Roboocyte2 Scripting

Table of Contents

1 Introduction

1.1 Java Script

Roboocyte2 scripts are written in a JavaScript like language. The syntax (control structures, variables, functions, arrays) is conforming to JavaScript, other JavaScript functionality, particularly the **document** (and other HTML specific things like DOM handling, libraries etc.) is **NOT** supported.

1.1.1 Roboocyte2 Specific Features

The Roboocyte2 is controlled via 4 JavaScript objects:

1. **Robo2** (the robot hardware)

2. **RecDisplay** (display of the recordings)

3. **ControlDisplay** (display of control recordings)

4. **Gilson** (Gilson liquid handler if applicable)

The commands for each object are listed in the tables below.

1.1.2 Oocyte Loop

In order to record from selected oocytes, functions must be executed within the so called **oocyte loop**. The array variable **SelectedWells** contains the indices of these wells and can be used in a for loop, e.g.

```
for (var i = 0; i < Robo2.SelectedWells.Count; i++)

{
var WellIndex = Robo2.SelectedWells[i];
Robo2.Log("now moving to well: " + Robo2.SelectedWellNames[i]);
Robo2.MoveToWell(WellIndex);
}
```

1.2 Variables

Variables come in different flavors:

1.2.1 JavaScript Variables

Variables can be created with the JavaScript keyword **var**. These variables can be used as in standard JavaScript to store values and use them later in the script.

var test = 1

generates a variable "test" with the value "1"

1.2.2 User Defined Interactive Variables

There are also user defined interactively changeable variables. These variable are defined by the command **SetDialogVariable**.

Robo2.SetDialogVariable("clampvoltage", -60, "Recording Voltage in mV");

generates the variable "clampvoltage" with the value -60. The comment "Recording Voltage in mV" will help you to identify the meaning of the variable when opening the dialog.

To interactively display and change the values the command **ShowDialog**() is used.

1.2.3 Predefined Variables

Some values which are important for the use of the Roboocyte2 are predefined (they do not have to be defined via **var**). These variables can be changed in the script source and also interactively via display of a dialog during script execution. Some of these variables are used as parameters in the high level commands, so that a script can be easily parameterized in the dialog without having to change the script source code.

All of these variables are of type **int**.

The syntax to set the values in the source is as follows: The command is a function call which starts with **Set_** and then the variable name (in upper case) is appended, e.g.

Set_DCOFFSET_RANGE(5);

which will assign the value **5** to the variable **DCOFFSET_RANGE**

To interactively display and change variable values the command **ShowStandardDialog**() is used.

List of all predefined variables, their default values, and respective command

Variable	Default	Description
Robo2.ResistanceCheck_I(MIN_RESISTANCE_I, MAX_RESISTANCE_I) Robo2.ResistanceCheck_U(MIN_RESISTANCE_U, MAX_RESISTANCE_U)		
MIN_RESISTANCE_I	100 Ohm	minimum TEVC probe resistance of the I electrode
MAX_RESISTANCE_I	1000 Ohm	maximum TEVC probe resistance of the I electrode
MIN_RESISTANCE_U	100 Ohm	minimum TEVC probe resistance of the U electrode
MAX_RESISTANCE_U	1000 Ohm	maximum TEVC probe resistance of the U electrode
Robo2.DCOffsetCorrection(DCOFFSET_RANGE, DCOFFSET_DELAY, DCOFFSET_WAIT, DCOFFSET_ATTEMPTS)		
DCOFFSET_RANGE	3 mV	max deviation of DC offset from 0
DCOFFSET_DELAY	20 sec	delay before DC offset measurement
DCOFFSET_WAIT	10 sec	wait after each check
DCOFFSET_ATTEMPTS	3	number of attempts to try DC

		offset check
Robo2.Impale(MIN_RMP, IMPALEMENT_STEPS, IMPALEMENT_STEP, IMPALEMENT_WAIT)		
MIN_RMP	-15 mV	minimum membrane potential
IMPALEMENT_STEPS_I	6	number of z axis steps to move down during impalement of the I electrode
IMPALEMENT_STEPS_U	2	number of z axis steps to move down during impalement of the U electrode
IMPALEMENT_STEPS	8	maximum number of z axis steps to move down during impalement
IMPALEMENT_STEP	50 µm	step size of impalement step
IMPALEMENT_WAIT	2 sec	wait after each z axis step
Robo2.InitialLeakCurrentCheck(MIN_INITIAL_LEAKCURRENT, MAX_INITIAL_LEAKCURRENT)		
MIN_INITIAL_LEAKCURRENT	-10000 nA	minimum leak current for initial check
MAX_INITIAL_LEAKCURRENT	200 nA	maximum leak current for initial check
Robo2.LeakCurrentCheck(MIN_LEAKCURRENT, MAX_LEAKCURRENT, LEAKCURRENT_ATTEMPTS, LEAKCURRENT_WAIT)		
MIN_LEAKCURRENT	-1000 nA	minimum leak current for second check
MAX_LEAKCURRENT	100 nA	maximum leak current for second check
LEAKCURRENT_WAIT	10	wait after each check [sec]
LEAKCURRENT_ATTEMPTS	3	number of attempts to try leak current check
Robo2.SetAmplifierCoefficients(AMPLIFIER_GAIN_P, AMPLIFIER_GAIN_I)		
AMPLIFIER_GAIN_P	1000 nA/mV	proportional coefficient 0 - 6000
AMPLIFIER_GAIN_I	100 1/s	integral coefficient 0 - 1000
Robo2.SetSampleRate(SAMPLERATE); Robo2.SetControlSampleRate(CONTROL_SAMPLERATE);		
SAMPLERATE	1000 Hz	recording sample rate 1 - 20000
CONTROL_SAMPLERATE	10 Hz	control recording sample rate 1 - 20000

Important note: These predefined variables are **not** active until they are explicitly used in a script command. E.g., when the SAMPLERATE variable is set to 10000, this sample rate is only used when the script command Robo2.SetSampleRate(SAMPLERATE) is executed.

Likewise, if the SAMPLERATE parameter is set to 10000, but the script command is Robo2.SetSampleRate(1000), the sample rate of 1000 is used for all recordings in the script.

1.2.4 Read-only Variables

These are variables the values of which cannot be changed by the user but are updated by the system.

Variable Robo2.	Description
SelectedWells	array of integer indices of the selected wells
SelectedWellNames	array of strings of the names of the selected wells
VALUE	last measured value (current or voltage)
VALUE_IC	last measured current at the current electrode
VALUE_UC	last measured voltage at the current electrode
VALUE_US	setpoint voltage (command potential)
VALUE_UV	last measured voltage at the voltage electrode
RESISTANCE_U	current impedance of the U electrode
RESISTANCE_I	current impedance of the U electrode
SCRIPT_FILE	file name of current script
DATE_NOW	current date (string) in the format yyyy-mm-dd, e.g. 2011-10-18
TIME_NOW	current time (string) in the format hh:mm:ss, e.g. 14:18:22, (Roboocyte2 V >= 1.1.5)
MINIMUM	minimum calculated value in the analysis ROI (to be used after a recording is finished
POS_MINIMUM	position of the calculated minimum
MAXIMUM	minimum calculated value in the analysis ROI
POS_MAXIMUM	position of the calculated maximum
AVERAGE	calculated average value in the analysis ROI
AREA	calculated area in the analysis ROI
BASELINE_AVERAGE	calculated baseline average in the baseline ROI
SCRIPT_FILE	name of the actual script file
TIME	the number of minutes passed since the last
TIME_S	the number of seconds passed since the last StartTimer command
ALIGNMENT_X	alignment x position in µm
ALIGNMENT_Y	alignment y position in µm
ALIGNMENT_Z	alignment z position in µm

1.3 Complete List of Commands

1.3.1 Conditions

Some Commands can be used only under certain conditions:

- **O+** must be within oocyte loop
- **O-** **not** within oocyte loop

- **W+** must be with carrier set to a well
- **L+** must be with z-axis moved into liquid
- **Oo+** must be with z-axis moved into oocyte
- **Oo-** must be with z-axis moved into oocyte

- **R+** must be within a recording
- **R-** **not** within a recording

Data types in Parameters:

- string: a text value, can either be a variable or a literal within double quotes (e.g. **„this is a string“**)
- int: an integer number (-5, 4, 5689, but not 1.4)
- bool: true or false
- --- no parameter needed

Some commands give back a return value or function as described

R: return value of function

1.3.2 Robo2. GUI-Commands

Robo2.	Parameter(s)	Action	Cond.	Example Script
Log("**x**")	string	Plots the message "**x**" in the Log window	---	Messages_eg.js
Information("**x**")	string	Opens a dialog with "**x**". Script execution is suspended until **OK** is pressed	R-	Messages_eg.js
Question("**x**")	string	Opens a dialog with "**x**". Different control paths in the script can be executed by using the return value of the function. The return value is **true** if **Yes** was clicked, **false** otherwise	R-	Messages_eg.js

1.3.3 Robo2. Variable Handling Commands

Robo2.	Parameter(s)	Action	Cond.	Example Script
SetDialogVariable(**x, y, z**)	string: **x** = name int: **y** = value string: **z** = description	defines and sets a value to a variable that can be changed interactively by a dialog during script execution (ShowDialog)	---	Variables_eg.js
ShowDialog()	---	shows dialog to check and change values for variables defined with SetDialogVariable()	R-	Variables_eg.js
ShowStandardDialog()	---	shows dialog to check and change values for the predefined (standard) variables	R-	Variables_eg.js

1.3.4 Robo2. Movement Commands

Robo2.	Parameter(s)	Action	Cond.	Example Script
MoveToWell(**x**)	int: well number	moves carrier to given well **x**, with x from 0 to 95	R-	Movement_eg.js
MoveToHomePos()	---	moves Z-axis and carrier to Home-Position	R-	Movement_eg.js
MoveToParkPos()	---	moves Z-axis and carrier to Park-Position	R-	Movement_eg.js
MoveToChangePlatePos()	---	moves Z-axis up and carrier to a position to change the plate	R-	Movement_eg.js
MoveToCoarsePos()	---	moves Z-axis and carrier to coarse position	R-	Movement_eg.js
ZMoveToLiquid()	---	moves Z-axis into the liquid z-height	R-	Movement_eg.js
ZMoveHome()	---	move Z-axis to uppermost position	R-	Movement_eg.js
ZMoveToOocyte()	---	moves Z-axis into the oocyte z-height	R-	Movement_eg.js
ZMoveStepDown(**x**)	int: 0 - 100	moves the z-axis down by **x** µm	---	---
ZMoveStepUp(**x**)	int: 0 - 100	moves the z-axis stepwise up by **x** µm	---	---
SetAxisLight()	---	turns the white LED at the axis on or off	---	---
GetAxisLight()	---	returns **true** if light is on	---	---
ReferenceXY()	---	performs a reference movement of the carrier	R-	---
ReferenceZ()	---	performs a reference movement of the z-axis	R-	---

1.3.5 Robo2. Amplifier and Data Acquisition Commands

Robo2.	Parameter(s)	Action	Cond.	Example Script
SetHoldingVoltage(**x**)	int: voltage in mV	sets clamp voltage to **x** mV	---	Acquisition_eg.js
SetVoltageClamp()	---	sets amplifier to voltage clamp mode	---	Acquisition_eg.js
SetHoldingCurrent(**x**)	int: current in nA	sets clamp current to **x** nA	---	Acquisition_eg.js
SetCurrentClamp()	---	sets amplifier to current clamp mode	---	Acquisition_eg.js
SetAmplifierCoefficients(**x, y**)	int: **x** = proportional gain int: **y** = integral gain	sets proportional and integral gain	---	Acquisition_eg.js
SetSampleRate(**x**)	int: samplerate	sets sample rate to **x** Hz for recordings	R-	Acquisition_eg.js
StartRecording()	---	starts data acquisition and recording of data	R-	Acquisition_eg.js
StopRecording()	---	stops data acquisition	R+	Acquisition_eg.js
SetControlSampleRate(**x**)	int: sample rate	sets control recording sample rate to **x** Hz (1-50)	R-	Acquisition_eg.js
StartControlRecording()	---	starts data acquisition, displayed in control window (not saved to file)	R-	Acquisition_eg.js
StopControlRecording()	---	stops control recording	R+	Acquisition_eg.js
RecordIVProtocol(**x**)	string: protocol name	executes the predefined IV protocol **x**	O+, R-	Acquisition_eg.js

1.3.6 Robo2. Data Analysis Commands

Robo2.	Parameter(s)	Action	Cond.	Example Script
SetBaselineROI(**x,y**)	int: **x** = left in s int: **y** = right in s	sets the region of interest (ROI) used for the baseline calculation.	---	Data_eg.js
SetBaselineROIMilliSec(**x,y**)	int: **x** = left in ms int: **y** = right in ms	sets the region of interest (ROI) used for the baseline calculation.	---	---
SetDriftCorrectionROI(**x,y**)	int: **x** = left in s int: **y** = right in s	sets the left and right cursor positions used to calculate the drift correction.	---	Data_eg.js
SetDriftCorrectionROIMilliSec(**x,y**)	int: **x** = left in ms int: **y** = right in ms	sets the left and right cursor positions used to calculate the drift correction.	---	---
SetAnalysisROI(**x,y**)	int: **x** = left in s int: **y** = right in s	sets the region of interest (ROI) from which results are calculated	---	Data_eg.js
SetAnalysisROIMilliSec(**x,y**)	int: **x** = left int: **y** = right in ms	sets the region of interest (ROI) from which results are calculated	---	---

1.3.7 Robo2. Database Commands

Robo2.	Parameter(s)	Action	Cond.	Example Script
TransmitRecording(**x**)	REC_TAG_VOLTAGE, REC_TAG_COMPOUND, REC_TAG_REF_VOLTAGE, REC_TAG_REF_COMPOUND	Writes recording tag to database. **REC_TAG_REF_COMPOUND**: not used for generating DR curves **REC_TAG_COMPOUND**: used for generating DR curves	R+	valves_tags_eg.js
TransmitVoltage(**x**)	int: value	writes voltage value for the current recording to the database	R+	
TransmitCompoundValve(**x**)	int: valve number	writes the valve number to the database	R+	valves_tags_eg.js
TransmitCompoundGilson(**x,y**)	int: **x** = slot int: **y** = tube	writes the Gilson slot/tube information to the database	R+	---
SetWellInfo(**x,y**)	string: **x** = key string: **y** = value	writes any user defined text to the database. Can be used to write additional information, e.g. value of electrode resistance or leak current to the database	W+	---
SetRecordingSeries(**x**)	int: series number	writes a series number to the database, valid for all following recordings. If one trace of the series is selected in Roboocyte2+, all other traces from the series are automatically used for plotting the dose-response curve.	R-	---

1.3.8 Robo2. Timing Commands

Robo2.	Parameter(s)	Action	Cond.	Example Script
Wait(**x**)	int: time	waits for **x** seconds before executing the next command	---	
WaitMilliSec(**x**)	int: time	waits for **x** milliseconds before executing the next command	---	
StartTimer()	---	starts timer, see also variables **TIME** and **TIME_S**	---	
WaitForTimer(**x**)	int: time	waits until **x** seconds from timer start have been passed.	---	

1.3.9 Robo2. Liquid Handling (Roboflow) Commands

Robo2.	Parameter(s)	Action	Cond.	Example Script
OpenValve(**x**)	int: valve number	opens valve **x**, valve index must be 1 ... 12	---	valves_tags_eg.js
CloseAllValves()	---	closes all valves	---	valves_tags_eg.js
WastePumpOn(**x**)	int: speed	turns waste pump on, speed **x** must be between 0 and 20000	---	valves_tags_eg.js
WastePumpOff()	---	turns off waste pump	---	valves_tags_eg.js
IsWastePumpOn()	R: bool	returns **true** if waste pump is on	---	---
ValvePumpOn(**x**)	int: speed	turns valve pump on, speed **x** must be between 0 and 10000	---	valves_tags_eg.js
ValvePumpOff()	---	turns off valve pump	---	valves_tags_eg.js
IsValvePumpOn()	R: bool	returns **true** if valve pump is on	---	---

1.3.10 Robo2. High Level Commands

Robo2.	Parameter(s)	Action	Cond.	Example Script
ResistanceCheck_I(**a,b**)	a = MIN_RESISTANCE_I b = MAX_RESISTANCE_I R: bool	checks if the resistance of the **current electrode** is between min and max (kOhm), returns **true** if this was the case	L+	HL-commands_eg.js
ResistanceCheck_U(**a,b**)	a = MIN_RESISTANCE_U b = MAX_RESISTANCE_U R: bool	checks if the resistance of the **voltage electrode** is between min and max (kOhm), returns **true** if this was the case	L+	HL-commands_eg.js
DCOffsetCorrection(**a,b,c,d**)	a = DCOFFSET_RANGE b = DCOFFSET_DELAY c = DCOFFSET_WAIT d = DCOFFSET_ATTEMPTS R: bool	checks if the offset of the I and U electrodes is within the range given by range (mV) delay: initial delay to wait until junction potential is established in seconds wait: wait time between repeats in seconds attempts: number of repeats.	L+	HL-commands_eg.js
Impale(**a,b,c,d**)	a = MIN_RMP b = IMPALEMENT_STEPS c = IMPALEMENT_STEP d = IMPALEMENT_WAIT R: bool	Impalement procedure **MIN_RMP**: minimum membrane potential **IMPALEMENT_STEPS**: maximum number of steps to "find" MIN_RMP on both electrodes **IMPALEMENT_STEP**: single step size in µm **IMPALEMENT_WAIT**: wait time in seconds between steps returns **true** if impalement was successful	W+	HL-commands_eg.js
InitialLeakCurrentCheck(**a,b**)	a = MIN_INITIAL_LEAKCURRENT b = MAX_INITIAL_LEAKCURRENT R: bool	initial leak current test, returns **true** if the measured leak current is between min and max (nA)	Oo+	HL-commands_eg.js
LeakCurrentCheck(**a,b,c,d**)	a = MIN_LEAKCURRENT b = MAX_LEAKCURRENT c = LEAKCURRENT_WAIT d = LEAKCURRENT_ATTEMPTS R: bool	leak current test, returns **true** if the measured leak current is between min and max (nA). Waits c seconds before determining the leak current and retries d times	Oo+	HL-commands_eg.js

1.3.11 RecDisplay. Recording Display Commands

RecDisplay.	**Parameter(s)**	**Action**	**Cond.**	**Example Script**
Clear()	---	clears the recording display	---	valves_tags_eg.js
SetMode(**x**)	x = **DISP_SINGLE** or **DISP_OVERLAY**	sets recording display to overlay or single trace mode	---	---
SetXAxis(**x,y**)	int: min int: max	sets x-axis range from min to max in seconds	---	valves_tags_eg.js
SetXAxisMilliSec(**x,y**)	int: min int: max	sets x-axis range from min to max in milliseconds	---	---
SetYAxis(**x,y**)	int: min int: max	set y axis range from min to max in nA	---	---
ZoomToFitX()	---	zooms the time axis to the actual data trace	---	Acquisition_eg.js
ZoomToFitY()	---	zooms the current axis to the actual data trace	---	---
RecDisplay.TrackYMax(**x**);	bool: true or false	true or false switches the maximum current tracking on or off	---	valves_tags_eg.js
RecDisplay.TrackYMin(**x**);	bool: true or false	true or false switches the minimum current tracking on or off	---	valves_tags_eg.js

1.3.12 ControlDisplay. Control Display Command

ControlDisplay.	**Parameter(s)**	**Action**	**Cond.**	**Example Script**
Clear()	---	clears the control display	---	---
SetXAxis(**x,y**)	int: min int: max	sets x-axis range from min to max in seconds	---	valves_tags_eg.js
SetYAxis_I(x,y)	int: min int: max	sets current axis range from min to max in seconds	---	valves_tags_eg.js
SetYAxis_U(x.y)	int: min int: max	sets voltage axis range from min to max in seconds	---	valves_tags_eg.js
ZoomToFitX()	---	zooms the time axis to the actual data trace	---	---
ZoomToFitY_U()	---	zooms the voltage axis to the actual data trace	---	---

1.3.13 Gilson. Gilson Commands

Gilson.	**Parameter(s)**	**Action**	**Cond.**	**Example Script**
MoveToTube(**x,y**)	int: **x** = slot int: **y**= tube	moves to tube in slot 1 .. 5, tube depending on selected rack	---	
MoveToPort(**x**)	int: port number (1 or 2)	moves to transfer ports	---	
MoveToRinse()	---	moves to the rinse station	---	
MoveToDrain()	---	moves to the drain station	---	
MoveUp()	---	moves the Gilson probe up	---	
MoveHome()	---	moves all to home position	---	
Reset()	---	resets the Gilson	---	
PumpBackward(**x**)	int: speed	starts Gilson peristaltic pump in backward direction with speed x	---	
PumpForward(**x**)	int: speed	starts Gilson peristaltic pump in forward direction with speed x	---	
Gilson.PumpStop()	---	stops Gilson peristaltic pump	---	
Valve1On()	---	opens valve 1 from transfer port	---	
Valve1Off()	---	closes valve 1 from transfer port	---	
Valve2On()	---	opens valve 2 from transfer port	---	
Valve2Off()	---	closes valve 1 from transfer port	---	

2 Example Script "Dose-Response"

In the following chapter, you will learn how a typical recording script is built. As an example we will use the script: "**dose-response.js**" which can be found on the Roboocyte Setup CD. Another example script which can be found on the Roboocyte CD is the script "**standard_procedures.js**". This script only includes the protocol independent part without the recording protocol part and without user-defined variables

The following chapters describe a typical recording protocol sequence step-by-step:

1. Defining variables and parameter uses in the script

2. Starting the oocyte loop

   3. Compensating electrode DC voltage offsets

   4. Testing electrode resistance and voltage offsets

   5. Oocyte impalement

   6. Starting voltage-clamp and leak-current test

   7. Executing the experimental protocol

   8. Continue with next selected oocyte

2.1 Defining Dialogue variables

2.1.1 User Defined Dialogue Variables

The first part in a script should be used to define all necessary variables, such as holding potential, incubation times, number of used valves etc. If these parameters are defined as dialogue variables, they can be easily modified in a dialogue before starting the script recording

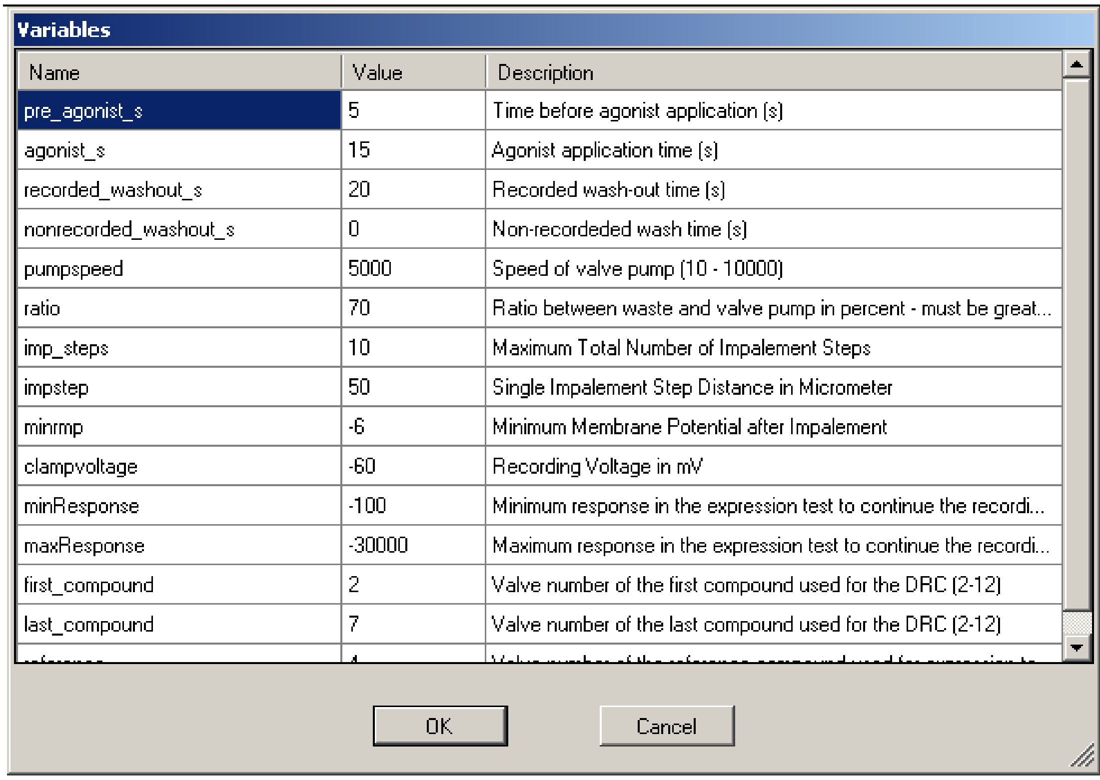
```
Robo2.Log("Script: " + SCRIPT_FILE)


// Definition of user defined variables

Robo2.SetDialogVariable("pre_agonist_s", 5, "Time before agonist application (s)");
Robo2.SetDialogVariable("agonist_s", 15, "Agonist application time (s)");
Robo2.SetDialogVariable("recorded_washout_s", 20, "Recorded wash-out time (s)");
Robo2.SetDialogVariable("nonrecorded_washout_s", 0, "Non-recordeded wash time (s)");

Robo2.SetDialogVariable("pumpspeed", 5000, "Speed of valve pump (10 - 10000)");
Robo2.SetDialogVariable("ratio", 70, "Ratio between waste and valve pump in percent - must be greater than 70");

Robo2.SetDialogVariable("imp_steps", 10, "Maximum Total Number of Impalement Steps");
Robo2.SetDialogVariable("impstep", 50, "Single Impalement Step Distance in Micrometer");
Robo2.SetDialogVariable("minrmp", -6, "Minimum Membrane Potential after Impalement");

Robo2.SetDialogVariable("clampvoltage", -60, "Recording Voltage in mV");
Robo2.SetDialogVariable("minResponse", -100, "Minimum response in the expression test to continue the recording (nA)");
Robo2.SetDialogVariable("maxResponse", -30000, "Maximum response in the expression test to continue the recording (nA)");
Robo2.SetDialogVariable("first_compound", 2, "Valve number of the first compound used for the DRC (2-12)");
Robo2.SetDialogVariable("last_compound", 7, "Valve number of the last compound used for the DRC (2-12)");
Robo2.SetDialogVariable("reference", 4, "Valve number of the reference compound used for expression test (3-12)");


Robo2.ShowDialog();

```

The command for defining a user defined (dialogue) variable has the following format::

Robo2.SetDialogVariable("**variable_name**", **variable value** , "**comment**");

I.e. the command "**Robo2.SetDialogVariable("pre_agonist_s**", 5, "Time before agonist application (s)"); creates the variable "**pre_agonist_s**" with the default value "**5**" and the comment "**Time before agonist application (s)**"

The command "**Robo2.ShowDialog();**" opens a window after starting the script listing all user defined variables. All values can be changed here, but are only valid for a single script execution. Please note that whenever you start the script a second time, default values will be reloaded.



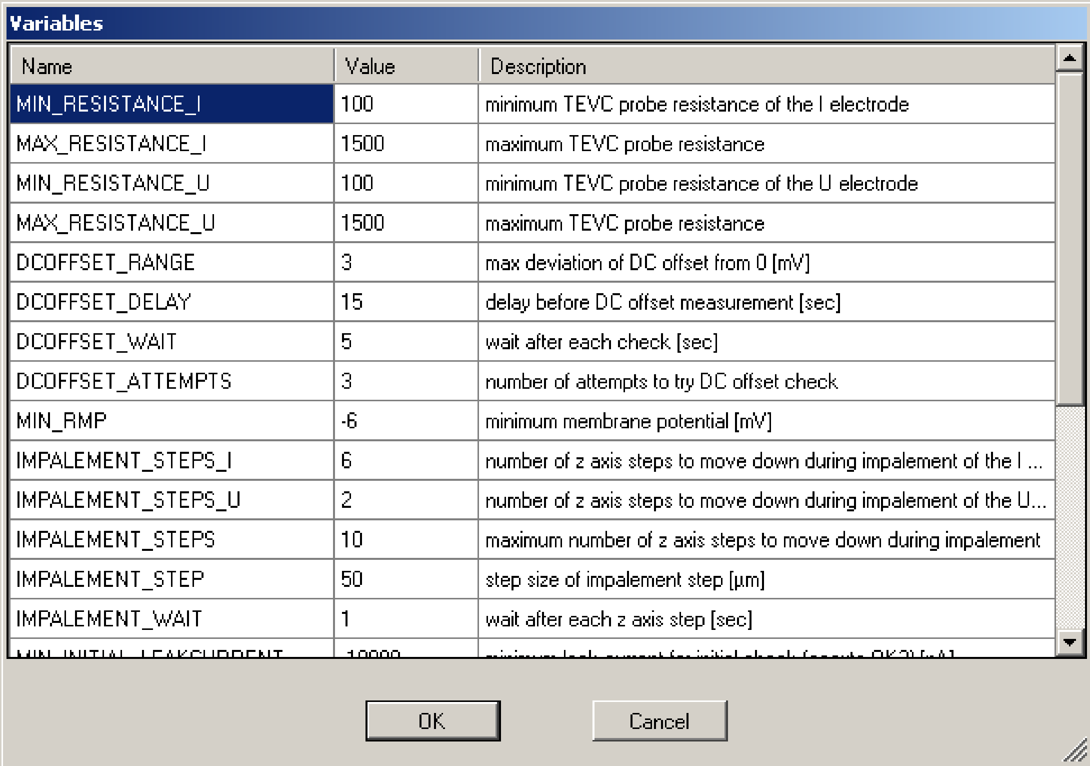
Name	Value	Description
pre_agonist_s	5	Time before agonist application [s]
agonist_s	15	Agonist application time [s]
recorded_washout_s	20	Recorded wash-out time [s]
nonrecorded_washout_s	0	Non-recordeded wash time [s]
pumpspeed	5000	Speed of valve pump (10 - 10000)
ratio	70	Ratio between waste and valve pump in percent - must be great...
imp_steps	10	Maximum Total Number of Impalement Steps
impstep	50	Single Impalement Step Distance in Micrometer
minrmp	-6	Minimum Membrane Potential after Impalement
clampvoltage	-60	Recording Voltage in mV
minResponse	-100	Minimum response in the expression test to continue the recordi...
maxResponse	-30000	Maximum response in the expression test to continue the recordi...
first_compound	2	Valve number of the first compound used for the DRC (2-12)
last_compound	7	Valve number of the last compound used for the DRC (2-12)

2.1.2 Working with Pre-defined Variables

Pre-defined variables are already existing variables with predefined default variable names and values (see table in chapter 1.2.3).

```

Set_SAMPLERATE(500)

Set_AMPLIFIER_GAIN_P(1000)
Set_AMPLIFIER_GAIN_I(100)

Set_DCOFFSET_RANGE(3);
Set_DCOFFSET_DELAY(15);
Set_DCOFFSET_WAIT(5);

Set_MIN_RMP(minrmp);
Set_IMPALEMENT_STEPS(imp_steps)
Set_IMPALEMENT_STEP(impstep)
Set_IMPALEMENT_WAIT(1)

Set_MIN_INITIAL_LEAKCURRENT(-10000)
Set_MAX_INITIAL_LEAKCURRENT(500)

Set_MIN_LEAKCURRENT(-1000)
Set_MAX_LEAKCURRENT(200)

Set_LEAKCURRENT_WAIT(10)
Set_LEAKCURRENT_ATTEMPTS(3)

Set_MIN_RESISTANCE_I(100)
Set_MAX_RESISTANCE_I(1500)

Set_MIN_RESISTANCE_U(100)
Set_MAX_RESISTANCE_U(1500)

// This opens the dialogue to change values for the built-in variables

Robo2.ShowStandardDialog()

```

The command "**Robo2.ShowStandardDialog();**" opens a window after starting the script listing all predefined variables. All values can be changed here, but again changes are not permanent and only valid for a single script execution. If you want to change values "permanently" you have to change them in the script text.



Name	Value	Description
MIN_RESISTANCE_I	100	minimum TEVC probe resistance of the I electrode
MAX_RESISTANCE_I	1500	maximum TEVC probe resistance
MIN_RESISTANCE_U	100	minimum TEVC probe resistance of the U electrode
MAX_RESISTANCE_U	1500	maximum TEVC probe resistance
DCOFFSET_RANGE	3	max deviation of DC offset from 0 [mV]
DCOFFSET_DELAY	15	delay before DC offset measurement [sec]
DCOFFSET_WAIT	5	wait after each check [sec]
DCOFFSET_ATTEMPTS	3	number of attempts to try DC offset check
MIN_RMP	-6	minimum membrane potential [mV]
IMPALEMENT_STEPS_I	6	number of z axis steps to move down during impalement of the I ...
IMPALEMENT_STEPS_U	2	number of z axis steps to move down during impalement of the U...
IMPALEMENT_STEPS	10	maximum number of z axis steps to move down during impalement
IMPALEMENT_STEP	50	step size of impalement step [µm]
IMPALEMENT_WAIT	1	wait after each z axis step [sec]

The next section in the script - Initialization - uses some of the predefined variables.

```
//** Initialisation - Setting sample rate and amplifier gain values
// -----------------

Robo2.SetAmplifierCoefficients(AMPLIFIER_GAIN_P, AMPLIFIER_GAIN_I);
RecDisplay.Clear();
Robo2.SetSampleRate(SAMPLERATE);
```

After defining variables and their respective values, the "oocyte loop" can be initiated.

2.2 The Oocyte Loop

The most basic function of the Roboocyte is to move the plate carrier "from one well to the other", or more correct to move selected wells exactly below the measuring head. This movement between well 1 and 95 is initiated and controlled by the so-called "oocyte loop", a for-loop in java script.

```
89  for (var i = 0; i < Robo2.SelectedWells.Count; i++)
90  {
91      var WellIndex = Robo2.SelectedWells[i];
92  
93  // ********************************************************************
94  // Electrode offset compensation, Impalement, Leak Current Checks etc.
95  // ********************************************************************
96  
97  // ********************************************************************
98  // Protocol specific part
99  // ********************************************************************
100 
101     Robo2.ValvePumpOff();
102     Robo2.CloseAllValves();
103     Robo2.WastePumpOff();
104 
105 }
```

The oocyte loop itself has only 4 lines (**lines 89, 90, 91 and 105** in the example shown above), but, before terminating the oocyte loop by the final curly bracket, do not forget to stop the valve pump to avoid flooding (**lines 101, 102 and 103**)

2.3 Standard Routines executed before Recording Protocol

Before starting the recording of data, a number of steps and routines have to be started which usually will not change for different kind of recording protocols.

2.3.1 Moving the Measuring Head into Liquid

The oocyte loop cares for incrementing the well number, the movement of carrier and z-axis have to be controlled by specific commands.

```
    // Measuring head moves to selected well

    Robo2.Log("_______________________________");
    Robo2.Log("Starting with new oocyte in well " + Robo2.SelectedWellNames[i] );
    Robo2.Log("     ");


        Robo2.MoveToWell(WellIndex);
        Robo2.WastePumpOn(pumpspeed*ratio/100);
        Robo2.Wait(1);
        Robo2.ZMoveToLiquid();
        Robo2.Wait(2);
```

Before moving the measuring head into liquid, one should switch the waste pump on to guarantee a constant buffer level relative to the electrodes. This is important to avoid offset artifacts during oocyte impalement.

Robo2.WastePumpOn(pumpspeed*ratio/100); starts the waste pump.

The waste pump "speed" is defined by the values for **pumpspeed** and **ratio** as defined before.

Robo2.MoveToWell(WellIndex); moves the carrier to the respective well.

Robo2.ZMoveToLiquid(); moves the z-axis down into the well.

The default liquid position can be changed in the Settings/Options menu of the Roboocyte software (default is 2000 µm above the well bottom).

2.3.2 Switching to Current Clamp Mode

After moving the electrodes into liquid, the amplifier is set to Current Clamp Mode.

Robo2.SetHoldingCurrent(0); sets holding current to 0 nA

Robo2.SetCurrentClamp(); sets the amplifier to current clamp mode

```
    ControlDisplay.SetXAxis(0, 30);
    ControlDisplay.SetYAxis_U(-100, 10);
    ControlDisplay.SetYAxis_I(-30, 30);

// Starting Current Clamp mode

    Robo2.Log("Current clamp at 0 nA");
    Robo2.SetHoldingCurrent(0);
    Robo2.SetCurrentClamp();
```

Before starting control recording, scaling of the Control Display axes is performed. Units are seconds, mV and nA, respectively.

2.3.3 Electrode Offset Compensation

After starting current clamp, control recording is started in order to perform electrode offset compensation, to determine electrode resistances, and to perform the impalement of the oocyte.

Control recording means that data are displayed in the Control Display, but not saved to disk.

Robo2.StartControlRecording() starts the control recording followed by the electrode offset compensation.

```
        Robo2.StartControlRecording();

        // Electrode offset reset
        Robo2.Log("Elektrode reset ...");

        if (!Robo2.DCOffsetCorrection(DCOFFSET_RANGE, DCOFFSET_DELAY, DCOFFSET_WAIT, DCOFFSET_ATTEMPTS))
        {
            continue;
        }

```

The offset compensation for both electrodes is performed by executing the command

Robo2.DCOffsetCorrection(DCOFFSET_RANGE, DCOFFSET_DELAY, DCOFFSET_WAIT, DCOFFSET_ATTEMPTS)

Variables for the offset compensation are by default the designated predefined variables, but you can also use numbers, such as **Robo2.DCOffsetCorrection(3, 10, 5, 3)** (see table in chapter 1.2.3) The command is embedded in an if-loop, controlling what happens after the compensation: If the electrode offset compensation was successful the script proceeds, if not, the measuring head moves to the next oocyte.

The if loop is terminated by "**continue**" if the electrode offset compensation fails. "**Continue**" means that the script jumps back to the start of the respective loop (oocyte loop in this case).

2.3.4 Electrode Resistance Test

After compensating the electrode offsets, the resistance of both electrodes should be tested with the command

Robo2.ResistanceCheck_I(MIN_RESISTANCE_I, MAX_RESISTANCE_I),

where **MIN_RESISTANCE** and **MAX_RESISTANCE** are predefined variables (see table in chapter 1.2.3).

```

        if (Robo2.ResistanceCheck_I(MIN_RESISTANCE_I, MAX_RESISTANCE_I))
        {
        }
        else
        {Robo2.StopControlRecording();
            continue;
        }

        if (Robo2.ResistanceCheck_U(MIN_RESISTANCE_U, MAX_RESISTANCE_U))
        {
        }
        else
        {Robo2.StopControlRecording();
            continue;
        }

        Robo2.StopControlRecording();

```

If the electrode resistances are within the range (between **MIN_RESISTANCE** and **MAX_RESISTANCE**) the script proceeds; if not (**else**), the script breaks and continues with the next oocyte.

Now, after completing electrode offset compensation, oocyte impalement can be started.

2.3.5 Oocyte Impalement

The oocyte impalement is performed by using the command:

Robo2.Impale(MIN_RMP, IMPALEMENT_STEPS, IMPALEMENT_STEP, IMPALEMENT_WAIT)

MIN_RMP, IMPALEMENT_STEPS, IMPALEMENT_STEP and IMPALEMENT_WAIT are predefined variables (see table in chapter 1.2.3).

The impalement process is started by a movement of the z-axis to the default oocyte impalement depth (default = 800 µm above the well plate bottom). Then n = **IMPALEMENT_STEPS** are performed with a step size of Δz = **IMPALEMENT_STEP** and a waiting time t = **IMPALEMENT_WAIT** between individual steps.

Default values for **IMPALEMENT_STEPS** and **IMPALEMENT_STEP** are 8 and 50 µm, respectively.

```

        Robo2.Log("Starting Oocyte Impalement...");

        Robo2.StartControlRecording();

        Robo2.Wait(2);

        ControlDisplay.SetXAxis(0, 30)
        ControlDisplay.SetYAxis_U(-60, 10)

        Robo2.Wait(5)

        if (!Robo2.Impale(MIN_RMP, IMPALEMENT_STEPS, IMPALEMENT_STEP, IMPALEMENT_WAIT))
        {
        Robo2.Log("impalement failed, " + MIN_RMP + " not reached --> next oocyte");
        Robo2.ValvePumpOff();
        Robo2.CloseAllValves();
        Robo2.WastePumpOff();
            continue;
        }

```

If the impalement was not successful, i.e. if the membrane potential **U = MIN_RMP** was not reached after **n = IMPALEMENT_STEPS** steps with the step distance **d = IMPALEMENT_STEP** the run is continued with the next oocyte. Before moving to the next oocyte pumps and valves should be closed (lines 170 - 172). **IMPALEMENT_WAIT** is the waiting time between individual steps of the z-axis in seconds.

If impalement was successful, the script continues after line 175. It usually makes sense to wait at least for 10 seconds after impalement to give the oocyte membrane enough time to recover from the impalement.

```
177
178        Robo2.Log("Waiting 10 s for resealing of membrane ...");
179        Robo2.Wait(10);
180        Robo2.Log("... Final membrane potential after 10 s = " + Robo2.VALUE + "mV");
181        Robo2.Log(" ");
182
183        Robo2.StopControlRecording();
184
```

2.3.6 Starting Voltage Clamp Mode

After successful impalement, the recording can be continued under voltage clamp. Before, scaling of the Control Display axes should be changed to meet the demands of the following Leak Current Test under voltage clamp.

```
186
187        Robo2.Log("Voltage clamp ...");
188
189        ControlDisplay.SetXAxis(0, 60)
190        ControlDisplay.SetYAxis_U(clampvoltage-10, clampvoltage+10)
191        ControlDisplay.SetYAxis_I(MIN_INITIAL_LEAKCURRENT, MAX_INITIAL_LEAKCURRENT)
192
193
194        Robo2.SetHoldingVoltage(clampvoltage);
195        Robo2.SetVoltageClamp();
196
```

Robo2.SetHoldingVoltage(clampvoltage); sets the command voltage .

"clampvoltage" is a user defined variable created at the beginning of the script.

Robo2.SetVoltageClamp();sets the amplifier to voltage clamp mode.

2.3.7 Initial Leak Current Test

After starting voltage clamp, a control recording is started in order to perform the initial leak current check.

```
198
199        Robo2.StartControlRecording();
200        Robo2.Wait(5);
201
202        if (!Robo2.InitialLeakCurrentCheck(MIN_INITIAL_LEAKCURRENT, MAX_INITIAL_LEAKCURRENT))
203        {
204        Robo2.Log("Proceeding with next oocyte");
205        Robo2.ValvePumpOff();
206        Robo2.CloseAllValves();
207        Robo2.WastePumpOff();
208        continue;
209        }
```

The leak current check is started by the command

Robo2.InitialLeakCurrentCheck(MIN_INITIAL_LEAKCURRENT, MAX_INITIAL_LEAKCURRENT)

where **MIN_INITIAL_LEAKCURRENT** and **MAX_INITIAL_LEAKCURRENT** are predefined variables with default values -10000 and +200, respectively. If the leak current is within the range the script continues after line 209; if not, the run continues with the next selected oocyte. Again, commands for stopping pumps and valves should be included (lines 205 - 207).

2.3.8 Final Leak Current Test with Perfusion

After the successful initial leak current check, i.e. if the oocyte is not completely leaky, the script continues with the final leak current check. Although you can start this final check directly after the initial leak current test, we recommend to start the buffer perfusion of the oocyte beforehand in order to check the viability of the oocyte under perfusion.

```

        Robo2.Log("Starting Perfusion ...");

        Robo2.Wait(5)
        Robo2.OpenValve(1);
        Robo2.ValvePumpOn(pumpspeed);
        Robo2.Wait(5);

```

Then, after a waiting time of 5 seconds, the final leak current check can be initiated by the command

Robo2.LeakCurrentCheck(MIN_LEAKCURRENT, MAX_LEAKCURRENT, LEAKCURRENT_ATTEMPTS, LEAKCURRENT_WAIT)

where **MIN_LEAKCURRENT, MAX_LEAKCURRENT, LEAKCURRENT_ATTEMPTS** and **LEAKCURRENT_WAIT** are predefined variables. Default values for **MIN_LEAKCURRENT** and **MAX_LEAKCURRENT** are -1000 nA and 100 nA, respectively. Default values for **LEAKCURRENT_ATTEMPTS** and **LEAKCURRENT_WAIT** are 3 and 10 seconds, respectively. This means that the leak current is determined up to 3 times with a waiting time of 10 seconds in between.

```
226
227        Robo2.Log("Final leak current check (maximum = " + MIN_LEAKCURRENT + " nA) ...");
228
229        if (!Robo2.LeakCurrentCheck(MIN_LEAKCURRENT, MAX_LEAKCURRENT, LEAKCURRENT_ATTEMPTS, LEAKCURRENT_WAIT))
230        {
231        Robo2.Log("leak current not in range");
232        Robo2.ValvePumpOff();
233        Robo2.CloseAllValves();
234        Robo2.WastePumpOff();
235        continue;
236        }
237
```

As soon as the leak current is within the given limits, the script continues after line 237. if the cell is still too leaky after 3 trials, the script continues with moving to the next well.

After successful leak test, the control recording has to be stopped, and the result of final leak test can be sent into the log window.

```
        Robo2.StopControlRecording();

        Robo2.Log("... Leak current with perfusion OK = " + Robo2.VALUE + " nA.");
            Robo2.Log("    ");
        Robo2.Wait(5)

```

Now, after finishing all necessary preparations and tests, the specific recording protocol can be commenced.

3 Recording Protocol Examples

3.1 Using the Dose-Response Script as a Template

The example script "**dose-response.js**" is a typical dose-response recording protocol which can be used as a template for all kind of dose-response like protocols. The recording protocol specific part is located between line 254 and 365.

```
// ***************************************************************************
// Protocol specific part starts here
// ***************************************************************************



// ***************************************************************************
// Protocol specific part ends here
// ***************************************************************************
```

The part of the script outside of "Protocol specific part starts here/ Protocol specific part stops here" comment can be used as a starting-point for your own scripts.

3.2 Ligand-gated Channels and Electrogenic Transporters

The example script "**dose-response.js**" is a typical dose-response recording protocol for ligand-gated ion channels. In addition, it can be used - after minor modifications - for electrogenic transporters, such as Na-pump, GATs, EAATs etc.

3.2.1 Expression Test

Before you start a time and compound-consuming protocol on an oocyte, it makes sense to test whether the expression i.e. the response to an agonist reference concentration is adequate. Therefore, you should usually perform an expression test before continuing the protocol.

Before the expression test is started, it is mandatory to define the ranges of baseline ROI (Region Of Interest) and analysis ROI, because these ROIs will be used for the calculation of the baseline subtracted response for expression test. They will also be used for the later analysis in Roboocyte2+. Although the ROI positions should be defined properly already in the script, they can be changed later in Roboocyte2+.

```
254
255     Robo2.Log("Starting expression test with reference compound from valve " + reference + " ");
256
257     RecDisplay.SetXAxis(0, pre_agonist_s + agonist_s + recorded_washout_s);
258     RecDisplay.TrackYMax(true);
259     RecDisplay.TrackYMin(true);
260     Robo2.SetBaselineROI(pre_agonist_s - 3, pre_agonist_s - 1);
261     Robo2.SetAnalysisROI(pre_agonist_s, pre_agonist_s + agonist_s);
262
263     ControlDisplay.SetXAxis(0, pre_agonist_s + agonist_s + recorded_washout_s)
264     ControlDisplay.SetYAxis_I(-10000, 100)
265
```

The user-defined variables **pre_agonist_s**, **agonist_s and recorded_washout_s** used later for controlling the recording time and solution exchange are also used for defining the ROIs and the time axis scaling with the effect that ROI boundaries and axis change accordingly when you change variable values at script start. If the ROI boundaries are coupled to these variables, they automatically change whenever changes to these variables are made.

Example:

RecDisplay.SetXAxis(0, pre_agonist_s + agonist_s + recorded_washout_s);

Robo2.SetBaselineROI(pre_agonist_s - 3, pre_agonist_s - 1);

Robo2.SetAnalysisROI(pre_agonist_s, pre_agonist_s + agonist_s);

ControlDisplay.SetXAxis(0, pre_agonist_s + agonist_s + recorded_washout_s)

ControlDisplay.SetYAxis_I(-10000, 100)

Above JavaScript commands lead to durations listed in the following table.

	Example A	**Example B**	**Example C**
Variable values (s)	**5/30/30**	**10/20/20**	**20/40/60**
RecDisplay x-axis (s)	65	50	120
Baseline ROI (s)	2 - 4	7 - 9	17 - 19
Analysis ROI (s)	5 - 35	10 - 30	20 - 60
ControlDisplay x-axis	65	50	120
RecDisplay y-axis	autoscaling: **RecDisplay.TrackYMax(true); RecDisplay.TrackYMin(true);**		

After these definitions, the expression test can be initiated.

```

    Robo2.StartRecording();
    Robo2.TransmitRecording(REC_TAG_REF_COMPOUND);
    Robo2.TransmitCompoundValve(reference);
    Robo2.Wait(pre_agonist_s);
    Robo2.OpenValve(reference);
    Robo2.Wait(agonist_s);
    Robo2.OpenValve(1);
    Robo2.Wait(recorded_washout_s);
    Robo2.StopRecording();

```

After starting the recording, two transmit commands are executed to send the right information about the compound used to the database and to link this information to the actual recording.

Robo2.TransmitRecording(REC_TAG_REF_COMPOUND); tags the recording as a **reference recording** which means that the result will **not** be used for dose-response fitting in Roboocyte2+.

Robo2.TransmitCompoundValve(reference); transmits the compound name and concentration to the database. This of course only works when the right entries have been made to the **Liquid Configuration** in the Roboocyte2 program.

pre_agonist_s seconds after recording start, the valve **reference** is opened. Then after **agonist_s** seconds valve **1** is opened and after another **recorded_washout_s** seconds the recording is stopped.

Then, after calculating and typing the baseline-subtracted response into the log window,

```

    Robo2.Log(" ");
    Robo2.Log("Baseline current before application = " + Robo2.BASELINE_AVERAGE + " nA");
    signal = Robo2.MINIMUM - Robo2.BASELINE_AVERAGE
    Robo2.Log("Response to reference concentration from valve " + reference + " = " + signal + " nA");
    Robo2.Log(" ");

```

the response can be compared to the values assigned to the user-defined variables **minResponse** and **maxResponse** the result of which determines whether the script continues at line 297 or is stopped and continued with the oocyte from the next well.

```
286
287     if (signal > minResponse || signal < maxResponse)
288     {
289         Robo2.Log("Response (" + signal + ") not in Range (" + minResponse + ", " + maxResponse + "), go to next oocyte");
290         Robo2.Wait(2);
291
292         Robo2.ValvePumpOff();
293         Robo2.CloseAllValves();
294         Robo2.WastePumpOff();
295         continue;
296     }
297
```

Example1: Negative response expected (inward current)

Condition: if (signal > minResponse || signal < maxResponse)

maxResponse = -10000

minResponse = -200

signal = -100 ==> signal > minResponse ==> next oocyte

signal = -12000 ==> signal < maxResponse ==> next oocyte

signal = -5000 ==> signal > maxResponse and signal < minResponse ==> script proceeds

Example2: Positive response expected

If the expected response is positive (outward current) operators < and > have to be exchanged.

Condition: if (signal < minResponse || signal > maxResponse)

maxResponse = 10000

minResponse = 200

signal = 100 ==> signal < minResponse ==> next oocyte

signal = 12000 ==> signal > maxResponse ==> next oocyte

signal = 5000 ==> signal < maxResponse and signal > minResponse ==> script proceeds

3.2.2 Dose-Response Protocol

After the oocyte has been tested successfully for sufficient expression and stability, the recording protocol continues with the does-response recording part of the script. Before the dose-response recording starts, the usual axis scaling and ROI definition commands are applied.

```

Robo2.Log("Starting dose response curve with liquids in valves " + first_compound + " to " + last_compound + " ...");

ControlDisplay.SetXAxis(0, pre_agonist_s + agonist_s + recorded_washout_s + nonrecorded_washout_s)

RecDisplay.SetXAxis(0, pre_agonist_s + agonist_s + recorded_washout_s);
RecDisplay.TrackYMax(true);
RecDisplay.TrackYMin(true);
Robo2.SetBaselineROI(pre_agonist_s - 3, pre_agonist_s - 1);
Robo2.SetAnalysisROI(pre_agonist_s, pre_agonist_s + agonist_s);

```

If the application of different agonist concentrations steps sequentially from valve a to valve a + 1 a for loop can be used.

```

    for (var valve = first_compound; valve <= last_compound; valve++)
    {

    }
```

The loop variable is the valve number, starting with the user defined variable first_compound and ending with last_compound.

If needed, a leak test can be placed before the application of every agonist application.

```
322         Robo2.StartControlRecording();
323
324         if (!Robo2.LeakCurrentCheck(MIN_LEAKCURRENT, MAX_LEAKCURRENT, LEAKCURRENT_ATTEMPTS, LEAKCURRENT_WAIT))
325         {
326             Robo2.Log("leak current not in range - next oocyte");
327             Robo2.ValvePumpOff();
328             Robo2.CloseAllValves();
329             Robo2.WastePumpOff();
330
331             break;
332         }
333         Robo2.StopControlRecording();
```

If the leak test fails, the message "leak current is not in range - next oocyte" is send to the log window, all pumps and valves are switched off and the protocol continues with the next selected oocyte. The break command in line 331 terminates the execution of the loop and the script is continued after the end of the loop at line 365. The curly bracket in line 368 designates the end of the oocyte loop, which means that the script will continue with the next selected well/oocyte at line 89.

```
365     Robo2.ValvePumpOff();
366     Robo2.CloseAllValves();
367     Robo2.WastePumpOff();
368 }
```

3.3 Voltage-gated Ion Channels

Because the external trigger for activating voltage-gated ion channels is usually a change in membrane potential and not application of an agonist, protocols usually concentrate on voltage changes instead of solution exchanges. Of course, voltage protocols and solution exchanges still can be combined in a script; e.g. to modify the activity of voltage-gated ion channels by application of different compounds.

3.3.1 Expression Test

The detailed design of an expression test will depend on the channel properties, but will always have a qualitatively similar structure: A voltage step from a potential at which the channels are inactive to a potential which activates the channels will be applied. The script "expression_test_vgic_part.js" used here as an example can be found on the Roboocyte2 CD.

First, new variables should be defined. This part can be moved to the beginning of the script.

```
    Robo2.SetDialogVariable("pre_pulse_ms", 100, "Prepulse for expression test (ms)");
    Robo2.SetDialogVariable("pulse_ms", 200, "Pulse duration for expression test (ms)");
    Robo2.SetDialogVariable("post_pulse_ms", 200, "Post-pilse duration for expression test (ms)");

    Robo2.SetDialogVariable("voltage_rest", -60, "Resting Voltage for expression test (mV)");
    Robo2.SetDialogVariable("voltage_et", 50, "Activating Voltage for expression test (mV)");
```

Secondly, axis scaling and ROI position are set. Using the variables from above guarantees that scaling and ROI automatically change whenever the pulse timing is changed.

```
    RecDisplay.SetXAxisMilliSec(0, pre_pulse_ms + pulse_ms + post_pulse_ms);
    RecDisplay.TrackYMax(true);
    RecDisplay.TrackYMin(true);

    Robo2.SetAnalysisROIMilliSec(pre_pulse_ms + 10, pre_pulse_ms + pulse_ms);
```

The analysis ROI covers the whole duration of the voltage (test) pulse except the first 10 ms in order to avoid interference with the capacitive transient caused by the voltage jump.

Next, the recording is started.

```
    Robo2.SetHoldingVoltage(voltage_rest);
    Robo2.SetVoltageClamp();

    Robo2.StartRecording();
    Robo2.TransmitRecording(REC_TAG_REF_VOLTAGE);
    Robo2.TransmitVoltage(voltage_et);

    Robo2.WaitMilliSec(100);

    Robo2.SetHoldingVoltage(voltage_et);
    Robo2.WaitMilliSec(200);

    Robo2.SetHoldingVoltage(voltage_rest);
    Robo2.WaitMilliSec(200);

    Robo2.StopRecording();
```

The sequence generates the following voltage jump:

a. Start the recording

b. 100 ms at -60 mV

c. 200 ms at +50 mV

d. 200 ms at -60 mV

e. Stop the recording

Finally, the current amplitude elicited by the (in this example depolarizing) voltage pulse is sent to the log window, and

```
    Robo2.Log(" ");
    signal = Robo2.MAXIMUM
    Robo2.Log("Response to reference voltage pulse to " + voltage_et + " mV = " + signal + " nA");
    Robo2.Log(" ");
```

the signal is compared to the user-defined limits minResponse and maxResponse. If the elicited signal is positive (outward current), the condition

if (signal < minResponse || signal > maxResponse)

should be used.

```
    if (signal < minResponse || signal > maxResponse)
    {
        Robo2.Log("Response (" + signal + " nA) not in Range (" + minResponse + ", " + maxResponse + "), go to next oocyte");
        Robo2.Wait(2);

        Robo2.ValvePumpOff();
        Robo2.CloseAllValves();
        Robo2.WastePumpOff();
        continue;
    }
```

If the expected signal is negative current (inward current) the operators > and < have to be exchanged.

Voltage Protocols (IV-dependencies)

Although voltage step protocols can be defined line by line with the script language, voltage protocols should be generated by the built-in graphical user interface (GUI). Please refer to the Roboocyte2 manual for details.

4 Using the Gilson Liquid Handler

Commands controlling the Gilson liquid handler can be separated in movement commands, commands controlling the peristaltic pump connected to the Gilson liquid handler, and commands controlling the magnetic valves of the Gilson's transfer ports. If you are using the liquid handler, scripts have to be modified, using Gilson commands instead of Roboflow commands.

Important note: Racks (and corresponding slots) used in a specific script have to be defined and selected by means of the Liquid Configuration feature of the Roboocyte software. Otherwise the script will not work.

4.1 Movement Commands

Gilson.	**Parameter(s)**	**Action**
MoveToTube(**x,y**)	int: **x** = slot int: **y**= tube	moves the probe to tube **y** in slot **x** (1 .. 5); the number of available tubes depends on the type of rack
MoveToPort(**x**)	int: port number (1 or 2)	moves probe to transfer ports **x** (1 or 2)
MoveToRinse()	---	moves probe to the rinse station
MoveToDrain()	---	moves probe to the drain station
MoveUp()	---	moves the Gilson probe up
MoveHome()	---	moves all to home position
Reset()	---	resets the Gilson

The "movement commands" move the Gilson probe (needle) in x, y, and z direction either to tubes, transfer ports, rinse station or drain station or just up (z-axis) or home (x = y = z = 0). The Reset command can be used to "reset" the machine, e.g. after some kind of malfunction, and is usually not used in a script.

4.2 Peristaltic Pump Commands

PumpBackward(**x**)	int: **x** = speed	starts the Gilson peristaltic pump in backward direction with speed **x**
PumpForward(**x**)	int: **x** = speed	starts the Gilson peristaltic pump in forward direction with speed **x**
Gilson.PumpStop()	---	stops the Gilson peristaltic pump

Whenever the Gilson liquid handler is used, the Minipulse peristaltic pump will be used. This pump will be then controlled by the Roboocyte software with the script commands listed above.

4.3 Transfer Port Valve Commands

Valve1On()	---	opens valve 1 from transfer port
Valve1Off()	---	closes valve 1 from transfer port
Valve2On()	---	opens valve 2 from transfer port
Valve2Off()	---	closes valve 2 from transfer port

The transfer ports offer you the option to work with larger solution volumes, e.g. for often used buffers or compounds. Using the transfer ports would mean moving the needle to the transfer port with the respective movement commands, opening the respective valves with the commands listed above, and finally switching the Minipulse pump on to aspirate solution from the transfer port.

4.4 Examples

Scripts that were written for using the Roboflow can be easily modified to work with the Gilson liquid handler instead. The following examples demonstrate how Roboflow solution exchange commands correspond to those of the Gilson liquid handler.

1. Opening a Roboflow valve corresponds to moving the Gilson needle into a tube or transfer port.

2. Turning on the Roboflow valve pump corresponds to starting the Minipulse pump in backward direction.

4.4.1 Differences between Roboflow and Gilson

1. Instead of using a manifold the Gilson liquid handler connects the different solution reservoirs and the measuring head with a single tubing. Therefore, an air gap has to be formed between two different solutions in the tubing to avoid mixing during transport from the probe to measuring head.

2. When using Roboflow, there is a delay of 2 to 3 s between the solution exchange command in the script and the arrival of the solution in the well (i.e. at the oocyte) when using pump speed 5000.

When using the Gilson liquid handler, this delay is much larger due to the tubing length between Gilson probe and measuring head. Depending on tubing length and the selected pump speed of the Minipulse pump, this delay can be tens of seconds and has to be considered in the design of the script.

4.4.2 Solution Exchange - Roboflow vs. Gilson

Defining a solution exchange sequence where solution1 runs for 5 seconds, followed by solution 2 for 10 seconds and finally solution 1 for 10 seconds will be realized by the following sequences of script commands.

The following table shows how Gilson commands correspond to Roboflow commands.

	Roboflow	**Gilson**
	Robo2.OpenValve(**1**);	Gilson.MoveToTube(1, **1**);
	Robo2.ValvePumpOn(**5000**);	Gilson.PumpBackward(**2600**);
	Robo2.Wait(**5**);	Robo2.Wait(**5**);
solution separating air gap (Gilson only)		Gilson.PumpStop(); Gilson.MoveUp(); Gilson.PumpBackward(**2600**); Robo2.WaitMilliSec(**600**); Gilson.PumpStop();
	Robo2.OpenValve(**2**);	Gilson.MoveToTube(1, **2**); Gilson.PumpBackward(**2600**);
	Robo2.Wait(**10**);	Robo2.Wait(**10**);
solution separating air gap		Gilson.PumpStop(); Gilson.MoveUp();

		Gilson.PumpBackward(**2600**); Robo2.WaitMilliSec(**600**); Gilson.PumpStop();
	Robo2.OpenValve(**1**);	Gilson.MoveToTube(1, **1**); Gilson.PumpBackward(**2600**);
	Robo2.Wait(**10**);	Robo2.Wait(**10**);

Same sequence from above listed sequentially.

Roboflow		**Gilson**
Robo2.OpenValve(**1**); Robo2.ValvePumpOn(**5000**); Robo2.Wait(**5**); Robo2.OpenValve(**2**); Robo2.Wait(**10**); Robo2.OpenValve(**1**); Robo2.Wait(**10**);		Gilson.MoveToTube(1, **1**); Gilson.PumpBackward(**2600**); Robo2.Wait(**5**); Gilson.PumpStop(); Gilson.MoveUp(); Gilson.PumpBackward(**2600**); Robo2.WaitMilliSec(**600**); Gilson.PumpStop(); Gilson.MoveToTube(1, **2**); Gilson.PumpBackward(**2600**); Robo2.Wait(**10**); Gilson.PumpStop(); Gilson.MoveUp(); Gilson.PumpBackward(**2600**); Robo2.WaitMilliSec(**600**); Gilson.PumpStop(); Gilson.MoveToTube(1, **1**); Gilson.PumpBackward(**2600**); Robo2.Wait(**10**);

Synchronizing this sequence with a recording is straightforward when using the Roboflow; put the command **Robo2.StartRecording();** and **Robo2.StopRecording();** before and after the sequence of commands, respectively.

4.4.3 Handling of the Gilson Delay

When using the Gilson liquid handler, the significant delay between switching to a respective solution and its arrival at the oocyte has to be considered. There are different ways to include this delay, but the easiest way is to add an additional wait command **Robo2.Wait(delay);** directly before the **Robo2.StopRecording();** command. The delay time depends on the used Minipulse pump speed and the length of the tubing between Gilson needle and measuring head. Therefore, delay time has to be determined experimentally beforehand.

Usually, buffer solution (solution 1) is already running before starting a solution exchange protocol. Therefore, the sequence after the **Robo2.StartRecording();** command starts with the first wait command **Robo2.Wait(5);.**

In addition, the respective transmit commands have to be included in order to send the appropriate compound and concentration information to the database.

Likewise, ROIs have to be defined in a way that the response of interest will be localized within the ROI.

Roboflow	**Gilson**
-->Robo2.OpenValve(**1**);	**-->**Gilson.MoveToTube(1, **1**);
-->Robo2.ValvePumpOn(**5000**);	**-->**Gilson.PumpBackward(**2600**);
Robo2.SetBaselineROI(**2,5**);	Robo2.SetBaselineROI(**2+delay, 4+delay**);
Robo2.SetAnalysisROI(**5, 15**);	Robo2.SetAnalysisROI(**5+delay, 15+delay**);
Robo2.StartRecording();	**Robo2.StartRecording();**
Robo2.TransmitRecording(REC_TAG_COMPOUND);	Robo2.TransmitRecording(REC_TAG_COMPOUND);
Robo2.TransmitCompoundValve(**2**);	Robo2.TransmitCompoundGilson(1, **2**);
Robo2.Wait(**5**);	Robo2.Wait(**5**);
Robo2.OpenValve(**2**);	Gilson.PumpStop();
Robo2.Wait(**10**);	Gilson.MoveUp();
Robo2.OpenValve(**1**);	Gilson.PumpBackward(**2600**);
Robo2.Wait(**10**);	Robo2.WaitMilliSec(**600**);
Robo2.StopRecording();	Gilson.PumpStop();
	Gilson.MoveToTube(1, **2**);
	Gilson.PumpBackward(**2600**);
	Robo2.Wait(**10**);
	Gilson.PumpStop();
	Gilson.MoveUp();
	Gilson.PumpBackward(**2600**);
	Robo2.WaitMilliSec(**600**);
	Gilson.PumpStop();
	Gilson.MoveToTube(1, **1**);
	Gilson.PumpBackward(**2600**);
	Robo2.Wait(**10**);
	Robo2.Wait(delay)
	Robo2.StopRecording();

Alternatively, you can work with the script commands **Robo2.StartTimer();** and **Robo2.WaitForTimer();**