



# **UNIVERSAL ROBOTS**

## **PolyScope Manual**

Version 1.8  
August 1, 2013

The information contained herein is the property of Universal Robots A/S and shall not be reproduced in whole or in part without prior written approval of Universal Robots A/S. The information herein is subject to change without notice and should not be construed as a commitment by Universal Robots A/S. This manual is periodically reviewed and revised.

Universal Robots A/S assumes no responsibility for any errors or omissions in this document.

Copyright ©2012 by Universal Robots A/S

The Universal Robots logo is a registered trademark of Universal Robots A/S.

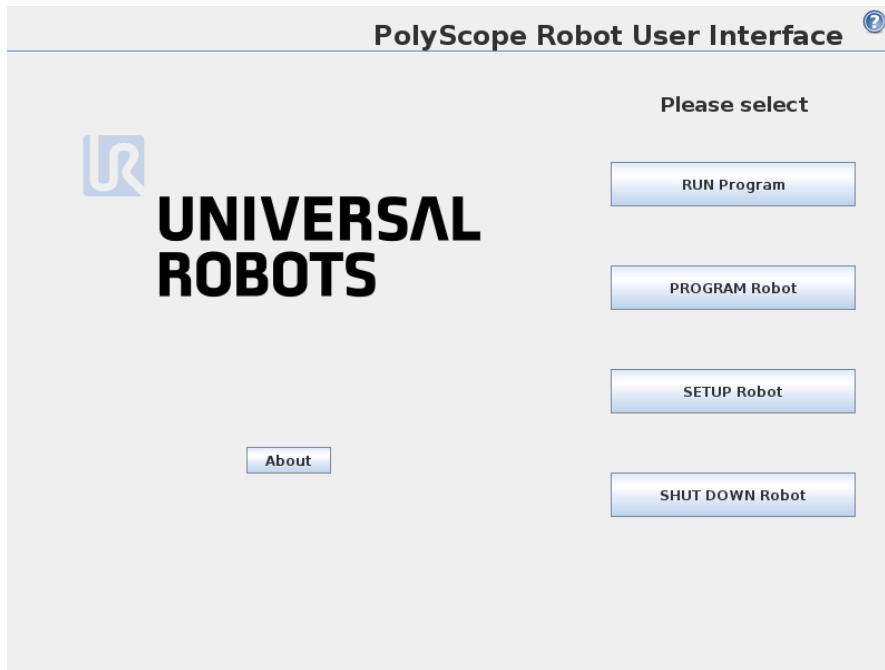
# Contents

1	Introduction . . . . .	5
1.1	Welcome Screen . . . . .	6
1.2	Initialization Screen . . . . .	7
2	On-screen Editors . . . . .	8
2.1	On-screen Keypad . . . . .	8
2.2	On-screen Keyboard . . . . .	9
2.3	On-screen Expression Editor . . . . .	9
2.4	Pose Editor Screen . . . . .	10
3	Robot Control . . . . .	11
3.1	Move Tab . . . . .	12
3.2	I/O Tab . . . . .	14
3.3	MODBUS client I/O . . . . .	14
3.4	AutoMove Tab . . . . .	16
3.5	Installation → Load/Save . . . . .	17
3.6	Installation → TCP Position . . . . .	17
3.7	Installation → Mounting . . . . .	18
3.8	Installation → I/O Setup . . . . .	19
3.9	Installation → Variables . . . . .	20
3.10	Installation → Default Program . . . . .	21
3.11	MODBUS client I/O Setup . . . . .	21
3.12	Features . . . . .	24
3.13	Log Tab . . . . .	27
3.14	Load Screen . . . . .	28
3.15	Run Tab . . . . .	29
4	Programming . . . . .	30
4.1	Program → New Program . . . . .	31
4.2	Program Tab . . . . .	31
4.3	Program → Command Tab, <Empty> . . . . .	32
4.4	Program → Command Tab, Move . . . . .	33
4.5	Program → Command Tab, Fixed Waypoint . . . . .	35
4.6	Setting the waypoint . . . . .	35
4.7	Program → Command Tab, Relative Waypoint . . . . .	37
4.8	Program → Command Tab, Variable Waypoint . . . . .	37
4.9	Program → Command Tab, Wait . . . . .	38
4.10	Program → Command Tab, Action . . . . .	39
4.11	Program → Command Tab, Popup . . . . .	39
4.12	Program → Command Tab, Halt . . . . .	40
4.13	Program → Command Tab, Comment . . . . .	40
4.14	Program → Command Tab, Folder . . . . .	41
4.15	Program → Command Tab, Loop . . . . .	41
4.16	Program → Command Tab, SubProgram . . . . .	42
4.17	Program → Command Tab, Assignment . . . . .	43
4.18	Program → Command Tab, If . . . . .	43

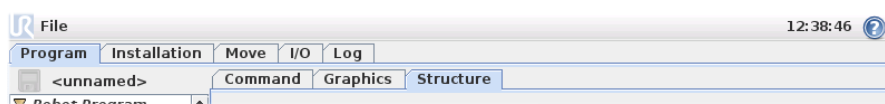
4.19	Program → Command Tab, Script . . . . .	44
4.20	Program → Command Tab, Event . . . . .	44
4.21	Program → Command Tab, Thread . . . . .	45
4.22	Program → Command Tab, Pattern . . . . .	45
4.23	Program → Command Tab, Force . . . . .	47
4.24	Program → Command Tab, Pallet . . . . .	49
4.25	Program → Command Tab, Seek . . . . .	50
4.26	Program → Command Tab, Suppress . . . . .	53
4.27	Program → Graphics Tab . . . . .	54
4.28	Program → Structure Tab . . . . .	55
4.29	Program → Variables Tab . . . . .	56
4.30	Program → Command Tab, Variables Initialization . . . . .	56
5	Setup . . . . .	57
5.1	Setup Screen . . . . .	57
5.2	Setup Screen → Initialize . . . . .	58
5.3	Setup Screen → Language Select . . . . .	58
5.4	Setup Screen → Update . . . . .	59
5.5	Setup Screen → Password . . . . .	60
5.6	Setup Screen → Calibrate Touch Screen . . . . .	60
5.7	Setup Screen → Network . . . . .	61
5.8	Setup Screen → Time . . . . .	61

## 1 Introduction

PolyScope is the graphical user interface (GUI) which lets you operate the robot, run existing robot programs or easily create new ones. PolyScope runs on the touch sensitive screen attached to the control box. To calibrate the touch screen, read section 5.6.



The picture above shows the Welcome Screen. The bluish areas of the screen are buttons that can be pressed by pressing a finger or the backside of a pen against the screen. PolyScope has a hierarchical structure of screens. In the programming environment, the screens are arranged in *tabs*, for easy access on the screens.



In this example, the `Program` tab is selected at the top level, and under that the `Structure` tab is selected. The `Program` tab holds information related to the currently loaded program. If the `Move` tab is selected, the screen changes to the 'Move' screen, from where the robot can be moved. Similarly, by selecting the `I/O` tab, the current state of the electrical I/O can be monitored and changed.

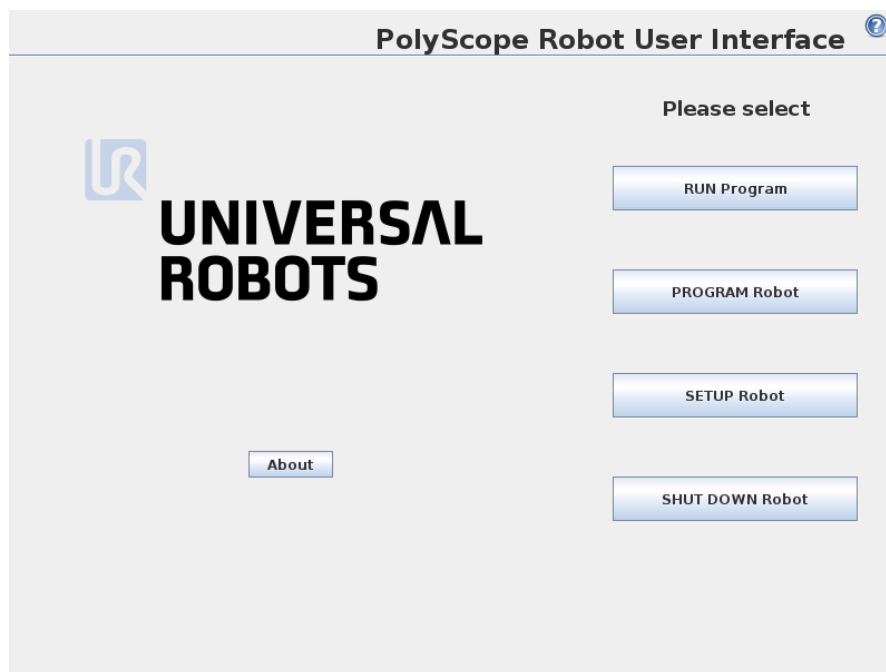
It is possible to connect a mouse and a keyboard to the controller box or the teach pendant; however, this is not required. Almost all text fields are touch-enabled, so touching them launches an on-screen keypad or keyboard. Non-touchable text fields have an editor icon next to them that launches the associated input editor.



The icons of the on-screen keypad, keyboard or expression editor are shown above.

The various screens of PolyScope are described in the following sections.

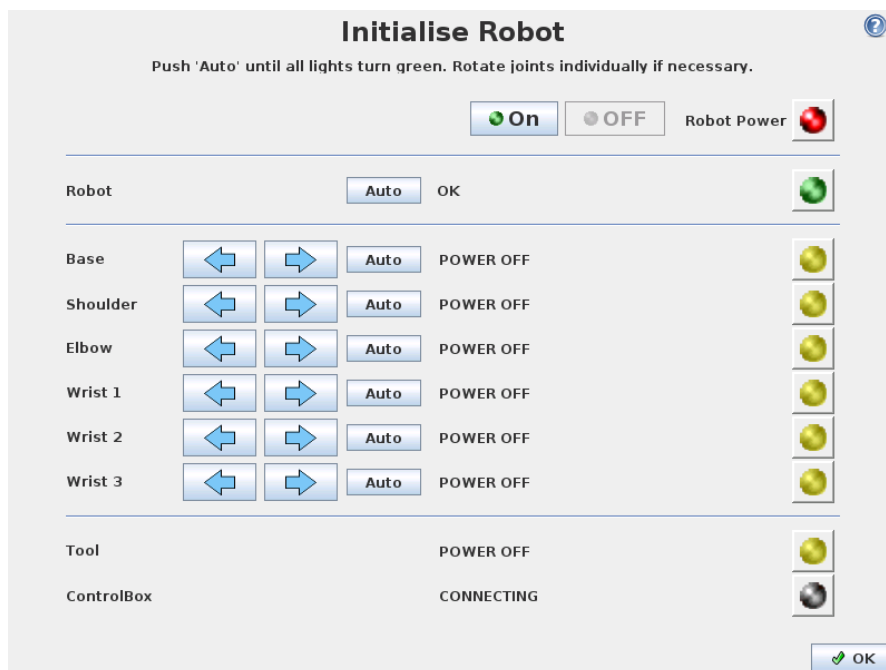
## 1.1 Welcome Screen



After booting up the controller PC, the welcome screen is shown. The screen offers the following options:

- **RUN Program:** Choose a program to run. This is the simplest way to operate the robot, but requires a suitable program to have already been produced.
- **PROGRAM Robot:** Change a program, or create a new program.
- **SETUP Robot:** Set passwords, upgrade software, request support, calibrate the touch screen, etc.
- **SHUT DOWN Robot:** Shuts down the Controller PC and powers off the robot.

## 1.2 Initialization Screen



On this screen you control the initialization of the robot. When turned on, the robot needs to find the positions of each joint. To get the joint positions, the robot needs to move each joint.



### WARNING:

1. The force protection during initialization can be higher than  $150N$ . The maximum force is only reduced by torque limits in the joints. Stay outside the robot workspace when the robot arm is moving.

### Status LEDs

The status LEDs give an indication of the joints' running state.

- A bright red LED tells that the robot is currently in a stopped state where the reasons can be several.
- A bright yellow LED indicates that the joint is running, but doesn't know its presents position and needs homing.
- Finally a green LED indicates that the joint is running correctly and is ready to execute.

All the LEDs have to be green in order for the robot to operate normally.

### Manual motion (By hand)

When the joints are *Ready* and the "Teach" button on the back of the screen is pressed, the joint modes change to *Backdrive*. In this mode, the joints will release the brakes when motion is detected. This way, the robot can be moved out of a machine manually, before being started up. The brakes will reactive as soon as the button is released again.

### Auto movement (Auto Buttons)

Normally it is always advisable to use the auto buttons to move the individual joints until they reach a known state. In order to operate the button, you have to press on the Auto button, and keep it pressed.

The auto buttons can be pressed individually for each joint, or for the whole robot.



**CAUTION:**

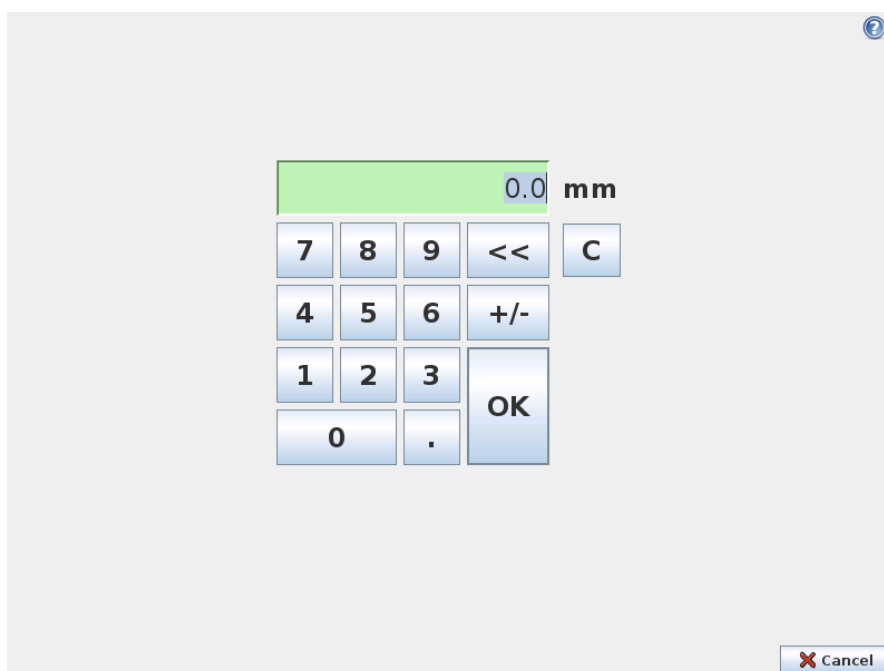
1. Great care should be taken if the robot is touching an obstacle or table, since driving the robot into the obstacle might damage a joint gearbox.

### Moving directly (Move Buttons)

In the case where a joint is in a position where there is a major risk that uncontrolled motion would cause damage to the robot or its surroundings, the operator can choose to home the robot manually for each joint. section 1.2.

## 2 On-screen Editors

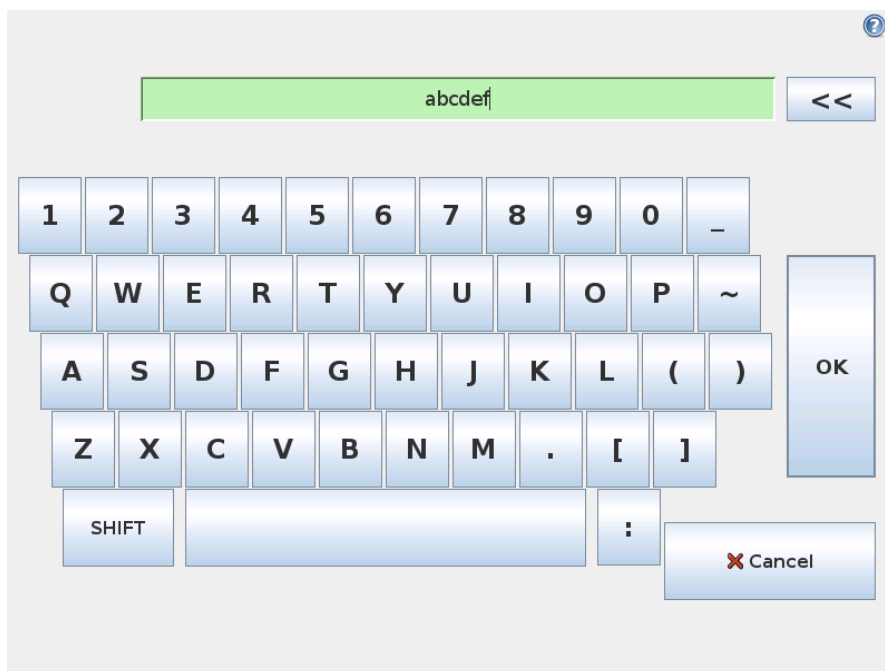
### 2.1 On-screen Keypad



Simple number typing and editing facilities. In many cases, the unit of the typed value is displayed next to the number.



## 2.2 On-screen Keyboard



Simple text typing and editing facilities. The `Shift` key can be used to get some additional special characters.

## 2.3 On-screen Expression Editor



While the expression itself is edited as text, the expression editor has a number of buttons and functions for inserting the special expression symbols, such as `*` for multiplication and `≤` for less than or equal to. The keyboard symbol button in the top right of the screen switches to text-editing of the expression. All defined variables can be found in the `Variable` selector, while the names of the input and output ports can be found in the `Input` and `Output` selectors. Some special functions are found in `Function`.

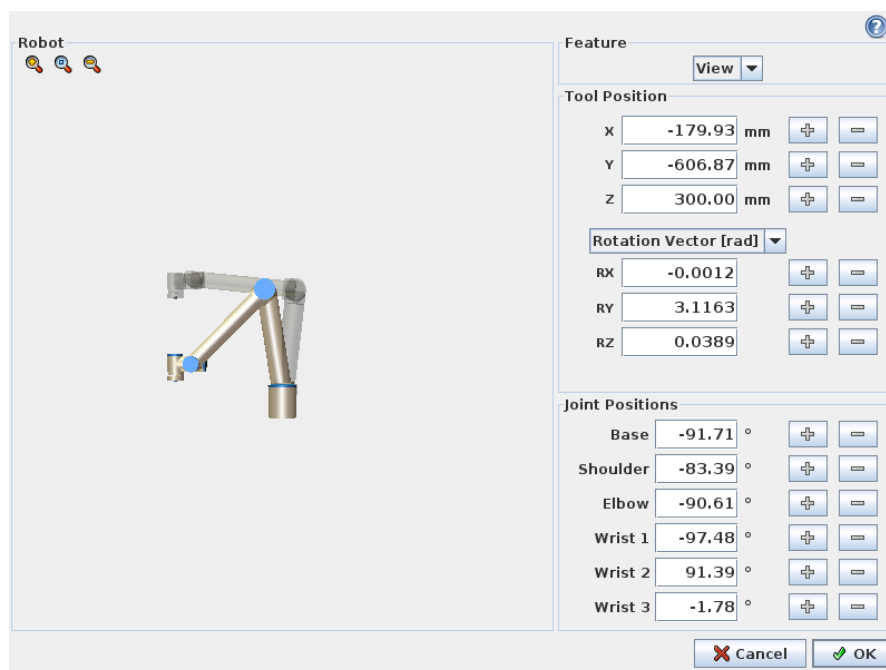
The expression is checked for grammatical errors when the **OK** button is pressed. The **Cancel** button leaves the screen, discarding all changes.

An expression can look like this:

```
digital_in[1]==True and analog_in[0]<0.5
```

## 2.4 Pose Editor Screen

On this screen you can specify target joint positions, or a target pose (position and orientation) of the robot tool. This screen is “offline” and does not control the physical robot directly.



### Robot

The current position of the robot and the specified new target position are shown in 3D graphics. The 3D drawing of the robot shows the current position of the robot, and the “shadow” of the robot shows the target position of the robot controlled by the specified values on the right hand side of the screen. Push the magnifying glass icons to zoom in/out or drag a finger across to change the view.

### Feature and tool position

At the top right part of the screen, the feature selector can be found. The feature selector defines which feature to control the robot relative to, while below it, the boxes display the full coordinate value for the tool relative to the selected feature. X, Y and Z control the position of the tool, while RX, RY and RZ control the orientation of the tool.

Use the drop down menu above the RX, RY and RZ boxes to choose the orientation representation. Available types are:

- **Rotation Vector [rad]** The orientation is given as a *rotation vector*. The length of the axis is the angle to be rotated in radians, and the vector itself gives the axis about which to rotate. This is the default setting.

- **Rotation Vector** [ $^{\circ}$ ] The orientation is given as a *rotation vector*, where the length of the vector is the angle to be rotated in degrees.
- **RPY** [**rad**] *Roll, pitch and yaw (RPY)* angles, where the angles are in radians. The RPY-rotation matrix (X, Y', Z" rotation) is given by:

$$R_{rpy}(\gamma, \beta, \alpha) = R_Z(\alpha) \cdot R_Y(\beta) \cdot R_X(\gamma)$$

- **RPY** [ $^{\circ}$ ] *Roll, pitch and yaw (RPY)* angles, where angles are in degrees.

Values can be edited by clicking on the coordinate. Clicking on the + or – buttons just to the right of a box allows you to add or subtract an amount to/from the current value. Pressing and holding down a button will directly increase/decrease the value. The longer the button is down, the larger the increase/decrease will be.

### Joint Positions

Allows the individual joint positions to be specified directly. Each joint position can have a value in the range from  $-360^{\circ}$  to  $+360^{\circ}$ , which are the *joint limits*. Values can be edited by clicking on the joint position. Clicking on the + or – buttons just to the right of a box allows you to add or subtract an amount to/from the current value. Pressing and holding down a button will directly increase/decrease the value. The longer the button is down, the larger the increase/decrease will be.

### OK button

If this screen was activated from the `Move` tab (see 3.1), clicking the `OK` button will return to the `Move` tab, where the robot will move to the specified target. If the last specified value was a tool coordinate, the robot will move to the target position using the *MoveL* movement type, while the robot will move to the target position using the *MoveJ* movement type, if a joint position was specified last. The different movement types are described in section 4.4.

### Cancel button

Clicking the `Cancel` button leaves the screen discarding all changes.

## 3 Robot Control

Use the “play”, “Pause”, “Stop” and “Step” buttons found at the bottom of the screen to start and stop robot programs. Also there is a speed slider which can slow down the speed of the program during verification of the robot program and movement.

**DANGER:**

1. Make sure to stay outside the robot workspace when the "Play" button is pressed. The movement you programmed may be different than expected.

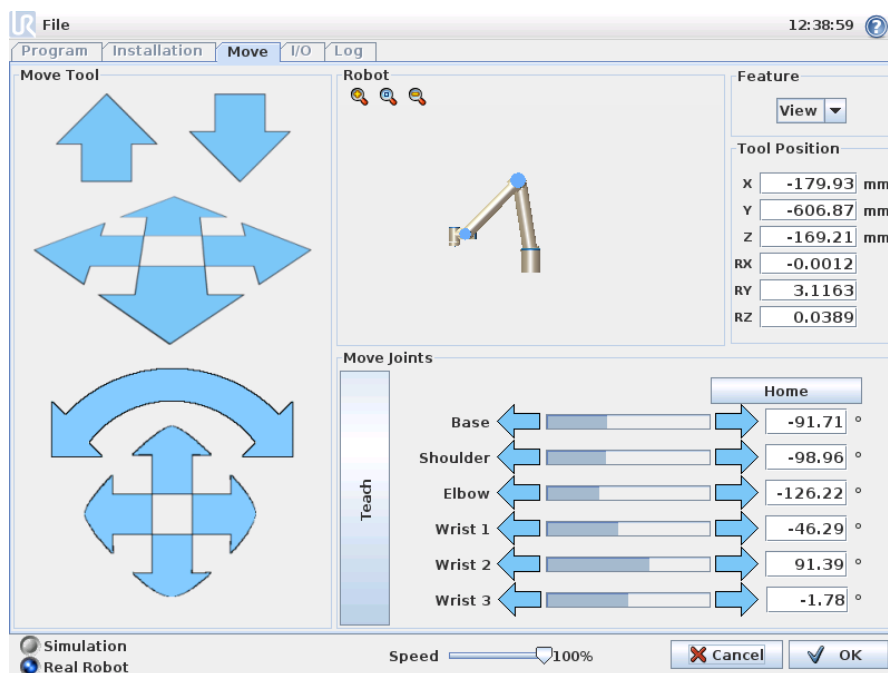


2. Make sure to stay outside the robot workspace when the "Step" button is pressed. The function of the "Step" button can be difficult to understand. Only use it when it is absolutely necessary.

3. Make sure to always test your program by reducing the speed with the speed slider. Logic programming errors made by the integrator might cause unexpected movements of the robot.

**3.1 Move Tab**

On this screen you can always move (jog) the robot directly, either by translating/rotating the robot tool, or by moving robot joints individually.



**Robot**

The current position of the robot is shown in 3D graphics. Push the magnifying glass icons to zoom in/out or drag a finger across to change the view. To get the best feel for controlling the robot, select the "View" feature and rotate the viewing angle of the 3D drawing to match your view of the real robot.

**Feature and tool position**

At the top right part of the screen, the feature selector can be found. The features selector defines which feature to control the robot relative to, while below it, the boxes display the full coordinate value for the tool relative to the selected feature.

Values can be edited manually by clicking on the coordinate or the joint position. This will take you to the pose editor screen (see section 2.4) where you can specify a target position and orientation for the tool or target joint positions.

### Move Tool

- Holding down a `translate` arrow (top) will move the tool-tip of the robot in the direction indicated.
- Holding down a `rotate` arrow (button) will change the orientation of the robot tool in the indicated direction. The point of rotation is the TCP, drawn as a small blue ball.

Note: *Release the button to stop the motion at any time!*

### Move Joints

Allows the individual joints to be controlled directly. Each joint can move from  $-360^\circ$  to  $+360^\circ$ , which are the *joint limits* illustrated by the horizontal bar for each joint. If a joint reaches its joint limit, it cannot be driven any further away from  $0^\circ$ .

### Teach

While the 'Teach' button is held down, it is possible to physically grab the robot and pull it to where you want it to be. If the gravity setting (see 3.7) in the `Setup` tab is wrong, or the robot carries a heavy load, the robot might start moving (falling) when the 'Teach' button is pressed. In that case, just release the 'Teach' button again.

#### **WARNING:**

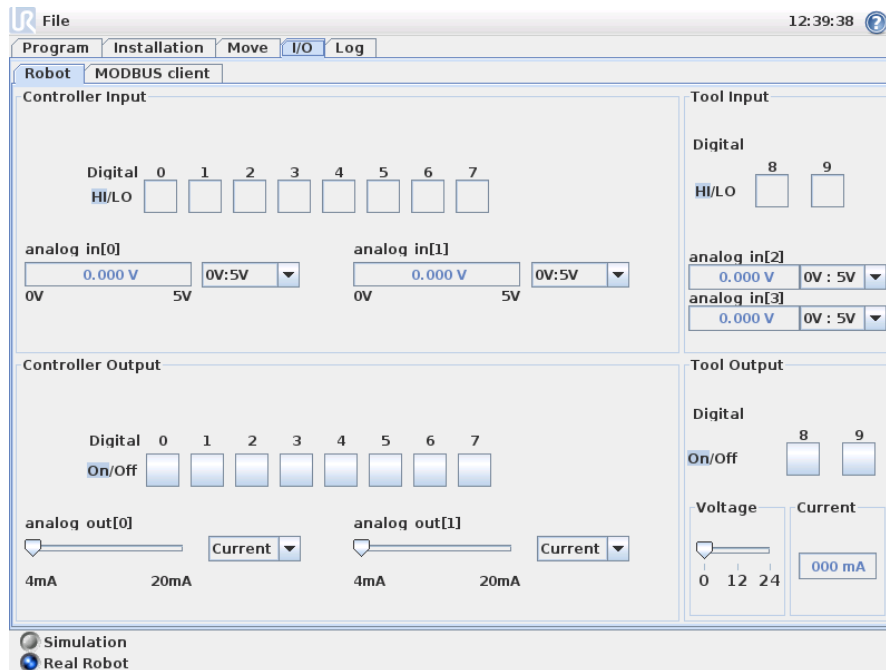
1. Make sure to use the correct installation settings (e.g. Robot mounting angle, weight in TCP, TCP offset). Save and load the installation files along with the program.

2. Make sure that the TCP settings and the robot mounting settings are set correctly before operating the "Teach" button. If these settings are not correct the robot will move when the "teach" button is activated.

3. The teach function (Impedance/backdrive) shall only be used in installations where the risk assessment allows it. Tools and obstacles shall not have sharp edges or pinch points. Make sure that all personnel remain outside the reach of the robot.



### 3.2 I/O Tab



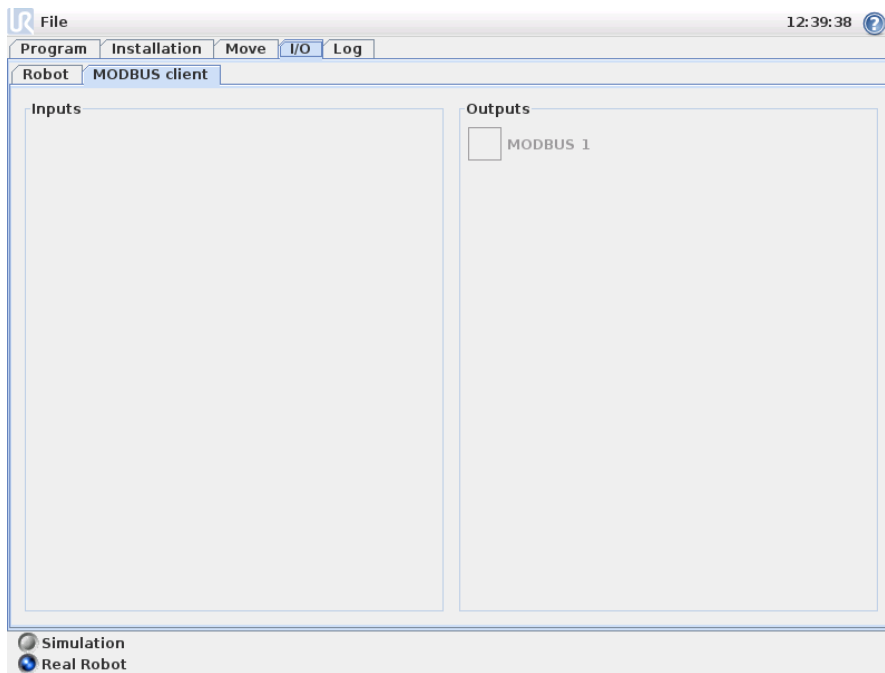
On this screen you can always monitor and set the live I/O signals from/to the robot. The screen displays the current state of the I/O, including during program execution. If anything is changed during program execution, the program will stop. At program stop, all output signals will retain their states. The screen is updated at only 10Hz, so a very fast signal might not display properly.

The electrical details of the signals are described in the user manual.

**Analog Range Settings** The analog output can be set to either current (4-20mA) or voltage (0-10V) output. The analog input ranges adjusted to be from (-10-10V) to (0-5V). The settings will be remembered for eventual later restarts of the robot controller when a program is saved.

### 3.3 MODBUS client I/O

Here, the digital MODBUS client I/O signals as set up in the installation are shown. If the signal connection is lost, the corresponding entry on this screen is disabled.



### Inputs

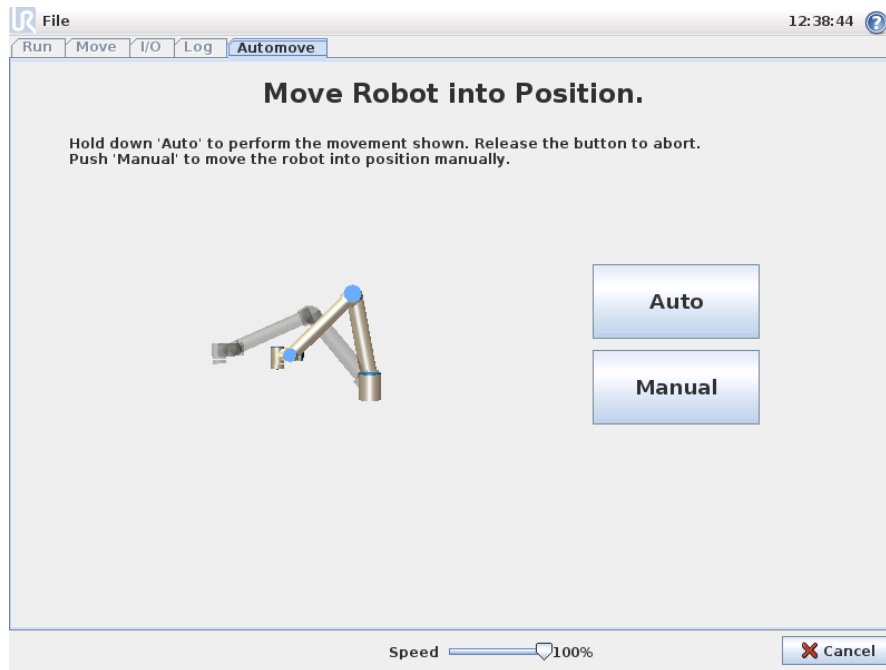
View the state of digital MODBUS client inputs.

### Outputs

View and toggle the state of digital MODBUS client outputs. A signal can only be toggled if the choice for I/O tab control (described in 3.8) allows it.

### 3.4 AutoMove Tab

The AutoMove tab is used when the robot has to move to a specific position in its workspace. Examples are when the robot has to move to the start position of a program before running it, or when moving to a waypoint while modifying a program.



#### Animation

The animation shows the movement the robot is about to perform. Compare the animation with the position of the real robot and make sure that robot can safely perform the movement without hitting any obstacles.



#### CAUTION:

1. The automove function moves in joint space, not in linear (cartesian) space. Collision might damage robot or equipment.

#### Auto

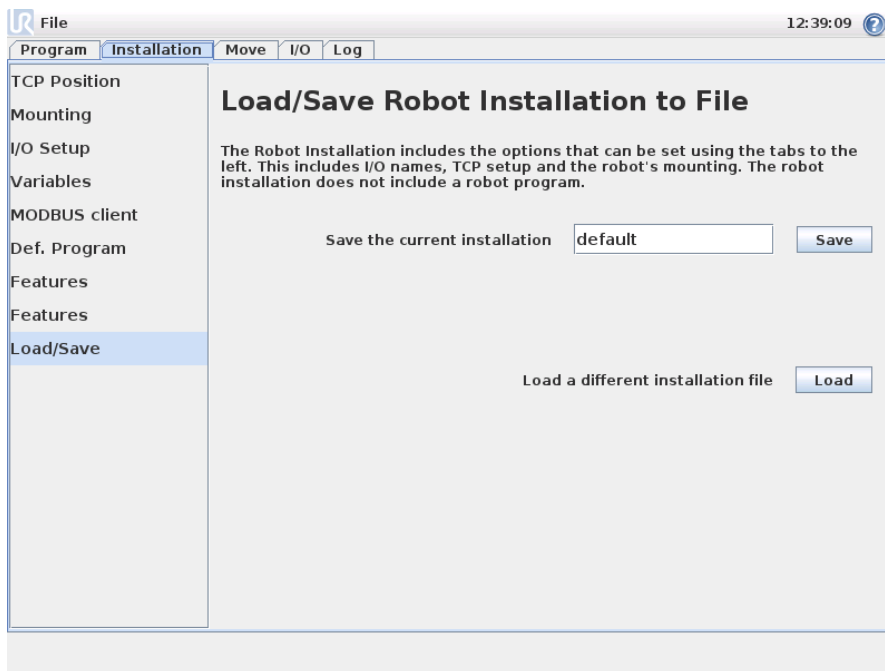
Hold down the `Auto` button to move the robot as shown in the animation. Note: *Release the button to stop the motion at any time!*

#### Manual

Pushing the `Manual` button will take you to the `MoveTab` where the robot can be moved manually. This is only needed if the movement in the animation is not preferable.

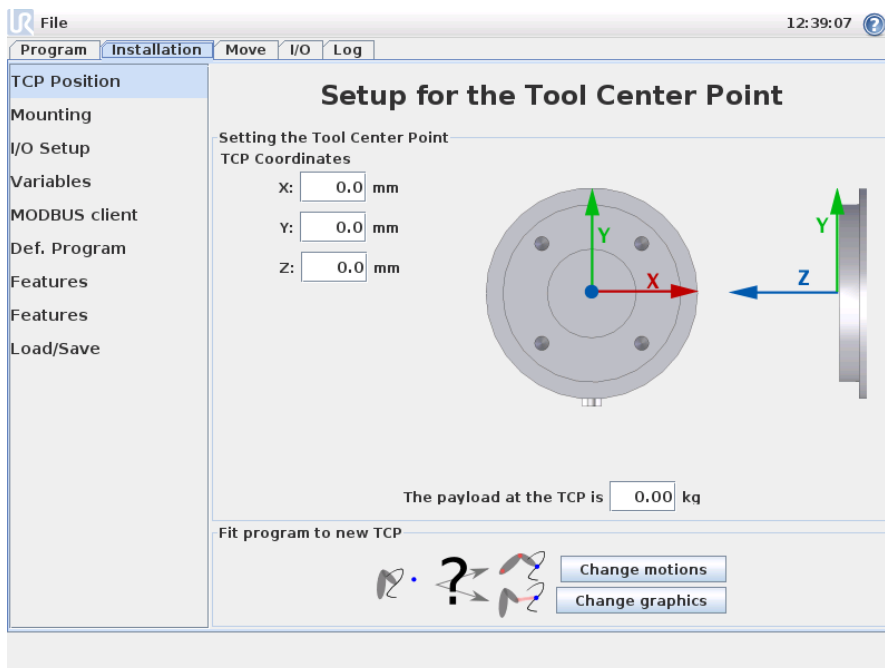


### 3.5 Installation → Load/Save



The installation covers aspects of how the robot is placed in its working environment, both mechanical mounting of the robot, and electrical connections to other equipment. These settings can be set using the various screens under the *Installation* tab. It is possible to have more than one installation file for the robot. Programs created will use the active installation, and will load this installation automatically when used. Any changes to an installation needs to be saved to be preserved after power down. Saving an installation can be done either by pressing the *Save* button or by saving a program using the installation.

### 3.6 Installation → TCP Position



The *Tool Center Point* (TCP) is the point at the end of the robot arm that gives a characteristic point on the robot's tool. When the robot moves linearly, it is this

point that moves in a straight line. It is also the motion of the TCP that is visualized on the graphics tab. The TCP is given relative to the center of the tool output flange, as indicated on the on-screen graphics.



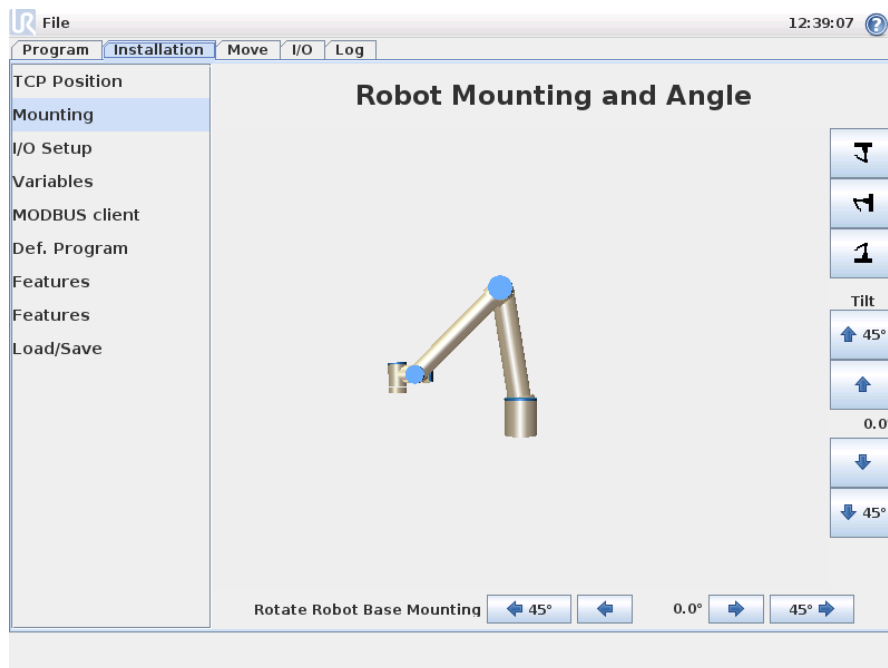
**WARNING:**

1. Make sure to use the correct installation settings. Save and load the installation files along with the program.

The two buttons on the bottom of the screen are relevant when the TCP is changed.

- **Change Motions** recalculates all positions in the robot program to fit the new TCP. This is relevant when the shape or size of the tools has been changed.
- **Change Graphics** redraws the graphics of the program to fit the new TCP. This is relevant when the TCP has been changed without any physical changes to the tool.

**3.7 Installation → Mounting**



Here the mounting of the robot can be specified. This serves two purposes:

1. Making the robot look right on the screen.
2. Telling the controller about the direction of gravity.

The controller uses an advanced dynamics model to give the robot smooth and precise motions, and to make the robot hold itself when backdriven. For this reason, it is important that the mounting of the robot is set correctly.

**WARNING:**

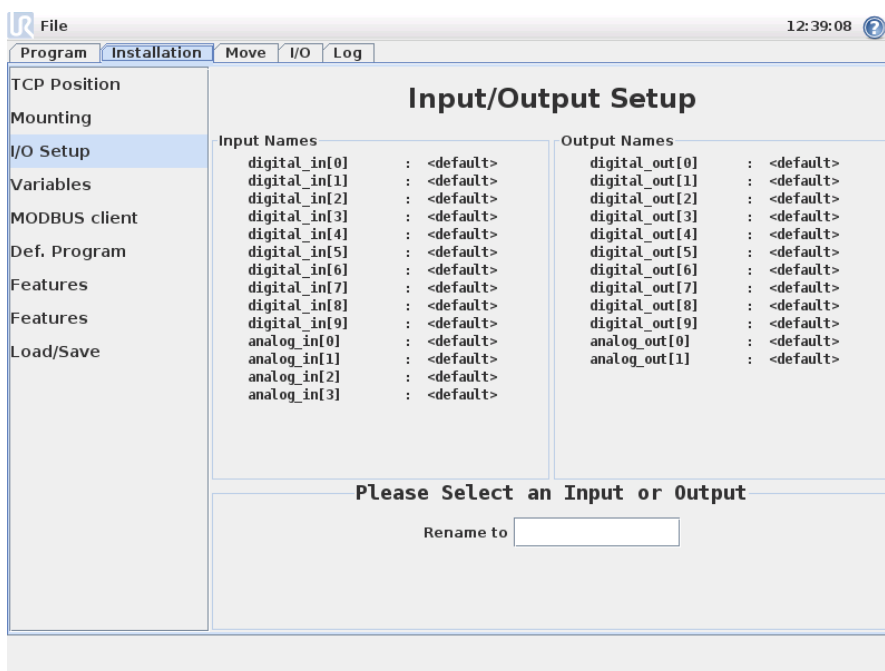
1. Failure to set the robot mounting correctly might result in frequent security stops, and/or a possibility that the robot will move when the teach button is pressed.

The default is that the robot is mounted on a flat table or floor, in which case no change is needed on this screen. However, if the robot is *ceiling mounted*, *wall mounted* or mounted at an angle this can be adjusted using the push-buttons. The buttons on the right side of the screen are for setting the angle of the robot's mounting. The three top right side buttons set the angle to *ceiling* (180°), *wall* (90°), *floor* (0°). The `Tilt` buttons can be used to set an arbitrary angle. The buttons on the lower part of the screen are used to rotate the mounting of the robot to match the actual mounting.

**WARNING:**

1. Make sure to use the correct installation settings. Save and load the installation files along with the program.

### 3.8 Installation → I/O Setup

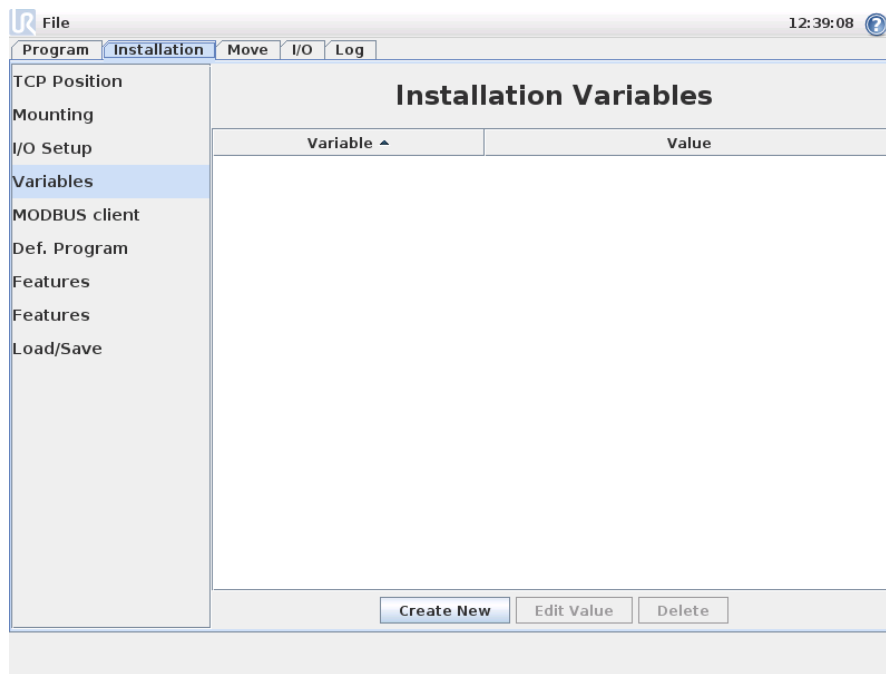


Input and output signals can be given names. This can make it easier to remember what the signal does when working with the robot. Select an I/O by clicking on it, and set the name using the on screen keyboard. You can set the name back by setting it to only blank characters.

When an output is selected, a few options are enabled. Using the check box, a default value for the output can set to either low or high. This means that the output will be set to this value when a program is not running. If the check box is not checked, the output will preserve its current state after a program ends. It is also possible to specify whether an output can be controlled on the

I/O tab (by either programmers, or both operators and programmers) or if it is only robot programs that may alter the output value.

### 3.9 Installation → Variables



Variables created here are called installation variables and can be used just like normal program variables. Installation variables are special because they keep their value even if a program is stopped and then started again, and when the robot is powered down and powered up again. The name and values are stored with the installation, so it is possible to use the same variable in multiple programs.

Pressing `Create New` will bring up a panel with a suggested name for the new variable. The name may be changed and its value may be entered by touching either text field. The `OK`-button can only be clicked if the new name is unused in this installation.

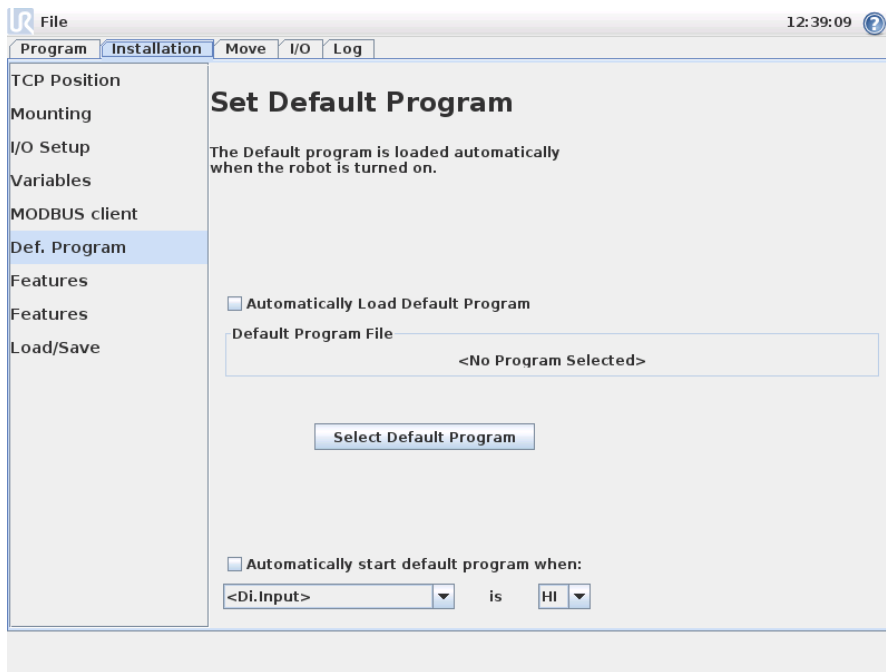
It is possible to change the value of an installation variable by highlighting the variable in the list and then clicking on `Edit Value`. This will bring up the edit panel, where a new value can be entered and confirmed with `OK`. Press `Cancel` to close the panel without updating the value. When the edit panel is shown, you can choose which variable to edit by selecting a new variable in the list above the panel.

To delete a variable, select it in the list, then click `Delete`.

After configuring the installation variables, the installation itself must be saved to keep this configuration, see 3.5.

If a program or an installation is loaded and one or more of the program variables have the same name as the installation variables, the user is presented with two options to resolve the issue: either use the installation variables of the same name instead of the program variable or have the conflicting variables renamed automatically.

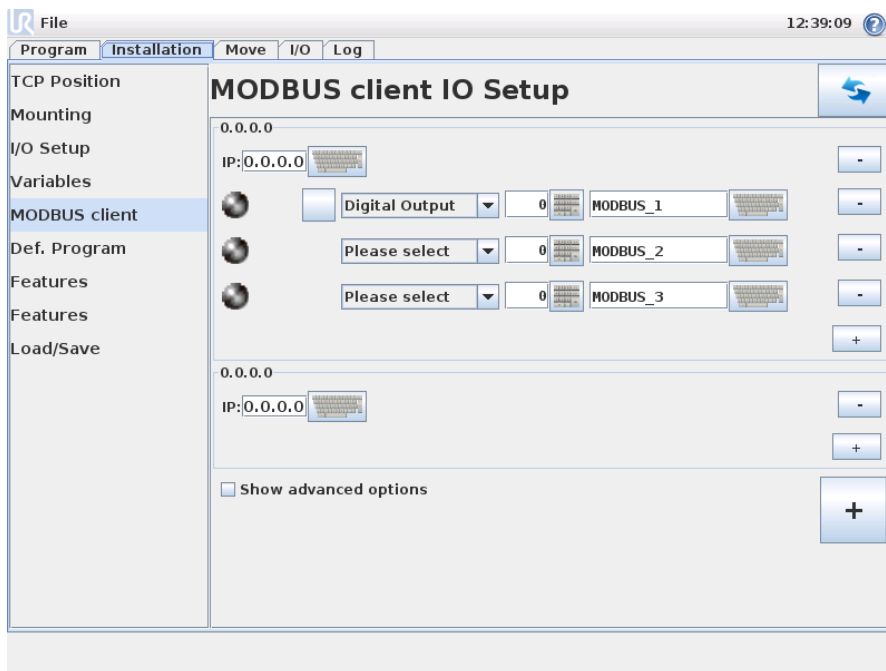
### 3.10 Installation → Default Program



The default program will be loaded when the control box is powered up.

### 3.11 MODBUS client I/O Setup

Here, the MODBUS client (master) signals can be set up. Connections to MODBUS servers (or slaves) on specified IP addresses can be created with input/output signals (registers or digital). Each signal has a unique name so it can be used in programs.



#### Refresh

Push this button to refreshes all MODBUS connections.

### Add unit

Push this button to add a new MODBUS unit.

### Delete unit

Push this button to delete the MODBUS unit and all signals on that unit.

### Set unit IP

Here the IP address of the MODBUS unit is shown. Press the button to change it.

### Add signal

Push this button to add a signal to the corresponding MODBUS unit.

### Delete signal

Push this button to delete a MODBUS signal from the corresponding MODBUS unit.

### Set signal type

Use this drop down menu to choose the signal type. Available types are:

- **Digital input:** A digital input (coil) is a one-bit quantity which is read from the MODBUS unit on the coil specified in the address field of the signal. Function code 0x02 (Read Discrete Inputs) is used.
- **Digital output:** A digital output (coil) is a one-bit quantity which can be set to either high or low. Before the value of this output has been set by the user, the value is read from the remote MODBUS unit. This means that function code 0x01 (Read Coils) is used. When the output has been set by a robot program or by pressing the "set signal value" button, the function code 0x05 (Write Single Coil) is used onwards.
- **Register input:** A register input is a 16-bit quantity read from the address specified in the address field. The function code 0x04 (Read Input Registers) is used.
- **Register output:** A register output is a 16-bit quantity which can be set by the user. Before the value of the register has been set, the value of it is read from the remote MODBUS unit. This means that function code 0x03 (Read Holding Registers) is used. When the output has been set by a robot program or by specifying a signal value in the "set signal value" field, function code 0x06 (Write Single Register) is used to set the value on the remote MODBUS unit.

### Set signal address

This field shows the address on the remote MODBUS server. Use the on-screen keypad to choose a different address. Valid addresses depends on the manufacturer and configuration of the remote MODBUS unit.

### Set signal name

Using the on-screen keyboard, the user can give the signal a name. This name is used when the signal is used in programs.

### Signal value

Here, the current value of the signal is shown. For register signals, the value is expressed as an unsigned integer. For output signals, the desired signal value can be set using the button. Again, for a register output, the value to write to the unit must be supplied as an unsigned integer.

### Signal connectivity status

This icon shows whether the signal can be properly read/written (green), or if the unit responds unexpected or is not reachable (gray). If a MODBUS exception response is received, the response code is displayed. The MODBUS-TCP Exception responses are:

- **E1 ILLEGAL FUNCTION (0x01):** The function code received in the query is not an allowable action for the server (or slave).
- **E2 ILLEGAL DATA ADDRESS (0x02):** The function code received in the query is not an allowable action for the server (or slave), check that the entered signal address corresponds to the setup of the remote MODBUS server.
- **E3 ILLEGAL DATA VALUE (0x03):** A value contained in the query data field is not an allowable value for server (or slave), check that the entered signal value is valid for the specified address on the remote MODBUS server.
- **E4 SLAVE DEVICE FAILURE (0x04):** An unrecoverable error occurred while the server (or slave) was attempting to perform the requested action.
- **E5 ACKNOWLEDGE (0x05):** Specialized use in conjunction with programming commands sent to the remote MODBUS unit.
- **E6 SLAVE DEVICE BUSY (0x06):** Specialized use in conjunction with programming commands sent to the remote MODBUS unit, the slave (server) is not able to respond now.

### Show Advanced Options

This check box shows/hides the advanced options for each signal.

### Advanced Options

- **Update Frequency:** This menu can be used to change the update frequency of the signal. This means the frequency with which requests are sent to the remote MODBUS unit for either reading or writing the signal value.
- **Slave Address:** This text field can be used to set a specific slave address for the requests corresponding to a specific signal. The value must be in the range 0-255 both included, and the default is 255. If you change this value, it is recommended to consult the manual of the remote MODBUS device to verify its functionality when changing slave address.

### 3.12 Features

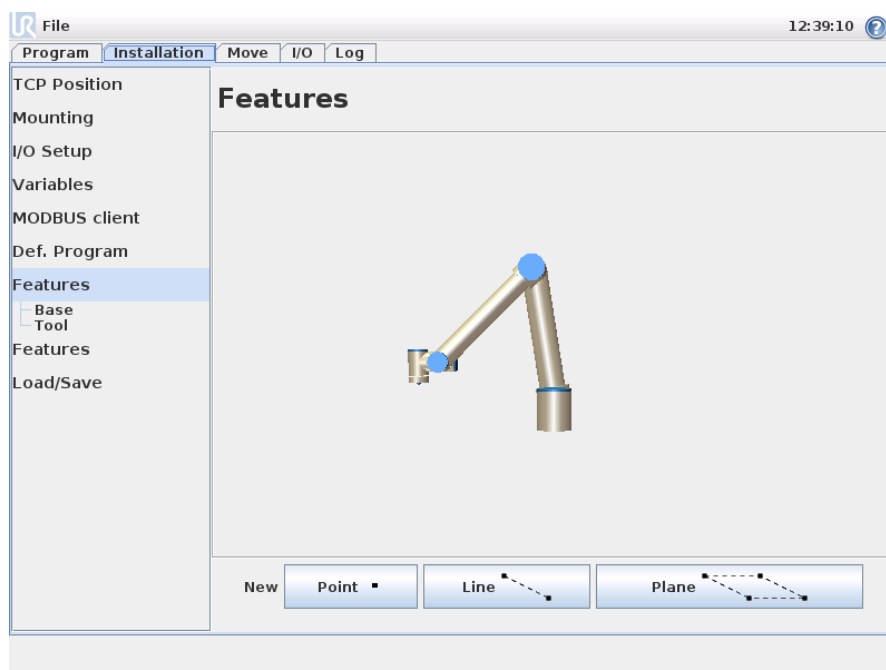
Customers that buy industrial robots generally want to be able to control or manipulate a robot, and to program the robot, relative to various objects and boundaries in the surroundings of the robot, such as machines, objects or blanks, fixtures, conveyers, pallets or vision systems. Traditionally, this is done by defining "frames" (coordinate systems) that relate the internal coordinate system of the robot (the base coordinate system) to the relevant object's coordinate system. Reference can both be made to "tool coordinates" and to "base coordinates" of the robot.

A problem with such frames is that a certain level of mathematical knowledge is required to be able to define such coordinate systems and also that it takes a considerable amount of time to do this, even for a person skilled in the art of robot programming and installation. Often this task involves the calculation of 4x4 matrices. Particularly, the representation of orientation is complicated for a person that lacks the required experience to understand this problem.

Questions often asked by customers are for instance:

- Will it be possible to move the robot 4 cm away from the claw of my computerised numerically controlled (CNC) machine?
- Is it possible to rotate the tool of the robot 45 degrees relative to the table?
- Can we make the robot move vertically downwards with the object, let the object loose, and then move the robot vertically upward again?

The meaning of such and similar questions is very straight forward to an average customer that intends to use a robot for instance at various stations in a production plant, and it may seem annoying and incomprehensible to the customer to be told that there may not be a simple answer to such - relevant - questions. There are several complicated reasons for this being the case, and in order to address these problems, Universal Robots has developed unique and simple ways for a customer to specify the location of various objects relative to the robot. Within a few steps, it is therefore possible to do exactly what was asked for in the above questions.





### **Rename**

This button makes it possible to rename a feature.

### **Delete**

This button deletes the selected feature and, if any, all sub-features.

### **Show Axes**

Choose whether the coordinate axes of the selected feature shall be visible on the 3D graphics. The choice applies on this screen and on the Move screen.

### **Joggable**

Select whether the selected feature shall be joggable. This determines whether the feature will appear in the feature menu on the Move screen.

### **Variable**

Select whether the selected feature can be used as a variable. If this option is selected a variable named the name of the feature succeeded by "\_var" will then be available when editing robot programs, and this variable can be assigned a new value in a program, which can then be used to control waypoints that depend on the value of a feature.

### **Set or Change Position**

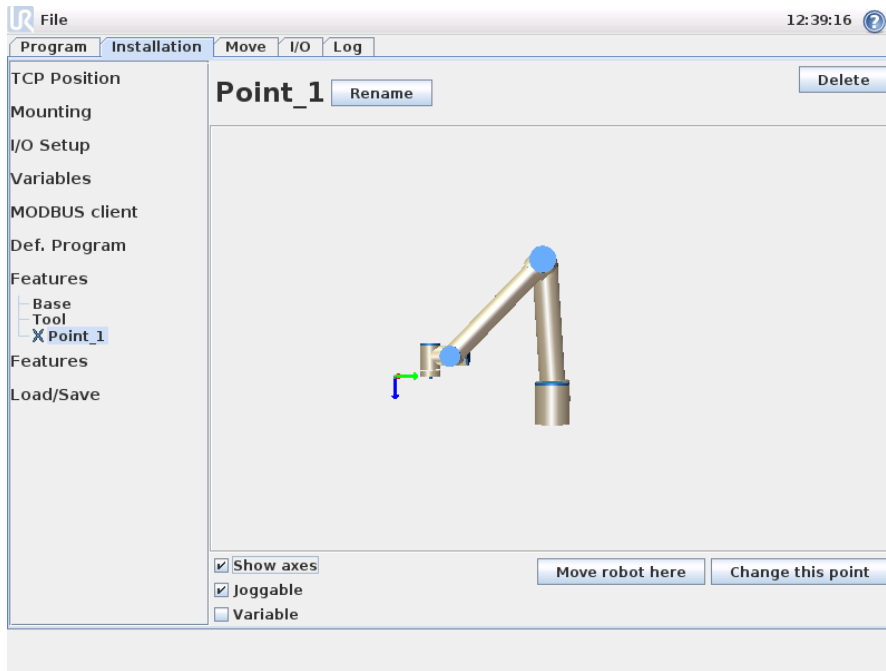
Use this button to set or change the selected feature. The Move screen will appear and a new position of the feature can be set.

### **Move Robot to Feature**

Pressing this button will move the robot towards the selected feature. At the end of this movement, the coordinate systems of the feature and the TCP will coincide, except for a 180 degree rotation about the x-axis.

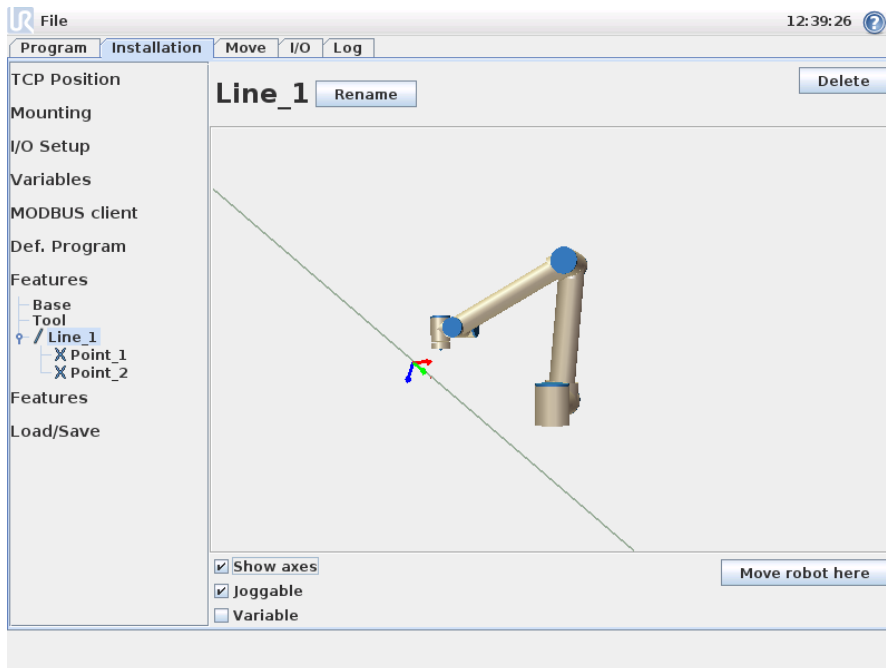
### **Add Point**

Push this button to add a point feature to the installation. The position of a point feature is defined as the position of the TCP at that point. The orientation of the point feature is the same as the TCP orientation, except that the feature coordinate system is rotated 180 degrees about its x-axis. This makes the z-axis of the point feature directed opposite than that of the TCP at that point.



### Add Line

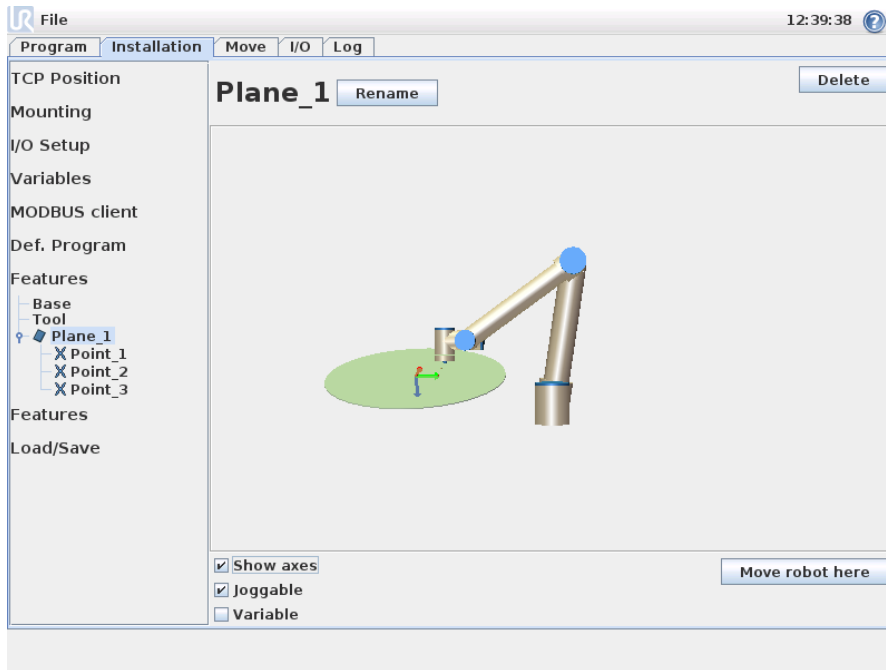
Push this button to add a line feature to the installation. A line is defined as an axis between two point features. This axis, directed from the first point towards the second point, will constitute the y-axis of the line coordinate system. The z-axis will be defined by the projection of the z-axis of the first sub point onto the plane perpendicular to the line. The position of the line coordinate system is the same as the position for the first sub point.



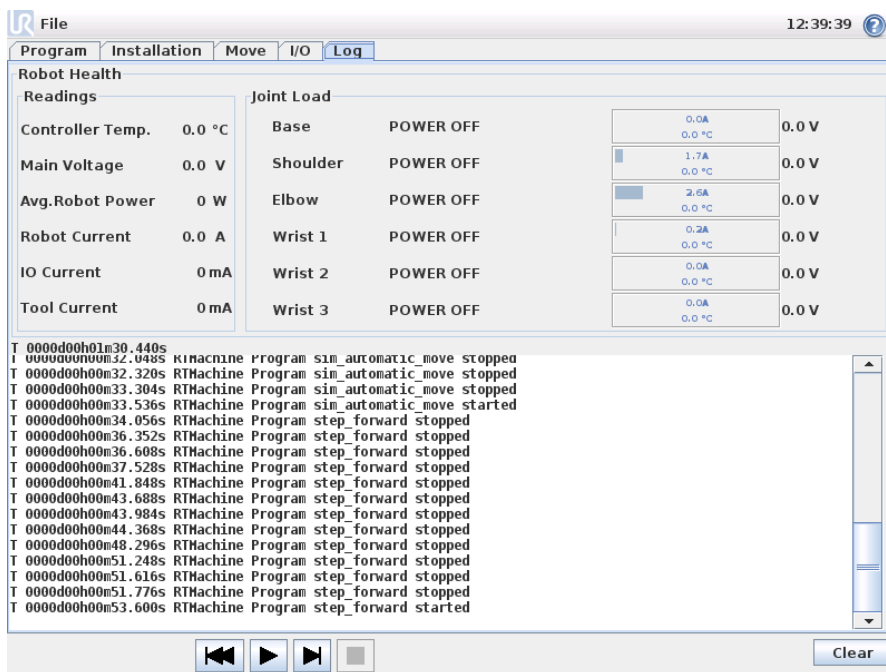
### Add Plane

Push this button to add a plane feature to the installation. A plane is defined by three sub point features. The position of the coordinate system is the same as the position for the first sub point. The z-axis is the plane normal, and the y-axis

is directed from the first point towards the second. The positive direction of the z-axis is set so that the angle between the z-axis of the plane and the z-axis of the first point is less than 180 degrees.



### 3.13 Log Tab



**Robot Health** The top half of the screen displays the health of the robot. The left part shows information related to the control box of the robot, while the right part shows information about each robot joint. Each robot joint shows information for temperature of the motor and electronics, the load of the joint and the voltage at the joint.

**Robot Log** On the bottom half of the screen log messages are shown. The first column shows the time of arrival of the message. The next column shows the sender of the message. The last column shows the message itself.

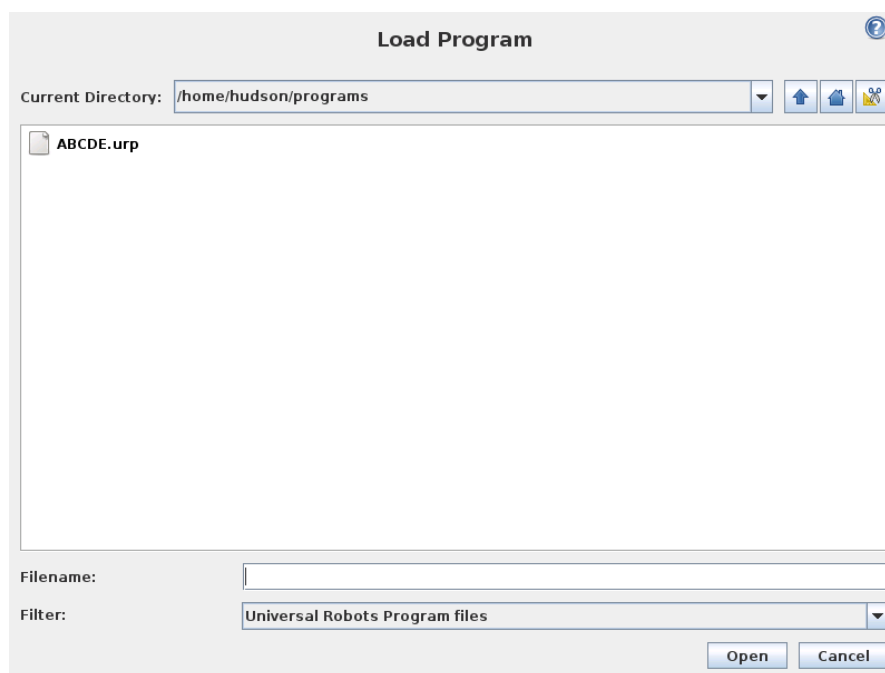
### 3.14 Load Screen

On this screen you choose which program to load. There are two versions of this screen: one that is to be used when you just want to load a program and execute it, and one that is used when you want to actually select and edit a files program.

The main difference lies in which actions are available to the user. In the basic load screen, the user will only be able to access files - not modify or delete them. Furthermore, the user is not allowed to leave the directory structure that descends from the `programs` folder. The user can descend to a sub-directory, but he cannot get any higher than the `programs` folder.

Therefore, all programs should be placed in the `programs` folder and/or sub folders under the `programs` folder.

#### Screen layout



This image shows the actual load screen. It consists of the following important areas and buttons:

**Path history** The path history shows a list of the paths leading up to the present location. This means that all parent directories up to the root of the computer are shown. Here you will notice that you may not be able to access all the directories above the `programs` folder.

By selecting a folder name in the list, the load dialog changes to that directory and displays it in the file selection area 3.14.

**File selection area** In this area of the dialog the contents of the actual area is present. It gives the user the option to select a file by single clicking on its name or to open the file by double clicking on its name.

In the case that the user double-clicks on a directory, the dialog descends into this folder and presents its contents.

**File filter** By using the file filter, one can limit the files shown to include the type of files that one wishes. By selecting "Backup Files" the file selection area will display the latest 10 saved versions of each program, where `.old0` is the newest and `.old9` is the oldest.

**File field** Here the currently selected file is shown. The user has the option to manually enter the file name of a file by clicking on the keyboard icon to the right of the field. This will cause an on-screen keyboard to pop up where the user can enter the file name directly on the screen.

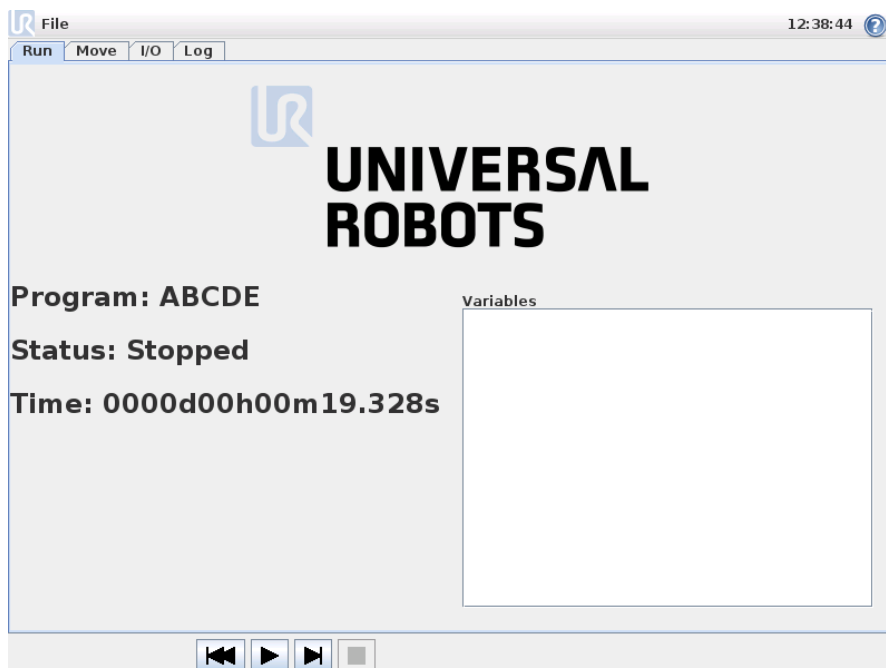
**Open button** Clicking on the Open button, will open the currently selected file and return to the previous screen.

**Cancel button** Clicking on the Cancel button will abort the current loading process and cause the screen to switch to the previous image.

**Action buttons** A series of buttons gives the user the ability to perform some of the actions that normally would be accessible by right-clicking on a file name in a conventional file dialog. Added to this is the ability to move up in the directory structure and directly to the program folder.

- Parent: Move up in the directory structure. The button will not be enabled in two cases: when the current directory is the top directory or if the screen is in the limited mode and the current directory is the program folder.
- Go to program folder: Go home
- Actions: Actions such as create directory, delete file etc.

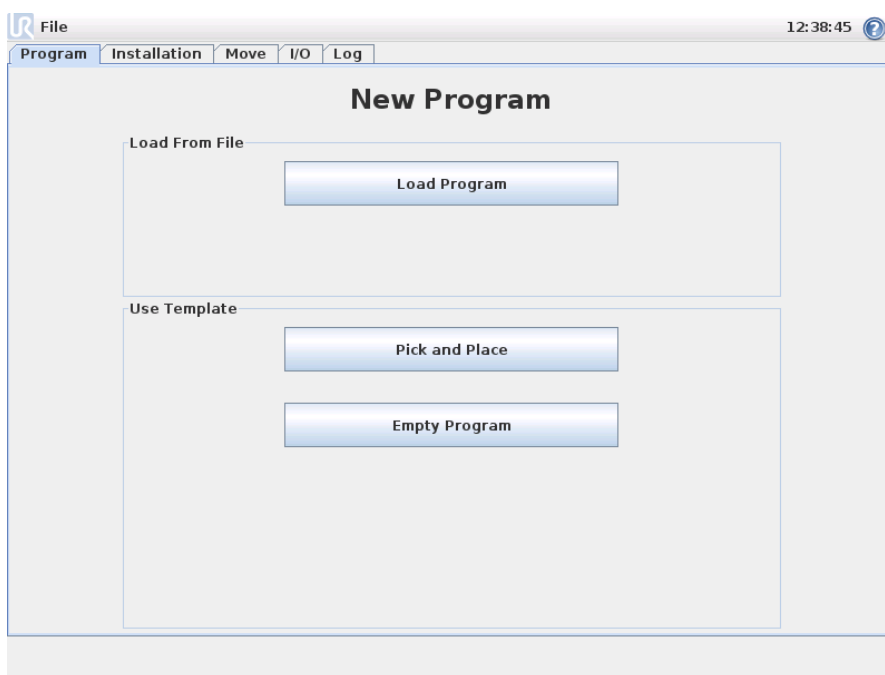
### 3.15 Run Tab



This tab provides a very simple way of operating the robot, with as few buttons and options as possible. This can be useful combined with password protecting the programming part of PolyScope (see section 5.5), to make the robot into a tool that can run exclusively pre-written programs.

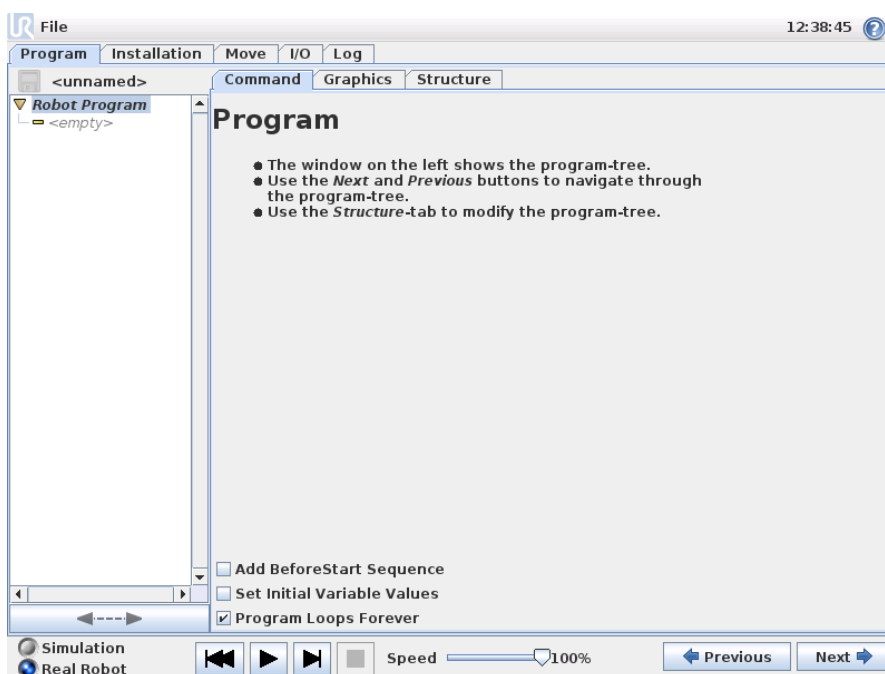
## 4 Programming

## 4.1 Program → New Program



A new robot program can start from either a *template* or from an existing (saved) robot program. A *template* can provide the overall program structure, so only the details of the program need to be filled in.

## 4.2 Program Tab



The program tab shows the current program being edited.

The *program tree* on the left side of the screen displays the program as a list of commands, while the area on the right side of the screen displays information relating to the current command. The current command is selected by clicking the command list, or by using the `Previous` and `Next` buttons on the bottom right of the screen. Commands can be inserted or removed using the

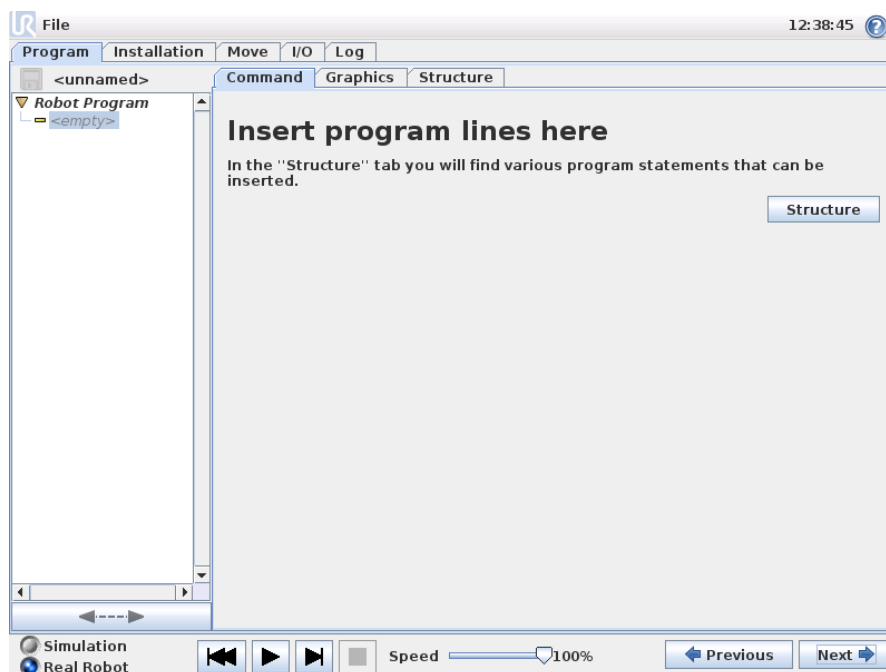
`Structure` tab, described in section 4.28. The program name is shown directly above the command list, with a small disk icon that can be clicked to quickly save the program.

The lowest part of the screen is the *Dashboard*. The *Dashboard* features a set of buttons similar to an old-fashioned tape recorder, from which programs can be started and stopped, single-stepped and restarted. The *speed slider* allow you to adjust the program speed at any time, which directly affects the speed at which the robot moves. To the left of the *Dashboard* the `Simulation` and `Real Robot` buttons toggle between running the program in a simulation, or running it on the real robot. When running in simulation, the robot does not move and thus cannot damage itself or any nearby equipment in collisions. Use simulation to test programs if unsure about what the robot will do.

While the program is being written, the resulting motion of the robot is illustrated using a 3D drawing on the `Graphics` tab, described in section 4.27.

Next to each program command is a small icon, which is either red, yellow or green. A red icon means that there is an error in that command, yellow means that the command is not finished, and green means that all is OK. A program can only be run when all commands are green.

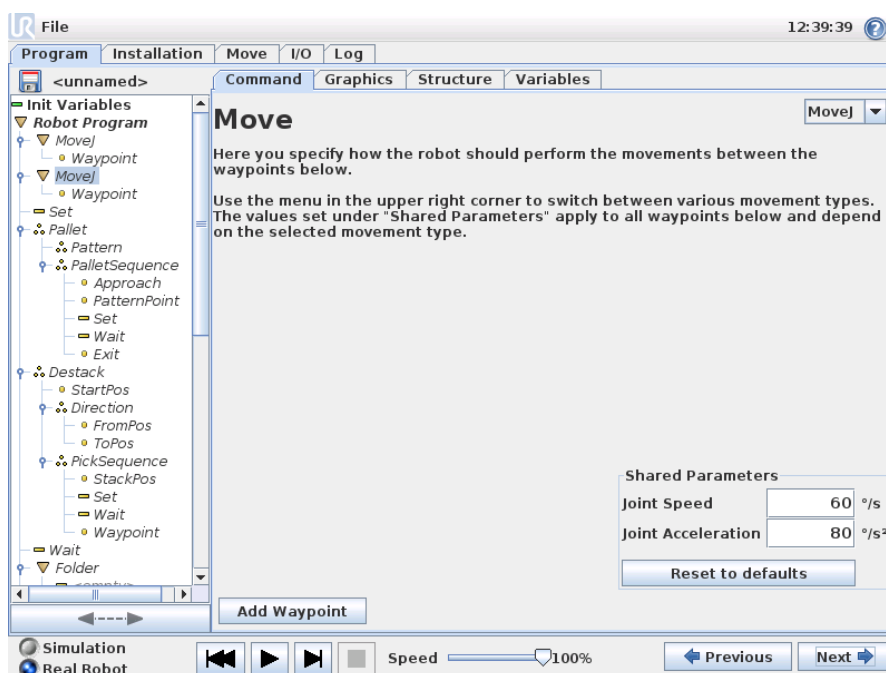
### 4.3 Program → Command Tab, <Empty>



Program commands need to be inserted here. Press the 'Structure' button to go to the structure tab, where the various selectable program lines can be found. A program cannot run before all lines are specified and defined.



#### 4.4 Program → Command Tab, Move



The Move command controls the robot motion through the underlying waypoints. Waypoints have to be under a Move command. The Move command defines the acceleration and the speed at which the robot will move between those waypoints.

#### Movement Types

It is possible to select one of three types of movements: *MoveJ*, *MoveL* and *MoveP* each explained below.

- **moveJ** will make movements that are calculated in the *joint space* of the robot. Each joint is controlled to reach the desired end location at the same time. This movement type results in a curved path for the tool. The shared parameters that apply to this movement type are the maximum joint speed and joint acceleration to use for the movement calculations, specified in  $deg/s$  and  $deg/s^2$ , respectively. If it is desired to have the robot move fast between waypoints, disregarding the path of the tool between those waypoints, this movement type is the favorable choice.
- **moveL** will make the tool move linearly between waypoints. This means that each joint performs a more complicated motion to keep the tool on a straight line path. The shared parameters that can be set for this movement type are the desired tool speed and tool acceleration specified in  $mm/s$  and  $mm/s^2$ , respectively, and also a feature. The selected feature will determine in which feature space the tool positions of the waypoints are represented in. Of specific interest concerning feature spaces are variable features and variable waypoints. Variable features can be used when the tool position of a waypoint need to be determined by the actual value of the variable feature when the robot program runs.
- **moveP** will move the tool linearly with constant speed with circular blends, and is intended for some process operations, like gluing or dispensing. The

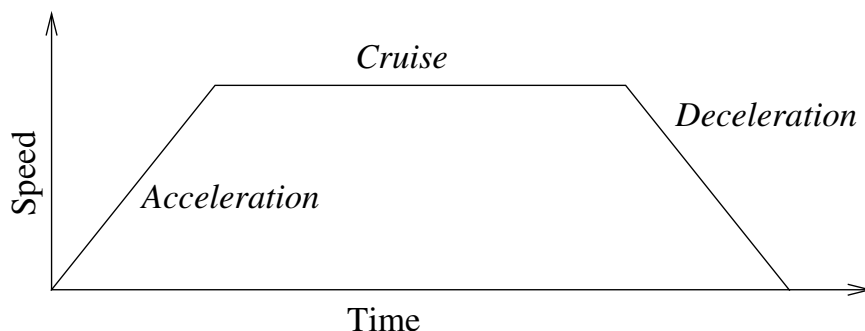
size of the blend radius is per default a shared value between all the waypoints. A smaller value will make the path turn sharper whereas a higher value will make the path smoother. While the robot is moving through the waypoints with constant speed, the robot cannot wait for either an I/O operation or an operator action. Doing so will might stop the robots motion, or cause a security stop.

### Feature selection

For *MoveL* and *MoveP* it is possible to select in which feature space the waypoints under the Move command should be represented when specifying these waypoints. This means that when setting a waypoint, the program will remember the tool coordinates in the feature space of the selected feature. There are a few circumstances that need detailed explanation.

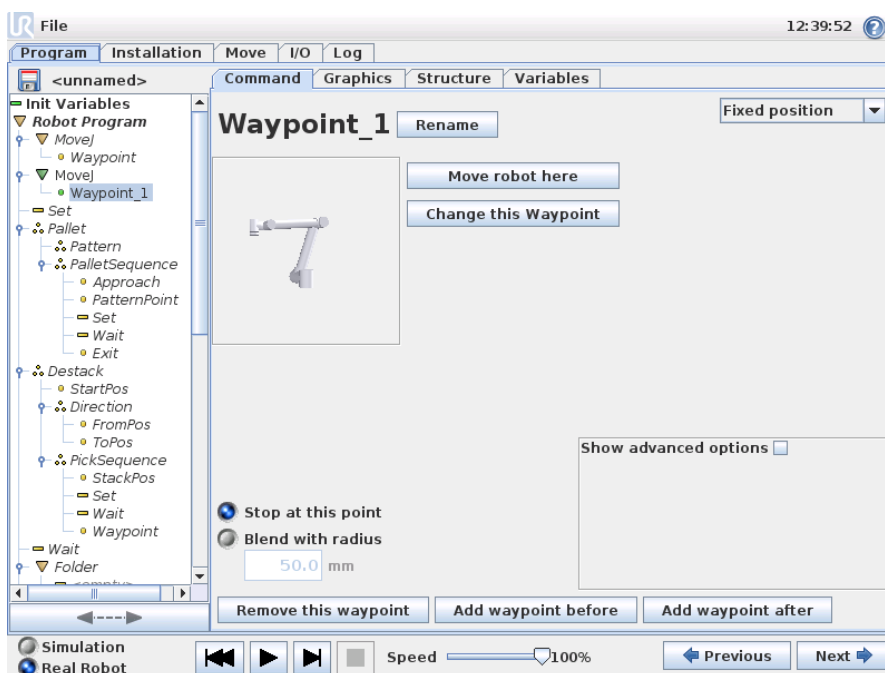
- **Fixed feature:** If a fixed feature, such as e.g. *Base*, is selected this will not have any effect on *Fixed* and *Relative* waypoints. The behavior for *Variable* waypoints is described below.
- **Variable feature:** If any of the features in the currently loaded installation are selected to be variable, these corresponding variables will also be selectable in the feature selection menu. If a feature variable (named by the name of the feature and proceeded by “\_var”) is selected, the robot movements (except to *Relative* waypoints) will depend on the actual value of the variable when the program is running. The initial value of a feature variable is the value of the actual feature. This means that the movements will only change if the feature variable is actively changed by the robot program.
- **Variable waypoint:** When the robot moves to a variable waypoint, the tool target position will always be calculated as the coordinates of the variable in the space of the selected feature. Therefore, the robot movement for a variable waypoint will always change if another feature is selected.

The settings of the Shared Parameters of a Move command apply to the path from the robot’s current position to the first waypoint under the command, and from there to each of the following waypoints. The Move command settings do not apply to the path going *from* the last waypoint under that Move command.



**Figure 1:** Speed profile for a motion. The curve is divided into three segments: *acceleration*, *cruise* and *deceleration*. The level of the *cruise* phase is given by the speed setting of the motion, while the steepness of the *acceleration* and *deceleration* phases is given by the acceleration parameter.

#### 4.5 Program → Command Tab, Fixed Waypoint



A point on the robot path. Waypoints are the most central part of a robot program, telling the robot where to be. A fixed position waypoint is given by physically moving the robot to the position.

#### 4.6 Setting the waypoint

Press this button to enter the Move screen where you can specify the robot position for this waypoint. If the waypoint is placed under a Move command in linear space (`moveL` or `moveP`), there need to be a valid feature selected at that Move command, in order for this button to be pressable.

##### Waypoint names

Waypoint names can be changed. Two waypoints with the same name is always the same waypoint. Waypoints are numbered as they are specified.

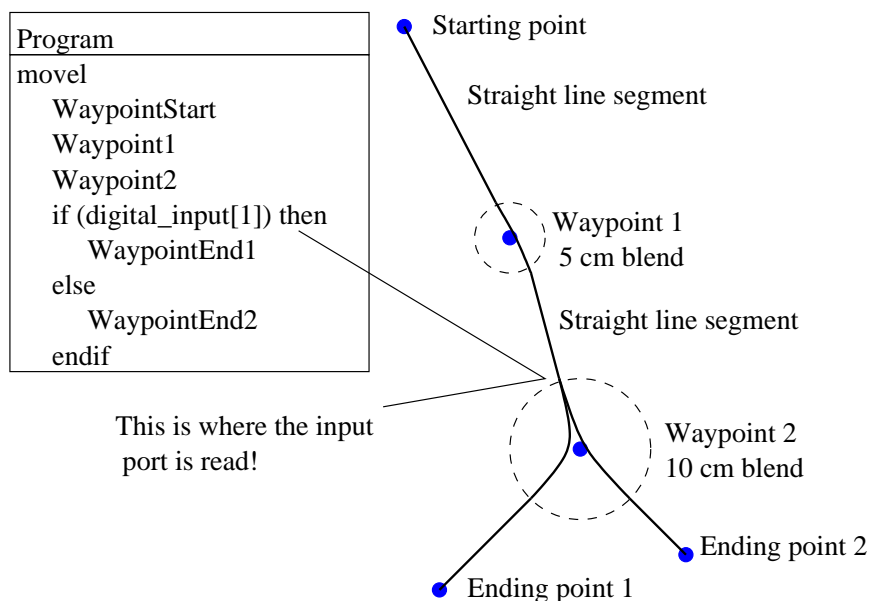
### Blend radius

If a blend radius is set, the robot trajectory blends around the waypoint, allowing the robot not to stop at the point. Blends cannot overlap, so it is not possible to set a blend radius that overlaps a blend radius for a previous or following waypoint. A stop point is a waypoint with a blend radius of  $0.0mm$ .

### Note on I/O Timing

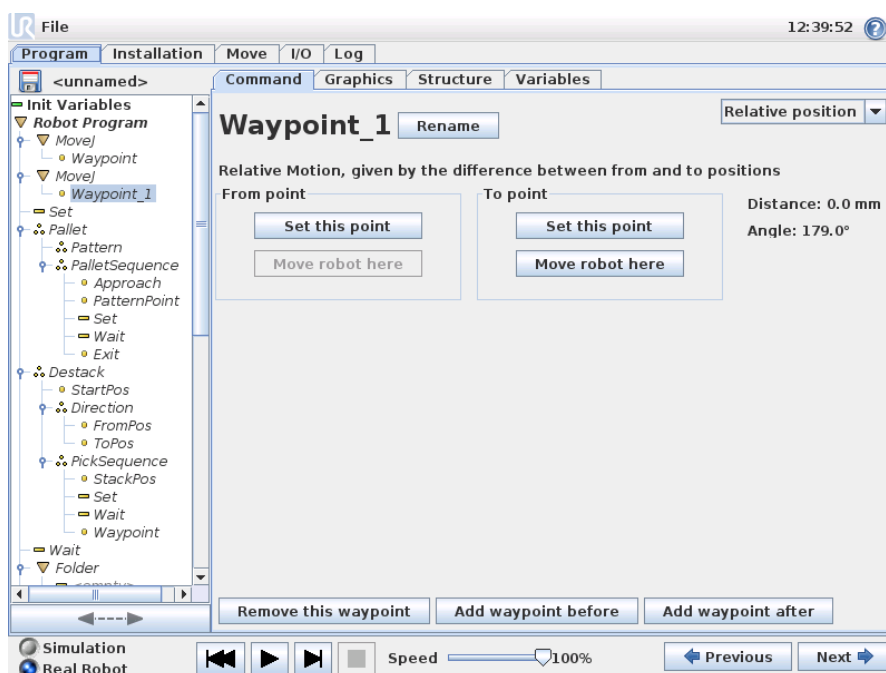
If a waypoint is a stop point with an I/O command as the next command, the I/O command is executed when the robot stops at the waypoint. However, if the waypoint has a blend radius, the following I/O command is executed when the robot enters the blend.

### Example



A small example in which a robot program moves the tool from a starting position to one of two ending positions, depending on the state of `digital_input[1]`. Notice that the tool trajectory (thick black line) moves in straight lines outside the blend areas (dashed circles), while the tool trajectory deviates from the straight line path inside the blend areas. Also notice that the state of the `digital_input[1]` sensor is read just as the robot is about to enter the blend area around `Waypoint 2`, even though the `if...then` command is after `Waypoint 2` in the program sequence. This is somewhat counter-intuitive, but is necessary to allow the robot to select the right blend path.

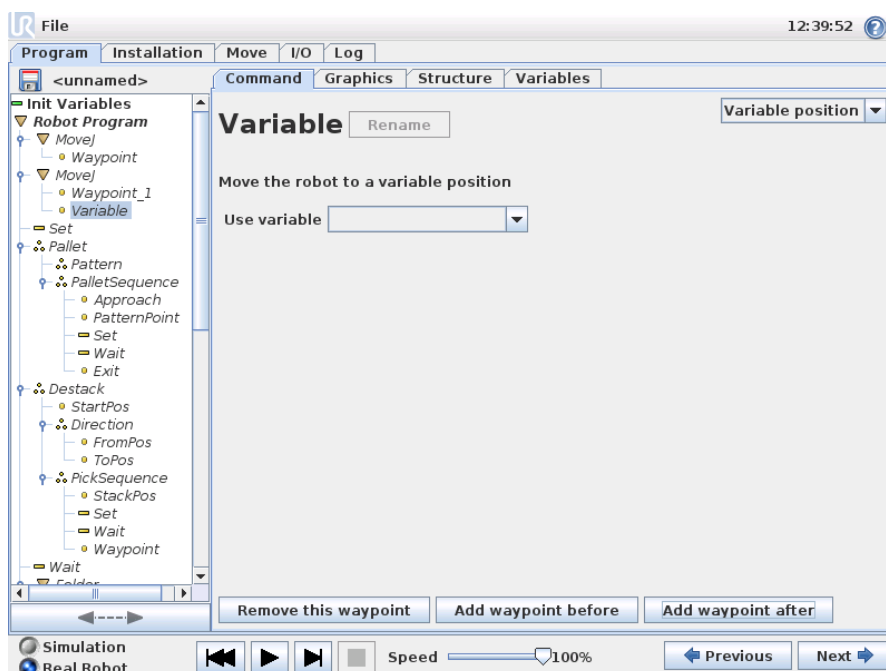
## 4.7 Program → Command Tab, Relative Waypoint



A waypoint with the position given relative to the robot's previous position, such as "two centimeters to the left". The relative position is defined as the difference between the two given positions (left to right). Note that repeated relative positions can move the robot out of its workspace.

The distance here is the Cartesian distance between the tcp in the two positions. The angle states how much the tcp orientation changes between the two positions. More precisely, the length of the rotation vector describing the change in orientation.

## 4.8 Program → Command Tab, Variable Waypoint



A waypoint with the position given by a variable, in this case `calculated_pos`. The variable has to be a pose such as

$var=p[0.5,0.0,0.0,3.14,0.0,0.0]$ . The first three are  $x,y,z$  and the last three are the orientation given as a *rotation vector* given by the vector  $rx,ry,rz$ . The length of the axis is the angle to be rotated in radians, and the vector itself gives the axis about which to rotate. The position is always given in relation to a reference frame or coordinate system, defined by the selected feature. The robot always moves linearly to a variable waypoint.

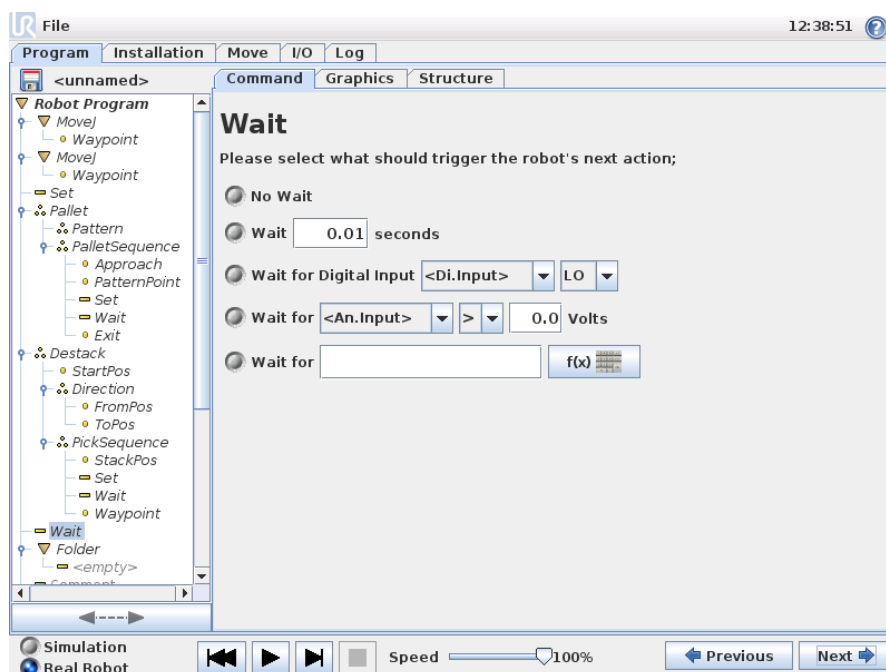
For example, to move the robot 20mm along the z-axis of the tool:

```
var_1=p[0,0,0.02,0,0,0]
```

```
MoveL
```

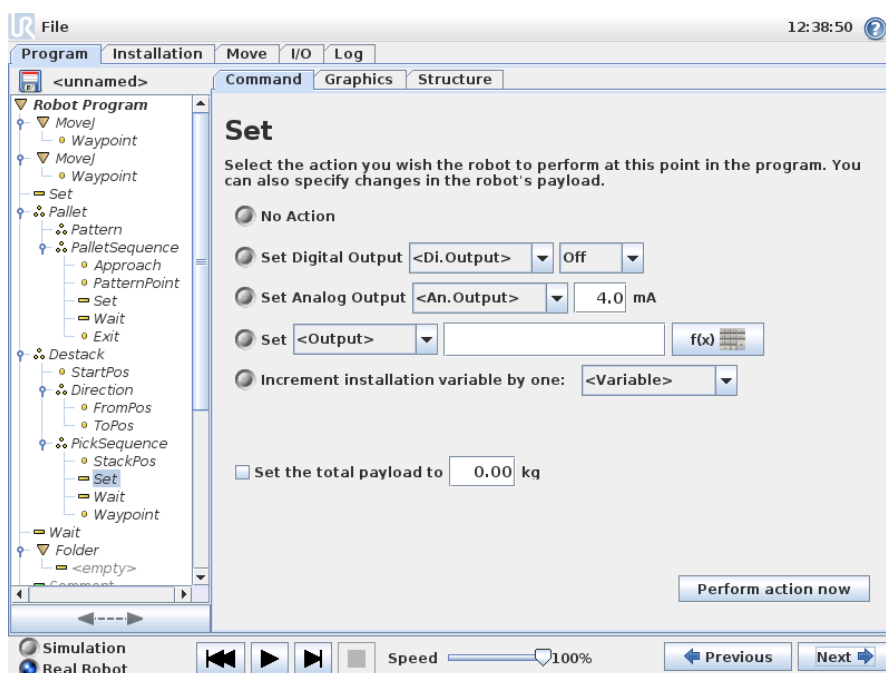
```
Waypoint_1 (varibale position): Use variable=var_1, Feature=Tool
```

## 4.9 Program → Command Tab, Wait



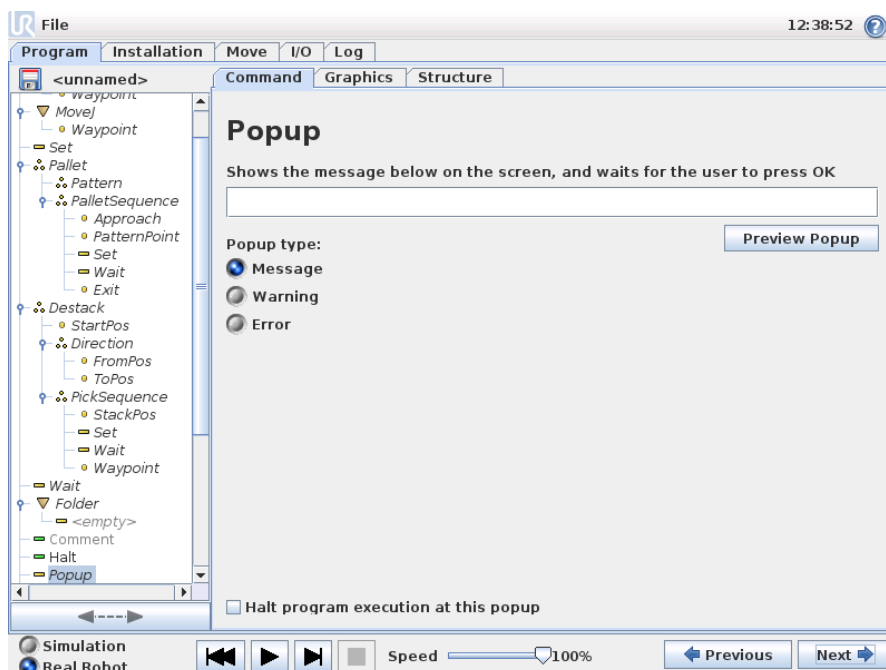
Waits for a given amount of time or for an I/O signal.

#### 4.10 Program → Command Tab, Action



Sets either digital or analog outputs to a given value. Can also be used to set the payload of the robot, for example the weight that is picked up as a consequence of this action. Adjusting the weight can be necessary to prevent the robot from security stopping unexpectedly, when the weight at the tool is different to that which is expected.

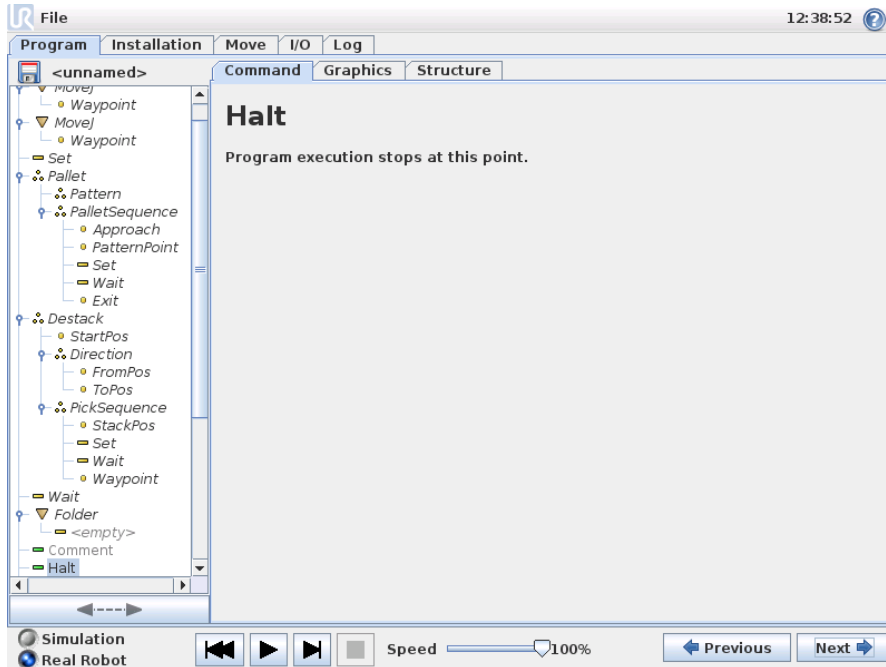
#### 4.11 Program → Command Tab, Popup



The popup is a message that appears on the screen when the program reaches this command. The style of the message can be selected, and the text itself can be given using the on-screen keyboard. The robot waits for the user/operator to press the "OK" button under the popup before continuing the

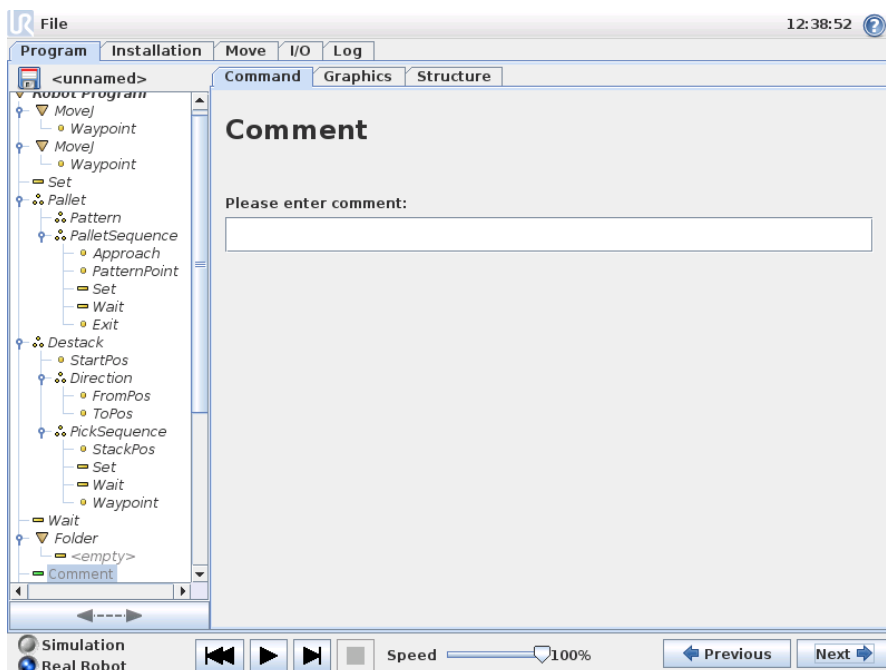
program. If the "Halt program execution" item is selected, the robot program halts at this popup.

#### 4.12 Program → Command Tab, Halt



The program execution stops at this point.

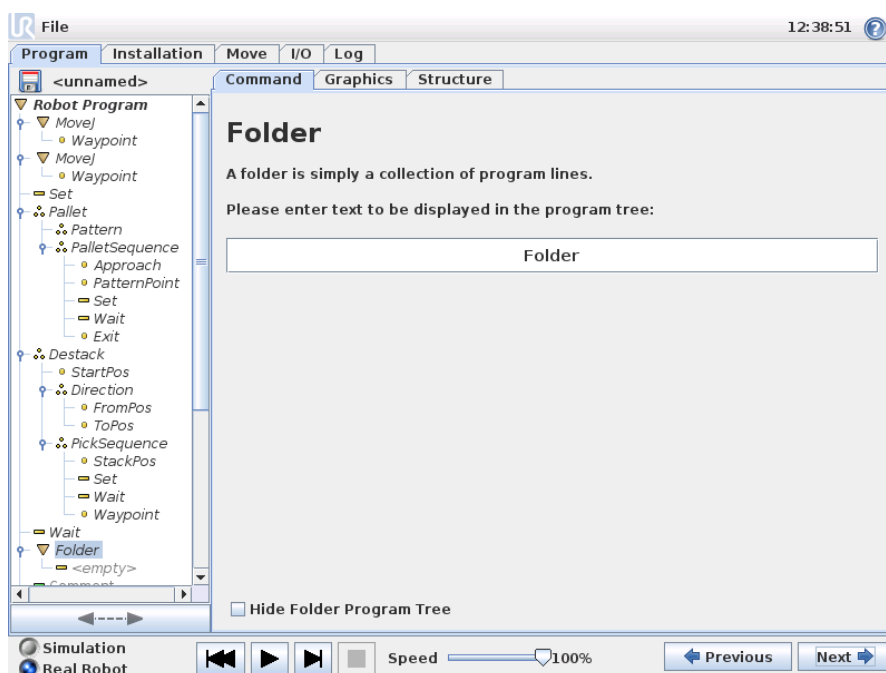
#### 4.13 Program → Command Tab, Comment



Gives the programmer an option to add a line of text to the program. This line of text does not do anything during program execution.

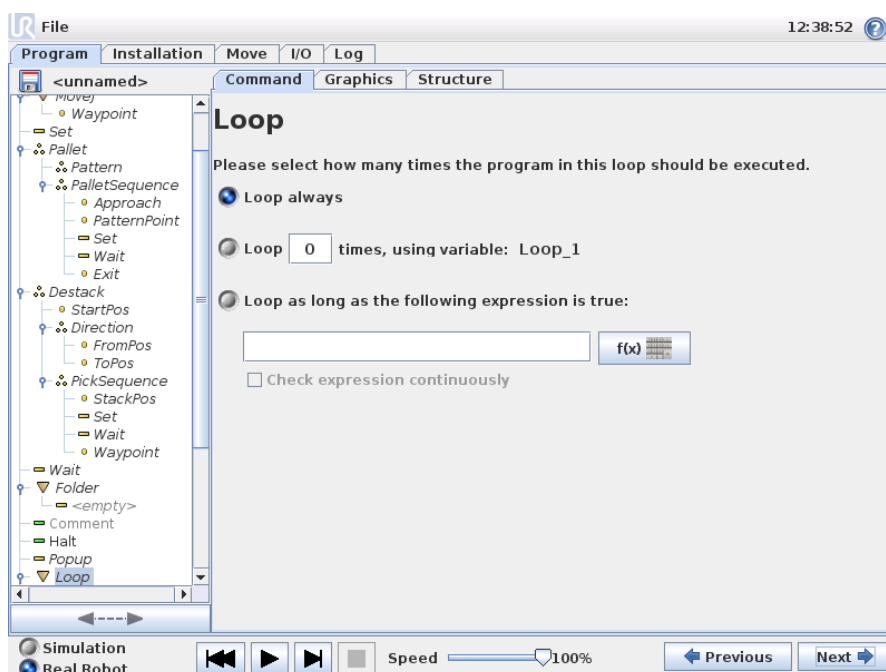


#### 4.14 Program → Command Tab, Folder



A folder is used to organize and label specific parts of a program, to clean up the program tree, and to make the program easier to read and navigate. A folder does not in itself do anything.

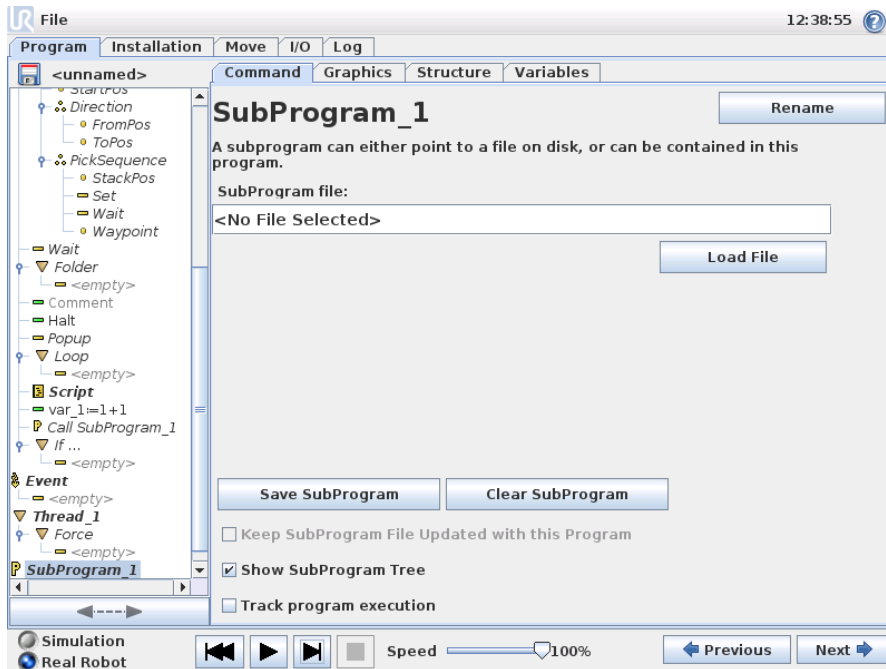
#### 4.15 Program → Command Tab, Loop



Loops the underlying program commands. Depending on the selection, the underlying program commands are either looped infinitely, a certain number of times or as long as the given condition is true. When looping a certain number of times, a dedicated loop variable (called `loop_1` in the screen shot above) is created, which can be used in expressions within the loop. The loop variable counts from 0 to  $N - 1$ .

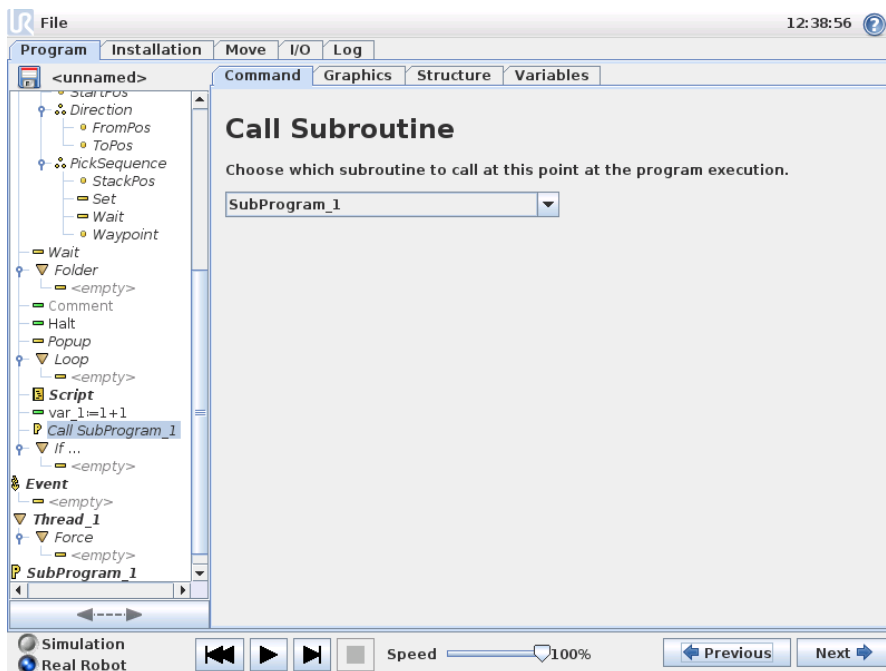
When looping using an expression as end condition, PolyScope provides an option for continuously evaluating that expression, so that the “loop” can be interrupted anytime during its execution, rather than just after each iteration.

#### 4.16 Program → Command Tab, SubProgram



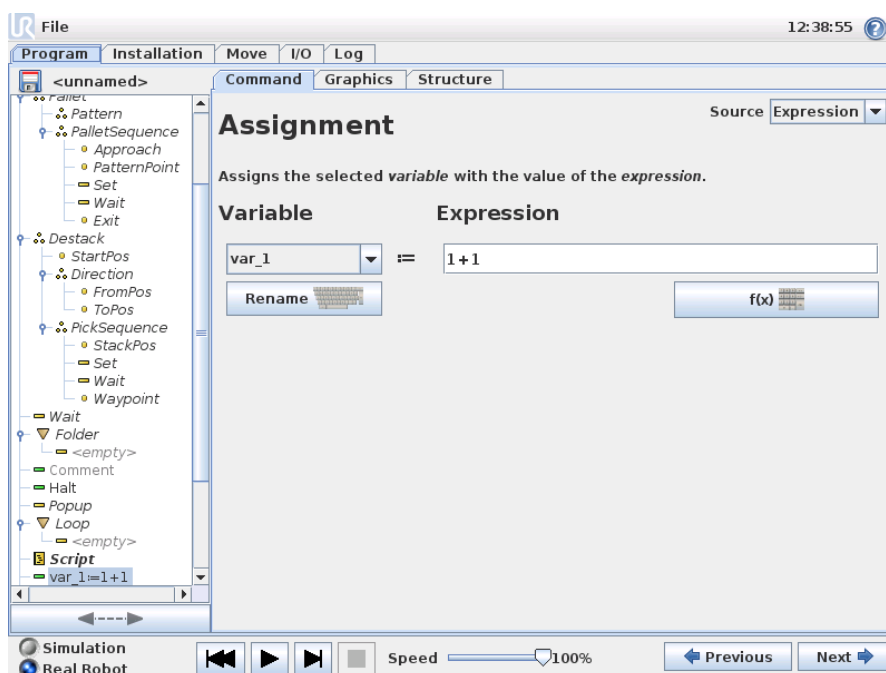
A Sub Program can hold program parts that are needed several places. A Sub Program can be a separate file on the disk, and can also be hidden to protect against accidental changes to the SubProgram.

#### Program → Command Tab, Call SubProgram



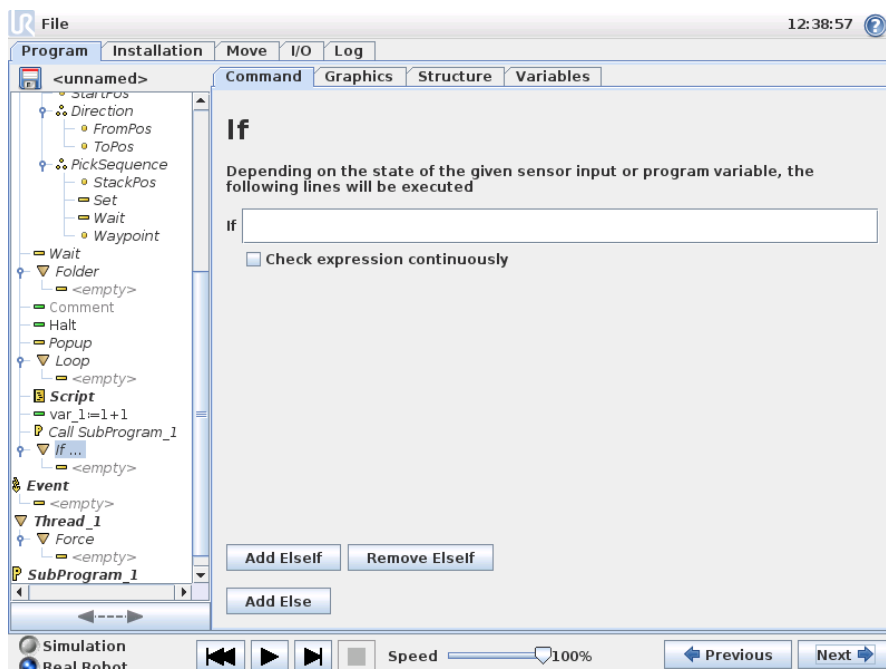
A call to a sub program will run the program lines in the sub program, and then return to the following line.

### 4.17 Program → Command Tab, Assignment



Assigns values to variables. An assignment puts the computed value of the right hand side into the variable on the left hand side. This can be useful in complex programs.

### 4.18 Program → Command Tab, If

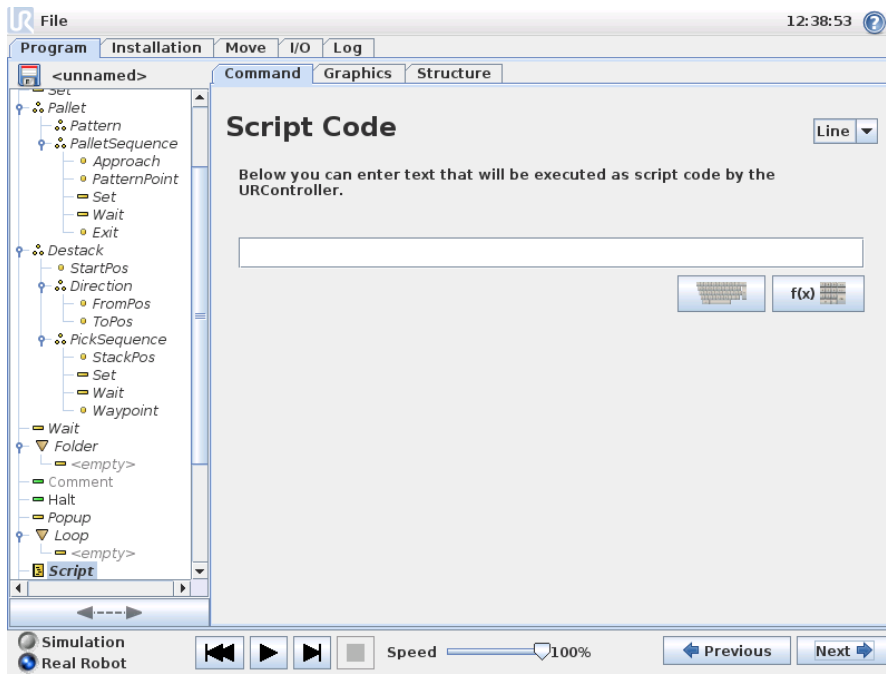


An “if..then..else” construction can make the robot change its behavior based on sensor inputs or variable values. Use the expression editor to describe the condition under which the robot should proceed to the sub-commands of this If. If the condition is evaluated to `True`, the lines inside this If are executed.

Each If can have several ElseIf and one Else command. These can be added using the buttons on the screen. An ElseIf command can be removed from the screen for that command.

The open Check Expression Continuously allow the conditions of the If and ElseIf statements to be evaluated while the contained lines are executed. If a expression evaluates to *False* while inside the body of the If-part, the following ElseIf or Else statement will be reached.

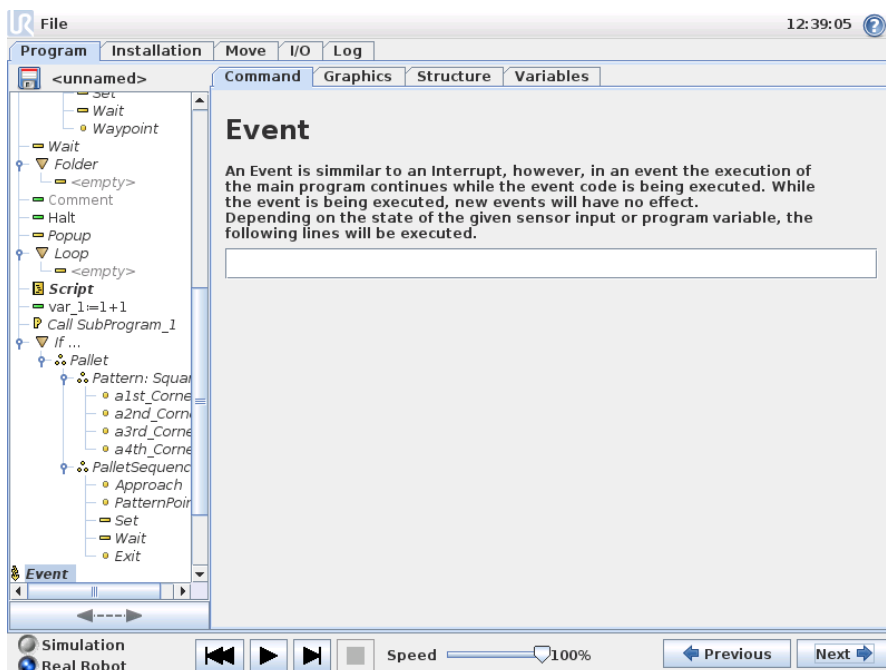
#### 4.19 Program → Command Tab, Script



This command gives access to the underlying real time script language that is executed by the robot controller. It is intended for advanced users only.

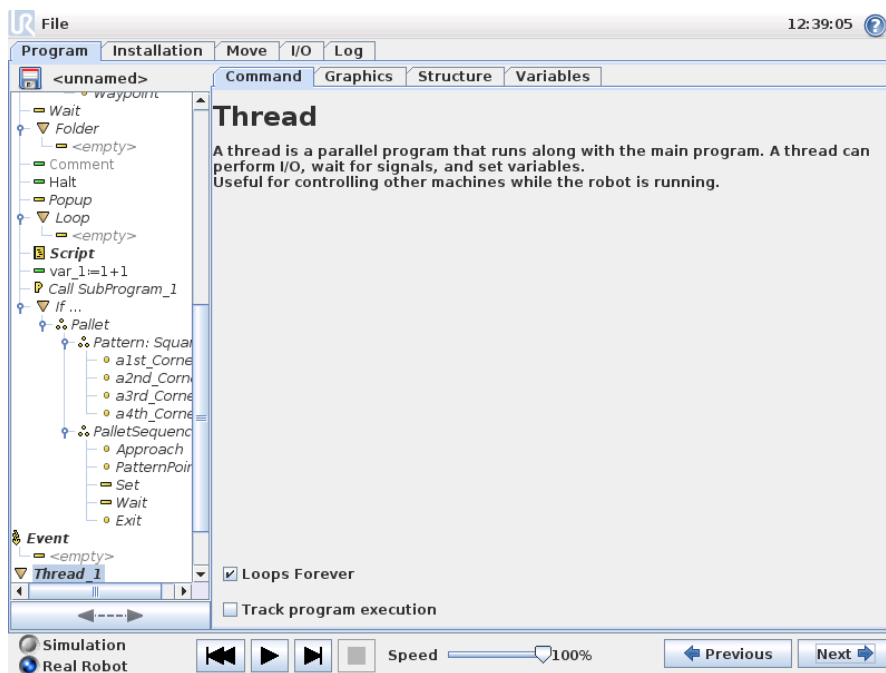
If the “File” option in the top left corner is chosen, it is possible to create and edit script programs files. This way, long and complex script programs can be used together with the operator-friendly programming of PolyScope.

#### 4.20 Program → Command Tab, Event



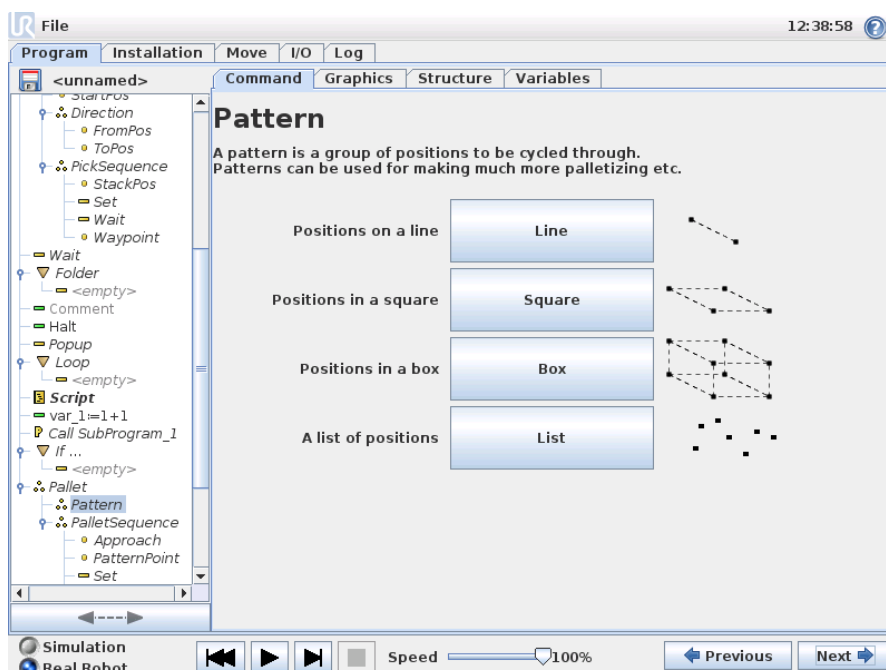
An event can be used to monitor an input signal, and perform some action or set a variable when that input signal goes high. For example, in the event that an output signal goes high, the event program can wait for 100ms and then set it back to low again. This can make the main program code a lot simpler in the case on an external machine triggering on a rising flank rather than a high input level.

#### 4.21 Program → Command Tab, Thread



A thread is a parallel process to the robot program. A thread can be used to control an external machine independently of the robot arm. A thread can communicate with the robot program with variables and output signals.

#### 4.22 Program → Command Tab, Pattern



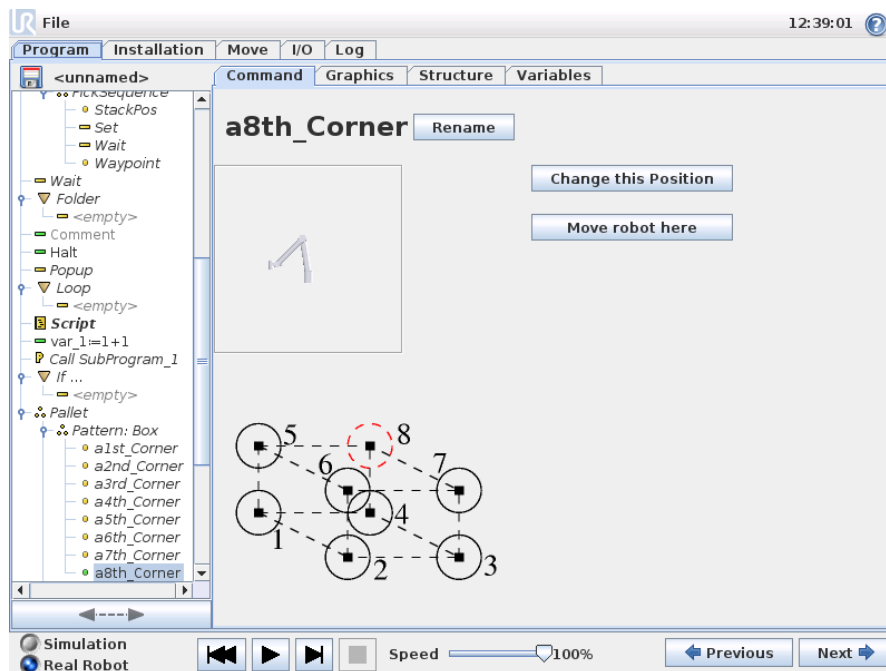
The Pattern command can be used to cycle through positions in the robots program. The pattern command corresponds to one position at each execution.

A pattern can be given as one of four types. The first three, "Line", "Square" or "Box" can be used for positions in a regular pattern. The regular patterns are defined by a number of characteristic points, where the points define the edges of the pattern. For "Line" this is the two end points, for "Square" this is three of the four corner points, where as for "Box" this is four of the eight corner points. The programmer enters the number of positions along each of the edges of the pattern. The robot controller then calculates the individual pattern positions by proportionally adding the edge vectors together.

If the positions to be traversed do not fall in a regular pattern, the "List" option can be chosen, where a list of all the positions is provided by the programmer. This way any kind of arrangement of the positions can be realized.

### Defining the Pattern

When the "Box" pattern is selected, the screen changes to what is shown below.



A "Box" pattern uses three vectors to define the side of the box. These three vectors are given as four points, where the first vector goes from point one to point two, the second vector goes from point two to point three, and the third vector goes from point three to point four. Each vector is divided by the interval count numbers. A specific position in the pattern is calculated by simply adding the interval vectors proportionally.

The "Line" and "Square" patterns work similarly.

A counter variable is used while traversing the positions of the pattern. The name of the variable can be seen on the Pattern command screen. The variable cycles through the numbers from 0 to  $X * Y * Z - 1$ , the number of points in the pattern. This variable can be manipulated using assignments, and can be used in expressions.

### 4.23 Program → Command Tab, Force

Force mode allows for compliance and forces in selectable axis in the robots workspace. All robot movements under a Force command will be in Force mode. When the robot is moving in force mode, it is possible to select one or more axes in which the robot is compliant. Along/around compliant axes the robot will comply with the environment, which means it will automatically adjust its position in order to achieve the desired force. It is also possible to make the robot itself apply a force to its environment, e.g. a workpiece.

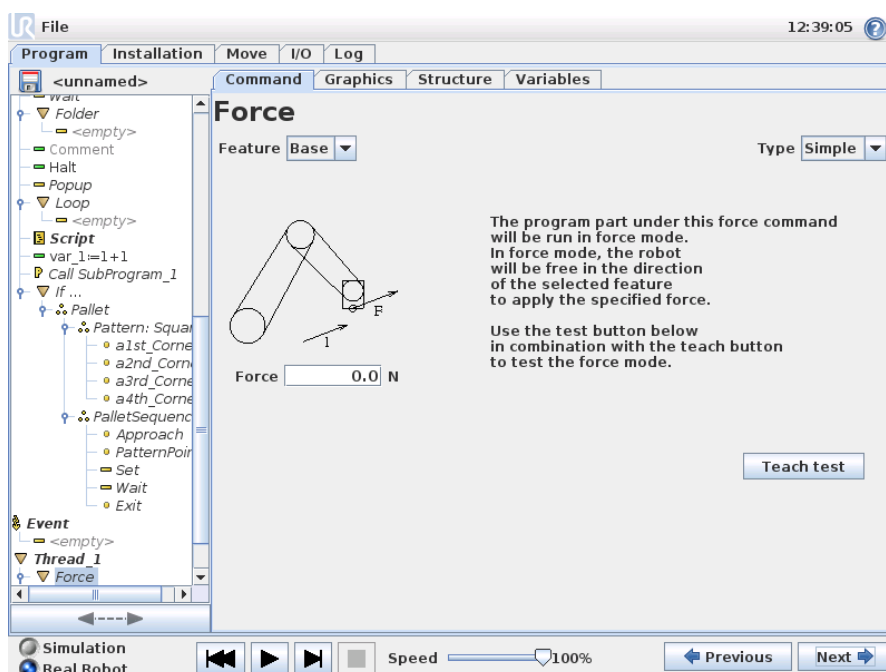
Force mode is suited for applications where the actual tcp position along a predefined axis is not important, but instead a desired force along that axis is required. For example if the robot tcp should roll against a curved surface, or when pushing or pulling a workpiece. Force mode also supports applying certain torques around predefined axes. Note that if no obstacles are met in an axis where a non-zero force is set, the robot will try to accelerate along/about that axis.

Although an axis has been selected to be compliant, the robot program will still try to move the robot along/around that axis. However, the force control assures that the robot will still approach the specified force.



#### WARNING:

1. If the force function is used incorrectly, it can produce a force of more than 150N. The programmed force shall be taken into consideration during risk assessment.



#### Feature selection

The Feature menu is used to select the coordinate system (axes) the robot will use while it is operating in force mode. The features in the menu are those which have been defined in the installation, see 3.12.

### Force mode type

There are four different types of force mode each determining the way in which the selected feature will be interpreted.

- **Simple:** Only one axis will be compliant in force mode. The force along this axis is adjustable. The desired force will always be applied along the z-axis of the selected feature. However, for Line features, it is along their y-axis.
- **Frame:** The Frame type allows for more advanced usage. Here, compliance and forces in all six degrees of freedom can be independently selected.
- **Point:** When Point is selected, the task frame has the y-axis pointing from the robot tcp towards the origin of the selected feature. The distance between the robot tcp and the origin of the selected feature is required to be at least 10 mm. Note that the task frame will change at runtime as the position of the robot tcp changes. The x- and z-axis of the task frame are dependent on the original orientation of the selected feature.
- **Motion:** Motion means that the task frame will change with the direction of the TCP motion. The x-axis of the task frame will be the projection of the tcp movement direction onto the plane spanned by the x- and y-axis of the selected feature. The y-axis will be perpendicular to the robot motion, and in the x-y plane of the selected feature. This can be useful when deburring along a complex path, where a force is needed perpendicular to the TCP motion. Note, when the robot is not moving: If force mode is entered with the robot standing still, there will be no compliant axes until the tcp speed is above zero. If, later on while still in force mode, the robot is again standing still, the task frame has the same orientation as the last time the tcp speed was larger than zero.

For the last three types, the actual task frame can be viewed at runtime on the graphics tab (4.27), when the robot is operating in force mode.

### Force value selection

A force can be set for both compliant and non-compliant axes, but the effects are different.

- **Compliant:** The robot will adjust its position to achieve the selected force.
- **Non-compliant:** The robot will follow its trajectory set by the program while accounting for an external force of the value set here.

For translational parameters, the force is specified in Newtons (N) and for rotational the torque is specified in Newton meters (Nm).

### Limits selection

For all axes a limit can be set, but these have different meanings corresponding to the axes being compliant or non-compliant.

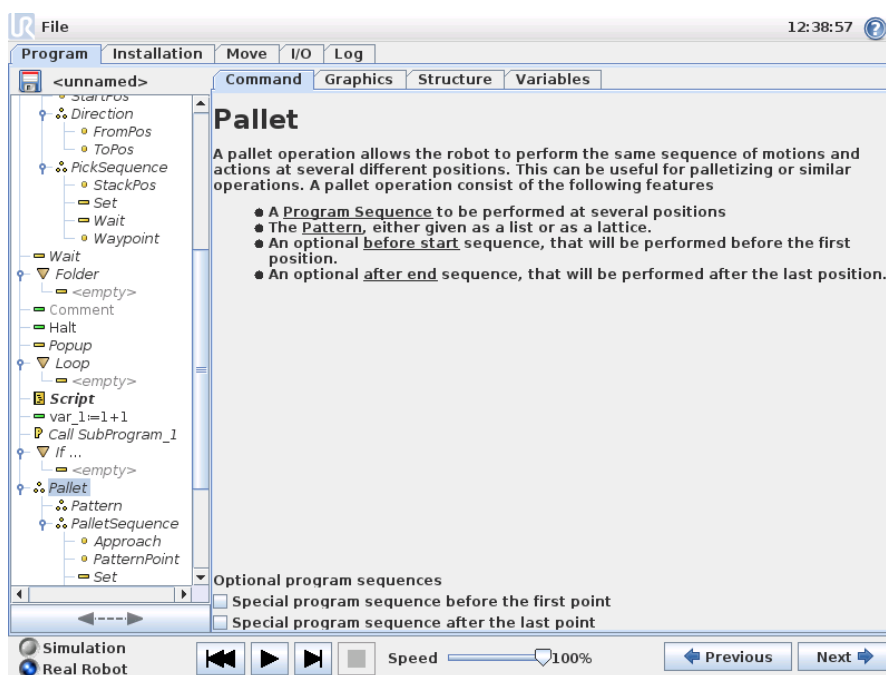
- **Compliant:** The limit is the maximum speed the tcp is allowed to attain along/about the axis. Units are (mm/s) and (deg/s).
- **Non-compliant:** The limit is the maximum deviation from the program trajectory which is allowed before the robot security stops. Units are (mm) and (deg).



## Test force settings

The on/off button, Teach Test, toggles the behavior of the Teach button on the back of the Teach Pendant from normal teaching mode to testing the force command. When the Teach Test button is on and the Teach button on the back of the Teach Pendant is pressed, the robot will perform as if the program had reached this force command, and this way the settings can be verified before actually running the complete program. Especially, this possibility is useful for verifying that compliant axes and forces have been selected correctly. Simply hold the robot tcp using one hand and press the Teach button with the other, and notice in which directions the robot can/cannot be moved. Upon leaving this screen, the Teach Test button automatically switches off, which means the Teach button on the back of the Teach Pendant button is again used for free teach mode. Note: The Teach button will only be effectual when a valid feature has been selected for the Force command.

### 4.24 Program → Command Tab, Pallet



A pallet operation can perform a sequence of motions in a set of places given as a pattern, as described in section 4.22. At each of the positions in the pattern, the sequence of motions will be run relative to the pattern position.

### Programming a Pallet Operation

The steps to go through are as follows;

1. Define the pattern.
2. Make a "PalletSequence" for picking up/placing at each single point. The sequence describes what should be done at each pattern position.
3. Use the selector on the sequence command screen to define which of the waypoints in the sequence should correspond to the pattern positions.

### Pallet Sequence/Anchorable Sequence

In an Pallet Sequence node, the motions of the robot are relative to the pallet position. The behavior of a sequence is such that the robot will be at the position specified by the pattern at the `Anchor Position/Pattern Point`. The remaining positions will all be moved to make this fit.

Do not use the `Move` command inside a sequence, as it will not be relative to the anchor position.

#### “BeforeStart”

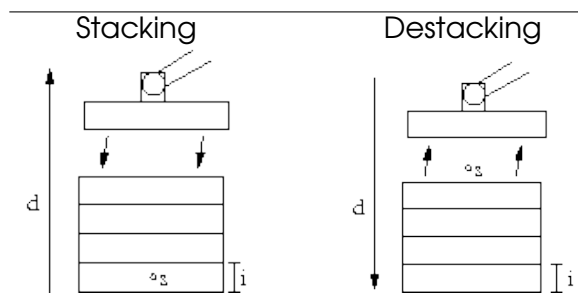
The optional `BeforeStart` sequence is run just before the operation starts. This can be used to wait for ready signals.

#### “AfterEnd”

The optional `AfterEnd` sequence is run when the operation is finished. This can be used to signal conveyor motion to start, preparing for the next pallet.

### 4.25 Program → Command Tab, Seek

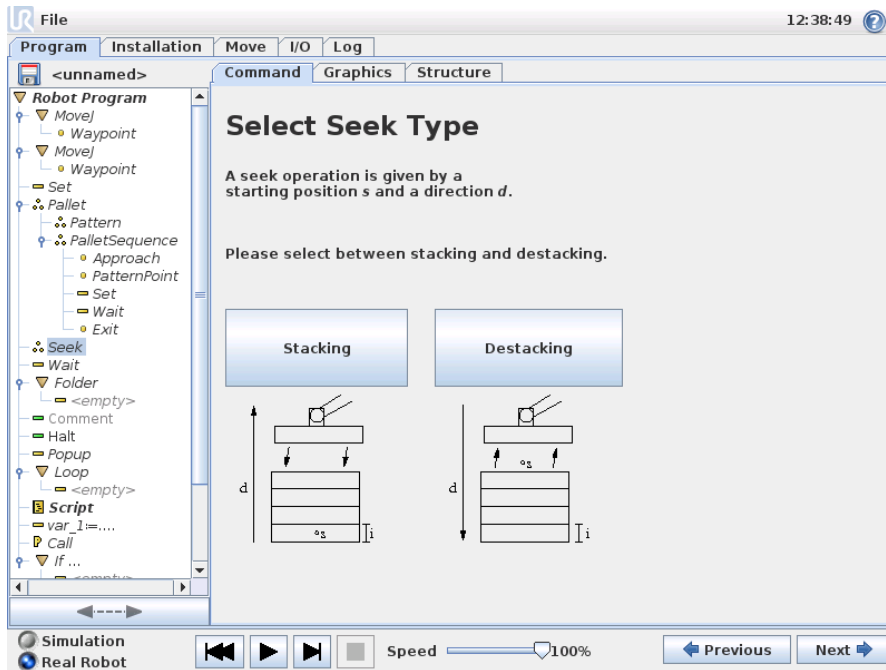
A seek function uses a sensor to determine when the correct position is reached to grab or drop an item. The sensor can be a push button switch, a pressure sensor or a capacitive sensor. This function is made for working on stacks of items with varying item thickness, or where the exact positions of the items are not known or too hard to program.



When programming a seek operation for working on a stack, one must define  $s$  the starting point,  $d$  the stack direction and  $i$  the thickness of the items in the stack.

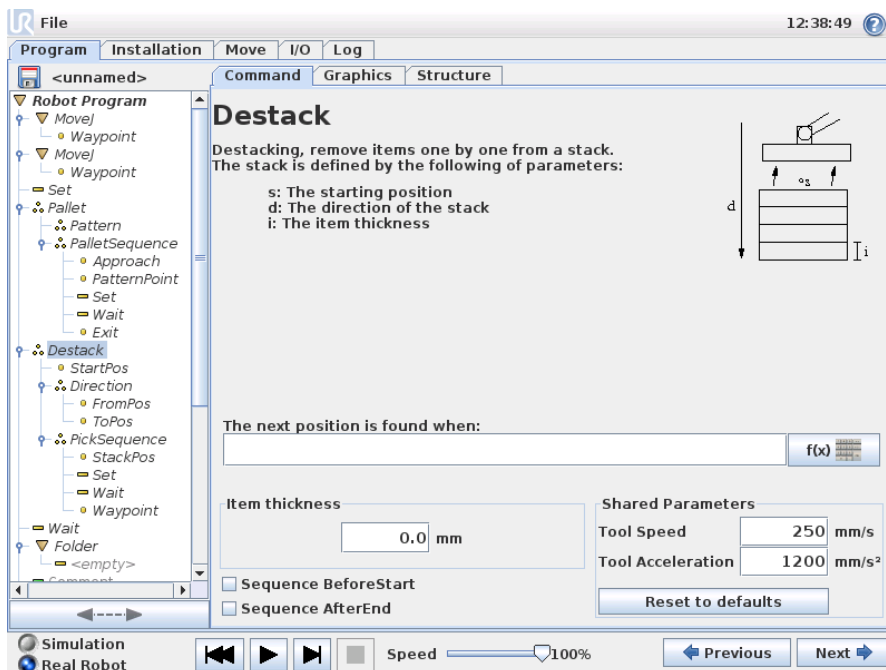
On top of this, one must define the condition for when the next stack position is reached, and a special program sequence that will be performed at each of the stack positions. Also speed and accelerations need to be given for the movement involved in the stack operation.

### Stacking



When stacking, the robot moves to the starting position, and then moves *opposite* the direction to search for the next stack position. When found, the robot remembers the position and performs the special sequence. The next time round, the robot starts the search from the remembered position incremented by the item thickness along the direction. The stacking is finished when the stack height is more than some defined number, or when a sensor gives a signal.

### Destacking



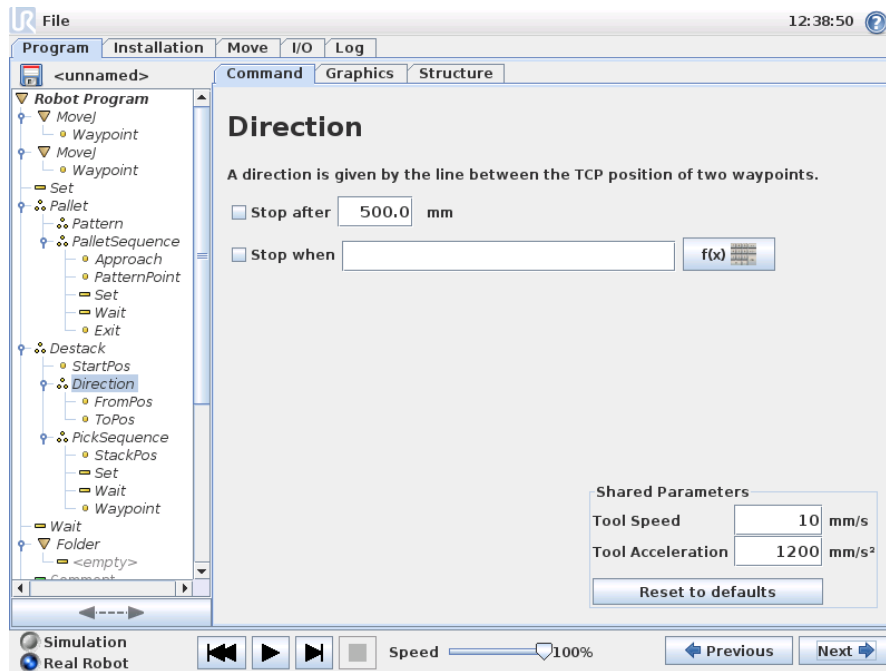
When destacking, the robot moves from the starting position in the given direction to search for the next item. When found, the robot remembers the position and performs the special sequence. The next time round, the robot starts the

search from the remembered position, incremented by the item thickness along the direction.

### Starting position

The starting position is where the stack operation starts. If the starting position is omitted, the stack starts at the robots current position.

### Direction



The direction is given by two positions, and is calculated as the position difference from the first positions TCP to the second positions TCP. Note: A direction does not consider the orientations of the points.

### Next Stacking Position Expression

The robot moves along the direction vector while continuously evaluating whether the next stack position has been reached. When the expression is evaluated to `True` the special sequence is executed.

### “BeforeStart”

The optional `BeforeStart` sequence is run just before the operation starts. This can be used to wait for ready signals.

### “AfterEnd”

The optional `AfterEnd` sequence is run when the operation is finished. This can be used to signal conveyor motion to start, preparing for the next stack.

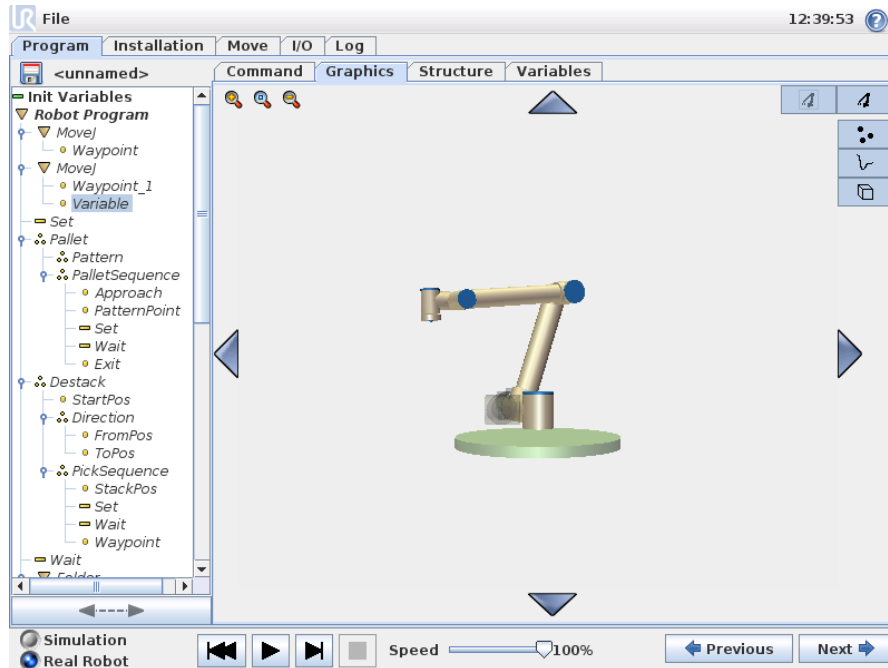
### Pick/Place Sequence

Like for the Pallet operation (4.24), a special program sequence is performed at each stack position.

#### 4.26 Program → Command Tab, Suppress

Suppressed program lines are simply skipped when the program is run. A suppressed line can be unsuppressed again at a later time. This is a quick way to make changes to a program without destroying the original contents.

### 4.27 Program → Graphics Tab

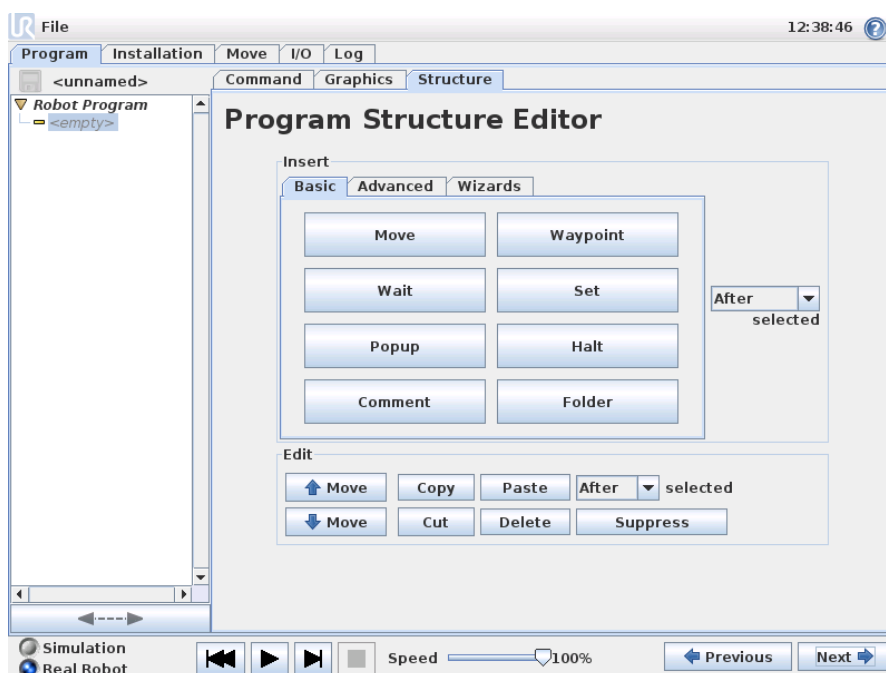


Graphical representation of the current robot program. The path of the TCP is shown in the 3D view, with motion segments in black, and blend segments (transitions between motion segments) shown in green. The green dots specify the positions of the TCP at each of the waypoints in the program. The 3D drawing of the robot shows the current position of the robot, and the “shadow” of the robot shows how the robot intends to reach the waypoint selected in the left hand side of the screen.

The 3D view can be zoomed and rotated to get a better view of the robot. The buttons in the top-right side of the screen can disable the various graphical components in the 3D view.

The motion segments shown depends on the selected program node. If a `Move` node is selected, the displayed path is the motion defined by that move. If a `Waypoint` node is selected, the display shows the following ~ 10 steps of movement.

## 4.28 Program → Structure Tab



The program structure tab gives an opportunity for inserting, moving, copying and removing the various types of commands.

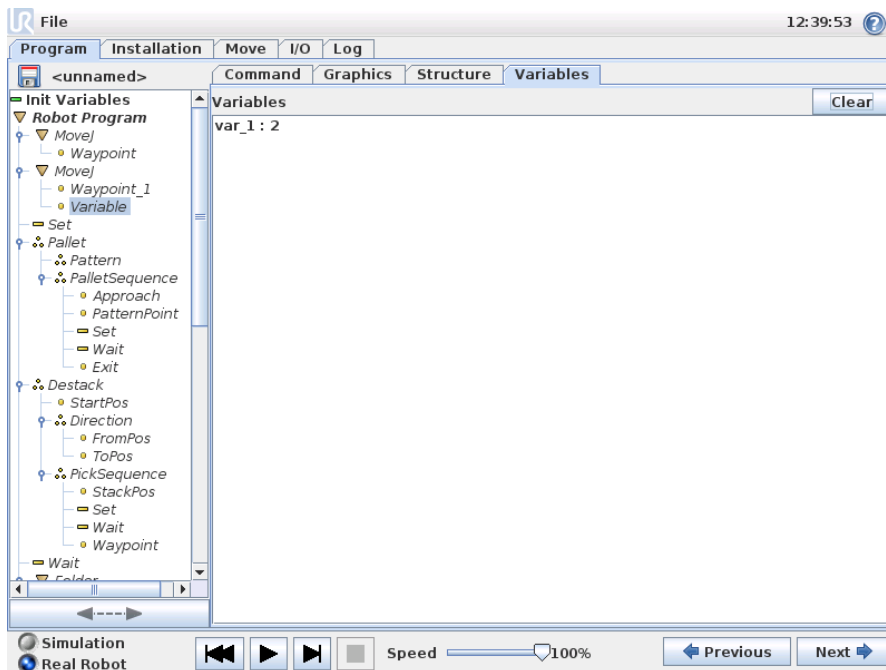
To insert new commands, perform the following steps:

- 1) Select an existing program command.
- 2) Select whether the new command should be inserted above or below the selected command.
- 3) Press the button for the command type you wish to insert. For adjusting the details for the new command, go to the `Command` tab.

Commands can be moved/cloned/deleted using the buttons in the edit frame. If a command has sub-commands (a triangle next to the command) all sub-commands are also moved/cloned/deleted.

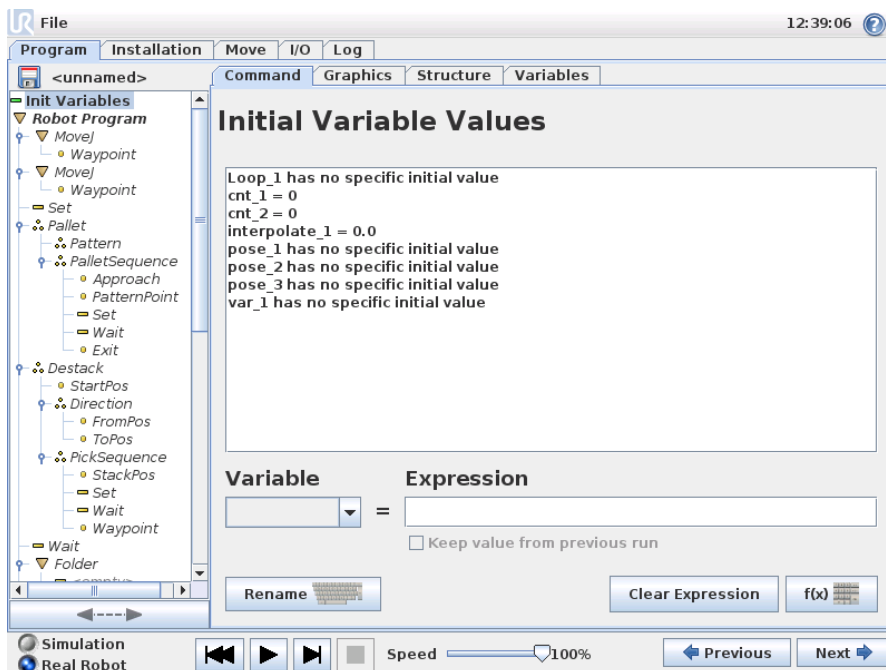
Not all commands fit at all places in a program. `Waypoints` must be under a `Move` command (not necessarily directly under). `ElseIf` and `Else` commands are required to be after an `If`. In general, moving `ElseIf` commands around can be messy. Variables must be assigned values before being used.

### 4.29 Program → Variables Tab



The Variables tab shows the live values of the variables in the running program, and keeps a list of variables and values between program runs. The variables tab appears only when it has information to display. The variable names on this screen are shown with at most 50 characters, and the values of the variables are shown with at most 500 characters.

### 4.30 Program → Command Tab, Variables Initialization



This screen allows setting variable values before the program (and any threads) start executing.

Select a variable from the list of variables by clicking on it, or by using the variable selector box. For a selected variable, an expression can be entered that will be used to set the variable value at program start.

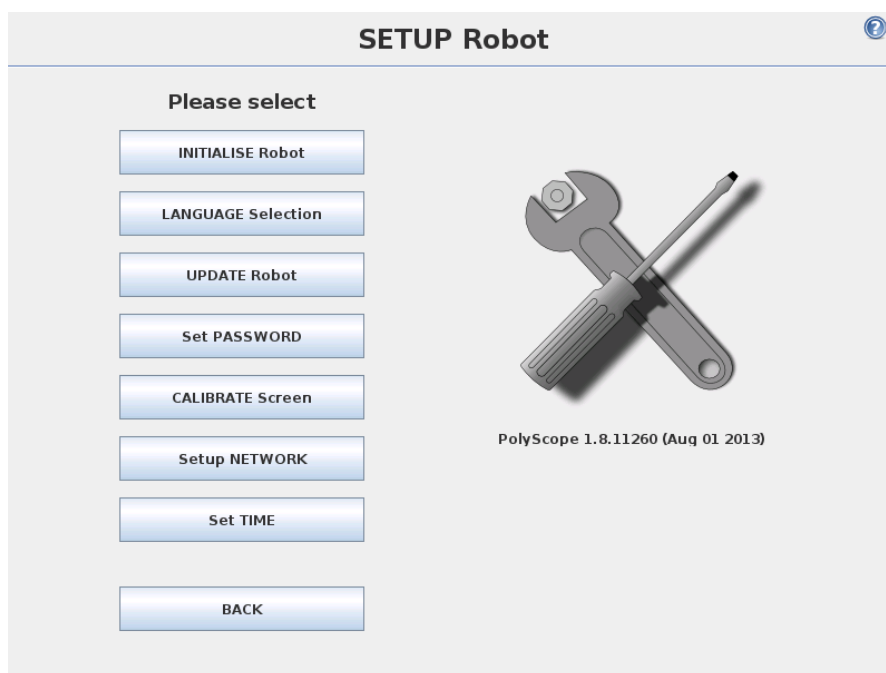


If the 'Prefers to keep value from last run' checkbox is selected, the variable will be initialized to the value found on the `Variables` tab, described in section 4.29. This permits variables to maintain their values between program executions. The variable will get its value from the expression if the program is run for the first time, or if the value tab has been cleared.

A variable can be deleted from the program by setting its name to blank (only spaces).

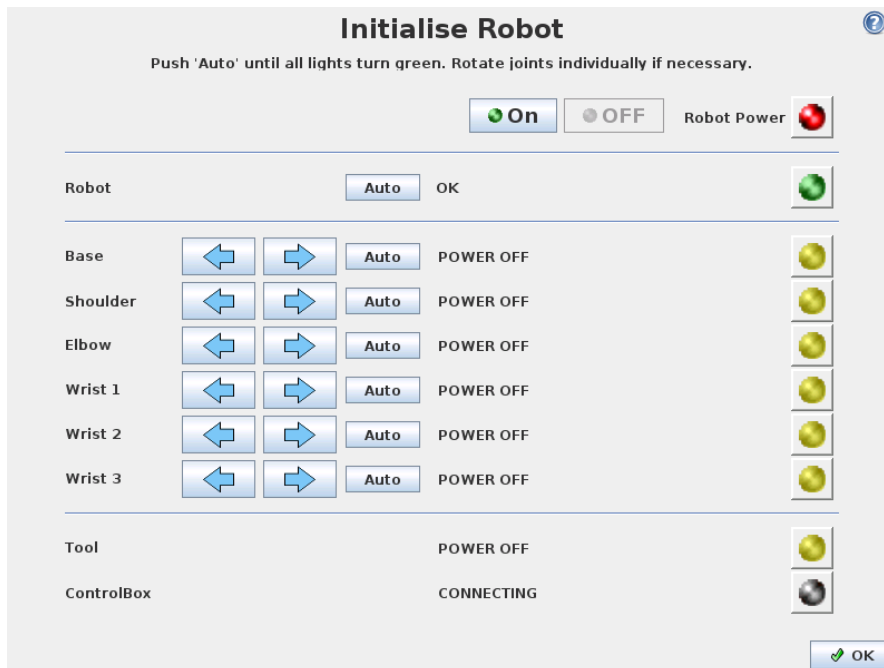
## 5 Setup

### 5.1 Setup Screen



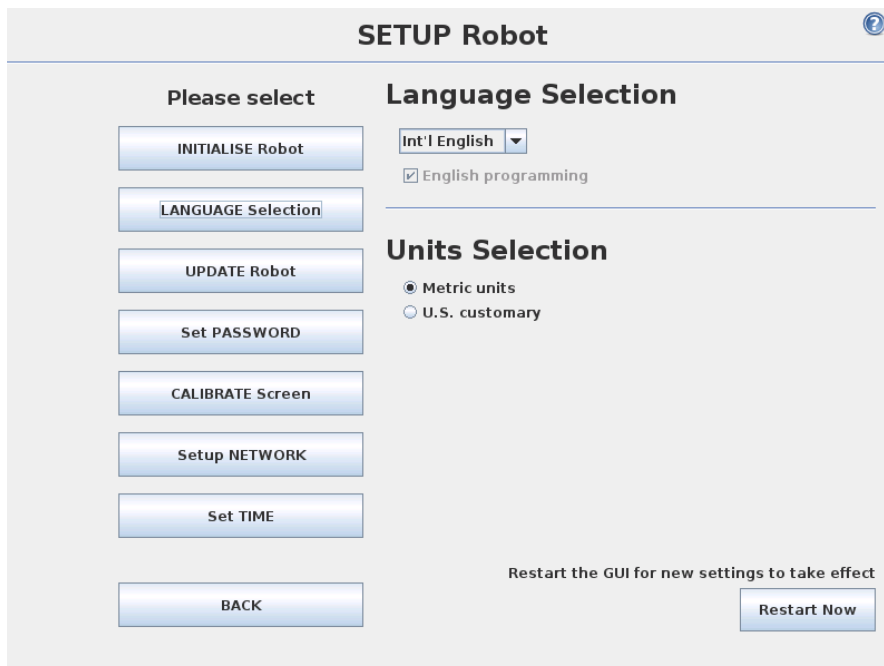
- **Initialize Robot** Goes to the initialization screen, see section 5.2.
- **Language** Configure the language and units of measurements for the user interface, see section 5.3.
- **Update** Upgrades the robot software to a newer version, see section 5.4.
- **Set Password** Provides the facility to lock the programming part of the robot to people without a password, see section 5.5.
- **Calibrate Screen** Calibrates the "touch" of the touch screen, see section 5.6.
- **Setup Network** Opens the interface for setting up the Ethernet network for the robot, see section 5.7.
- **Time** Set the time and date for the system and configure the display formats for the clock, see section 5.8.
- **Back** Returns to the Welcome Screen.

### 5.2 Setup Screen → Initialize



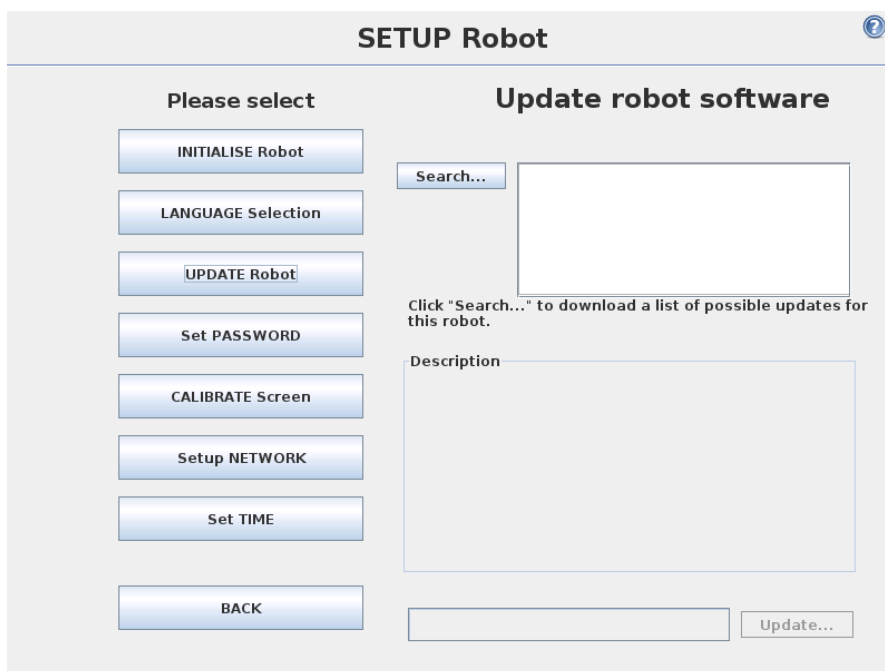
This screen is used when powering up the robot. Before the robot can operate normally, each joint needs to move a little (about 20°) to find its exact position. The `Auto` button drives all joints until they are `OK`. The joints change drive direction when the button is released and pressed again.

### 5.3 Setup Screen → Language Select



This screen provides for the selection of language and units of measurements used for the PolyScope software, and for the help. Tick off “English programming” to have the programming in English. The GUI needs to be restarted for changes to take effect.

## 5.4 Setup Screen → Update



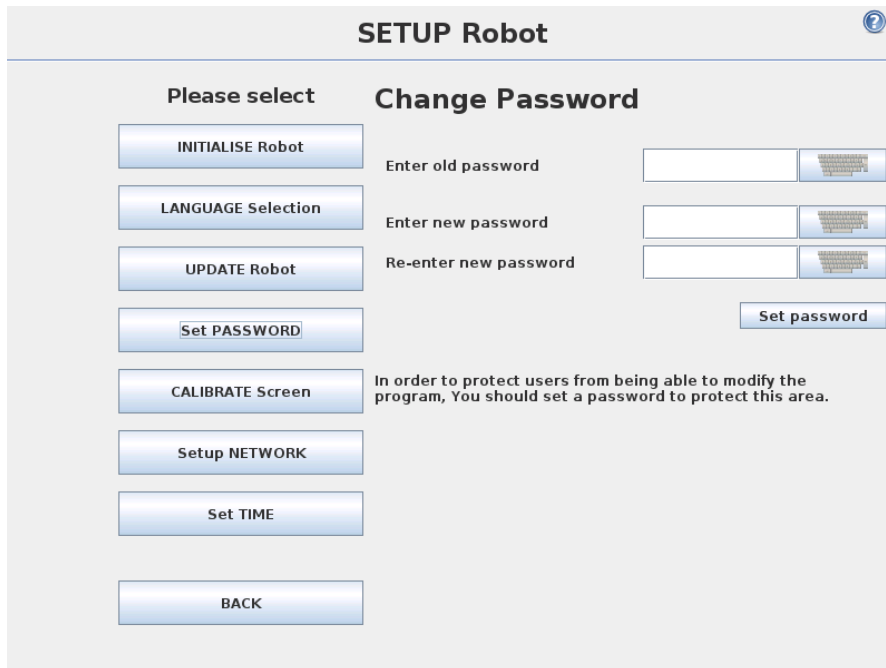
Software updates can be installed from USB flash memory. Insert an USB memory stick and click **Search** to list its contents. To perform an update, select a file, click **Update**, and follow the on-screen instructions.



### **WARNING:**

1. Always check your programs after a software upgrade. The upgrade might change trajectories in your program. The updated software specifications can be found by pushing the "?" button located at the top right corner of the GUI. Hardware specifications remain the same and can be found in the original manual.

### 5.5 Setup Screen → Password



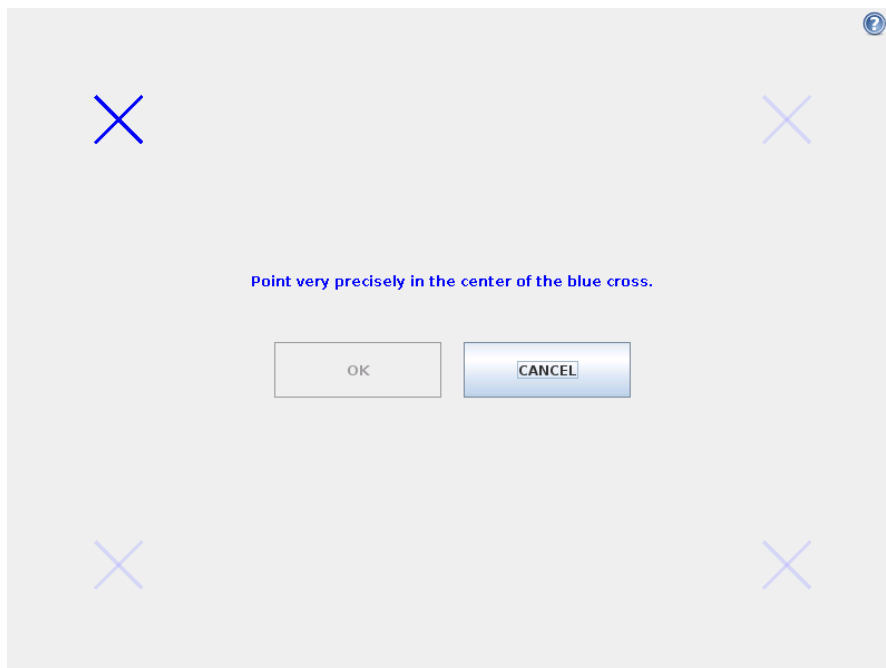
The programming part of the software can be locked using a password. When locked, programs can be loaded and run without the password, but a password is required to create or change programs.



**WARNING:**

1. Add a password to prevent non-authorized personnel from changing the robot installation.

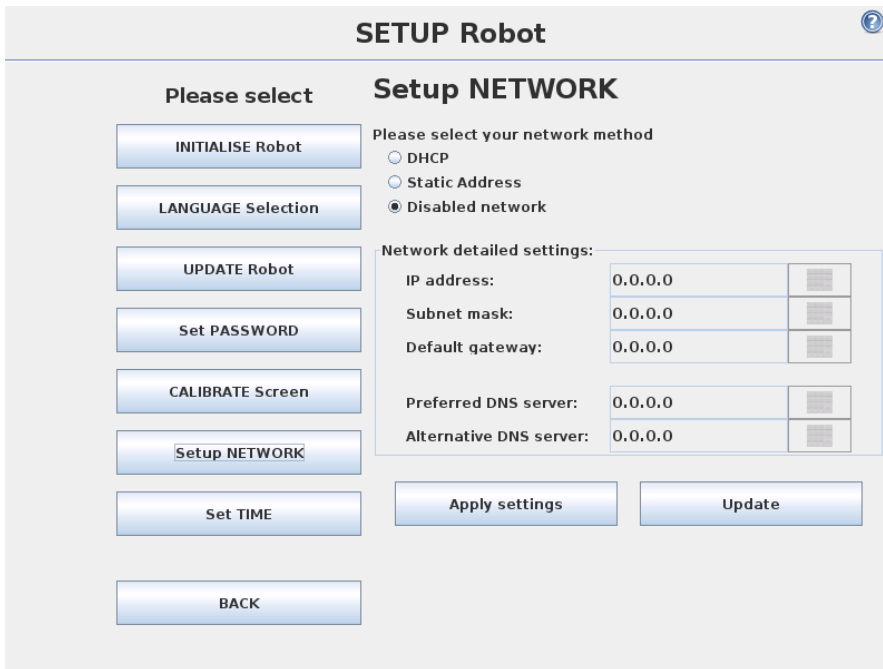
### 5.6 Setup Screen → Calibrate Touch Screen



Calibrating the touch screen. Follow the on-screen instructions to calibrate the touch screen. Preferably use a pointed non-metallic object, such as a

closed pen. Patience and care help achieve a better result.

## 5.7 Setup Screen → Network

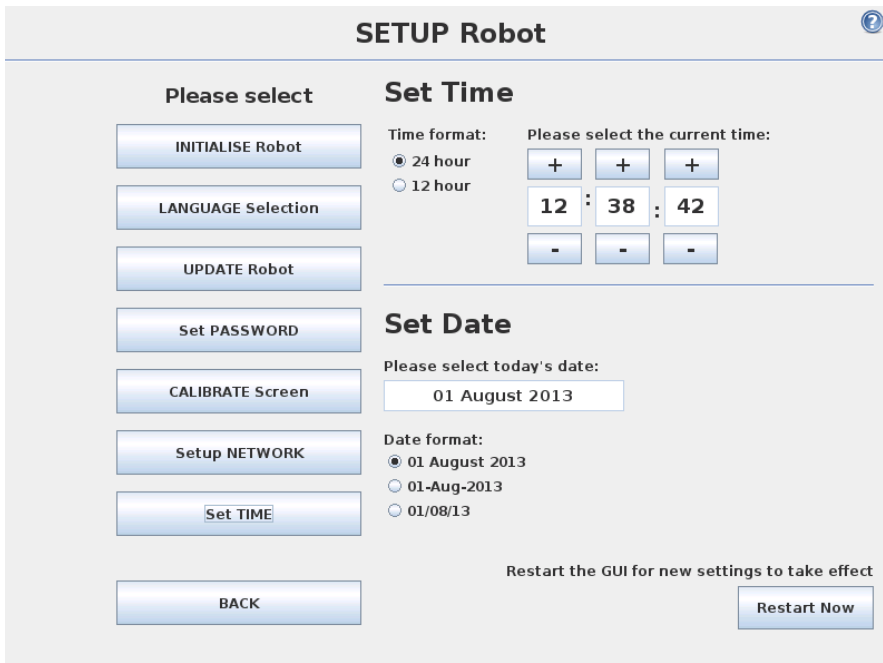


The screenshot shows the 'SETUP Robot' interface with the 'Setup NETWORK' section active. On the left, a 'Please select' menu lists options: INITIALISE Robot, LANGUAGE Selection, UPDATE Robot, Set PASSWORD, CALIBRATE Screen, Setup NETWORK (highlighted), Set TIME, and BACK. The main area is titled 'Setup NETWORK' and contains the following elements:

- Please select your network method:**
  - DHCP
  - Static Address
  - Disabled network
- Network detailed settings:**
  - IP address: 0.0.0.0
  - Subnet mask: 0.0.0.0
  - Default gateway: 0.0.0.0
  - Preferred DNS server: 0.0.0.0
  - Alternative DNS server: 0.0.0.0
- Buttons:** 'Apply settings' and 'Update'.

Panel for setting up the Ethernet network. An Ethernet connection is not necessary for the basic robot functions, and is disabled by default.

## 5.8 Setup Screen → Time



The screenshot shows the 'SETUP Robot' interface with the 'Set Time' section active. On the left, the 'Please select' menu is the same as in the previous screen, with 'Set TIME' highlighted. The main area is titled 'Set Time' and contains the following elements:

- Time format:**
  - 24 hour
  - 12 hour
- Please select the current time:**
  - Buttons: +, +, + (for hours, minutes, seconds)
  - Current time: 12 : 38 : 42
  - Buttons: -, -, - (for hours, minutes, seconds)
- Set Date:**
  - Please select today's date:** 01 August 2013
  - Date format:**
    - 01 August 2013
    - 01-Aug-2013
    - 01/08/13
- Restart Now:** A button labeled 'Restart Now' with the text 'Restart the GUI for new settings to take effect' above it.

Set the time and date for the system and configure the display formats for the clock. The clock is displayed at the top of the **RUN Program** and **PROGRAM Robot** screens. Tapping on it will show the date briefly. The GUI needs to be restarted for changes to take effect.