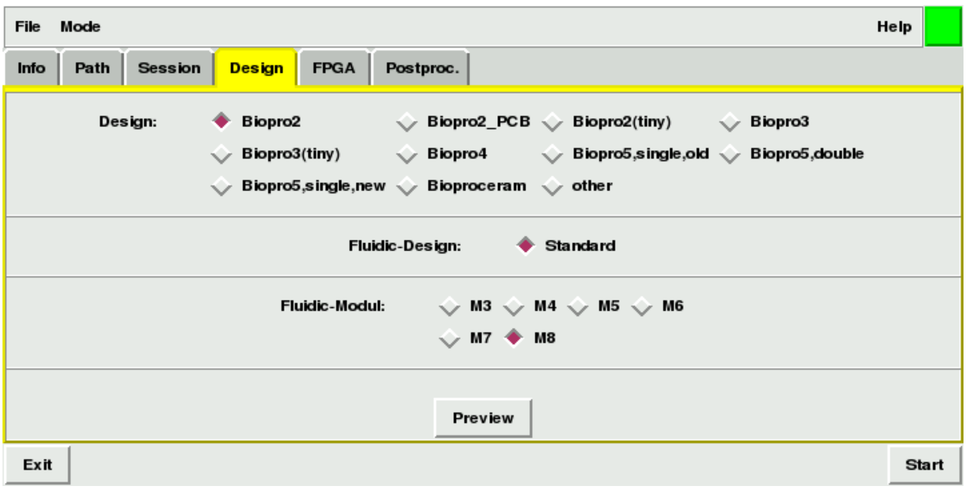
Bitfiles:
/u/utangen/bioenv//design/fpga

Electrodes:
/u/utangen/bioenv//design/electric

If the user doesn't have a special environment configured the paths given in this section probably would not need any modifications. In case you want to test different designs, which are not given in the standard installation, these paths let you define what electrode-designs to be used, where the bitstreams for the FPGAs to be found and what the required other fluidic-variants are.

The user is allowed to specify different design-directory-paths using the 'Select'-button on the right side. If something changed in these directories during runtime of the 'startbio'-script the user might press the 'ReScan'-button. The 'startbio'-script then regenerates the scripts used when launching the ng_biopro-software, see section 3.7.

2.1.3 Design-tab



Design: [✓ Biopro2] Biopro2_PCB Biopro2(tiny) Biopro3 Biopro3(tiny) Biopro4 Biopro5,single,old Biopro5,double Biopro5,single,new Bioproceram other

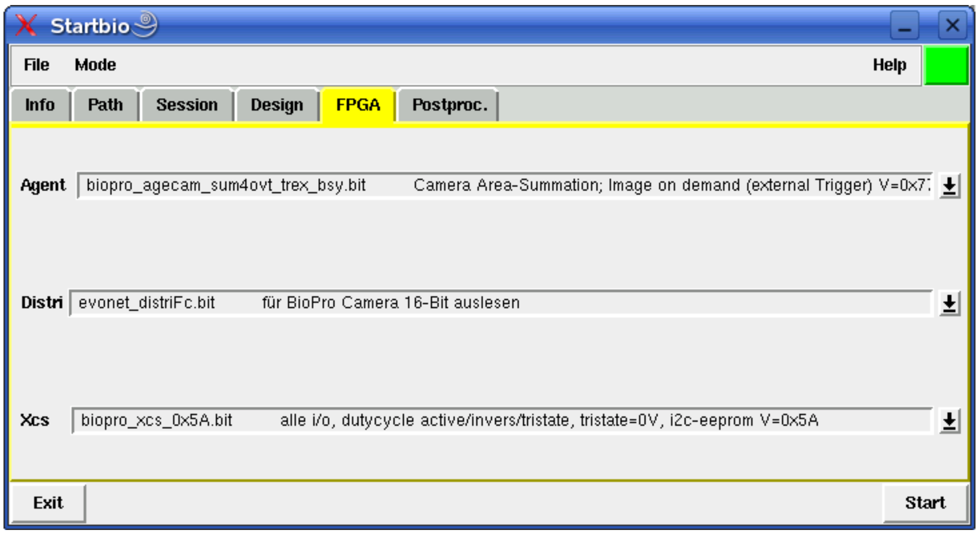
Fluidic-Design: [✓ Standard]

Fluidic-Modul: M3 M4 M5 M6 M7 [✓ M8]

Depending on the BioModule attached and the concrete fluidic-variant realized, the user has to specify the design, e.g. 'Biopro3'. After defining the correct type, the Perl-TK-interface provides the available fluidic-designs, in this example 'Bubble', 'Fan', 'SegmFlow' or 'Standard'. With the fluidic design chosen, e.g. 'Fan', a further section becomes available: the fluidic-module - which shows check-boxes of the possible detailed variants of the 'Fan'-reactor design in this case. In total several dozens different fluidic-designs are available. The preview-button is especially useful because it allows the user to look at the designs quickly and to find out which biomodule really is mounted and what the specific design-name is.

These specifications have to be done, because otherwise the user almost certainly will be not able to specify which electrodes in the fluidic-channels are effective and have contact to fluids.

2.1.4 FPGAs and bit-files



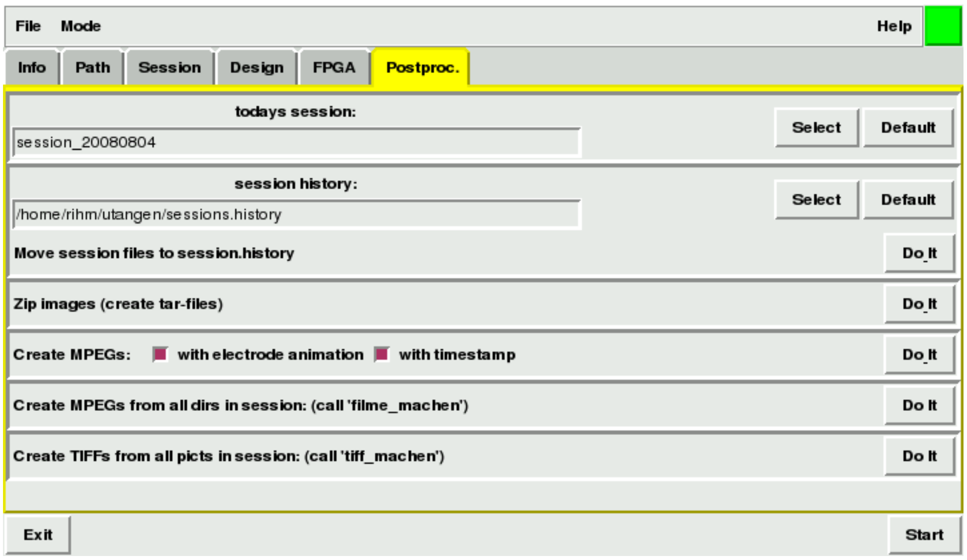
Agent biopro_agecam_sum4ovt_trex_bsy.bit Camera Area-Summation; Image on demand (external Trigger) V=0x7

Distri evonet_distriFc.bit für BioPro Camera 16-Bit auslesen

Xcs biopro_xcs_0x5A.bit alle i/o, dutycycle active/invers/tristate, tristate=0V, i2c-eeprom V=0x5A

Depending on the attached hardware this tab allows the user to specify the bitstreams used for the configuration of the FPGAs. The two upper sections in this tab are only needed when the MereGen-board is used as the major hardware control unit. The lower section shows the currently available bit-files used to configure the FPGA of the BioModule.

2.1.5 Post processing



todays session:
session_20080804

session history:
/home/rihm/utangen/sessions.history

Move session files to session.history

Zip images (create tar-files)

Create MPEGs: with electrode animation with timestamp

Create MPEGs from all dirs in session: (call 'filme_machen')

Create TIFFs from all picts in session: (call 'tiff_machen')

Some simple standard processing steps can be undertaken after experiments have been conducted. For example, the videos stored, are written as pgm-files and probably needed to be converted into a mpeg-movie, or they should be compressed to save disk-space.

2.2 Console terminal

A basic user-interface is given with a console window. This console is created as an Xterm-window after pressing the 'Start'-button in the Perl-TK-script or the ng_biopro-software is started itself from a console without any Perl-TK-script. The sole function of the Perl-TK-script is to provide the wished for environment. This environment is defined via two interfaces, firstly the program's runtime-parameters and secondly, with script-files, which have been created by the Perl-TK-script. Runtime-parameters and these script-files are open. They provide all the necessary data for operation in case of debugging or a non-graphical user-interface.

Many additional commands are available on this low-level-interface which are not needed in typical cases and thus are almost not documented in this user-manual.

Especially when using the ng_biopro-software in server-mode the main user-interface is provided with the ng_biopro-software working in client-mode somewhere else in the world. This client-server-communication is realized via a socket-based TCP/IP communication channel, see section 3.4.

2.3 Windows as user-interface

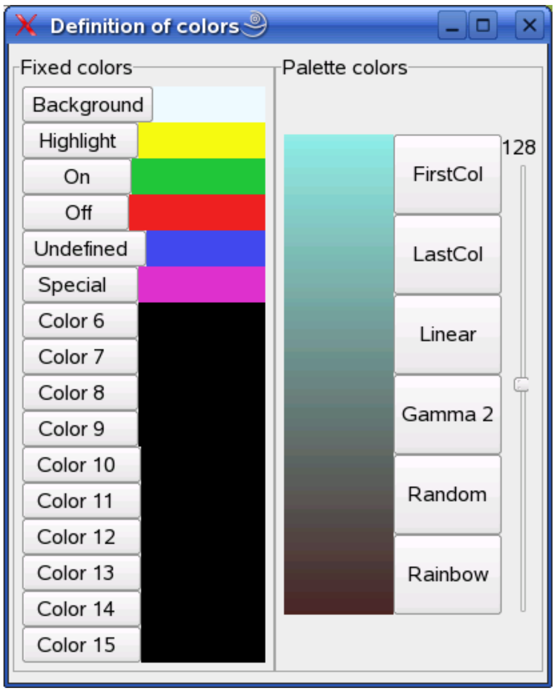
The major communication interface with the ng_biopro-software is realized via typically two different graphical windows. One of these provide an on line-camera screen and the other the electronic- and fluidic design to allow the user to navigate to areas of interest and to provide her with full low-level control of the electrodes.

Each of these windows has different functionalities comprised in a menu-area at the top of each window. Certain icons (with corresponding short texts) usually raise new windows with specialized functions. Each of these top-level-windows in addition contain two buttons each, for specifying colors and text-fonts.

2.3.1 Colors and fonts etc.

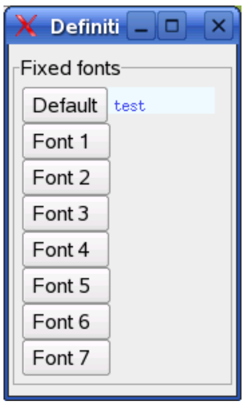
Defining colors and fonts is done via two popup-windows shown below. The icon shown represents the according functionality in the menu-bar of these top-level-windows.

- Define and change colors

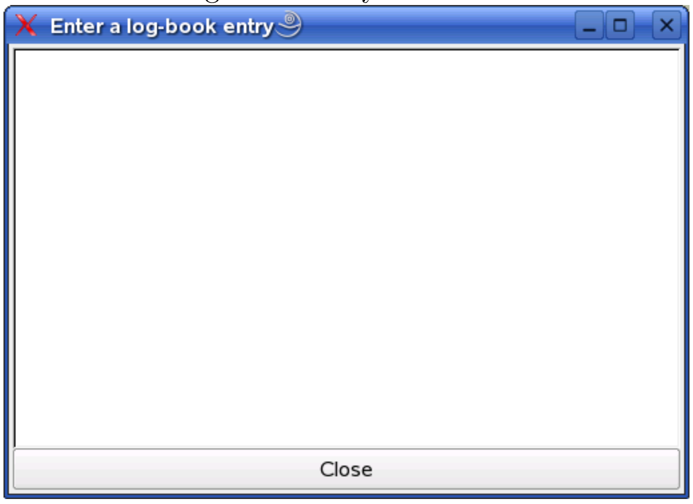


There are two different types of colors to be specified. Discrete colors which are used for selected items, switched on- or off-states of electrodes, sensor elements and fluidic-channel representations. 16 different colors can be specified in this way. What color is used for what functionality in the user-interface is currently hard-coded in the ng_biopro-software and is not customizable.
The second type of colors is a palette which might be a definable gradient, a random collection of colors or a rainbow. This palette is effective for the camera-window where many different colors are needed. A suitably defined color-space allows the experimenter to enhance certain aspects of the camera-screen. The colors defined here only affect what the user sees on the display and do not influence what is stored on disk as a sequence of images, see section 2.4.2 for further information on video-sequences.

- Load fonts and map them for textual output. You only need to provide a suitable name for the font. With xfontsel you will find out which fonts are currently available.



- Flip drawing-area horizontally
  Only the view is flipped. The internal calculations are not affected.
- Flip drawing-area vertically
  Only the view is flipped. The internal calculations are not affected.
- Add a log-book entry



The log-book-entry popup-window gives the experimenter the ability to add notes at any time during the experiment. It is just a very simple editable text field. After pressing the 'Close'-button the contents of the window is printed in the log-book of this experiment accompanied by two time-stamps (one for opening the log-book-entry-form and the second for the time of closing the window).

2.3.2 Zooming

Zooming is only possible in the design-window. There is no restriction on the depth of zooming. The procedure to zoom into a region of the design is two draw a rectangular area with the left mouse-button (see section 2.3.3 for notations) and then after releasing the left mouse-button clicking the middle button. Zooming out is done with the right button, just clicking it at the design window. Then the zoom jumps back to the former rectangular area. The sketch of this area is shown in addition. Zooming into this now old region is done as before, just clicking the middle mouse-button.

There is no limit (apart from memory considerations) in how many zoom-levels can be created.

The once created rectangular area can be modified. Pressing the left-mouse-button inside the area means moving it, pressing the left-mouse-button in direct vicinity of the edges, see 2.3.3 for further details, means changing size.

Creating rectangles with the left-mouse-button in other top-level-windows is also possible but zooming into these areas is not implemented, see 2.3.3.

2.3.3 Mouse buttons and cursor-keys

In general, the ng_biopro-software expects a three-button mouse at least. This might be a problem for Windows oriented two-button-mouse-users as well as for MAC-OS oriented single-button mouse users. In the sequel the discussion about left and right is based on the assumption that the mouse is used by a right-handed user. For left-handed users and thus exchanged button-meanings left and right are reversed.

2.3.3.1 Left-mouse button

Usually the left mouse button serves for selecting objects, drawing sensor-areas and zoom-areas, see section 2.3.2 for details on zooming. There are essentially four different types of action when using the left-mouse-button:

- Press-and-hold-and-move
  With this operation the user creates a rectangular area. Currently, the minimum-size of such a rectangular area is 10 pixels. After releasing the left-mouse-button the rectangular area remains visible. The further meaning of the area is then determined by the next mouse-click.

- Clicking the left mouse-button
  Several different possible reactions occur, depending on the context.

Usually, clicking the left-mouse-button means selecting what is below the pointer. If there are several types of selectable objects a popup-menu lets the user choose which of these types are to be selected. Selected objects are highlighted. The highlighting color can be changed, see paragraph 2.3.1 on how to do this.
Already selected objects are deselected. If this should not happen, then press the CTRL-button at the same time when the clicking is done.
If already a rectangular area has been drawn, the effective selecting-area of the mouse-button is increased to this size, meaning that all objects inside this rectangle are selected. Again a popup-menu appears, if different types of objects are to be selected. After selection, the rectangular area disappears.
There is a special feature for electrodes (or pins) which lets the user define the polarity of these electrodes. When selecting an electrode the default polarity is positive (usually 3.3V with Spartan-XL FPGAs). Pressing and halting the SHIFT-button before clicking on the electrode reverses the polarity, in our example resulting in 0V potential at the electrode. Just clicking on the electrodes does not actually mean activation of the electrodes. They stay in the state possessed before, see section 2.3.5 for further details.
Every simple object has one selection point. This selection point is at the center of the electrodes and at the upper left edge of the sensors, see section 2.3.5 for details. If sensors or electrodes are overlapping unfavorably then zooming into the region usually helps to distinguish these objects.

- Pressing the left mouse-button:
  Some objects, mostly sensors, are allowed to be moved around. The usual procedure to accomplish this, is to select the object, you see the highlighted color, and then, while pressing the SHIFT-button in addition, dragging the object over the screen. Typically you will see a rectangular area emerging, which is removed after the object has been moved.
  With a rectangle is existing, the behavior depends on the position of the mouse pointer. If the mouse-pointer is well inside the rectangular area then the user is able to move this rectangular area by dragging the pointer while pressing the left mouse-button.
  If the mouse-pointer is fairly precise at an edge or a vertex of the rectangular area then the size of that rectangular area can be changed by dragging the pointer while pressing the left mouse-button.

- Double clicking the left mouse-button:
  Currently, no functionality behind this event.

2.3.3.2 The center mouse-button:

- Zooming-in into an already given rectangular area, when clicking. It is not necessary to have the pointer inside this rectangular area, just clicking is sufficient. All other possible events, pressing, double-clicking and in conjunction with other keys are not implemented.

- Pressing the shift-button while clicking the center-mouse-button and being located over a selectable object results in a bunch of further information given in the session-log. This information is meant for debugging strange behavior and is usually not needed.

2.3.3.3 The right mouse-button typically means inversion of operation.

- When clicking the right mouse-button inside a zoom-state then a zoom-out-operation occurs. After zoom-out the rectangular area of the former zoom-in-rectangle is visible, see section 2.3.2 for further details. If a rectangular area is visible then this area is removed.

- If no-zoom-out is possible and no rectangular area visible then all already selected objects are deselected.

- Double click of the right mouse-button always means deselection of all selected objects.

2.3.4 Data base, export and import of objects

The ng_biopro-software is equipped with an integrated database. The main purpose for this database is to store all the complex configuration options and user-defined control-structures to allow for a seaming-less continuation in case of halting the program. This database is either stored in a binary container, e.g. 'ng_biopro_i686_Vbase.db_0' or a human-readable ASCII-file 'close.Vbase'. The database itself is a distributed object-oriented database-system. Especially for client-server operations the communication between client and server is mediated via this database.

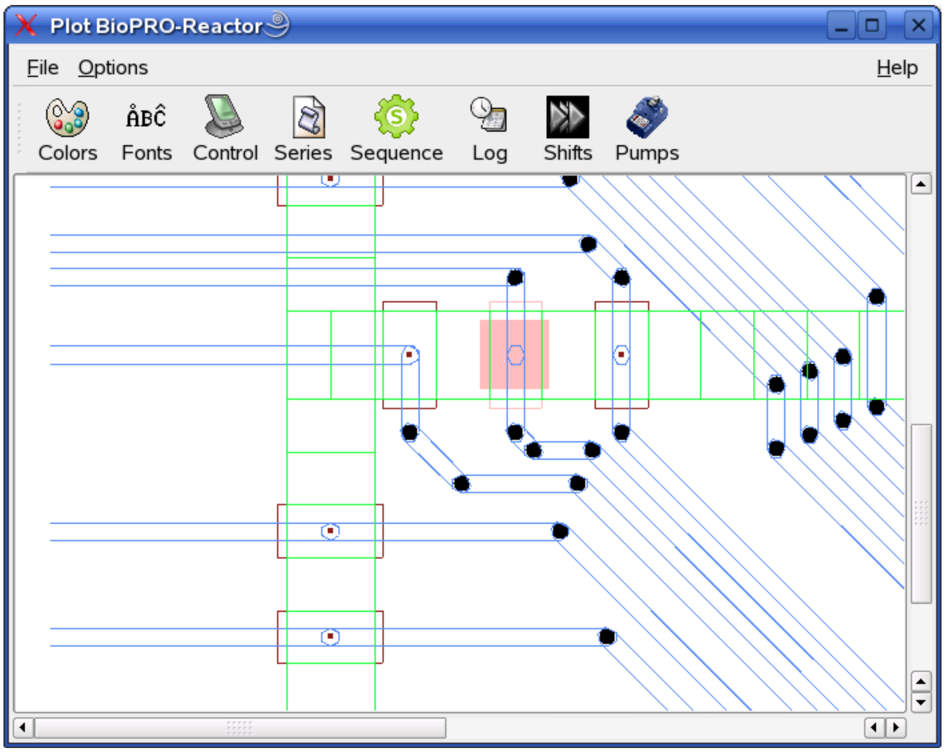
For reasons as backup and debugging it is possible to export the internal database into a human-readable ASCII-form. The syntax is LISP-like with all objects and their corresponding data is written in clear-text as is shown in the file close.Vbase. Of course, importing such an ASCII-database is also possible. Import and export is available below the FILE-drop-down-menu in the upper menu-bar.

Remark: the binary form of the database is neutral to little- and big-endian machines.

2.3.5 Sensors and actors

Besides the pure hardware control like pumps, camera and shutters the electronic control of electrodes is at the heart of the ng_biopro-software. Sensors in the sense of this section have to be distinguished from e.g. temperature-sensors which are also controlled by this software but do have a clear physical representation. The sensors in this section and if not otherwise noted in the whole document are simple rectangular areas in the top-level windows.

In the camera-window these rectangular areas are regions of intensity integration. All intensity values of pixels in this region are summed up and divided by the number of pixels. Arbitrary many sensors (in the current version is an upper limit of 512 active sensors) can be defined and used to derive decisions on whether to switch-on electrodes or not. The sensors in the design-window are exactly the same with only two exceptions, firstly, they are pre-generated and placed along interesting areas of the fluidic-channels (this eases the selection considerably). And secondly, the intensity values are only available after camera- and design-window had been synchronized. How this is done is explained in section 2.4.5.

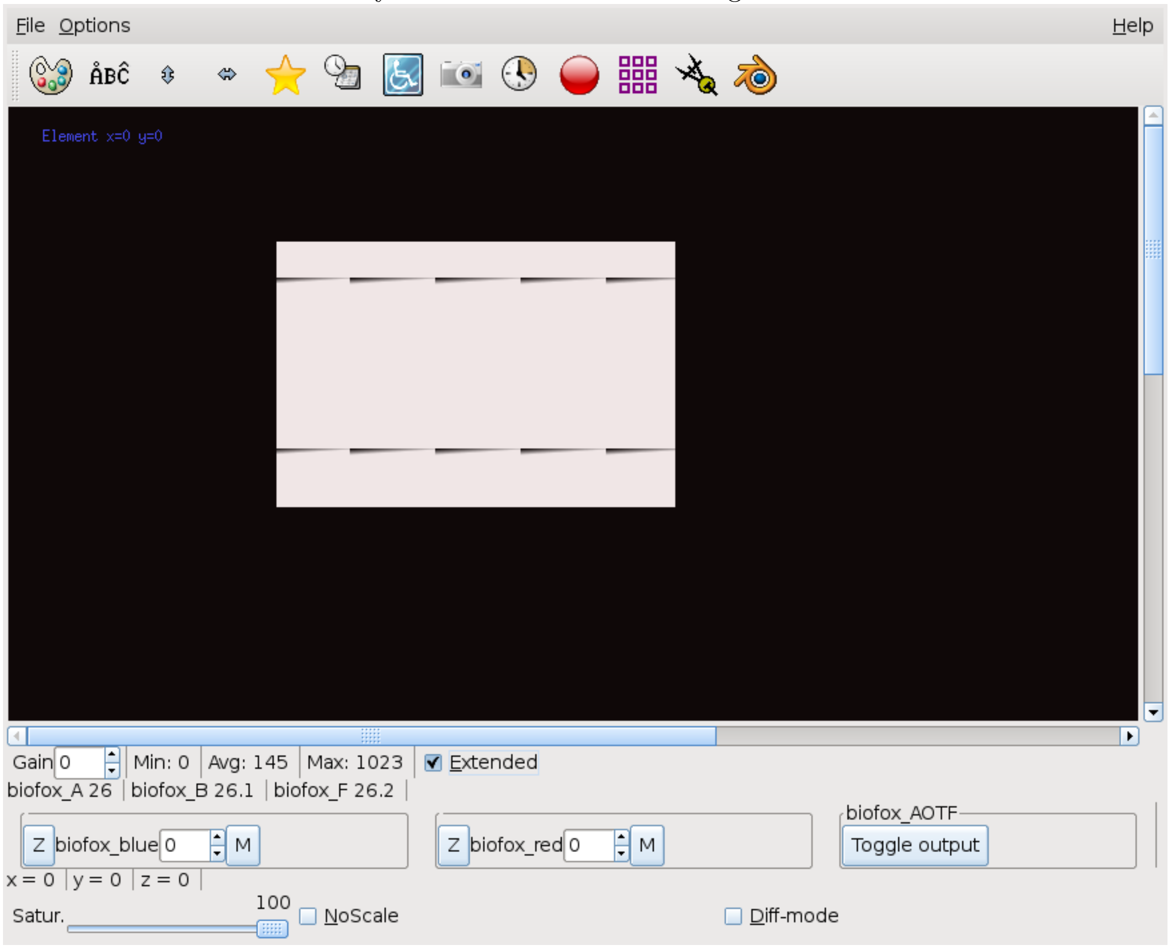


Actors on the other side are simple electrodes which are electrically accessible for the fluids in the channels. Not all electrodes are always accessible because certain fluidic-designs do not allow arbitrary dense channel-networks. Though in the image above not only sensors (green rectangular areas) and actors (brown rectangular areas) are shown but also blue traces which actually are wires connecting the electrodes with the controlling FPGA. The galvanic accessibility is only given for the electrodes. All other parts of the design are covered with an isolating SiN_2- or SiO_2 layer. Of course the isolation-layer is quite thin but for the frequencies and areas used here no visible influence during the experiments can be observed.

2.4 Camera window

The camera-window is the first top-level window. Essentially, it is a combination of four parts: a menu-bar with two drop-down-menus, File and Options and a Help-button, an icon-menu-bar with often used functions, a drawing-area with the cameras output and some status-information and refinement knobs from the camera-control.

Generally, the camera-features used here are optimized in advance for the task given and the user is only confronted with the most important options. The camera-output shown in this window is only meant for cursory observation and not for high-definition precise microscopic views. High-precision observations with the camera are reserved for programs specialized on that task. Here the focus is on the attachability of a broad range of cameras without the users-need to study all the subtleties of the given hardware.



The scroll-bars at the drawing-area are dimensioned such that in principle the whole camera-output can be viewed by scrolling through the different areas. Please note that the actual images picked up from the camera are only from the visible part on the screen. Non-visible-parts do not enter the circular image-buffer and are thus not stored in the movies taken. When scrolling, certain delays might occur when changing the visible area.

2.4.1 Camera-knobs and status-information

The main focus on the camera-features presented via the ng_biopro-software is to have a generalized interface to a multitude of different cameras and not to get out the most advanced features from a specific camera. Each camera driver has been optimized to provide the relevant information and the really needed fine-adjustments. On the screen this range is mapped onto an adjustable number of colors, e.g. 128 with an emphasize on the lower intensity regions. This might result in an overshooting of the high-intensity areas. In addition, not every image is mapped, but a floating average of six images provides the general intensity scale. The available features are the following:

- **Gain**
  Each camera has a different dynamical range. Many CCD-cameras provide a considerable dynamical range, e.g. 12-bits or even more. An adjustable preamplifier even can increase the sensitivity traded with noise in the image.

- **Exposure time**
  Some cameras in addition allow the specification of an exposure time. The scale shown here is in seconds of exposure. What exactly seconds means depends on the camera. The camera-driver tries to map the special camera-features to allow the definition of an effective exposure-time which is comparable to other camera-types.

2.4.2 Storing video-sequences

The ng_biopro-software maintains a ring-buffer of images taken from the camera output. In the current version this image-buffer comprises 400 images with an exact (in micro-seconds resolution) time-stamp attached.

When pressing the 'Video'-button this ring-buffer is written onto the disk as a series of pgm-files. The name of the files is derived from the current date and time and is sequentially numbered. The pgm-files are not compressed and the pixel-values do have a depth of 16-bit, images are raw unmodified camera data. This means that the visible image which is subjected to a further contrast and brightness operation might significantly differ from what is stored in the pgm-files. This apparent difference is the price for allowing a later precise evaluation of the pgm-files.

With the post-processing procedures mentioned in section 2.1.5, the pgm-files can be compressed losslessly or transformed into mpeg-streams with mapped-in electrodes if the camera-window had been synchronized with the design-window, see section 2.4.5. With special tools these electrodes can be inserted in the mpeg-stream, even later, without having synchronized both windows.

Be aware, this type of storing video-sequences means: storing the history of an interesting event. If the experimenter realizes some interesting behavior he or she might wait a little further and then press the 'Video'-button. It depends on the camera and the effective exposure-time used, of how many seconds of the experiment can be captured via this procedure, e.g. with a frame-rate of 10 images per second and 400 images in the ring-buffer the resulting video-sequence is 40 seconds long.

2.4.2.1 Information appended to each image

Each image produced gets an individual annotation with several informations:

- current time in micro-seconds resolution ('t=')
- all available temperatures ('s=')
- filter-wheel states ('f=')
- light-states (currently light is controlled by an AOTF (acusto optical transmission filter) ('l=')
- the current position of the xy-table and the z-stage ('p=')
- absolute counter of the images taken in this session ('r=')

It follows an example of the data appended to each image:

```
t="1178095092_124578: Wed May 2 10:38:12 2007"
f=" 1 'emission' '1white'"
l=" 0 'biofox_blue' 0"
l=" 1 'biofox_red' 0"
p=" 0 'x =' 2450955"
p=" 1 'y =' -59685"
p=" 2 'PI =' 39985"
s=" 0 'biofox_A' 2645"
s=" 1 'biofox_B' -24645"
s=" 2 'biofox_F' -355"
r=2682
```

2.4.3 Storing snap-shots

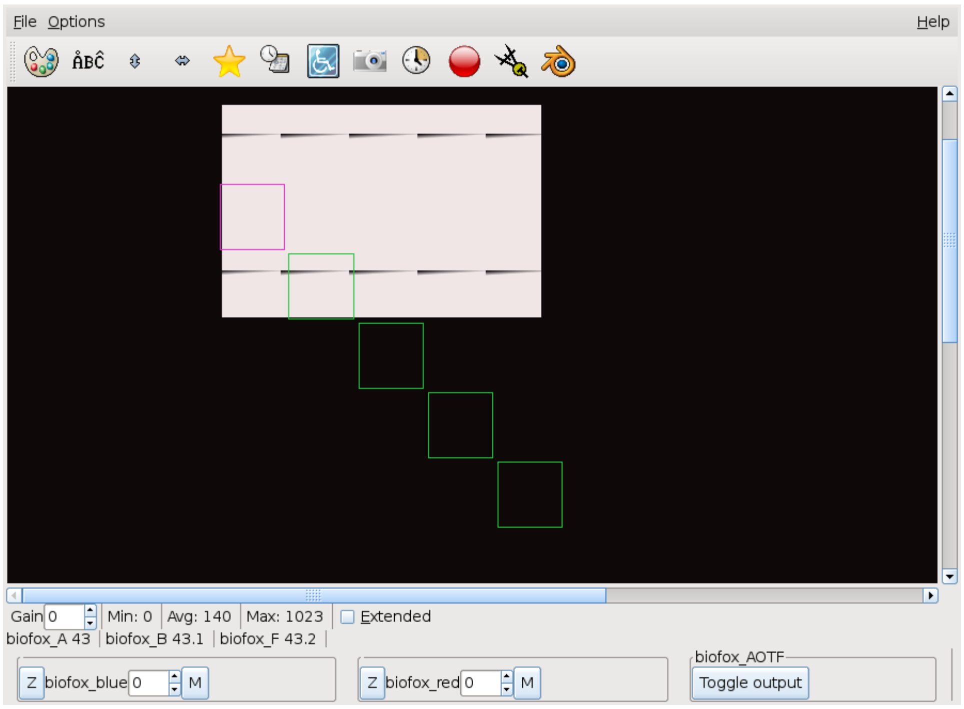
There might be situations where the video-images taken in section2.4.2 do not show all interesting features at the same time and the maximum possible camera-view is needed. For these occasions a full cameras-screen-dump is stored as a pgm-file on disk. As already described in section 2.4.2, the pixel-values are stored in 16-bit pgm-format (P5) and the file-name is derived from the current date and time. *Two different operations are possible: either store a single snapshot on the disk or create a series of snapshots each many seconds apart to allow a long-term observation of what is going on in the system.*

As was described in section 2.4.2.1, additional information for later image-processing is appended at the end of the snapshots.

2.4.4 Long-term measurements

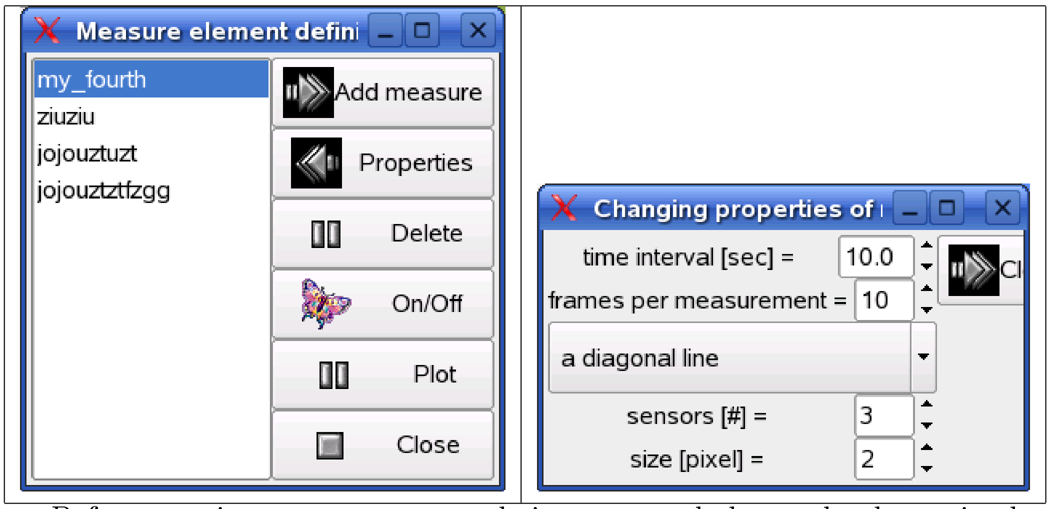
Arbitrary many measurements with recorded intensity-values can be undertaken. These measurements directly work on the images retrieved from the camera. Each measurement consists of a line with at least three rectangular areas. All pixel intensities in each of the areas are summed up and divided by the number of pixels.

2.4.4.1 General principle



An example of such a measurement chain is shown above. The violet square is area zero and the bottom square area four. When starting a measurement an output-file is created, with the current time in the file-name, additional information concerning the position of the measurement-areas is given as well. After the specified time a new measurement is scheduled and all averaged intensity values are written in a single line with the exact time of the measurement stated, see section 2.4.4.5. Furthermore, an ongoing measurement can be dynamically visualized on the screen to allow the experimenter a fast feedback on how the reactions are going, press button 'Plot' in the window 'Measure element definition'.

2.4.4.2 How to create a new measurement chain



my_fourth
ziuziu
jojouztuzt
jojouztztfzgg

time interval [sec] = 10.0
frames per measurement = 10
a diagonal line
sensors [#] = 3
size [pixel] = 2

Before creating a measurement chain a rectangle has to be drawn in the camera window, see section 2.3.3. This rectangle roughly shows the covered area of the future measurement areas. Pressing the button 'Add measure' the user is asked for a name for this new measurement chain. After creating the measurement chain three rectangles (one with color violet and the others two with color green) are drawn in the camera-window, with one of these rectangle covering the upper-left- and the other covering the lower-right-edge of the user-drawn rectangular area. This is the default measurement area, which now can easily be customized to fit the user's need. When pressing the button 'Properties', the right window shown in the figure above, pops up and the user is able to adjust the number of elements in the measurement chain. It is good practice to let this chain cover the regions-of-interest as well as some reference areas which can later be used to calibrate the interesting measurements. Secondly, the individual measurement-area should not be too small to be overwhelmed by noise and not to large to incorporate inhomogenities in the measurements.

2.4.4.3 Changing the measurement chain

Changing the measurement chain can be done as long as this measurement is not active or ongoing. The number of elements in that chain can be adjusted easily when turning at the spinner-wheel denoted with 'sensors #'. Typical suitable values range between five and 20 of these rectangles.

Without changing the size manually the rectangles'-sizes are automatically adapted to cover a maximum area when doing the measurement. When turning the spinner-wheel, called 'size [pixels]' the size of each rectangle is specified in pixels by the value of the spinner-wheel. This manual adjustment might be useful when trying to use unconventional measurement areas.

2.4.4.4 Moving the measurement chain

The measurement chain can be moved into arbitrary parts of the camera-window. Size and orientation can be changed as well.

Moving the chain: Select one or two of the sensors (rectangles) in the middle of the chain, press the SHIFT-button on your keyboard and move the mouse (while pressing the left-mouse-button) into to the direction of your targeted middle position of the chain.

Changing the size and orientation of the chain: As with moving the chain select now one of the outer sensors (rectangles) of the chain (visible through yellow color). Pressing the SHIFT-button and moving with the pressed left-mouse-button towards the target position of this sensor lets the chain stretch or shrink. The opposite boundary sensor remains stationary. In the second step the opposite boundary-sensor can be selected and moved accordingly to its target position.

2.4.4.5 Modifying measurement parameters

There are further parameters which can be changed via using the 'Properties'-window of a certain measurement: 'time interval' and 'frames per measurement'. The time-interval defines the time in seconds between two measurements. Frames per measurement defines the number of measurements done before the average of these measurements is written either on disk and/or updated in the plot-window. Default values are 10 seconds break between two measurements and three measurements for one plot-update (disk-write which is done in any case).

2.4.5 Synchronizing camera's output with the fluidic-design

To let the ng_biopro-software know which area in the design-window exactly corresponds to which area in the camera-window a certain procedure has to be undertaken. Especially when using fluorescence-microscopy edges of electrodes or wires are hard to see and mostly only partially visible. More important, fluid-channels in almost all cases have to be guessed.

The principle alignment-procedure is as follows:

- Pressing the Adjustment-button initiates the alignment procedure.
  The ng_biopro-software highlights two electrodes (actors) in opposite corners of the design-window. It might be that these electrodes are not immediately visible, the user has to scroll the design-window till he or she sees these highlighted actors - typically these are quite big black filled circles.
  The next step is to find the according region in the camera window. Then the user is requested to click on one of the two black circles, which is then immediately vanishing and the user in the sequel clicks at the center of the corresponding electrode in the camera-window. A blue square at that position will blink once to indicate that this position has been acquainted.
  This procedure is repeated with the second black circle.
  With these two reference-points the ng_biopro-software rotates and maps the camera-window and hither-two both windows are aligned. Sensors in the design-window can now be used with the correct intensity values recorded. In addition, the electrodes' outlines are shown in the camera window to let the user verify the correct mapping.

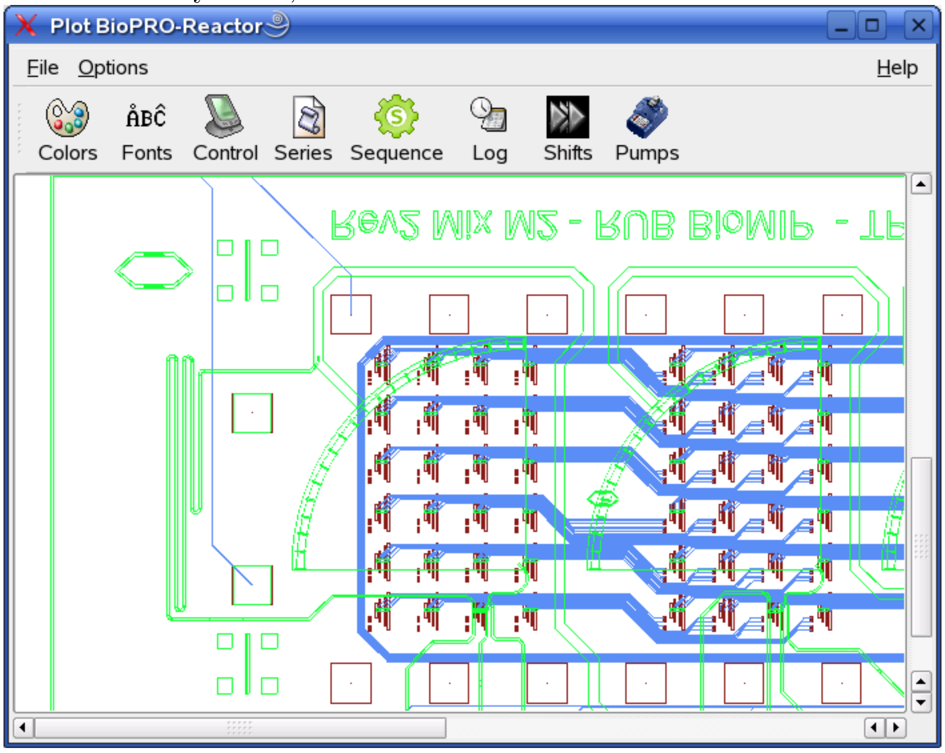
- Rotating the camera window at each image is a time consuming procedure and slows down the camera acquisition. If only the synchronization is needed and not the exact camera view, the on-line-rotation can be relaxed and the camera-view is as before the synchronization procedure with the exception that the electrodes from the design-window are still superposed onto the camera's view. Clicking the Adjustment-Reverse button a second time switches on the camera's view rotation again.

Remark:

All higher-level regulatory feedback elements require the synchronization of the camera-window with the design window. They cannot be activated if synchronization is missing!

2.5 Design window

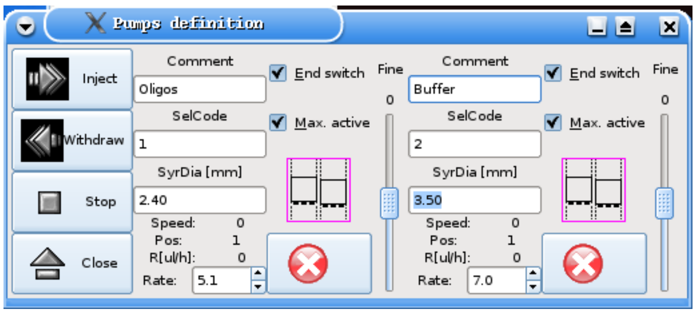
The design-window shows the micro-fluidics structure from the computer-aided-design point of view. It is derived from the original data which has been used in the construction of the micro-fluidics device. As with the camera-window is it composed of several sections, a menu-bar with default pull-down-menus, a menu-bar with icons often used and the micro-fluidics-device's design. In contrast to the camera-window the user can zoom into the design-window arbitrarily often, see section 2.3.2.



In the following the buttons (icons) apart from the standard 'Colors' and 'Fonts'-buttons, see section 2.3.1, are described. Navigating in this window is described in section 2.3.3.

2.5.1 Pumps-control and definition

There are several possibilities to connect pumps to the computer and to access them via serial interfaces. The number of pumps addressable is arbitrary and the user defines in the Perl-TK-script how many pumps to be shown when pressing the pumps-button. The pump-devices are specified in the *ng_biopro_c.cmd* file, which will be described in section 3.7.



Comment: Oligos
[✓] End switch
Fine: 0
SelCode: 1
[✓] Max. active
SyrDia [mm]: 2.40
Speed: 0
Pos: 1
R[ul/h]: 0
Rate: 5.1

Comment: Buffer
[✓] End switch
Fine: 0
SelCode: 2
[✓] Max. active
SyrDia [mm]: 3.50
Speed: 0
Pos: 1
R[ul/h]: 0
Rate: 7.0

The number entered in the field 'SelCode' defines the absolute identifier of the pump. This identifier is either specified in a ROM-area of the pump or a specified hardware address given in *ng_biopro_c.cmd*. Furthermore, the inner diameter of the syringe used and the wanted injection rate can be chosen. For fine-tuning a slider is available which has a 10 times smaller effect than the min-tic at the rate entry. Currently, only the 'Inject'-button is active. 'Inject' and 'Stop' affect all pumps at the same time. Each pump can be deactivated locally when the icon is visible. With the icon the according pump can be restarted again.

About every second the current state of the pumps is updated, the absolute number of revolutions per second is shown as well as the absolute position and the measured rate. A pictogram indicates the left slider going up and the right slider going down (these pictogram are only meaningful with MMT-pumps) and the pictogram resembles the opposite case whereby denotes the pumps being stopped. There is a software-based end-position detection which can be deactivated by dechecking the 'End switch'-button. This end-position can only be detected if the software is running.

A further entry-field is a comment-area which helps the experimenter to remember which chemicals are driven by which pump. The button 'Close' closes the window. The pumps themselves are not affected by this closing.

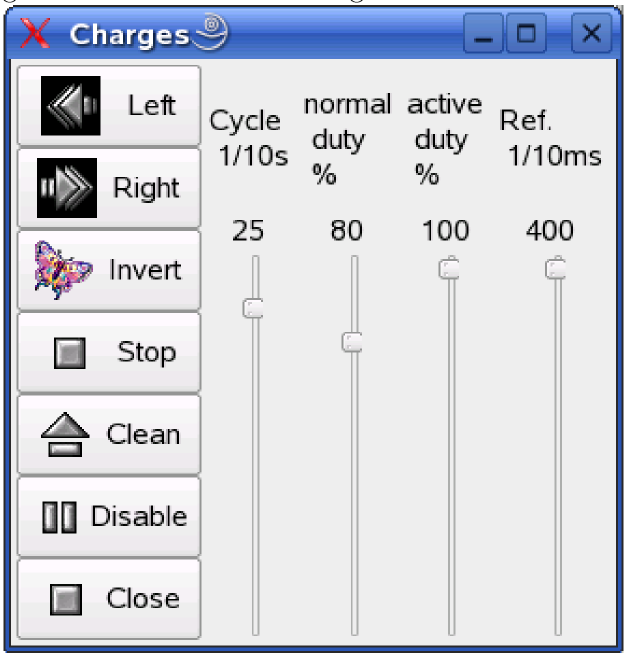
Remark:

Exiting the program does not change the pumps status, if they are active, they stay active. Of course a change of the pumps rate is then not possible apart from manually regulating the pumps. After restarting the program a short pressing of the 'Inject'-button reestablishes the contact of the software with the pumps. This was implemented as a security-feature because strange flow-rates only become active after the 'Inject'-button be pressed.

A further security measure has been implemented to avoid incidentally high flow-rates. With the default setting of the check-box 'Max. active' the flow-rate is restricted to $100\mu l$ per hour. Please be alert, especially when starting the program anew and adjusting the pump-parameters to verify that the syringe diameter is the correct one. ***If you are changing the syringe from 1.03mm diameter to 4.16mm without adjusting the value in the parameters-window huge flow-rates can be generated which jeopardize the attached micro-fluidic structure!***

2.5.2 Shift-register and basic electrode control

The main purpose of the ng_biopro-software is to provide control of in principle arbitrary many electrodes. It is implicitly supposed that these electrodes are of digital nature and that individual voltage-levels are fixed. Generally, all electrodes are able to express three different states: 0V, Vss (with XCS20 3.3V) and tristate (high-impedance regime). These three states are in a certain ideal state. The potential of 0V is quite accurate, typical voltage-levels are below 50mV. Vss with 3.3V usually exerts voltages between 3.0V and 3.2V, depending on the concrete supply-voltage of the FPGA. With Spartan 4 circuits up to five different voltage-levels can be supplied varying between 1.2V and 3.3V. These voltage-levels are generated from outside the FPGAs. The most peculiar state is the high-impedance state (tristate). Ideally, there should be no potential and the impedance between one electrode and any other electrode of the chip should be infinite. In classical digital electronics this is not a real problem, because impedances from more than 1M-Ohm can be interpreted as infinite. In microfluidic scenarios typical fluidic-resistances are in the several M-Ohm regime and an impedance of about one M-ohm is already interpreted as fully connected. Furthermore, due to requirements of CMOS-logic potentials electrodes should not float in the 0.8V to 2.0V range, because arbitrary large currents could occur. This means that the manufacturer artificially clamps the voltage in the high-impedance regime to about 2.0V if the electrode had a high-potential before or 0V if the electrode had been in the 0V-state and worse, the impedance is set to about 300kOhm. These values are not normed and might change from one output-driver-technology to the next. Electrode control in the micro-fluidics area has to suffer from a mismatch of technology and physics. Nevertheless, a clever usage of the electrodes timing behavior can mask these sorts of problems.

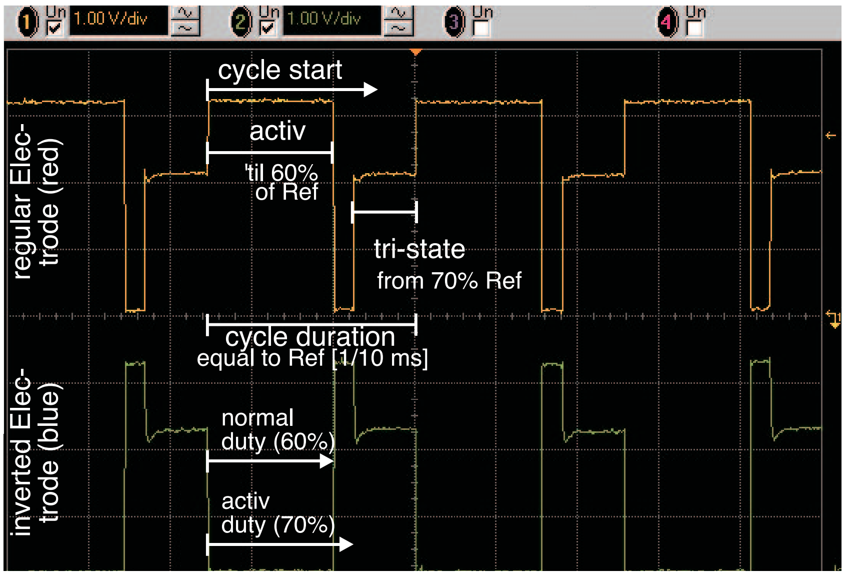


Cycle 1/10s: 25
normal duty %: 80
active duty %: 100
Ref. 1/10ms: 400

The electrodes-control described in this section is at the lowest possible level. Electrodes which have been selected beforehand, see sections 2.3.3 and 2.3.5, can be switched on or off. These electrodes can be subjected to a periodic switching between states. The switching frequency of the period is defined with the slider 'Ref' in tenths of milliseconds. This means, e.g. for a value of 400 every 40 milliseconds the period starts anew. A period has two characteristics: the normal-duty cycle and the active-duty cycle.

The normal-duty cycle defines with what percentage per period the electrodes are subjected to the values they have been specified. The rest of the period the electrodes do have the opposite polarity (the idea behind this, is to stop charged molecules from attaching to surfaces).

This normal-duty cycle is controlled by the active-duty cycle which in addition defines the percentage of the electrodes being active and not in the high-impedance state. The following screen-shot of a measurement illustrates the real electrode behavior.

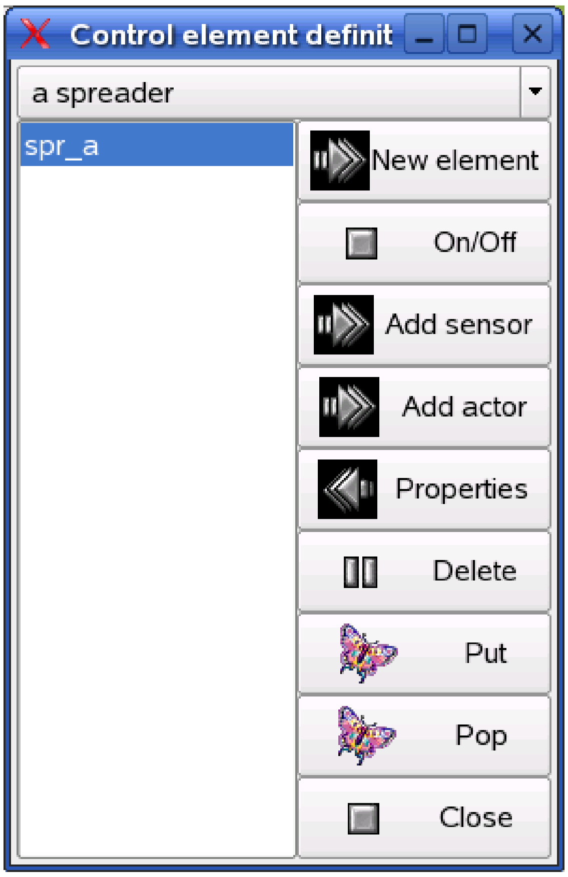


The normal-duty cycle is set to 60% meaning that the electrodes will have the potential they have been specified with in 60% of the reference-cycle period of about 15ms. After 60% of the reference-period are passed the polarity of the electrodes is inverted. The active-duty cycle in addition is set to 70% meaning that 70% of the reference-period the electrodes are active and in the rest of the 30% the electrodes are switched to tristate. See the explanations in the screen-shot shown above.

2.5.3 Level-one elements control

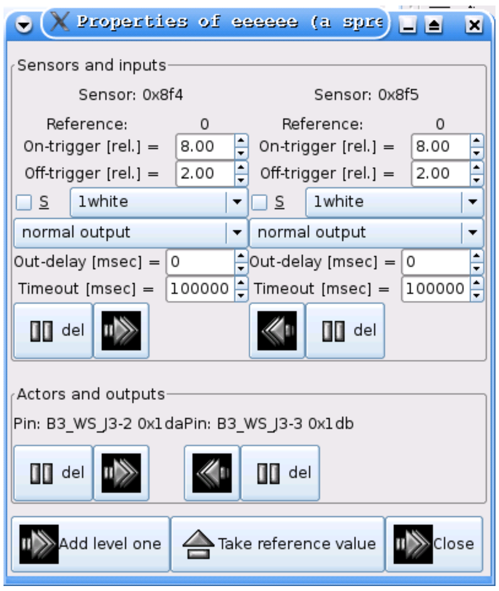
Level-one elements represent the next higher level in the hierarchy of possible control-structures.

Several different types can be specified, of which currently six are possible: the spreader, the catcher, the neuron, the trap, the trigger and the barrier. The general idea of these level-one-elements is to provide one layer of control. These level-one-elements are constructed such that they can be combined to networks of regulation. The connection between these level-one-elements is realized purely in software. Physical delays of remote elements can be dealt with. Additionally, several timeout-features are available to allow for the buildup of robust regulators.



All known level-one-elements in this design are shown in a simple list. A graphical representation of the connections is not yet available. A new element can be created, to the element sensors and actors can be added as long as not all requirements are fulfilled. Note: Before you can define a new level-one-element you first have to select (means highlighting) a sensor which should be part of the level-one-element. The buttons 'Put' and 'Pop' are currently not implemented.

When pressing the 'Properties'-button a popup-window of the following type appears.



Properties of eeeeee (a spre

Sensors and inputs

Sensor: 0x8f4 | Sensor: 0x8f5

Reference: 0 | Reference: 0

On-trigger [rel.] = 8.00 | On-trigger [rel.] = 8.00

Off-trigger [rel.] = 2.00 | Off-trigger [rel.] = 2.00

S 1white | S 1white

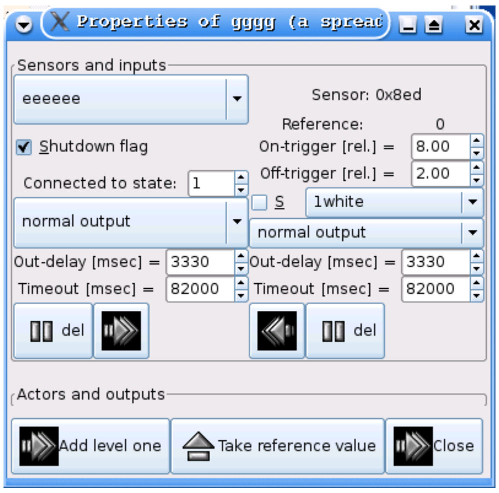
normal output | normal output

Out-delay [msec] = 0 | Out-delay [msec] = 0

Timeout [msec] = 100000 | Timeout [msec] = 100000

Actors and outputs

Pin: B3_WS_J3-2 0x1daPin: B3_WS_J3-3 0x1db



Properties of gggg (a spread

Sensors and inputs

eeeeee | Sensor: 0x8ed

Reference: 0

Shutdown flag | On-trigger [rel.] = 8.00

Connected to state: 1 | Off-trigger [rel.] = 2.00

S 1white

normal output | normal output

Out-delay [msec] = 3330 | Out-delay [msec] = 3330

Timeout [msec] = 82000 | Timeout [msec] = 82000

Actors and outputs

The upper section of the 'Properties'-window (left image) shows the sensors chosen for this particular level-one-element: a spreader. As is shown in more detail in section 4.2.6, two sensors and three actors have to be specified. In the picture above only two actors are specified yet at the left image and no actors in the right image. For each sensor an intensity value can be specified at which the sensor tells the level-one-element that the existence of the ON-condition is fullfilled and a second intensity-value can be specified below which the sensor activates the OFF-condition. The sensors' intensity values are dimensionless and relative to a reference value which can be adjusted by pressing the 'Take reference value'-button. This reference-value is simply the average intensity of the current camera-view.

Because the sensors might have a considerable distance to the site where the actual decision has to be made, a delay in milli-seconds can be specified after which the recognized ON- or OFF-event is evaluated in the software. If the sensor, when it is just actively looked at (this depends on the type of the level-one-element), does not see an event in a specified time, a timeout-event is generated and the level-one-element's state-machine is reset, see section 4 for further details on how level-one-elements are implemented.

Each sensore can be specified in the context of a certain filter. If the 'S'-checkbox is active then before a new measurement is done the according filter is moved into the microscope-beam to assure that the measured values are really corresponding to the parameters set. With the 'S'-box not checked the filter-definition only has a function as commenting the context how the intensity-threshold have to be interpreted.

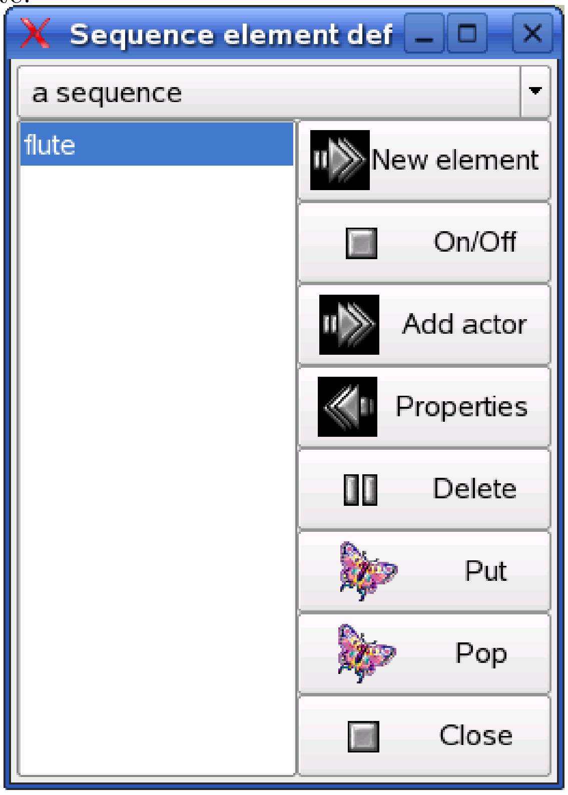
Instead of using a sensor as input another already existing level-one-element can be chosen. See an example in the right image of the figure above where the spreader 'eeeeee' serves as input for the spreader 'qqqq'. The output of this level-one-element is then taken as input for this level-one-element and provides the basis for building networks of low-level-regulators.

2.5.4 Sequences control

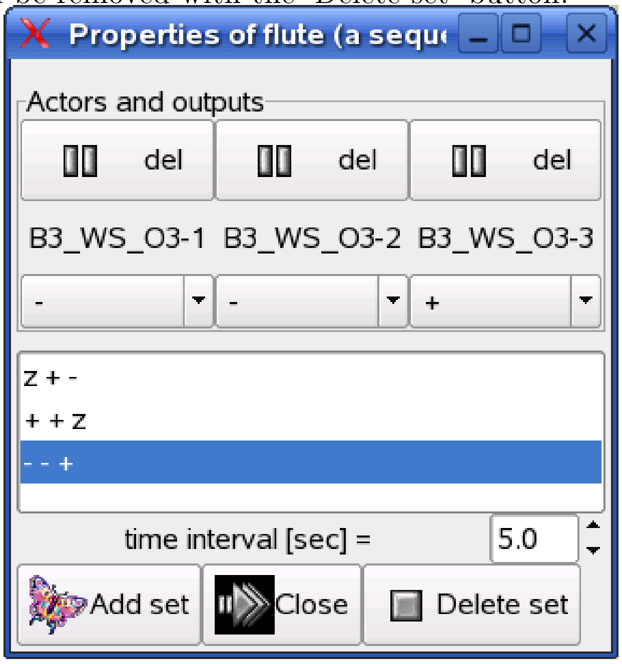
Generating complex time-series of electrode-potentials can be realized with sequence elements. To create a new sequence-element at least one actor has to be selected (highlighted) in the design-window. Pressing the button 'New element' popups a window which asks for the name of the new sequence-element.

Selected sequence elements can be switched on or off using the 'On/Off'-button. Each sequence element can contain arbitrary many different actors. These additional actors have to be selected and then via the button 'Add actor' added.

Without any further specification in the according property-window of the selected sequence-element each actor always is in the high-impedance state.



After pressing the button 'Properties' the following window pop's up in which the user is able to define the polarity at each discrete time step. Take the pull-down menu below the actual electrode-names to define the polarity at the selected time-step. The value 'z' means high-impedance. Time is running from top to bottom and the time-period is defined in the scale at the bottom-right side, e.g. 5 seconds. This means every 5 seconds the next step of the time-sequence is activated. After the bottom step (set) has been executed the next time-step starts from the top anew. Adding a new time-step (set) is done with the 'Add set'-button. All electrode-potentials are preset with the default value of high-impedance. Each selected time-step can be removed with the 'Delete set'-button.



Drop-downs: - | - | +

z + -
+ + z
- - +

time interval [sec] = 5.0

Chapter 3

Interfaces

The purpose of this chapter is to declare the many different interfaces needed in the ng_biopro-software. It is meant as a help for developers of drivers to attach new hardware to the ng_biopro-software. The ng_biopro-software is written in C using an object-oriented paradigm. Thus the programming style is a mixture between C and C++.

3.1 Hardware Interfaces

Hardware interfaces are realized with a procedure oriented set of functions. These function headers will be declared in detail in the current section. The naming convention is a two to three letter prefix pointing to the type of hardware (e.g. for camera-drivers the prefix is 'cam'), followed by a two letter prefix for the specific hardware (e.g. for the Andor-camera this is 'an') and then by the procedure-name giving hints on the real functionality. All prefixes and the procedure-names are connected via underscores '_'. Each type of hardware has a special data-structure containing all specific details. Local data, only relevant for the specific driver, has to be defined as static in the drivers source-file, as long as this static-declaration does not interfere with multiple driver-instantiations. In these type-specific data-structures each function does have an entry-point which is used by the ng_biopro-software to operate the hardware. An initialization function has to be provided by the hardware-interface which initializes this data-structure with the appropriate procedure calls.

All procedures are to be of type integer with a '-1'-return indicating an error-condition. This error-condition usually lets the program stop with a series of error-messages indicating the cause of this error. Return-codes less then '-1' indicate warnings which are not further evaluated but can be used for self-test-software.

In addition, when obeying the rules, user-interactions can be modified. Simple buttons, scales, popup-windows and so on are available, see section 3.2 for further definitions.

3.1.1 Camera

The camera interface certainly is the most important and far-reaching interface, because of the performance requirements involved.

3.1.1.1 Firewire-camera, e.g. Vosskuehler

3.1.1.2 Andor CCD

3.1.1.3 Cascade II (Princeton Imaging)

3.1.2 BioModule and electrodes

3.1.3 Pumps

3.1.4 Filter-wheel

3.1.5 Temperature control

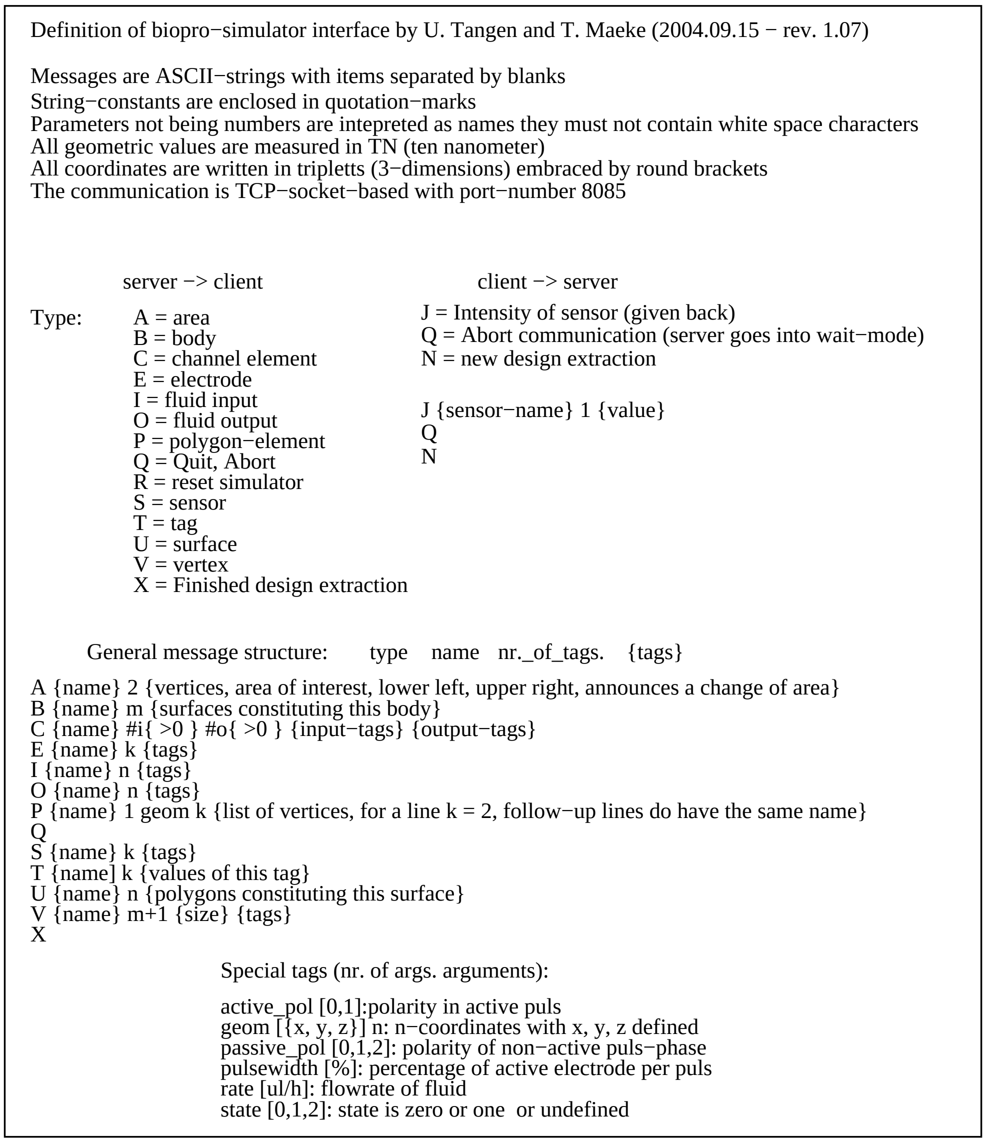
3.1.6 AOTF- and light-sources

3.1.7 The parallel-port

3.2 Graphical interface

3.3 DPD-Interface (dissipative particle dynamics)

The purpose of the DPD-interface is to allow an intimate connection between the experimental Omega-machine and a simulation-tool, in this a case Dissipative Particle Dynamics which is a meso-scale simulation system. Of course, any other molecular dynamics code or partial differential equation-system would fit as well. The role of the ng_biopro-software is twofold: firstly to give the simulation-code the ability to operate the Omega-machine, especially creating a loop between observation (camera-output) and actuation (electrode-control). This would allow an informed feedback-loop with the simulation providing the none-observable details and knowledge of the chemistry going on in the micro-fluidic system. The second purpose of this interface is to provide the simulation with the exact geometric and experimental details of the experiments. This is especially useful if the simulation tries to incorporate phase-boundaries and inhomogenous geometries.

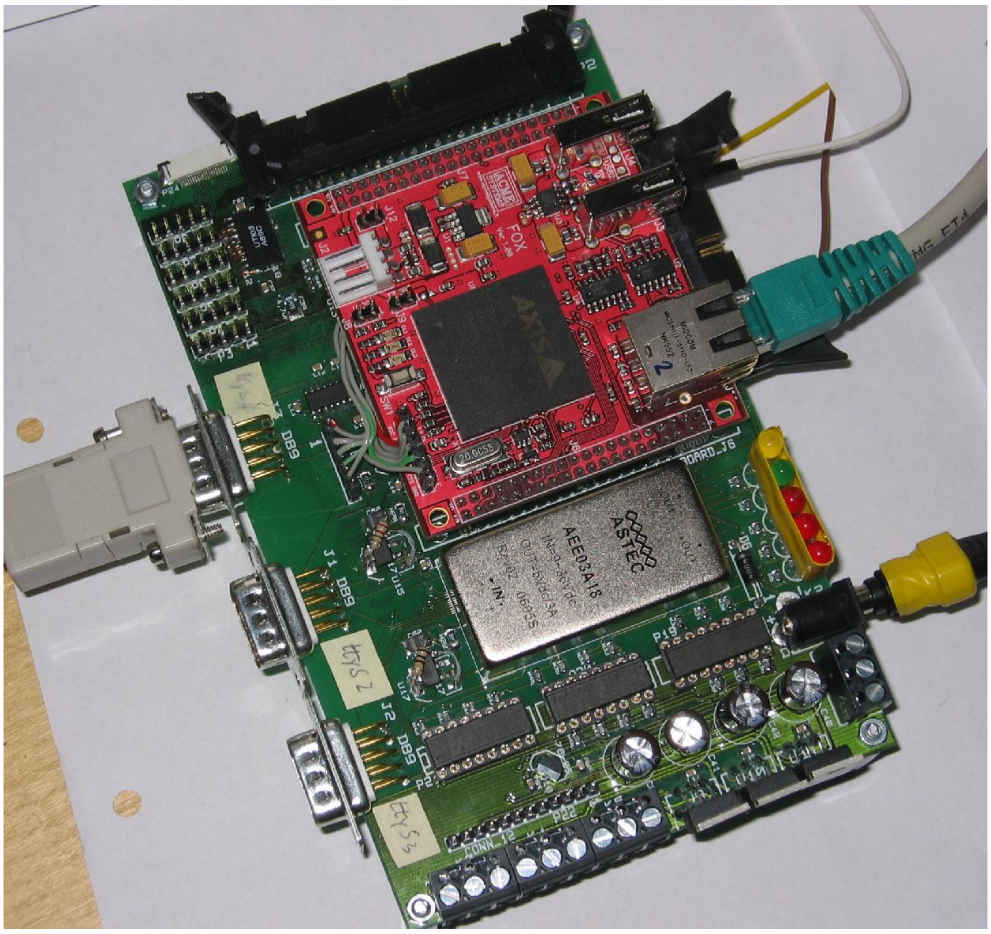


```
Definition of biopro−simulator interface by U. Tangen and T. Maeke (2004.09.15 − rev. 1.07)

Messages are ASCII−strings with items separated by blanks
String−constants are enclosed in quotation−marks
Parameters not being numbers are intepreted as names they must not contain white space characters
All geometric values are measured in TN (ten nanometer)
All coordinates are written in tripletts (3−dimensions) embraced by round brackets
The communication is TCP−socket−based with port−number 8085


            server −> client                      client −> server

Type:       A = area                          J = Intensity of sensor (given back)
            B = body                          Q = Abort communication (server goes into wait−mode)
            C = channel element               N = new design extraction
            E = electrode
            I = fluid input                   J {sensor−name} 1 {value}
            O = fluid output                  Q
            P = polygon−element               N
            Q = Quit, Abort
            R = reset simulator
            S = sensor
            T = tag
            U = surface
            V = vertex
            X = Finished design extraction


        General message structure:     type   name   nr._of_tags.   {tags}

A {name} 2 {vertices, area of interest, lower left, upper right, announces a change of area}
B {name} m {surfaces constituting this body}
C {name} #i{ >0 } #o{ >0 } {input−tags} {output−tags}
E {name} k {tags}
I {name} n {tags}
O {name} n {tags}
P {name} 1 geom k {list of vertices, for a line k = 2, follow−up lines do have the same name}
Q
S {name} k {tags}
T {name] k {values of this tag}
U {name} n {polygons constituting this surface}
V {name} m+1 {size} {tags}
X
                      Special tags (nr. of args. arguments):

                      active_pol [0,1]:polarity in active puls
                      geom [{x, y, z}] n: n−coordinates with x, y, z defined
                      passive_pol [0,1,2]: polarity of non−active puls−phase
                      pulsewidth [%]: percentage of active electrode per puls
                      rate [ul/h]: flowrate of fluid
                      state [0,1,2]: state is zero or one  or undefined
```

3.4 Client-Server Interface

The purpose of the client-server interface is to allow external operators (scripts) to access the hardware and provide a full remote control of the experiments. The interface-declaration is specified in an additional document with name ext_interface.pdf. The communication is realized via a TCP-connection at port 8023.

3.5 Bio@Fox-Interface



The Bio@Fox-board shown is presented without the protecting cover. It consists of an embeeded 100MHz micro-controller (Fox-board from ACME Systems Srl) with Linux as operating-system and several hardware-interfaces. See the hardware-manual of the Bio@Fox-board for further information.

The communication is typically done via Ethernet (RJ45-connector) and a TCP-socket connection between the ng_biopro-software executed on the Bio@Fox-board and the ng_biopro-software executed at the controlling PC. Low-level communication with the Bio@Fox-board is either done via a telnet- or ssh-connection or directly using the console-output at the serial-interface ttyS0 (at the left side in the figure).

3.6 Runtime-parameters

Runtime-parameters are specified using the standard Unix-convention. A minus-sign plus the type of the parameter, then a blank and the parameter value following. Strings containing blanks have to be enclosed in double-quote-signs. The available parameters are:

- -cam {an integer value} [0]
  This parameter defines the camera-interface to be used for image acquisition. Depending on the version of the ng_biopro-software used different types of cameras can be operated with. Use value 0x10 when an Andor-camera is attached and 0x20 if a Vosskuehler-firewire camera is used.

- -design {name of the BioModule-design to be used}

3.7 Scripting Interface

3.7.1 ng_biopro_c.cmd

3.8 VBase Interface

3.8.1 .Vconf-file

3.8.2 .vbase - ASCII base

3.8.3 Message passing to VBase

3.9 Design-environment in detail

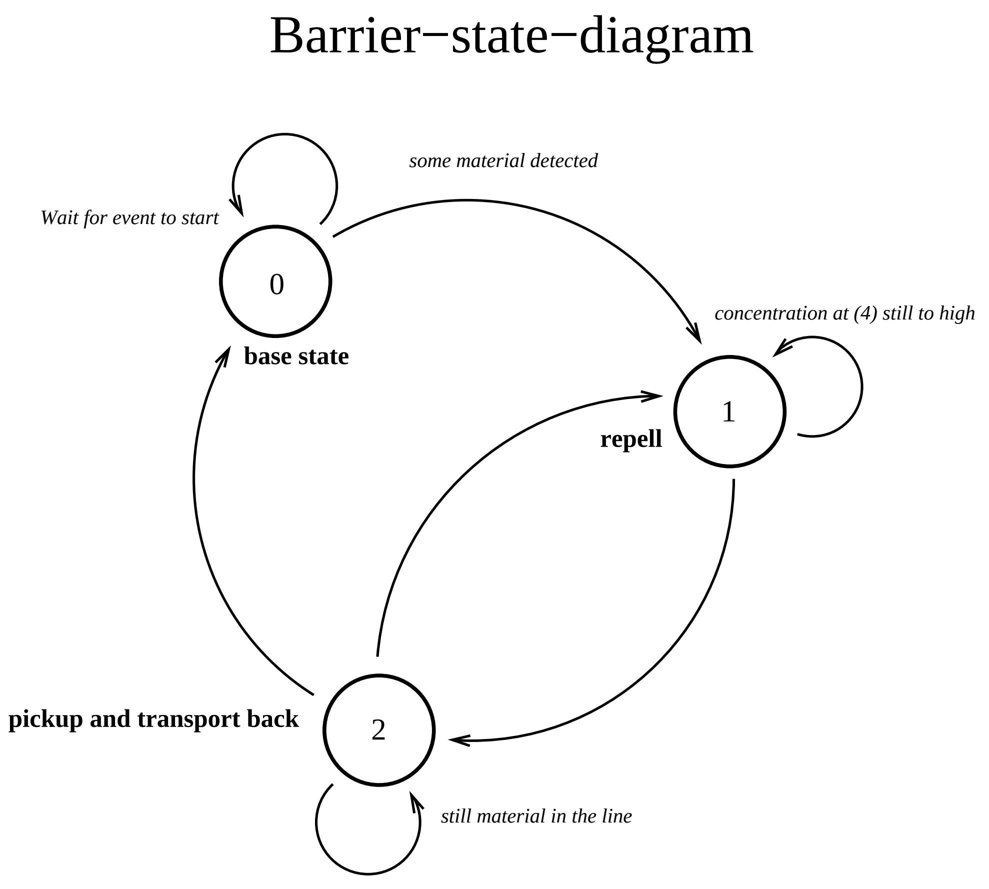
Chapter 4

Level-One

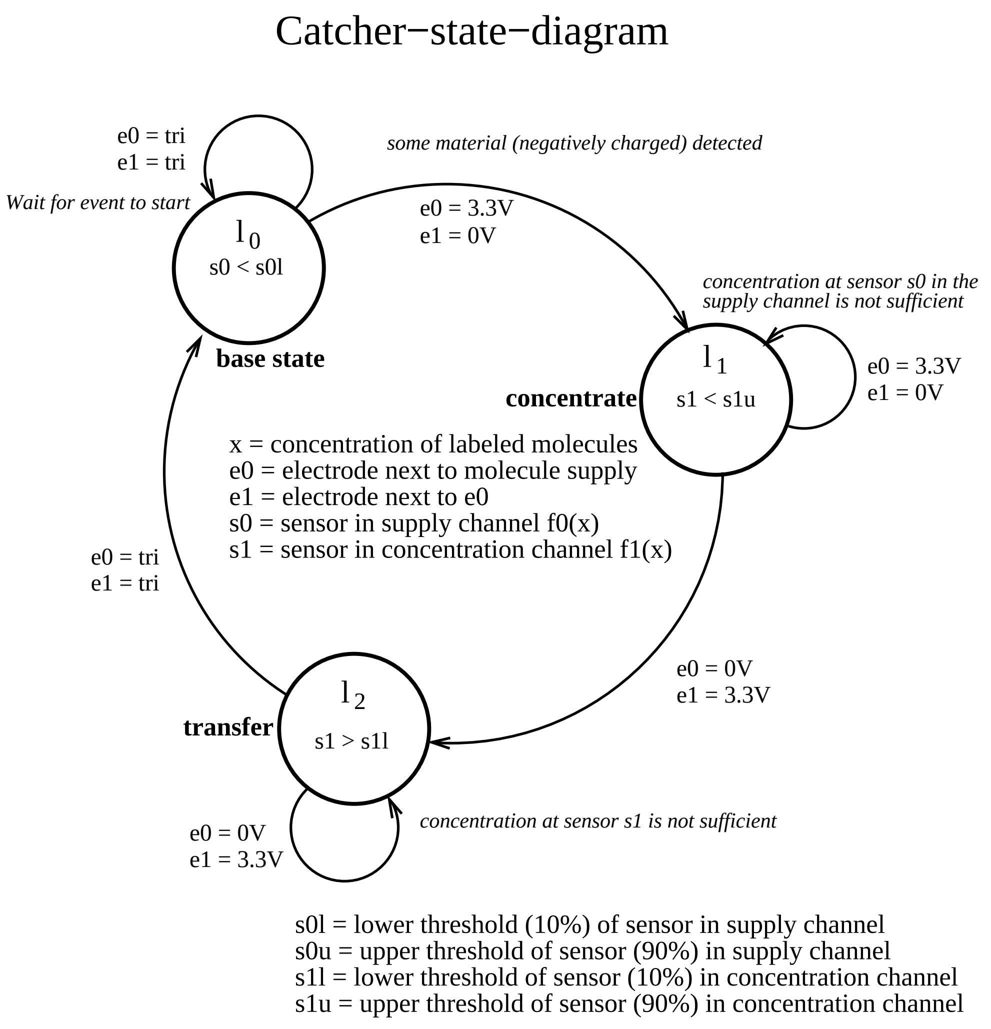
4.1 Introduction of level-one-elements

4.2 Basic elements

4.2.1 BARRIER

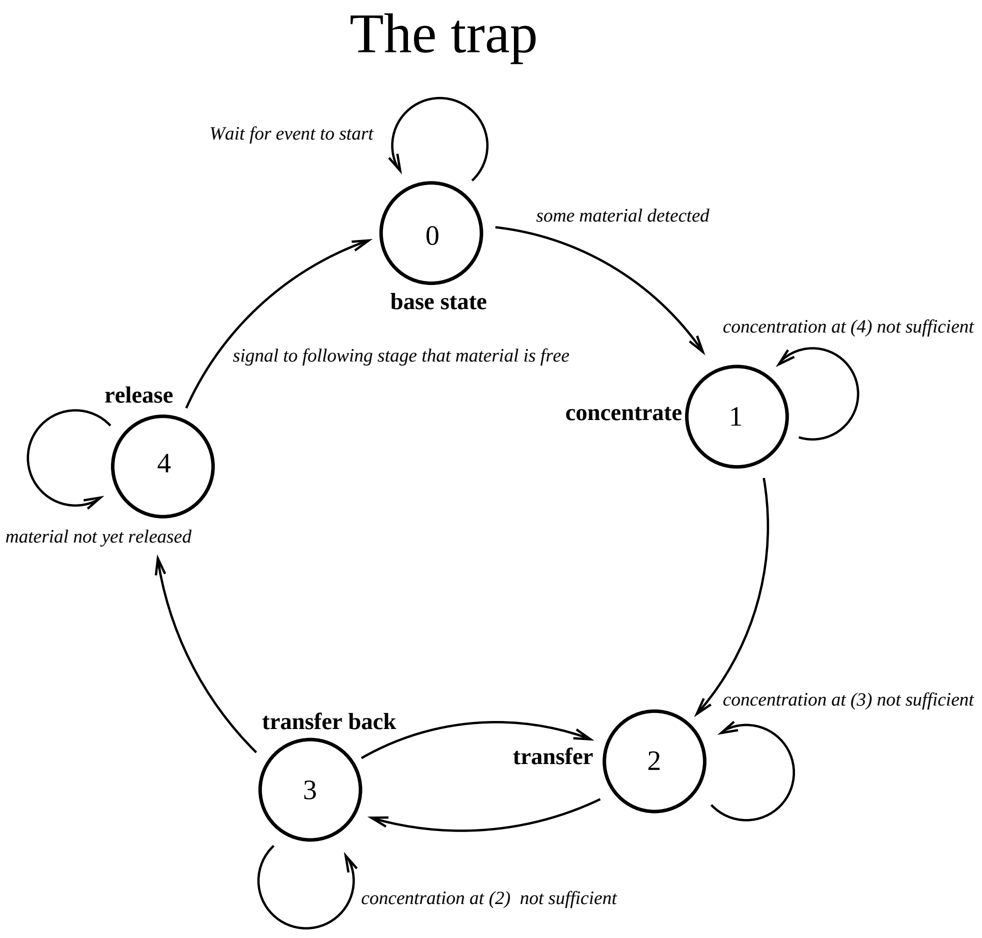

Barrier−state−diagram
Wait for event to start
some material detected
0
base state
concentration at (4) still to high
1
repell
pickup and transport back
2
still material in the line

4.2.2 CATCHER

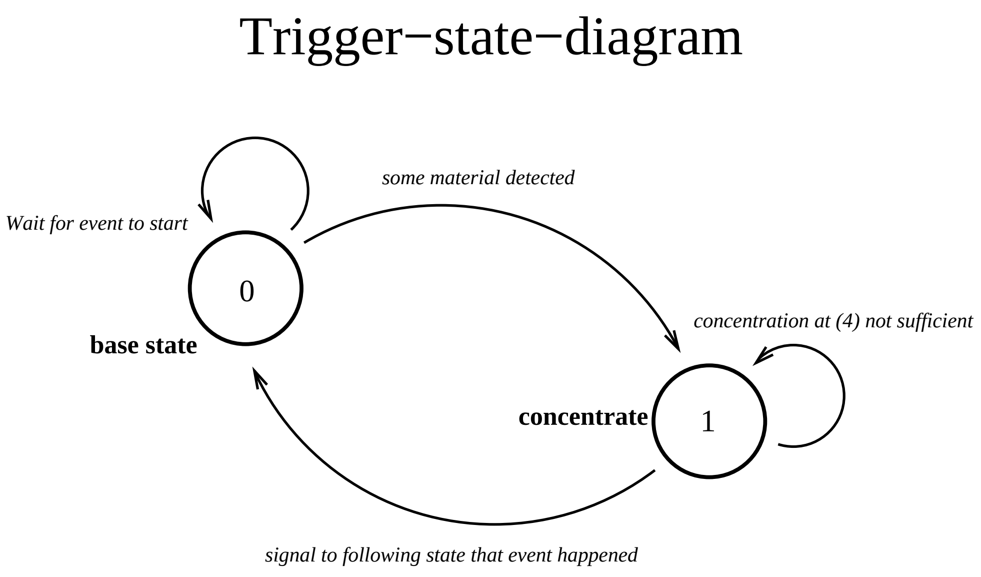

Catcher−state−diagram
e0 = tri
e1 = tri
Wait for event to start
some material (negatively charged) detected
e0 = 3.3V
e1 = 0V
l0
s0 < s0l
base state
concentration at sensor s0 in the supply channel is not sufficient
l1
s1 < s1u
concentrate
e0 = 3.3V
e1 = 0V
x = concentration of labeled molecules
e0 = electrode next to molecule supply
e1 = electrode next to e0
s0 = sensor in supply channel f0(x)
s1 = sensor in concentration channel f1(x)
e0 = tri
e1 = tri
e0 = 0V
e1 = 3.3V
l2
s1 > s1l
transfer
e0 = 0V
e1 = 3.3V
concentration at sensor s1 is not sufficient
s0l = lower threshold (10%) of sensor in supply channel
s0u = upper threshold of sensor (90%) in supply channel
s1l = lower threshold of sensor (10%) in concentration channel
s1u = upper threshold of sensor (90%) in concentration channel

4.2.3 Neuron

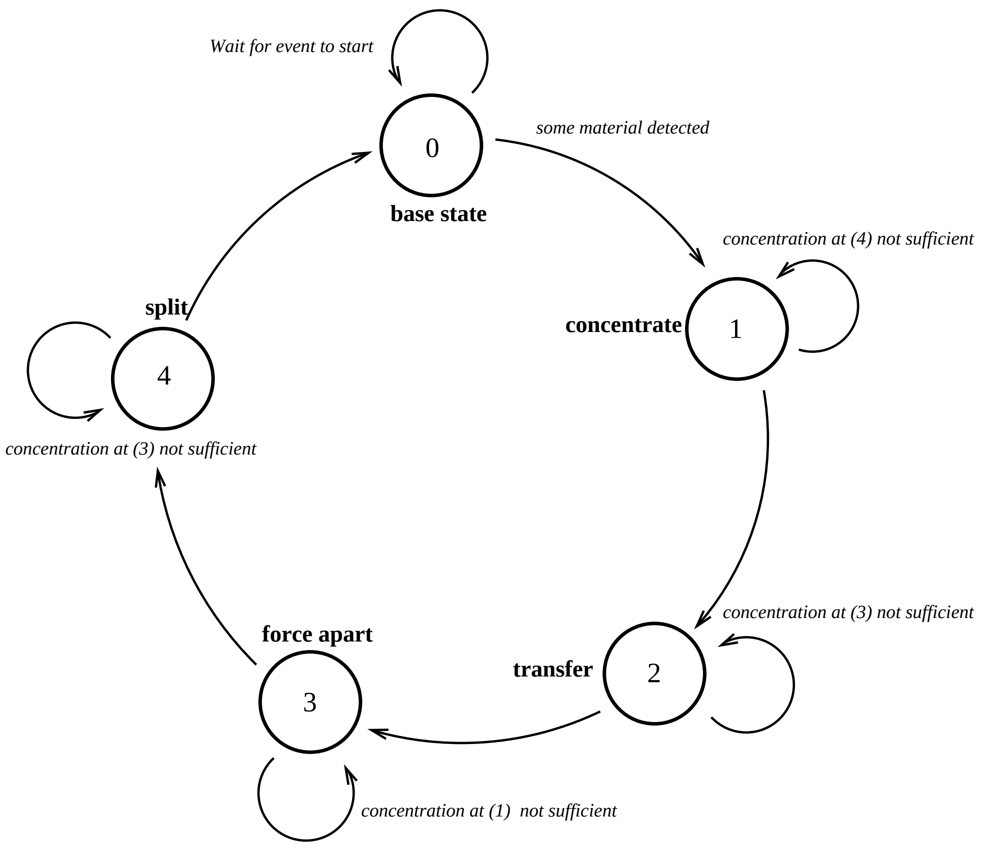
4.2.4 TRAP


The trap
Wait for event to start
0
base state
some material detected
concentration at (4) not sufficient
concentrate
1
signal to following stage that material is free
release
4
material not yet released
concentration at (3) not sufficient
transfer back
transfer
2
3
concentration at (2) not sufficient

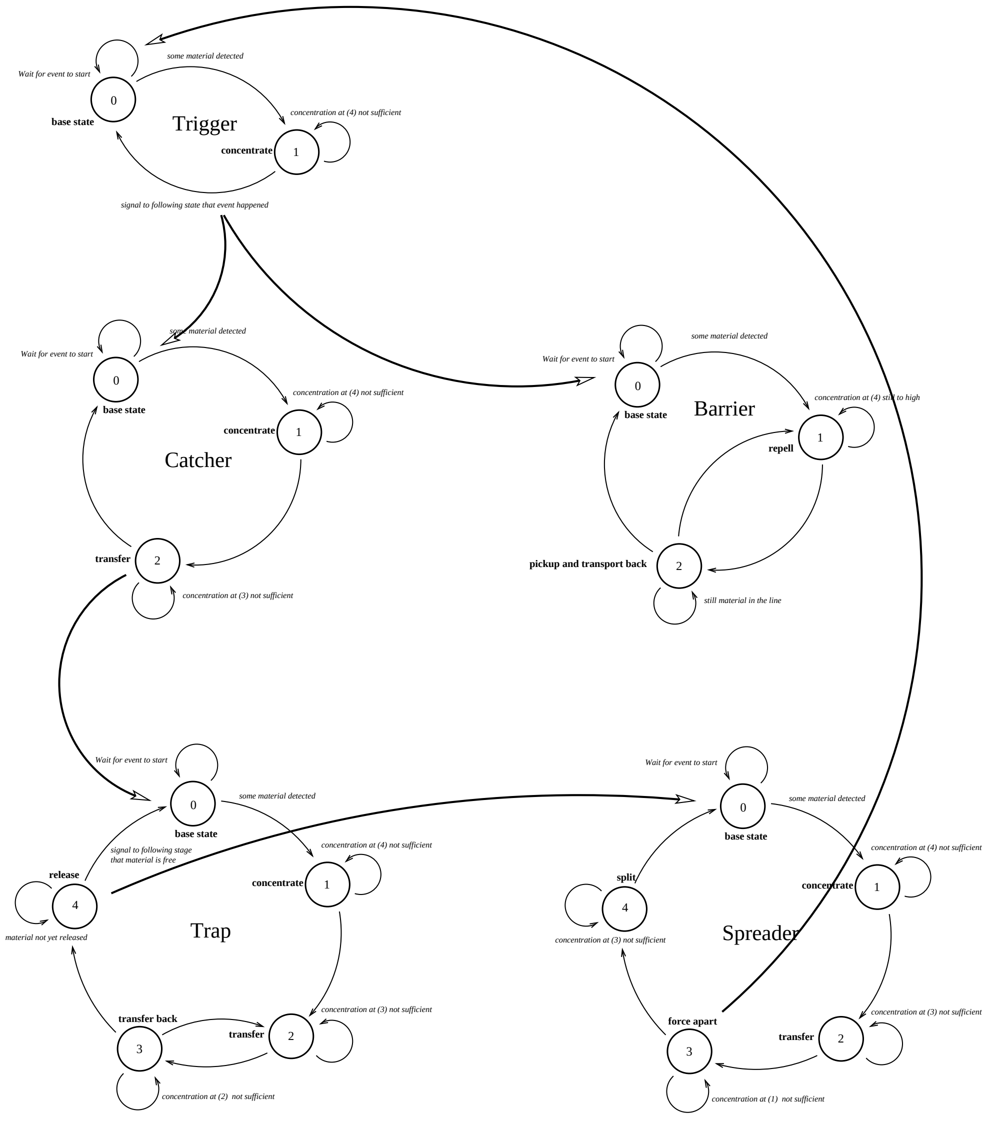
4.2.5 TRIGGER



4.2.6 SPREADER


Wait for event to start
0
some material detected
base state
concentration at (4) not sufficient
split
concentrate
1
4
concentration at (3) not sufficient
concentration at (3) not sufficient
force apart
transfer
2
3
concentration at (1) not sufficient

4.3 Networks


Trigger
Wait for event to start
0
base state
some material detected
concentration at (4) not sufficient
concentrate
1
signal to following state that event happened
Catcher
Wait for event to start
0
base state
some material detected
concentration at (4) not sufficient
concentrate
1
transfer
2
concentration at (3) not sufficient
Barrier
Wait for event to start
0
base state
some material detected
concentration at (4) still to high
repell
1
pickup and transport back
2
still material in the line
Trap
Wait for event to start
0
base state
some material detected
concentration at (4) not sufficient
concentrate
1
signal to following stage
that material is free
release
4
material not yet released
transfer back
3
transfer
2
concentration at (3) not sufficient
concentration at (2) not sufficient
Spreader
Wait for event to start
0
base state
some material detected
concentration at (4) not sufficient
concentrate
1
split
4
concentration at (3) not sufficient
force apart
3
transfer
2
concentration at (3) not sufficient
concentration at (1) not sufficient

Chapter 5

Programmers-Manual

5.1 ng_biopro-Library

5.2 Client-Server message passing

5.3 Drivers interface

5.3.1 Parallel-port

```
Pin-assignment of PC-parallel-port to adapter
```

Signal Name	Register Bit	DB-25 Pin	I/O Direction	Board-BioPRO	G88-40
-Strobe	¬C0	1	Output	GCK5	30
+Data Bit 0	D0	2	Output	TMS	25
+Data Bit 1	D1	3	Output	TDI	26
+Data Bit 2	D2	4	Output	TCK	27
+Data Bit 3	D3	5	Output	-|>|—– Vcc-+	
+Data Bit 4	D4	6	Output	-|>|—– Vcc-+	
+Data Bit 5	D5	7	Output	-|>|—– Vcc-+	
+Data Bit 6	D6	8	Output	-|>|—– Vcc-+	
+Data Bit 7	D7	9	Output	-|>|—– Vcc-+–	37,39
-Acknowledge	S6	10	Input		
+Busy	¬S7	11	Input	TDO	28
+Paper End	S5	12	Input		
+Select In	S4	13	Input		
-Auto Feed	¬C1	14	Output	/Prog	22
-Error	S3	15	Input	/Init	21
-Initialize	C2	16	Output	CClk	23
-Select	¬C3	17	Output	DIN	24
Ground	-	18-25	-	GND	29,40

```
VCC at adapter GP1 = ST53 or GP3 = ST55 to ST62 connected
S7, C0, C1 & C3 signals are inverted
Initial state of port is Dat=0xff Sts=0x7f Ctl=0xf Flags = 0x0 Phase =
0x1
```

```
typedef struct s_ng_parport ng_parport; /* Data of parallel-port-class */
typedef int (*pa_port_info)(void);
typedef int (*pa_set_all)(int);
typedef int (*pa_set_din)(int);
typedef int (*pa_set_prog)(int);
typedef int (*pa_set_cclk)(int);
typedef int (*pa_set_gck5)(int);
typedef int (*pa_set_tms)(int);
typedef int (*pa_set_tdi)(int);
typedef int (*pa_set_tck)(int);
typedef int (*pa_set_vcc)(int);
typedef int (*pa_get_init)(void);
typedef int (*pa_get_tdo)(void);
typedef int (*pa_get_tms)(void);
struct s_ng_parport {
    NG_OBJECT_HDR /* p_data->p_ana, p_user->? */
    pa_port_info f_port_info; /* Printout some information of parport */
    pa_set_all f_set_all; /* Set all parport pins to value x */
    pa_set_din f_set_din; /* Set DIN of FPGA */
    pa_set_prog f_set_prog; /* Pull the PROG-pin and initialize FPGA */
    pa_set_cclk f_set_cclk; /* Set the CCLK to value x */
    pa_set_gck5 f_set_gck5; /* Set the global clock 5 GCK5 to x */
    pa_set_tms f_set_tms; /* Set TMS (at FPGA) to value x */
    pa_set_tdi f_set_tdi; /* Set TDI (at FPGA) to value x */
    pa_set_tck f_set_tck; /* Set TCK (at FPGA) to value x */
    pa_set_vcc f_set_vcc; /* Activate power-supply via parport */
    pa_get_init f_get_init; /* Get INIT-pin status */
    pa_get_tdo f_get_tdo; /* Get value of TDO at FPGA */
    pa_get_tms f_get_tms; /* Get current TMS value */
    NG_OBJECT_FOOT
};
```

5.3.2 Camera-driver

```
typedef int (*c_get_sens_bounds)(ng_ngen *, int *, int *);
typedef int (*c_set_sens)(ng_ngen *, int);
typedef int (*c_set_exposure)(ng_ngen *, double);
typedef int (*c_get_expo_bounds)(ng_ngen *, double *, double *);
typedef int (*c_rdpicqq8)(ng_ngen *, int, int, int, int, pixel8_t *);
typedef unsigned int (*c_rd_info)(ng_ngen *);
typedef int (*c_init_sum_mems)(ng_ngen *, int);
typedef int (*c_sum_clear)(ng_ngen *, int);
typedef int (*c_set_sumarea)(ng_ngen *, unsigned int, unsigned int, unsigned int, un-
signed int, unsigned int);
typedef int (*c_age_wr_videoram)(ng_ngen *, unsigned int, unsigned int, unsigned int);
typedef unsigned int (*c_get_sumpixels)(ng_ngen *, int);
typedef unsigned int (*c_get_sumram)(ng_ngen *, int);
typedef int (*c_read_sumram)(ng_ngen *, int);
typedef int (*c_wr_ofs_fac)(ng_ngen *, unsigned int, unsigned int);
typedef int (*c_ioctl)(ng_ngen *, int);
typedef int (*c_min_avg_max)(ng_ngen *, int *, int *, int *);
typedef int (*c_image_size_x)(void);
typedef int (*c_image_size_y)(void);
typedef int (*c_on)(ng_ngen *);
typedef int (*c_off)(ng_ngen *);
typedef int (*c_status)(ng_ngen *, ng_grafik *);
typedef int (*c_read_vbase)(Vbase *);
typedef int (*c_write_vbase)(Vbase *);
```

5.3.2.1 Camera data-structure

```
struct s_ng_cam {
    NG_OBJECT_HDR /* p_data–>p_ana, p_user–>? */
    ...
    c_get_sens_bounds f_get_sens_bounds; /* Get sensitivity bounds */
    c_get_expo_bounds f_get_expo_bounds; /* Get bounds of exposures */
    c_set_sens f_set_sens; /* Set sensitivity gain */
    c_set_exposure f_set_exposure; /* Set exposure time */
    c_rdpicqq8 f_rdpicqq8; /* Read an byte-image */
    c_rd_info f_rd_info; /* Get info about camera */
    c_init_sum_mems f_init_sum_mems; /* Initialize summen-memories */
    c_sum_clear f_sum_clear; /* Clear summen-memories */
    c_set_sumarea f_set_sumarea; /* Define a summen-area */
    c_age_wr_videoram f_age_wr_videoram; /* Write something into cambuf */
    c_get_sumpixels f_get_sumpixels; /* Get size of summen-area */
    c_get_sumram f_get_sumram; /* Get integrated value of sum */
    c_read_sumram f_read_sumram; /* Read value of sum and clean */
    c_wr_ofs_fac f_wr_ofs_fac; /* Define offset factor */
    c_ioctl f_ioctl; /* I/O-control */
    c_min_avg_max f_min_avg_max; /* Readout min, avg and max-va */
    c_image_size_x f_image_size_x; /* Return the cameras nr. x-pi */
    c_image_size_y f_image_size_y; /* Return the cammers nr. y-pi */
    c_on f_on; /* Switch the camera on */
    c_off f_off; /* Switch the camera off */
    c_status f_status; /* Present a status-window */
    c_read_vbase f_read_vbase; /* Read presets from VBase */
    c_write_vbase f_write_vbase; /* Store presets in VBase */
    ...
    NG_OBJECT_FOOT
};
```

5.3.2.2 Vosskuehler-camera

5.3.2.3 Andor-camera

5.3.2.4 Cascade II-camera

5.3.3 BioModule-driver

5.3.4 Bio@Fox-driver

5.3.5 Pumps-driver

5.3.6 MereGen-board

Chapter 6

Miscelleanous

6.1 GTK+-2-installation instructions for MAC-OSX and other UNIX-derivates.

After installing XCode-Tools and X11-development-environment the following packages (all are available from public-domain-online-sources, e.g. sourceforge.net) need to be installed: atk-1.9.1.tar.bz2 cairo-1.2.6.tar.gz jpegsrc.v6b.tar.gz expat-2.0.0.tar.gz libXrender-0.9.0.tar.gz libpng-1.2.8.tar.bz2 fontconfig-2.4.2.tar.gz pango-1.12.4.tar.bz2 freetype-2.1.4.tar.gz pkg-config-0.20.tar.gz gettext-0.16.1.tar.gz glib-2.12.12.tar.bz2 render-0.8.tar.gz gtk+-2.10.12.tar.bz2 renderext-0.9.tar.gz tiff-3.7.4.tar.gz. The revisions shown here reflect the situation in May 2007. Later revisions should be unproblematic though if not requirements are given with the configure-scripts of the individual packages.

Unfortunately the sequence of installation is not at all arbitrary: pkg-config, gettext, expat, jpegsrc, tiff, libpng have to be installed first. Then follows render, libXrender, freetype and fontconfig. The last chunk are glib, cairo, atk and pango. The last package to be installed is gtk itself. In almost all packages configure-scripts are provided. Sometimes, special runtime options have to be provided. 'configure' can always be called as : configure –help, resulting in an extensive list of options possible. As usual, after a successful run of 'configure', 'make' and 'make install' have to be issued. Alternatively, fink, adapt, synaptic or other software-installation utilities can be used as well. Please be aware, that the software from the packages normally will be install in the /usr/local/.. tree whereby fink for examples uses a /sw/.. tree. This can result in not found libraries and include-files and has to be resolved individually.

6.2 Revision history

Version 0.1 was released in April 2006

Version 0.2 was release in May 2007 with further descriptions of hardware-interfaces.

6.3 Release Notes

Index

Bibliography