**Brief description of the programming language**

***Introduction***

**variables:**

all variables must start and end with a dollar sign.

Example:

$x$

Variables are case sensitive, so $x$ is different from $X$

The variable $return$ is RESERVED. It is used to return values from macros.

Variables named $1$, $2$... are automatically assigned when parameters are sent to a macro.

To assign an initial value to a variable the eval command must be used:

eval $x$,1

corresponds to $x$=1

**comments:**

line following the semicolon (;) are considered as comment and ignored

**log file:**

When program is executed and the connect button is pressed the variables recorded (temperature etc.) will be automatically appended to log.txt file every 500 ms.

The following header is added first:

----------------------------------

- PROCESS STARTS -

----------------------------------

date time

When the disconnect button is pressed the logging is interrupted and the following text is appended to log.txt:

---------------------------------

- PROCESS ENDED -

---------------------------------

date time

This helps to find the log data in the file.

***Commands list***

**log**

Print string to log file

use: log text

The command adds the text to the log queue.

variables contained in text will be replaced automatically.

Example1:

log sample taken

it writes 'sample taken' into the log queue

Example2:

eval $x$,1

log x=$x$

it writes 'x=1' in the log queue

**send**

Send a Gcode command

use: send command,where

where indicates the number of the device for gcode output (0=SyringeBot, 1=Robot).

Example:

send G28 X Y,1

Please note that due to unsynchronized connections, only few commands (<10) can be sent in a sequence. For better performances and longer sequences use commands **buffer** and **print**

Example:

buffer

macro "syr1\_X\_ml"30

macro "syr1\_X\_ml"60

macro "syr1\_X\_ml"60

macro "purge syringe 1"

print

**buffer**

Buffer all commands; send later (with print). Used for long gcode sequences where the base command **send** will fail.

**print**

Send all buffered commands (see **buffer** command). Used for long gcode sequences where **send** will fail.

**ask**

Ask to the user the insertion of a numerical input.

use: ask $varname$,title,question,initialvalue,minvalue,maxvalue

The procedure creates a window with title *title*, outputs the text contained in *question*, sets the initial value to *initialvalue* and limits the input range from *minvalue* to *maxvalue*.

Example:

buffer

ask $ML$,Syringe 1,Enter how many ml to put,10,10,60

macro "syr1\_X\_ml"$ML$

print

**eval**

Evaluate a mathematical expression.

use: eval $varname$,math\_expression

Example:

ask $a$,a value,Enter the value of a,1,1,10

ask $b$,b value,Enter the value of b,1,1,10

eval $sum$,$a$+$b$

echo the sum of a and b is $sum$

**exec**

Executes Python's code. Code might contain variables ($..$). Code on multiple lines must be separated with /n

use: exec code!,varname1=$var1$,...

**macro**

Call an existing macro

use: macro "macroname" var1,var2...

This command executes a macro with the name macroname.

All the macro files are saved in the directory macros with the extensions .txt and they can be edited with a plain text editor. Macro name can contain spaces.

It is possible to pass parameters to a macro by indicating their values: var1,var2...

Example:

macro "syr1\_X\_ml"20

If parameters are passed (var1,var2...) variables $1$... will be automatically created and assigned.

In the example above, the macro syr1\_X\_ml is called and the value 20 is inserted in the variable $1$ of the macro

**echo**

Print to the shell console. Variables are automatically substituted with their values before printing.

use: echo text $varname$

Example:

ask $X$,Vial X position,Enter vial X position,1,1,8

ask $Y$,Vial Y position,Enter vial Y position,1,1,6

eval $X$,58-($X$-1)\*15.71328

eval $Y$,121+($Y$-1)\*14.8

echo X=$X$ Y=$Y$

**message**

Like **echo** but outputs the message as a window message box

use: message text $varname$

Example:

eval $a$,1

message The value of a is $a$