UNIVERSAL ROBOTS

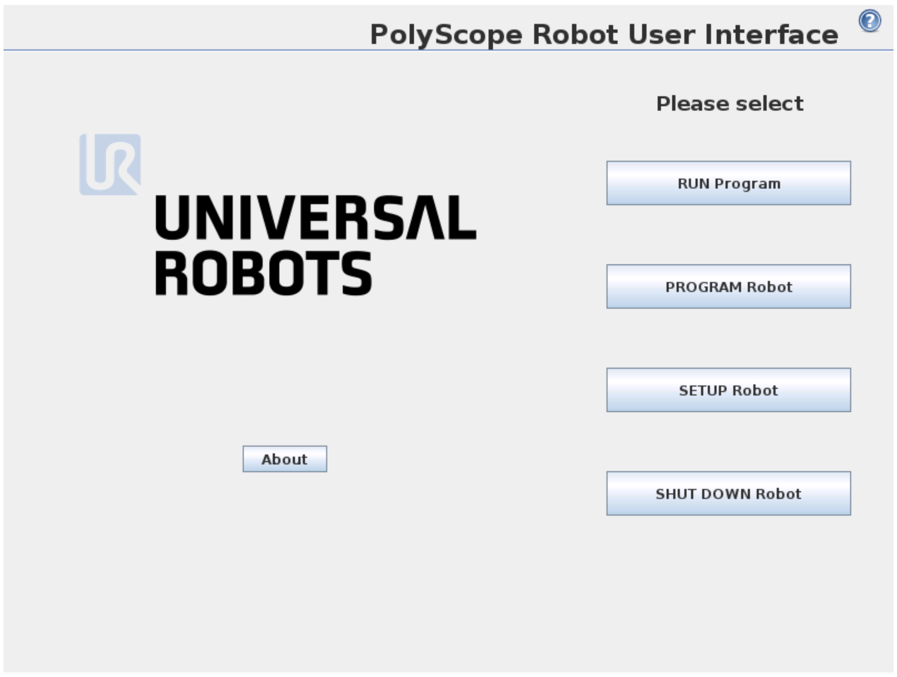
PolyScope Manual

Version 1.7
December 27, 2012

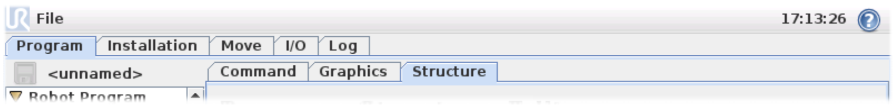
Contents

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.

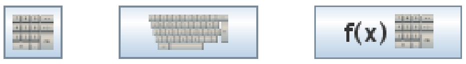


The picture above shows the Welcome Screen. The bluish areas of the screen are buttons that can be pressed by pressing a finger or the backside of a pen against the screen. PolyScope has a hierarchical structure of screens. In the programming environment, the screens are arranged in *tabs*, for easy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

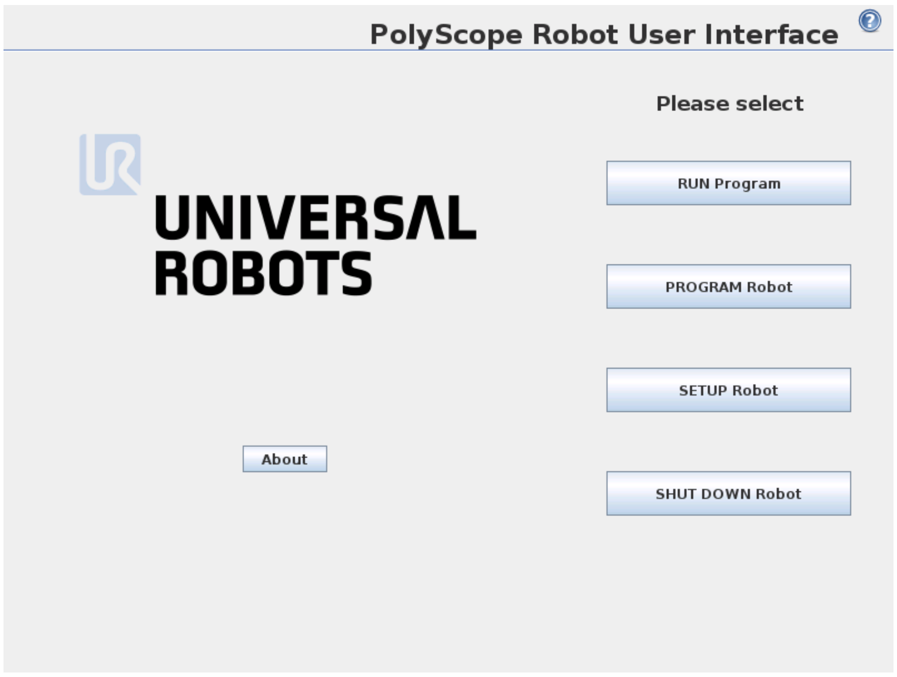
It is possible to connect a mouse and a keyboard to the controller box or the teach pendant; however, this is not required. Almost all text fields are touch-enabled, so touching them launches an on-screen keypad or keyboard. Non-touchable text fields have an editor icon next to them that launches the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

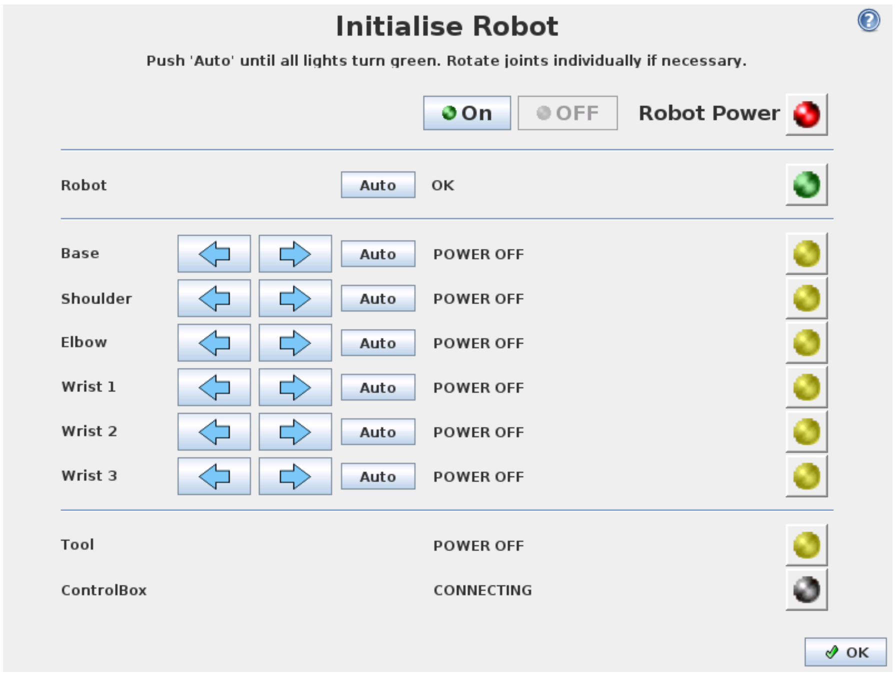
1.1 Welcome Screen



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 via the Internet, request support, calibrate the touch screen, etc.
- **SHUT DOWN Robot:** Shuts down the Controller PC and powers off the robot.

1.2 Initialization Screen



Initialise Robot

Push 'Auto' until all lights turn green. Rotate joints individually if necessary.

Robot	Auto	OK	●
Base	Auto	POWER OFF	●
Shoulder	Auto	POWER OFF	●
Elbow	Auto	POWER OFF	●
Wrist 1	Auto	POWER OFF	●
Wrist 2	Auto	POWER OFF	●
Wrist 3	Auto	POWER OFF	●
Tool		POWER OFF	●
ControlBox		CONNECTING	●

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 joints' positions, the robot needs to move each joint.

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 dosn'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 "Freedrive" 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.

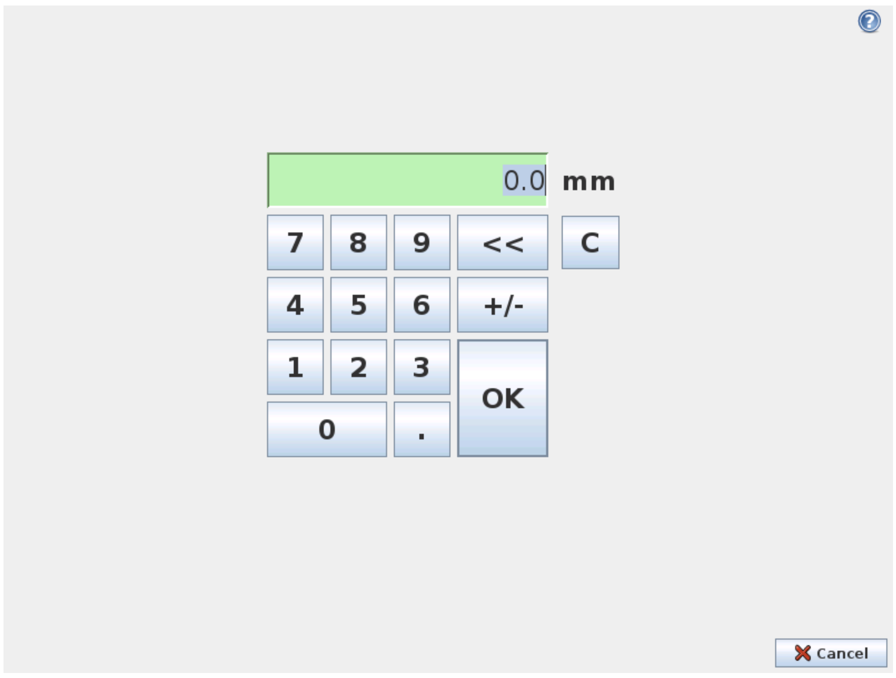
The auto buttons can be pressed individually for each joint, or for the whole robot. Great care should be taken if the robot is touching an obstacle or table, since driving the robot into the obstacle might damage a joint gearbox.

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.

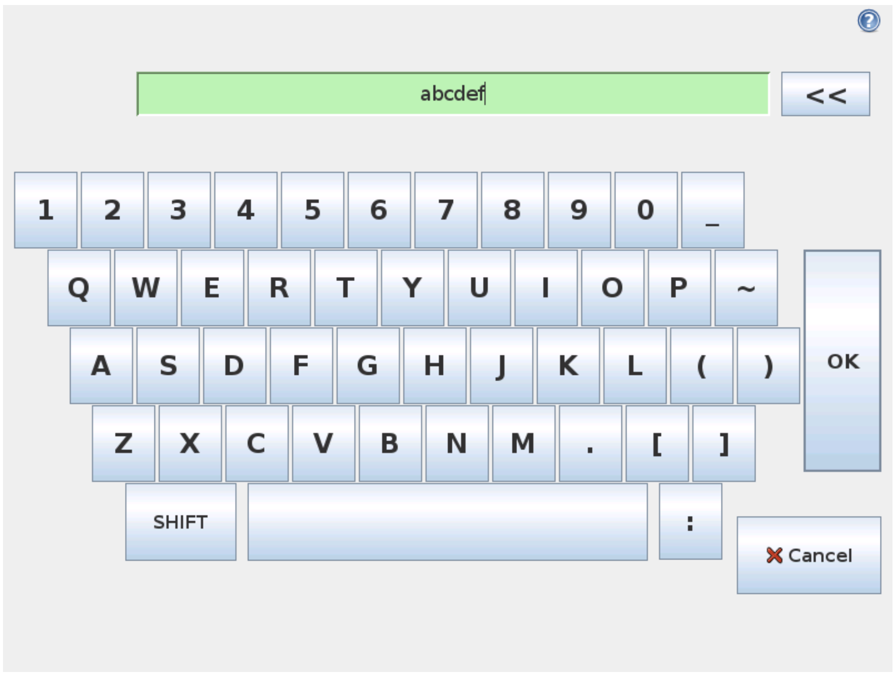
2 On-screen Editors

2.1 On-screen Keypad



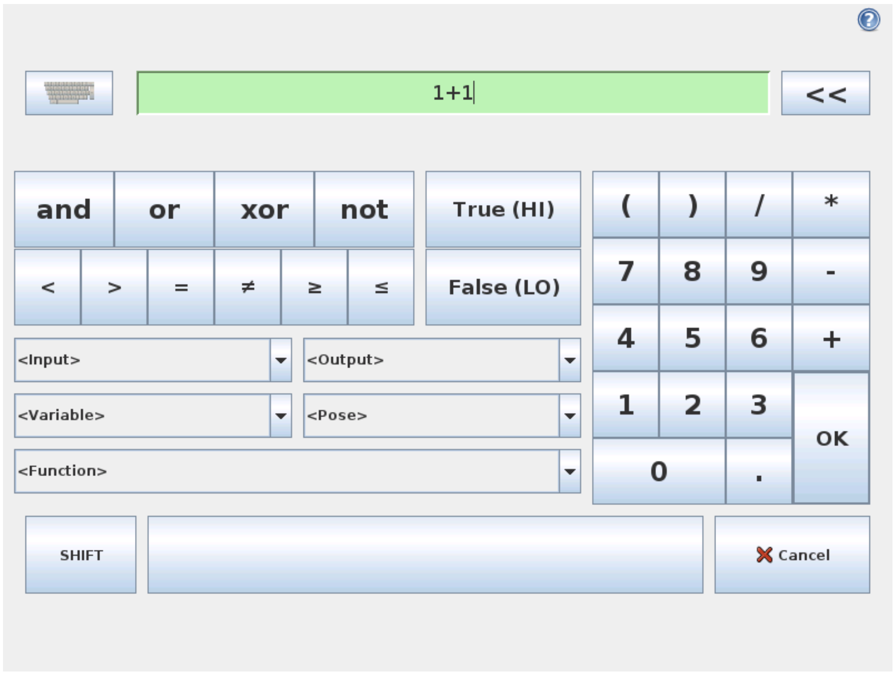
Simple number typing and editing facilities. In many cases, the unit of the typed value is displayed next to the number.

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 $\leq$ 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.

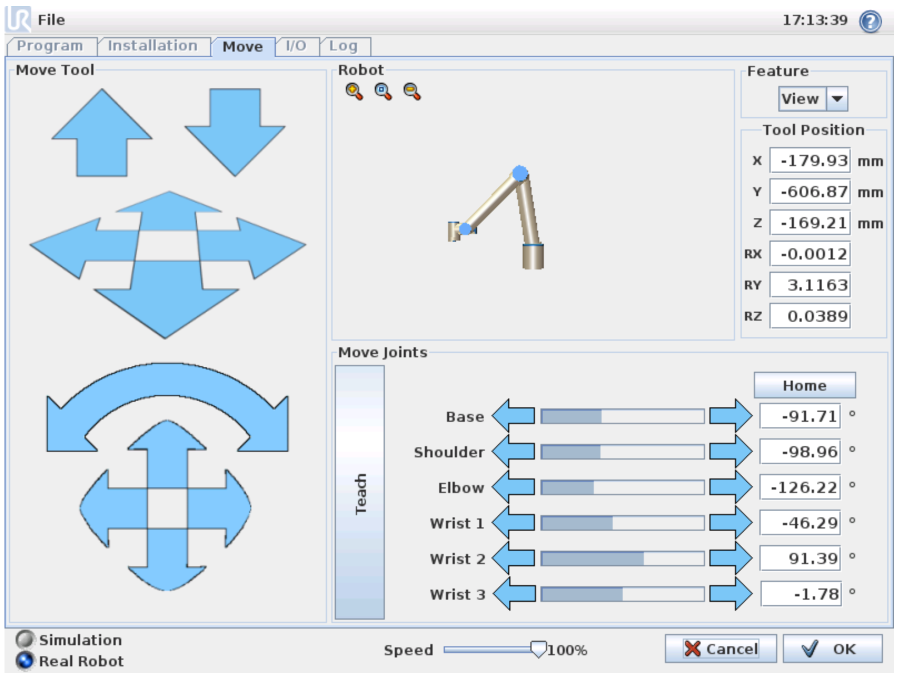
An expression can look like this:

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

3 Robot Control

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.



X	-179.93	mm
Y	-606.87	mm
Z	-169.21	mm
RX	-0.0012	
RY	3.1163	
RZ	0.0389	

Base	-91.71 °
Shoulder	-98.96 °
Elbow	-126.22 °
Wrist 1	-46.29 °
Wrist 2	91.39 °
Wrist 3	-1.78 °

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.

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!*

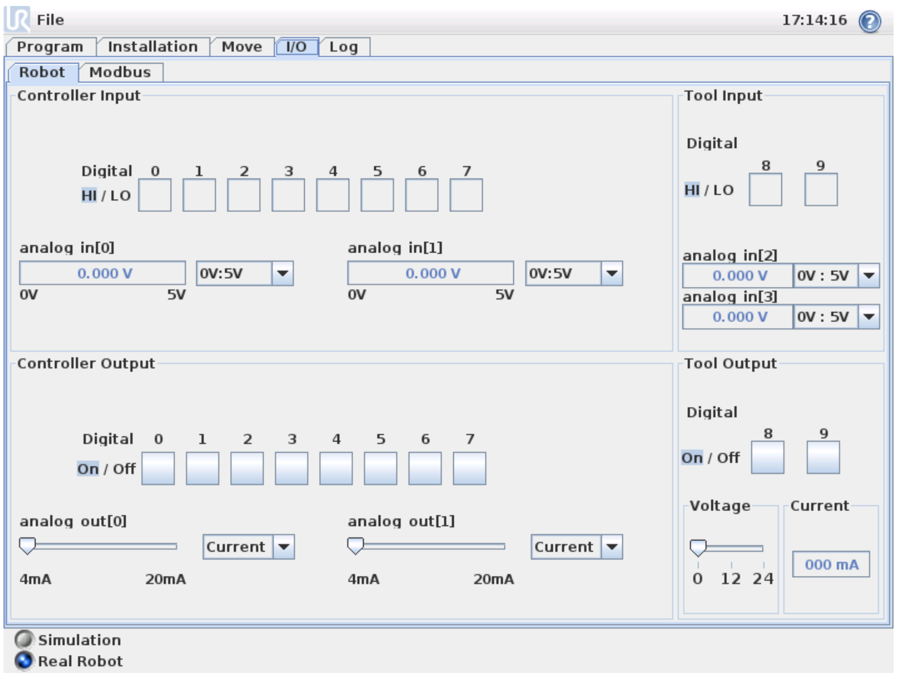
Move Joints

Allows the individual joints to be controlled directly. Each joint can move from $-360°$ to $+360°$, 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 $0°$.

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.

3.2 I/O Tab



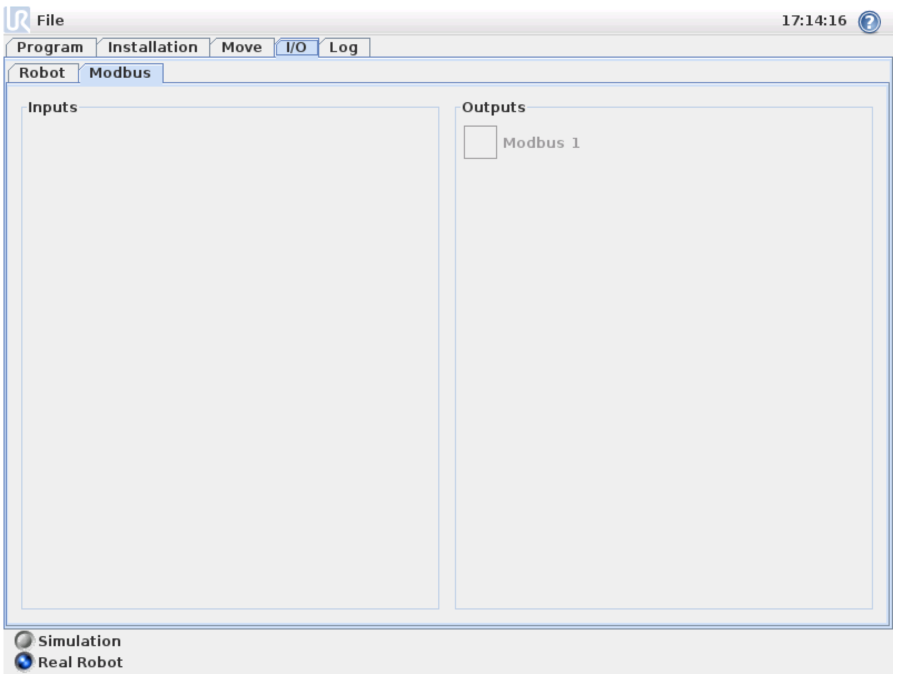
On this screen you can always monitor and set the live I/O signals from/to the robot. The screen displays the current state of the I/O, inluding during program execution. If anything is changed during program execution, the program will stop. At program stop, all output signals will retain their states. 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 I/O

Here, the digital modbus 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

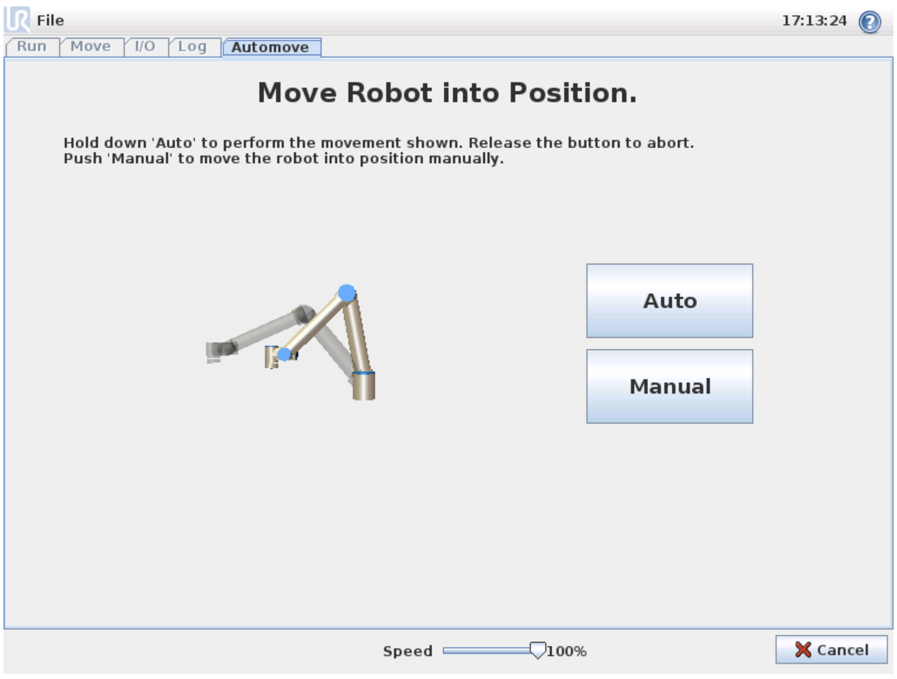
View the state of digital modbus inputs.

Outputs

View and toggle the state of digital modbus 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.



Move Robot into Position.

Hold down 'Auto' to perform the movement shown. Release the button to abort.
Push 'Manual' to move the robot into position manually.

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.

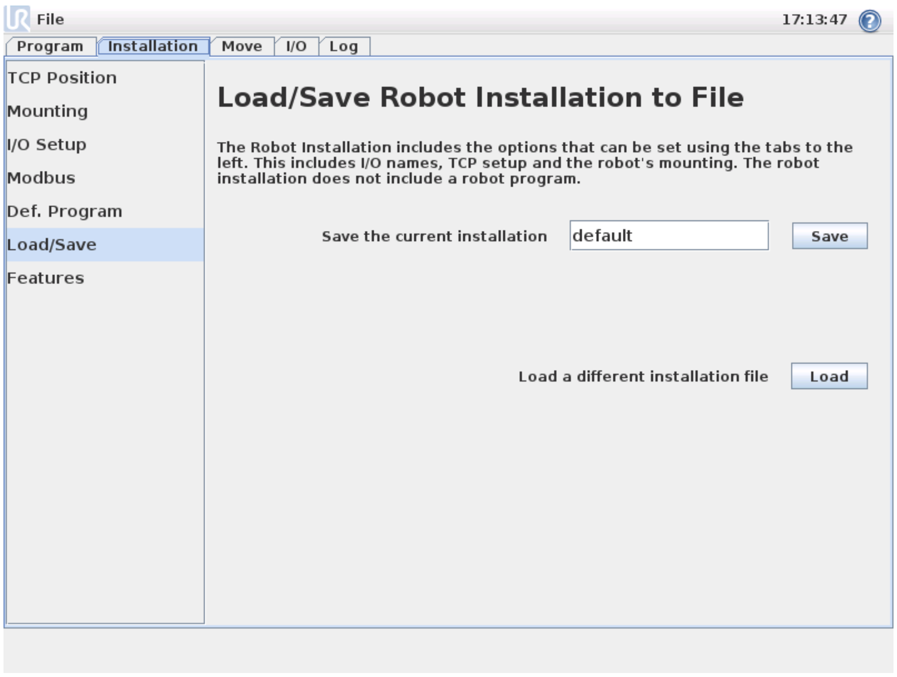
Auto

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



Load/Save Robot Installation to File

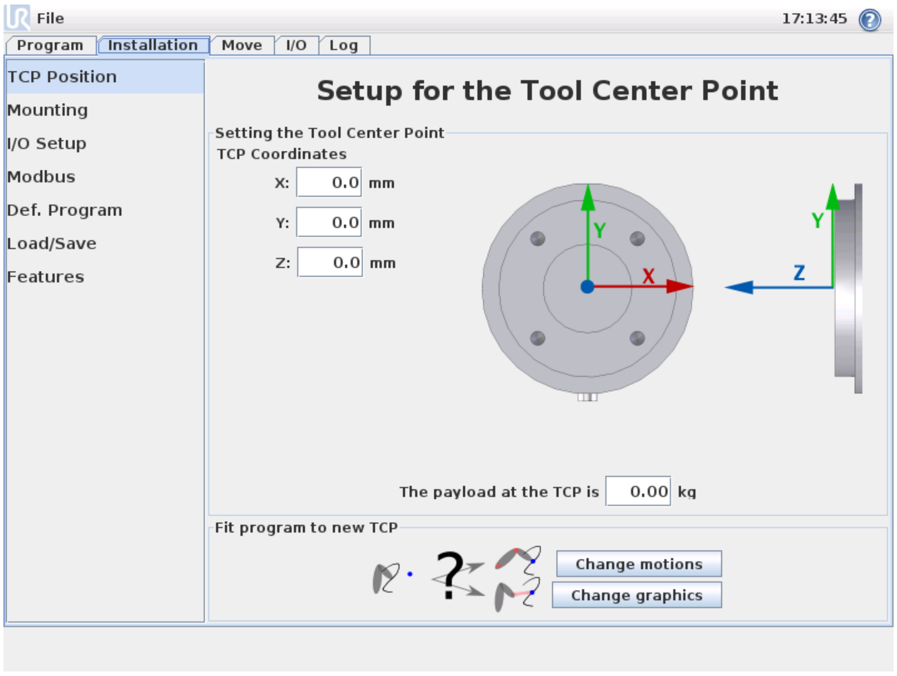
The Robot Installation includes the options that can be set using the tabs to the left. This includes I/O names, TCP setup and the robot's mounting. The robot installation does not include a robot program.

Save the current installation: default

Load a different installation file

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



Setup for the Tool Center Point

Setting the Tool Center Point

TCP Coordinates

X: 0.0 mm

Y: 0.0 mm

Z: 0.0 mm

The payload at the TCP is 0.00 kg

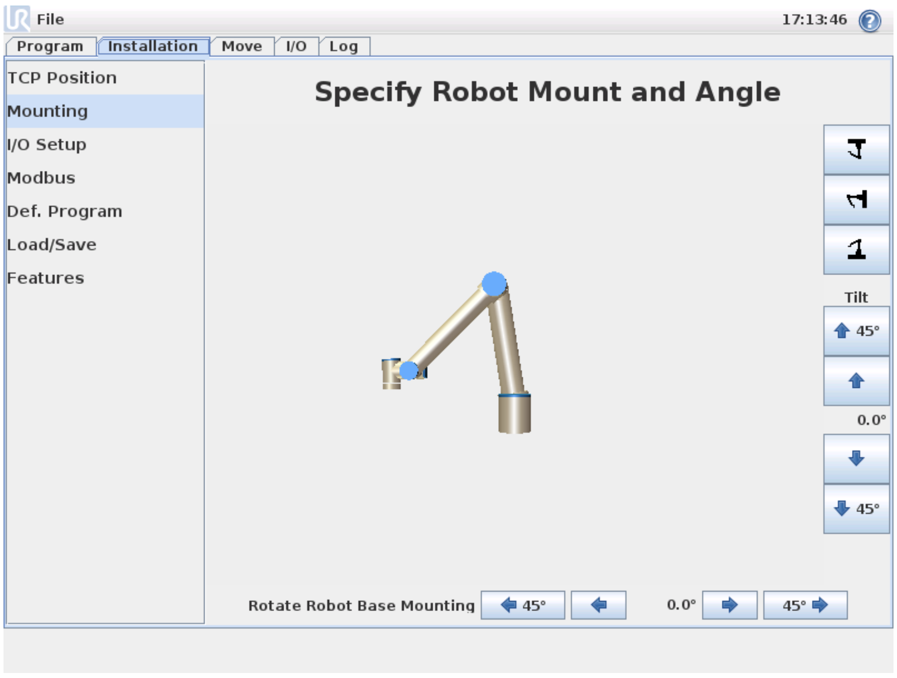
Fit program to new TCP

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.

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



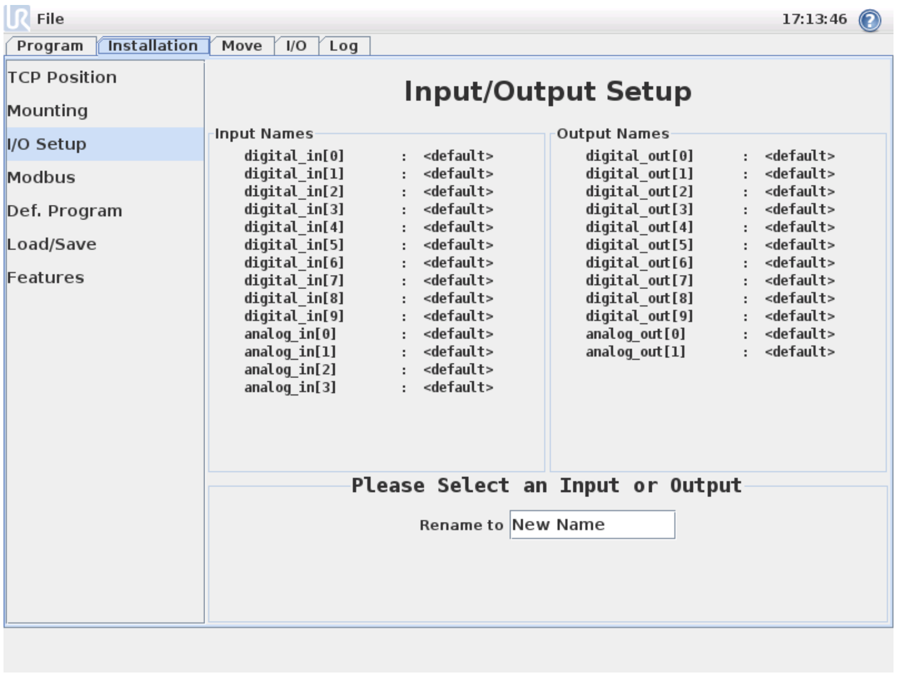
Here the mounting of the robot can be specified. This serves two purposes:

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.

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.

3.8 Installation → I/O Setup



Input/Output Setup

Input Names

```
digital_in[0]    :  <default>
digital_in[1]    :  <default>
digital_in[2]    :  <default>
digital_in[3]    :  <default>
digital_in[4]    :  <default>
digital_in[5]    :  <default>
digital_in[6]    :  <default>
digital_in[7]    :  <default>
digital_in[8]    :  <default>
digital_in[9]    :  <default>
analog_in[0]     :  <default>
analog_in[1]     :  <default>
analog_in[2]     :  <default>
analog_in[3]     :  <default>
```

Output Names

```
digital_out[0]   :  <default>
digital_out[1]   :  <default>
digital_out[2]   :  <default>
digital_out[3]   :  <default>
digital_out[4]   :  <default>
digital_out[5]   :  <default>
digital_out[6]   :  <default>
digital_out[7]   :  <default>
digital_out[8]   :  <default>
digital_out[9]   :  <default>
analog_out[0]    :  <default>
analog_out[1]    :  <default>
```

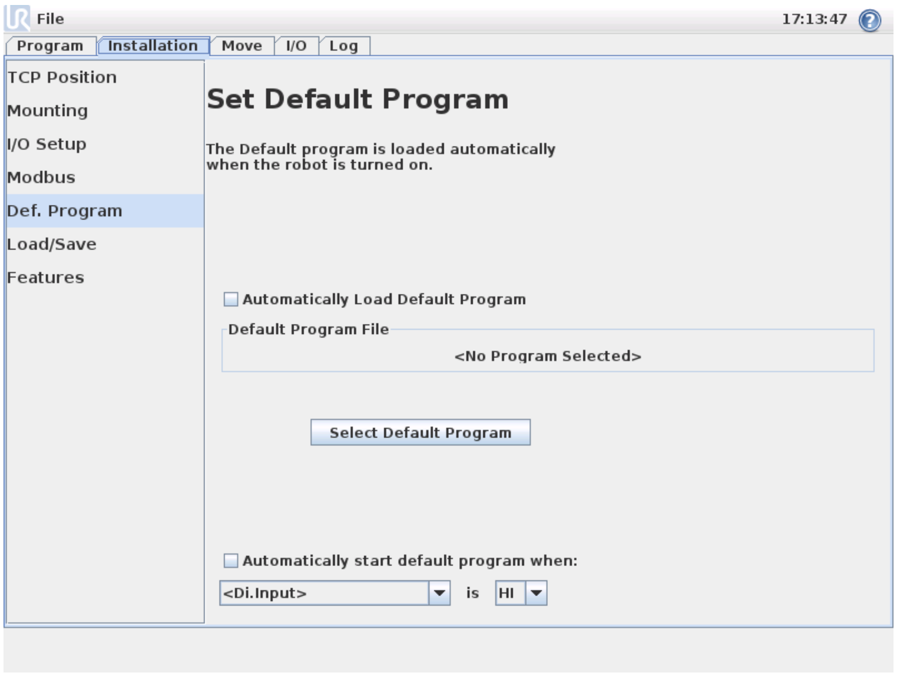
Please Select an Input or Output

Rename to: New Name

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 → Default Program



Set Default Program

The Default program is loaded automatically
when the robot is turned on.

☐ Automatically Load Default Program

Default Program File

<No Program Selected>

Select Default Program

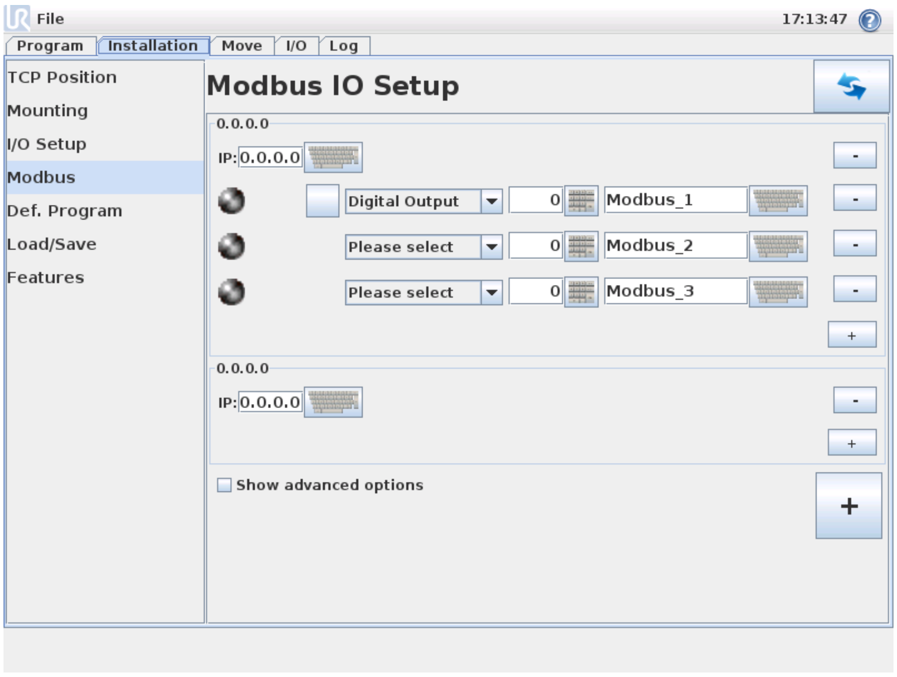
☐ Automatically start default program when:

<Di.Input> is HI

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

3.10 Modbus I/O Setup

Here, the modbus I/O signals can be set up. Modbus units on specific IP addresses can be added/deleted and input/output signals (registers or digital) on these units can be added/deleted as well. Each signal must be supplied with a unique name. However, several signals with different names may reference the same modbus signal, but the user is advised to avoid this. Below, the different buttons and fields are explained in detail.



Modbus IO Setup

0.0.0.0

IP:0.0.0.0

Digital Output	0	Modbus_1	-
Please select	0	Modbus_2	-
Please select	0	Modbus_3	-

0.0.0.0

IP:0.0.0.0

☐ Show advanced options

Refresh

Push this button to refresh the connectivity status of all modbus signals in the current installation.

Add unit

Push this button to add a new modbus unit to the robot installation.

Delete unit

Push this button to delete the modbus unit and all signals added to the 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 robot installation which can be found on the corresponding modbus unit.

Delete signal

Push this button to delete the modbus signal from the installation.

Set signal type

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

- **Digital input:** A digital input 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 is a one-bit quantity which can be set to either high or low according to the configuration of the corresponding modbus terminal. Until the value of this output has been set by the user, the value is read from the unit. This means that function code 0x01 (Read Coils) is used until the output has been set, and then when either 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. Until the value of the register has been set, the value of it is simply read. This means that function code 0x03 (Read Holding Registers) is used until the signal is set either by a robot program or by specifying a signal value in the "set signal value" field, and after that function code 0x06 (Write Single Register) is used onwards.

Set signal address

This field shows the address of the signal. Use the on-screen keypad to choose a different address. Valid addresses depends on the manufacturer and configuration of the modbus unit. It is necessary to have a good understanding of the internal memory map of the Modbus controller in order to make sure the signal address actually corresponds to what is the intention of the signal. Especially, it might be worth verifying the meaning of a signal address when different function codes are used. See 3.10 for a description of the function codes associated with the different signal types.

Set signal name

Using the on-screen keyboard, the user may give the signal a meaningful name which will provide a more intuitive programming of the robot using the signal. Signal names are unique which means that two signals cannot be assigned the same name. Signal names are restricted to be composed of no more than 10 characters.

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).

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 Modbus controller 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 that you consult the manual of your Modbus devices to verify their functionality with a changed slave address.

3.11 Features

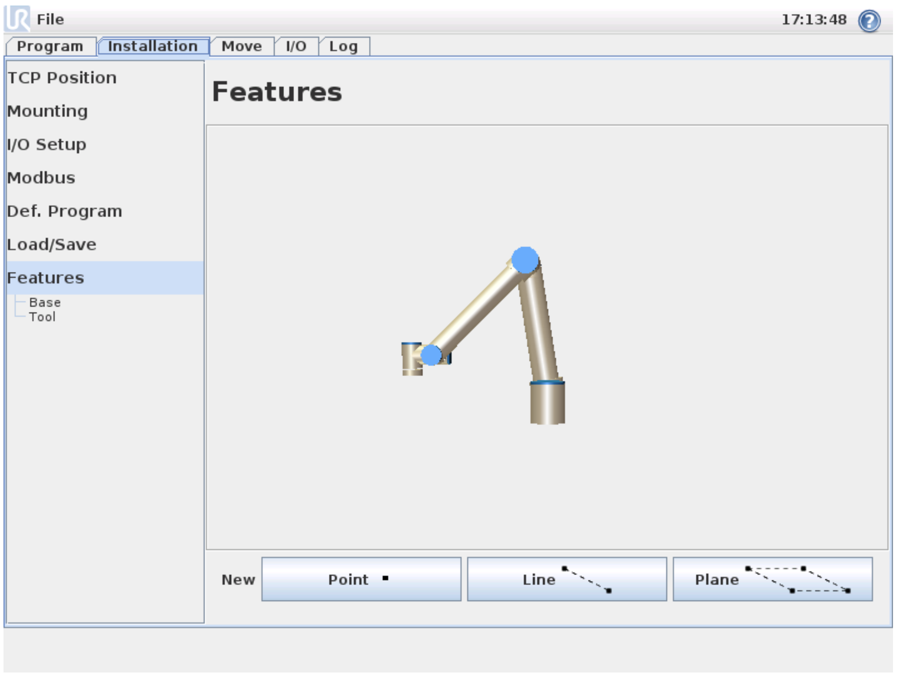
Customers that buy industrial robots generally want to be able to control or manipulate a robot, and to program the robot, relative to various objects and boundaries in the surroundings of the robot, such as machines, objects or blanks, fixtures, conveyers, pallets or vision systems. Traditionally, this 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 ammount 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 suceeded 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

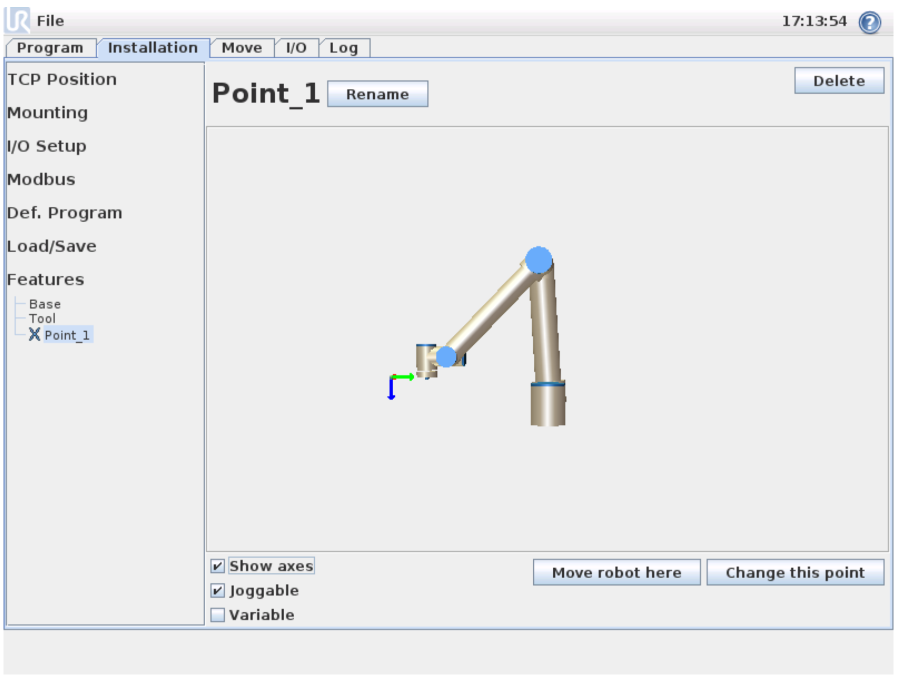
Use this button to set or change the selected feature. The Move screen will appear and a new position of the feature can be set.

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.

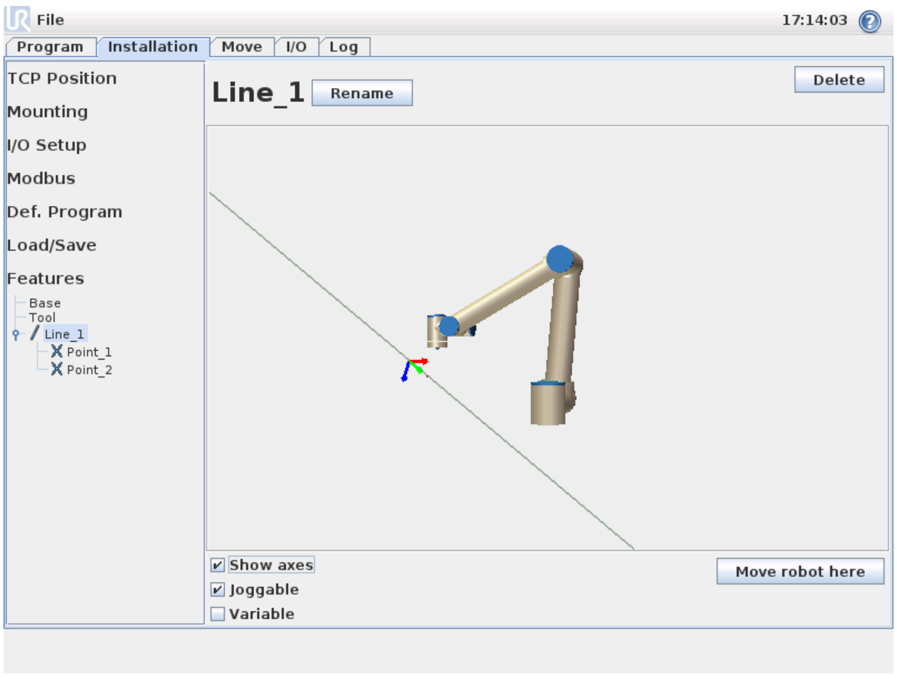
Add Point

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.



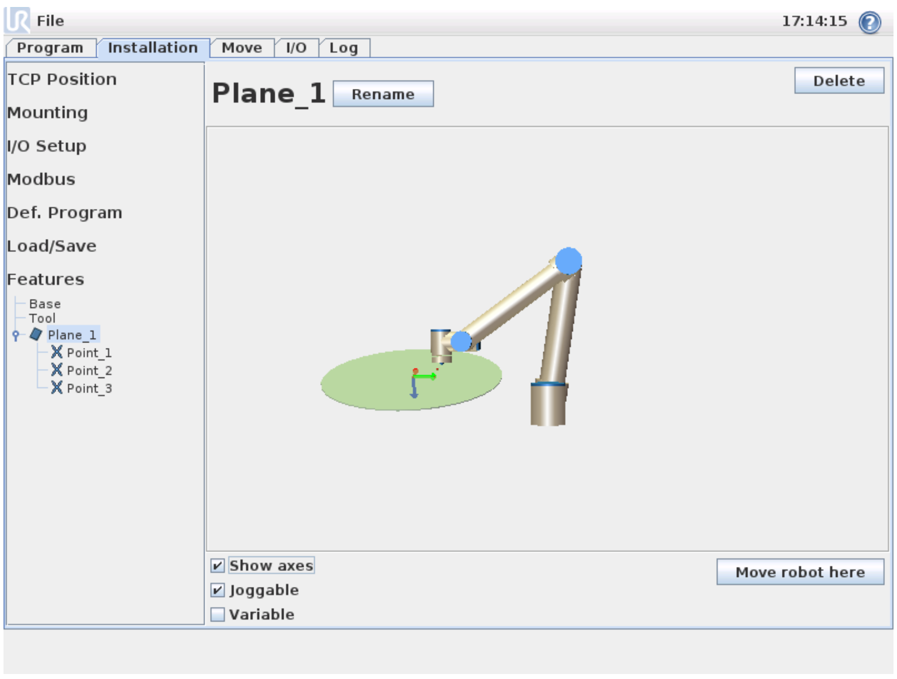
Add Line

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.

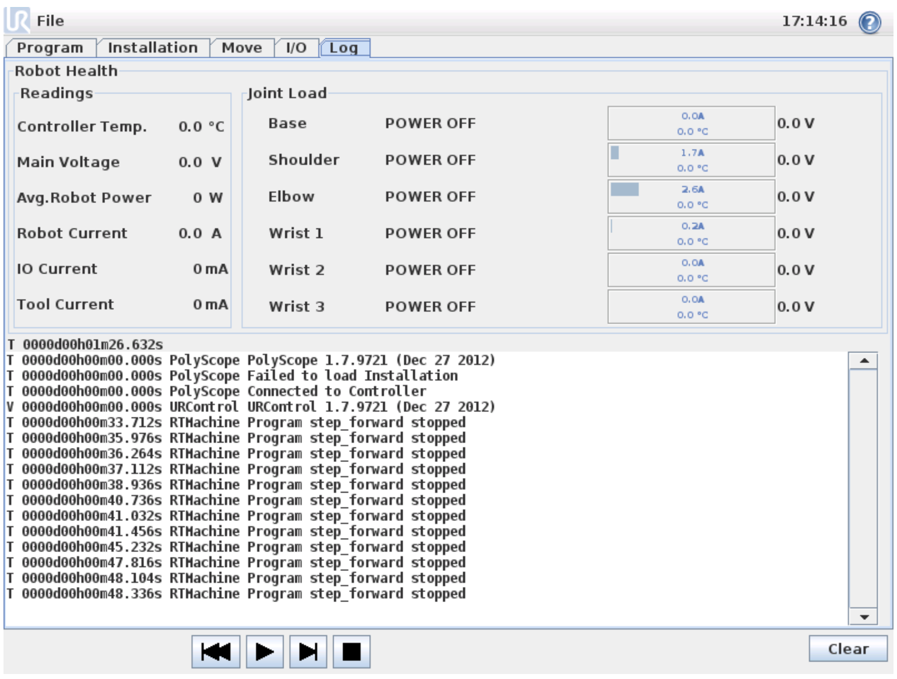


Add Plane

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.



3.12 Log Tab



Robot Health

Readings

Controller Temp.	0.0 °C
Main Voltage	0.0 V
Avg.Robot Power	0 W
Robot Current	0.0 A
IO Current	0 mA
Tool Current	0 mA

Joint Load

Base	POWER OFF	0.0A 0.0 °C	0.0 V
Shoulder	POWER OFF	1.7A 0.0 °C	0.0 V
Elbow	POWER OFF	2.6A 0.0 °C	0.0 V
Wrist 1	POWER OFF	0.2A 0.0 °C	0.0 V
Wrist 2	POWER OFF	0.0A 0.0 °C	0.0 V
Wrist 3	POWER OFF	0.0A 0.0 °C	0.0 V

```
T 0000d00h01m26.632s
T 0000d00h00m00.000s PolyScope PolyScope 1.7.9721 (Dec 27 2012)
T 0000d00h00m00.000s PolyScope Failed to load Installation
T 0000d00h00m00.000s PolyScope Connected to Controller
V 0000d00h00m00.000s URControl URControl 1.7.9721 (Dec 27 2012)
T 0000d00h00m33.712s RTMachine Program step_forward stopped
T 0000d00h00m35.976s RTMachine Program step_forward stopped
T 0000d00h00m36.264s RTMachine Program step_forward stopped
T 0000d00h00m37.112s RTMachine Program step_forward stopped
T 0000d00h00m38.936s RTMachine Program step_forward stopped
T 0000d00h00m40.736s RTMachine Program step_forward stopped
T 0000d00h00m41.032s RTMachine Program step_forward stopped
T 0000d00h00m41.456s RTMachine Program step_forward stopped
T 0000d00h00m45.232s RTMachine Program step_forward stopped
T 0000d00h00m47.816s RTMachine Program step_forward stopped
T 0000d00h00m48.104s RTMachine Program step_forward stopped
T 0000d00h00m48.336s RTMachine Program step_forward stopped
```

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 temperaure 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.

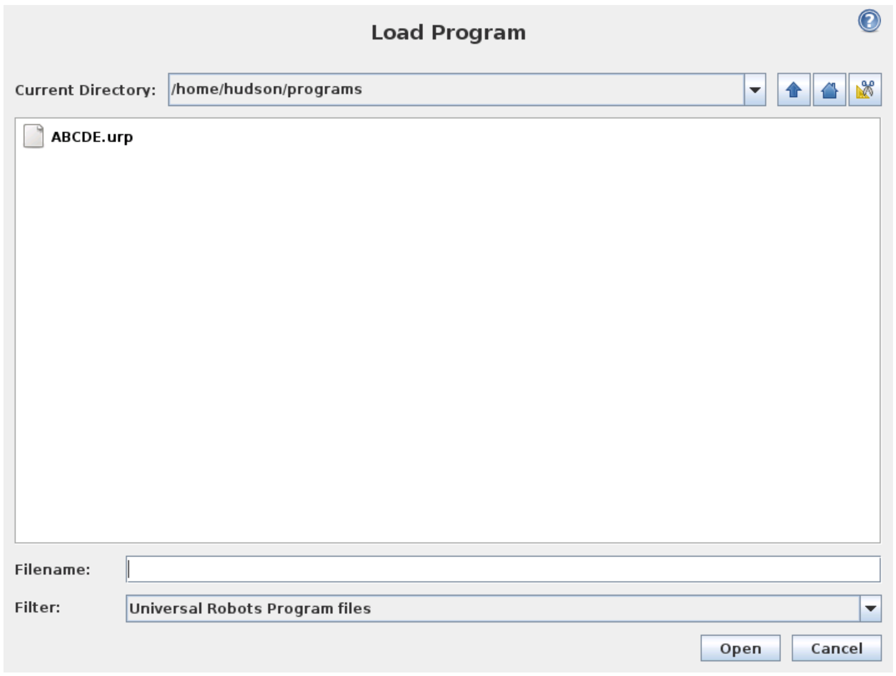
3.13 Load Screen

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


Load Program
Current Directory: /home/hudson/programs
ABCDE.urp
Filename:
Filter: Universal Robots Program files
Open
Cancel

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 3.13.

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.

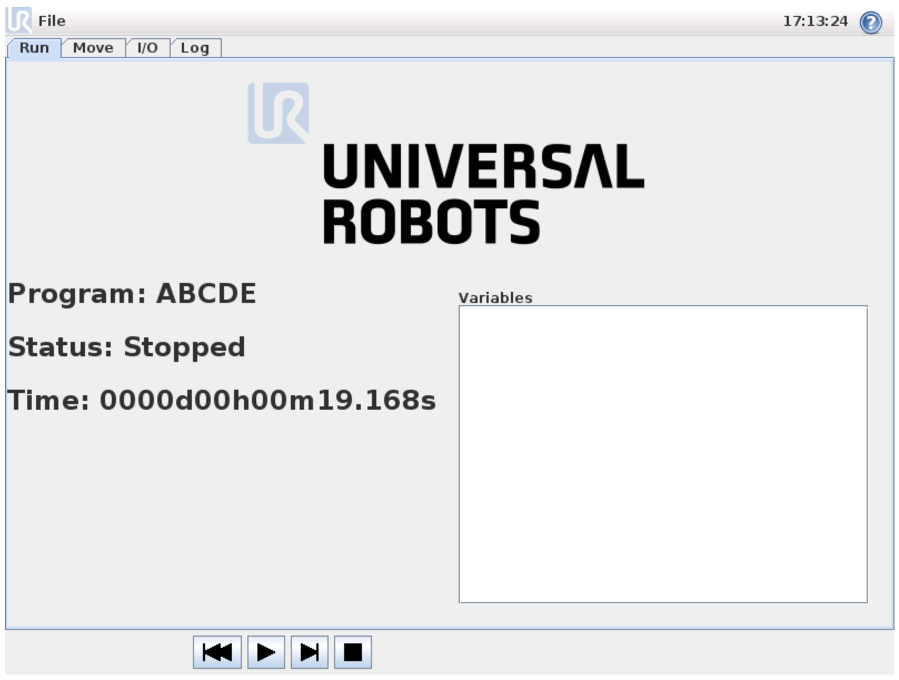
Open button Clicking on the Open button, will open the currently selected file and return to the previous screen.

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.

3.14 Run Tab



Program: ABCDE

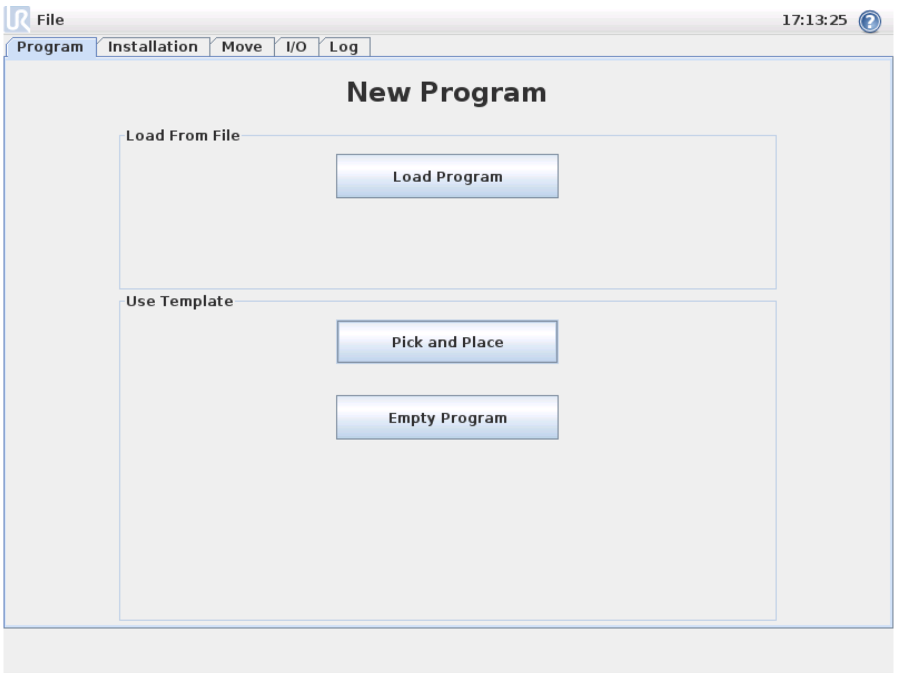
Status: Stopped

Time: 0000d00h00m19.168s

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.

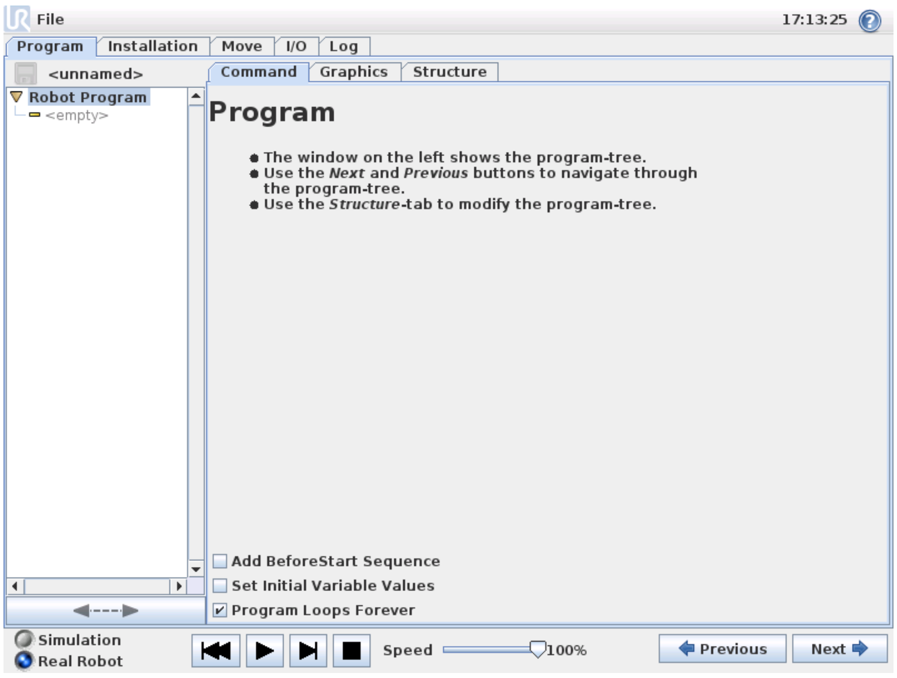
4 Programming

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



Program

- The window on the left shows the program-tree.
- Use the *Next* and *Previous* buttons to navigate through the program-tree.
- Use the *Structure*-tab to modify the program-tree.

☐ Add BeforeStart Sequence
☐ Set Initial Variable Values
☑ Program Loops Forever

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

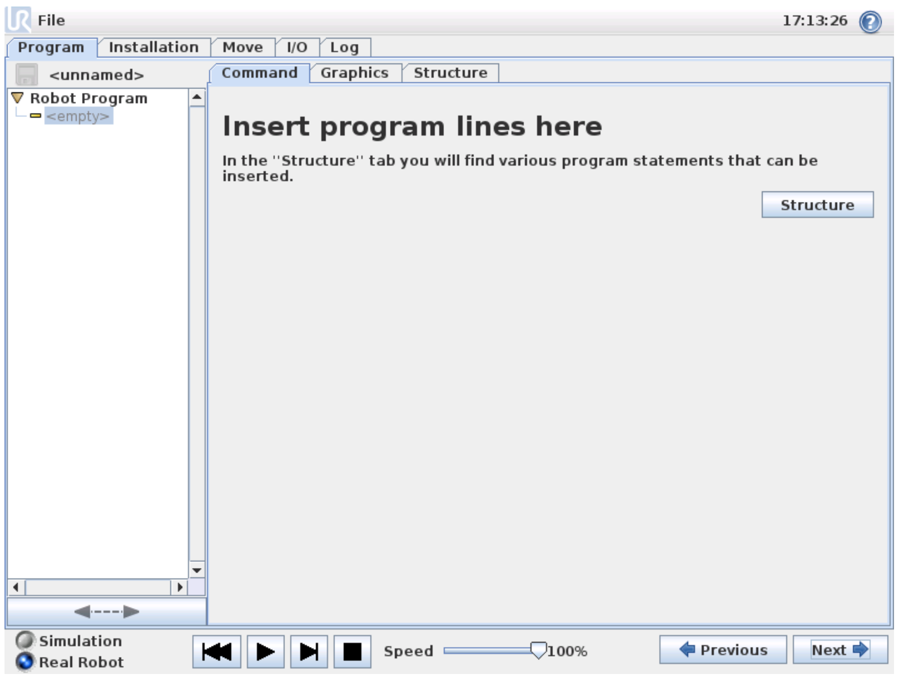
Structure tab, described in section 4.28. The program name is shown directly above the command list, with a small disk icon that can be clicked to quickly save the program.

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>

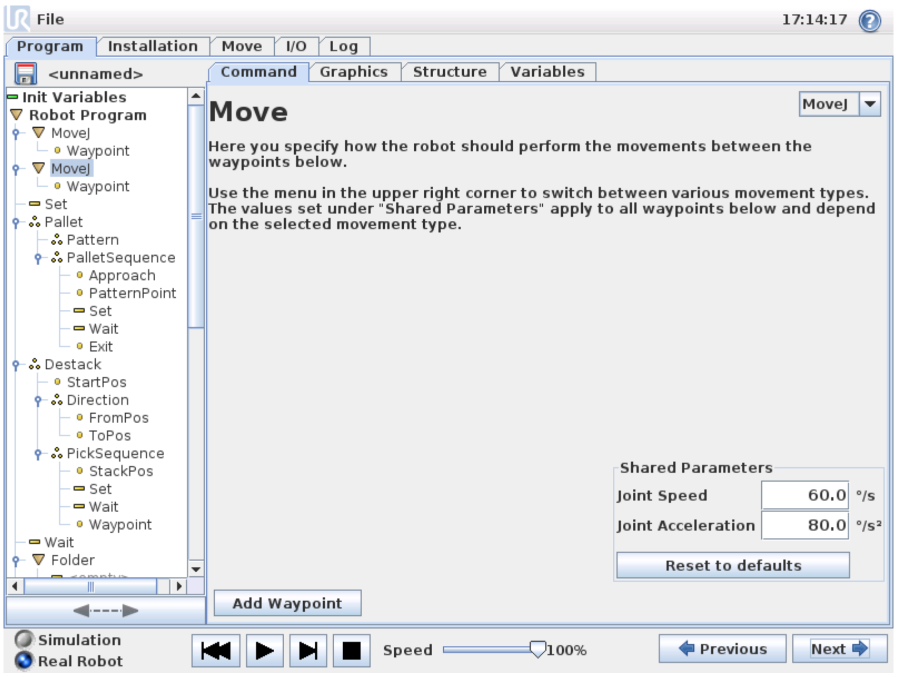


Insert program lines here

In the "Structure" tab you will find various program statements that can be inserted.

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



Move

Here you specify how the robot should perform the movements between the waypoints below.

Use the menu in the upper right corner to switch between various movement types. The values set under "Shared Parameters" apply to all waypoints below and depend on the selected movement type.

Shared Parameters

Joint Speed	60.0	°/s
Joint Acceleration	80.0	°/s²

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 which 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 waypoint. 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.

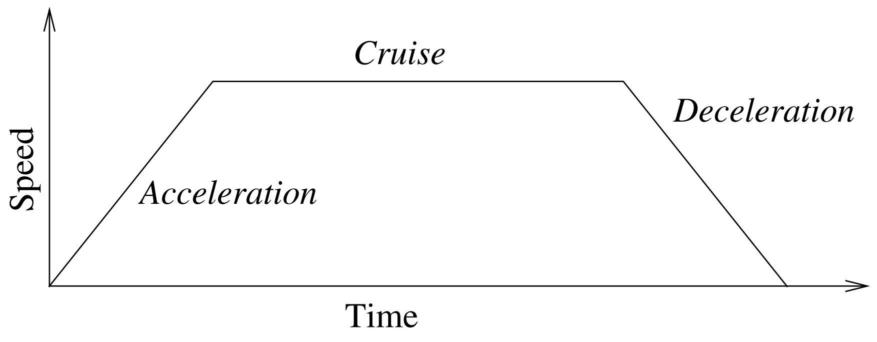
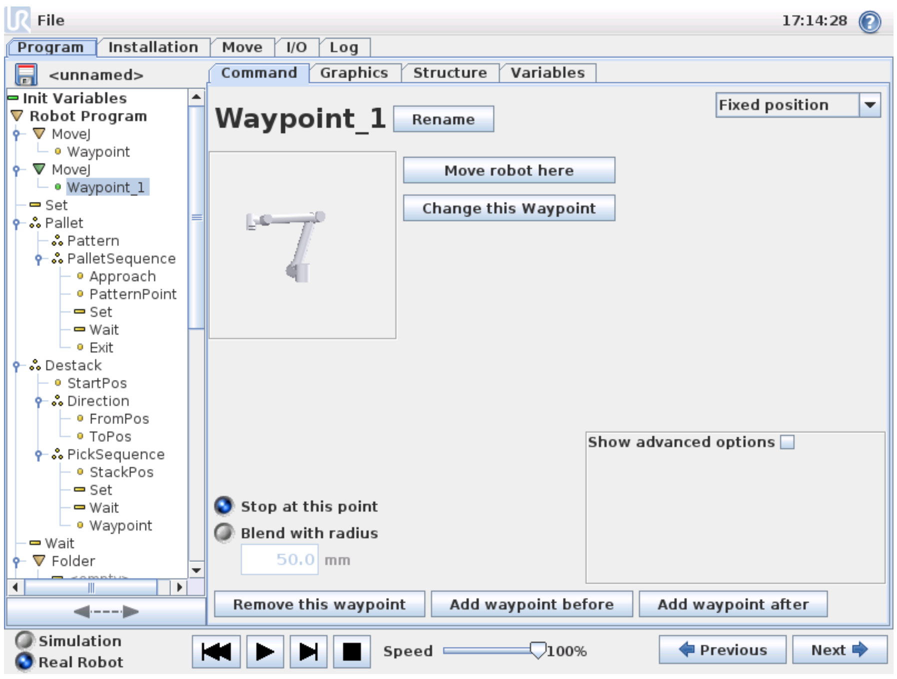


Figure 1: Speed profile for a motion. The curve is divided into three segments: *acceleration*, *cruise* and *deceleration*. The level of the *cruise* phase is given by the speed setting of the motion, while the steepness of the *acceleration* and *deceleration* phases is given by the acceleration parameter.

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.

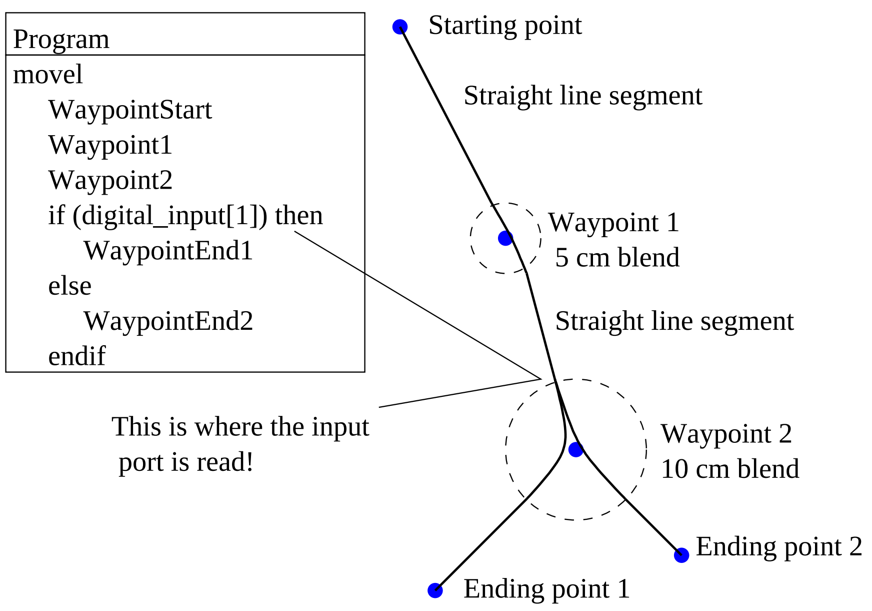
Blend radius

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 waypont. A stop point is a waypoint with a blend radius of $0.0mm$.

Note on I/O Timing

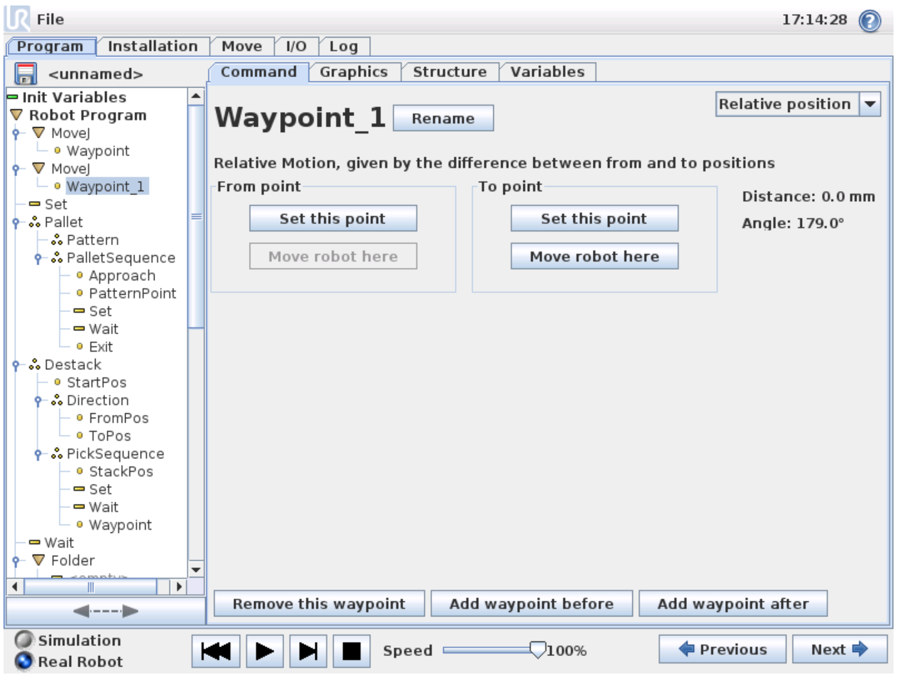
If a waypoint is a stop point with an I/O command as the next command, the I/O command is executed when the robot stops at the waypoint. However, if the waypoint has a blend radius, the following I/O command is executed when the robot enters the blend.

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



Relative Motion, given by the difference between from and to positions

From point: Set this point / Move robot here

To point: Set this point / Move robot here

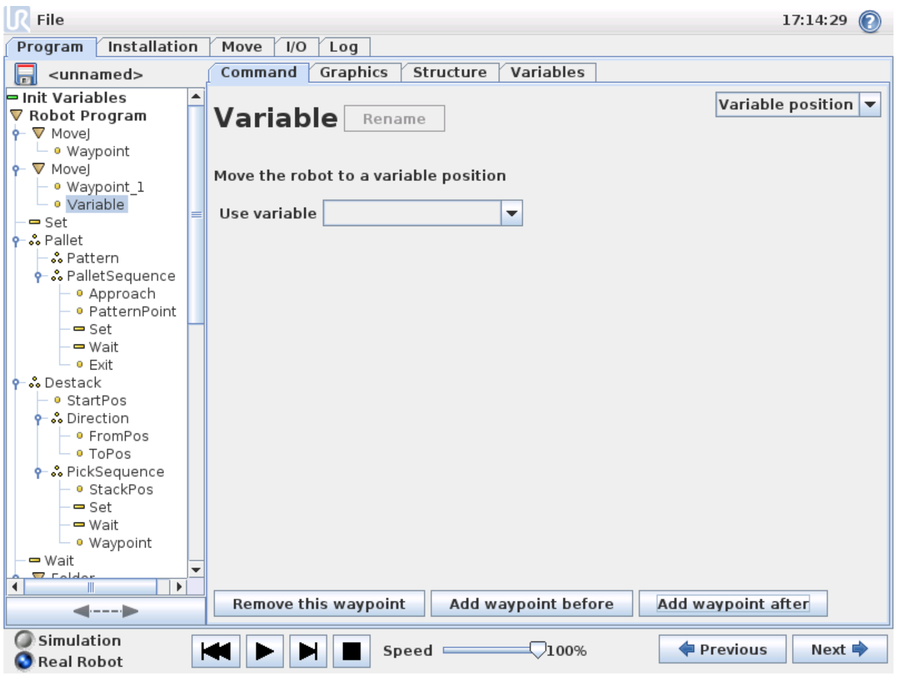
Distance: 0.0 mm

Angle: 179.0°

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



Move the robot to a variable position

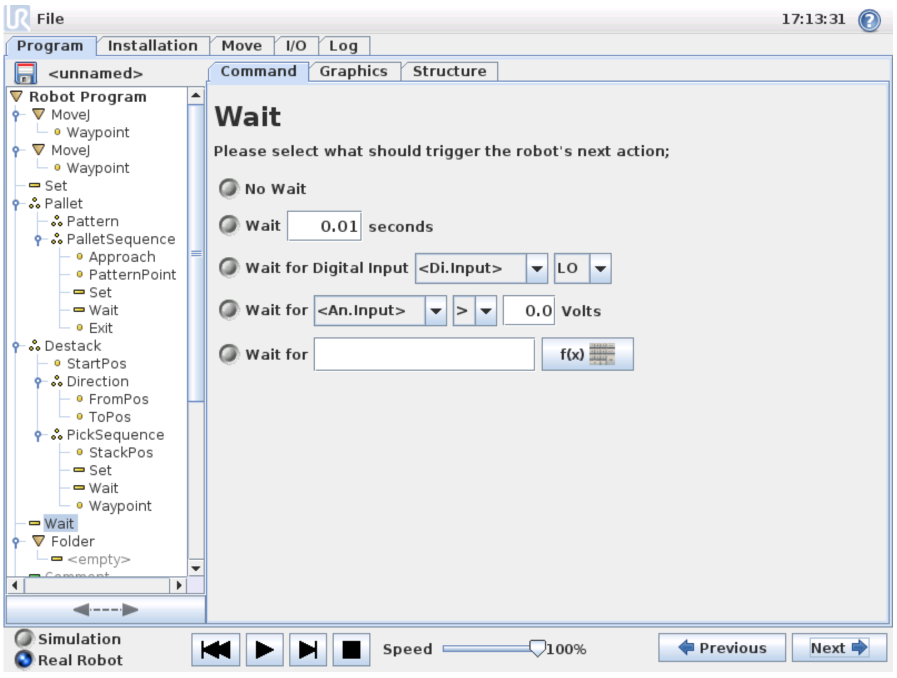
Use variable

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



Wait

Please select what should trigger the robot's next action;

No Wait

Wait 0.01 seconds

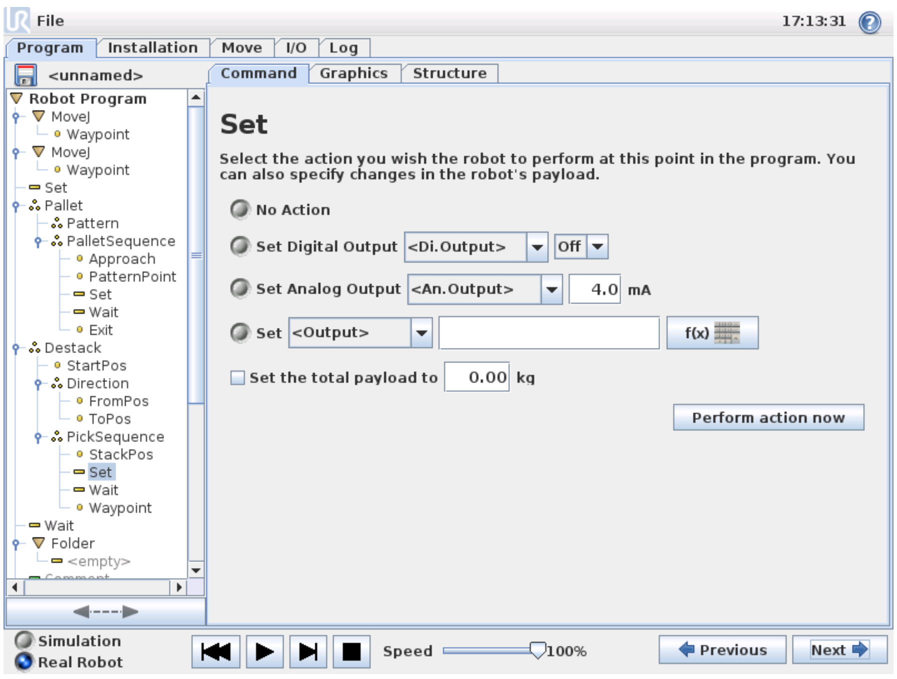
Wait for Digital Input <Di.Input> LO

Wait for <An.Input> > 0.0 Volts

Wait for f(x)

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

4.10 Program → Command Tab, Action



File 17:13:31

Program | Installation | Move | I/O | Log

<unnamed>

Command | Graphics | Structure

Robot Program
- MoveJ
  - Waypoint
- MoveJ
  - Waypoint
- Set
- Pallet
  - Pattern
  - PalletSequence
    - Approach
    - PatternPoint
    - Set
    - Wait
    - Exit
- Destack
  - StartPos
  - Direction
    - FromPos
    - ToPos
  - PickSequence
    - StackPos
    - Set
    - Wait
    - Waypoint
- Wait
- Folder
  - <empty>

Set

Select the action you wish the robot to perform at this point in the program. You can also specify changes in the robot's payload.

- No Action
- Set Digital Output <Di.Output> Off
- Set Analog Output <An.Output> 4.0 mA
- Set <Output> f(x)

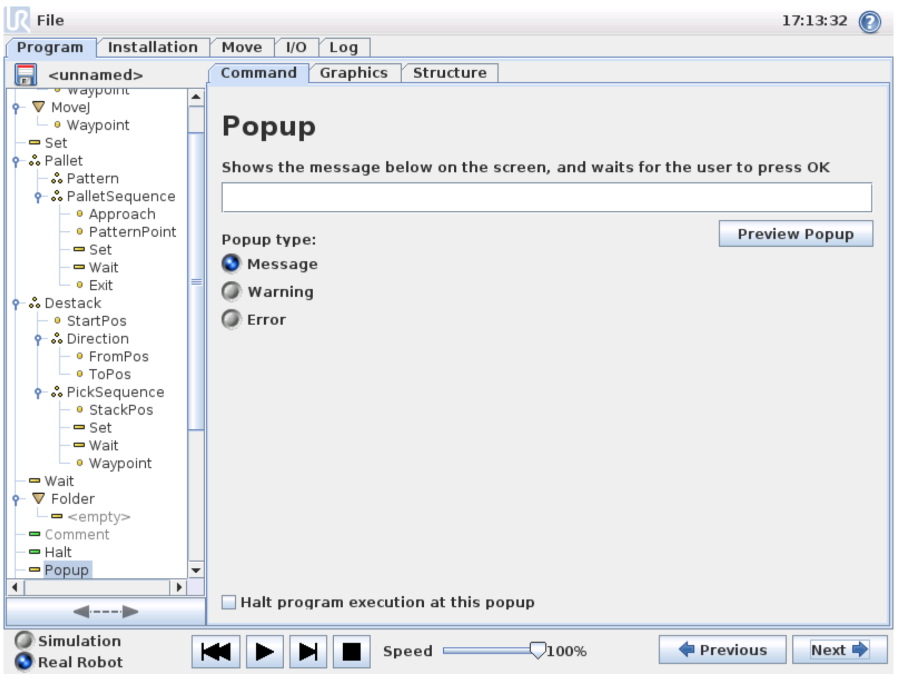
☐ Set the total payload to 0.00 kg

Perform action now

Simulation | Real Robot | Speed 100% | Previous | Next

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 neccesary 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



File 17:13:32

Program | Installation | Move | I/O | Log

<unnamed>

Command | Graphics | Structure

- Waypoint
- MoveJ
  - Waypoint
- Set
- Pallet
  - Pattern
  - PalletSequence
    - Approach
    - PatternPoint
    - Set
    - Wait
    - Exit
- Destack
  - StartPos
  - Direction
    - FromPos
    - ToPos
  - PickSequence
    - StackPos
    - Set
    - Wait
    - Waypoint
- Wait
- Folder
  - <empty>
- Comment
- Halt
- Popup

Popup

Shows the message below on the screen, and waits for the user to press OK

Preview Popup

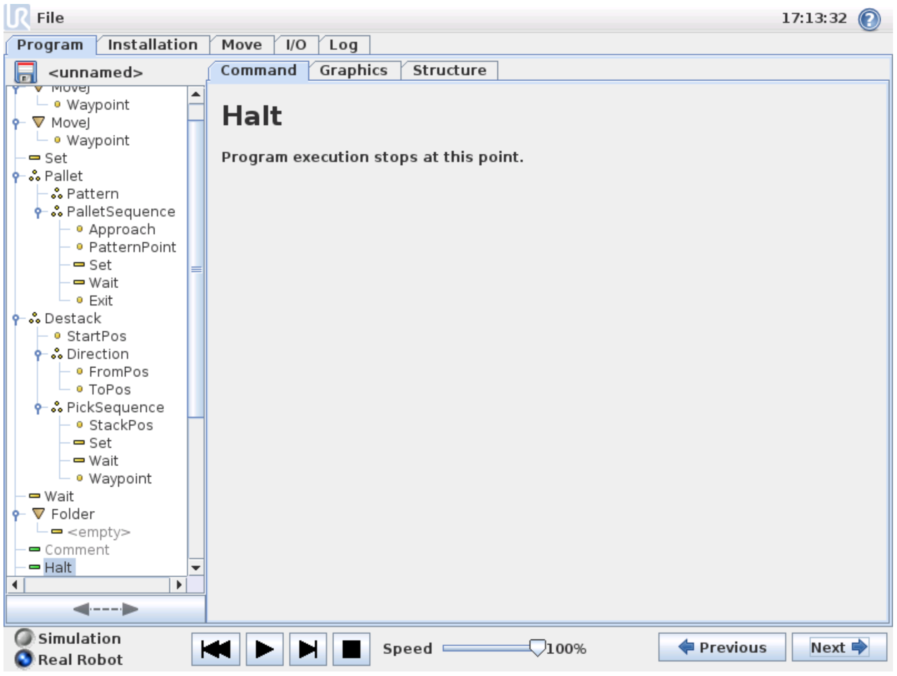
Popup type:
- [✓ Message]
- Warning
- Error

☐ Halt program execution at this popup

Simulation | Real Robot | Speed 100% | Previous | Next

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

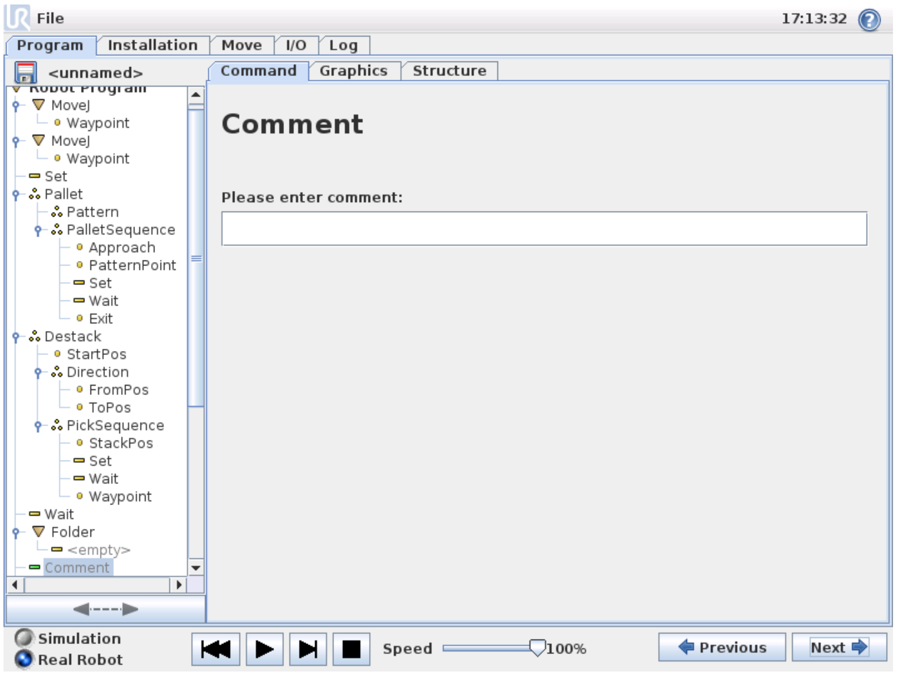


Halt

Program execution stops at this point.

The program execution stops at this point.

4.13 Program → Command Tab, Comment

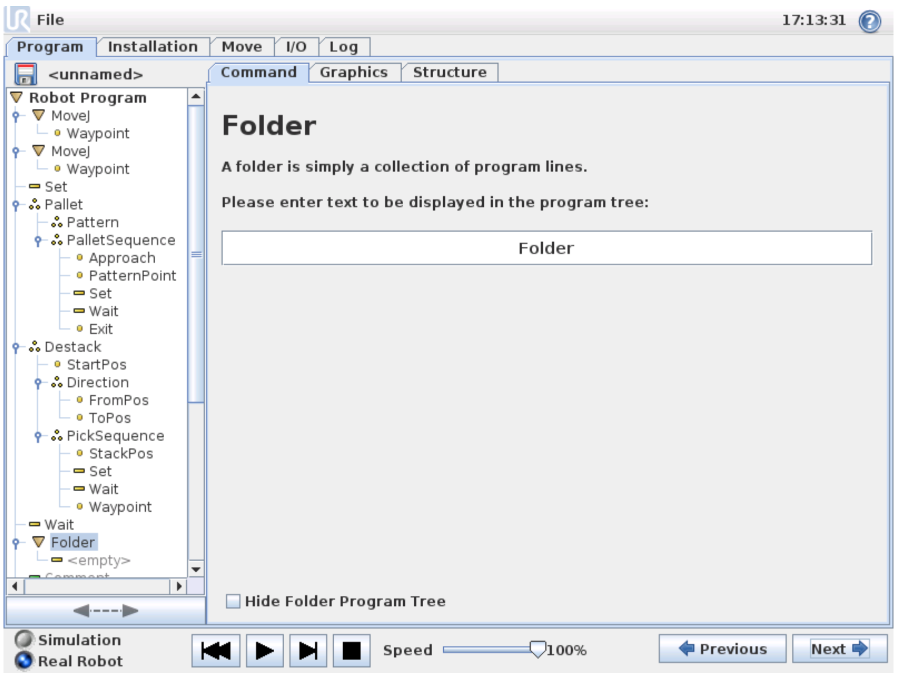


Comment

Please enter 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



Folder

A folder is simply a collection of program lines.

Please enter text to be displayed in the program tree:

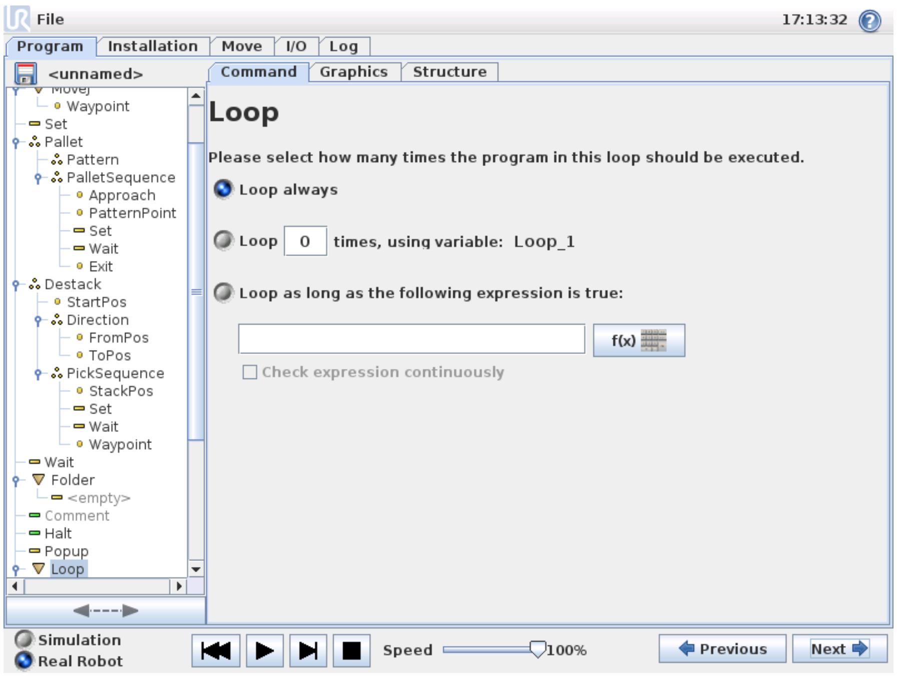
Folder

☐ Hide Folder Program Tree

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



Loop

Please select how many times the program in this loop should be executed.

● Loop always

○ Loop 0 times, using variable: Loop_1

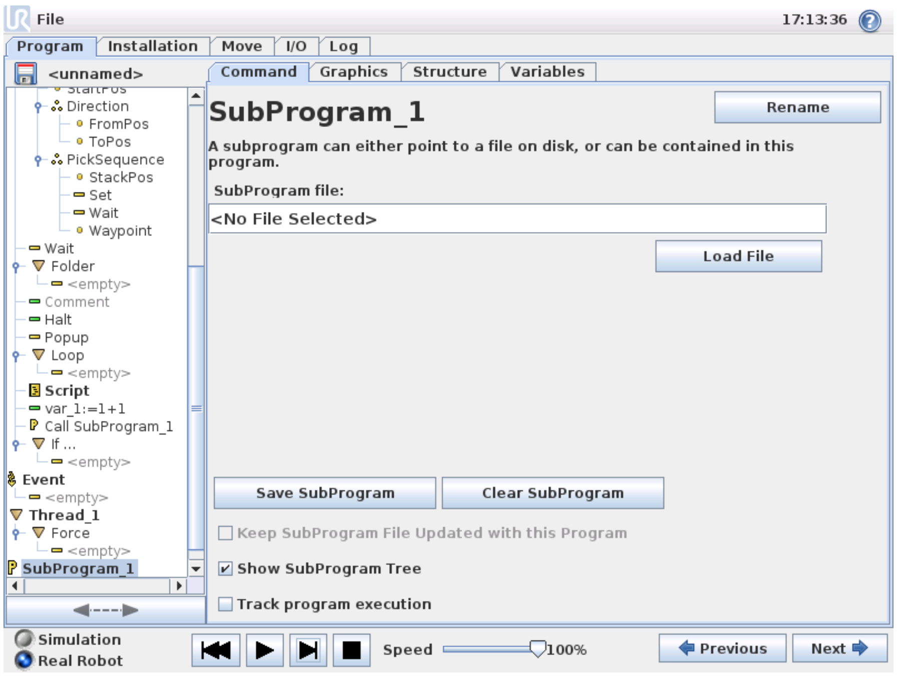
○ Loop as long as the following expression is true:

☐ Check expression continuously

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



SubProgram_1

A subprogram can either point to a file on disk, or can be contained in this program.

SubProgram file: <No File Selected>

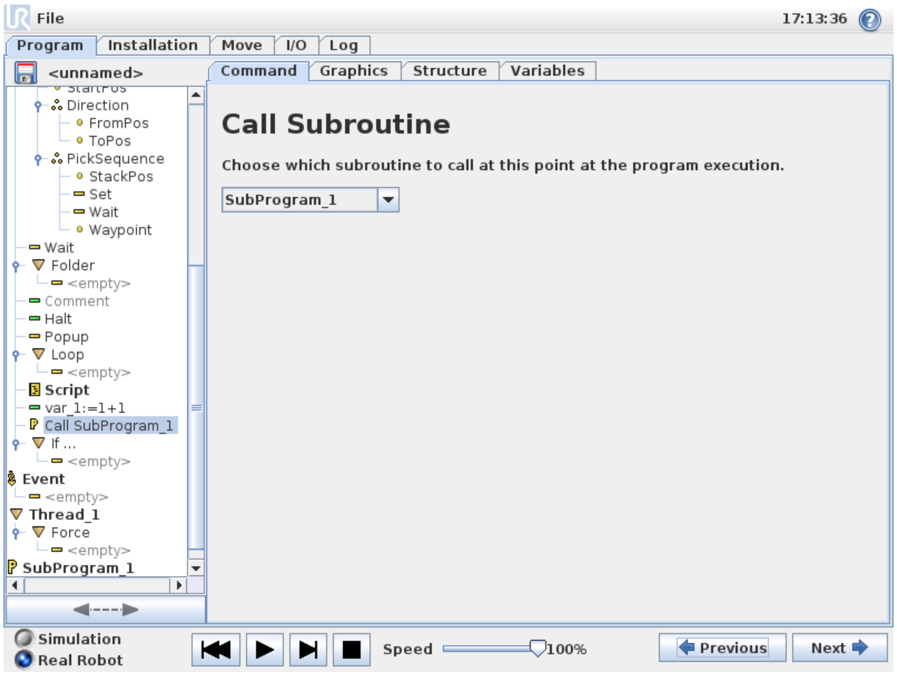
☐ Keep SubProgram File Updated with this Program

☑ Show SubProgram Tree

☐ Track program execution

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

Program → Command Tab, Call SubProgram



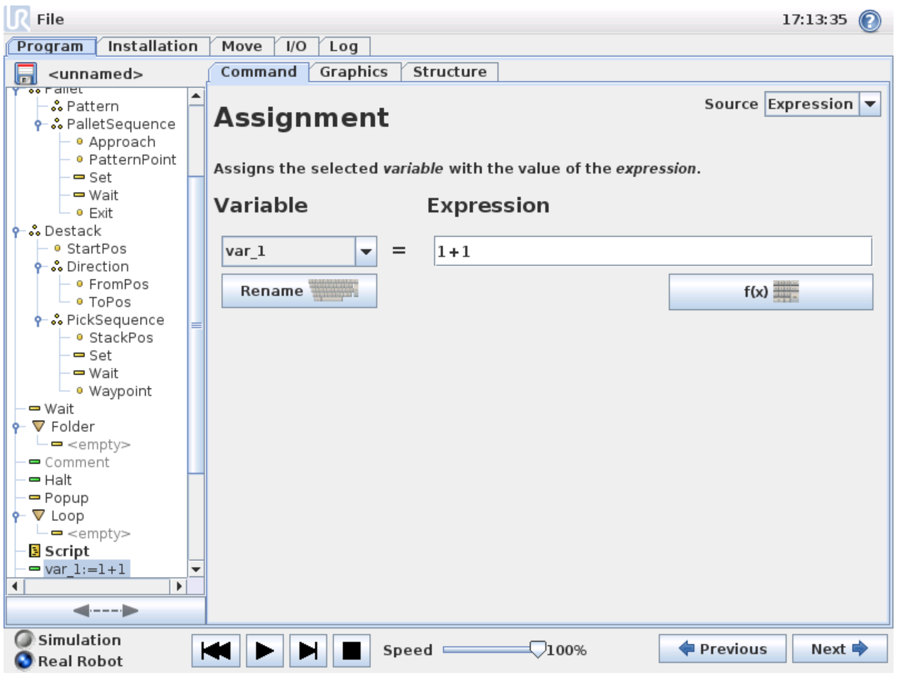
Call Subroutine

Choose which subroutine to call at this point at the program execution.

SubProgram_1

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



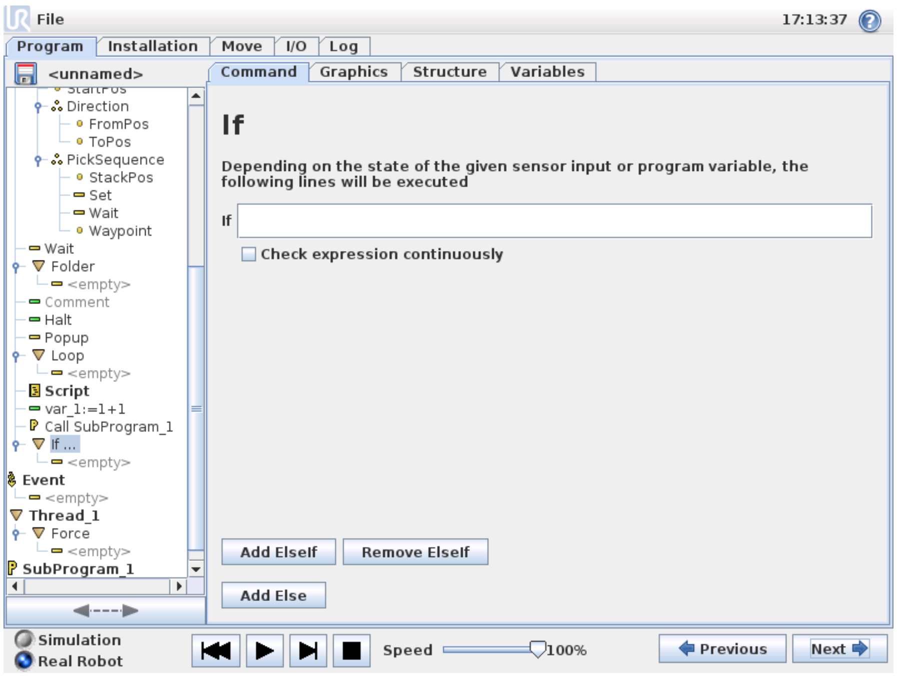
Assignment

Assigns the selected *variable* with the value of the *expression*.

Variable		Expression
var_1	=	1+1

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



If

Depending on the state of the given sensor input or program variable, the following lines will be executed

If

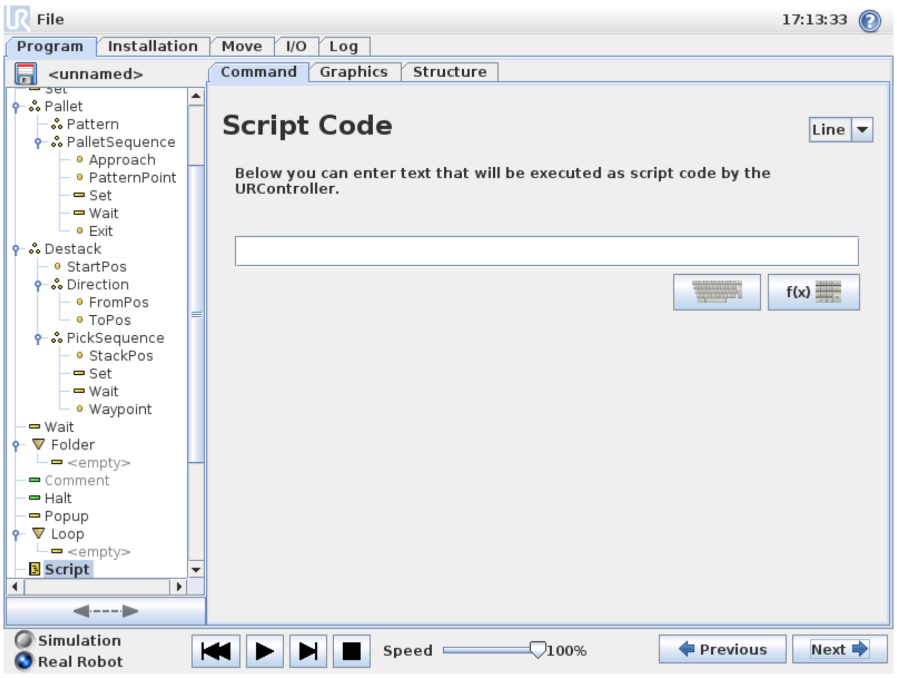
☐ Check expression continuously

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



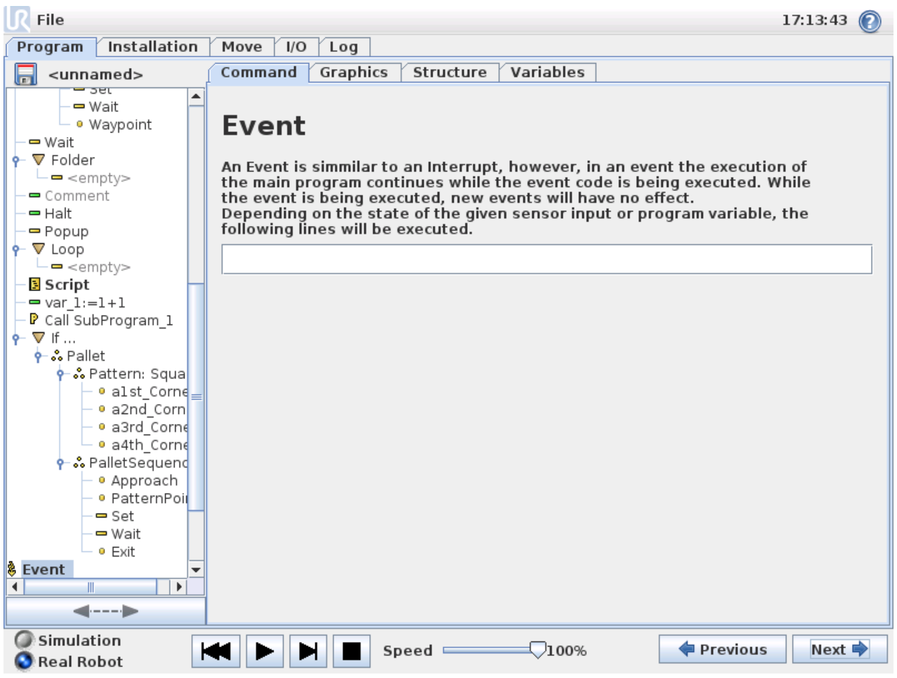
Script Code

Below you can enter text that will be executed as script code by the URController.

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 choosen, 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

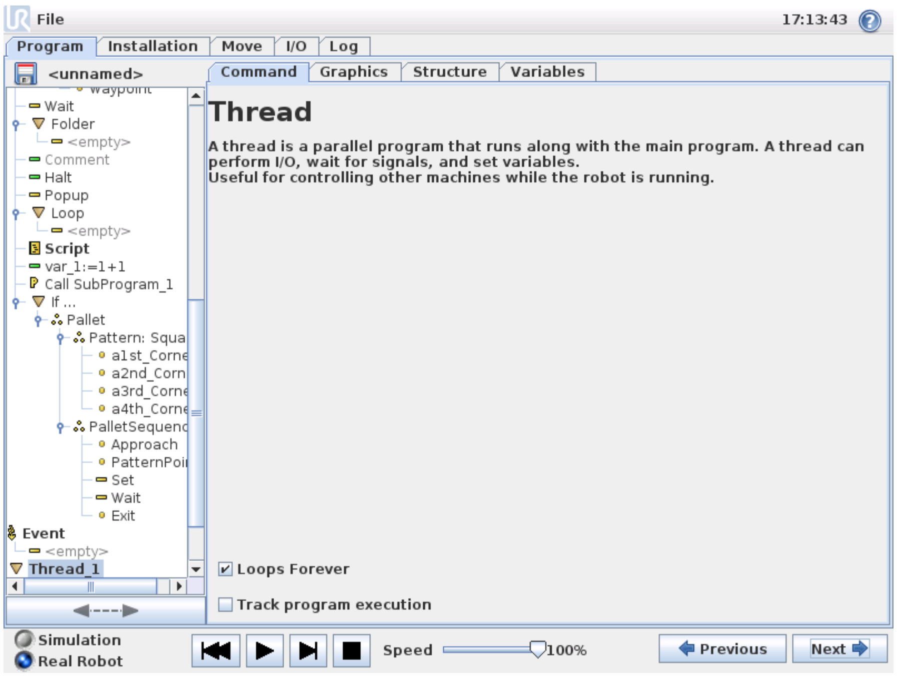


Event

An Event is simmilar to an Interrupt, however, in an event the execution of the main program continues while the event code is being executed. While the event is being executed, new events will have no effect.
Depending on the state of the given sensor input or program variable, the following lines will be executed.

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



Thread

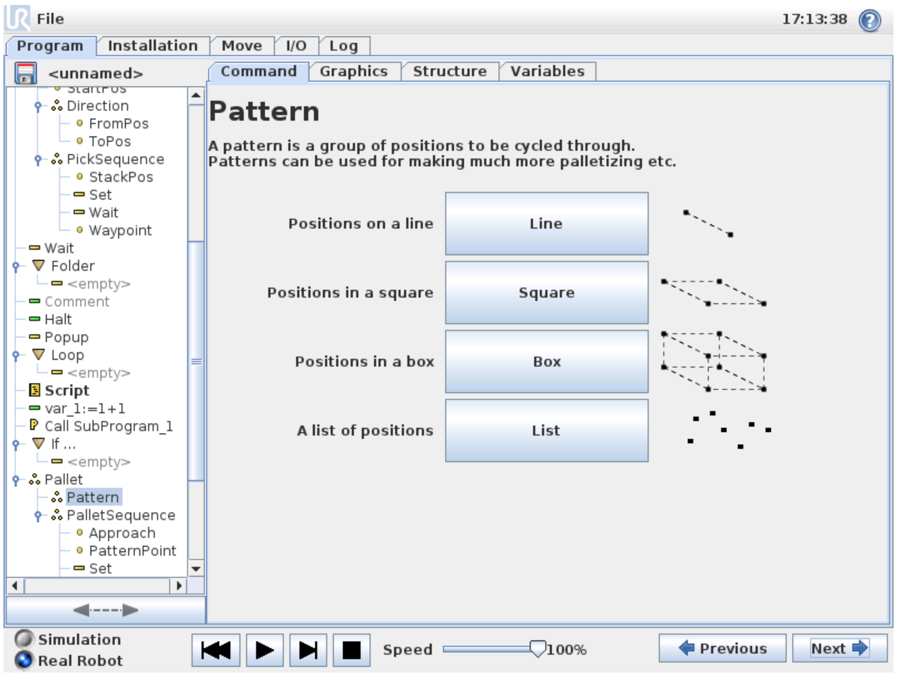
A thread is a parallel program that runs along with the main program. A thread can perform I/O, wait for signals, and set variables.
Useful for controlling other machines while the robot is running.

☑ Loops Forever

☐ Track program execution

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



Pattern

A pattern is a group of positions to be cycled through.
Patterns can be used for making much more palletizing etc.

Positions on a line	Line
Positions in a square	Square
Positions in a box	Box
A list of positions	List

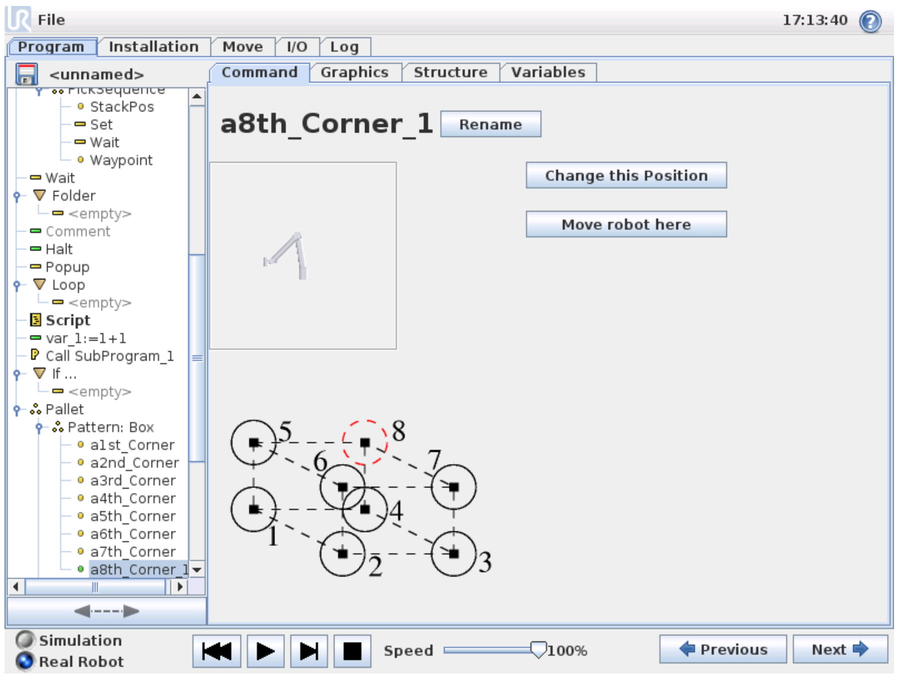
The Pattern command can be used to cycle through positions in the robots program. The pattern command corresponds to one position at each execution.

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.

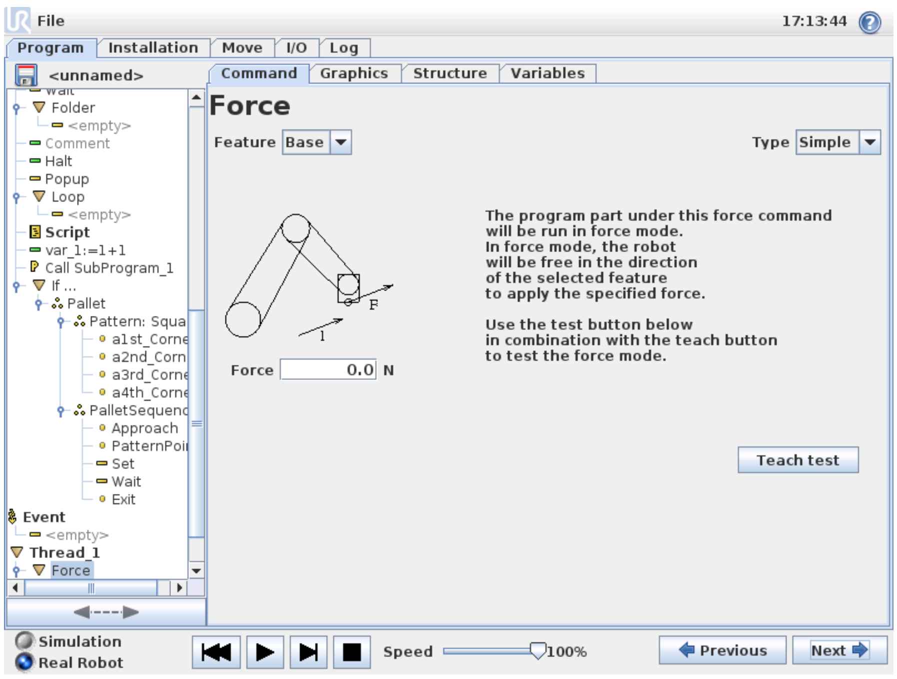
A counter variable is used while traversing the positions of the pattern. The name of the variable can be seen on the `Pattern` command screen. The variable cycles through the numbers from 0 to $X * Y * Z - 1$, the number of points in the pattern. This variable can be manipulated using assignments, and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



Force

Feature Base

Type Simple

Force 0.0 N

The program part under this force command will be run in force mode. In force mode, the robot will be free in the direction of the selected feature to apply the specified force.

Use the test button below in combination with the teach button to test the force mode.

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 3.11.

Force mode type

The 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 origo of the selected feature. The distance between the robot tcp and the origo 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 usefull 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 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

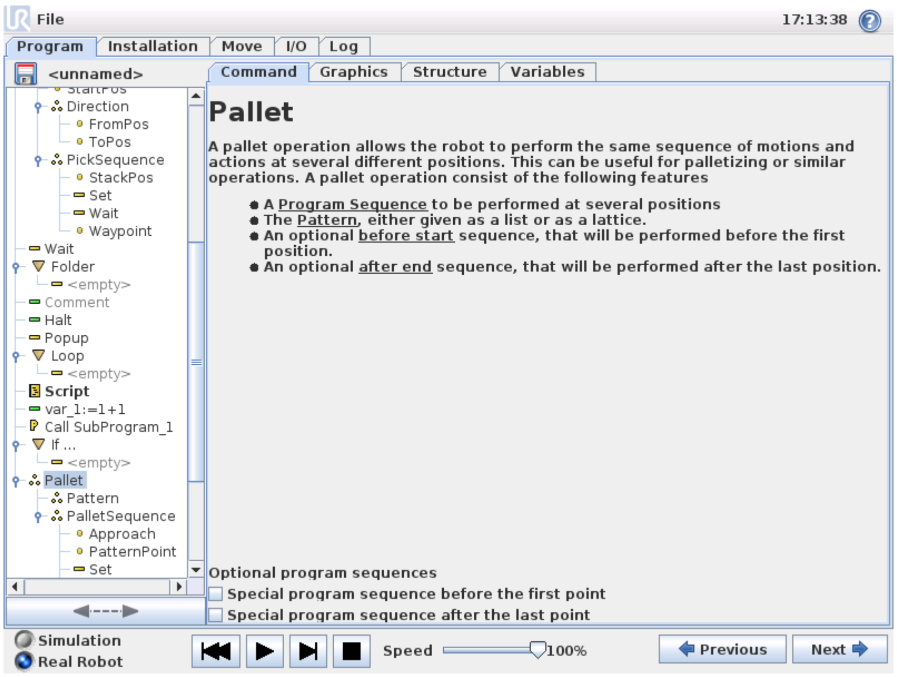
For all axes a limit can be set, but these have different meaning corresponding to the axes being complian or non-compliant.

- **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



Pallet

A pallet operation allows the robot to perform the same sequence of motions and actions at several different positions. This can be useful for palletizing or similar operations. A pallet operation consist of the following features

- A Program Sequence to be performed at several positions
- The Pattern, either given as a list or as a lattice.
- An optional before start sequence, that will be performed before the first position.
- An optional after end sequence, that will be performed after the last position.

Optional program sequences

☐ Special program sequence before the first point

☐ Special program sequence after the last point

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"

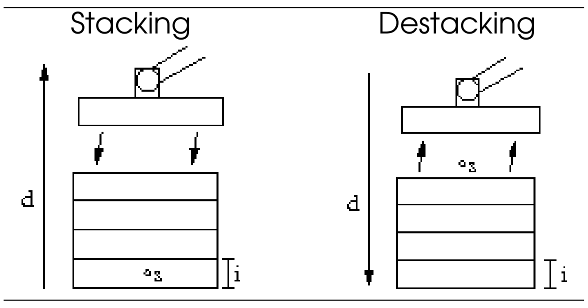
The optional `BeforeStart` sequence is run just before the operation starts. This can be used to wait for ready signals.

"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

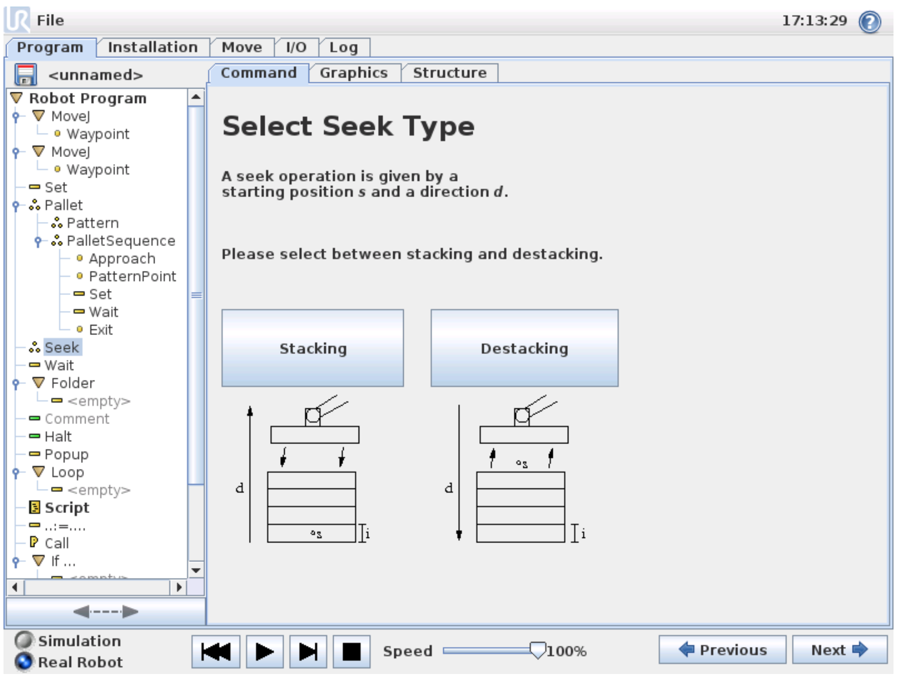
A seek function uses a sensor to determine when the correct position is reached to grab or drop an item. The sensor can be a push button switch, a pressure sensor or a capacitive sensor. This function is made for working on stacks of items with varying item thickness, or where the exact positions of 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

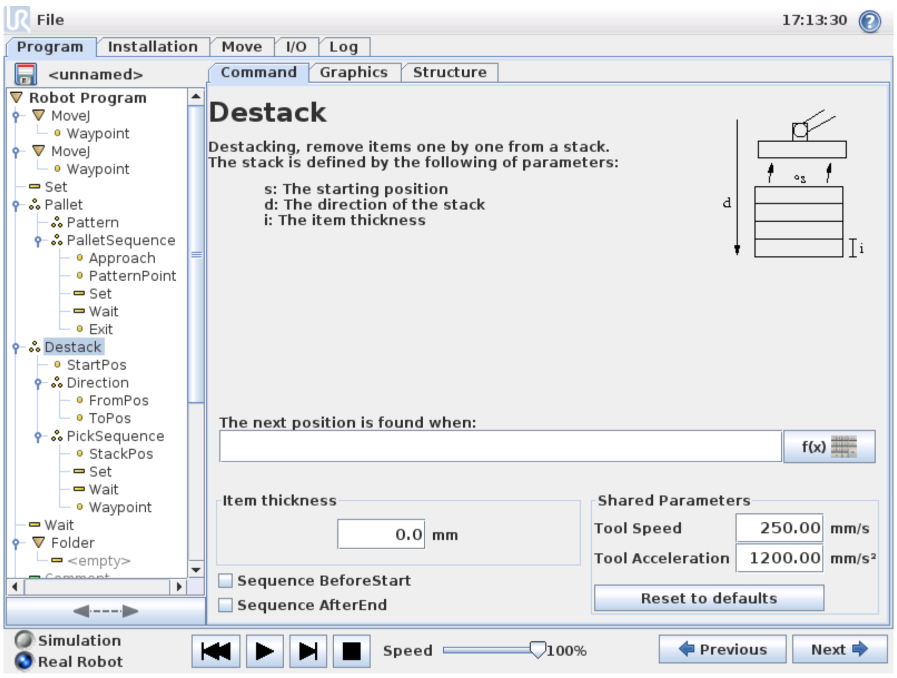

Select Seek Type

A seek operation is given by a starting position *s* and a direction *d*.

Please select between stacking and destacking.

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 hight is more than some defined number, or when a sensor gives a signal.

Destacking


Destack

Destacking, remove items one by one from a stack.
The stack is defined by the following of parameters:

s: The starting position
d: The direction of the stack
i: The item thickness

The next position is found when:

Item thickness: 0.0 mm

Shared Parameters

Tool Speed: 250.00 mm/s

Tool Acceleration: 1200.00 mm/s²

Sequence BeforeStart

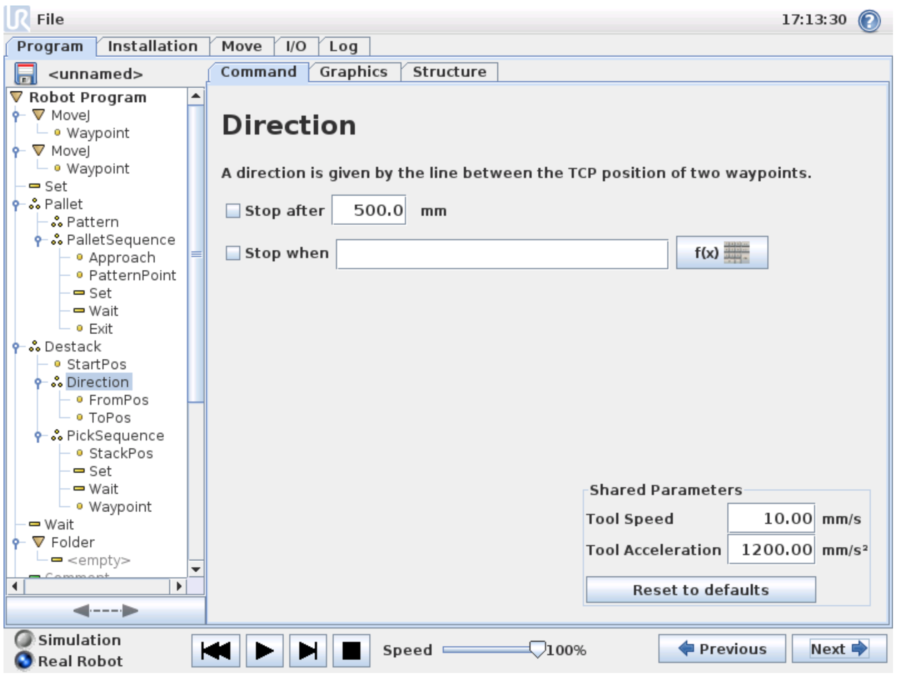
Sequence AfterEnd

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



Direction

A direction is given by the line between the TCP position of two waypoints.

Stop after 500.0 mm

Stop when

Shared Parameters

Tool Speed	10.00	mm/s
Tool Acceleration	1200.00	mm/s²

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.

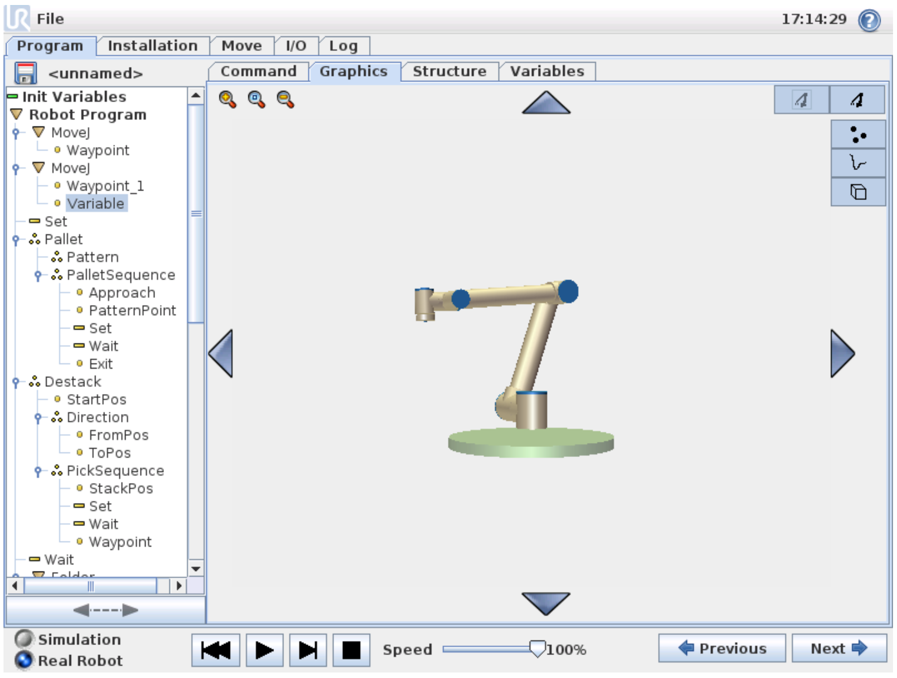
Pick/Place Sequence

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

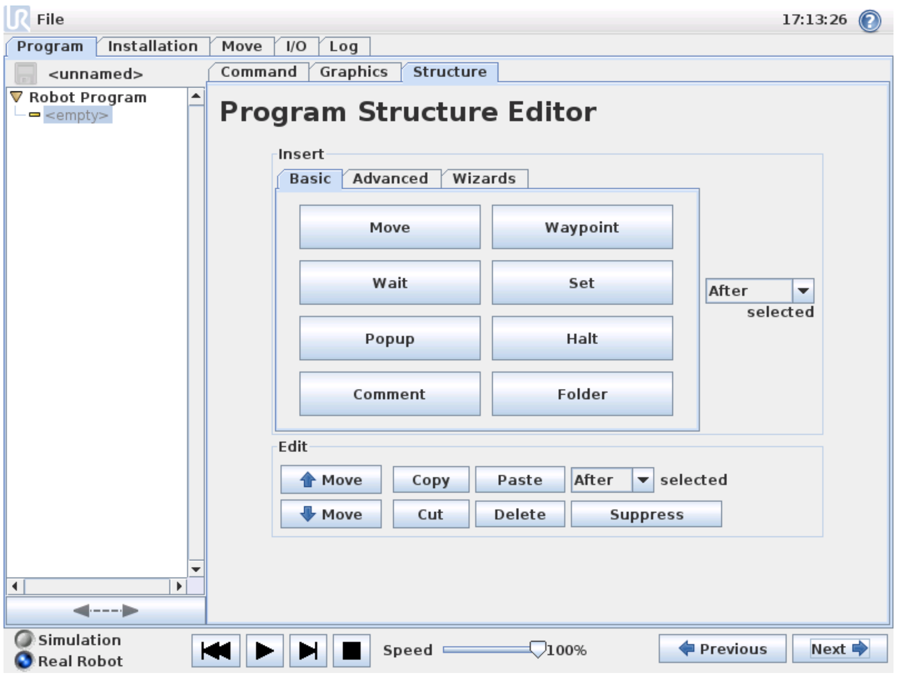


Graphical representation of the current robot program. The path of the TCP is shown in the 3D view, with motion segments in black, and blend segments (transitions between motion segments) shown in green. The green dots specify the positions of the TCP at each of the waypoints in the program. The 3D drawing 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.

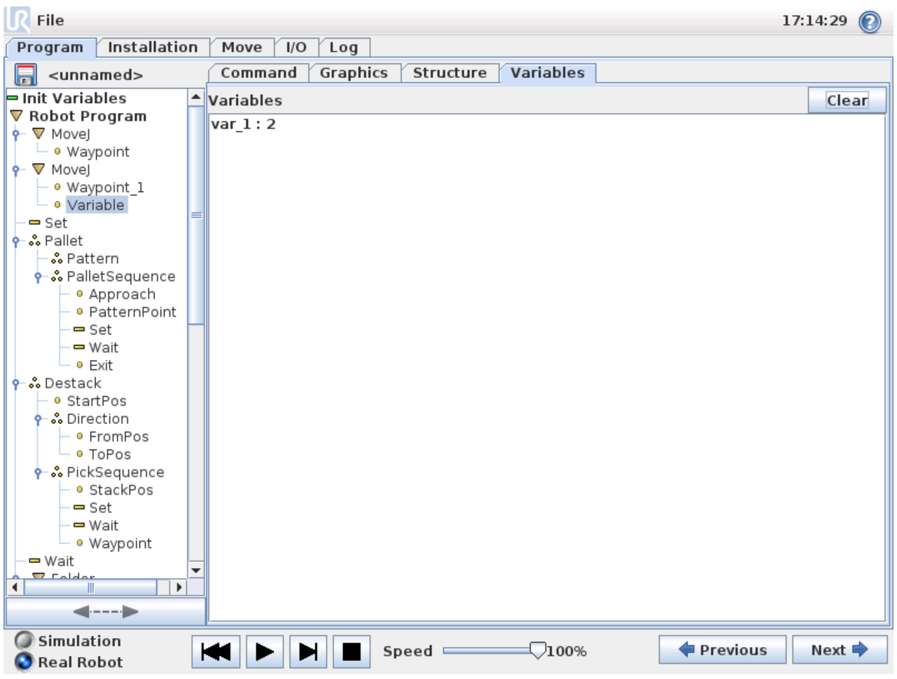
2) Select whether the new command should be inserted above or below the selected command.

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.

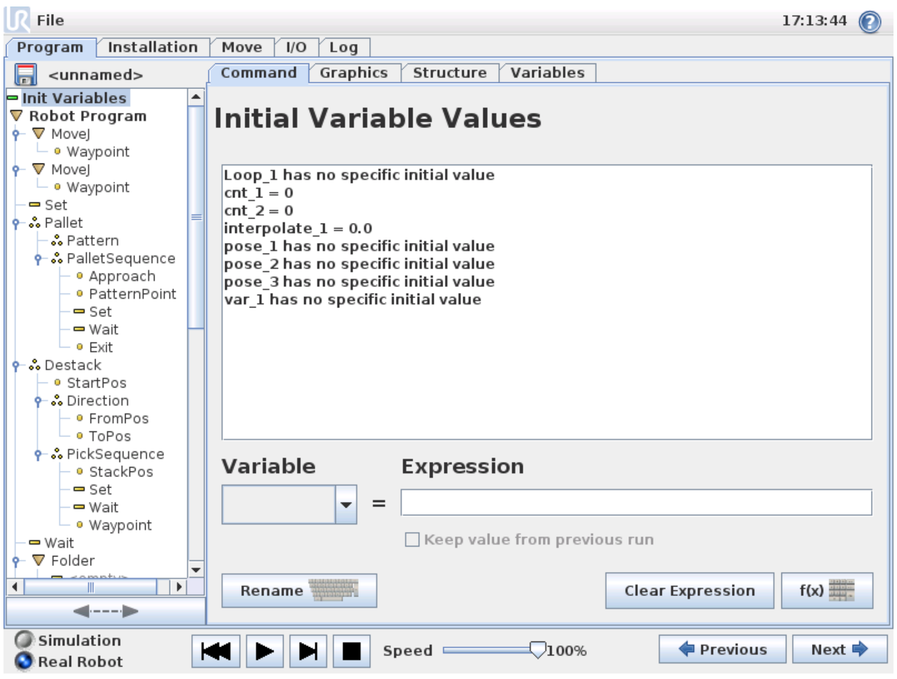
4.29 Program → Variables Tab



```
var_1 : 2
```

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



```
Loop_1 has no specific initial value
cnt_1 = 0
cnt_2 = 0
interpolate_1 = 0.0
pose_1 has no specific initial value
pose_2 has no specific initial value
pose_3 has no specific initial value
var_1 has no specific initial value
```

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

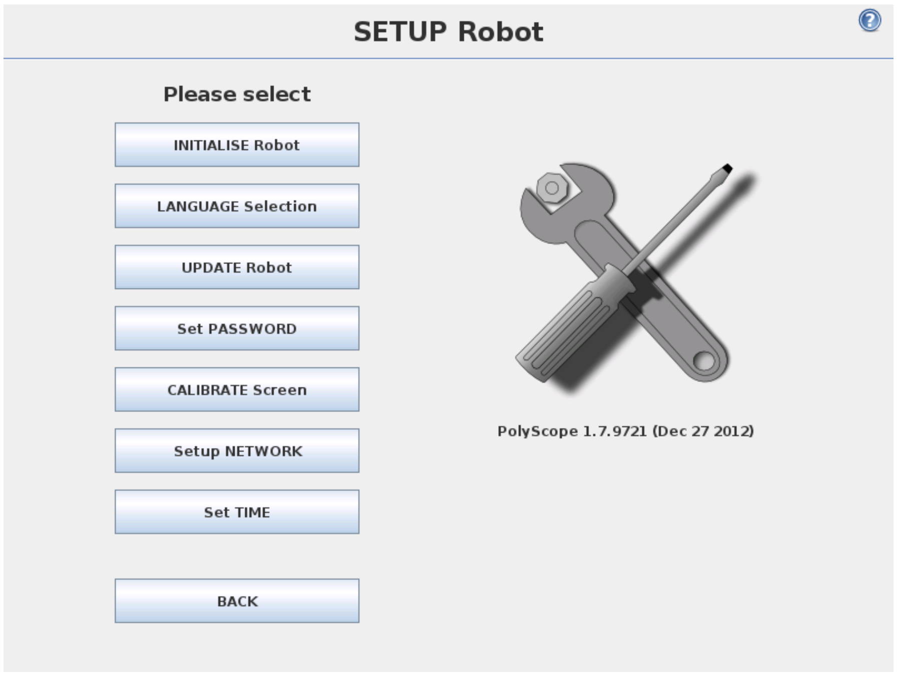
Select a variable from the list of variables by clicking on it, or by using the variable selector box. For a selected variable, an expression can be entered that will be used to set the variable value at program start.

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).

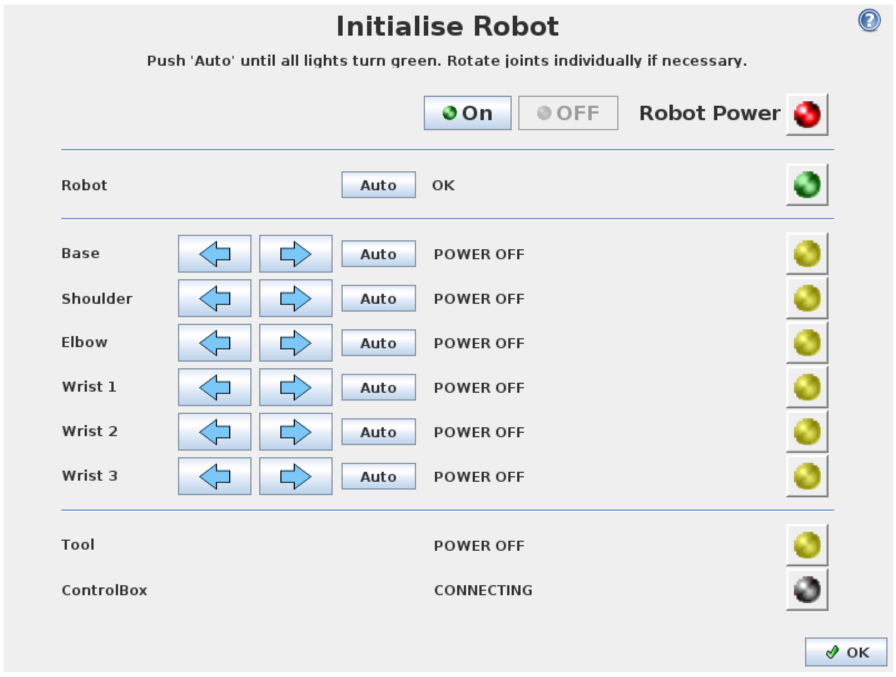
5 Setup

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



Initialise Robot

Push 'Auto' until all lights turn green. Rotate joints individually if necessary.

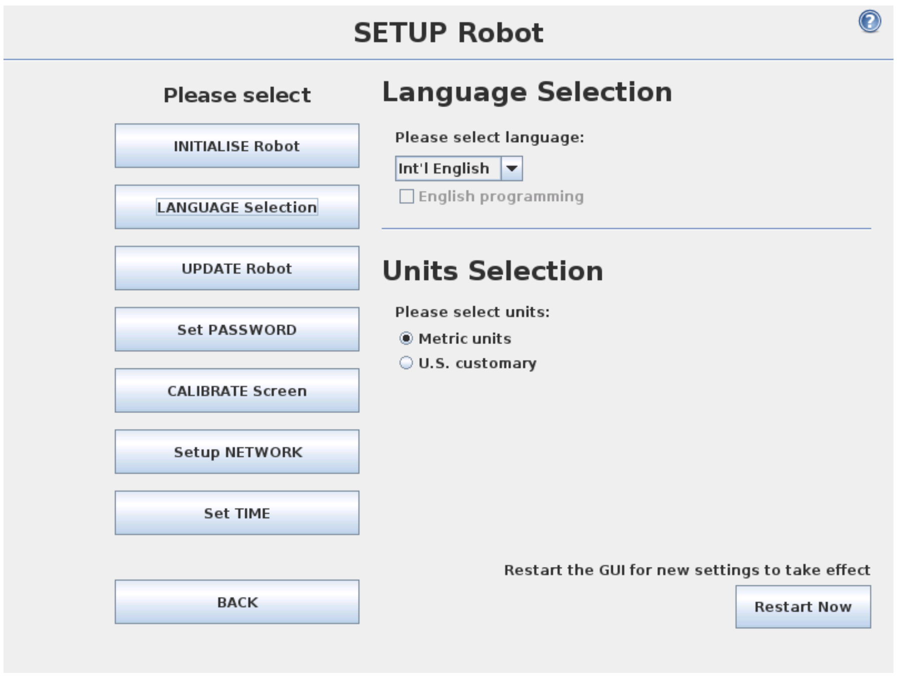
On | OFF | Robot Power

Robot			Auto	OK	●
Base	⇦	⇨	Auto	POWER OFF	●
Shoulder	⇦	⇨	Auto	POWER OFF	●
Elbow	⇦	⇨	Auto	POWER OFF	●
Wrist 1	⇦	⇨	Auto	POWER OFF	●
Wrist 2	⇦	⇨	Auto	POWER OFF	●
Wrist 3	⇦	⇨	Auto	POWER OFF	●
Tool				POWER OFF	●
ControlBox				CONNECTING	●

✓ OK

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 finds 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



SETUP Robot

Please select

- INITIALISE Robot
- LANGUAGE Selection
- UPDATE Robot
- Set PASSWORD
- CALIBRATE Screen
- Setup NETWORK
- Set TIME
- BACK

Language Selection

Please select language: Int'l English

☐ English programming

Units Selection

Please select units:

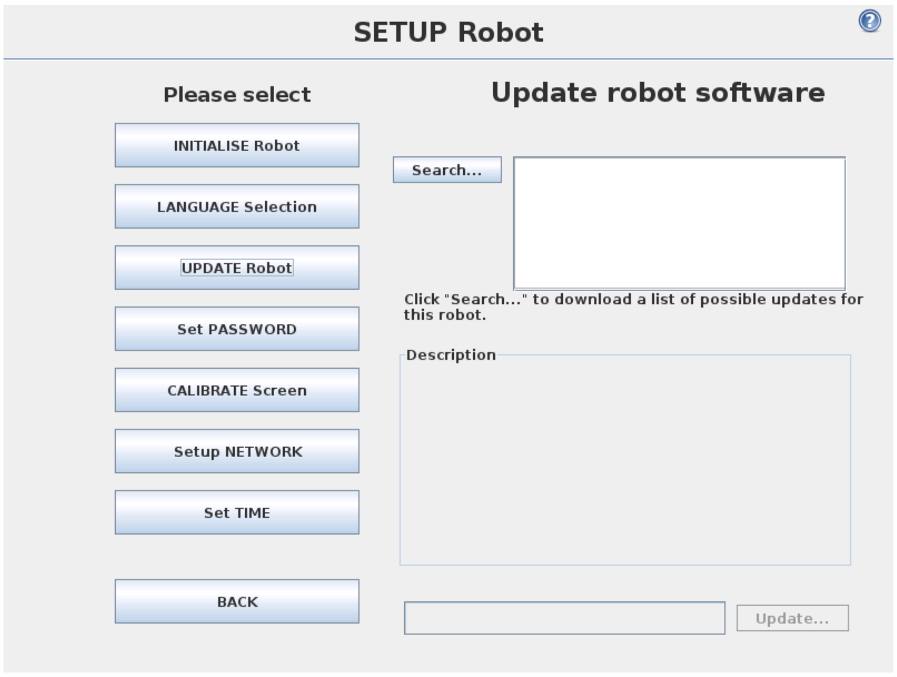
[✓ Metric units] U.S. customary

Restart the GUI for new settings to take effect

Restart Now

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.

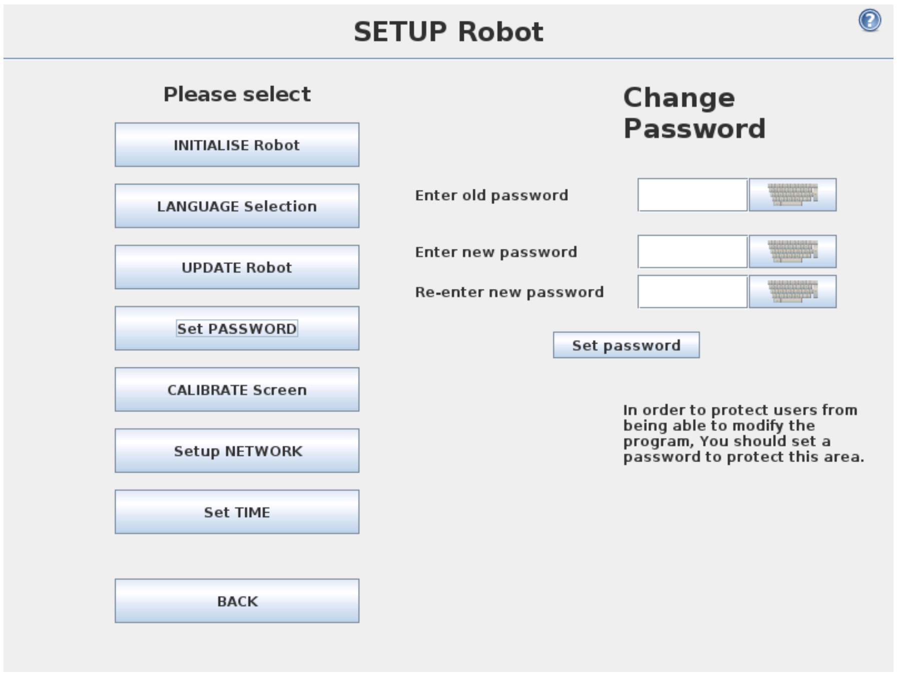
5.4 Setup Screen → Update



Click "Search..." to download a list of possible updates for this robot.

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.

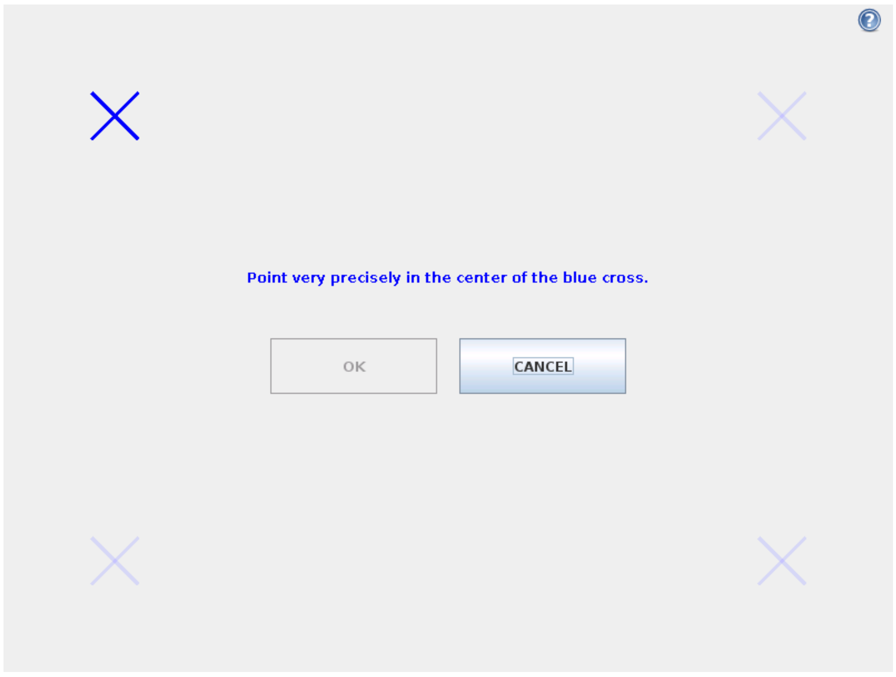
5.5 Setup Screen → Password



In order to protect users from being able to modify the program, You should set a password to protect this area.

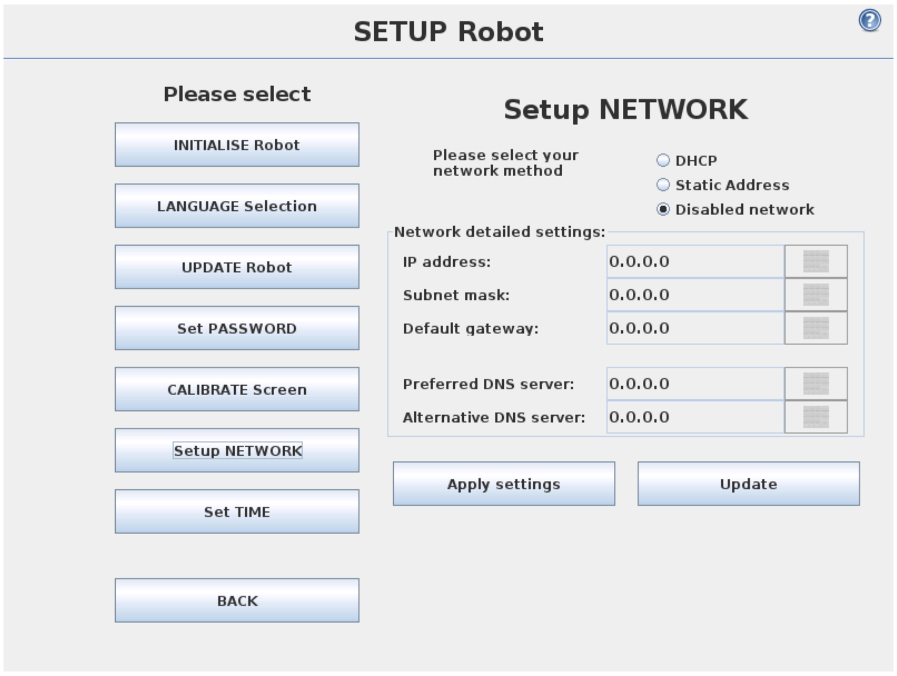
The programming part of the software can be locked using a password. When locked, programs can be loaded and run without the password, but a password is required to create or change programs.

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



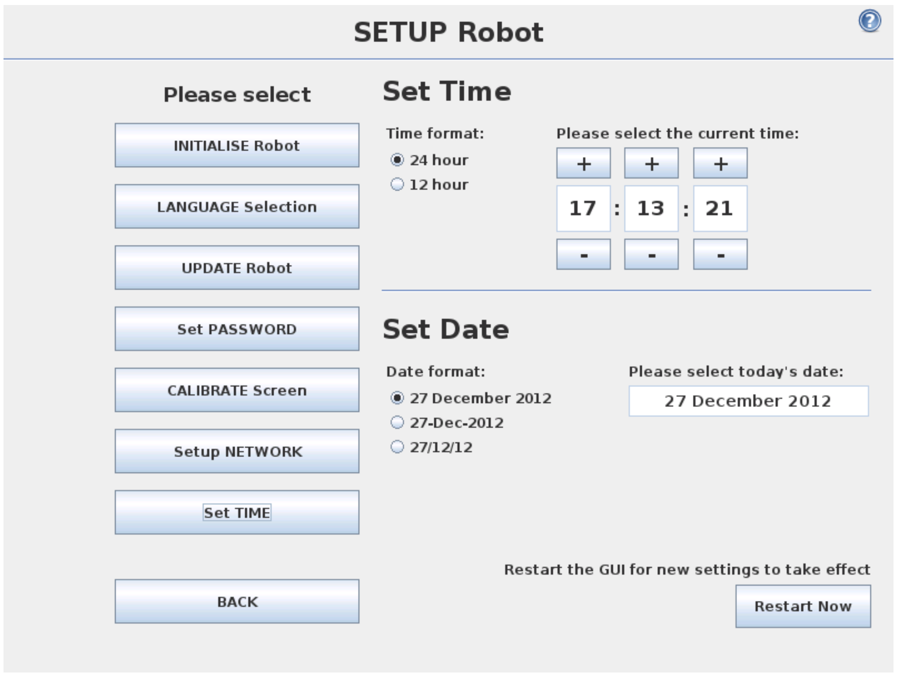
Please select your network method: ○ DHCP ○ Static Address [✓ Disabled network]

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

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



Set Time

Time format: [✓ 24 hour] 12 hour

Please select the current time: 17 : 13 : 21

Set Date

Date format: [✓ 27 December 2012] 27-Dec-2012 27/12/12

Please select today's date: 27 December 2012

Restart the GUI for new settings to take effect

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.