Something about the 'PHD 2000 Programmable' from Harvard Apparatus

Oliver Müller

September 16, 2008

Abstract

This report was done during the **Summer Student Program 2008** at **DESY** under the leadership by the **HASYLAB** member Dr. S. V. Roth.

A command line software was written in PERL, which enables a User to remote control a stepper motor driven syringe pump[1]. The pump in combination with this software will mainly be used to create small droplets and constant flow rates. Due to different syringes and needles dropsizes whitin $60\mu l$ to $5.5\mu l$ had been achieved by using water. Flow rates up to $130ml/min$ are possible.

All settings were tested with water. Thus differences can be observerd if liquids with highly different surface tension and compressibilities are used.

[1] *PHD 2000 Programmable* from *Harvard Apparatus*

Contents

Chapter 1

Motivation

The ordering of solutions, e.g. colloidal or polymeric blends, on solid substrates is of utmost importance for many areas of science and technology [APL07]. Especially flow-induced [Mou08] or droplet-based deposition [APL07] is a very useful, but also very complex process, where many hydrodynamic, thermodynamic or diffusive interactions are involved. In order to better understand the process of inkjet printing, the non-equilibrium kinetics of drying colloidal solutions have previously been investigated using nanobeam grazing incidence small-angle x-ray scattering (nanoGISAXS) [APL07].

In order to prepare such experiments at HASYLAB, especially in view of the μSAXS/WAXS beamline at PETRA III, we prepared a multi-purpose syringe pump. The general measurement geometry shown in Figure will be implemented first in user experiments at BW4. Here, the droplet is deposited on a solid substrate, and GISAXS using a microbeam is used to observe the colloidal ordering at the liquid/solid/air interface.

Chapter 2

The Software

The command line Software was written in PERL, using the *Active PERL 5.8.8* interpreter. Hence the script should run on MS Windows and Linux based machines. This was verified with the *Microsoft Windows XP SP2* and *SUSE Linux 10.1* operating system (OS). The following section 2.1 describes the additionally needed packages for the use of the serial port.

2.1 Perl Packages

There are several packages available[1] which allow the commuincation via the serial port. Each OS needs its own specific package which usually include different instructions to access the serial port. With regard to a multiplatform code the following libraries were chosen.

For MS Windows systems the libraries *Win32-SerialPort-0.19* and *Win32-API-0.55* have to be installed. To use the script with a Linux OS the script requires the library *Device-SerialPort-1.04*. The main advantage of these packages is the identical instruction set and syntax, due to it the source code can be kept efficiently short for both platforms.

2.2 Serial Port

Currently the software will try to establish a connection through the first serial port. If the pump is connected to another port the Code has to be modified accordingly. In this case there are only two lines which need to be corrected (cf. Fig. 2.1). It should be mentioned that the first Serial Port on Windows computers is called *COM1* whereas on the Linux site it

[1] `http://search.cpan.org`

```
...
if($OS_win){
        $port = "COM1"; #has to be modified eventually
        $PortObj = Win32::SerialPort->new($port);
}
else{
        $port = "/dev/ttyS0"; #has to be modified eventually
        $PortObj = Device::SerialPort->new($port);
}
die "Can't open serial port $port: $^E\n" unless($PortObj);
...
```

Figure 2.1: Part of the Perl code. The `port`-variable has to be altered if another Serial Port is used.

is defined by */dev/ttyS0*. So the different numberation should be kept in mind if changing the code becomes necessary.

2.3 Menu Structure

All available functions are divided into different menus. Each menu can be closed by typing **exit**. The starting menu provides two submenues **main**, **set** and the function **debug**.

```
------- Program v0.8 ---------------------------
 main....... Pump & Refill
 set........ Settings
 debug...... Debug Mode
 exit....... Exit Programm
------------------------------------------------
>
```

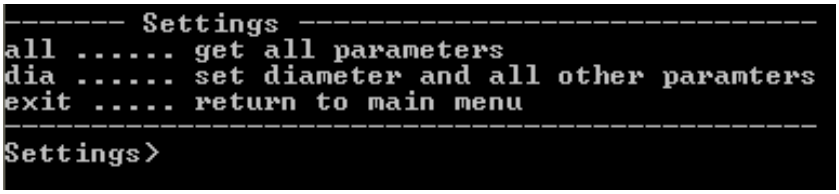
The starting menu. All avaliable functions are accessible through the submenues.

The later is to be understood as a kind of terminal program which offers a direct access to the pump. Here the pump instructions as given in the pumps manual can directly be typed and the response verified. In the following figure this is shown with the instruction **DIA** which requests the current diameter.

The *'debug'* function provides full access to the pump. Here the *'DIA'* instruction is shown.

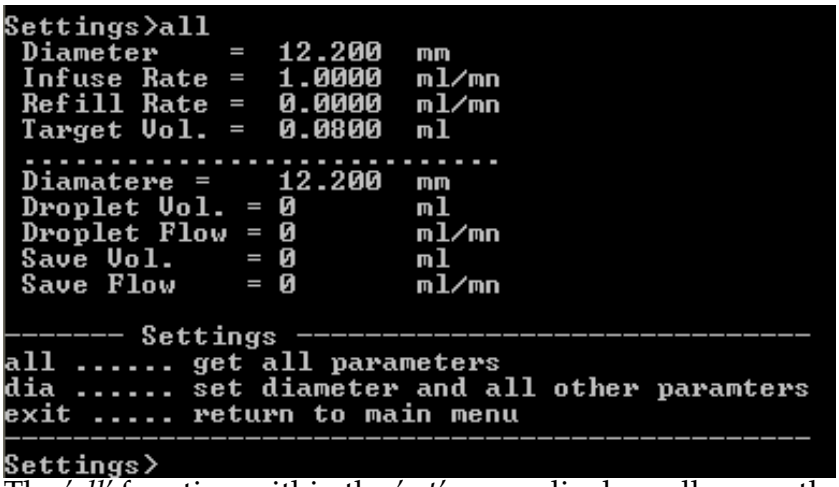
2.3.1 Creating droplets

In order to remote controll the pump via this software it is necessary to run the **'set'** menu first.



The *'set'* menu.

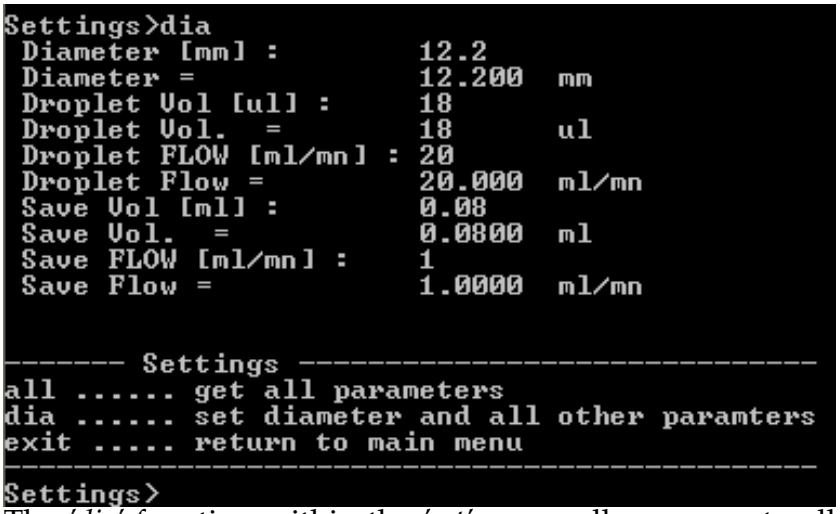
Both functions which are provided by this menu allow to check on the pumps parameters and the softwares global variables. The function **'all'** querys all currently stored parameters and variables. The varaibles are separated by a dashed line.



The *'all'* function within the *'set'* menu displays all currently stored parameters and variables.

If this function is executed after a fresh start of the program all variables, except the diameter value, are undefined indicated by a zero value. Hence

it is essential to define these variables with the function **'dia'**. This function requests all variables including the diameter and transmits them immediately to the pump, in order to verify theire validation. That is why a following **'all'** call can return different pump parameters as they were obtained before. If zero values are entered, except the diameter value, they will not be transmitted to the pump.



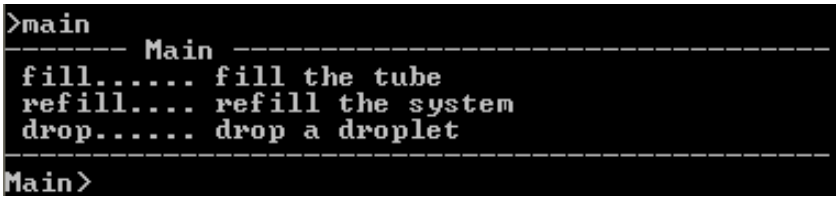
The *'dia'* function within the *'set'* menu allows acces to all important parameters.

If a typed value does not match the pumps limitations an error message appears and the current menu will be closed. Already accepted parameter values will not be rejected but the function still need to be run again untill all parameters have been defined successfully.

The variable *'Droplet Vol [ul]'* expects the Volume of one single droplet in $[\mu l]$. How this value can be determined is described in Chapter 5. The variable *'Save Vol [ml]'* is thought of a buffer which retracts the piston in order to aspirate the liquid unitl it is completely covered inside the tube. Thus avoiding accidental dripping when the pump is currently not in use. Because of mechanical backlash higher volumes are preferred but too high values can lead to an aspiration of air which should be avoided in any case. Experience show that approximately > 10 times the droplets volume achieves adequat results. To ensure high precision at this point it is recommended that the *'Save Flow [ml/mn]'* is set to a low flow rate (e.g. $1ml/min$). If it is necesseary the buffer *'Save Vol [ml]'* can be switched off by setting both variables[2] to zero.

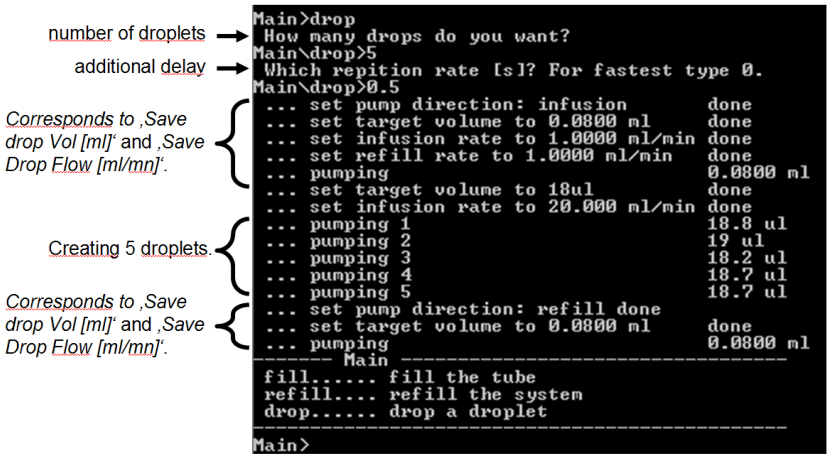
If all variables are defined the **'set'** menu can be closed by typing **'exit'** and the **'main'** menu can finally be entered.

[2] *'Save Vol [ml]'* and *'Save Flow [ml/mn]'*



The *'main'* menu.

The most important function within this menu is the **'drop'** instruction. This function allows the creation of droplets with respect to the defined parameters before. Additionally an arbitrary delay ($0.1s$ resolution) between multiple droplets can be defined.



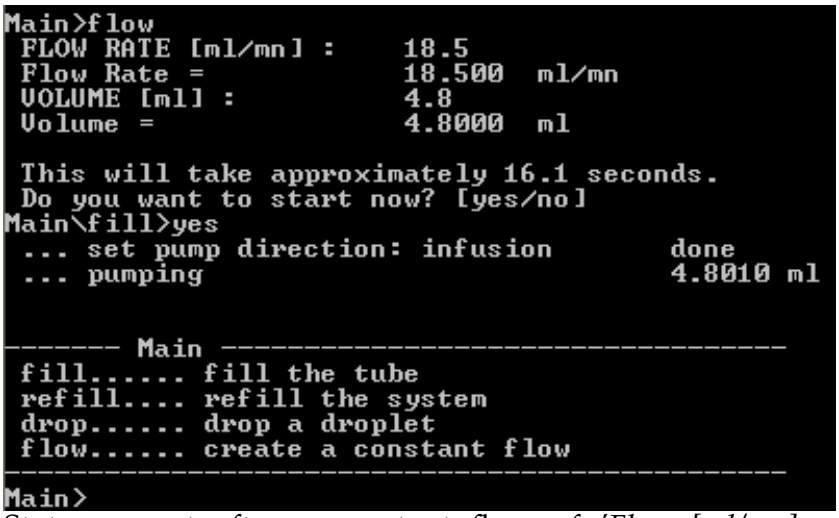
Status report after 5 droplets have been created. The following settings had been used: *'Droplet Vol [ul]* = $18\mu l$*'*, *'Droplet Flow [ml/min] = 20 ml/min'*, *'Save Vol [ml]* = $0.08ml$*'* and *'Save Flow [ml/mn]* = $1ml/min$*'*.

As it can be seen in the picture above the entire communication is simultanously logged at the command line. After each send instruction the software querys the current state of the pump. This also enables the readout of the actually pumped volume.

2.3.2 Creating a constant flow

The **'main'** menu provides a function which can establish constant flow rates as well. This function is executed by typing **'flow'**. It is very important to have the diameter parameter defined correctly, before running this function. This can either be done by using the pumps interface (cf. 3.3),

e.g. when installing the syringe[3], or by the **'set->dia'** function as described above (cf. 2.3.1). In this case the redundant values concerning the droplets can be set to zero.



Status report after a constant flow of *'Flow [ml/mn]* = $18.5ml/min$' had been established.

2.3.3 Miscellaneous

The pump has two parameters in order to store flow rates. These are *'Infuse Rate'* and *'Refill Rate'*. Both are distinguished by the flow direction. Within a staus report (cf. pictures above) these parameters are set accordingly to the predefined values, e.g. *'Droplet Flow [ml/mn]'* or *'Save Flow [ml/mn]'*.

To abort the commandline software at any time the shortcut *'ctrl'* + *'c'* can be used. This will immediatly close the programm whereas the last single instruction will still be executed since the instruction was already transmitted to the pump. Futhermore the sofware variables are lost and have to be set up again.

There is no stall detection implemented in the software. Although some functions query the pump if an interrupt occured, just to inform the user, it should always be assured that there is enough liquid in the syringe.

[3]It is strongly recommended to update the diameter value immediately after a new syinge has been installed. Thus avoiding any failures from the beginning.

Chapter 3

The Syringe Pump

For this report the programmable syringe pump *PHD 2000 Programmable* from *Harvard Apparatus* has been used. The pump offers two slots where syringes can be installed parallel to each other. The syringes are fixed with one clamp from above. For syringes with small respectively large diamters the clamp need to be turned over to ensure stable holding. Even if only one syringe is needed a second one should be installed to increase rigidity and to minimize the backlash. In combination with the *'Save Vol [ml]'* buffer (cf. reflable), this can in fact lead to a major problem.

Futhermore the pump offers a serial RS232 interface. Due to two RJ-12 connectors[1] the serial signal is looped through the pump to enable cascadability. In this way several pumps can be integrated into a so called pump-chain.

3.1 RS-232

In order to establish the physical connection between a computer and the pump a special adapter and cable was build. A RJ-12 and a RJ-14 connector are mounted on the ends of a 6wire telephone cable. The two outer pins of the RJ-45 plug are left blank. The adapters pin assignement can be extracted from figure 3.1.

The software expects the pump to run with certain RS-232 settings. All settings concerning the serial interface are avaialbe by pressing the *RS-232* button. In a remtote controlled environment, like in this case, the pump need to be set up in the *'PUMP CHAIN'* Mode. This can be selected by pressing the *RS-232* button several times until the message appears on the

[1] These conncetors are labeld *IN* and *OUT*

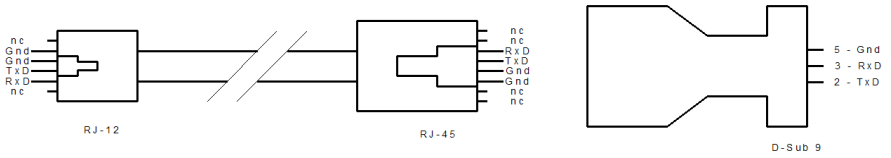


Figure 3.1: Cable and adapter (female DSUB-9) to connect the pump to a computer. The left hand side is connected to the pump, the other end to a computer

LCD. The new setting is confirmed with the *'enter'* button. In the following the pump requests an address, which will identify the pump if it is integrated into a pump chain. Currently the software doesen't support pump chains, so it is necessary to leave the adress[2] unchanged. Pressing the *'enter'* button again will confirm the entered address and skip to the next query where the baud rate can be choosen. The software requiers a baud rate of 9600bps.

If it becomes necessary to run the serial interface at another baud rate the pump offeres different settings as shown below. But therefore the code[3] has to be modified.

Baud Rate: 1200, 2400, 9600, 19200
Word Size: 8
Stop Bits: 2
Parity: none

3.2 Pump Chain Protocol

The pump offers two protocol modes. Here the more comprehensive protocol *Model 44 Protocol* ist used. The protocol can be choosen by presseing the button *'SET'* following the button *'1'*. By pressing this button again it will toggle between two available protocols. The current setting is displayed on the pumps LCD and can be confirmed by pressing *'enter'*.

Futhermoter the pump should be operating in *'Vol Mode'* which can be selected by pressing the *'Select Mode'* button.

[2]The default adress is: 00
[3]PortObj->baudrate(9600);

3.3 Manuel Settings

Sometimes it can be useful to operate the pump manual, e.g. to refill the syringe, and thus it is necessary to cope with the pumps own interface. In the following describtion it is always expected that the pump operates in *'Vol Mode'*, which can be selected by the *'Select Mode'* button[4]. The pumping direction can be choosen via the *'Infuse/Refill'* button.

The pump is to be started by pressing *'Run/Stop'*. The pump can be stoped at any time by hitting this button again. In this case the pump will display an interrupt message including the pumped volume since the volume counter has been reset. The counter is reset each time its current value reaches the *'Target Volume'* amount. Besides while the pump is not activ the counter can be reset manually by hitting the *'Infuse/Refill'* button twice.

In order to change an arbitrary parameter a first hit of the *'Set'* button is necessary, followed by the parameter of interest. Wihtout pressing the *'Set'* button first the current value of the parameter will be displayed as long as the parameter button is pressed. The new value can be entered by using the keypad and confirmed with the *'enter'* button. If the value does not lie within the pumps limitations an error message showing *'OUT OF RANGE'* will be displayed. By pressing any button now the pump rejects the typed value and expects a new input. This will go on until an applicable value is entered.

All inputs are interpreted as values in units[5] which are shown in front of the parameter value. Hitting the correspondent parameter button while the pump expects a numeric input will change the unit.

[4]There are three modes: *'Vol Mode'*, *'Prog Mode'* and *'Pump Mode'*

[5]These are [mm], [um/mn], [um/hr], [ml/mn] and [ml/hr]

Chapter 4

Syringes

Four different Syringes were tested. Each of them has a standard LUER connector. Pictures of all tested syringes can be found in the Appendix. As it can be seen in Figure 4.1 the maximmum flow is limited by the pumps feeding speed and thus by the syringe diameter. The feeding speed was determined by increasing the flow rate until an error message appeared. The maximal feeding speed is $7.94mm/s$

Volume	Diameter	max. Inf. Rate	Manufacturer
1 ml	4.75 mm	3.3788 ml/min	BBraun
5 ml	12.2 mm	22.289 ml/min	Terumo
10 ml	16.2 mm	39.302 ml/min	Terumo
20 ml	20.15 mm	60.804 ml/min	Terumo
60 ml	29.5 mm	130.32 ml/min	Terumo

Table 4.1: Tested syringes for the PHD 2000 pump.

Chapter 5

Dropsize

Due to surface tension the size of a droplet mainly depends on the tubes geometry. This is valid as long as the flow of the liquid isn't too high. For slow flow rates a droplet can grow at the end of the tube and it will seperate from the tube if the weight exceeds the attractive forces due to surface tension and adhesion, cohesion. But this is a comparativley slow[1] and imprecise process. Pertubations like mechanical vibrations can lead to an earlie release before the actual size of the droplet is reached. It was observerd that for certain pumping speeds even the stepper motor inside the pump reached resonances which created enough vibration to shake the droplet off the needle.

A better approach is to 'shoot' a droplet out of the tube. Thus there is no time for the droplet to develop a large contact area to the tube or needle. The conditions of a separating droplet are much more well defined as in the described way before. This was verified in an experiment where the weight of five times 100 droplets where measured at different flow rates. The measurements show that the actual variance in dropsize at high flow rates ($> 20ml/min$) depends on the pumps accuracy and isn't influenced by external pertubations any longer.

5.1 Syringes and Tubes

Experiments showed that with flow rates greater than $16ml/min$ adequate results are achievable. With regard to the accesible flow rates, as it can be seen in Figure 4.1 and the possible accurancy which can be extracted from Figures A.2-A.2 a 5ml Syringe is generally recommended (cf. 6.1).

[1] For instance $50\mu l$ with a flow rate of $2ml/min$ takes $1.5s$ whereas wiht $20ml/min$ the same amount lasts only $16.7ms$

Syringe	Aperture	Flow Rate	Target Vol	Real Vol
5 ml	0.7 mm Needle	$16ml/min$	$5\mu l$	$5.5 \pm 0.1\mu l$
5 ml	1.0 mm Needle	$16ml/min$	$13\mu l$	$13.2 \pm 0.1\mu l$
5 ml	1.0 mm Needle	$20ml/min$	$18\mu l$	$18.6 \pm 0.2\mu l$
5 ml	3 mm silicone Tube	22 ml/min	60 μl	$62.5 \pm 0.3\mu l$
20 ml	3 mm PTFE Tube	20 ml/min	50 μl	$50.9 \pm 0.1\mu l$

Table 5.1: Tested combinations.

Especially if droplets about $10\mu l$ or less are desired there is no reasonable alternative.

Besides, each time a droplet separates from a needle or tube a smaller part is left behind. This can lead to an accumulation of a secondaray droplet which will also separate if its volume gets to big. Therefore it is very important to estimate the sufficient volume of the primary droplet. Thus the accucumulation would be kept small. Table 5.1 shows some usefull combinations and settings which are recommended if water or similar liquids[2] are used. A short procedure how to determin the sufficient volume is described in section 5.1.1. The column *'Real Volume'* in table 5.1 was determined by the actual pumped amount which is displayed on the pumps LCD, assuming that the variance in the actual volume of a droplet is negligible.

Furthermore, to obtain very small droplets ($< 10\mu l$) it is cruicial to ensure that no air is left inside the syringe. For high flow rates and small needles or tube apertures the air inside would rather be compressed than the liquid pushed out. In the following the slowly relaxing air would then lead to an decreased flowrate of the liquid which should be avoided as mentioned above.

5.1.1 Determining the Dropsize

If new parameters have to be determined there is a quick method which step by step approaches the optimal settings. But it should be taken into account that during this process a few $100\mu l$ of the liquid will be spend.

First of all the pump has to be set up with the syringe[3] and needle of choice. The liquid inside the syringe should also completely fill the needle. Therefore you can pump some liquid at a low flow rate[4] through

[2] In terms of compressibillity and surface tension, etc.

[3] Don't forget to update the diameter paramter accordingly.

[4] e.g. *'Infuse Rate* = $1ml/min$

the needle. If a constant dripping is observable the entire needle is filled with the liquid. Anyway any air inside the system has still to be avoided.

Before determining the actual dropsize it is necessary to know the absolut upper limit of a droplets volume. This is the volume a droplet would have if it could grow as slow as possible. In order to estimate this value the liquid is pumped through the needle again. This time the pump has to be stopped as soon as the first droplet separates. If no droplets are created, because a stream of liquid is produced the flow rate needs to be decreased. During this process the pumping Volume is set to a comparably high value[5] of e.g. *'Target Volume'* = $1ml$. If the pump is stopped as mentioned, this volume isn't spend entirely and the actualy pumped volume equals the desired value of the upper limit. The pumped volume is displayed on the LCD. Of course it is recommended to remove every droplet which juts out of the needle before. If necessary the volume counter can be reset by pressing the *'Infuse/Refill'* button twice.

Finally the flow rate, respectively the *'Infuse Rate'* can be increased to the final value of about $> 16ml/min$. Because they show a better reproducibilitty higher flow rates are generally prefered. But the generation of noise and vibrations should also be considered. Furthermore the pumps accurancy can play an important role (cf. Fig. A.2-A.2).

Nevertheless starting with the maximum drop size (upper limit) the *'Target Volume'* can be decreased step by step until only one single droplet is created each time the *'RUN/STOP'* button is pressed. Due to fine tuning the *'Target Volume'* it is possible to diminish the remaining part, after a droplet separated, unitl it almost vanishes.

5.2 Needle Preparation

The contact are between a droplet and a tube should be kept small. This increases reproducibility in dropsize and enables smaller droplets. Due to their bevel spike especially hypodermic needles need a special treatment. In addition the asymmetric spike complicates a precise readjustment of the needle, e.g. after a refill. Since the gravitational pull is also responsible for separating a droplet from the needle an oblique alignment lead to a variation in dropsize.

For these reasons all needles have been manually ground until a clean tube-like ending was achieved (cf. Fig. 5.1). In order to prevent the needle from clogging due to grit the needle was regularly flushed with water

[5] It doesn't really matter but the volume has to be lager than the volume which is to be determined

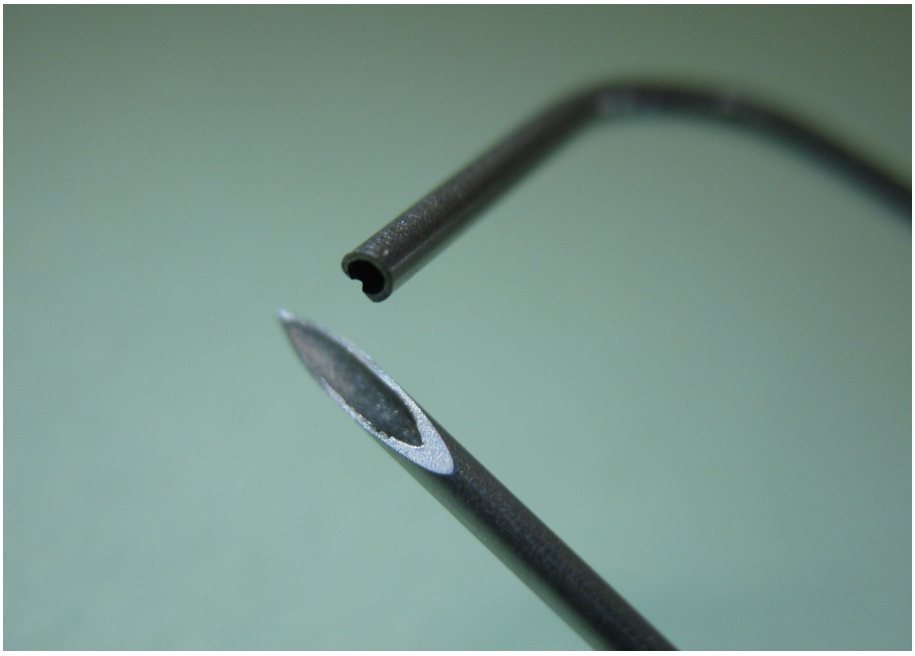


Figure 5.1: In order to minimize the cross section surface, the bevel spike was completley removed by manual grinding on sandpaper (400).

during the grinding. Some needles, particularly spinal needles, come with an inner none-hollow needle which also can ensure the outer needle not to clog if it is applied during the grinding.

Afterwards the needle needs to be cleaned. This was done by flushing and washing the needle with water and isopropanol. Because residua of the isopropanol can alter the wetting property of the metallic needle the first few droplets might differ in volume critically.

Chapter 6

Experiments

Doris comissioning 15.09.2008. First user experiments no experiments at the Beamline.

6.1 Minimal Volume

The minimal available pumping volume had been investigated in dependency on the applied flow rate and syringe diameter. Therefore the *'Target Volume'* was set to the lowest possible value of $0.1\mu l$. The *'Diameter'* was set with respect to $1ml$, $5ml$ and $10ml$ syringes (cf. 4.1).

The actual measurement was done by hitting the *'Run / Stop'* button once and reading off the volume counter, which is displayed on the LCD. In order to get an average value this process was repeated 5 times. Afterwards the flow rate was changed. Therefore the *'Infusion Rate'* was altered during the measurement in equidistant steps until the maximum was reached (cf. 4.1).

Figure 6.1 shows the result for $5ml$ syringes. The measurement for $1ml$ and $10ml$ syringes can be found in the appendix (cf. A.2-A.2). These plots obviously show that the minimal pumping volume is neither a constant value nor is there a global linear behaviour. This result is very important if small droplest ($< 10\mu l$) have to be otbained.

6.2 Droplets Diameter

When a droplet hits the surface of a substrate it will spread depending on the substrates wetting property and the height of fall. Therefore the droplets' diameter had been studied when varying the needle/substrate

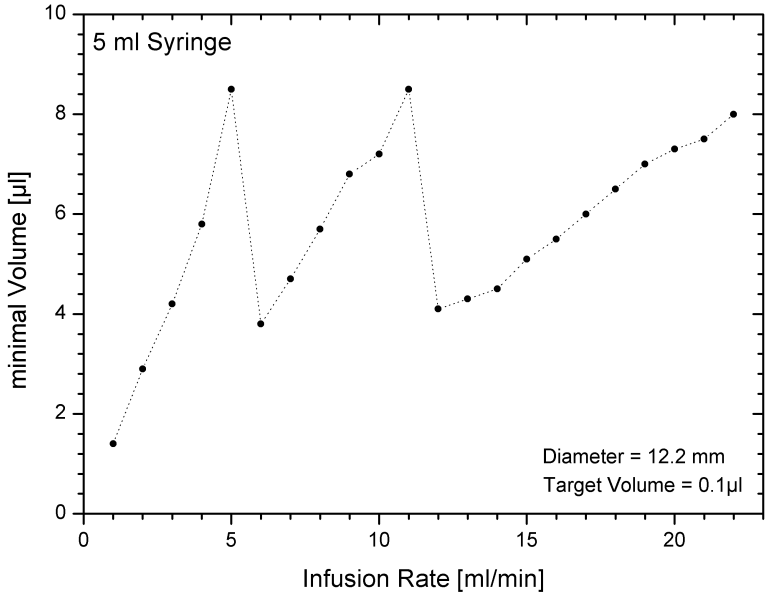


Figure 6.1:

distance. During this investigation common *Post-It* papers were chosen as substrate. In combination with water *Post-It*s show a very poor wettablity, thus we expect a strong dependency.

A 5ml Syringe together with a spinal needle (black, 0.7 x 90mm) was choosen to create tiny droplets. The Target Volume was set upt to $5\mu l$ and the Infuse Rate to $16ml/min$. In this way all droplets had the same volume of $5.5\mu l$ and the deviation was smaller than $0.1\mu l$ since the pump wasn't able to resolve it. The pump was manuelly operated. The height of fall was adjusted by a laboratory jack and measured with a caliper.

The results are shown in Figure 6.2. A caliper at the top end of the picture allows the measurement of the vertical and horizontal droplet diameter. Due to different scaling[1] of height and width a separate treatment is necessary. The diameter of each droplet was determined by averaging over horizontal and vertical diameter. If an uncertainty about 2 pixels in diameter measurement and 5 pixels in scaling is supposed the achievable failure will be within $\pm 0.04mm$. This is much smaller than the variances

[1]width-to-heithg ratio: first picture 0.692 , second picture 0.872

of the droplets diameter, as it can be seen in Figure 6.3.

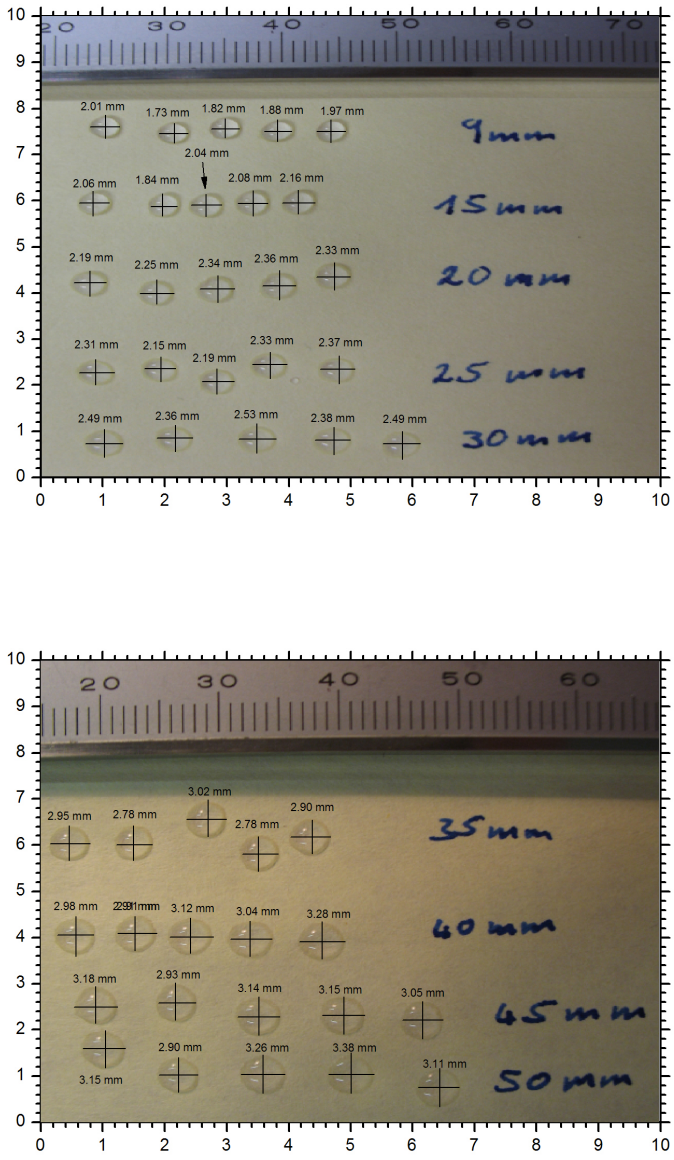


Figure 6.2: Du to paper thickness an offset of $-1.2mm$ has to be taken into account.

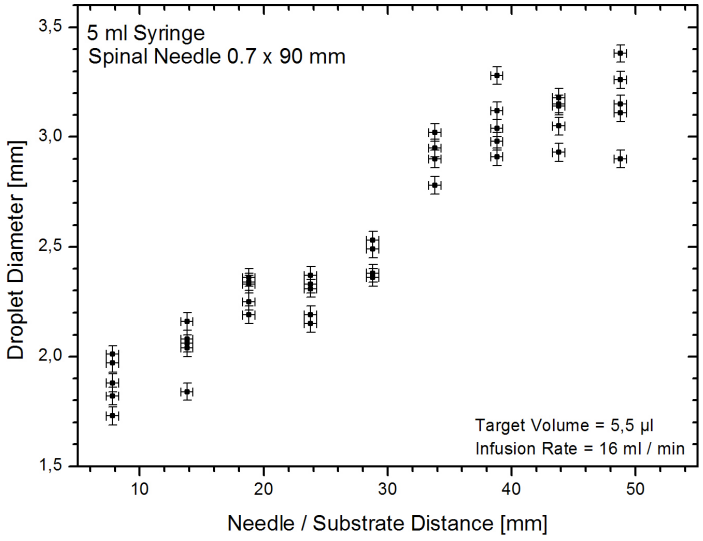
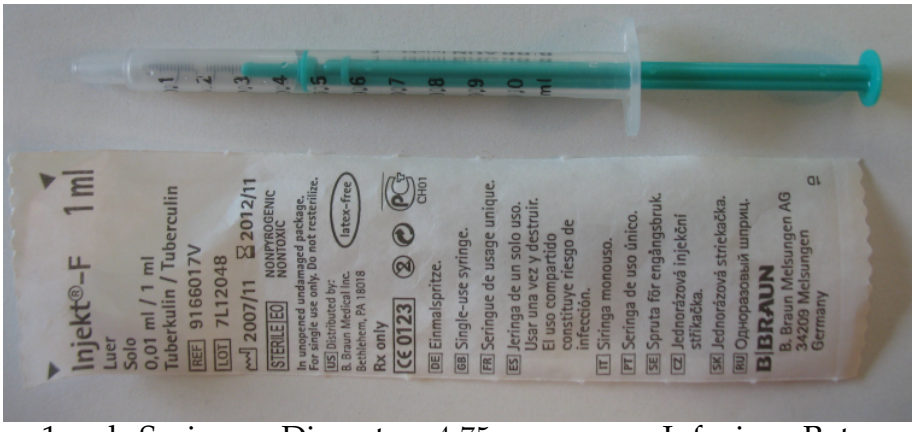


Figure 6.3: 2 pixel, 0.5mm height

Appendix A

Syringes and Needles

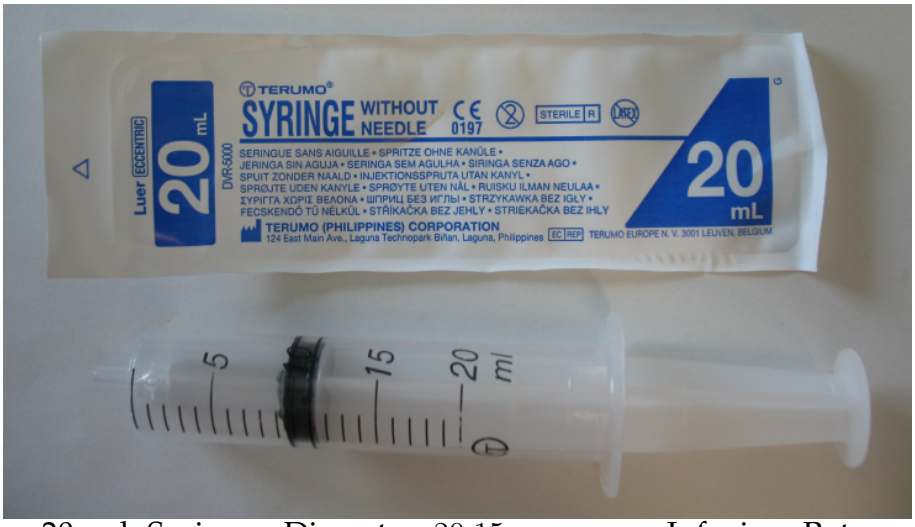
A.1 Syringes



1 ml Syringe, Diameter $4.75mm$, max Infusion Rate: $3.3788ml/min$

5 ml Syringe, Diameter $12.2mm$, max Infusion Rate: $22.289ml/min$

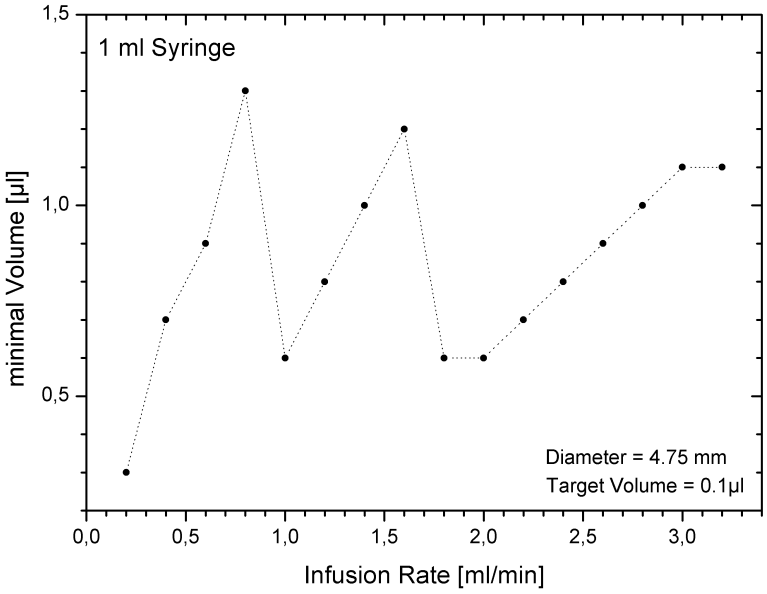
10 ml Syringe, Diameter $16.2mm$, max Infusion Rate: $39.302ml/min$



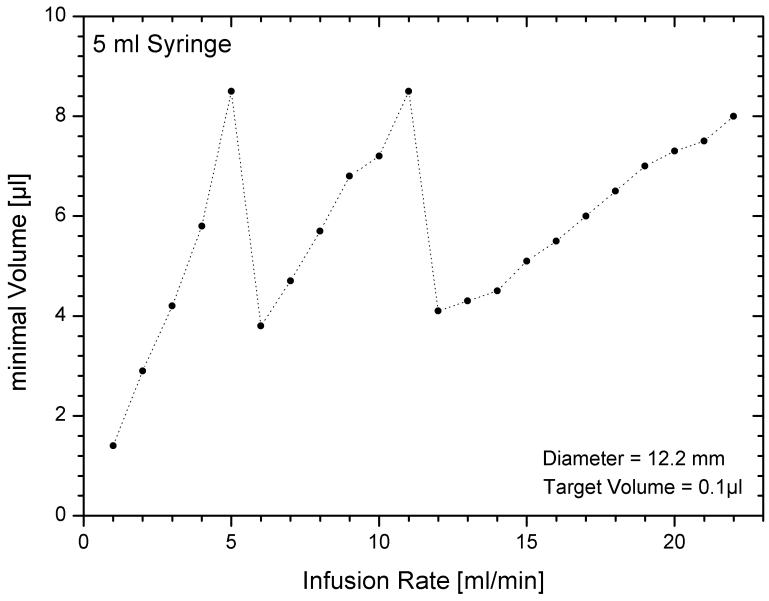
20 ml Syringe, Diameter $20.15mm$, max Infusion Rate: $60.804ml/min$

50 ml Syringe, Diameter $29.5mm$, max Infusion Rate: $130.32ml/min$

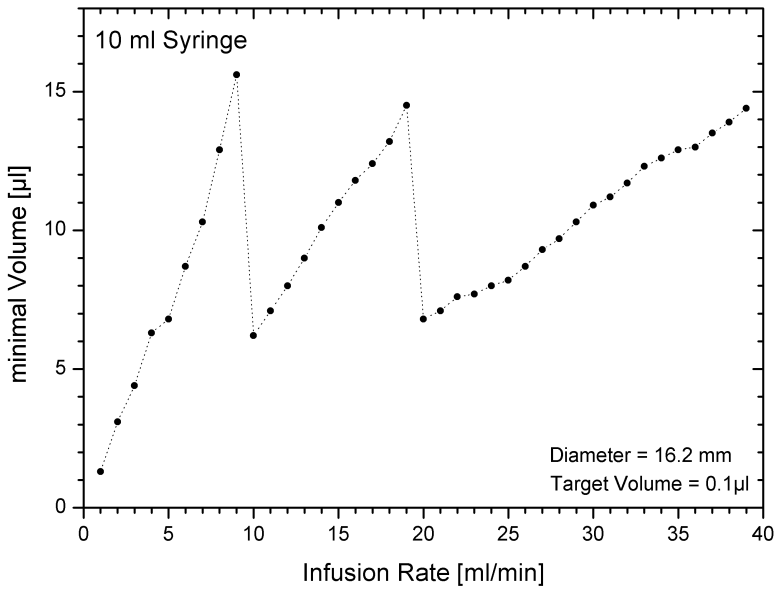
A.2 min Volume



minimal Volume vs. Infusion Rate : 1 ml Syringe



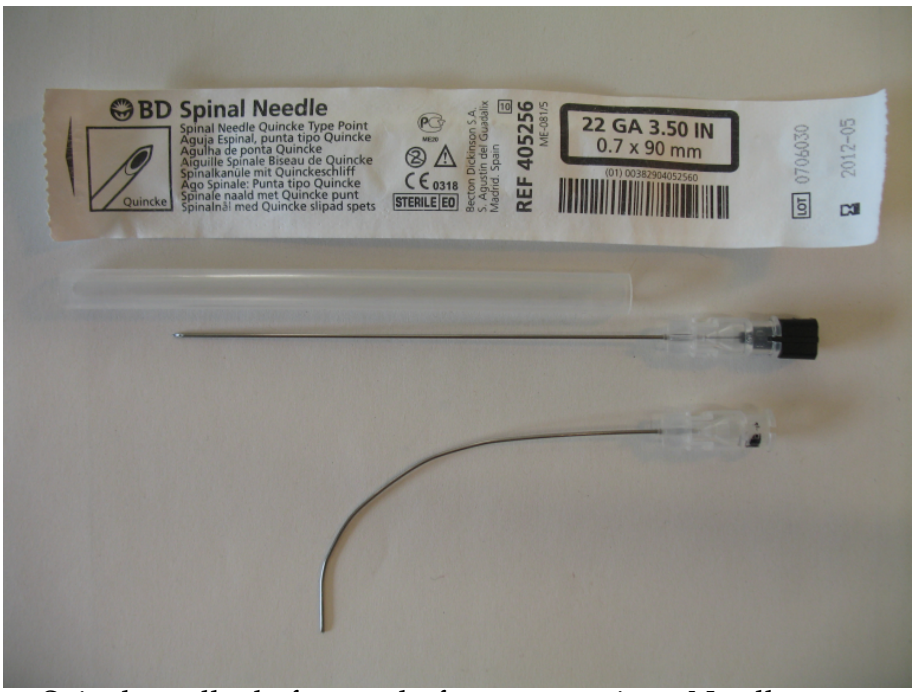
minimal Volume vs. Infusion Rate : 5 ml Syringe



minimal Volume vs. Infusion Rate : 10 ml Syringe

A.3 Needles

Hypodermic needle, before and after preparation. Needle aperture: 1 mm



Spinal needle, before and after preparation. Needle aperture: 0.7 mm

Spinal needle, before and after preparation. Needle aperture: 0.5 mm

Appendix B

The Perl Code

```
#!perl -w

use strict;


use vars qw($OS_win $PortObj $port);

BEGIN{
        $OS_win = ($^O eq "MSWin32") ? 1 : 0;
        if($OS_win){
                eval "use Win32::SerialPort qw(:STAT 0.19)";
                die "$@\n" if ($@);
        }
        else{
                eval "use Device::SerialPort";
                die "$@\n" if ($@);
        }
}


if($OS_win) {
        $port = "COM1"; #which port shell be used?
        $PortObj = Win32::SerialPort->new ($port);
        }
else {
        $port = '/dev/ttyS0'; #equals COM1
        $PortObj = Device::SerialPort->new($port);
}
die "Can't open serial port $port: $^E\n" unless ($PortObj);


#my $PortObj = Win32::SerialPort->new ("COM1")
#       or die "Can't open COM1: $^E\n";
$PortObj->baudrate(9600);
$PortObj->parity("none");
$PortObj->databits(8);
$PortObj->stopbits(2);
$PortObj->handshake("none");
$PortObj->buffers(4096,4096);


$PortObj->write_settings || undef $PortObj;



#----------------------
my $diameter;

my $inf_rate;
my $inf_unit;

my $ref_rate;
my $ref_unit;

my $target_vol;

```

```
58 my $tube_vol;
59
60 #--------------------
61 my $drop_vol = 0;                # [ml]
62 my $drop_save_vol = 0;  # [ml]
63 my $drop_flow = 0;               # [ml/min]
64 my $drop_save_flow = 0; # [ml/min]
65 #--------------------
66
67
68
69
70 my $str;
71 my $exit=1;
72 while($exit!=0){
73
74         print "\n-------- Program v0.8 ----------------------------\n";
75         print " main ..... Pump & Refill \n";
76         print " set ..... Settings\n";
77         print " debug ..... Debug Mode\n";
78         print " exit ..... Exit Programm\n";
79         print "-------------------------------------------------\n>";
80
81         chomp($str = <STDIN>);
82
83         if($str eq 'exit'){
84                 $exit=0;
85                 print " Exit Program!\n"
86         }
87         elsif($str eq 'main'){
88                 if($drop_vol == 0 || $drop_save_vol == 0 || $drop_flow == 0 || $drop_save_flow == 0){
89                 print " At least one value wasn't defined yet.\n Run the 'set -> dia' programme again.\n";
90                 }
91                 else{&MAIN;}
92         }
93         elsif($str eq 'set'){&SET;}
94         elsif($str eq 'debug'){&DEBUG;}
95         else{print "Wrong command!\n"}
96 }
97
98
99
100
101 sub MAIN{
102         my $cmd;
103         my $in = "";
104         my $x=1;
105
106         my $drops;
107         my $pause;
108
109         # calculate the some delays
110         my $drop_time = sprintf("%.1f",($drop_vol / $drop_flow * 60)+0.5);
111         my $drop_time_save = sprintf("%.0f",($drop_save_vol / $drop_save_flow * 60)+0.5);
112
113
114         while($x!=0){
115
116                 print "-------- Main ----------------------------------\n";
117                 print " fill ..... fill the tube\n";
118                 print " refill .... refill your system\n";
119                 print " drop ..... drop 1 drop\n";
120                 #print " flow ..... create a constant flow\n";
121                 #print " stop ..... stop the pump\n";
122                 print "-------------------------------------------------\nMain>";
123
124                 chomp($cmd = <STDIN>);
125                 if($cmd eq 'exit'){$x=0;}
126
127
128                 elsif($cmd eq 'drop'){
129
130
131                         print " How many drops do you want?\nMain\\drop>";
132                         chomp($drops = <STDIN>);
133
134
135                         if($drops>1){
136                                 print " Which repition rate [s]? For fastest type 0.\nMain\\drop>";
137                                 chomp($pause = <STDIN>);
138                         }
139                         else{$pause=0;}
140
```

```
141
142                     $PortObj->write("DIRINF\r");
143                     print "    set pump direction: infusion";
144                             select(undef,undef,undef,0.2);
145                     $in = $PortObj->input;
146                     if(length($in)>2 && substr($in,1,2) eq "0:"){print "\tdone\n";}
147                     else{print "\terror!\n";last;}
148
149
150                     $PortObj->write("TGT");
151                     $PortObj->write($drop_save_vol);
152                     $PortObj->write("\r");
153                     print "    set target volume to ", $drop_save_vol," ml";
154                             select(undef,undef,undef,0.2);
155                     $in = $PortObj->input;
156                     if(length($in)>2 && substr($in,1,2) eq "0:"){print "\tdone\n";}
157                     else{print "\terror!\n";last;}
158
159
160                     $PortObj->write("RAT");
161                     $PortObj->write($drop_save_flow);
162                     $PortObj->write("\r");
163                     print "    set infusion rate to ", $drop_save_flow, " ml/min";
164                             select(undef,undef,undef,0.2);
165                     $in = $PortObj->input;
166                     if(length($in)>2 && substr($in,1,2) eq "0:"){print "\tdone\n";}
167                     else{print "\terror!\n";last;}
168
169
170                     $PortObj->write("RFR");
171                     $PortObj->write($drop_save_flow);
172                     $PortObj->write("\r");
173                     print "    set refill rate to ", $drop_save_flow," ml/min";
174                             select(undef,undef,undef,0.2);
175                     $in = $PortObj->input;
176                     if(length($in)>2 && substr($in,1,2) eq "0:"){print "\tdone\n";}
177                     else{print "\terror!\n";last;}
178
179
180                     $PortObj->write("RUN\r");
181                     print "    pumping";
182                             sleep $drop_time_save;
183                     $PortObj->write("DEL\r");
184                             select(undef,undef,undef,0.2);
185                     $in = $PortObj->input;
186                     print "\t\t\t",substr($in,6,6)," ml\n";
187
188
189                     $PortObj->write("TGT");
190                     $PortObj->write($drop_vol); ### drop size
191                     $PortObj->write("\r");
192                     print "    set target volume to ", $drop_vol *1000, "ul";
193                             select(undef,undef,undef,0.2);
194                     $in = $PortObj->input;
195                     if(length($in)>2 && substr($in,1,2) eq "0:"){print "\t\tdone\n";}
196                     else{print "\t\terror!\n";last;}
197
198
199                     $PortObj->write("RAT");
200                     $PortObj->write($drop_flow);
201                     $PortObj->write("\r");
202                     print "    set infusion rate to ", $drop_flow," ml/min";
203                             select(undef,undef,undef,0.2);
204                     $in = $PortObj->input;
205                     if(length($in)>2 && substr($in,1,2) eq "0:"){print "\tdone\n";}
206                     else{print "\terror!\n";last;}
207
208
209                     for (my $i=1;$i<=$drops;$i++){
210                             $PortObj->write("RUN\r");
211                             print "    pumping ", $i;
212                                     select(undef,undef,undef,$drop_time+$pause);
213                             $PortObj->write("DEL\r");
214                                     select(undef,undef,undef,0.2);
215                             $in = $PortObj->input;
216                             if(length($in)>13){print "\t\t\t",substr($in,6,6)*1000," ul\n";}
217                             else{print "\terror!\n";last;}
218                     }
219
220
221                     $PortObj->write("DIRREF\r");
222                     print "    set pump direction: refill ...";
223                             select(undef,undef,undef,0.2);
```

```
224                     $in = $PortObj->input;
225                     if(length($in)>2 && substr($in,1,2) eq "0:"){print "\tdone\n";}
226                     else{print "\terror!\n";last;}
227
228
229                     $PortObj->write("TGT");
230                     $PortObj->write($drop_save_vol);
231                     $PortObj->write("\r");
232                     print "    ....set target volume to ", $drop_save_vol," ml";
233                             select(undef,undef,undef,0.2);
234                     $in = $PortObj->input;
235                     if(length($in)>2 && substr($in,1,2) eq "0:"){print "\tdone\n";}
236                     else{print "\terror!\n";last;}
237
238
239                     $PortObj->write("RUN\r");
240                     print "    ....pumping";
241                             sleep $drop_time_save;
242                     $PortObj->write("DEL\r");
243                             select(undef,undef,undef,0.2);
244                     $in = $PortObj->input;
245                     print "\t\t\t",substr($in,6,6)," ml\n";
246
247
248             }
249
250
251             elsif($cmd eq 'flow'){
252
253
254                     print " Please specify the desired infusion rate [ml/s] \nMain\\flow>";
255                     chomp($inf_rate = <STDIN>);
256                     $PortObj->write("RAT");
257                     $PortObj->write($inf_rate*60);
258                     $PortObj->write("MM\r");
259
260                     select(undef,undef,undef,0.2);
261
262                     $PortObj->write("RAT");
263                     $PortObj->write("\r");
264
265                     select(undef,undef,undef,0.2);
266
267                     $in = $PortObj->input;
268
269                     if(substr($in,3,3) eq 'OOR'){
270                             print "\a This is not possible. The original value has not been changed.\n";
271                             $inf_rate = substr($in,12,6);
272                             $inf_unit = substr($in,20,5);
273                     }
274                     else{
275                             $inf_rate = substr($in,6,6);
276                             $inf_unit = substr($in,13,5);
277                     }
278                     if($inf_unit eq 'ml/mn'){$inf_rate=$inf_rate/60; $inf_unit=" ml/s";}
279                     print "\a Infuse Rate = ", $inf_rate , $inf_unit,"\n\n";
280
281
282
283
284                     print " Please specify the target volume [ml]\nMain\\flow>";
285                     chomp($target_vol = <STDIN>);
286
287                     $PortObj->write("TGT");
288                     $PortObj->write($target_vol);
289                     $PortObj->write("\r");
290
291                     select(undef,undef,undef,0.2);
292
293                     $PortObj->write("TGT");
294                     $PortObj->write("\r");
295                     select(undef,undef,undef,0.2);
296                     $in = $PortObj->input;
297
298                     if(substr($in,3,1) eq '?'){
299                             print "\a This is not possible. The original value has not been changed.\n";
300                             $target_vol = substr($in,12,6);
301                     }
302                     else{
303                             $target_vol = substr($in,6,6);
304                     }
305                     print "\a Target Volume = ", $target_vol, " ml\n\n";
306
```

```
307
308                     $PortObj->write("DIRINF\r");
309                     print "    set pump direction: infusion";
310                     select(undef,undef,undef,0.2);
311                     print "\tdone\n";
312
313
314
315                     print "\n Do you want to start now? [yes/no]\nMain\\fill>";
316                     chomp($in = <STDIN>);
317                     if($in eq 'yes'){
318                             $PortObj->write("RUN\r");
319                             print "    pumping   ";
320                     }
321                     else{print "    aborting\n\n";}
322
323
324
325
326
327
328             }
329
330
331             elsif($cmd eq 'fill'){
332                     print " Please enter the tube Volume [ml].\nMain\\fill>";
333                     chomp($tube_vol = <STDIN>);
334
335                     print "    set pump direction: infusion ";
336                     $PortObj->write("DIRINF\r");
337                     sleep 1;
338                     print "\tdone\n";
339
340                     $PortObj->write("TGT");
341                     $PortObj->write($tube_vol);
342                     $PortObj->write("\r");
343                     print "    set target volume to ", $tube_vol, "ml.";
344                     sleep 1;
345                     print "\t\tdone\n";
346
347                     $PortObj->write("RAT50MM\r");
348                     print "    set infusion rate to 50 ml/min ";
349                     sleep 1;
350                     print "\tdone\n";
351
352                     print "\n Do you want to fill the tube now? [yes/no]\nMain\\fill>";
353                     chomp($in = <STDIN>);
354                     if($in eq 'yes'){
355
356                             $pause = sprintf("%.0f",($tube_vol / 50 * 60)+1);
357
358                             $PortObj->write("RUN\r");
359                             print "    pumping ";
360                             select(undef,undef,undef,$pause);
361                             print "\t\t\tdone\n";
362
363
364                             $PortObj->write("DIRREF\r");
365                             print "    set pump direction: refill ";
366                             select(undef,undef,undef,0.2);
367                             print "\tdone\n";
368
369                             $PortObj->write("RFR");
370                             $PortObj->write($drop_save_flow);
371                             $PortObj->write("\r");
372                             print "    set refill rate to ", $drop_save_flow," ml/min";
373                             select(undef,undef,undef,0.2);
374                             print "\tdone\n";
375
376
377                             $PortObj->write("TGT");
378                             $PortObj->write($drop_save_vol);
379                             $PortObj->write("\r");
380                             print "    set target volume to ", $drop_save_vol," ml";
381
382                             select(undef,undef,undef,0.2);
383                             print "\tdone\n";
384
385                             $PortObj->write("RUN\r");
386                             print "    pumping";
387                             sleep $drop_time_save;
388                             print"\t\t\tdone\n\n";
389
```

```
390
391
392                     }
393                     else{print "    ... aborting ...\n\n";}
394             }
395
396
397             elsif($cmd eq 'refill'){
398
399                     print "Please enter the refill volume?\nMain\\refill>";
400                     chomp($target_vol = <STDIN>);
401
402
403
404                     $PortObj->write("DIRINF\r");
405                     print "    ... set pump direction: infusion";
406                     select(undef,undef,undef,0.2);
407                     print "\tdone\n";
408
409                     $PortObj->write("TGT");
410                     $PortObj->write($drop_save_vol);
411                     $PortObj->write("\r");
412                     print "    ... set target volume to ", $drop_save_vol," ml";
413                     select(undef,undef,undef,0.2);
414                     print "\tdone\n";
415
416                     $PortObj->write("RAT");
417                     $PortObj->write($drop_save_flow);
418                     $PortObj->write("\r");
419                     print "    ... set infusion rate to ", $drop_save_flow, " ml/min";
420                     select(undef,undef,undef,0.2);
421                     print "\tdone\n";
422
423
424                     print "\n Make sure the syringe has access to the liquid.\nDo you want to refill now? [yes/no]\nMain\\refill>";
425                     chomp($in = <STDIN>);
426                     if($in eq 'yes'){
427
428                             $pause = sprintf("%.0f",(($target_vol+1) / 50 * 60)+1);
429
430
431                             $PortObj->write("RUN\r");
432                             print "    ... pumping";
433                             sleep $drop_time_save;
434                             print"\t\t\tdone\n\n";
435
436                             $PortObj->write("DIRREF\r");
437                             print "    ... set pump direction: refill";
438                             select(undef,undef,undef,0.2);
439                             print "\tdone\n";
440
441                             $PortObj->write("RFR50MM\r");
442                             print "    ... set refill rate to 50 ml/min";
443                             select(undef,undef,undef,0.2);
444                             print "\tdone\n";
445
446                             $PortObj->write("TGT");
447                             $PortObj->write($target_vol+1);
448                             $PortObj->write("\r");
449                             print "    ... set target volume to ", $target_vol+1," ml";
450                             select(undef,undef,undef,0.2);
451                             print "\tdone\n";
452
453                             $PortObj->write("RUN\r");
454                             print "    ... pumping ";
455                             select(undef,undef,undef,$pause);
456                             print "\t\t\tdone\n\n";
457
458
459
460
461                             $pause = sprintf("%.0f",((1) / 20 * 60)+1);
462
463                             $PortObj->write("DIRINF\r");
464                             print "    ... set pump direction: infuse";
465                             select(undef,undef,undef,0.2);
466                             print "\tdone\n";
467
468                             $PortObj->write("RFR20MM\r");
469                             print "    ... set refill rate to 20 ml/min";
470                             select(undef,undef,undef,0.2);
471                             print "\tdone\n";
472
```

```
473                                     $PortObj->write("TGT1\r");
474                                     print "    set target volume to 1 ml";
475                                     select(undef,undef,undef,0.2);
476                                     print "\tdone\n";
477
478                                     $PortObj->write("RUN\r");
479                                     print "    pumping ";
480                                     select(undef,undef,undef,$pause);
481                                     print "\t\t\tdone\n\n";
482
483
484
485
486
487
488                                     $in="";
489                                     print "\n Make sure the syringe has NO access to the liquid anymore.[ok]\nMain\\refill>";
490                                     chomp($in = <STDIN>);
491                                     if($in eq 'ok'){
492
493                                             $PortObj->write("DIRREF\r");
494                                             print "    set pump direction: refill ";
495                                             select(undef,undef,undef,0.2);
496                                             print "\tdone\n";
497
498                                             $PortObj->write("RFR");
499                                             $PortObj->write($drop_save_flow);
500                                             $PortObj->write("\r");
501                                             print "    set refill rate to ", $drop_save_flow," ml/min";
502                                             select(undef,undef,undef,0.2);
503                                             print "\tdone\n";
504
505
506                                             $PortObj->write("TGT");
507                                             $PortObj->write($drop_save_vol);
508                                             $PortObj->write("\r");
509                                             print "    set target volume to ", $drop_save_vol,"ml";
510
511                                             select(undef,undef,undef,0.2);
512                                             print "\tdone\n";
513
514                                             $PortObj->write("RUN\r");
515                                             print "    pumping";
516                                             sleep $drop_time_save;
517                                             print"\t\t\tdone\n\n";
518                                     }
519                                     else{print "    aborting \n\n";}
520
521
522                             }
523                             else{print "    aborting\n\n";}
524
525
526
527
528
529
530
531                     }
532
533
534                     elsif($cmd eq 'stop'){
535                             $PortObj->write("STP");
536                             $PortObj->write("\r");
537                     }
538
539                     else{print "Wrong command!\n"}
540             }
541
542 print "\n\n ";
543
544
545 }
546
547
548
549
550 sub SET{
551             my $x=1;
552             my $cmd;
553             my $in="";
554
555
```

```
556         while($x!=0){
557                 print "--------Settings----------------------------------\n";
558                 print "all ..... get all parameters\n";
559                 print "dia ..... set diameter and all other paramters\n";
560                 #print "inf ..... set infuse rate\n";
561                 #print "ref ..... set refill rate\n";
562                 #print "vol ..... set target volume\n";
563                 print "exit ..... return to main menu\n";
564                 print "-----------------------------------------------\nSettings>";
565
566                 chomp($cmd = <STDIN>);
567                 if($cmd eq 'exit'){$x=0;}
568
569                 elsif($cmd eq 'all'){
570
571
572                         # GET DIAMETER
573                         $PortObj->write("DIA\r");
574                         print " Diameter    =\t";
575                                 select(undef,undef,undef,0.2);
576                         $in = $PortObj->input;
577                         if(length($in)>2 && substr($in,11,2) eq "0:"){
578                                 print substr($in,3,6), "\tmm\n";
579                                 }
580                         else{print "\terror!\n";last;}
581
582
583                         # GET INFUSE RATE
584                         $PortObj->write("RAT\r");
585                         print " Infuse Rate =\t";
586                                 select(undef,undef,undef,0.2);
587                         $in = $PortObj->input;
588                         if(length($in)>2 && substr($in,17,2) eq "0:"){
589                                 $inf_unit = substr($in,10,5);
590                                 $inf_rate = substr($in,3,6);
591                                 print $inf_rate, "\t", $inf_unit,"\n";
592                                 }
593                         else{print "\terror!\n";last;}
594
595
596                         # GET REFILL RATE
597                         $PortObj->write("RFR\r");
598                         print " Refill Rate =\t";
599                                 select(undef,undef,undef,0.2);
600                         $in = $PortObj->input;
601                         if(length($in)>2 && substr($in,17,2) eq "0:"){
602                                 $ref_unit = substr($in,10,5);
603                                 $ref_rate = substr($in,3,6);
604                                 print $ref_rate,"\t". $ref_unit,"\n";
605                                 }
606                         else{print "\terror!\n";last;}
607
608
609                         # GET TARGET VOLUME
610                         $PortObj->write("TGT\r");
611                         print " Target Vol. =\t";
612                                 select(undef,undef,undef,0.2);
613                         $in = $PortObj->input;
614                         if(length($in)>2 && substr($in,11,2) eq "0:"){
615                                 print substr($in,3,6), "\tml\n";
616                                 }
617                         else{print "\terror!\n";last;}
618
619
620                         print".....\n";
621                         print " Droplet Vol. =\t", $drop_vol, "\tml\n";
622                         print " Droplet Flow =\t", $drop_flow, "\tml/mn\n";
623                         print " Save Vol.    =\t", $drop_save_vol, "\tml\n";
624                         print " Save Flow    =\t", $drop_save_flow, "\tml/mn\n\n";
625
626
627                 }
628
629                 elsif($cmd eq 'dia'){
630
631
632                         # SET DIAMETER
633                         print " Diameter [mm] :\t";
634                         chomp($cmd = <STDIN>);
635                         $PortObj->write("DIA");
636                         $PortObj->write($cmd);
637                         $PortObj->write("\r");
638                                 select(undef,undef,undef,0.2);
```

```
$in = $PortObj->input;
if(substr($in,3,3) eq 'OOR'){
        print "\a This is not possible. The original value has not been changed.\n";
        last;
}
elsif(substr($in,1,2) eq '0:'){
        # GET DIAMETER
        $PortObj->write("DIA\r");
        print " Diameter =\t\t";
                select(undef,undef,undef,0.2);
        $in = $PortObj->input;
        if(length($in)>2 && substr($in,11,2) eq "0:"){
                print substr($in,3,6), "\tmm\n";
                $diameter = substr($in,3,6);
                }
        else{print "\terror!\n";last;}
}
else{print "\terror!\n";last;}


# SET DROPLET VOLUME
print " Droplet Vol [ul] :\t";
chomp($cmd = <STDIN>);
$PortObj->write("TGT");
$PortObj->write($cmd/1000);
$PortObj->write("\r");
        select(undef,undef,undef,0.2);
$in = $PortObj->input;
if(substr($in,3,3) eq 'OOR'){
        print "\a This is not possible.\n";
        last;
}
elsif(substr($in,1,2) eq '0:'){
        # GET TARGET VOLUME
        $PortObj->write("TGT\r");
        print " Droplet Vol.   =\t";
                select(undef,undef,undef,0.2);
        $in = $PortObj->input;
        if(length($in)>2 && substr($in,11,2) eq "0:"){
                print substr($in,3,6)*1000, "\tul\n";
                $drop_vol = substr($in,3,6);
                }
        else{print "\terror!\n";last;}
}
else{print "\terror!\n";last;}


# SET DROPLET FLOW
print " Droplet FLOW [ml/mn] :\t";
chomp($cmd = <STDIN>);
$PortObj->write("RAT");
$PortObj->write($cmd);
$PortObj->write("MM\r");
        select(undef,undef,undef,0.2);
$in = $PortObj->input;
if(substr($in,3,3) eq 'OOR'){
        print "\a This is not possible.\n";
        last;
}
elsif(substr($in,1,2) eq '0:'){
        # GET INFUSE RATE
        $PortObj->write("RAT\r");
        print " Droplet Flow =\t\t";
                select(undef,undef,undef,0.2);
        $in = $PortObj->input;
        if(length($in)>2 && substr($in,17,2) eq "0:"){
                $inf_unit = substr($in,10,5);
                $inf_rate = substr($in,3,6);
                print $inf_rate, "\t", $inf_unit,"\n";
                $drop_flow = substr($in,3,6);
        }
        else{print "\terror!\n";last;}
}
else{print "\terror!\n";last;}


# SET SAVE VOLUME
print " Save Vol [ml] :\t";
chomp($cmd = <STDIN>);
$PortObj->write("TGT");
$PortObj->write($cmd);
$PortObj->write("\r");
        select(undef,undef,undef,0.2);
```

```
722                     $in = $PortObj->input;
723                     if(substr($in,3,3) eq 'OOR'){
724                             print "\a This is not possible.\n";
725                             last;
726                     }
727                     elsif(substr($in,1,2) eq '0:'){
728                             # GET TARGET VOLUME
729                             $PortObj->write("TGT\r");
730                             print " Save Vol. = \t\t";
731                                     select(undef,undef,undef,0.2);
732                             $in = $PortObj->input;
733                             if(length($in)>2 && substr($in,11,2) eq "0:"){
734                                     print substr($in,3,6), "\tml\n";
735                                     $drop_save_vol = substr($in,3,6);
736                                     }
737                             else{print "\terror!\n";last;}
738                     }
739                     else{print "\terror!\n";last;}
740
741
742                     # SET Save FLOW
743                     print " Save FLOW [ml/mn] :\t";
744                     chomp($cmd = <STDIN>);
745                     $PortObj->write("RAT");
746                     $PortObj->write($cmd);
747                     $PortObj->write("MM\r");
748                             select(undef,undef,undef,0.2);
749                     $in = $PortObj->input;
750                     if(substr($in,3,3) eq 'OOR'){
751                             print "\a This is not possible.\n";
752                             last;
753                     }
754                     elsif(substr($in,1,2) eq '0:'){
755                             # GET INFUSE RATE
756                             $PortObj->write("RAT\r");
757                             print " Save Flow =\t\t";
758                                     select(undef,undef,undef,0.2);
759                             $in = $PortObj->input;
760                             if(length($in)>2 && substr($in,17,2) eq "0:"){
761                                     $inf_unit = substr($in,10,5);
762                                     $inf_rate = substr($in,3,6);
763                                     print $inf_rate, "\t", $inf_unit,"\n";
764                                     $drop_save_flow = substr($in,3,6);
765                             }
766                             else{print "\terror!\n";last;}
767                     }
768                     else{print "\terror!\n";last;}
769
770
771                     print "\n\n";
772
773             }
774
775             elsif($cmd eq 'inf'){
776
777                     print " Please specify the desired infusion rate [ul/s] \nSettings\\Infusion Rate>";
778                     chomp($cmd = <STDIN>);
779                     $PortObj->write("RAT");
780                     $PortObj->write($cmd*60);
781                     $PortObj->write("UM\r");
782
783                     #sleep 1;
784                     select(undef,undef,undef,0.2);
785
786                     $PortObj->write("RAT");
787                     $PortObj->write("\r");
788                     #sleep 1;
789                     select(undef,undef,undef,0.2);
790                     $in = $PortObj->input;
791
792                     #print " \a", $in, "\n";
793
794                     if(substr($in,3,3) eq 'OOR'){
795                             print "\a This is not possible. The original value has not been changed.\n";
796                             $inf_rate = substr($in,12,6);
797                             $inf_unit = substr($in,20,5);
798                     }
799                     else{
800                             $inf_rate = substr($in,6,6);
801                             $inf_unit = substr($in,13,5);
802                     }
803                     if($inf_unit eq 'ul/mn'){$inf_rate=$inf_rate/60; $inf_unit=" ul/s";}
804                     print "\a Infuse Rate = ", $inf_rate , $inf_unit,"\n\n";
```

```
805
806
807                     $in="";
808
809             }
810
811             elsif($cmd eq 'ref'){
812
813                     print "␣Please␣specify␣the␣desired␣refill␣rate␣[ml/min]␣\nSettings\\Refill␣Rate>";
814                     chomp($cmd = <STDIN>);
815                     $PortObj->write("RFR");
816                     $PortObj->write($cmd);
817                     $PortObj->write("MM\r");
818
819                     #sleep 1;
820                     select(undef,undef,undef,0.2);
821
822                     $PortObj->write("RFR");
823                     $PortObj->write("\r");
824                     #sleep 1;
825                     select(undef,undef,undef,0.2);
826                     $in = $PortObj->input;
827
828                     #print " \a", $in, "\n";
829
830                     if(substr($in,3,3) eq 'OOR'){
831                             print "\a␣This␣is␣not␣possible.␣The␣original␣value␣has␣not␣been␣changed.\n";
832                             $ref_rate = substr($in,12,6);
833                             $ref_unit = substr($in,20,5);
834                     }
835                     else{
836                             $ref_rate = substr($in,6,6);
837                             $ref_unit = substr($in,13,5);
838                     }
839                     if($ref_unit eq 'ml/mn'){$inf_rate=$inf_rate; $ref_unit="␣ml/min";}
840                     print "\a␣Refill␣Rate␣=␣", $ref_rate , $ref_unit,"\n\n";
841
842
843                     $in="";
844
845             }
846
847             elsif($cmd eq 'vol'){
848
849                     print "␣Please␣specify␣the␣target␣volume␣[ml]␣\nSettings\\Target␣Volume>";
850                     chomp($cmd = <STDIN>);
851                     $PortObj->write("TGT");
852                     $PortObj->write($cmd);
853                     $PortObj->write("\r");
854
855                     #sleep 1;
856                     select(undef,undef,undef,0.2);
857
858                     $PortObj->write("TGT");
859                     $PortObj->write("\r");
860                     #sleep 1;
861                     select(undef,undef,undef,0.2);
862                     $in = $PortObj->input;
863
864                     #print " \a", $in, "\n";
865
866
867                     if(substr($in,3,1) eq '?'){
868                             print "\a␣This␣is␣not␣possible.␣The␣original␣value␣has␣not␣been␣changed.\n";
869                             $target_vol = substr($in,12,6);
870                     }
871                     else{
872                             $target_vol = substr($in,6,6);
873                     }
874
875
876
877                     print "\a␣Target␣Volume␣=␣", $target_vol, "␣ml\n\n";
878
879
880
881                     $in="";
882
883             }
884
885
886
887             else{print "Wrong␣command!\n";}
```

```
888
889             }
890             print"\n\n";
891     }
892
893
894
895
896     sub DEBUG{
897
898             my $cmd;
899             my $in = "";
900             my $x=1;
901
902             print "--------Debug----------\nDEBUG>";
903
904             while($x!=0){
905                     chomp($cmd = <STDIN>);
906                     if($cmd eq 'exit'){$x=0;}
907                     else{
908                             $PortObj->write($cmd);
909                             $PortObj->write("\r");
910                             #sleep 1;
911                             select(undef,undef,undef,0.2);
912                             $in = $PortObj->input;
913                             print " ",$in;
914                             $in="";
915                     print "\n----------\nDEBUG>";
916                     }
917             }
918             print "\n\n";
919     }
920
921
922
923
924
925
926
927
928
929     $PortObj->close
930             or die "failed to close";
931     undef $PortObj;
```

Bibliography

[APL07] APL 2007, Shiffalovic Small and PRB

[Mou08] J.-F. Moulin, S.V. Roth, and P. Müller-Buschbaum, Review of Scientific Instruments ,**79**, 015109 (2008)