



מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE

Robolab-User Manual

By Shai Kaplan



Table of Contents

TABLE OF CONTENTS	2
1 ROBOLAB – A LANGUAGE FOR LABORATORY ROBOT	
PROGRAMMING	5
1.1 About Robolab	5
1.2 Introduction to Tecan Genesis Laboratory Robot.....	5
1.3 Description of the Development Environment	8
1.4 Robolab Program Structure.....	10
1.5 A simple Robolab program example	10
1.6 Abstraction of Robot Entities.....	13
1.6.1 Reagents & Mixes.....	13
1.6.2 Table Locations.....	16
1.6.3 Plates and Wells	16
1.6.4 Liquid Policies	17
1.6.5 Tip Policies	17
1.6.6 Abstraction of Pre-Condition (Checklist)	18
1.6.7 Abstraction of Liquid Handling Tasks.....	18
1.6.8 Abstraction of an entire protocol (kit implementation in a single command)	24
1.6.9 Abstraction of Robotic Arm tasks.....	24
1.6.10 Integration of external equipment (PCR).....	25
1.6.11 Run-Time logging.....	25
1.6.12 Examples of <i>Robolab</i> Usage.....	26
2 CREATING AND EXECUTING YOUR OWN PROGRAM	27
2.1 Create the program.....	27
2.2 Compile the program	27
2.3 Execute the program	28
3 DEFINITIONS AND DECLARATIONS	29
3.1 Global Definitions.....	29
3.1.1 DOC/ENDDOC	29
3.1.2 TABLE.....	29
3.2 Reagent Definition.....	29
3.2.1 REAGENT	29
3.2.2 LOAD	30
3.2.3 REAGENT_LIST	30
3.2.4 MIXDEF	30
3.3 Plate and Wells Definition	30
3.3.1 PLATE	30

3.3.2	WELL_LIST	31
3.4	Variable and List Definitions	31
3.4.1	LIST/ENDLIST	31
3.4.2	MATRIX_LIST name rowlist collist delimiter	32
3.4.3	Variables	32
3.4.4	OPTION	32
4	SCRIPT COMMANDS	33
4.1	General Script Flags	33
4.1.1	SCRIPT\ENDSCRIPT flags	33
4.2	Liquid Handling Commands	33
4.2.1	Option String	33
4.2.2	Description of Liquid Classes	34
4.2.3	Adding new Liquid Classes	34
4.2.4	TRANSFER_WELLS	34
4.2.5	DIST_REAGENT	34
4.2.6	DIST_WELL	35
4.2.7	TRANSFER_SAMPLES	36
4.2.8	MIX_WELLS	36
4.2.9	PREPARE_MIX	36
4.2.10	SERIAL_DILUTION	36
4.2.11	PREPARE_LIST	37
4.2.12	PREPARE_MATRIX	37
4.2.13	EQUAL_CONC	37
4.2.14	ELUTE_SAMPLE	38
4.3	Generic Program Control	38
4.3.1	PROMPT	38
4.3.2	WAIT	38
4.3.3	START_TIMER	38
4.3.4	WAIT_TIMER	39
4.4	Commercial Kits Implementation	39
4.4.1	SEQ_PURE	39
4.4.2	PCR_PURE	39
4.4.3	MP_MINIPREP	39
4.4.4	GET_TIPS	39
4.4.5	DROP_TIPS	40
4.5	Integrated Devices Commands	40
4.5.1	PCR_RUN	40
4.5.2	PCR_RUN_OPEN	40
4.5.3	PCR_OPEN	40
4.5.4	PCR_CLOSE	40
4.5.5	PCR_COVER	40
4.5.6	PCR_UNCOVER	41
4.5.7	SCHEDULE_PCR	41
4.5.8	VACUUM	41
4.6	ROMA related commands	41

4.6.1	MOVE_OBJECT	41
4.6.2	MOVE_PLATE	42
4.7	Logging and Export commands	42
4.7.1	LOGLIST	42
4.7.2	LOGMIX.....	42
4.7.3	LOGPLATE.....	42
4.7.4	CEPLATE.....	43
4.7.5	SEQPLATE.....	43
5	<i>ROUTINES AND MAINTENANCE</i>	44
5.1	Daily routine	44
5.2	Weekly routine.....	44

1 *Robolab* – a language for laboratory robot programming

1.1 *About Robolab*

The *Robolab* is a high level language for laboratory robots that was developed as part of my M.Sc. thesis. The goal of this document is to serve the users of the language. The first part is taken from the thesis and introduces the language and its capabilities using some examples. The other parts describe the language commands and their usage and may be used as a quick reference to the language..

1.2 *Introduction to Tecan Genesis Laboratory Robot*

The Laboratory Robot we use is a modular and programmable open platform. It consists of a modular table space and two robotic arms (see Figure 1). The table can be equipped with various carriers and racks for tubes, microplates, tips and reagents, as well as external integrated equipment such as a PCR machine, vacuum manifold, plate readers, etc. One of the robot arms is the liquid handling arm (LIHA see Figure 1b) which consists of 8 disposable-tip pipettes. Each pipette is connected to a different syringe and is thus capable of handling different volumes simultaneously. The LIHA can detect the liquid level in each tube automatically using the robot's disposable tips and can set each pipette to a different height accordingly. A second robot arm is the Robotic Manipulation Arm (ROMA see Figure 1d) which can handle square shaped objects such as microplates, and can load or unload them onto the robot's integrated devices. The robot is controlled by a personal computer (PC) using a software program called Gemini (developed by Tecan Group Ltd.). This program enables the user to run robot scripts called GEM files. The GEM files are in fact files written in the robot assembly language that includes information regarding the robot table organization as well as the script flow. Gemini also supplies the user with a graphic user interface environment for the development of GEM file scripts. However, development of new scripts is still a time-consuming task even

when prepared by robot experts. Another role of Gemini is to maintain the system definitions. The definitions include the properties of standard carriers and racks for tubes and microplates and liquid handling policies for various types of liquids and tips. This set of definitions can be extended by an advanced user to integrate new equipment to the robot system.

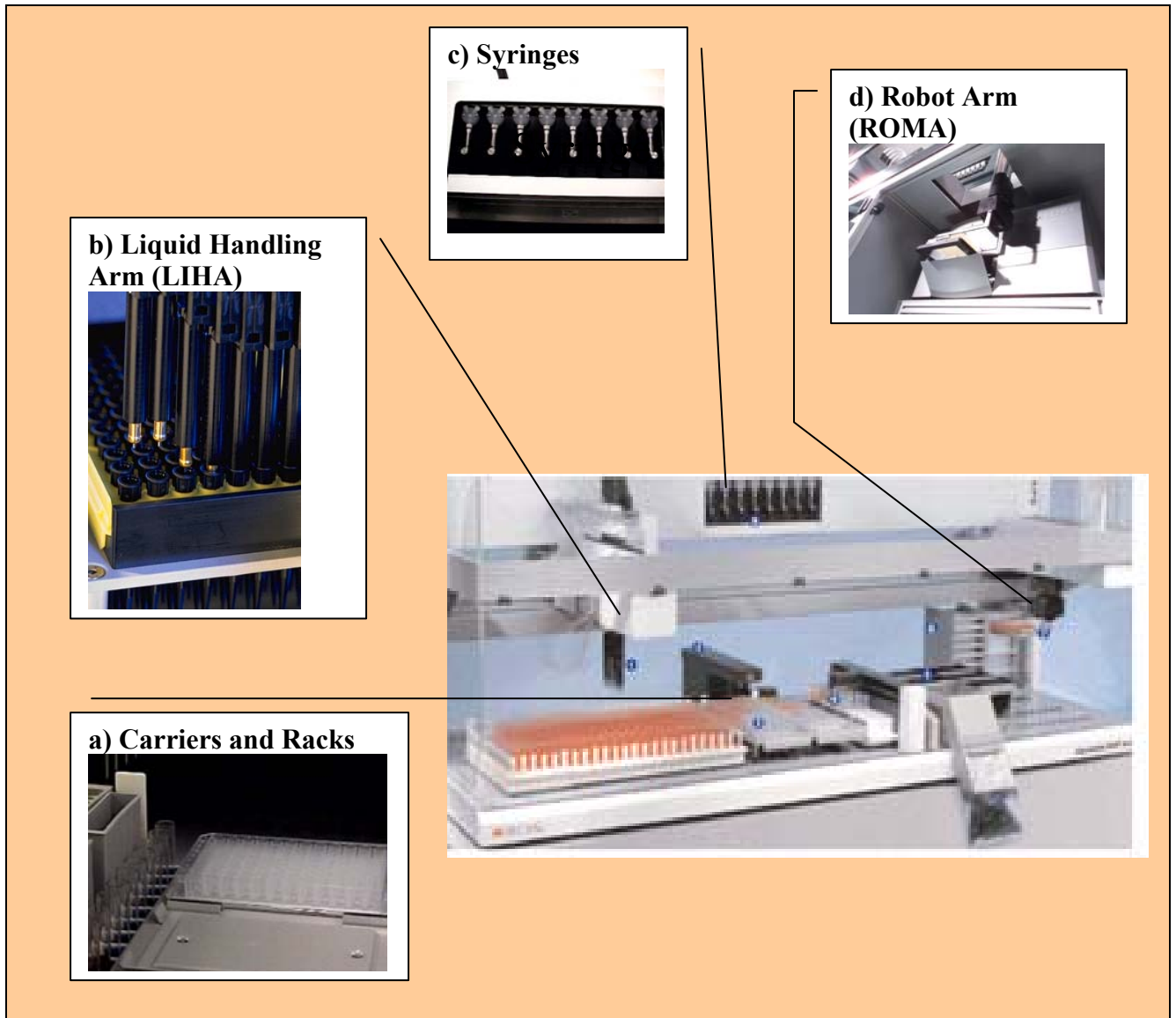


Figure 1 : Tecan Genesis laboratory robot. a) Carriers and Racks for various types of plates and tubes are placed on the robot table according to the needs. b) Liquid handling arm (LIHA) uses disposable tips with auto-detection capacity. c) 8 syringes enable simultaneous pipetting of different volumes with the LIHA. d) the Robot manipulating arm transfer plates and loads integrated devices.

1.3 Description of the Development Environment

The *Robolab* environment consists of several processing units, the compiler, the Gemini Software and the Robot. Figure 2 provides a schematic view of the environment components and data flow. A *Robolab* experiment program is written as a text file and is compiled using a compiler called *robocom.pl* (The compiler is implemented in Perl). The compiler compiles the *Robolab* experiment program into a GEM file. The GEM file, which is written in the robot assembly language can be executed by the Gemini software, and performs the experiment automatically. The compiler also validates correctness of the program and reports when errors are detected. For this purpose it uses a system configuration file that contains description of the table organization and other pre-defined system information. This enables the compiler to validate that the program may run on the current configured system. Besides the GEM file, the compiler produces a checklist which is used during the actual run-time of the compiled script and is aimed to validate the precondition of the system. In addition, the compiler produces a log file which documents the experiment. This log file is further updated during run-time logging events as they are happening. The *Robolab* program file and all other files produced by the compiler are located in the user experiment directory which helps the user to track their many experiments more easily. The system global configuration files and the code are located in a global area and thus make the maintenance of the system and its environment easier.

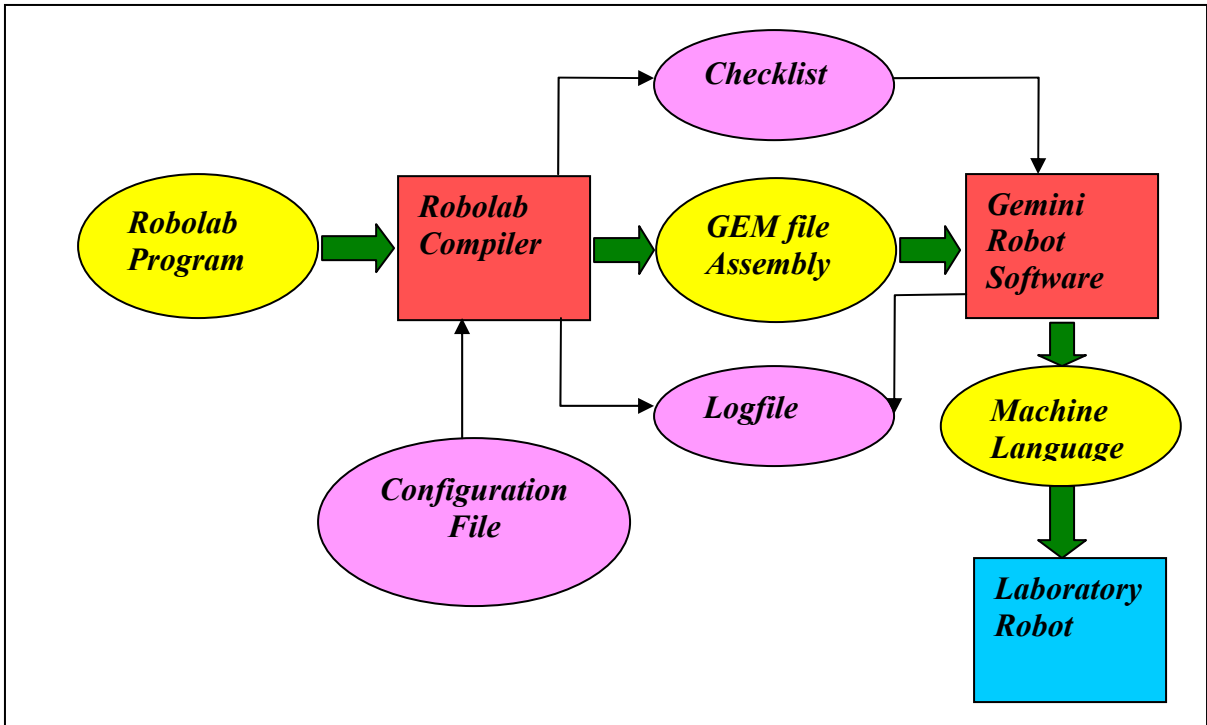


Figure 2 : a schematic flow diagram of the *Robolab* environment components. The red squares mark the software processing units, the blue square mark the robot, and the ellipses are the data objects. The green arrows and yellow ellipses describe the main data flow path in the system. A *Robolab* high-level program is compiled to a GEM file assembly language which is then translated to a machine language and executed on the robot. The purple ellipse describes the additional data involved in the system flow. See the main text for details.

1.4 Robolab Program Structure

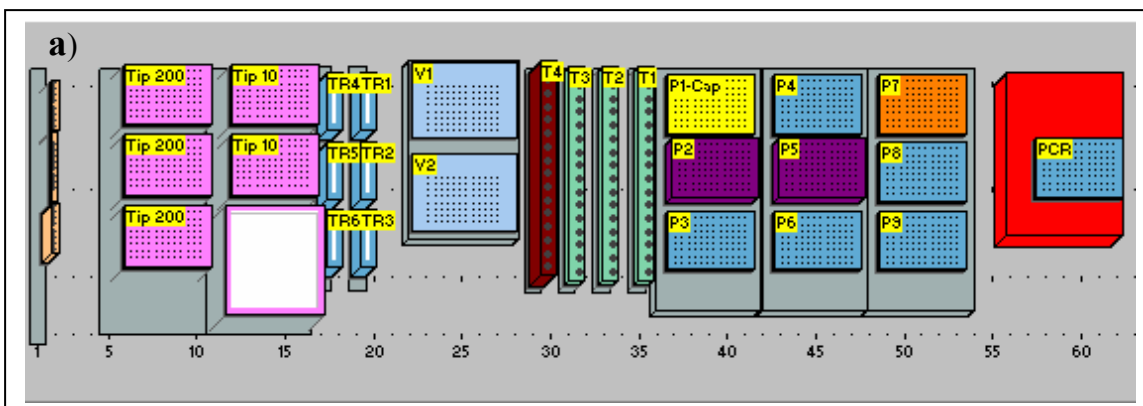
The *Robolab* experiment program is written as an ASCII text file. This text file has two distinct parts (See Figure 3b). The first part contains definitions and declarations, and starts at the beginning of the file. In this part the user can declare and define the various entities he intends to use in the current experiment such as reagents, variables, lists, mixes, etc. The second part of the program is the script segment. This part appears between two defining lines; a line with the word SCRIPT, and a line with the word ENDSRIPT (See Figure 3b). This segment contains the robot script commands that can make use of the definitions which appear in the definitions segment. For example, it may contain commands to distribute a defined reagent to a list of wells or prepare a certain liquid mix according to its definition. The compiler ignores empty lines and lines that start with “#” which may be used for comments. Other lines should start with one of the reserved words which are the definitions and commands of the program. Some of the reserved words such as DOC open a segment that ends with another reserved word such as ENDDOC. The lines within the segment are handled according to the declared segment. For example, the text between DOC and ENDDOC is treated as a free text that is being logged in the experiment log file and describes the experiment goals (See Figure 3b).

1.5 A simple Robolab program example

In Figure 3b there is a simple *Robolab* program that distribute water to several destinations on the robot table. The first section in blue is a documentation section that describes the purpose of the program in a free text that will be attached to the program log-file. The water reagent definition is in green. It declares a new reagent named WATER that is located at tube holder T1 at the first tube position (see table description Figure 3a). It also associates this new reagent with a liquid-policy (LCWAUTOAIR) that describes the default way this reagent should be handled. In this case the water level

should be auto-detected by the robot during aspiration from the source well and dispensed from the air at the target wells (for more details about liquid-policy see chapter 1.6.4).

The program instructions are in the script section of the program and are marked in red. The first instruction transfers 20 μ l of water to another tube located in tube holder T2 fifth position using the water default liquid-policy. The instruction orders the robot to take a new tip, aspirate 20 μ l of water from its source tube as defined in the beginning of the code, dispense 20 μ l of water to the target tube and finally drop the tip to the tip waste. The second instruction is similar to the first but with a different target. This time the target is the first column (8 wells) of plate P1. The robot is ordered to take 8 tips, and then aspirate water to each of the tips from the source tube. Since the robot cannot enter with 8 tips to a single tube the aspiration step consists of 8 different aspirations, each with a different tip. The target wells however are 8 sequential different wells and therefore the robot can dispense the water simultaneously to all wells. The decision when it is needed to access one tip at a time and when the tips can go simultaneously is automatically taken by the compiler during compilation time. Finally the robot drops the 8 tips to the waste. The last instruction fills an entire plate located at P2. The robot repeats the entire sequence of operations from the previous instruction for each column of the plate, replacing the tips between each column. The three examples of the instructions are nearly identical at the *Robolab* level however the generated assembly code is significantly different in its size. Furthermore, if the user would like to distribute the water more efficiently, using 8 water tubes that enable simultaneous water aspiration of 8 tips it may only require a short change in the definition line (see chapter 1.6.1 for details) and does not affect the program code at all.



b)

```

# Example of simple water distribution
DOC
This program distribute water from a tube to
  1) another tube
  2) to a single column in plate P1
  3) to entire plate P2

ENDDOC

# Define the water reagent
REAGENT WATER T1 1 LCWAUTOAIR

# The Script
SCRIPT
# Distribute 20 ul of water to first tube of tube holder T2
DIST_REAGENT WATER T2 5 20 DEFAULT
# Distribute 20 ul of water to first column of plate P1
DIST_REAGENT WATER P1 A1+8 20 DEFAULT
# Distribute 20 ul of water to entire plate P2
DIST_REAGENT WATER P2 A1+96 20 DEFAULT
ENDSCRIPT

```

Figure 3 : a) a robot table diagram describing the different entities on the robot table and their equivalent label in the *Robolab* language. The various colors of plates refers to their various types such as deep-well plate, vacuum manifold tube-holder etc. b) A simple *Robolab* program that distributes water from a source tube to several locations on the table. Comments marked in blue, water reagent definition in green and the script with the distribution instructions is in red. (See main text for details).

1.6 Abstraction of Robot Entities

1.6.1 Reagents & Mixes

An important role of a High-Level Robot Programming Language is the abstraction of the real-world entities such as solutions and reagents. As a lab worker plans an experiment, he uses various reagents and solutions to compose his reactions. Therefore, the language should allow the user to use a reagent or solution name for various tasks such as dilutions and reaction preparation. This requires that reagent properties required by the robot such as its location on the table will be predefined. In *Robolab* the user uses various definition commands to associate a reagent name with a specific location on the robot table (See Figure 4). In addition the user defines the default liquid policy (See chapter 1.6.4 for details) used to handle this reagent. The user may also define that a reagent is located in several tubes or wells and may be accessed by the LIHA simultaneously with several consecutive tips (see the definition of double distilled water (DDW) in Figure 4). The language enables the programmer to define a plate of reagents using a single command by loading an excel “plate file” that describes the plate content and maps the entire plate to a specific location on the table and a specific liquid policy (see LineagePrimers definition in Figure 4). Such a definition of all the properties of a specific reagent allows the user to use the reagent name in various high-level commands with no need to specify its location and how it should be handled. In addition, if the user chooses to change the reagent location or number of tubes he can do so in a single part of the program code without affecting the rest of the code. In some experiments, there is a need to create a master mix of reagents to be used in many reactions. The definition section also allows describing how to make a single quota of such a mix from other defined reagents. This definition can be used in the script section to prepare a required amount of the mix for many reactions (see PCRMIX definition in Figure 4). As it is sometimes the case that a specific reagent we use cannot be placed open on the robot table due to technical or economical reasons, I enabled the use of “virtual reagents”. Those reagents are defined as “external” and when the robot is asked to use them with a specific volume it halts the program and notifies the user to add the reagent at the

required volume to the specific tube. Once the user has completed the addition he confirms the completion and the robot continues with the program execution (See dNTP definition in Figure 4).

```

# Example of Reagent and Mix definition code

REAGENT PCRMIX T1 1 LCWAUTOBOT

REAGENT DDW T4 1 LCWAUTOAIR 4
REAGENT MgBuf T1 2 LCWAUTOBOT 1
REAGENT HotstartTAQ EXTERNAL 1 LCWAUTOBOT 1
REAGENT dNTP EXTERNAL 1 LCWAUTOBOT 1
LOAD LineagePrimers P2 LCWAUTOBOT
#Define how to prepare a single reaction of PCRMIX from other reagents
MIXDEF PCRMIX DDW 13.35 MgBuf 8 dNTP 0.4 HotstartTAQ 0.25

# below is the script section

SCRIPT
#prepare PCRMIX using the PCRMIX definition for 32 reactions
PREPARE_MIX PCRMIX 32
# Distribute 22 ul of PCRMIX reagent to plate position P3
# wells 1 to 32 using the PCRMIX default liquid policy
# then mix 3 times using 10 ul
DIST_REAGENT PCRMIX P3 A1+32 22 DEFAULT MIX:3x10
# Distribute 5 ul from plate P5 wells 1-32 to plate P3 wells 1-32
# Using liquid policy LCWBOT then mix 3 times using 10 ul
TRANSFER_WELLS P2 A1+32 P3 A1+32 5 LCWBOT MIX:3x5
ENDSCRIPT

```

Figure 4 : Example of simple *Robolab* program. The definition section lines are marked with green and the script section is marked with red. The parameters of the PCRMIX definition are annotated. Comments lines are in blue. The script prepares PCRMIX for 32 reactions and distributes the mix to a PCR plate. Then it transfers primers from the primers plate to the PCR reaction plate.

1.6.2 Table Locations

Locations on the Robot table are usually given in the Gemini software language by a Grid/Site pair which refers to coordinates on the Robot table. In the *Robolab* language each place on the table is labeled with a string. The user may give a meaningful label name such as PLATE1, PCR, VACUUM1, etc. In addition, the user may rename the pre-configured labels with labels of his choice for a specific experiment. This allows the user to use a label “PCR_PLATE” rather than “PLATE1” which makes the code language more user-friendly. Figure 3a show a diagram of a typical robot table arrangement with the labels of the various positions on the table.

1.6.3 Plates and Wells

There are many lab experiments that are done on a number of wells which is smaller than the entire 96 well plate. This brought a need to enable specification of ordered lists of wells in a simple way. For this I created a well-list string format that can easily describe an ordered well list in a simple fashion. In general, the robot enumerates a plate’s wells from 1 to 96 column by column, such that the first column is enumerated from 1 to 8 and the last column 89 to 96. The format I have created enables the user to refer to a well either by its number (from 1 to 96) or by its letter/number (row, column) as it appears on the microplate. The format allows the user to refer to a list of consecutive wells by either specifying the first well and the number of wells on the list (e.g. A1+16) or by using from-to format (e.g. A1-H2). The user can also concatenate several well lists using a comma. Another format available is the matrix format where the user specifies the upper left corner well of a matrix of wells on the plate, followed by the number of rows and the number of columns in the matrix. (Example: B2:6:10 is a matrix of wells that are not on the border of the microplate, starting from B2 as the matrix’s top left corner, and including 6 rows and 10 columns). A user can name a well list string using the word WELL_LIST in the definition section and can then use this name later on in places where a well list string is expected. For more details on this format see **Error! Reference source not found.** appendix 1 “The *Robolab* user manual”.

1.6.4 Liquid Policies

Liquid policies define how a liquid would be handled during aspirate and dispense actions. This includes a large set of definitions regarding the speed and position of the tips as they aspirate or dispense liquid. The user may state the position of the tips for aspiration and dispense actions relative to 3 reference points: the bottom of the well, the liquid level (using auto-detection option) and the top of the well (from the air). The user may control how fast the liquid is drawn into the tip and delay times after aspiration. It is also possible to control the tip retraction speed of the tip from the liquid. This is necessary to avoid mishandling of liquid with various viscosities. The *Robolab* includes a pre-definition of liquid policies that are commonly used. Their name reflects whether to aspirate with auto-detection or from the bottom and whether to dispense from the air near the bottom or near the liquid level. For example, LCWAUTOBOT is a liquid class for water-based solutions that should be aspirated using auto detection and dispensed near the bottom of the well. The pre-defined liquid-policies usually cover most of the needs of an average user, however if a new policy is required it may be defined in the system configuration.

1.6.5 Tip Policies

Tip policies define how the disposable tips are handled for a specific command. The robot liquid-handling arm has eight pipettes, however a command may include the handling of a much longer list of wells or even an entire plate. By default, the policy is to take new tips and drop the used tips for each well, to avoid contaminations. However, in many cases it is possible to reuse the tips. For example, in a case where a reagent is distributed to an empty plate or dispensed from the air it may be reused over the entire plate. The language includes a policy of KEEPTIP that may order a specific command to reuse the disposable tips for the entire command. Alternatively, a sequence of several commands can be done with the same tips. In such a case the user would like to take the

tips with a designated command and then perform the sequence of commands without taking or dropping any tip. For this purpose, a user may select the NOTIP policy that tells the command to skip any tip handling action. These features allow more economical and efficient use of the robot in carrying out tasks.

1.6.6 Abstraction of Pre-Condition (Checklist)

A user that programs automated experiments assumes that the correct reagents are placed with sufficient volume in the correct tube and in the correct place on the table. This raised a challenge of matching between what assumed by the program to be on the robot table and the actual state of the robot table and reagents. The *Robolab* meets this challenge with a mechanism of a checklist. The checklist file is produced by the compiler during compilation time and is presented to the user automatically at the beginning of the run-time execution. The user is asked during run-time to go over the checklist and commit that the items on the list are fulfilled. The items include a list of volume consumed from each reagent and a list of plates and tubes that are used by the program. In addition, the number of tips consumed of each tip type is specified. This mechanism has significantly reduced the common problems we faced when started to work with the robot which included misplacement of tubes and plates and colliding of tips with a covered tube that the user forgot to open prior to the robot run.

1.6.7 Abstraction of Liquid Handling Tasks

As a high-level programming language the *Robolab* should supply the user with high-level commands. Each command should enable the user to perform a complex task. Yet, the commands need to be modular and generic enough to enable their efficient reuse in the code. In this chapter I describe several examples of high-level commands that are included in the language and demonstrate their use.

1.6.7.1 Basic Liquid Handling Commands

There are two most common liquid handling basic commands: `DIST_REAGENT` that distributes a reagent to a list of wells, and `TRANSFER_WELLS` which can transfer liquid from a source list of wells to a destination list of wells (See Figure 4). These commands are responsible for a complete pipetting action including taking and dropping tips, aspirating the liquid, dispensing it according to a liquid policy, and mixing after the dispense if necessary. The user may specify which liquid policy will be used, what would be the tip type and how to conduct the mixing. In addition, he may specify a tip policy such as reusing the same tips or replacing tips for each well. Both commands enable the user to specify a single volume to all wells or a list of different volume for each well.

1.6.7.2 Handling a reaction list

The user can define a reaction list in a very convenient format. Each reaction is defined in a single line as a list of reagent name and its volume in the current reaction (see Figure 5). The command `PREPARE_LIST` processes the list and delivers each reaction content to a target well while keeping the order of reagents per each reaction. This is done very efficiently by handling up to 8 reactions simultaneously with the LIHA. The main advantage here is that the user plans a large set of reactions with various reagents and volumes as if it is planned in his notebook. The command produces the entire resulting set of actions to efficiently carry out these reactions. This significantly reduces human error when pipetting many reactions with many variable parameters. This feature was used in our lab to produce many variations of assembly PCRs, asymmetric PCR and other DNA enzymatic reactions. For example, the users were able to easily scan many different combinations of primer quantities, and to calibrate asymmetric PCR reactions.

```
LIST R1

GFP1_2f 3 GFP2_2f 3 GFP1F_5p 5 GFP2R_5p 5
GFP3_2f 3 GFP4_2f 3 GFP3F_5p 5 GFP4R_5p 5
GFP5_2f 3 GFP6_2f 3 GFP5F_5p 5 GFP6R_5p 5
GFP7_2f 3 GFP8_2f 3 GFP7F_5p 5 GFP8R_5p 5
GFP9_2f 3 GFP10_2f 3 GFP9F_5p 5 GFP10R_5p 5
GFP11_2f 3 GFP12_2f 3 GFP11F_5p 5 GFP12R_5p 5
GFP13_2f 3 GFP14_2f 3 GFP13F_5p 5 GFP14R_5p 5
GFP15_2f 3 GFP16_2f 3 GFP15F_5p 5 GFP16R_5p 5

ENDLIST

SCRIPT
PREPARE_LIST R1 P3 A1+8 LCWAUTOBOT
ENDSCRIPT
```

Figure 5 : A reaction list for assembly PCR of GFP molecule marked with blue. Each line holds the reagents and volumes of a specific reaction that assembles part of the molecule. The list name R1 is then given as a parameter to the PREPARE_LIST command marked in red. The command enable flexible reaction plan with various reagent number and various volumes per reaction which is carried out automatically with no errors.

1.6.7.3 Matrix of reactions

Often, an experiment requires producing a matrix of reactions. For example amplification of a set of few samples by several primer pairs is rather common in our lab for cell lineage analysis. The PREPARE_MATRIX command offers the programmer a single command to produce such a matrix. The programmer supplies a list of row reagents and a list of column reagents and the command perform the entire liquid handling process needed to produce this matrix efficiently (See Figure 6). This feature was used in our lab to produce many PCR reactions for the “Cell Lineage” project (See chapter **Error! Reference source not found.**) where a set of primers for microsattelites is used to amplify a set of DNA samples from various cell clones.

```

REAGENT ABGene4X T3 1 LCWAUTOBOT 1
REAGENT DDW T3 2 LCWAUTOBOT 1
#define list of DNA
LIST DNA
DNA50_0
DNA25_0
DNA12_5
DNA6_25
DNA3_125
DNA1_562
NC
ENDLIST
REAGENT_LIST DNA T1 1+7 LCWAUTOBOT
#define list of Primers
LIST Primers
mmg1
mmg2
mmg3
mmg4
ENDLIST
REAGENT_LIST Primers T3 4+4 LCWAUTOBOT

SCRIPT
# dilute the DNA by a factor of 2 five times
SERIAL_DILUTION DDW T2 1 T2 2+5 50 50 LCWBOT
DIST_REAGENT DDW T1 1+7 28.75 DEFAULT
DIST_REAGENT ABGene4X T1 1+7 31.25 DEFAULT MIX:7x40
TRANSFER_WELLS T2 1+7 T1 1+7 25 LCWBOT MIX:7x40

# Create a matrix of PCR reactions of all DNA concentration on
#all primers groups

PREPARE_MATRIX DNA Primers P3 A1 17 8 LCWBOT 1 MIX:3x7

#move the plate to PCR and execute pcr program
MOVE_PLATE P3 PCR
RUN_PCR 9 1
ENDSCRIPT

```

Figure 6 : The *Robolab* program above is for calibration of required DNA template concentrations for multiplex PCR amplification of 4 primers groups. The DNA is diluted using serial dilution (purple) to generate a DNA gradient of concentration. Then, a PCR mix with the diluted DNA is prepared. The “prepare matrix” command (red) generates a matrix of reactions of all DNA concentrations (blue) by all primer groups (green). Finally, the plate is loaded to PCR and Amplification program is operated (pink).

1.6.7.4 Serial Dilution

Many experiments require a concentration gradient of one or more reagents. The SERIAL_DILUTION commands enable conduction of serial dilution for a list of wells using any diluter selected by the user for this purpose in a simple manner (See Figure 6). Furthermore, it is sometimes necessary to optimize the concentration of two or more reagents that interact in a specific reaction. For this purpose there is a need to generate a matrix of reactions with two orthogonal gradients. Such a multi dimensional gradient may be produced by a very short *Robolab* code (See Figure 7). This efficient parameter scan would be labor intensive if done manually. The use of automated program for this purpose may help to scan the parameter space at a better resolution and avoid in accurate estimations of optimal concentrations.

```

REAGENT DDW TR3 1 LCWAUTOAIR 8
REAGENT Red T1 1 LCWAUTOBOT 1
REAGENT Blue T2 1 LCWAUTOBOT 1
REAGENT RedD T1 1 LCWAUTOBOT 8

SCRIPT
SERIAL_DILUTION DDW T1 1 T1 2+7 300 300 LCWAUTOBOT
DIST_REAGENT RedD P3 A1+96 20 LCWAUTOBOT TIP:KEEPTIP
DIST_REAGENT Blue P3 A1+8 20 LCWAUTOBOT MIX:10x30
SERIAL_DILUTION DDW P3 A1+8 P3 A2+88 0 20 LCWBOT MIX:7x15
ENDSCRIPT

```

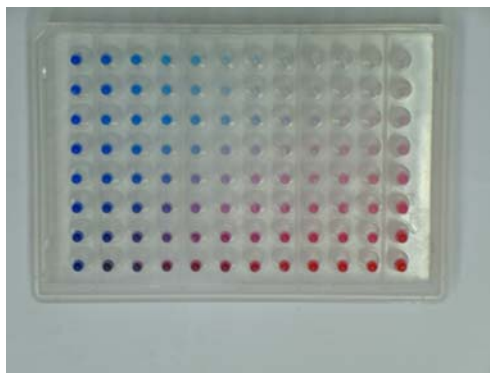


Figure 7 : The above *Robolab* code generates a matrix of two gradients. The red gradient is horizontal while the blue is vertical. The resulting plate is presented below. This enables one to scan parameters and optimize a reaction with two dependent reagents and find its optimal concentration combination.

1.6.7.5 Level of Concentration

It is a rather common need to have a set of samples in various concentration levels that need to be diluted to a given target concentration. A common example for this is the preparation of equal DNA concentration samples after purification. The command `EQUAL_CONC` receive a list of wells and their current concentration (as measured by a plate reader for example) and a diluter name and brings the set of wells to a given target concentration. The amount of required dilution for each well is calculated automatically by this command.

1.6.8 Abstraction of an entire protocol (kit implementation in a single command)

Lab routines are often implemented using commercial kits such as DNA purification kits or Miniprep kits (kits for plasmid preparation from bacteria). Today many of those kits are available in automation-friendly format. For example, the miniprep kit requires the use of five solutions for a timely controlled lysis process and several cycles of washing and filtration. The *Robolab* enable fully automated operation of those kits using a single command. The command gets as parameters the location of source samples and the target locations for the processed samples and performs the entire protocol for up to 96 samples, thus saving a large amount of manual lab work. *Robolab* currently implements DNA purification kits and sequencing reaction purification kits as well as Miniprep kits, all of which are routinely used in our lab.

1.6.9 Abstraction of Robotic Arm tasks

The robotic arm is originally operated by two types of command in the assembly language. Those commands operate a predefined vector of movement in the system. The vector handles the directions of the arm from a predefined safe position to end position. The assembly command allows the user to operate the vector on a specific carrier and site and also allows one to state the gripper status (open or close). A different vector is implemented for each carrier on the table. The *Robolab* language abstracts this to the simplistic commands `MOVE_PLATE` and `MOVE_OBJECT` (see example in Figure 6). Those commands are compiled to the set of assembly commands that control the correct vector and gripper status. First, access the source with the gripper open and taking the requested object (or plate) then move to target location with the gripper holding the object and releasing the gripper at the target position. Finally the move the ROMA arm back to its home position so it will not interfere with the LIHA arm as it moves above the table space. This abstraction significantly simplifies the operation of the robot ROMA arm and reduces the amount of coding and parameters necessary.

1.6.10 Integration of external equipment (PCR)

While some devices such as the vacuum manifold are already integrated to the robot language, other third party lab equipment such as a Robotic PCR thermo-cycler or Gel Capillary Electrophoresis machine require integration. The Robotic PCR machine is controlled with dedicated software from the same computer that runs the Gemini. Using the assembly language ability to execute other programs I implemented a set of commands to control the Robotic PCR machine. For example, the `SCHEDULE_PCR` command allows the programmer to schedule several plates for PCR. The robot loads the plate to the PCR machine, closes its lid, and runs a specific PCR program. When the program has completed, the lid is opened, the plate is unloaded from the PCR machine, and the next scheduled plate is loaded. This enabled overnight utilization of the PCR machine, and higher throughput of process sampling. Another example of integration is that of the Gel Capillary Electrophoresis (CE) machine (Applied Bio-Systems Avant-3100). In this case due to lack of appropriate hardware, complete integration was not possible. However, it was easy to use the *Robolab* language to generate a complete input file that accompanies a plate of samples prepared by the robot for a CE run or a sequencing run on the CE machine. This file saves a lot of work required to define a plate of samples on the CE machine (see `CEFILE` and `SEQFILE` commands in the *Robolab* user manual appendix.).

1.6.11 Run-Time logging

When working with automated environments the user must have run-time information regarding the completion and the status of the experiment. Some of the run-time error mechanisms such as problems in liquid detection are handled by the robot Gemini software. However, if a problem occurs during run-time, the user must be notified that his program has not completed successfully. The user may also want to know what steps were completed before the program halted. For this purpose I added to the language runtime event log capabilities. As the program progresses in run-time, it reports back to the *Robolab* logfile (created during compilation) and writes significant events with their

time of occurrence. This feature increases the possible walk-away time of the operator of the robot by assuring that important events during the run-time will not be ignored

1.6.12 Examples of *Robolab* Usage

To demonstrate the usability of the *Robolab* language, listed below are examples of different uses of the language as it was used by its users until now.

- DCPIE project – with the main goal of developing a protocol for DNA synthesis (see chapter **Error! Reference source not found.**) many of the below tasks were programmed separately and combined and conducted automatically by the robot.
 - PCR amplifications- including preparation of Real-Time PCR
 - Assembly PCR
 - Asymmetric PCR
 - PCR purification
 - Gel Sample Preparations.
 - Capillary Electrophoresis Sample preparation.
 - Calibration of various Enzymatic Reactions (Example: Lambda Exonuclease)
 - Sequencing-including sequencing reaction purification and sample preparation for sequencing machine.
 - Plasmid Purification
 - Stock preparation and dilution

- Cell Lineage project –the below tasks were programmed as part of the development of the Cell Lineage Reconstruction method (See chapter **Error! Reference source not found.**).
 - High-Throughput PCR amplification-matrix amplification in samples by n primer pairs.
 - Scheduling of PCR plates for night runs
 - Mouse Genotyping

- Pedigree analysis
 - Multiplex PCR – (Including DNA gradient calibrations)
 - Capillary Electrophoresis Sample preparation.
 - Stock preparation and maintenance
- Bimolecular Computer project –the below tasks were programmed as part of the calibrations of a finite automaton and the involved enzymatic reaction.
 - Calibration of FOK enzymatic reaction under temperature control.
 - Scanning of performance of various molecular-computer designs.
 - Capillary Electrophoresis Sample preparation.

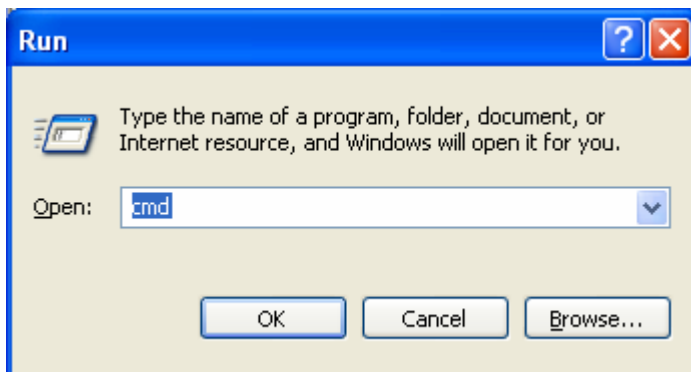
2 Creating and executing your own program

2.1 Create the program

- In your user workarea create a new folder for a new experiment
- Open a new text file (using a text editor) and name it for example **myExperiment.conf**
- Document your experiment using DOC/ENDDOC
- Define your reagents
- Write your script (between SCRIPT\ENDSCRIPT)

2.2 Compile the program

- Open a command window using Start→Run (type **cmd** in the run box as below)



- Change directory to your experiment:
 - Type: **F:**
 - Type: **CD F:\username\myExpDir**
- Compile your program using robocomp.pl (See below)

```

C:\> Command Prompt
C:\Documents and Settings\shaik.MATH233-PC>f:
F:\>cd shaik\test
F:\Shaik\test>robocomp.pl test.conf
CONF: test.conf
Compilation completed....
Robot gem file: F:/Shaik/test/test.gem

F:\Shaik\test>dir
Volume in drive F is IMAGE
Volume Serial Number is 064C-C86C

Directory of F:\Shaik\test

07/25/2005  05:32 PM    <DIR>          .
07/25/2005  05:32 PM    <DIR>          ..
07/26/2005  06:31 PM                115 test.conf
10/10/2005  11:24 AM            3,418 test.gem
10/10/2005  11:24 AM             210 test.log
10/10/2005  11:24 AM             672 test.checklist
               4 File(s)              4,415 bytes
               2 Dir(s)          9,178,890,240 bytes free

F:\Shaik\test>_
  
```

- Handle errors if needed verify the creation of GEM file LOG file and CHECKLIST file (See above)

2.3 Execute the program

- Open the Gemini software and login
- From the menu choose File→Open
- Browse and select your GEM file
- Verify that it does what you planned
- Organize the Table and materials as you planned

- Run GEM file
- When the checklist appear verify all its bullets and confirm.
- Your automated script is now running.

3 Definitions and Declarations

3.1 Global Definitions

3.1.1 DOC/ENDDOC

The tags are used to dedicate a section for free text documentation of the experiment. This documentation will also go to the logfile of the experiment.

Example:

```
DOC
Experiment of Mouse 1
On primers 1 2 3 4
ENDDOC
```

3.1.2 TABLE

Select a different table file than the default. Program assumes that the carriers are placed the same and only rack are changed.

Example:

```
TABLE table7.gem
```

3.2 Reagent Definition

3.2.1 REAGENT

Define a single reagent that can be later used for pipeting commands. A user may also define Reagent as "EXTERNAL" by putting EXTERNAL in the rack field. In this case the Robot will prompt the user to add the reagent at the right volume to right tube when requested in PREPARE_LIST or PREPARE_MIX commands.

Usage:

```
REAGENT reagent_name rack first_well liquidclass [ number of tubes; default is 1]
```

Example:

```
REAGENT ddw T1 1 LCWAUTOBOT 2
REAGENT Enzyme EXTERNAL 1 LCWAUTOBOT
```

3.2.2 LOAD

Description:

Define many reagents at once by loading their names and location from a plate file (excel file) They can be later used for pipeting commands.

Usage:

LOAD plate_name rack liquidclass

Example:

LOAD gfp_primers P5 LCWAUTOAIR

3.2.3 REAGENT_LIST

Description:

Usage:

Defines reagent list on a well list with single command.

Example:

REAGENT_LIST reagent_list rack well_list liquidclass

3.2.4 MIXDEF

Description:

First the user have to define a new reagent with same name using the REAGENT command. Then the MIXDEF command define how to prepare a single shot of the mix from other reagents. During application a user may ask to prepare the mix for any number of reactions. The command will generate pipeting command for making the mix at the required amount. Alternatively the user may get a log of how to prepare the mix for a required amount of reactions.

MIXDEF name reagent1 volume1 reagent2 volume2 reagent3 volume3...

Example:

MIXDEF pcrmix DDW 32.35 Buffer10X_MgCl 8 dNTP 0.4 Enzime 0.25

3.3 Plate and Wells Definition

3.3.1 PLATE

Description:

Define a plate name and assign its table position (rack) and its first well. This can be used to later allocate wells according to the number of samples to process starting at the first available well.

Usage:

PLATE plate_name rack first_well

Example:

PLATE pcr P3 A7
PLATE template T1 9

3.3.2 WELL_LIST

Description:

Define a list of wells using the well list string format. The list can later be used in the pipeting commands. Wells are considered enumerated vertically column after column. The first column is 1-8 the second column is 9-16 etc..

Usage:

WELL_LIST name well_list_str

Example:

WELL_LIST pcr_wells A1+8 – Starting with A1, 8 wells
WELL_LIST pcr_wells A1-H2 –From A1 to H2 (16 wells)
WELL_LIST pcr_wells 56+8 – Starting well 56, 8 wells
WELL_LIST pcr_wells 1-16 – From well 1 to well 16 (16 wells)
WELL_LIST pcr_wells 1,3,17,19,A8,H3 - list of single wells (6 wells)
WELL_LIST pcr_wells A2+8,A5+8,A10+8
WELL_LIST pcr_wells A1+8x4- 4 repeats of A1+8 equivalent to A1+8, A1+8, A1+8, A1+8
WELL_LIST pcr_wells B2:3:4 matrix of 3 row 4 columns corner B2 is upper left ordered one column after another
WELL_LIST pcr_wells A1+8|2 – Produces a list of 16 wells. Each well from A1 to H1 is duplicated so the list is: A1,A1,B1,B1,C1,C1...H1,H1

3.4 Variable and List Definitions

3.4.1 LIST/ENDLIST

Description:

Define a list of items (string numbers etc.) each line is an item. This can be used for reaction names, reaction compositions, volume lists etc. The list name may be later use in some logging and pipeting commands.

Usage:

LIST list_name
Item1
Item2
.
.
.
ENDLIST

Example:

```
LIST volume_list1
2
3
4
5
4.5
ENDLIST
```

3.4.2 MATRIX_LIST name rowlist collist delimiter

Defines a list named name. each item is column-delimiter-row. Columns are concatenated one after another.

Example:

```
MATRIX_LIST name rowlist collist delimiter
```

3.4.3 Variables

Description:

Any line that starts with un-reserved word following = sign is considered a variable. The value is the string/number that follows the = sign upto the end of the line. The value can later be used as volume or can be read into the application for further use.

```
variable_name = value
```

Example:

```
X = 4.5
```

```
NAME = pie1
```

3.4.4 OPTION

Description:

Used to set perl variable with a value. The perl variable is a global option that its default value changed using this command.

Example:

```
OPTION COVER_AUTOMATION_ENABLED 1
```

```
OPTION ROMA_SLOW_SPEED 1
```

4 Script Commands

4.1 General Script Flags

4.1.1 SCRIPT\ENDSCRIPT flags

The script commands should be placed within the line SCRIPT and the line ENDSCRIPT. For Example:

```
SCRIPT
Command1
Command2
Command3
.
.
.
ENDSCRIPT
```

4.2 Liquid Handling Commands

4.2.1 Option String

In some of the pipeting commands there is an option parameter that enable to override the default tip and mix behaviour of the system.

4.2.1.1 MIX option

By default there is no mix after each dispense action.

A MIX option may have the following formats:

- MIX:3x10 - This format means mix 3 times of 10 ul at the bottom of the well
- MIX:LCWMXAUTO:3x10 - This format means mix 3 times of 10 ul at the liquid level of the well

Other Mix policies may be added to the system and used as the second format.

4.2.1.2 TIP option

By default the tip policy is to take and drop tips after each pipeting action. However in some cases the user may want to act differently.

TIPMODE:KEEPTIP: For example, if distributing water or other mix to empty wells the user may want to reuse the mounted tips again and again.

TIPMODE:NOTIP: For example, sometimes the user wants to handle the tips by him self in order to combine several consequent pipeting command using same tips. In such a case

the user may handle tips externally using GET_TIPS and DROP_TIPS commands and execute all pipeting actions with no tip handling at all.
 TIPTTYPE:10 – for changing the tip type to 10 or 1000;

4.2.2 Description of Liquid Classes

LCWBOT –	Aspirate-bottom despense-bottom
LCWAIR –	Aspirate-bottom despense-air
LCWAUTO –	Aspirate-bottom despense-autodetect
LCWAUTOAIR –	Aspirate-autodetect despense-air
LCWAUTOBOT –	Aspirate-autodetect despense-bottom
LCWAUTOBOT_SLOW –	Aspirate-autodetect despense-bottom (for loading buffer)
MTGSEQ-	Aspirate -montageseq bottom Despense- bottom
LCWMX –	Mix near the bottom
LCWMXSLOW –	Mix near the bottom but gently
LCWMXAUTO-	Mix near liquid level

4.2.3 Adding new Liquid Classes

A user may define a new liquid class using Gemini to adjust its properties. Then in order to define it for Robolab usage it should be included in the file liquid_class.conf which is in the lib directory of the global area of the system.

4.2.4 TRANSFER_WELLS

Description: pipeting from many wells to many wells.

Usage:

TRANSFER_WELLS source_plate source_wells target_plate target_wells volume liquidclass [option_string]

Example:

```
TRANSFER_WELLS P3 A1+16 P5 A2+16 30 LCWBOT
TRANSFER_WELLS P3 A1+16 P5 A2+16 30 LCWBOT TIPMODE:KEEPTIP
TRANSFER_WELLS P3 A1+16 P5 A2+16 30 LCWBOT MIX:3x10
TRANSFER_WELLS P3 sw P5 dw 30 LCWBOT MIX:LCWMXAUTO:3x10
```

4.2.5 DIST_REAGENT

Description: distribute a reagent to many wells. If reagent is multitube (in decleration) then all tubes will be used sequencialy.

Usage:

DIST_REAGENT reagent_name target_plate target_wells liquid_class [option_string]

Example:

```
DIST_REAGENT DDW P3 A2+18,A7+6 12 LCWBOT
DIST_REAGENT DDW P3 A2+18,A7+6 12 LCWBOT TIPMODE:KEEPTIP
```

4.2.6 DIST_WELL

Description: distribute a single well to many wells.

Usage:

DIST_WELL source_plate source_well target_plate target_wells liquid_class [option_string]

Example:

DIST_WELL T4 1 MONTAGE A2+18,A7+6 WVOL LCWBOT TIPMODE:KEEPTIP

4.2.7 TRANSFER_SAMPLES

Description: In case where plates were defined with a first well and all samples are assigned in sequential well order a user can use the number of samples and plate definitions instead of the well list format as in TRANSFER_WELLS.

Usage:

TRANSFER_SAMPLES num_of_samples source_plate target_plate 15 liquid_class [option_string]

Example:

TRANSFER_SAMPLES 32 PCR MONTAGE 15 LCWBOT TIPMODE:KEEPTIP

4.2.8 MIX_WELLS

Description: Mix a list of wells

Usage:

MIX_WELLS plate_name wells times volume liquid_class [tip_option]

Example:

MIX_WELLS P3 1-17 3 10 LCWMX

MIX_WELLS P3 1-17 3 10 LCWMX NOTIP

4.2.9 PREPARE_MIX

Description: Prepare a mix of reagents from its ingredient reagents according to the MIXDEF definitions. A user may define a mix (for example for PCR) for single reaction then use prepare_mix to make the mix the required amount of reactions with some margin. If the mix is defined as reagent with several tubes the total amount will be divided equally among the tubes. Later on the user may use DIST_REAGENT to distribute the mix to wells.

Usage:

PREPARE_MIX mix_name number_of_shots [margin_percent]

Example:

PREPARE_MIX PCRMIX 10

PREPARE_MIX PCRMIX 10 0.1

4.2.10 SERIAL_DILUTION

Description: Dilute a list of sources serial dilution. First diluter is distributed at the required amount (diluter_volume) to all target wells. Then, each source is taken at the required amount (transfer_volume) and mixed at the first-step dilution well. The same transfer amount is taken again and mixed in the second step wells. The target wells must be a multiplication of the source wells and the function assume that the ratio of target wells to source wells is the number of dilution steps required. Putting zero as the diluter volume will skip the diluter distribution (in cases done diluter was distributed other way.)

Usage:

SERIAL_DILUTION diluter source_plate source_wells target_plate target_wells diluter_volume
transfer_volume [liquid_class mix_options]

Example:

```
SERIAL_DILUTION DDW P3 1 P3 2-8 40 10
SERIAL_DILUTION DDW P3 A1+8 P3 A2+32 40 10 LCWAUTOAIR
SERIAL_DILUTION DDW P3 A1+8 P3 A2+32 40 10 LCWAUTOAIR MIX:LXWMXAUTO:5x20
```

4.2.11 PREPARE_LIST

Description: Prepare a list of reactions. Each line describe one reaction well content using reagent and volume pairs. A reagent maybe specified in position format (example P3:A1).

Usage:

```
PREPARE_LIST listmix_name rack well_list liquid_class [mix_option]
```

Example:

Reaction list(in definition section):

```
LIST R1
gfp1 3 gfp2 4.5 ddw 7
gfp3 4 gfp4 5 ddw 2
P3:C2 6 gfp1 3 gfp2 4.5 ddw 4
ENDLIST
```

command:

```
PREPARE_LIST R1 P3 A1+3 LCWAUTOBOT
PREPARE_LIST R1 P3 A1+3 LCWAUTOBOT MIX:1x10
```

4.2.12 PREPARE_MATRIX

Description: Prepare a matrix of reactions. A list of reagents for rows and a list of reagents for columns are combined. Rows are distributed first enable an option of keptip. The mix option is only valid for the columns. The matrix location on the plate is defined by the corner well (upper left) and the sizes of the reagent lists of rows and columns. The parameter keptip_forrows = 1 means that keptip option will apply for rows.

Usage:

```
PREPARE_MATRIX rowlist collist rack corner_well rowvol colvol liquid_class keptip_for_rows  
[mix_option]
```

Example:

4.2.13 EQUAL_CONC

Description: Brings a plate with wells of various concentration to a target concentration. Useful for DNA concentration leveling after measure in a plate-reader.

Usage:

```
EQUAL_CONC diluter rack well_list current_vol conc_list target_conc [mix_option]
```

Example:

Concentration list(in definition section):

LIST R1

3

6.2

8.5

ENDLIST

command:

EQUAL_CONC DDW P1 A1+3 30 R1 1 MIX:2x30

4.2.14 ELUTE_SAMPLE

Description: Elute samples from the vacuum plate to a target well list.

Mix before and after the wait time default mix is 20 times of 0.8 of the volume.

Usage:

ELUTE_SAMPLE vac_plate vac_well_list target_plate target_well_list buffer volume wait_time [options]

Example:

ELUTE_SAMPLE V1 A1+8 P1 A1+8 DDW 30 45

4.3 Generic Program Control

4.3.1 PROMPT

Description: Open a userprompt windows with the text

Usage:

PROMPT text

Example:

PROMPT remove tube covers!!!!

4.3.2 WAIT

Description: time: time to wait in seconds

Usage:

WAIT time

Example:

WAIT 600

4.3.3 START_TIMER

Description: start timer

Usage:

START_TIMER timerid

Example:

START_TIMER 1

4.3.4 WAIT_TIMER

Description: wait for timer timerid to reach time.

Usage:

WAIT timerid time

Example:

WAIT_TIMER 1 600

4.4 Commercial Kits Implementation

4.4.1 SEQ_PURE

Description: purify sequencing reaction on the montage seq kit.

Usage:

SEQ_PURE sample_plate sample_well_list vac_plate vac_well_list target_plate target_well_list
injection_buffer

Example:

SEQ_PURE P3 A1+8 V1 A1+8 P1 A1+8 INJBUF

4.4.2 PCR_PURE

Description: purify pcr reaction on the montage seq kit. Reaction vol is completed with injbuf to 100 ul .

Usage:

PCR_PURE sample_plate sample_well_list vac_plate vac_well_list target_plate target_well_list
injection_buffer reaction_vol elution_vol

Example:

SEQ_PURE P3 A1+8 V1 A1+8 P1 A1+8 INJBUF

4.4.3 MP_MINIPREP

Description: miniprep according to milipore kit. Assuming SOL1-SOL5 are defined.

Usage:

MP_MINIPREP sample_plate sample_well_list vac_plate vac_well_list target_plate target_well_list

Example:

MP_MINIPREP P3 A1+8 V1 A1+8 P1 A1+8 INJBUF

4.4.4 GET_TIPS

Description: Take new tips

Usage:

GET_TIPS num_of_tips

Example:

GET_TIPS 8

4.4.5 DROP_TIPS

Description: drop all tips

Usage:

DROP_TIPS

Example:

DROP_TIPS

4.5 Integrated Devices Commands

4.5.1 PCR_RUN

Description: run a pcr program

Usage:

PCR_RUN pcr_dir pcr_prog

Example:

PCR_RUN 0 1

4.5.2 PCR_RUN_OPEN

Description: run a pcr program when the lid is open (for temp control)

Usage:

PCR_RUN_OPEN pcr_dir pcr_prog

Example:

PCR_RUN_OPEN 0 1

4.5.3 PCR_OPEN

Description: Open pcr lid

Usage:

PCR_OPEN

Example:

PCR_OPEN

4.5.4 PCR_CLOSE

Description: Open pcr lid

Usage:

PCR_CLOSE

Example:

PCR_CLOSE

4.5.5 PCR_COVER

Description: Put Cover on PCR

Usage:

PCR_COVER cover_rack

Example:

PCR_COVER HA9

4.5.6 PCR_UNCOVER

Description: remove cover from PCR

Usage:

PCR_UNCOVER cover_rack

Example:

PCR_UNCOVER HA9

4.5.7 SCHEDULE_PCR

Description: schedule plate for a pcr run. The robot will take the plate to the PCR assuming it is covered and will run the program. If the flag open_pcr = 1 once the program done the robot remove the plate back to the table same position. The program must not include a PAUSE in this case. This enable a scheduling of several PCR plates for a night run.

Usage:

PCR_SCHEDULE rack pcrdir pcrprog open_pcr

Example:

PCR_SCHEDULE P3 1 1 1

PCR_SCHEDULE P6 2 2 0

4.5.8 VACUUM

Description: Activates vaccum at a given amount of pressure for a given amount of time.

Usage:

VACUUM vacuum_num pressure time(seconds)

Example:

VACUUM 1 700 120

4.6 ROMA related commands

4.6.1 MOVE_OBJECT

Robot arm move the object from location to location

Usage:

MOVE_OBJECT object from to

Example:

```
MOVE_OBJECT PLATE P3 PCR
MOVE_OBJECT VBLOCK V1 V2
MOVE_OBJECT PLATE HA9 HB4
```

4.6.2 MOVE_PLATE

Same as move_object but the object is always plate here.

Usage:

```
MOVE_PLATE from to
```

Example:

```
MOVE_PLATE P3 PCR
MOVE_PLATE VBLOCK V1 V2
MOVE_PLATE HA9 HB4
```

4.7 Logging and Export commands

4.7.1 LOGLIST

Description:

Log in the program's logfile each item in the list (for example reaction name). Each reaction is assigned to a well from the well list. This way a user can document what reaction went to which well. The plate name should note the label of the plate.

Usage:

```
LOGLIST list_name well_list plate_name
```

4.7.2 LOGMIX

Description:

Log the mix's reagents and their volume per single reaction (as defined in MIXDEF) as well as the total of each reagent to the required number of shots. (Automatically add 0.08 margin of the total to each reagent). Logs exactly what PREPARE_MIX will do. This log can be used to manually prepare the required mix.

Usage:

```
LOGMIX mix_name num_of_reactions
```

4.7.3 LOGPLATE

Description:

Create an excel file in the plate folder that assign each item on the list to the relevant box in the excel. This file can later be loaded to define the reagents.

Usage:

```
LOGPLATE plate_name listname well_list
```

4.7.4 CEPLATE

Description:

Create a file that can be imported to the capillary electrophoresis machine for fragment analysis.

Usage:

CEPLATE plate_name list_name first_well result_group [inst-protocol]

4.7.5 SEQPLATE

Description:

Create a file that can be imported to the capillary electrophoresis machine for sequencing.

Usage:

SEQPLATE plate_name list_name first_well result_group [inst-protocol]

5 Routines and Maintenance

5.1 Daily routine

- Check enough system liquid exist.
- Flush instrument
- Replace tip waste bag

5.2 Weekly routine

- Change system liquid and waste
- Check syringe are sealed