UNIVERSAL ROBOTS

PolyScope Manual

Version 3.13

Original instructions (en)

US version

Contents

II PolyScope Manual **II-1**

10 Safety Configuration **II-3**

10.1 Introduction . . . II-3
10.2 Changing the Safety Configuration . . . II-5
10.3 Safety Synchronization and Errors . . . II-5
10.4 Tolerances . . . II-6
10.5 Safety Checksum . . . II-7
10.6 Safety Modes . . . II-7
10.7 Freedrive Mode . . . II-7
10.7.1 Backdrive . . . II-8
10.8 Password Lock . . . II-8
10.9 Apply . . . II-8
10.10 General Limits . . . II-9
10.11 Joint Limits . . . II-11
10.12 Boundaries . . . II-12
10.12.1 Selecting a boundary to configure . . . II-13
10.12.2 3D visualization . . . II-14
10.12.3 Safety plane configuration . . . II-14
10.12.4 Tool Boundary configuration . . . II-18
10.13 Safety I/O . . . II-20
10.13.1 Input Signals . . . II-20
10.13.2 Output Signals . . . II-22

11 Begin programming **II-25**

11.1 Introduction . . . II-25
11.2 Getting Started . . . II-25
11.2.1 Installing the Robot Arm and Control Box . . . II-25
11.2.2 Turning the Control Box On and Off . . . II-26
11.2.3 Turning the Robot Arm On and Off . . . II-26
11.2.4 Quick Start . . . II-27
11.2.5 The First Program . . . II-27
11.3 PolyScope Programming Interface . . . II-28
11.4 Welcome Screen . . . II-30
11.5 Initialization Screen . . . II-31

12 On-screen Editors **II-33**
12.1 On-screen Expression Editor . . . II-33
12.2 Pose Editor Screen . . . II-33

13 Robot Control **II-37**
13.1 Move Tab . . . II-37
13.1.1 Robot . . . II-37
13.1.2 Feature and Tool Position . . . II-38
13.1.3 Move Tool . . . II-38
13.1.4 Move Joints . . . II-38
13.1.5 Freedrive . . . II-38
13.2 I/O Tab . . . II-39
13.3 MODBUS . . . II-40
13.4 AutoMove Tab . . . II-41
13.5 Installation → Load/Save . . . II-42
13.6 Installation → TCP Configuration . . . II-43
13.6.1 Adding, renaming, modifying and removing TCPs . . . II-44
13.6.2 Active TCP . . . II-44
13.6.3 Default TCP . . . II-44
13.6.4 Teaching TCP position . . . II-44
13.6.5 Teaching TCP orientation . . . II-45
13.6.6 Payload . . . II-45
13.6.7 Center of gravity . . . II-46
13.7 Installation → Mounting . . . II-46
13.8 Installation → I/O Setup . . . II-47
13.8.1 I/O Signal Type . . . II-48
13.8.2 Assigning User-defined Names . . . II-48
13.8.3 I/O Actions and I/O Tab Control . . . II-48
13.9 Installation → Safety . . . II-49
13.10 Installation → Variables . . . II-49
13.11 Installation → MODBUS client I/O Setup . . . II-50
13.12 Installation → Features . . . II-54
13.12.1 Using a feature . . . II-55
13.12.2 New Point . . . II-56
13.12.3 New Line . . . II-56
13.12.4 Plane Feature . . . II-58
13.12.5 Example: Manually Updating a Feature to Adjust a Program . . . II-58
13.12.6 Example: Dynamically Updating a Feature Pose . . . II-59
13.13 Conveyor Tracking Setup . . . II-60
13.14 Smooth Transition Between Safety Modes . . . II-61
13.14.1 Adjusting Acceleration/Deceleration Settings . . . II-61
13.15 Installation → Default Program . . . II-62
13.16 Log Tab . . . II-63

13.16.1 Saving Error Reports II-64
13.16.2 Technical Support File II-64
13.17 Load Screen II-65
13.18 Run Tab II-67

14 Programming **II-69**
14.1 New Program II-69
14.2 Program Tab II-70
14.2.1 Program Tree II-70
14.2.2 Program Execution Indication II-71
14.2.3 Search Button II-71
14.2.4 Undo/Redo Buttons II-72
14.2.5 Program Dashboard II-72
14.3 Variables II-73
14.4 Command: Move II-73
14.5 Command: Fixed Waypoint II-77
14.6 Command: Relative Waypoint II-83
14.7 Command: Variable Waypoint II-84
14.8 Command: Direction II-84
14.9 Command: Until II-85
14.10 Command: Wait II-87
14.11 Command: Set II-88
14.12 Command: Popup II-89
14.13 Command: Halt II-90
14.14 Command: Comment II-91
14.15 Command: Folder II-92
14.16 Command: Loop II-93
14.17 Command: If II-93
14.18 Command: SubProgram II-95
14.19 Command: Assignment II-97
14.20 Command: Script II-98
14.21 Command: Event II-99
14.22 Command: Thread II-100
14.23 Command: Switch II-101
14.23.1 Timer II-102
14.24 Command: Pattern II-103
14.25 Command: Force II-104
14.26 Command: Pallet II-108
14.27 Command: Seek II-109
14.28 Command: Conveyor Tracking II-113
14.29 Command: Suppress II-113
14.30 Graphics Tab II-113
14.31 Structure Tab II-115

14.32 Variables Tab II-116
14.33 Command: Variables Initialization II-117

15 Setup Screen **II-119**
15.1 Language and Units II-120
15.2 Update Robot II-121
15.3 Set Password II-122
15.4 Calibrate Screen II-123
15.5 Setup Network II-124
15.6 Set Time II-125
15.7 URCaps Setup II-126

Part II

PolyScope Manual

10 Safety Configuration

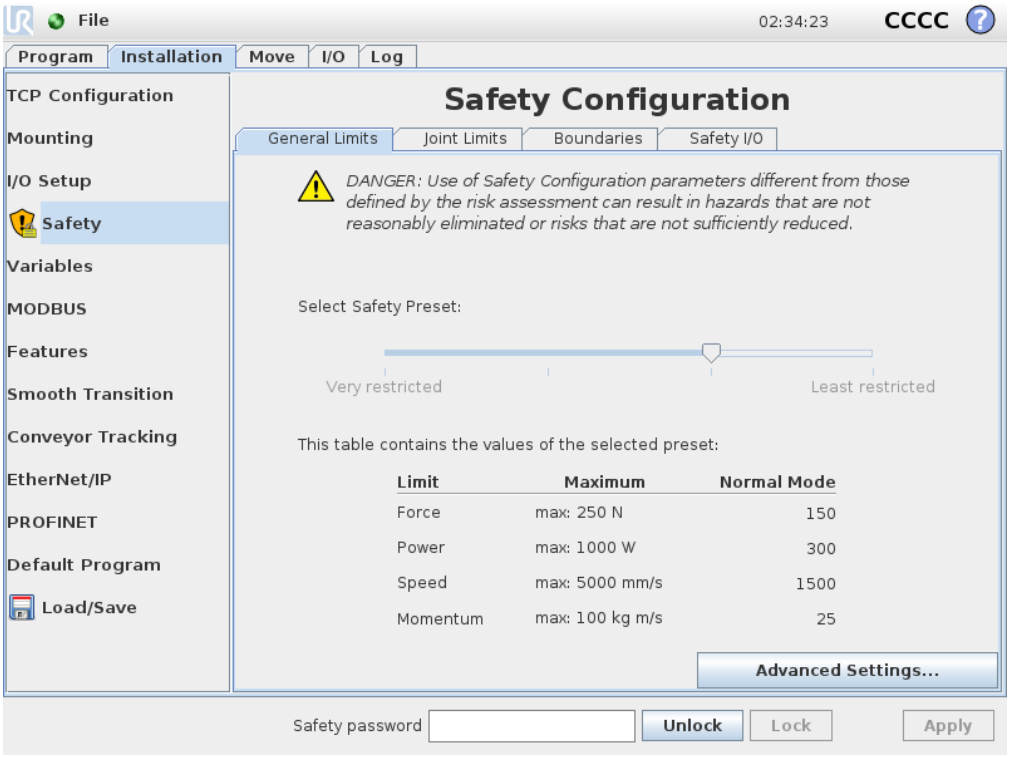
10.1 Introduction

The robot is equipped with an advanced safety system. Depending on the particular characteristics of the robot workspace, the settings for the safety system must be configured to guarantee the safety of all personnel and equipment around the robot. Applying settings defined by the risk assessment is the first thing the integrator must do. For details on the safety system, see the Hardware Installation Manual.

DANGER:

1. Use and configuration of safety-related functions and interfaces must be done according to the risk assessment that the integrator conducts for a specific robot application, see the Hardware Installation Manual.
2. Safety configuration settings for set-up and teaching must be applied according to the risk assessment conducted by the integrator and before the robot arm is powered on for the first time.
3. All safety configuration settings accessible on this screen and its subtabs are required to be set according to the risk assessment conducted by the integrator.
4. The integrator is required to ensure that all changes to the safety configuration settings are done in compliance with the integrator's own risk assessment.
5. The integrator must prevent unauthorized persons from changing the safety configuration, e.g. by use of password protection.

The `Safety Configuration` screen can be accessed from the `Welcome` screen (see 11.4) by pressing the `Program Robot` button, selecting the `Installation` tab and tapping `Safety`. The safety configuration is password protected, see 10.8.



The safety settings consist of a number of limit values used to constrain the movements of the robot arm, and of safety function settings for the configurable inputs and outputs. They are defined in the following subtabs of the safety screen:

- The `General Limits` subtab defines the maximum *force*, *power*, *speed* and *momentum* of the robot arm. When the risk of hitting a human or colliding with a part of its environment is particularly high, these settings need to be set to low values. If the risk is low, higher general limits enable the robot to move faster and exert more force on its environment. For further details, see 10.10.
- The `Joint Limits` subtab consists of *joint speed* and *joint position* limits. The *joint speed* limits define the maximum angular velocity of individual joints and serve to further limit the speed of the robot arm. The *joint position* limits define the allowed position range of individual joints (in joint space). For further details, see 10.11.
- The `Boundaries` subtab defines safety planes (in Cartesian space) and a tool orientation boundary for the robot TCP. The safety planes can be configured either as hard limits for the position of the robot TCP, or triggers for activating the *Reduced* mode safety limits (see 10.6)). The tool orientation boundary puts a hard limit on the orientation of the robot TCP. For further details, see 10.12.
- The `Safety I/O` subtab defines safety functions for configurable inputs and outputs (see 13.2). For example, *Emergency Stop* can be configured as an input. For further details, see 10.13.

10.2 Changing the Safety Configuration

The safety configuration settings shall only be changed in compliance with the risk assessment conducted by the integrator.

The recommended procedure for changing the safety configuration is as follows:

1. Make sure that the changes are in compliance with the risk assessment conducted by the integrator.
2. Adjust safety settings to the appropriate level defined by the risk assessment conducted by the integrator.
3. Verify that the safety settings are applied.
4. Put the following text in the operators' manuals: "Before working near the robot, make sure that the safety configuration is as expected. This can be verified e.g. by inspecting the checksum in the top right corner of the PolyScope (see 10.5 in the PolyScope Manual)."

10.3 Safety Synchronization and Errors

The state of the applied safety configuration in comparison to what robot installation the GUI has loaded, is depicted by the shield icon next to the text `Safety` on the left side of the screen. These icons provide a quick indicator to the current state. They are defined below:


Configuration Synchronized: Shows the GUI installation is identical to the currently applied safety configuration. No changes have been made.


Configuration Altered: Shows the GUI installation is different from the currently applied safety configuration.

When editing the safety configuration, the shield icon will inform you whether or not the current settings have been applied.

If any of the text fields in the `Safety` tab contain any invalid input, the safety configuration is in an error state. This is indicated in several ways:

1. A red error icon is displayed next to the text `Safety` on the left side of the screen.
2. The subtab(s) with errors are marked with a red error icon at the top.
3. Text fields containing errors are marked with a red background.

When errors exist and attempting to navigate away from the `Installation` tab, a dialog appears with the following options:

1. Resolve the issue(s) so that all errors have been removed. This will be visible when the red error icon is no longer displayed next to the text `Safety` on the left side of the screen.
2. Revert back to the previously applied safety configuration. This will disregard all changes and allow you to continue to the desired destination.

If no errors exist and attempting to navigate away, a different dialog appears with the following options:

1. Apply changes and restart the system. This will apply the safety configuration modifications to the system and restart. Note: This does not imply that any changes have been saved; shut-down of the robot at this point will lose all changes to the robot installation including the Safety configuration.
2. Revert back to the previously applied safety configuration. This will disregard all changes and allow you to continue to the desired selected destination.

10.4 Tolerances

The *Robot Arm* uses built-in tolerances that prevent safety violations. A safety tolerance is the difference between a safety limit and a maximum operational value. For example, the general speed tolerance is $-150mm/s$. This means that if the user configures a $250mm/s$ speed limit, then the maximum operational speed will be $250 - 150 = 100mm/s$. Safety tolerances prevent safety violations while allowing for fluctuations in program behavior. For example, when handling a heavy payload, there may be situations where the *Robot Arm* needs to briefly operate above the normal maximum operational speed to follow a programmed trajectory. An example of such a situation is shown in figure 10.1.

WARNING:
A risk assessment is always required using the limit values without tolerances.

WARNING:
Tolerances are specific to the version of the software. Updating the software may change the tolerances. Consult the release notes for changes between versions.

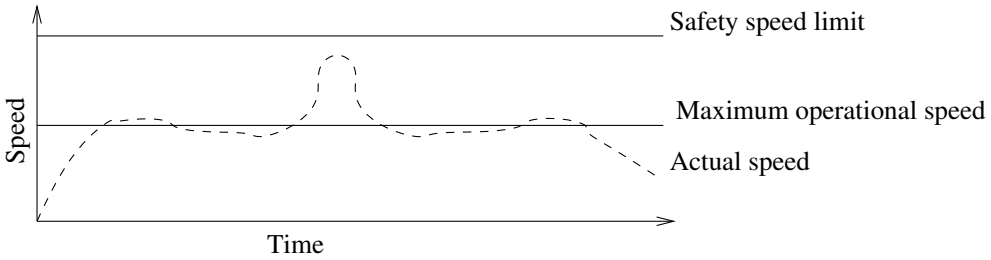


Figure 10.1: Safety tolerance example.

10.5 Safety Checksum

The text in the top right corner of the screen gives a shorthand representation of the safety configuration currently used by the robot. When the text changes, this indicates that the current safety configuration has changed as well. Clicking on the checksum displays the details about the currently active safety configuration.

10.6 Safety Modes

Under normal conditions (i.e. when no protective stop is in effect), the safety system operates in one of the following *safety modes*, each with an associated set of safety limits:

Normal mode: The safety mode that is active by default;

Reduced mode: Active when the robot TCP is positioned beyond a *Trigger Reduced mode* plane (see 10.12), or when triggered using a configurable input (see 10.13).

Recovery mode: When the robot arm is in violation of one of the other modes (i.e. *Normal* or *Reduced* mode) and a Stop Category 0 has occurred,[1] the robot arm will start up in **Recovery** mode. This mode allows the robot to move slowly back to the allowed area using **MoveTab** or **Freedrive**. It is not possible to run programs for the robot in this mode.

WARNING:
Note that limits for *joint position*, *TCP position* and *TCP orientation* are disabled in *Recovery* mode, so take caution when moving the robot arm back within the limits.

The subtabs of the `Safety Configuration` screen enable the user to define separate sets of safety limits for *Normal* and *Reduced* mode. For the tool and joints, *Reduced* mode limits regarding speed and momentum are required to be more restrictive than their *Normal* mode counterparts.

When a safety limit from the active limit set is violated, the robot arm performs a Stop Category 0. If an active safety limit, such as a joint position limit or a safety boundary, is violated already when the robot arm is powered on, it starts up in *Recovery* mode. This makes it possible to move the robot arm back within the safety limits. While in *Recovery* mode, the movement of the robot arm is limited by a fixed limit set that is not customizable by the user. For details about *Recovery* mode limits, see the Hardware Installation Manual.

10.7 Freedrive Mode

When in *Freedrive* mode (see 13.1.5) and the movement of the robot arm comes close to certain limits, the user will feel a repelling force. This force is generated for limits on the position, orientation and speed of the robot TCP and the position and speed of the joints.

The purpose of this repelling force is to inform the user that the current position or speed is close to a limit and to prevent the robot from violating that limit. However, if enough force is applied

[1] According to IEC 60204-1, see Glossary for more details.

by the user to the robot arm, the limit can be violated. The magnitude of the force increases as the robot arm comes closer to the limit.

10.7.1 Backdrive

In *Freedrive* mode, the robot joints can be moved with relatively little force because the brakes are released. During initialization of the robot arm, minor vibrations may be observed when the robot brakes are released. In some situations, such as when the robot is close to collision, these tremors are undesirable and the *Backdrive* feature can be used to forcefully move specific joints to a desired position without releasing all brakes in the robot arm.

To enable *Backdrive*:

1. Press ON to enable power for the joints. The robot state is set to "Idle". Do **not** release the brakes (i.e. do not press START).
2. Press and hold the *Freedrive* button. The robot state changes to "Backdrive".
3. Brakes will only be released in the joints to which significant pressure is applied, as long as the *Freedrive* button is engaged/pressed. While using *Backdrive*, the robot feels heavy to move around.

10.8 Password Lock

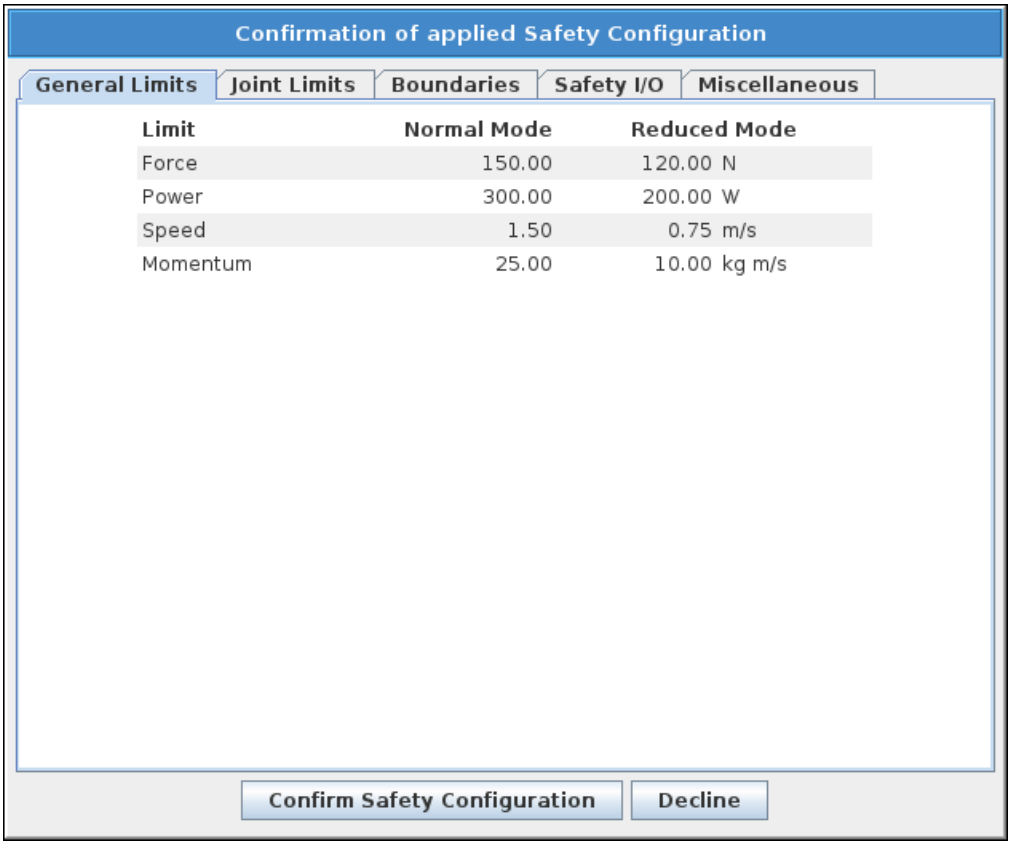
All settings on this screen are locked until the correct safety password (see 15.3) is entered in the white text field at the bottom of the screen and the `Unlock` button is pressed. The screen can be locked again by clicking the `Lock` button. The `Safety` tab is automatically locked when navigating away from the safety configuration screen. When the settings are locked, a lock icon is visible next to the text `Safety` on the left side of the screen. An unlock icon is shown when the settings are unlocked.

NOTE:
Note that the robot arm is powered off when the safety Configuration screen is unlocked.

10.9 Apply

When unlocking the safety configuration, the robot arm will be powered off while changes are being made. The robot arm cannot be powered on until the changes have been applied or reverted, and a manual power on is performed from the initialization screen.

Any changes to the safety configuration must be applied or reverted, before navigating away from the Installation tab. These changes are *not* in effect until after the `Apply` button is pressed and confirmation is performed. Confirmation requires visual inspection of the changes given to the robot arm. For safety reasons, the information shown is given in SI Units. An example of the confirmation dialog is shown below.



Furthermore, on confirmation the changes are automatically saved as part of the current robot installation. See 13.5 for further information on saving the robot installation.

10.10 General Limits

The general safety limits serve to limit the linear speed of the robot TCP as well as the force it may exert on the environment. They are composed of the following values:

Force: A limit for the maximum force that the robot TCP exerts on the environment.

Power: A limit for the maximum mechanical work produced by the robot on the environment, considering that the payload is part of the robot and not of the environment.

Speed: A limit for the maximum linear speed of the robot TCP.

Momentum: A limit for the maximum momentum of the robot arm.

There are two means available for configuring the general safety limits within the installation; *Basic Settings* and *Advanced Settings* which are described more fully below.

Defining the general safety limits only defines the limits for the tool, and not the overall limits of the robot arm. This means that although a speed limit is specified, it does *not* guarantee that other parts of the robot arm will obey this same limitation.

When in *Freedrive* mode (see 13.1.5), and the current speed of the robot TCP is close to the *Speed* limit, the user will feel a repelling force which increases in magnitude the closer the speed comes

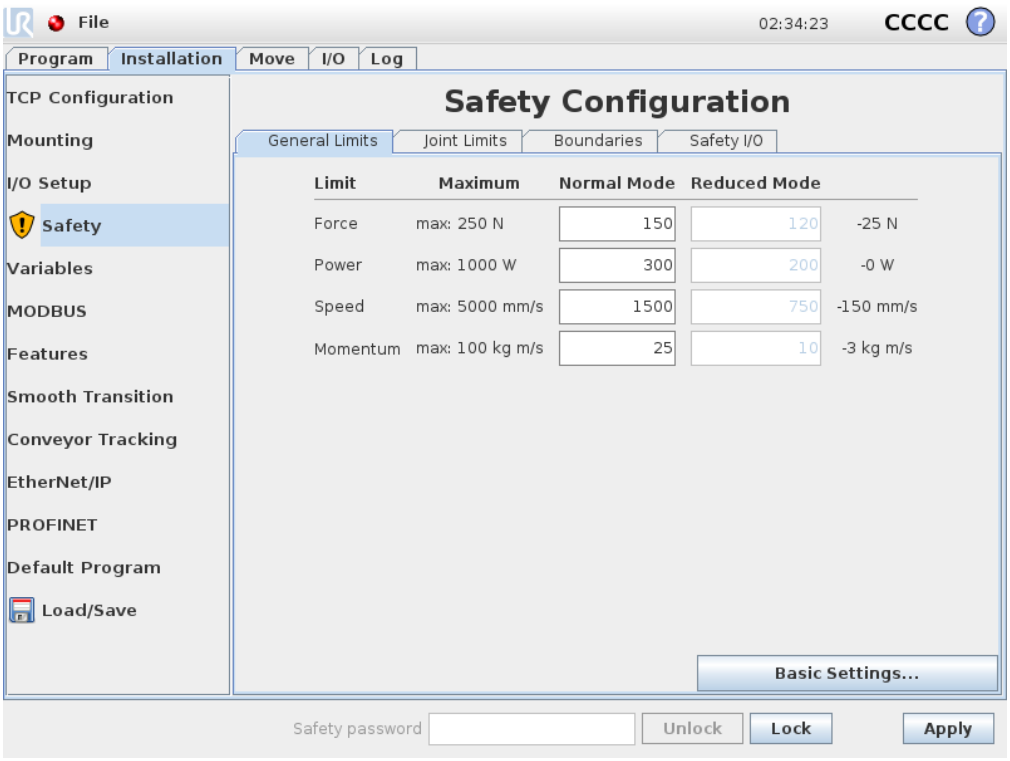
to the limit. The force is generated when the current speed is within approximately 250 mm/s of the limit.

Basic Settings The initial general limits subpanel, shown as the default screen, features a slider with four predefined sets of values for force, power, speed, and momentum limits in both *Normal* and *Reduced* mode.

The specific sets of values are shown in the GUI. Predefined sets of values are suggestions only and shall not substitute a proper risk assessment.

Switching to Advanced Settings Should *none* of the predefined sets of values be satisfactory, the `Advanced Settings...` button can be pressed to enter the advanced general limits screen.

Advanced Settings



Here, each of the general limits, described in 10.10, can be modified independently of the others. This is done by tapping the corresponding text field and entering the new value. The highest accepted value for each of the limits is listed in the column titled `Maximum`. The force limit can be set to a value between 100 N and 250 N, and the power limit can be set to a value between 80 W and 1000 W.

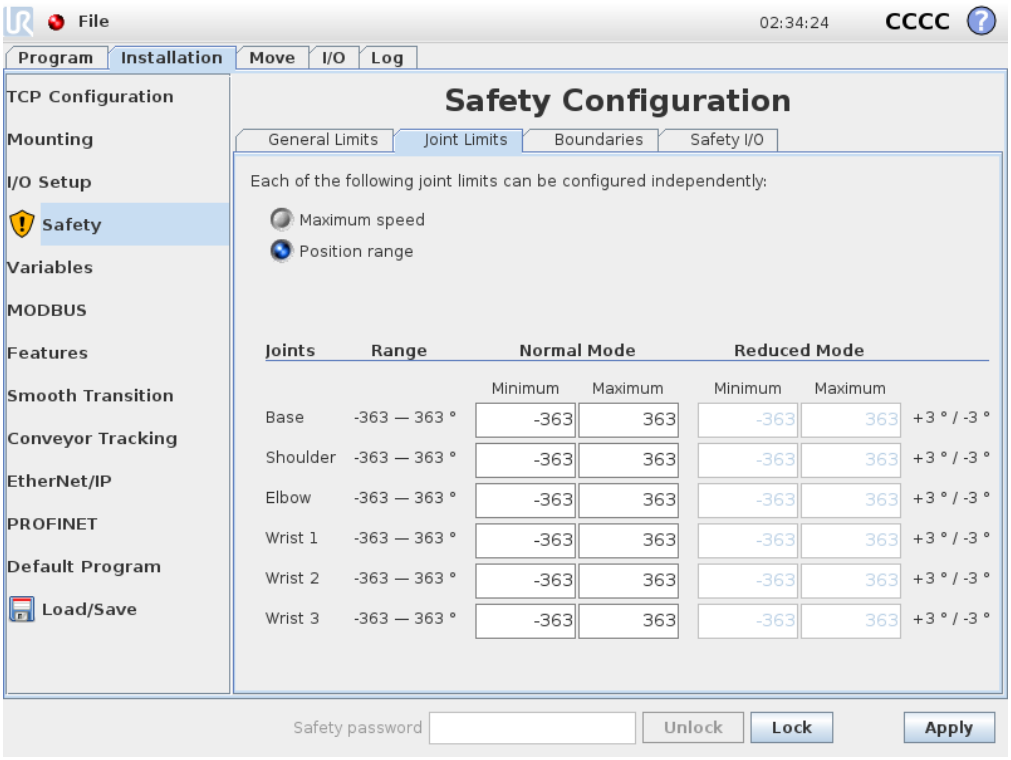
Note: The fields for limits in *Reduced* mode are disabled when neither a safety plane nor a configurable input is set to trigger it (see 10.12 and 10.13 for more details). Furthermore, the *Speed* and *Momentum* limits in *Reduced* mode must not be higher than their *Normal* mode counterparts.

The tolerance and unit for each limit are listed at the end of the row that corresponds to it. When a program is running, the speed of the robot arm is automatically adjusted in order to not exceed any of the entered values minus the tolerance (see 10.4). Note that the minus sign displayed with the tolerance value is only there to indicate that the tolerance is subtracted from the actual entered value. The safety system performs a Stop Category 0, should the robot arm exceed the limit (without tolerance).

WARNING:
The speed limit is imposed only on the robot TCP, so other parts of the robot arm may move faster than the defined value.

Switching to Basic Settings Pressing the `Basic Settings...` button switches back to the basic general limits screen and all general limits are reset to their *Default* preset. Should this cause any customized values to be lost, a popup dialog is shown to confirm the action.

10.11 Joint Limits



Joint limits restrict the movement of individual joints in joint space, i.e. they do not refer to Cartesian space but rather to the internal (rotational) position of the joints and their rotational speed. The radio buttons in the upper portion of the subpanel make it possible to independently set up `Maximum Speed` and `Position Range` for the joints.

When in *Freedrive* mode (see 13.1.5), and the current position or speed of a joint is close to the limit, the user will feel a repelling force which increases in magnitude as the joint approaches the limit. The force is generated when joint speed is within approximately 20 °/s of the speed limit or joint position is within approximately 8 ° of the position limit.

The Wrist 3 position range is unlimited by default. When using cables attached to the robot, you must first disable the *Unrestricted Range for Wrist 3* checkbox to avoid cable tension and protective stops.

Maximum Speed This option defines the maximum angular velocity for each joint. This is done by tapping the corresponding text field and entering the new value. The highest accepted value is listed in the column titled `Maximum`. None of the values can be set below the tolerance value.

Note that the fields for limits in *Reduced* mode are disabled when neither a safety plane nor a configurable input is set to trigger it (see 10.12 and 10.13 for more details). Furthermore, the limits for *Reduced* mode must not be higher than their *Normal* mode counterparts.

The tolerance and unit for each limit are listed at the end of the row that corresponds to it. When a program is running, the speed of the robot arm is automatically adjusted in order to not exceed any of the entered values minus the tolerance (see 10.4). Note that the minus sign displayed with each tolerance value is only there to indicate that the tolerance is subtracted from the actual entered value. Nevertheless, should the angular velocity of some joint exceed the entered value (without tolerance), the safety system performs a Stop Category 0.

Position Range This screen defines the position range for each joint. This is done by tapping the corresponding text fields and entering new values for the lower and upper joint position boundary. The entered interval must fall within the values listed in the column titled `Range` and the lower boundary cannot exceed the upper boundary.

Note: The fields for limits in *Reduced* mode are disabled when neither a safety plane nor a configurable input is set to trigger it (see 10.12 and 10.13 for more details).

The tolerances and unit for each limit are listed at the end of the row that corresponds to it. The first tolerance value applies to the minimum value and the second applies to the maximum value. Program execution is aborted when the position of a joint is about to exceed the range resulting from adding the first tolerance to the entered minimum value and subtracting the second tolerance from the entered maximum value, if it continues moving along the predicted trajectory. Note that the minus sign displayed with the tolerance value is only there to indicate that the tolerance is subtracted from the actual entered value. Nevertheless, should the joint position exceed the entered range, the safety system performs a Stop Category 0.

10.12 Boundaries

In this tab you can configure boundary limits consisting of safety planes and a limit on the maximum allowed deviation of the robot tool orientation. It is also possible to define planes that trigger a transition into *Reduced* mode.

Safety planes can be used to restrict the allowed workspace of the robot by enforcing that the robot TCP stay on the correct side of the defined planes and not pass through them. Up to eight safety planes can be configured. The constraint on the orientation of tool can be utilized to ensure that the robot tool orientation does not deviate more than a certain specified amount from a desired orientation.

WARNING:
Defining safety planes only limits the TCP and not the overall limit for the robot arm. This means that although a safety plane is specified, it does *not* guarantee that other parts of the robot arm will obey this restriction.

The configuration of each boundary limit is based on one of the features defined in the current robot installation (see 13.12).

NOTE:
It is highly recommended, that you create all features needed for the configuration of all the desired boundary limits and assign them appropriate names before editing the safety configuration. Note that since the robot arm is powered off once the *Safety* tab has been unlocked, the `Tool` feature (containing the current position and orientation of the robot TCP) as well as *Freedrive* mode (see 13.1.5) will not be available.

When in *Freedrive* mode (see 13.1.5), and the current position of the robot TCP is close to a safety plane, or the deviation of the orientation of the robot tool from the desired orientation is close to the specified maximum deviation, the user will feel a repelling force which increases in magnitude as the TCP approaches the limit. The force is generated when the TCP is within approximately 5 cm of a safety plane, or the deviation of the orientation of the tool is approximately 3 ° from the specified maximum deviation.

When a plane is defined as a *Trigger Reduced mode* plane and the TCP goes beyond this boundary, the safety system transitions into *Reduced* mode which applies the *Reduced* mode safety settings. Trigger planes follow the same rules as regular safety planes except they allow the robot arm to pass through them.

10.12.1 Selecting a boundary to configure

The `Safety Boundaries` panel on the left side of the tab is used to select a boundary limit to configure.

To set up a safety plane, click on one of the top eight entries listed in the panel. If the selected safety plane has already been configured, the corresponding 3D representation of the plane is highlighted in the `3D View` (see 10.12.2) to the right of this panel. The safety plane can be set up in the `Safety Plane Properties` section (see 10.12.3) at the bottom of the tab.

Click the `Tool Boundary` entry to configure the orientation boundary limit for the robot tool. The configuration of the limit can be specified in the `Tool Boundary Properties` section (see 10.12.4) at the bottom of the tab.

Click the / button to toggle the 3D visualization of the boundary limit on/off. If a boundary limit is active, the *safety mode* (see 10.12.3 and 10.12.4) is indicated by one of the following icons / / / .

10.12.2 3D visualization

The `3D View` displays the configured safety planes and the orientation boundary limit for the robot tool together with the current position of the robot arm. All configured boundary entries where the visibility toggle is selected (i.e. showing icon) in the `Safety Boundaries` section are displayed together with the current selected boundary limit.

The (active) safety planes are shown in yellow and black with a small arrow representing the plane normal, which indicates the side of the plane on which the robot TCP is allowed to be positioned. Trigger planes are displayed in blue and green. A small arrow illustrates the side of the plane that does *not* trigger the transition into *Reduced* mode. If a safety plane has been selected in the panel on the left side of the tab, the corresponding 3D representation is highlighted.

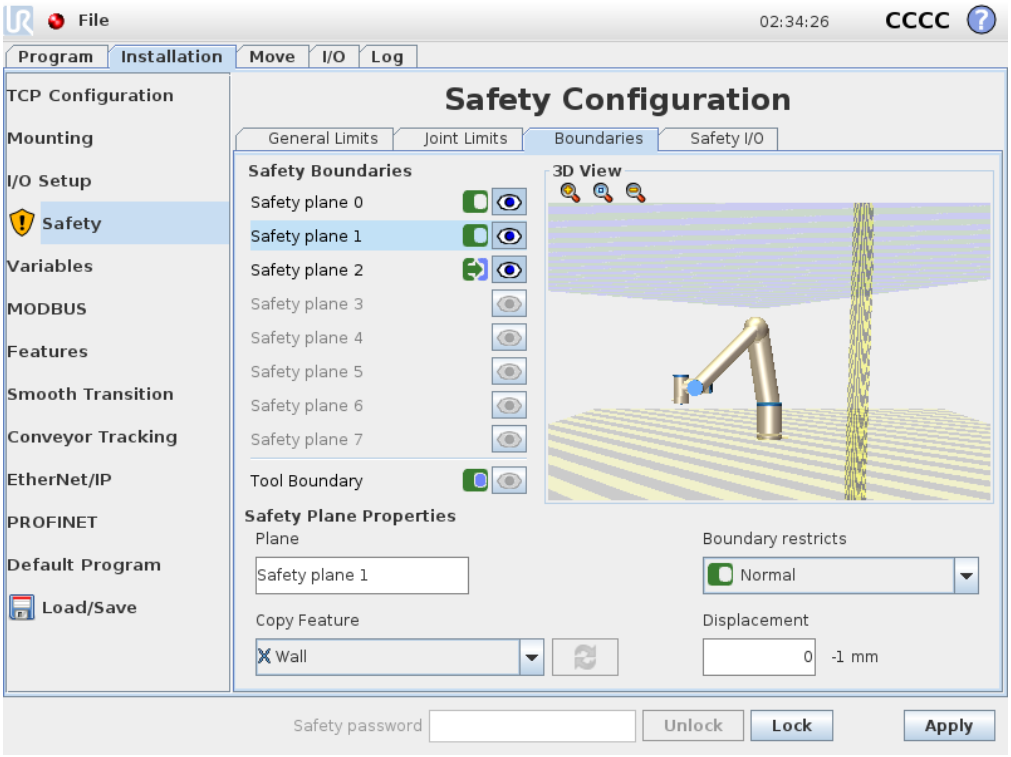
The tool orientation boundary limit is visualized with a spherical cone together with a vector indicating the current orientation of the robot tool. The inside of the cone represents the allowed area for the tool orientation (vector).

When a plane or the tool orientation boundary limit is configured but not active, the visualization is gray.

Push the magnifying glass icons to zoom in/out or drag a finger across to change the view.

10.12.3 Safety plane configuration

The `Safety Plane Properties` section at the bottom of the tab defines the configuration of the selected safety plane in the `Safety Boundaries` panel in the upper left portion of the tab.



Name The `Name` text field allows the user to assign a name to the selected safety plane. Change the name by tapping the text field and entering a new name.

Copy Feature The position and normal of the safety plane is specified using a feature (see 13.12) from the current robot installation. Use the drop-down box in the lower left portion of the `Safety Plane Properties` section to select a feature. Only the point and plane type features are available. Choosing the `<Undefined>` item clears the configuration of the plane.

The z-axis of the selected feature will point to the disallowed area and the plane normal will point in the opposite direction, except when the `Base` feature is selected, in which case the plane normal will point in the same direction. If the plane is configured as a *Trigger Reduced mode* plane (see 10.12.3), the plane normal indicates the side of the plane that does *not* trigger transition into *Reduced* mode.

It should be noted that when the safety plane has been configured by selecting a feature, the position information is only *copied* to the safety plane; the plane is *not* linked to that feature. This means that if there are changes to the position or orientation of a feature which has been used to configure a safety plane, the safety plane is not automatically updated. If the feature has changed, this is indicated by a ⚠ icon positioned over the feature selector. Click the 🔄 button next to the selector to update the safety plane with the current position and orientation of the feature. The ⚠ icon is also displayed if the selected feature has been deleted from the installation.

Safety mode The drop down menu on the right hand side of the `Safety Plane Properties` panel is used to choose the *safety mode* for the safety plane, with the following modes available:

`Disabled`	The safety plane is *never active*.
`Normal`	When the safety system is in *Normal* mode, a `Normal` mode plane is *active* and it acts as a *strict limit* on the position of the robot TCP.
`Reduced`	When the safety system is in *Reduced* mode, a `Reduced` mode plane is *active* and it acts as a *strict limit* on the position of the robot TCP.
`Normal & Reduced`	When the safety system is either in *Normal* or *Reduced* mode, a `Normal & Reduced` mode plane is *active* and it acts as a *strict limit* on the position of the robot TCP.
`Trigger Reduced mode`	When the safety system is either in *Normal* or *Reduced* mode, a *Trigger Reduced mode* plane is *active* and it causes the safety system to switch to *Reduced* mode for as long as the robot TCP is positioned beyond it.

The selected *safety mode* is indicated by an icon in the corresponding entry in the `Safety Boundaries` panel. If the *safety mode* is set to `Disabled`, no icon is shown.

Displacement When a feature has been selected in the drop down box in the lower left portion of the `Safety Plane Properties` panel, the safety plane can be translated by tapping the `Displacement` text field in the lower right portion of this panel and entering a value. Entering in a positive value increases the allowed workspace of the robot by moving the plane in the opposite direction of the plane normal, while entering a negative value decreases the allowed area by moving the plane in the direction of the plane normal.

The tolerance and unit for the displacement of the boundary plane are shown to the right of the text field.

Effect of *strict limit* planes Program execution is aborted when the TCP position is about to cross an active, strict limit safety plane minus the tolerance (see 10.4), if it continues moving along the predicted trajectory. Note that the minus sign displayed with the tolerance value is only there to indicate that the tolerance is subtracted from the actual entered value. The safety system will perform a Stop Category 0, should the TCP position exceed the specified limit safety plane (without tolerance).

Effect of *Trigger Reduced mode* planes When no protective stop is in effect and the safety system is not in the special *Recovery* mode (see 10.6), it operates either in *Normal* or *Reduced* mode and the movements of the robot arm are limited by the respective limit set.

By default, the safety system is in *Normal* mode. It transitions into *Reduced* mode whenever one of the following situations occurs:

a) The robot TCP is positioned beyond some *Trigger Reduced mode* plane, i.e. it is located on the side of the plane that is *opposite to* the direction of the small arrow in the visualization of the plane.

b) The `Reduced Mode` safety input function is configured and the input signals are low (see 10.13 for more details).

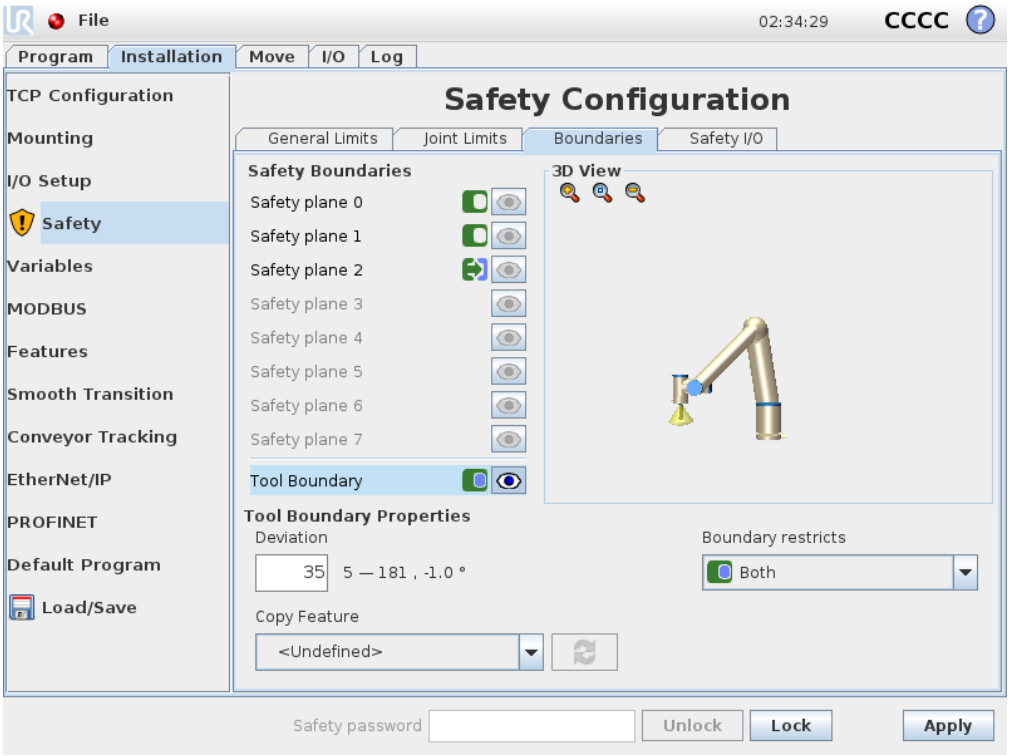
When none of the above is the case any longer, the safety system transitions back to *Normal* mode.

When the transition from *Normal* to *Reduced* mode is caused by passing through a *Trigger Reduced mode* plane, a transition from the *Normal* mode limit set to the *Reduced* mode limit set occurs. As soon as the robot TCP is positioned 20 mm or closer to the *Trigger Reduced mode* plane (but still on the *Normal* mode side), the more permissive of the *Normal* and *Reduced* mode limits is applied for each limit value. Once the robot TCP passes through the *Trigger Reduced mode* plane, the *Normal* mode limit set is no longer active and the *Reduced* mode limit set is enforced.

When a transition from *Reduced* to *Normal* mode is caused by passing through a *Trigger Reduced mode* plane, a transition from the *Reduced* mode limit set to the *Normal* mode limit set occurs. As soon as the robot TCP passes through the *Trigger Reduced mode* plane, the more permissive of the *Normal* and *Reduced* mode limits is applied for each limit value. Once the robot TCP is positioned 20 mm or further from the *Trigger Reduced mode* plane (on the *Normal* mode side), the *Reduced* mode limit set is no longer active and the *Normal* mode limit set is enforced.

If the predicted trajectory takes the robot TCP through a *Trigger Reduced mode* plane, the robot arm will start decelerating even before passing through the plane if it is about to exceed joint speed, tool speed or momentum limit in the new limit set. Note that since these limits are required to be more restrictive in the *Reduced* mode limit set, such premature deceleration can occur only when transitioning from *Normal* to *Reduced* mode.

10.12.4 Tool Boundary configuration



The `Tool Boundary Properties` panel at the bottom of the tab defines a limit on the orientation of robot tool composed of a desired tool orientation and a value for the maximum allowed deviation from this orientation.

Deviation The `Deviation` text field shows the value for the maximum allowed deviation of the orientation of the robot tool from the desired orientation. Modify this value by tapping the text field and entering the new value.

The accepted value range together with the tolerance and unit of the deviation are listed next to the text field.

Copy Feature The desired orientation of the robot tool is specified using a feature (see 13.12) from the current robot installation. The z-axis of the selected feature will be used as the desired tool orientation vector for this limit.

Use the drop down box in the lower left portion of the `Tool Boundary Properties` panel to select a feature. Only the point and plane type features are available. Choosing the `<Undefined>` item clears the configuration of the plane.

It should be noted that when the limit has been configured by selecting a feature, the orientation information is only *copied* to the limit; the limit is *not* linked to that feature. This means that if there are changes to the position and orientation of a feature, which has been used to configure the limit, the limit is not automatically updated. If the feature has changed, this is indicated by a ⚠ icon

positioned over the feature selector. Click the button next to the selector to update the limit with the current orientation of the feature. The icon is also displayed if the selected feature has been deleted from the installation.

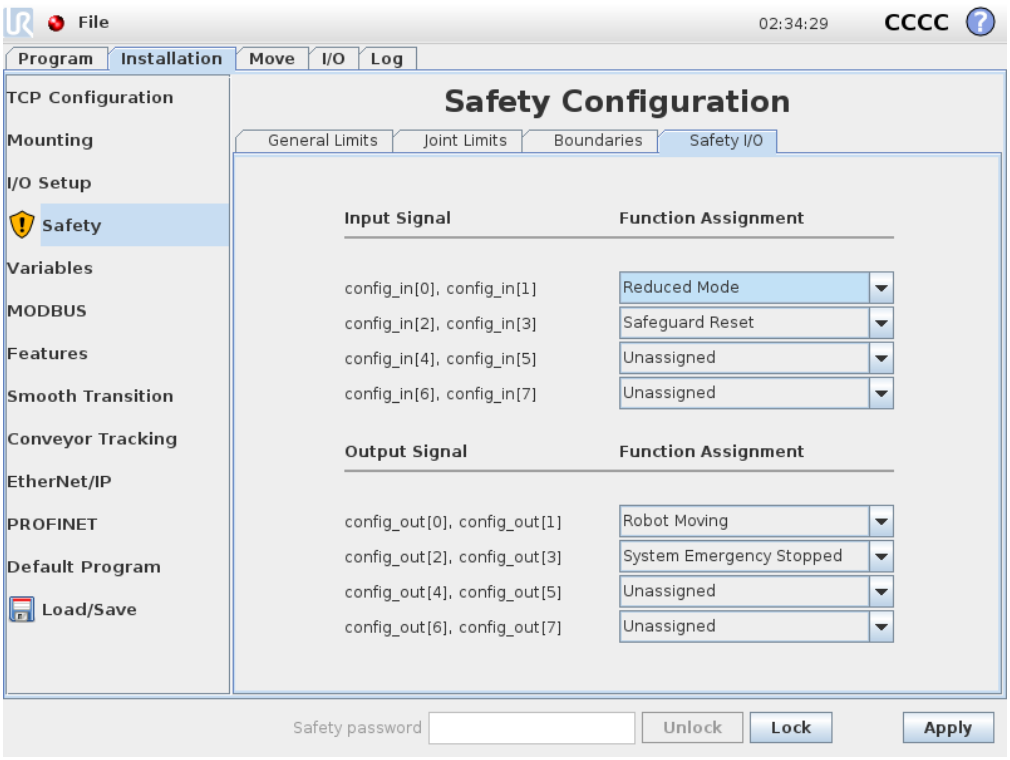
Safety mode The drop down menu on the right hand side of the `Tool Boundary Properties` panel is used to choose the *safety mode* for the tool orientation boundary. The available options are:

`Disabled`	The tool boundary limit is never active.
`Normal`	When the safety system is in *Normal* mode, the tool boundary limit is active.
`Reduced`	When the safety system is in *Reduced* mode, the tool boundary limit is active.
`Normal & Reduced`	When the safety system is either in *Normal* or *Reduced* mode, the tool boundary limit is active.

The selected *safety mode* is indicated by an icon in the corresponding entry in the `Safety Boundaries` panel. If the *safety mode* is set to `Disabled`, no icon is shown.

Effect Program execution is aborted when the deviation of the tool orientation is about to exceed the entered maximum deviation minus the tolerance (see 10.4), if it continues moving along the predicted trajectory. Note that the minus sign displayed with the tolerance value is only there to indicate that the tolerance is subtracted from the actual entered value. The safety system will perform a Stop Category 0, should the deviation of the tool orientation exceed the limit (without tolerance).

10.13 Safety I/O



This screen defines the *Safety functions* for configurable inputs and outputs (I/Os). The I/Os are divided between the inputs and outputs, and are paired up so that each function is providing a Category[2] 3 and PLd I/O.

Each *Safety function* can only control one pair of I/Os. Trying to select the same safety function a second time removes it from the first pair of I/Os previously defined. There are 5 *Safety functions* for input signals, and 5 for output signals.

Note: Applying a safety function to a set of pins will override any I/O actions specified for pins in the I/O Setup (see 13.8).

10.13.1 Input Signals

For input signals, the following *Safety functions* can be selected: `System Emergency Stop`, `Reduced Mode`, `Safeguard Reset`, `Three-Position Enabling Device` and `Operational Mode`.

`System Emergency Stop` When configured, it allows for having an additional `Emergency Stop` button besides the `Emergency Stop Button` on the Teach Pendant. This functionality requires the use of an ISO 13850-compliant device.

[2] According to ISO 13849-1, see Glossary for more details.

Reduced Mode All safety limits have two modes in which they can be applied: *Normal* mode, which specifies the default safety configuration, and *Reduced* mode (see 10.6 for more details). When this input safety function is selected, a low signal given to the inputs causes the safety system to transition to *Reduced* mode. If necessary, the robot arm then decelerates to satisfy the *Reduced* mode limit set. Should the robot arm still violate any of the *Reduced* mode limits, it performs a Stop Category 0. The transition back to *Normal* mode happens in the same manner. Note that safety planes can also cause a transition to *Reduced* mode (see 10.12.3 for more details).

Safeguard Reset If `Safeguard Stop` is wired in the safety I/Os, then this input is used to ensure the Safeguard Stopped state continues until a reset is triggered. The robot arm will not move when in Safeguard Stopped state.

WARNING:
By default, the `Safeguard Reset` function is configured for input pins 0 and 1. Disabling it altogether implies that the robot arm ceases to be Safeguard Stopped as soon as the `Safeguard Stop` input becomes high. In other words, without a `Safeguard Reset`, the `Safeguard Stop` inputs SI0 and SI1 (see the Hardware Installation Manual) fully determine whether the Safeguard Stopped state is active or not.

Three-Position Enabling Device and Operational Mode These allow for using a 3-position enabling device as an additional protective measure during setup and programming of the robot. With the `Three-Position Enabling Device` input configured, the robot is either in "running mode" or "programming mode". An icon will appear in the upper right corner displaying the current operational mode:

- *Running mode*: Robot can perform only pre-defined tasks. The Move tab and Freedrive mode are unavailable.
- *Programming mode*: The restrictions present in *Running mode* are lifted. However, whenever the `Three-Position Enabling Device` input is low, the robot is Safeguard Stopped. Also, the speed slider is set at an initial value that corresponds to 250 mm/s and can be incrementally increased to reach higher speed. The speed slider is reset to the low value whenever the `Three-Position Enabling Device` input goes from low to high.

There are two methods for configuring operational mode selection:

1. To select the operational mode using an external mode selection device, configure the `Operational Mode` input. The option to configure it will appear in the drop-down menus once the `Three-Position Enabling Device` input is configured. The robot will be in *Running mode* when the `Operational Mode` input is low and in *Programming mode* when it is high.
2. To select the operational mode from Polyscope, only the `Three-Position Enabling Device` input must be configured and applied to the Safety Configuration. In this case, the default

mode is *Running*. In order to switch to *Programming mode*, choose the "Program Robot" button on the Welcome screen. To switch back to *Running mode*, simply exit the "Program Robot" screen.

NOTE:

- After the Safety I/O configuration with `Three-Position Enabling Device` enabled is confirmed, the Welcome screen is automatically shown. The Welcome screen is also automatically displayed when the operational mode changes from *Programming* to *Running*.
- The physical mode selector, if used, must completely adhere to ISO 10218-1: article 5.7.1 for selection.
- The 3-position switch, along with its behavior, performance characteristics and operation, must thoroughly comply with ISO 10218-1: article 5.8.3 for an enabling device.

10.13.2 Output Signals

For the output signals the following *Safety functions* can be applied. All signals return to low when the state which triggered the high signal has ended:

`System Emergency Stop` Low signal is only given when the safety system has been triggered into an `Emergency Stopped` state by the Robot Emergency Stop input or Emergency Stop Button. To avoid deadlocks, if the `Emergency Stopped` state is triggered by the System Emergency Stop input, low signal will not be given.

NOTE:

External machinery obtaining the `Emergency Stop` state from the robot through `System Emergency Stop` output must be complying with ISO 13850. This is particularly necessary in setups where the Robot Emergency Stop input is connected to an external Emergency Stop device. In such cases, the `System Emergency Stop` output will become high when the external Emergency Stop device is released. This implies that the emergency stop state at the external machinery will be reset with no manual action needed from the robot's operator. Hence, to comply with safety standards, the external machinery must require manual action in order to resume.

`Robot Moving` A low signal is given whenever the robot arm is in a mobile state. When the robot arm is in a fixed position, a high signal is given.

`Robot Not Stopping` The signal is High when the robot is stopped or in the process of stopping due to an emergency stop or safeguard stop. Otherwise the signal is logic low.

`Reduced Mode` Sends a low signal when the robot arm is placed in *Reduced* mode or if the safety input is configured with a `Reduced Mode` input and the signal is currently low. Otherwise the signal is high.

`Not Reduced Mode` This is the inverse of the `Reduced Mode` defined above.

11 Begin programming

11.1 Introduction

The Universal Robot arm is composed of tubes and joints. The joints with their usual names are shown in Figure 11.1. The **Base** is where the robot is mounted, and at the other end (**Wrist 3**) the tool of the robot is attached. By coordinating the motion of each of the joints, the robot can move its tool around freely, with the exception of the area directly above and directly below the base.

PolyScope is the graphical user interface (GUI) which lets you operate the robot arm and control box, execute robot programs and easily create new ones.

The following section gets you started with the robot. Afterwards, the screens and functionality of PolyScope are explained in more detail.

DANGER:

1. The Hardware Installation Manual contains important safety information, which must be read and understood by the integrator of UR robots before the robot is powered on for the first time.
2. The integrator must set the safety configuration parameters defined by the risk assessment before powering on the robot arm for the first time, see chapter 10.

11.2 Getting Started

Before using PolyScope, the robot arm and control box must be installed and the control box switched on.

11.2.1 Installing the Robot Arm and Control Box

To install the robot arm and control box, do the following:

1. Unpack the robot arm and the control box.
2. Mount the robot on a sturdy surface strong enough to withstand at least 10 times the full torque of the base joint and at least 5 times the weight of the robot arm. The surface shall be vibration-free.
3. Place the control box on its foot.
4. Plug on the robot cable between the robot and the control box.
5. Plug in the mains plug of the control box.

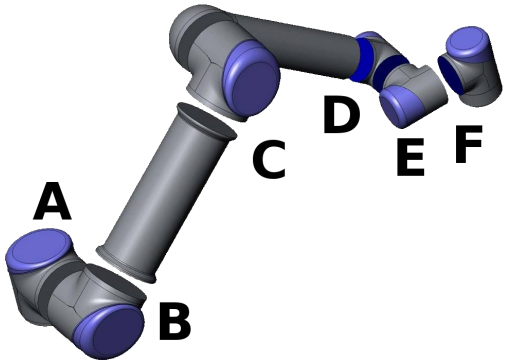


Figure 11.1: Joints of the robot. A: *Base*, B: *Shoulder*, C: *Elbow* and D, E, F: *Wrist 1, 2, 3*

WARNING:
Tipping hazard. If the robot is not securely placed on a sturdy surface, the robot can fall over and cause an injury.

Detailed installation instructions can be found in the Hardware Installation Manual. Note that a risk assessment is required before using the robot arm to do any work.

11.2.2 Turning the Control Box On and Off

The control box is turned on by pressing the power button at the front side of the panel with the touch screen. This panel is usually referred to as the *teach pendant*. When the control box is turned on, text from the underlying operating system will appear on the touch screen. After about one minute, a few buttons appear on the screen and a popup guides the user to the initialization screen (see 11.5).

To shut down the control box, press the green power button on the screen, or use the `Shut Down` button on the welcome screen (see 11.4).

WARNING:
Shutting down by pulling the power cord from the wall socket may cause corruption of the robot's file system, which may result in robot malfunction.

11.2.3 Turning the Robot Arm On and Off

The robot arm can be turned on if the control box is turned on, and if no emergency stop button is activated. Turning the robot arm on is done in the initialization screen (see 11.5) by touching the `ON` button on that screen, and then pressing `Start`. When a robot is started, it makes a sound and moves a little while releasing the brakes.

The power to the robot arm can be turned off by touching the OFF button on the initialization screen. The robot arm is also powered off automatically when the control box shuts down.

11.2.4 Quick Start

To quickly start up the robot after it has been installed, perform the following steps:

1. Press the emergency stop button on the front side of the teach pendant.
2. Press the power button on the teach pendant.
3. Wait a minute while the system is starting up, displaying text on the touch screen.
4. When the system is ready, a popup will be shown on the touch screen, stating that the robot needs to be initialized.
5. Touch the button on the popup dialog. You will be taken to the initialization screen.
6. Wait for the Confirmation of applied Safety Configuration dialog and press the Confirm Safety Configuration button. This applies an initial set of safety parameters that need to be adjusted based on a risk assessment.
7. Unlock the emergency stop button. The robot state changes from Emergency Stopped to Power off.
8. Step outside the reach (workspace) of the robot.
9. Touch the On button on the touch screen. Wait a few seconds until robot state changes to Idle.
10. Verify that the payload mass and selected mounting are correct. You will be notified if the mounting detected based on sensor data does not match the selected mounting.
11. Touch the Start button on the touch screen. The robot now makes a sound and moves a little while releasing the brakes.
12. Touch the OK button, bringing you to the Welcome screen.

11.2.5 The First Program

A program is a list of commands telling the robot what to do. PolyScope allows people with only little programming experience to program the robot. For most tasks, programming is done entirely using the touch panel without typing in any cryptic commands.

Tool motion is the part of a robot program that teaches the Robot Arm how to move. In PolyScope, tool motions are set using a series of **waypoints**. The combined waypoints form a path that the Robot Arm follows. A waypoint is set by using the Move Tab, manually moving (teaching) the robot to a certain position, or it can be calculated by software. Use the Move tab (see 13.1) to move the Robot Arm to a desired position, or teach the position by pulling the Robot Arm into place while holding the Freedrive button behind the Teach Pendant.

Besides moving through waypoints, the program can send I/O signals to other machines at certain points in the robot's path, and perform commands like **if...then** and **loop**, based on variables and I/O signals.

The following is a simple program that allows a Robot arm that has been started up, to move between two waypoints.

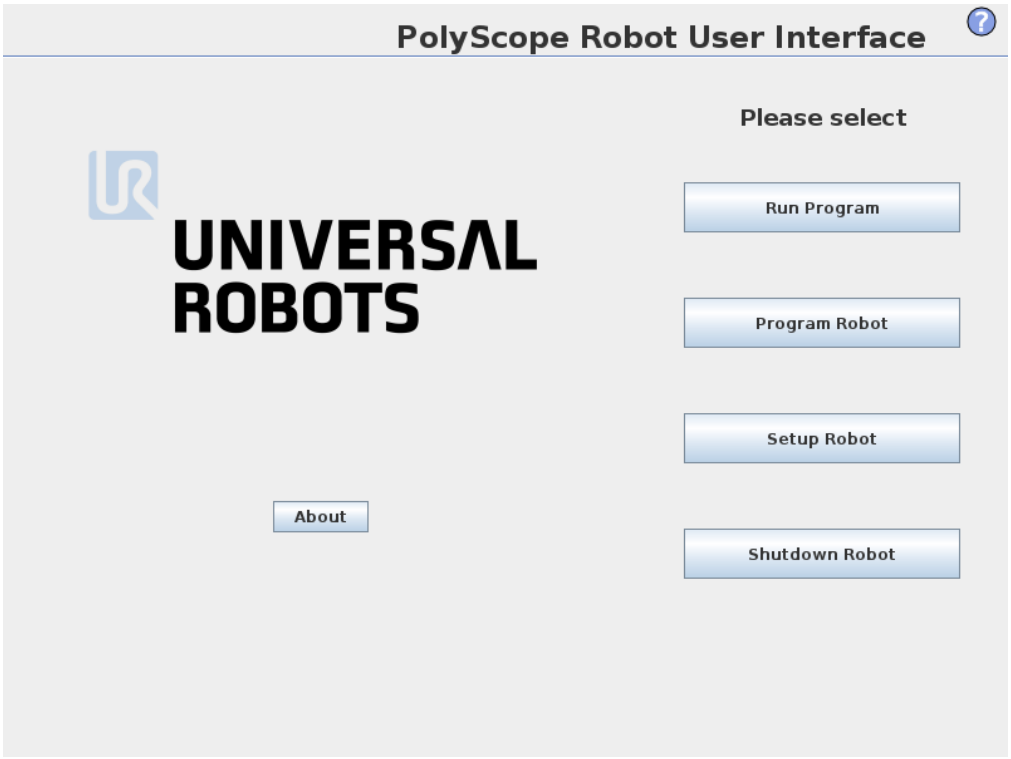
1. Touch the Program Robot button and select Empty Program.
2. Touch the Next button (bottom right) so that the <empty> line is selected in the tree structure on the left side of the screen.
3. Go to the Structure tab.
4. Touch the Move button.
5. Go to the Command tab.
6. Press the Next button, to go to the Waypoint settings.
7. Press the Set this waypoint button next to the "?" picture.
8. On the Move screen, move the robot by pressing the various blue arrows, or move the robot by holding the Freedrive button, placed on the backside of the teach pendant, while pulling the robot arm.
9. Press OK.
10. Press Add waypoint before.
11. Press the Set this waypoint button next to the "?" picture.
12. On the Move screen, move the robot by pressing the various blue arrows, or move the robot by holding the Freedrive button while pulling the robot arm.
13. Press OK.
14. Your program is ready. The robot will move between the two points when you press the "Play" symbol. Stand clear, hold on to the emergency stop button and press "Play".
15. Congratulations! You have now produced your first robot program that moves the robot between the two given waypoints.

WARNING:

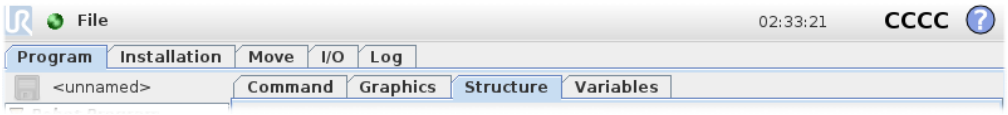
1. Do not drive the robot into itself or anything else as this may cause damage to the robot.
2. Keep your head and torso outside the reach (workspace) of the robot. Do not place fingers where they can be caught.
3. This is only a quick start guide to show how easy it is to use a UR robot. It assumes a harmless environment and a very careful user. Do not increase the speed or acceleration above the default values. Always conduct a risk assessment before placing the robot into operation.

11.3 PolyScope Programming Interface

PolyScope runs on the touch sensitive screen attached to the control box.



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.

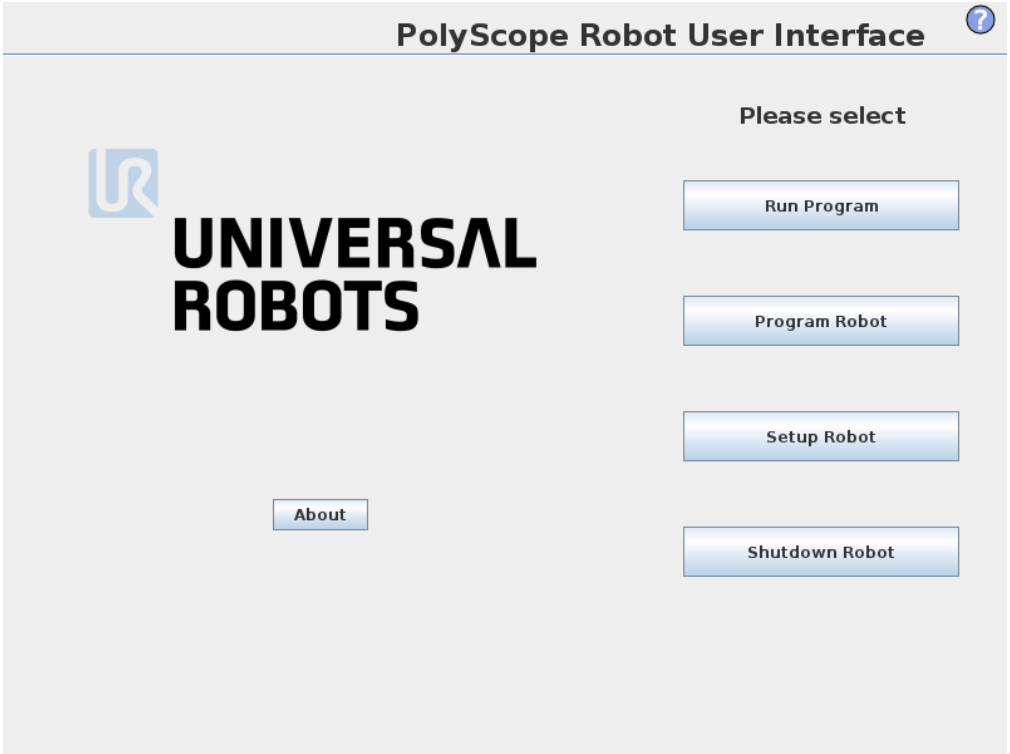


In this example, the **Program** tab is selected at the top level, and under that the `Structure` tab is selected. The `Program` tab holds information related to the currently loaded program. If the **Move** tab is selected, the screen changes to the **Move** screen, from where the robot arm can be moved. 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 control 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.

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

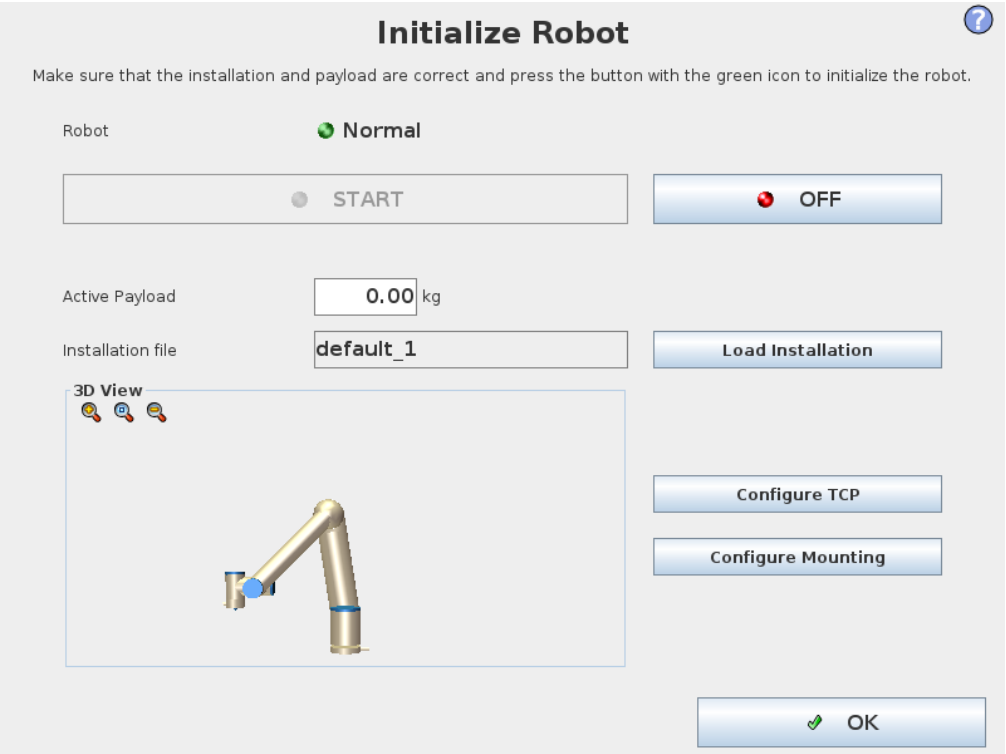
11.4 Welcome Screen



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

- **Run Program:** Choose and run an existing program. This is the simplest way to operate the robot arm and control box.
- **Program Robot:** Change a program, or create a new program.
- **Setup Robot:** Change the language, set passwords, upgrade software, etc.
- **Shutdown Robot:** Powers off the robot arm and shuts down the control box.
- **About:** Provides details related to software versions, hostname, IP address, serial number and legal information.

11.5 Initialization Screen



On this screen you control the initialization of the robot arm.

Robot arm state indicator

The status LED gives an indication of the robot arm's running state:

- A bright red LED indicates that the robot arm is currently in a stopped state where the reasons can be several.
- A bright yellow LED indicates that the robot arm is powered on, but is not ready for normal operation.
- Finally, a green LED indicates that the robot arm is powered on, and ready for normal operation.

The text appearing next to the LED further specifies the current state of the robot arm.

Active payload and installation

When the robot arm is powered on, the payload mass used by the controller when operating the robot arm is shown in the small white text field. This value can be modified by tapping the text field and entering a new value.
Note: setting this value does not modify the payload in the robot's installation (see 13.6), it only sets the payload mass to be used by the controller.

Similarly, the name of the installation file that is currently loaded is shown in the grey text field. A different installation can be loaded by tapping the text field or by using the **Load** button next to it. Alternatively, the loaded installation can be customized using the buttons next to the 3D view in the lower part of the screen.

Before starting up the robot arm, it is very important to verify that both the active payload and the active installation correspond to the actual situation the robot arm is currently in.

Initializing the robot arm

DANGER:
Always verify that the actual payload and installation are correct before starting up the robot arm. If these settings are wrong, the robot arm and control box will not function correctly and may become dangerous to people or equipment around them.

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

The large button with the green icon on it serves to perform the actual initialization of the robot arm. The text on it, and the action it performs, change depending on the current state of the robot arm.

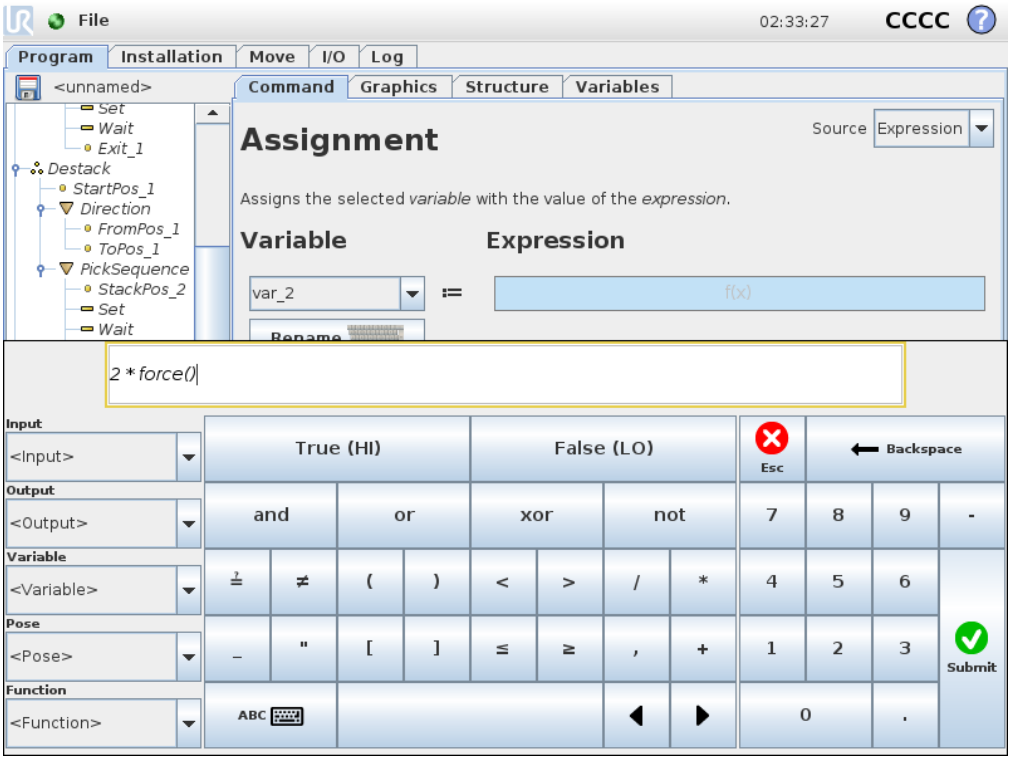
- After the controller PC boots up, the button needs to be tapped once to power the robot arm on. The robot arm state then turns to **Power on** and subsequently to **Idle**. Note that when an emergency stop is in place, the robot arm cannot be powered on, so the button will be disabled.
- When the robot arm state is **Idle**, the button needs to be tapped once again to start the robot arm up. At this point, sensor data is checked against the configured mounting of the robot arm. If a mismatch is found (with a tolerance of 30°), the button is disabled and an error message is displayed below it.

  If the mounting verification passes, tapping the button releases all joint brakes and the robot arm becomes ready for normal operation. Note that the robot makes a sound and moves a little while releasing the brakes.
- If the robot arm violates one of the safety limits after it starts up, it operates in a special **Recovery mode**. In this mode, tapping the button switches to a recovery move screen where the robot arm can be moved back within the safety limits.
- If a fault occurs, the controller can be restarted using the button.
- If the controller is currently not running, tapping the button starts it.

Finally, the smaller button with the red icon on it serves to power off the robot arm.

12 On-screen Editors

12.1 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 left 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`.

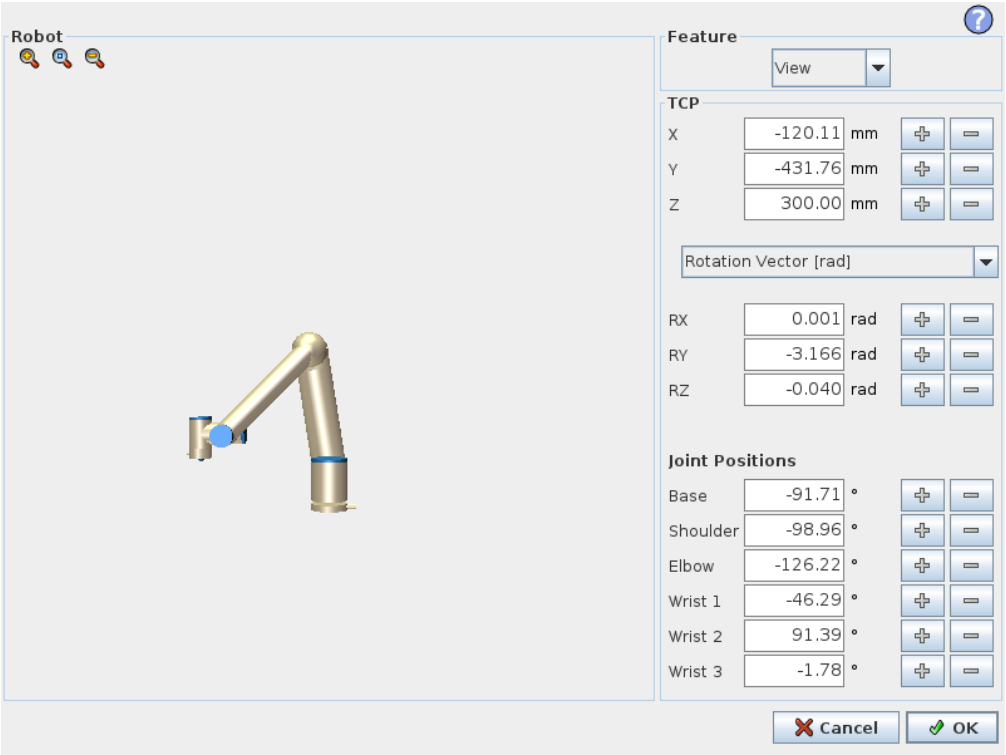
The expression is checked for grammatical errors when the `Ok` button is pressed. The `Cancel` button leaves the screen, discarding all changes.

An expression can look like this:

$$\texttt{digital\_in[1]} \overset{?}{=} \texttt{True and analog\_in[0]<0.5}$$

12.2 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 robot arm directly.



Robot

The current position of the robot arm and the specified new target position are shown in 3D graphics. The 3D drawing of the robot arm shows the current position of the robot arm, and the "shadow" of the robot arm shows the target position of the robot arm 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.

If the specified target position of the robot TCP is close to a safety or trigger plane, or the orientation of robot tool is near the tool orientation boundary limit (see 10.12), a 3D representation of the proximate boundary limit is shown.

Safety planes are visualized in yellow and black with a small arrow representing the plane normal, which indicates the side of the plane on which the robot TCP is allowed to be positioned. Trigger planes are displayed in blue and green and a small arrow pointing to the side of the plane, where the *Normal* mode limits (see 10.6) are active. The tool orientation boundary limit is visualized with a spherical cone together with a vector indicating the current orientation of the robot tool. The inside of the cone represents the allowed area for the tool orientation (vector).

When the target robot TCP no longer is in the proximity of the limit, the 3D representation disappears. If the target TCP is in violation or very close to violating a boundary limit, the visualization of the limit turns red.

Feature and tool position

In the top right corner of the screen, the feature selector can be found. The feature selector defines which feature to control the robot arm relative to

Below the feature selector, the name of the currently active Tool Center Point (TCP) is displayed. For further information about configuring several named TCPs, see 13.6. The text boxes show the full coordinate values of that TCP 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 [°]** 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 [°]** *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 $-360°$ to $+360°$, 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 13.1), clicking the `OK` button will return to the `Move` tab, where the robot arm will move to the specified target. If the last specified value was a tool coordinate, the robot arm will move to the target position using the *MoveL* movement type, while the robot arm 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 14.4.

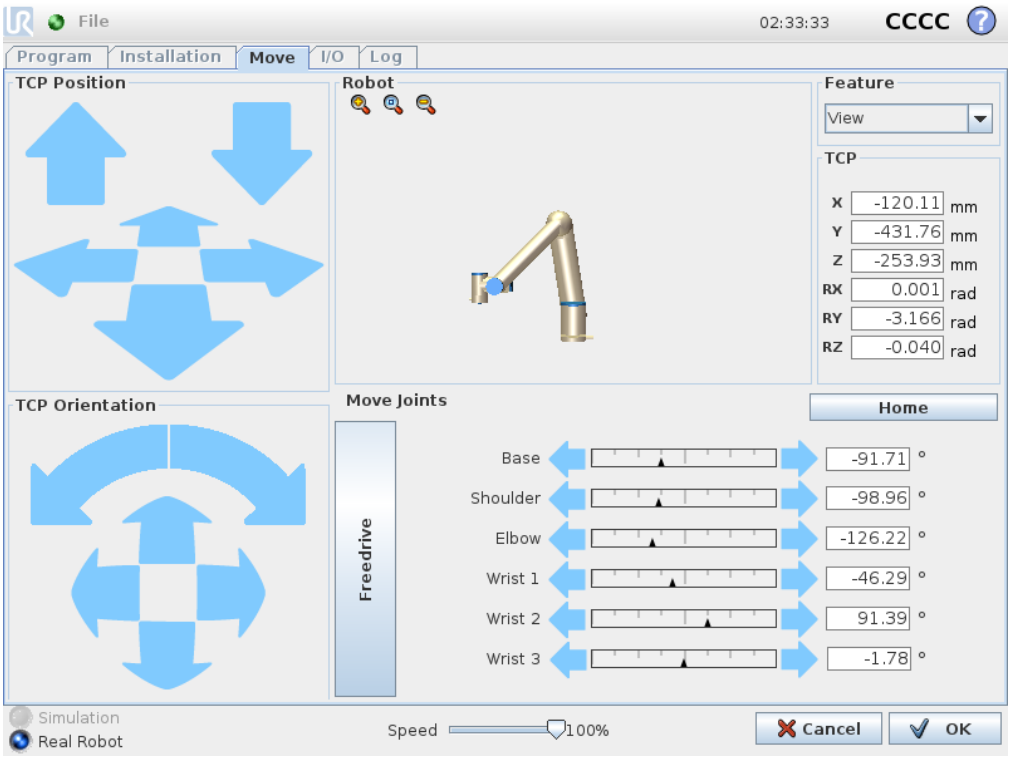
Cancel button

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

13 Robot Control

13.1 Move Tab

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



13.1.1 Robot

The current position of the robot arm 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 arm, select the **View** feature and rotate the viewing angle of the 3D drawing to match your view of the real robot arm.

If the current position of the robot TCP comes close to a safety or trigger plane, or the orientation of robot tool is near the tool orientation boundary limit (see 10.12), a 3D representation of the proximate boundary limit is shown. Note that when the robot is running a program, the visualization of boundary limits will be disabled.

Safety planes are visualized in yellow and black with a small arrow representing the plane normal, which indicates the side of the plane on which the robot TCP is allowed to be positioned. Trigger planes are displayed in blue and green and a small arrow pointing to the side of the plane, where the **Normal** mode limits (see 10.6) are active. The tool orientation boundary limit is visualized

with a spherical cone together with a vector indicating the current orientation of the robot tool. The inside of the cone represents the allowed area for the tool orientation (vector).

When the robot TCP no longer is in the proximity of the limit, the 3D representation disappears. If the TCP is in violation or very close to violating a boundary limit, the visualization of the limit turns red.

13.1.2 Feature and Tool Position

In the top right corner of the screen, the feature selector can be found. It defines which feature to control the robot arm relative to.

The name of the currently active Tool Center Point (TCP) is displayed below the feature selector. The text boxes show the full coordinate values of that TCP relative to the selected feature. For further information about configuring several named TCPs, (see 13.6).

Values can be edited manually by clicking on the coordinate or the joint position. This will take you to the pose editor screen (see 12.2) where you can specify a target position and orientation for the tool or target joint positions.

13.1.3 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` (bottom) will change the orientation of the robot tool in the indicated direction. The point of rotation is the Tool Center Point (TCP), i.e. the point at the end of the robot arm that gives a characteristic point on the robot's tool. The TCP is shown as a small blue ball.

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

13.1.4 Move Joints

Allows the individual joints to be controlled directly. Each joint can move from -360° to $+360^\circ$, which are the default *joint limits* illustrated by the horizontal bar for each joint. If a joint reaches its joint limit, it cannot be driven any further. If the limits for a joint have been configured with a position range different from the default (see 10.11), this range is indicated with red in the horizontal bar.

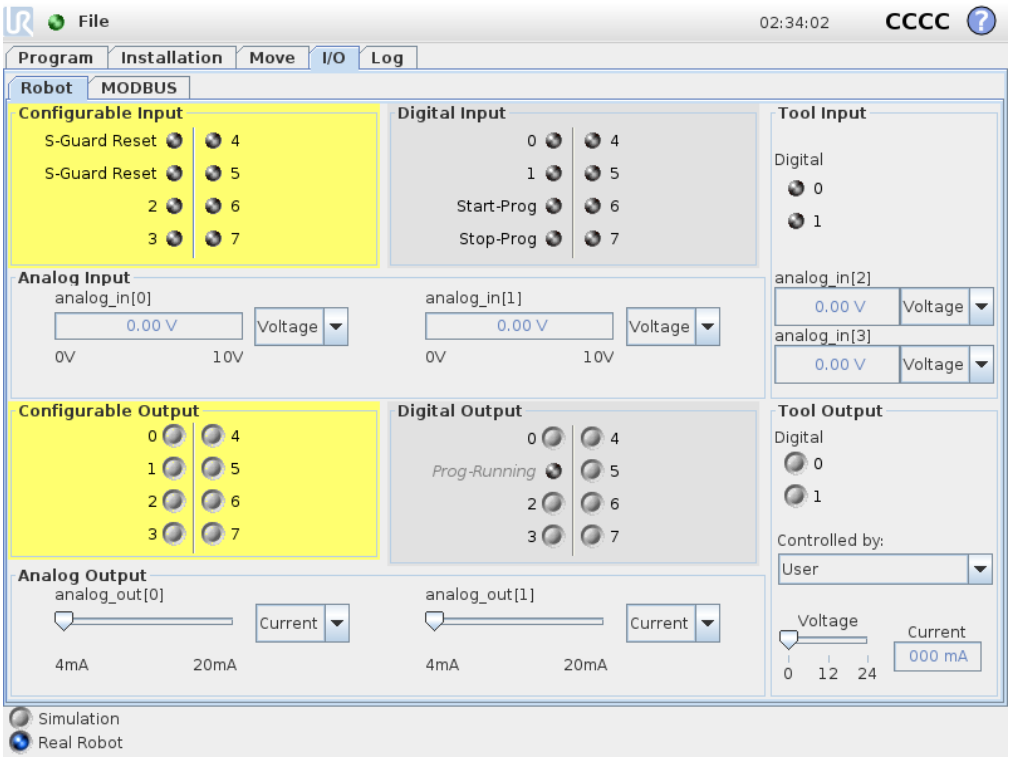
13.1.5 Freedrive

While the *Freedrive* button is held down, it is possible to physically grab the robot arm and pull it to where you want it to be. If the gravity setting (see 13.7) in the **Setup** tab is wrong, or the robot arm carries a heavy load, the robot arm might start moving (falling) when the **Freedrive** button is pressed. In that case, just release the **Freedrive** 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 **Freedrive** button. If these settings are not correct, the robot arm will move when the **Freedrive** button is activated.
3. The Freedrive 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 arm.

13.2 I/O Tab



On this screen you can always monitor and set the live I/O signals from/to the robot control box. 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.

Configurable I/O's can be reserved for special safety settings defined in the safety I/O configuration section of the installation (see 10.13); those which are reserved will have the name of the safety function in place of the default or user defined name. Configurable outputs that are reserved for safety settings are not togglable and will be displayed as LED's only.

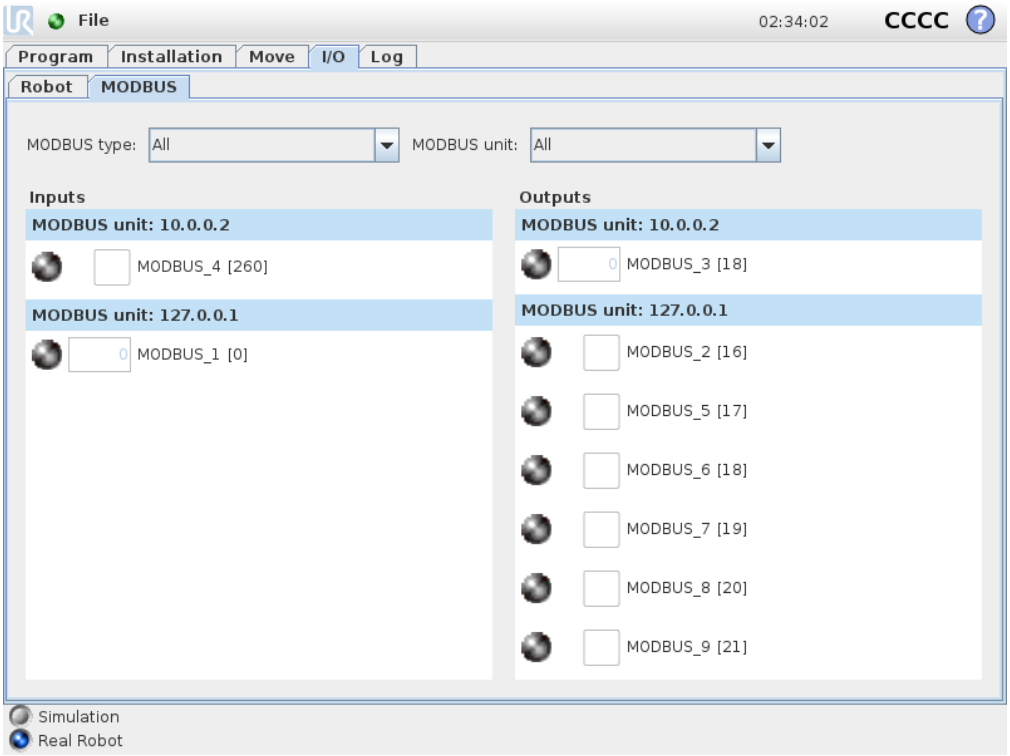
The electrical details of the signals are described in chapter **??**.

Voltage In Tool Output, Voltage can be configured only when Tool Output is controlled by the User. Selecting a URCap removes access to the Voltage.

Analog Domain Settings The analog I/O's can be set to either current [4-20mA] or voltage [0-10V] output. The settings will be remembered for eventual later restarts of the robot controller when a program is saved. Selecting a URCap, in Tool Output, removes access to the Domain Settings for the analog Tool Inputs.

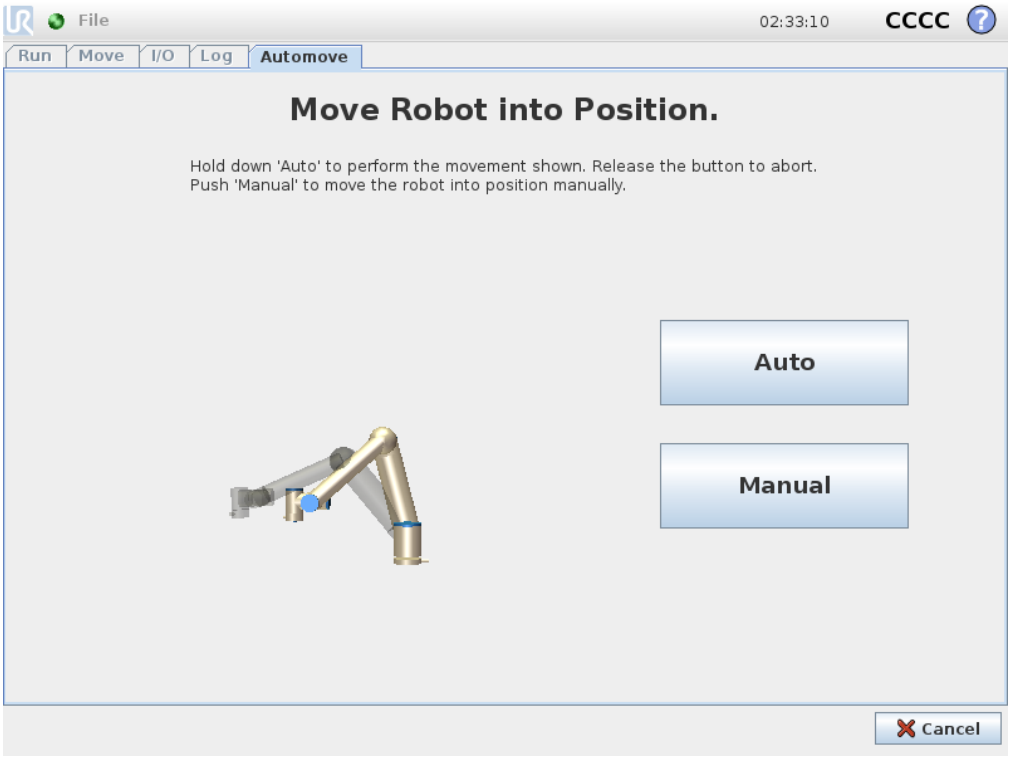
13.3 MODBUS

The screenshot below displays the MODBUS client I/O signals as they are set up in the installation. Using the drop-down menus at the top of the screen, you can change the displayed content based on signal type and MODBUS unit if more than one is configured. Each signal in the lists contains its connections status, value, name, and signal address. The output signals can be toggled if the connection status and the choice for I/O tab control allows it (see 13.8).



13.4 AutoMove Tab

The AutoMove tab is used when the robot arm has to move to a specific position in its workspace. Examples are when the robot arm 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 arm is about to perform.

CAUTION:
Compare the animation with the position of the real robot arm and make sure that the robot arm can safely perform the movement without hitting any obstacles.

CAUTION:
The automove function moves along the robot along the shadow trajectory. Collision might damage the robot or other equipment.

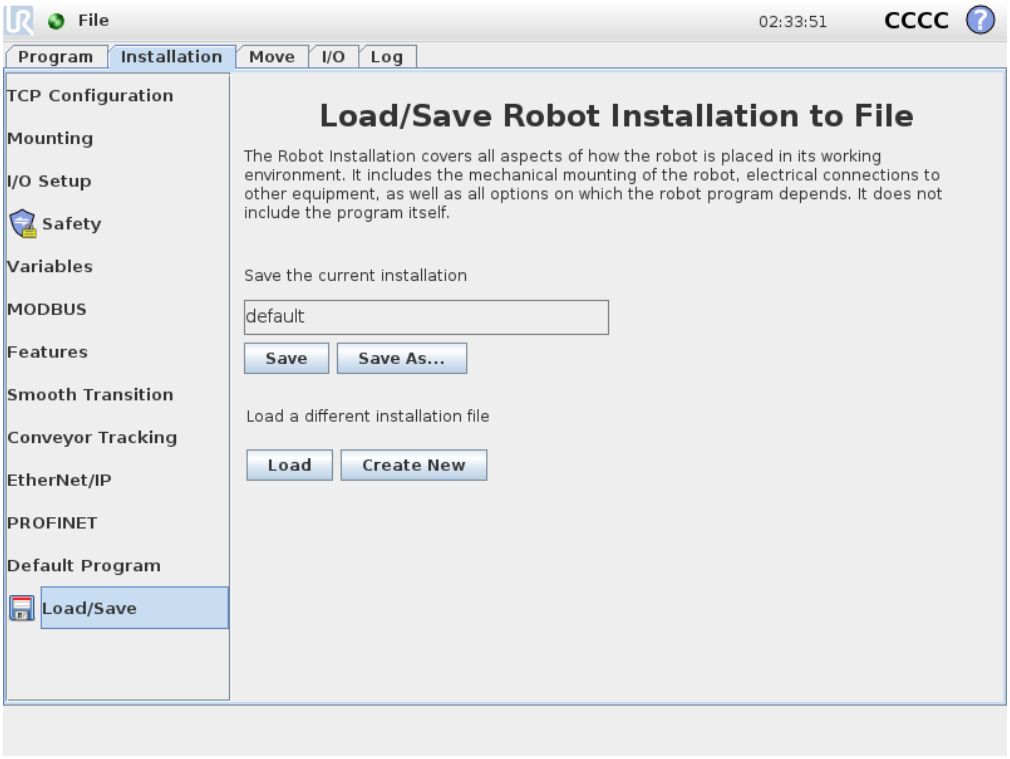
Auto

Hold down the **Auto** button to move the robot arm 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 **Move** tab where the robot arm can be moved manually. This is only needed if the movement in the animation is not preferable.

13.5 Installation → Load/Save



The Robot Installation covers all aspects of how the robot arm and control box are placed in the working environment. It includes the mechanical mounting of the robot arm, electrical connections to other equipment, as well as all options on which the robot program depends. It does not include the program itself.

These settings can be set using the various screens under the **Installation** tab, except for the I/O domains which are set in the **I/O** tab (see 13.2).

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.

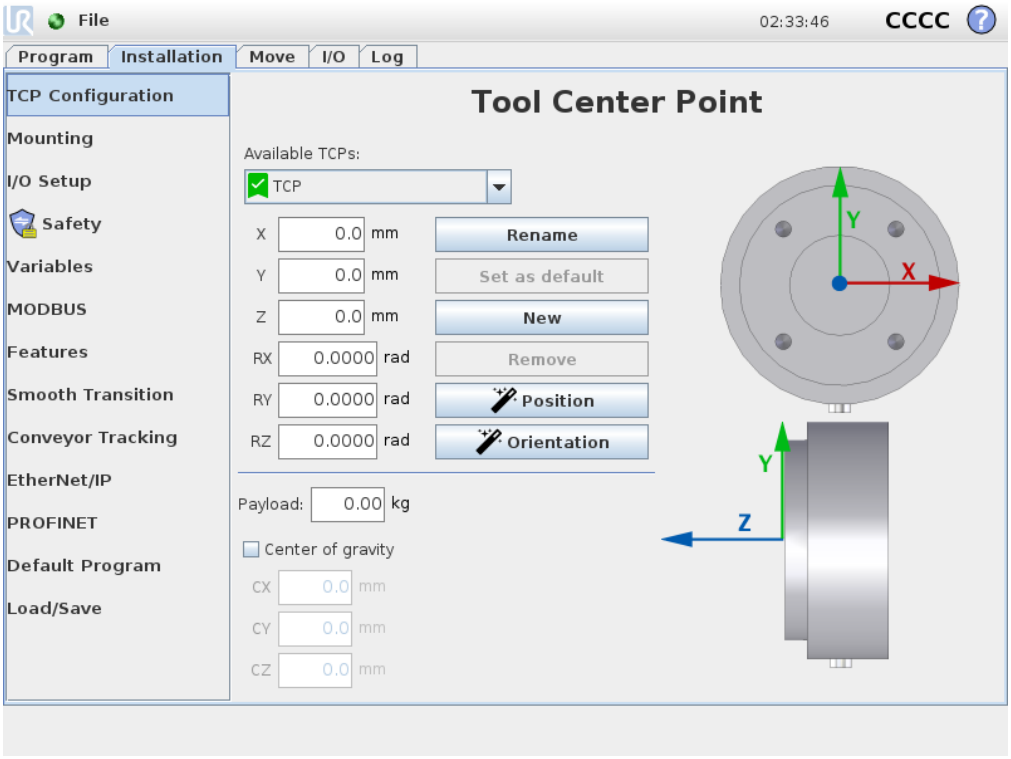
Any changes to an installation need to be saved to be preserved after power down. If there are unsaved changes in the installation, a floppy disk icon is shown next to the **Load/Save** text on the left side of the **Installation** tab.

Saving an installation can be done by pressing the **Save** or **Save As...** button. Alternatively, saving a program also saves the active installation. To load a different installation file, use the **Load** button. The **Create New** button resets all of the settings in the Robot Installation to their factory defaults.

CAUTION:
Using the robot with an installation loaded from a USB drive is not recommended. To use an installation stored on a USB drive, first load it and then save it in the local **programs** folder using the **Save As...** button.

13.6 Installation → TCP Configuration

A **Tool Center Point** (TCP) is a point on the robot's tool. Each TCP contains a translation and a rotation relative to the center of the tool output flange.
When programmed to return to a previously stored waypoint, a robot moves the TCP to the position and orientation saved within the waypoint. When programmed for linear motion, the TCP moves linearly.



Position

The X, Y, Z coordinates specify the TCP position. When all values (including orientation) are zero, the TCP coincides with the center point of the tool output flange and adopts the coordinate system depicted on the screen.

Orientation

The RX, RY, RZ coordinate boxes specify the TCP orientation. Similar to the Move Tab, use the Units drop down menu above the RX, RY, RZ boxes to select the orientation coordinates (see 12.2).

13.6.1 Adding, renaming, modifying and removing TCPs

Tap the **New** button to define a new TCP. The created TCP automatically receives a unique name and becomes selectable in the drop-down menu. To rename a TCP, tap the **Rename** button. To remove the selected TCP, tap the **Remove** button. The last TCP cannot be removed.
The translation and rotation of a selected TCP can be modified by entering new values into the fields.

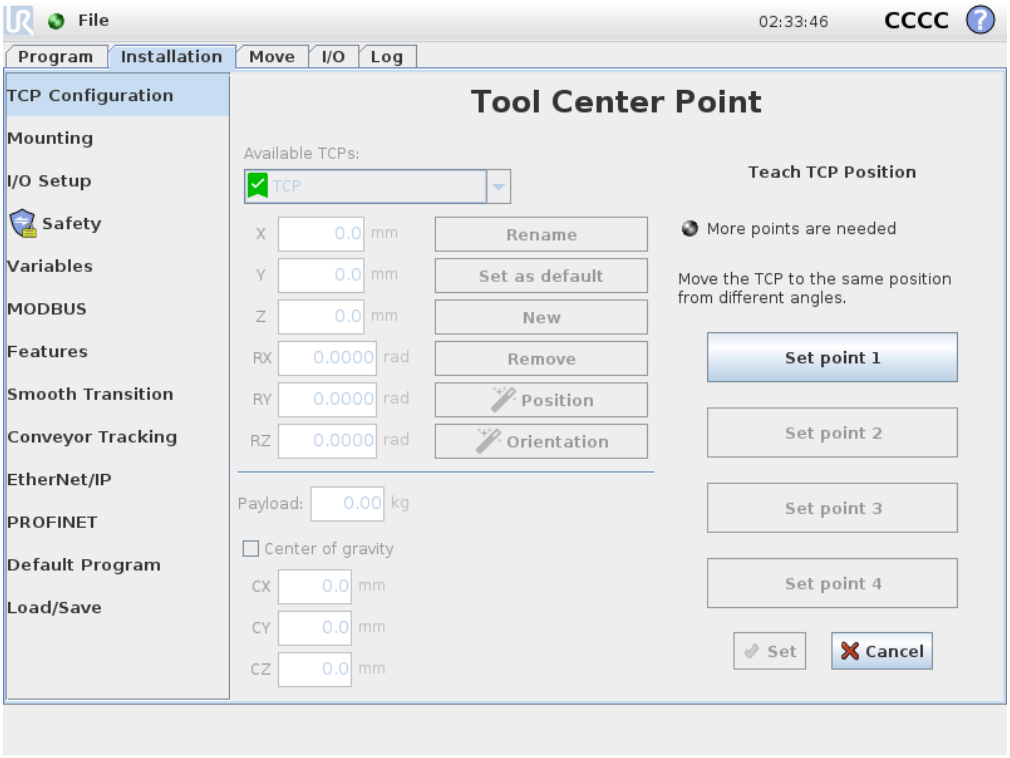
13.6.2 Active TCP

When moving linearly, the robot always uses the active TCP to determine the TCP offset. The active TCP can be changed using a Move command `(see 14.4)` or a Set command.

13.6.3 Default TCP

The Default TCP must be set as the active TCP before running a program. Select the desired TCP and tap **Set as default** to set a TCP as the default. The green icon in the available drop-down menu indicates the default configured TCP.

13.6.4 Teaching TCP position



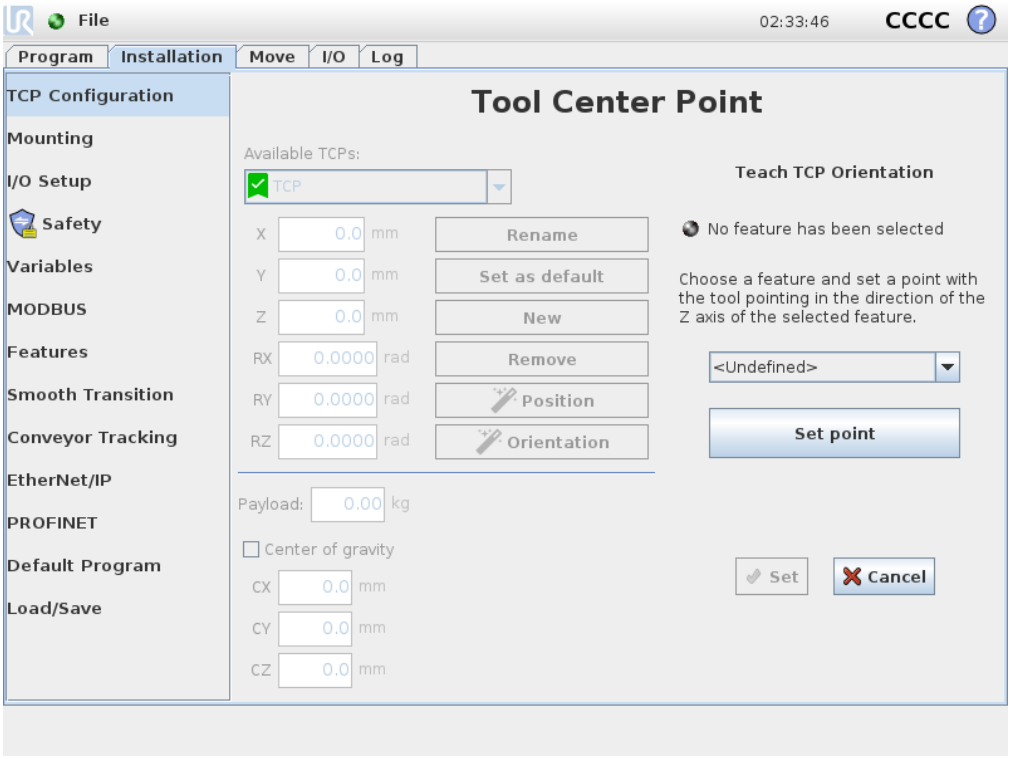
TCP position coordinates can be calculated automatically as follows:

1. Tap the **TCP Position Wizard**.
2. Choose a fixed point in the workspace of the robot.

3. Use the position arrows on the right side of the screen to move the TCP from at least three different angles and to save the corresponding positions of the tool output flange.
4. Use the **Set** button to apply the verified coordinates to the appropriate TCP. The positions must be sufficiently diverse for the calculation to work correctly. If they are not sufficiently diverse, the status LED above the buttons turns red.

Though three positions are sufficient to determine the TCP, a fourth position can be used to further verify the calculation is correct. The quality of each saved point, with respect to the calculated TCP, is indicated using a green, yellow, or red LED on the corresponding button.

13.6.5 Teaching TCP orientation



1. Tap the **TCP Orientation Wizard**.
2. Select a feature from the drop-down list. `(See 13.12)` for additional information on defining new features
3. Tap **Select point** and use **Move tool arrows** to a position where the tool's orientation and the corresponding TCP coincide with the selected features's coordinate system.
4. Verify the calculated TCP orientation and apply it to the selected TCP by tapping **Set**.

13.6.6 Payload

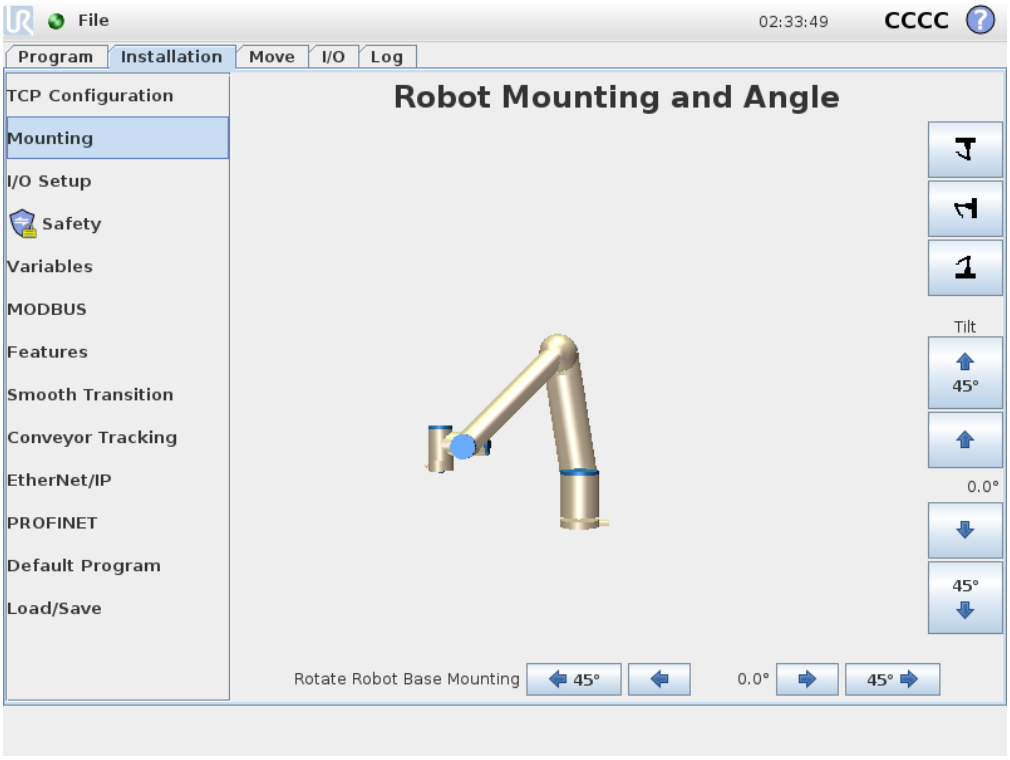
The weight of the robot's tool is specified in the lower part of the screen. To change this setting, simply tap the white text field and enter a new weight. The setting applies to all defined TCPs. For details about the maximum allowed payload, `see the Hardware Installation Manual`.

13.6.7 Center of gravity

The tool's center of gravity is specified using the fields CX, CY and CZ. The settings apply to all defined TCPs. Installations created before version 3.8 support the center of gravity being set to the TCP if they were previously set. If the center of gravity is manually set, in 3.8 or higher, the ability to set the center of gravity for the TCP is permanently removed.

WARNING:
Use the correct installation settings. Save and load the installation files with the program.

13.7 Installation → Mounting



Specifying the mounting of the Robot arm serves two purposes:

1. Making the Robot arm appear correctly on screen.
2. Telling the controller about the direction of gravity.

An advanced dynamics model gives the Robot arm smooth and precise motions, as well as allows the Robot arm to hold itself in **Freedrive Mode**. For this reason, it is important to mount the Robot arm correctly.

WARNING:
Failure to mount the Robot's arm correctly may result in frequent Protective Stops, and/or the Robot arm will move when pressing the **Freedrive** button.

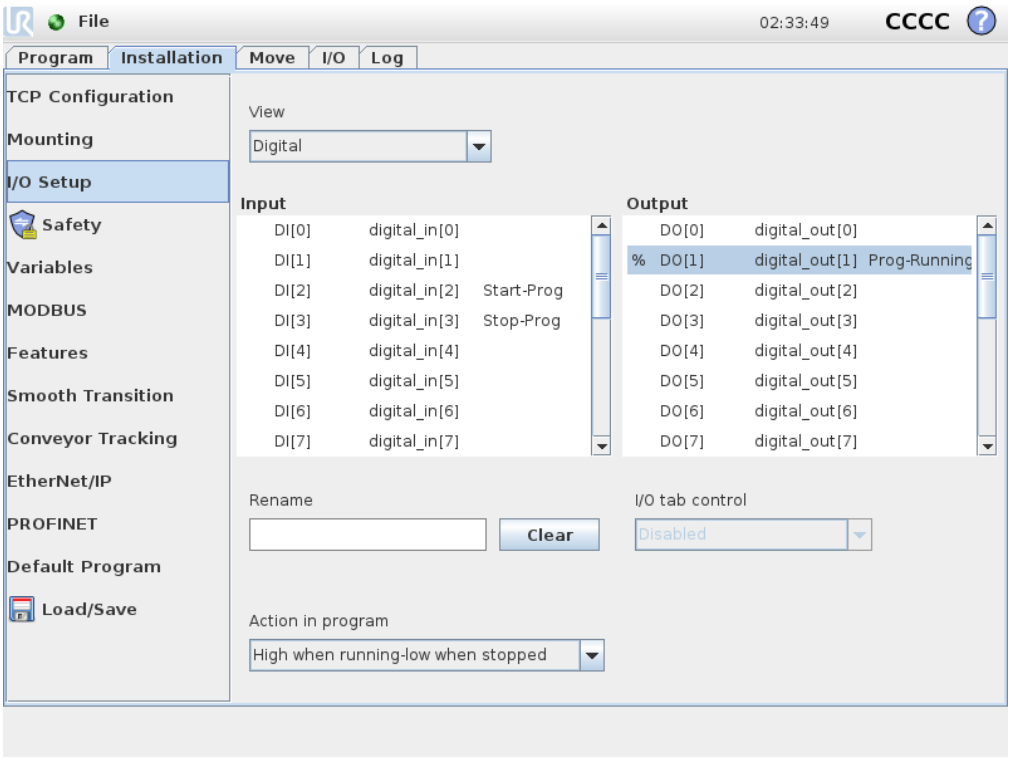
If the Robot arm is mounted on a flat table or floor, no change is needed on this screen. However, if the Robot arm is **ceiling mounted**, **wall mounted**, or **mounted at an angle**, this needs to be adjusted using the buttons.

The buttons on the right side of the screen are for setting the angle of the Robot arm's mounting. The top three right side buttons set the angle to **ceiling** (180°), **wall** (90°), **floor** (0°). The **Tilt** buttons set an arbitrary angle.

The buttons on the lower part of the screen are used to rotate the mounting of the Robot arm to match the actual mounting.

WARNING:
Use the correct installation settings. Save and load the installation files with the program.

13.8 Installation → I/O Setup



On the I/O Setup screen, users can define I/O signals and configure actions with the I/O tab control.

The **Input** and **Output** sections list types of I/O signals such as:

- Digital standard general purpose, configurable and tool
- Analog standard general purpose and tool
- MODBUS
- General purpose registers (boolean, integer and float) The general purpose registers can be accessed by a fieldbus (e.g., Profinet and EtherNet/IP).

13.8.1 I/O Signal Type

To limit the number of signals listed in the **Input** and **Output** sections, use the **View** drop-down menu at the top of the screen to change the displayed content based on signal type.

13.8.2 Assigning User-defined Names

To easily remember what the signals do while working with the robot, users can associate names to Input and Output signals.

1. Select the desired signal
2. Tap the text field in the lower part of the screen to set the name.
3. To reset the name to default, tap **Clear**.

A general purpose register must be given a user-defined name to make it available in the program (i.e., for a **Wait** command or the conditional expression of an **If** command) The **Wait** and **If** commands are described in `(14.10)` and `(14.17)`, respectively. Named general purpose registers can be found in the **Input** or **Output** selector on the **Expression Editor** screen.

13.8.3 I/O Actions and I/O Tab Control

Input and Output Actions: Physical and Fieldbus digital I/Os can be used to trigger actions or react to the status of a program.

Available Input Actions:

- Start: starts or resumes the current program on a rising edge.
- Stop: Stops the current program on a rising edge.
- Pause: Pauses the current program on a rising edge.
- Freedrive: When the input is high, the robot is in freedrive (similar to the freedrive button). The input is ignored if a program is running or other conditions disallow freedrive.

WARNING:
If the robot is stopped while using the Start input action, the robot slowly moves to the first waypoint of the program before executing that program. If the robot is paused while using the Start input action, the robot slowly moves to the position from where it was paused before resuming that program.

Available Output Actions:

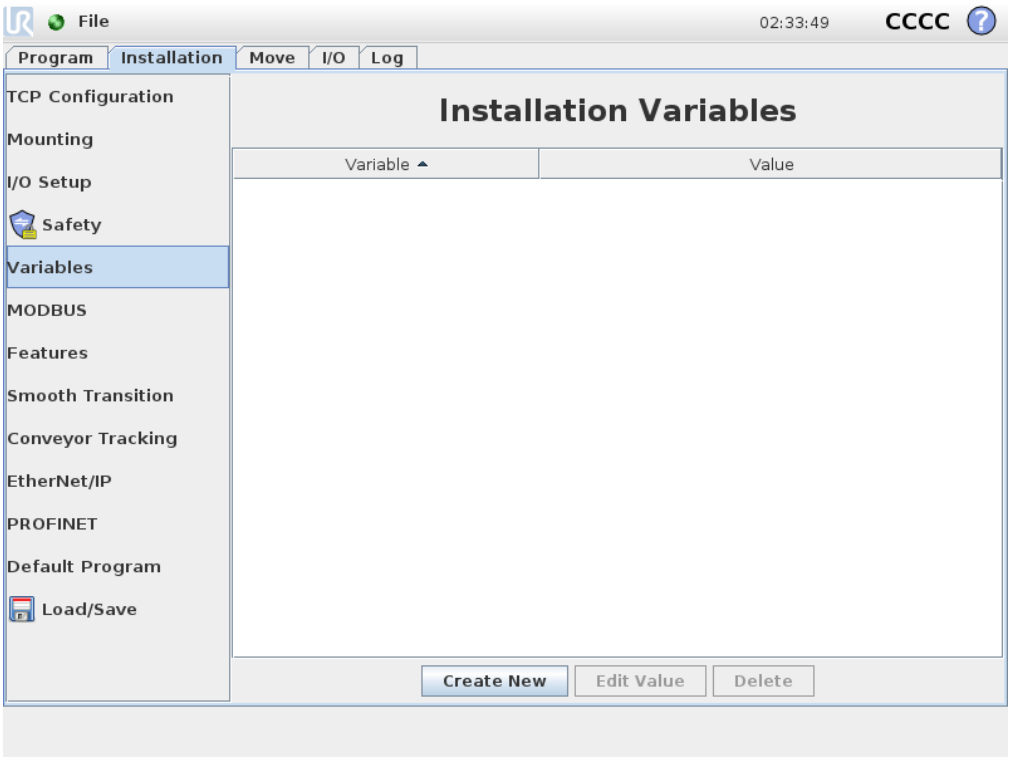
- Low when not running: Output is low when the program state is "stopped" or "paused".
- High when not running: Output is high when the program state is "stopped" or "paused".
- High when running, low when stopped: Output is low when the program state is "stopped" or "paused" and high when it is running.
- Continuous Pulse: Output alternates between high and low for a specified number of seconds, while the program is running. Pause or stop the program to maintain the pulse state.

I/O Tab Control: Specify whether an output is controlled on the I/O tab (by either programmers, or both operators and programmers), or if it is controlled by the robot programs.

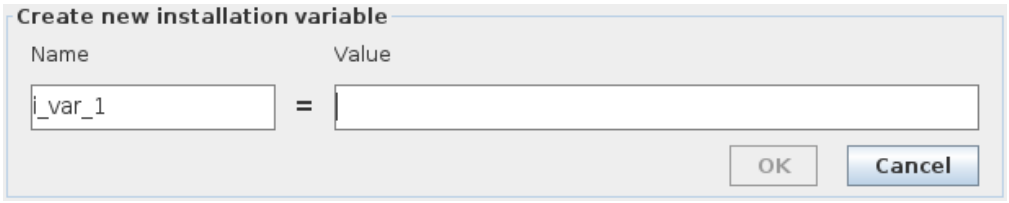
13.9 Installation → Safety

See chapter 10.

13.10 Installation → Variables



Variables created on the Variables screen are called Installation Variables and are used like normal program variables. Installation Variables are distinct because they keep their value even if a program stops and then starts again, and when the Robot arm and/or Control Box is powered down and powered up again. Their names and values are stored with the installation, therefore it is possible to use the same variable in multiple programs.



Pressing **Create New** brings 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 tapped 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**.

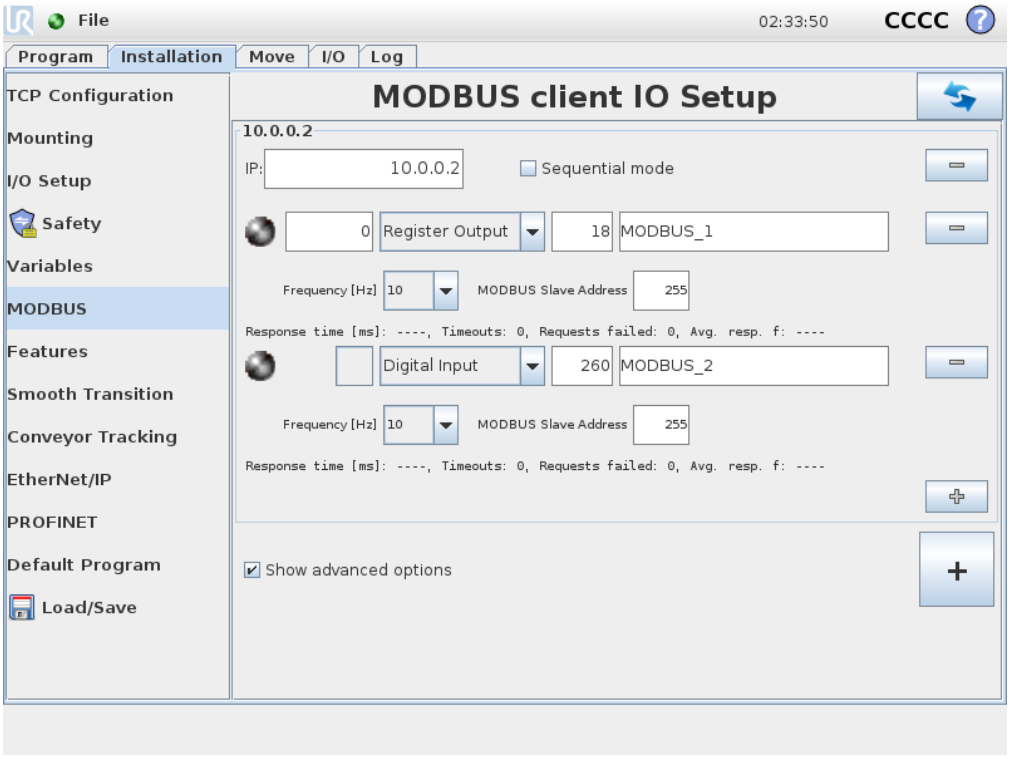
To delete a variable, select it and tap **Delete**.

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

The installation variables and their values are saved automatically every 10 minutes.

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 options to either resolve the issue using the installation variables of the same name instead of the program variable or rsolve the issue by having the conflicting variables renamed automatically.

13.11 Installation → 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 refresh all MODBUS connections. Refreshing disconnects all modbus units, and connects them back again. All statistics are cleared.

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.

Sequential mode

Available only when Show Advanced Options (see 13.11) is selected. Selecting this checkbox forces the modbus client to wait for a response before sending the next request. This mode is required by some fieldbus units. Turning this option on may help when there are multiple signals, and increasing request frequency results in signal disconnects. Note that the actual signal frequency may be lower than requested when multiple signals are defined in sequential mode. Actual signal frequency can be observed in signal statistics (see section 13.11). The signal indicator will turn yellow if the actual signal frequency is less than half of the value selected from the ”Frequency” drop-down list.

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 enterd 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. When the frequency is set to 0, then modbus requests are initiated on demand using a *modbus_get_signal_status*, *modbus_set_output_register*, and *modbus_set_output_signal* script functions.

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.

Reconnect count: Number of times TCP connection was closed, and connected again.

Connection status: TCP connection status.

Response time [ms:] Time between modbus request sent, and response received - this is updated only when communication is active.

Modbus packet errors: Number of received packets that contained errors (i.e. invalid lenght, missing data, TCP socket error).

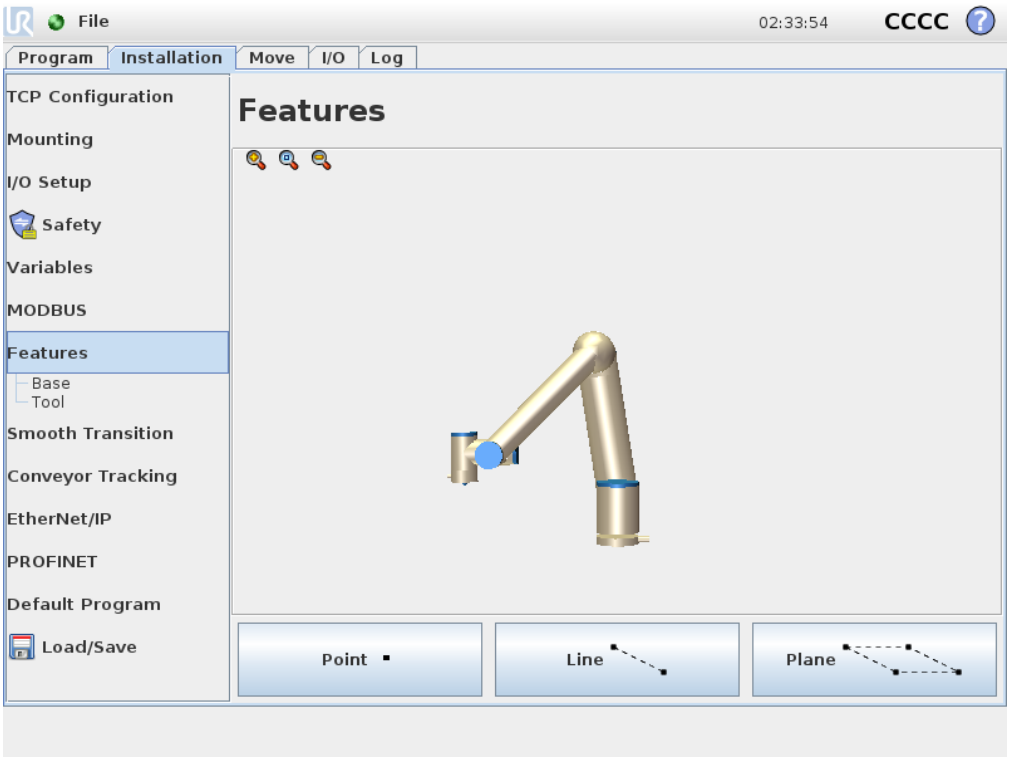
Timeouts: Number of modbus requests that didn't get response.

Requests failed: Number of packets that could not be sent due to invalid socket status.

Actual freq.: The average frequency of client (master) signal status updates. This value is recalculated each time the signal receives a response from the server (or slave).

All counters count up to 65535, and then wrap back to 0.

13.12 Installation → Features



The **Feature**, is a representation of an object that is defined with a name for future reference and a six dimensional pose (position and orientation) relative to the robot base.
Some subparts of a robot program consist of movements executed relative to specific objects other than the base of the Robot arm. These objects could be tables, other machines, workpieces, conveyors, pallets, vision systems, blanks, or boundaries which exist in the surroundings of the Robot arm. Two predefined features always exist for the robot. Each feature has its pose defined by the configuration of the Robot arm itself:

- The Base feature is located with origin in the centre of the robot base `(see figure 13.1)`
- The Tool feature is located with origin in the centre of the current TCP `(see figure 13.2)`

User-defined features are positioned through a method that uses the current pose of the TCP in the work area. This means the users can teach feature locations using Freedrive Mode or ”jogging” to move the robot to the desired pose.

Three different strategies exist (**Point**, **Line** and **Plane**) for defining a feature pose. The best strategy for a given application depends on the type of object being used and the precision requirements. In general a feature based on more input points (**Line** and **Plane**) is be preferred if applicable to the specific object.
To accurately define the direction of a linear conveyor, define two points of a Line feature with as much physical separation as possible. The Point feature can also be used to define a linear conveyor,

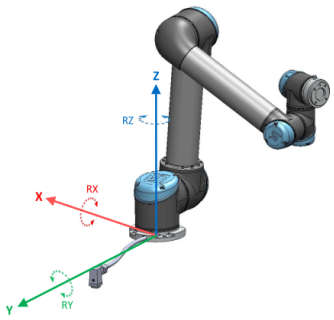


Figure 13.1: Base feature

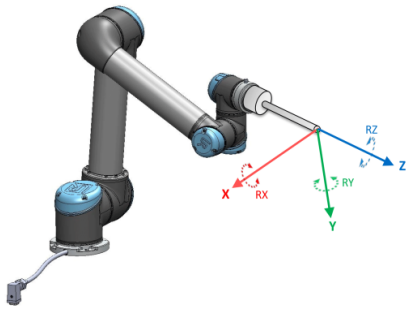


Figure 13.2: Tool (TCP) feature

however, the user must point the TCP in the direction of the conveyor movement.
Using more points to define the pose of a table means that the orientation is based on the positions rather than the orientation of a single TCP. A single TCP orientation is harder to configure with high precision.
To learn about the different methods to define a feature see `(sections: 13.12.2)`, `(13.12.3)` and `(13.12.4)`.

13.12.1 Using a feature

When a feature is defined in the installation, you can refer to it from the robot program to relate robot movements (e.g. **MoveL** and **MoveP** commands) to the feature `(see section 14.4)`. This allows for easy adaptation of a robot program (e.g., when there are multiple robot stations, when an object is moved during program runtime, or when an object is permanently moved in the scene). By adjusting the feature of an object, all program movements relative to the object is moved accordingly. For further examples, see `(sections 13.12.5)` and `(13.12.6)`. Features configured as joggable are also useful tools when manually moving the robot in the Move Tab `(section 13.1)` or the **Pose Editor** screen `(see 12.2)`. When a feature is chosen as a reference, the Move Tool buttons for translation and rotation operate in the selected feature space `(see 13.1.2)` and `(13.1.3)`, reading of the TCP coordinates. For example, if a table is defined as a feature and is chosen as a reference in the Move Tab, the translation arrows (i.e., up/down, left/right, forward/backward) move the robot in these directions relative to the table. Additionally, the TCP coordinates will be in the frame of the table.

Rename

This button renames a feature.

Delete

This button deletes a selected feature and any of its sub-features.

Show Axes

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

Changing the point

Use the **Change this point** button to set or change the selected feature. The **Move** tab `(section 13.1)` appears and a new feature position can be set.

Joggable

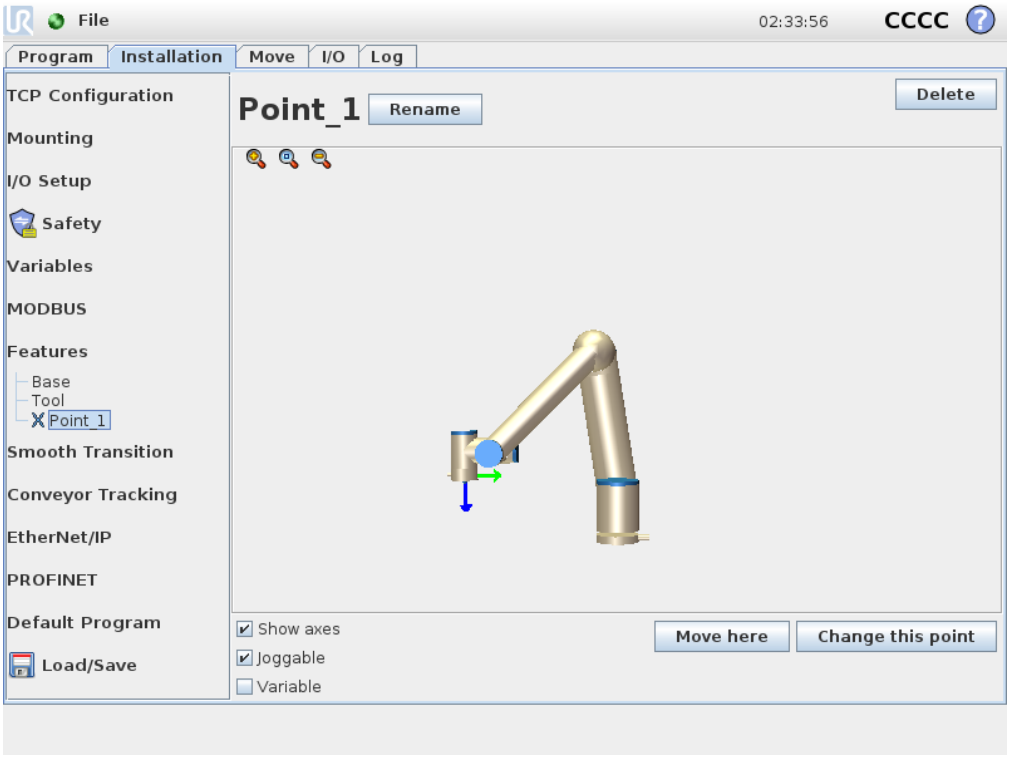
Choose whether the selected feature should be joggable. This determines whether the feature will appear in the feature menu on the Move screen.

Using Move robot here

Push the **Move robot here** button to move the Robot arm towards the selected feature. At the end of this movement, the coordinate systems of the feature and the TCP will coincide.

13.12.2 New Point

Push the **Point**button to add a point feature to the installation. The point feature defines a safety boundary or a global home configuration of the Robot arm. The point feature pose is defined as the position and orientation of the TCP.



13.12.3 New Line

Push the **Line** button to add a line feature to the installation. The line feature defines lines that the robot needs to follow. (e.g., when using conveyor tracking). A line l is defined as an axis between two point features $p1$ and $p2$ as shown in figure 13.3.

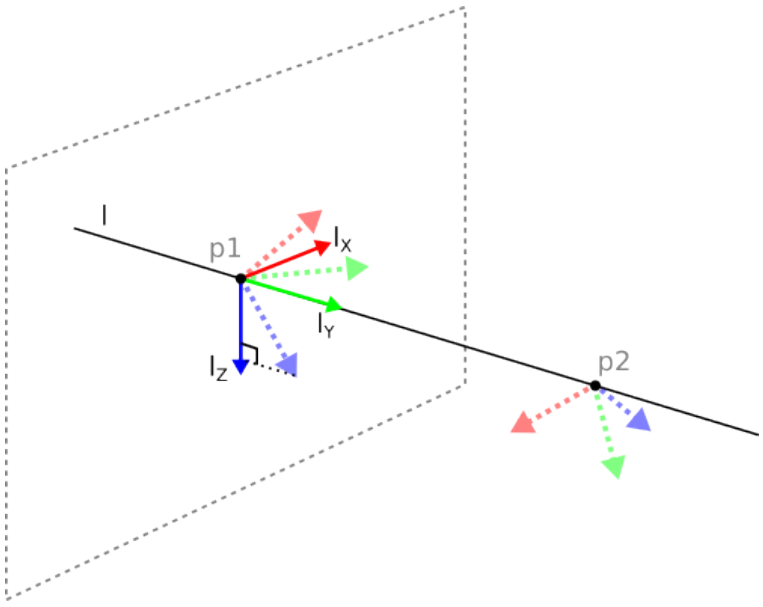
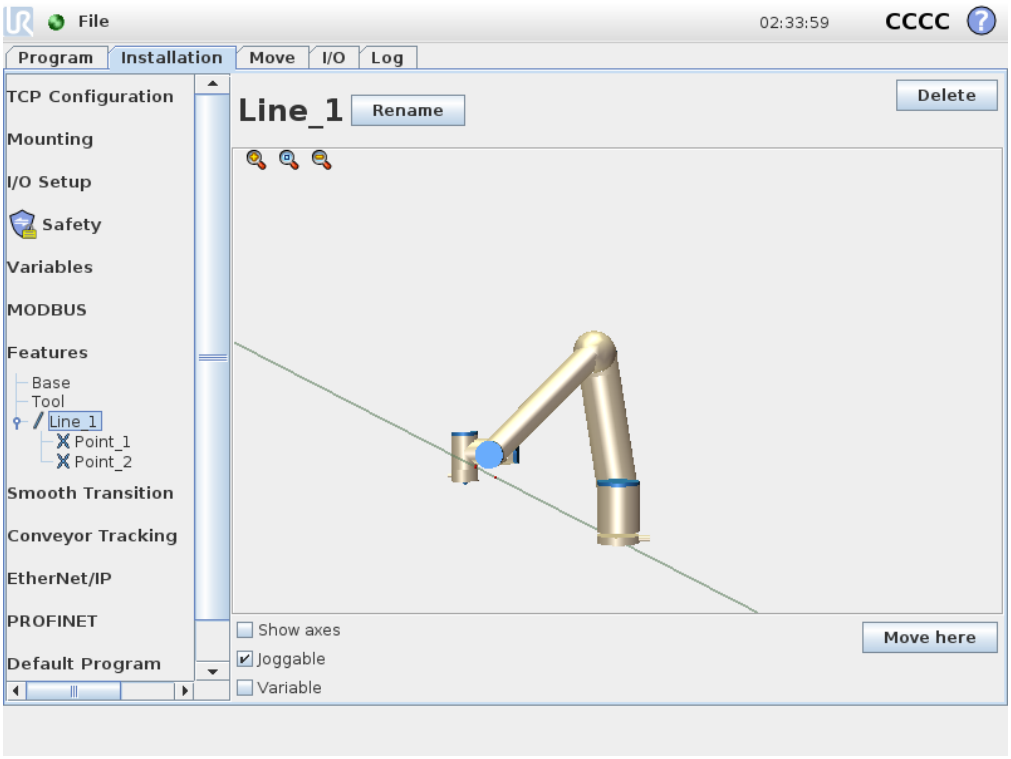


Figure 13.3: Definition of the line feature

In figure 13.3 the axis directed from the first point towards the second point, constitutes the y-axis of the line coordinate system. The z-axis is defined by the projection of the z-axis of *p1* onto the plane perpendicular to the line. The position of the line coordinate system is the same as the position of *p1*.



13.12.4 Plane Feature

Select the plane feature when you need a frame with high precision: e.g., when working with a vision system or doing movements relative to a table.

Adding a plane

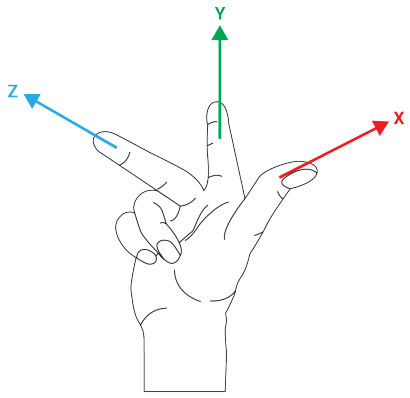
1. In Installation, select **Features**.
2. Under Features select **Plane**.

Teaching a plane

When you press the plane button to create a new plane, the on-screen guide assists you creating a plane.

1. Select Origo
2. Move robot to define the direction of the positive x-axis of the plane
3. Move robot to define the direction of the positive y-axis of the plane

The plane is defined using the right hand rule so the z- axis is the cross product of the x-axis and the y-axis, as illustrated below.



NOTE:
You can re-teach the plane in the opposite direction of the x-axis, if you want that plane to be normal in the opposite direction.

Modify an existing plane by selecting Plane and pressing Modify Plane. You will then use the same guide as for teaching a new plane.

13.12.5 Example: Manually Updating a Feature to Adjust a Program

Consider an application where multiple parts of a robot program is relative to a table. Figure 13.4 illustrates the movement from waypoints `wp1` through `wp4`.

```
Robot Program
    MoveJ
        S1
    MoveL # Feature: P1_var
        wp1
        wp2
        wp3
        wp4
```

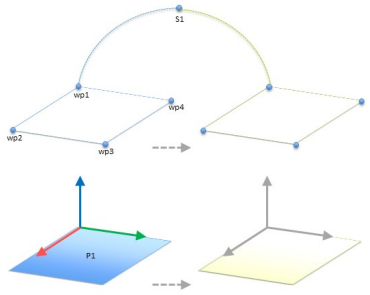


Figure 13.4: Simple program with four waypoints relative to a feature plane manually updated by changing the feature

The application requires the program to be reused for multiple robot installations where the position of the table varies slightly. The movement relative to the table is identical. By defining the table position as a feature *P1* in the installation, the program with a *MoveL* command configured relative to the plane can be easily applied on additional robots by just updating the installation with the actual position of the table.

The concept applies to a number of Features in an application to achieve a flexible program can solve the same task on many robots even though if other places in the work space varies between installations.

13.12.6 Example: Dynamically Updating a Feature Pose

Consider a similar application where the robot must move in a specific pattern on top of a table to solve a particular task (see 13.5).

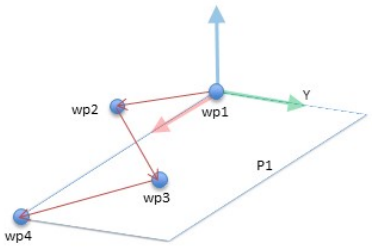


Figure 13.5: A *MoveL* command with four waypoints relative to a plane feature

The movement relative to *P1* is repeated a number of times, each time by an offset *o*. In this example the offset is set to 10 cm in the Y-direction (see figure 13.6, offsets *O1* and *O2*). This is achieved using *pose_add()* or *pose_trans()* script functions to manipulate the variable. It is possible to switch to a different feature while the program is running instead of adding an offset. This is shown in the example below (see figure 13.7) where the reference feature for the *MoveL* command *P1_var* can switch between two planes *P1* and *P2*.

```
Robot Program
    MoveJ
        wp1
    y = 0.01
    o = p[0,y,0,0,0,0]
    P1_var = pose_trans(P1_var, o)
    MoveL # Feature: P1_var
        wp1
        wp2
        wp3
        wp4
```

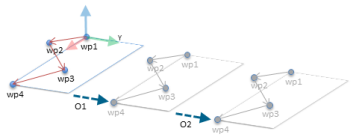


Figure 13.6: Applying an offset to the plane feature

```
Robot Program
    MoveJ
        S1
    if (digital_input[0]) then
        P1_var = P1
    else
        P1_var = P2
    MoveL # Feature: P1_var
        wp1
        wp2
        wp3
        wp4
```

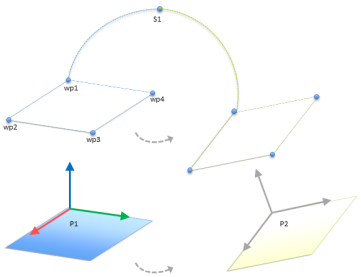


Figure 13.7: Switching from one plane feature to another

13.13 Conveyor Tracking Setup

The Conveyor Tracking Setup allows the movement of up to two separate conveyors to be configured. The Conveyor Tracking Setup provides options for configuring the robot to work with absolute or incremental encoders, as well as linear or circular conveyors.

Conveyor Parameters

Incremental: encoders can be connected to Digital Inputs 0 to 3. Decoding of digital signals runs at 40kHz. Using a **Quadrature** encoder (requiring two inputs), the robot can determine the speed and direction of the conveyor. If the direction of the conveyor is constant, a single input can be used to detect *Rising*, *Falling*, or *Rise and Fall* edges which determine conveyor speed.

Absolute: encoders can be connected through a MODBUS signal. This requires a Digital MODBUS Output register preconfigured in `(section 13.11)`.

Tracking Parameters

Linear Conveyors: When a linear conveyor is selected, a line feature must be configured in the **Features** part of the installation to determine the direction of the conveyor. Ensure accuracy by placing the line feature parallel to the direction of the conveyor, with a large distance between the two points that define the line feature. Configure the line feature by placing the tool firmly

against the side of the conveyor when teaching the two points. If the line feature's direction is opposite to the conveyor's movement, use the **Reverse direction** button.

The **Ticks per meter** field displays the number of ticks the encoder generates when the conveyor moves one meter.

Circular Conveyors: When tracking a circular conveyor, the conveyor center point must be defined.

1. Define the center point in the **Features** part of the installation. The value of **Ticks per revolution** must be the number of ticks the encoder generates when the conveyor rotates one full revolution.
2. Select the **Rotate tool with conveyor** checkbox for the tool orientation to track the conveyor rotation.

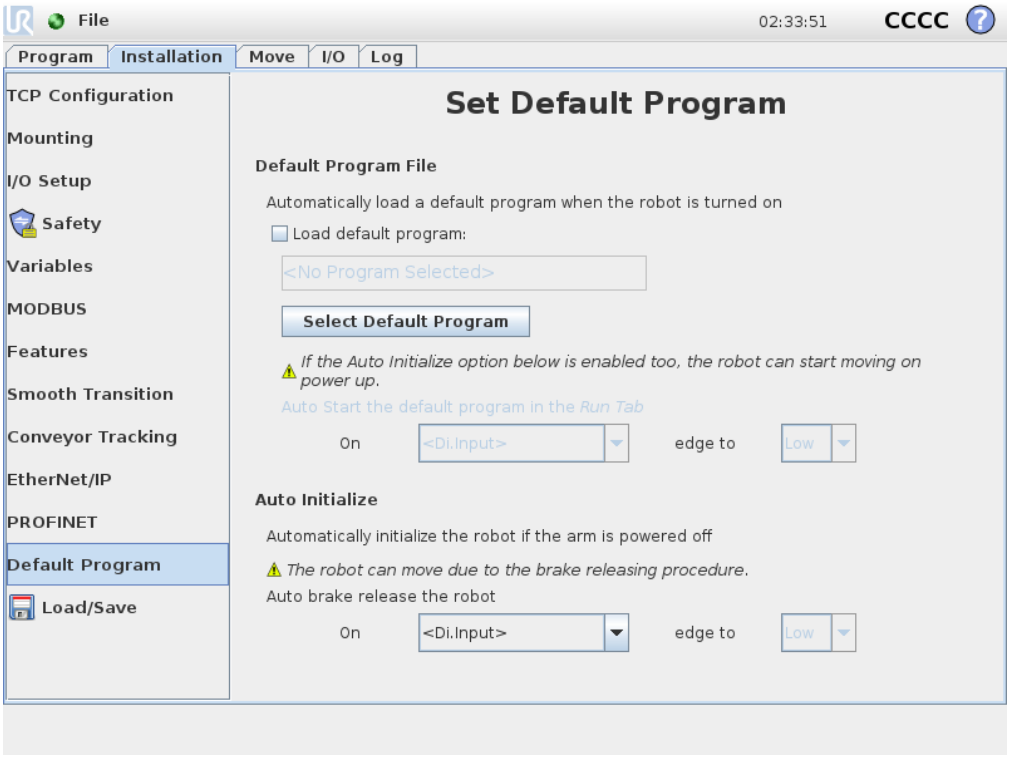
13.14 Smooth Transition Between Safety Modes

When switching between safety modes during events (i.e., Reduced Mode Input, Reduced Mode Trigger Planes, Safeguard Stop, and Three-Position Enabling Device), the Robot Arm aims to use 0.4s to create a "soft" transition. Existing applications have unchanged behavior which corresponds to the "hard" setting. New installation files default to the "soft" setting.

13.14.1 Adjusting Acceleration/Deceleration Settings

- Press the Installation tab.
- In the Side Menu on the left, select Smooth Transition.
- Select Hard to have a higher acceleration/deceleration or select Soft for the smoother default transition setting.

13.15 Installation → Default Program



The Startup screen contains settings for automatically loading and starting a default program, and for auto-initializing the Robot arm during power up.

WARNING:

1. When autoload, auto start and auto initialize are enabled, the robot runs the program as soon as the Control Box is powered up as long as the input signal matches the selected signal level. For example, the edge transition to the selected signal level will not be required in this case.
2. Use caution when the signal level is set to LOW. Input signals are low by default, leading the program to automatically run without being triggered by an external signal.

Loading a Default Program

A default program is loaded after the Control Box is powered up. Furthermore, the default program is auto loaded when the **Run Program** screen `(see 11.4)` is entered and no program is loaded.

Starting a Default Program

The default program is auto started in the **Run Program** screen. When the default program is loaded and the specified external input signal edge transition is detected, the program is started automatically.

On Startup, the current input signal level is undefined. Choosing a transition that matches the signal level on startup starts the program immediately. Furthermore, leaving the **Run Program** screen or tapping the Stop button in the Dashboard disables the auto start feature until the Run button is pressed again.

Auto Initialization

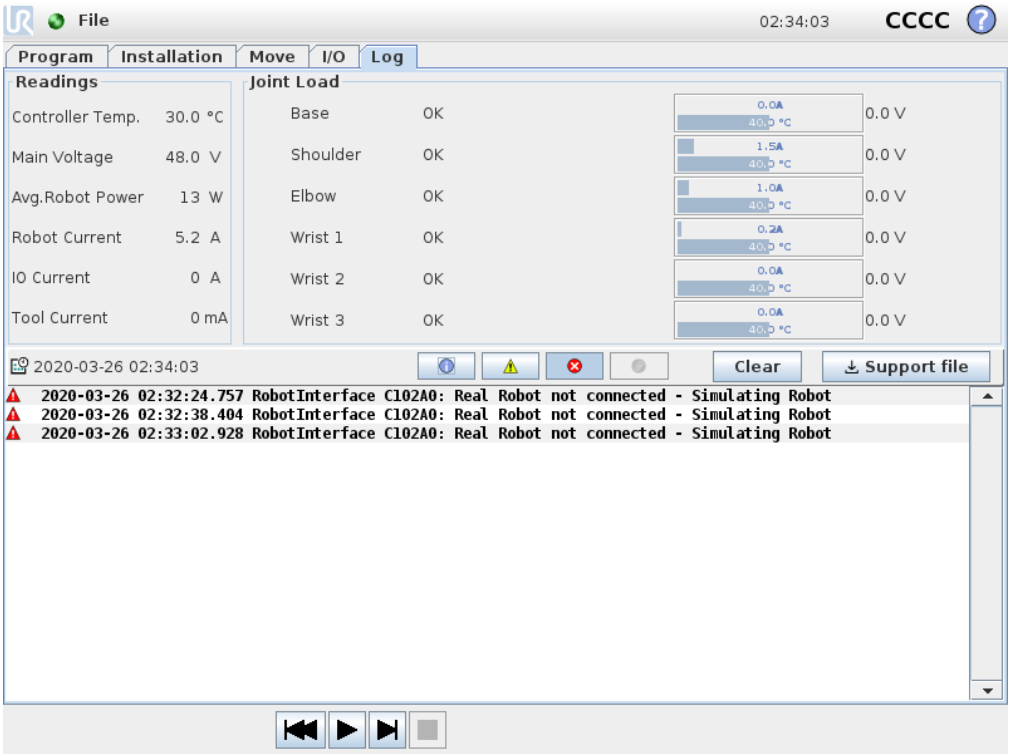
The Robot arm is automatically initialized. On the specified external input signal edge transition, the Robot arm is completely initialized, regardless of the visible screen.

Brake Release is the final initialization stage. During Brake Release, the Robot arm makes a minor movement and a clicking noise. Furthermore, the brakes cannot be automatically released if the configured mounting does not match the mounting detected (based on sensor data). In this case, the robot must be initialized manually on the initialization screen (see 11.5).

On Startup, the current input signal level is undefined. Choosing a transition that matches the signal level on startup initializes the Robot arm immediately.

The auto initialization feature works when only the Robot arm is powered off.

13.16 Log Tab



Robot Health The top half of the screen displays the "health" of the Robot Arm and Control Box. The left side of the screen shows information related to the Control Box, while the right side of the screen displays robot joint information. Each joint dsiplays the temperature of the motor and electronics, the load of the joint, and the voltage.

Robot Log Messages are displayed on the bottom half of the screen. The first column categorizes the severity of the log entry. The second column shows the messages' time of arrival. The next column shows the sender of the message. The last column shows the message itself. Messages can be filtered by selecting the toggle buttons which correspond to the severity of the log entry. The figure above shows errors will be displayed while information and warning messages will be filtered. Some log messages are designed to provide more information that is accessible by selecting the log entry.

13.16.1 Saving Error Reports

When an error occurs in PolyScope, a log of the error is generated. In the Log Tab, you can track and/or export generated reports to a USB drive (see 13.16). The following list of errors can be tracked and exported:

- Fault
- Internal PolyScope exceptions
- Protective Stop
- Unhandled exception in URCap
- Violation

The exported report contains: a user program, a history log, an installation and a list of running services.

Error Report A detailed status report is available when a paper clip icon appears on the log line.

- Select a log line and tap the Save Report button to save the report to a USB drive.
- The report can be saved while a program is running.

NOTE:
The oldest report is deleted when a new one is generated. Only the five most recent reports are stored.

13.16.2 Technical Support File

The report file contains information that is helpful to diagnose and reproduce issues. The file contains records of previous robot failures, as well as current robot configurations, programs and installations. The report file can be saved to external USB drive. On the Log screen, tap **Support file** and follow the on-screen instructions to access the function.

NOTE:
The export process can take up to 10 minutes depending on USB drive speed and the size of files collected from robot file system. The report is saved as a regular zip file, that is not password protected, and can be edited before sending to technical support.

13.17 Load Screen

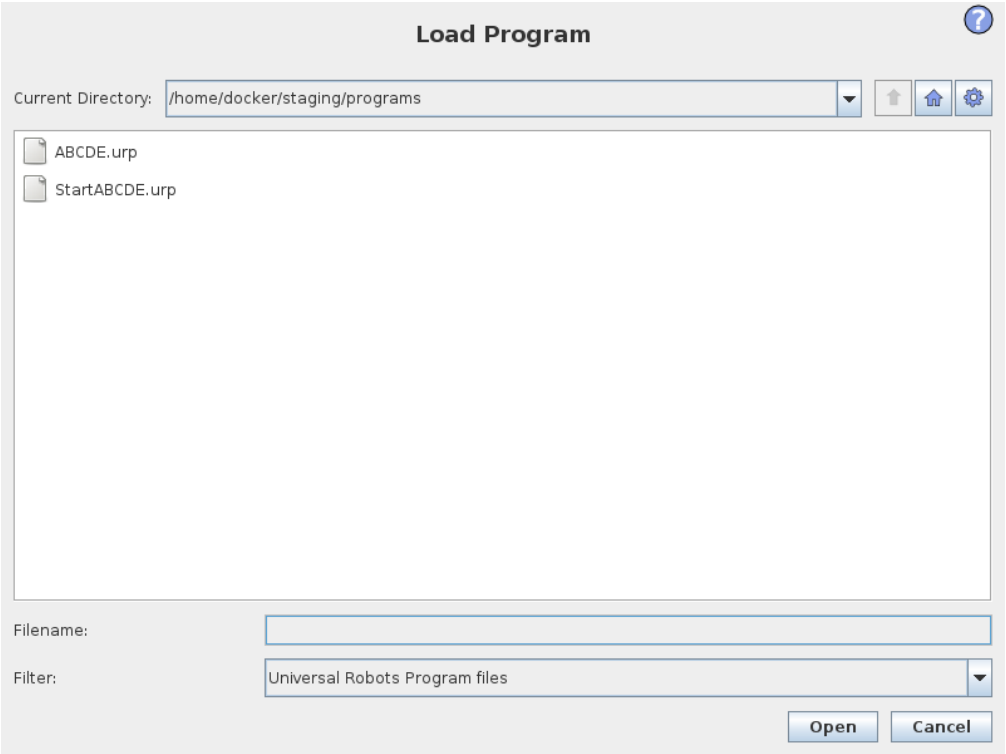
On this screen you choose which program to load. There are two versions of this screen: one that is to be used when you just want to load a program and execute it, and one that is used when you want to actually edit a program.

NOTE:
Running a program from a USB drive is not recommended. To run a program stored on a USB drive, first load it and then save it in the local `programs` folder using the **Save As...** option in the **File** menu.

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 (see 13.17).

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.

Directories are selected by a long press of approximately 0.5 s. Descending into a folder and presenting its content is done by single clicking it.

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.

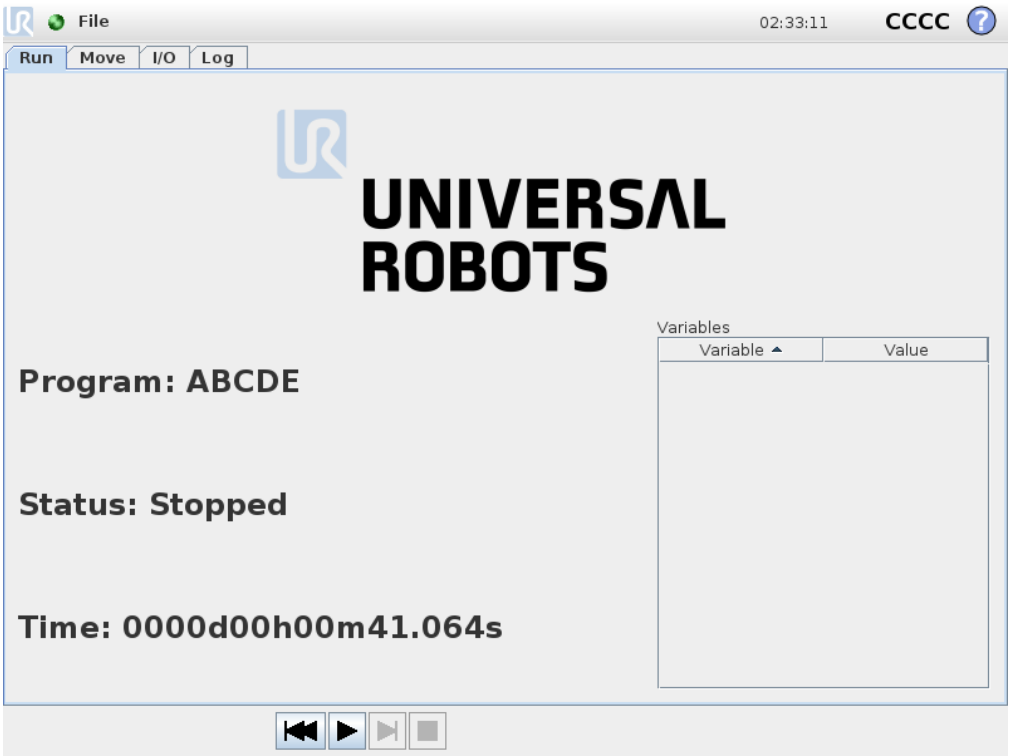
Open button Clicking on the Open button, will open the currently selected file and return to the previous screen.

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.

13.18 Run Tab

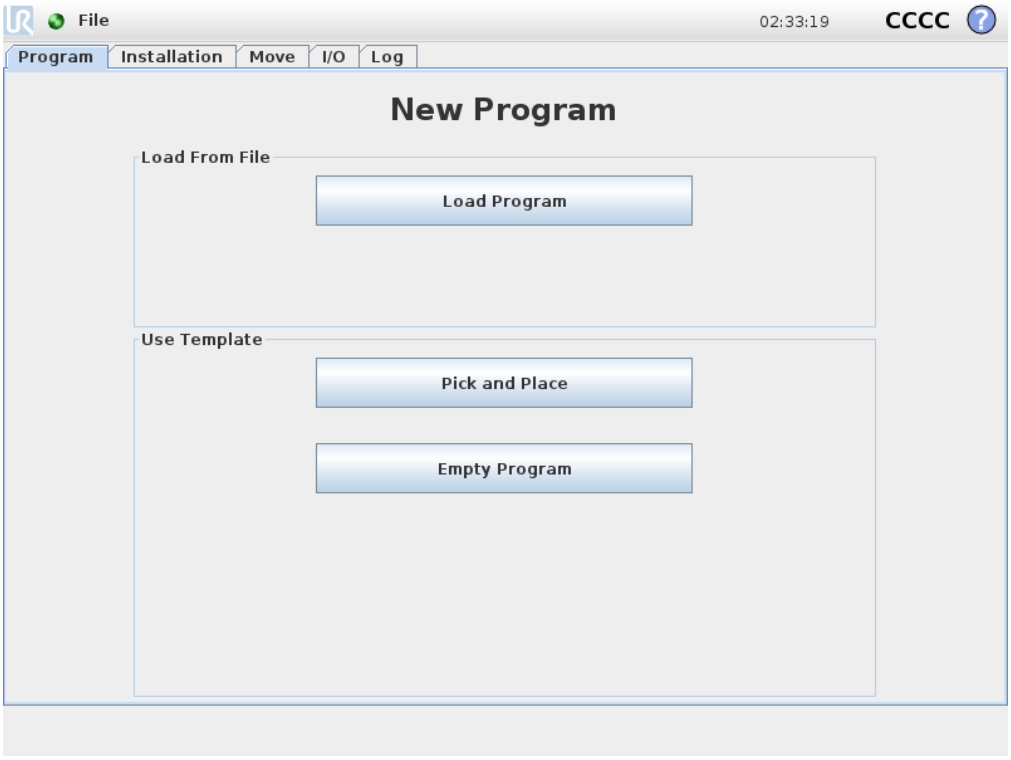


This tab provides a very simple way of operating the robot arm and control box, with as few buttons and options as possible. This can be usefully combined with password protecting the programming part of PolyScope (see 15.3), to make the robot into a tool that can run exclusively pre-written programs.

Furthermore, in this tab a default program can be automatically loaded and started based on an external input signal edge transition (see 13.15). The combination of auto loading and starting of a default program and auto initialization on power up can, for instance, be used to integrate the robot arm into other machinery.

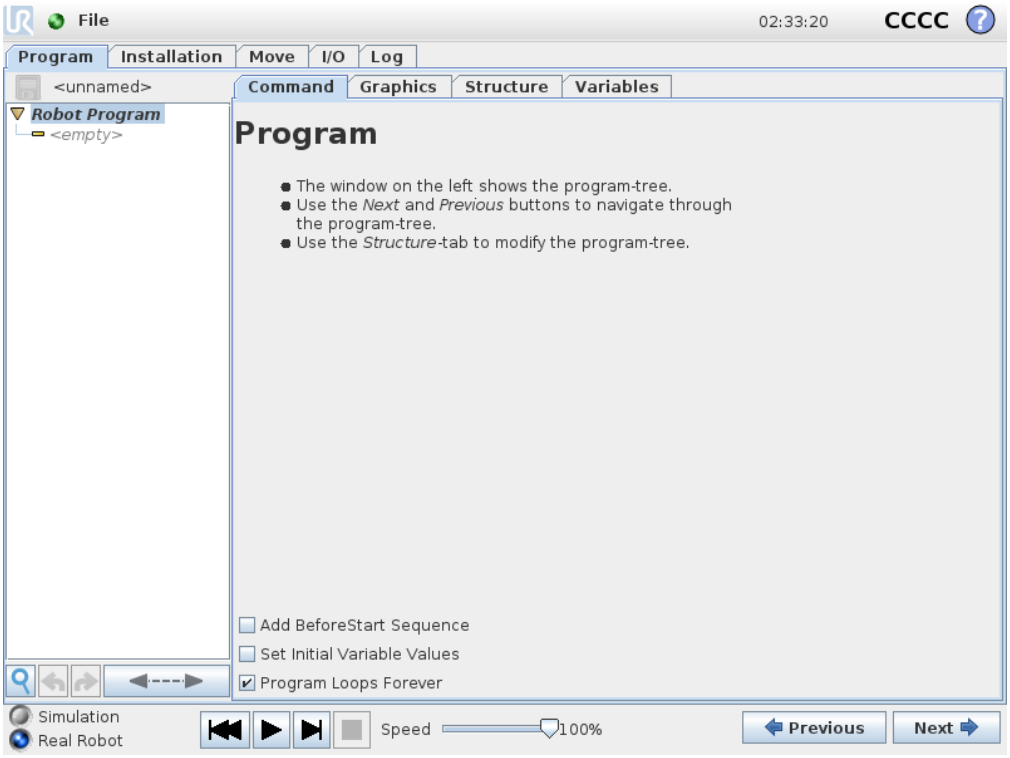
14 Programming

14.1 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.

14.2 Program Tab



The program tab shows the current program being edited.

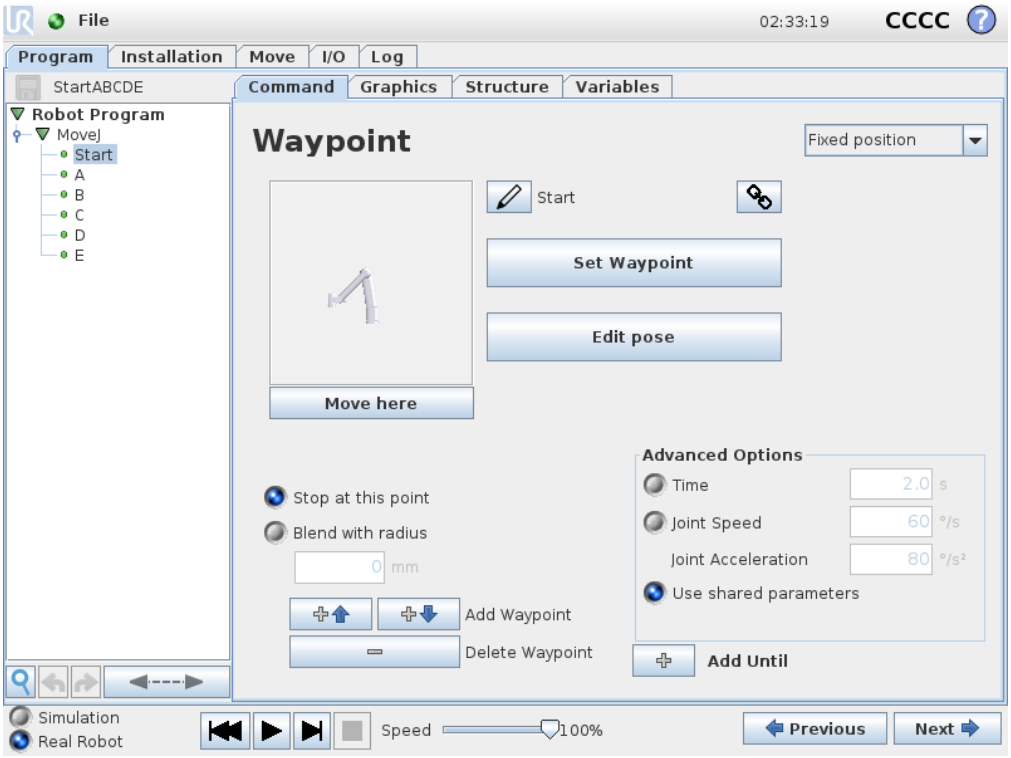
14.2.1 Program Tree

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

In the Program Tree, the command that is currently being executed is highlighted as described in 14.2.2.

14.2.2 Program Execution Indication



The **Program Tree** contains visual cues informing about the command currently being executed by the robot controller. A small ➧ indicator icon is displayed to the left of the command icon, and the name of the executing command and any commands of which this command is a sub-command (typically identified by the ▶/▼ command icons) are highlighted with blue. This aids the user in locating the executing command in the tree. For example, if the robot arm is moving towards a waypoint, the corresponding waypoint sub-command is marked with the ➧ icon and its name together with the name of the Move command `(see 14.4)` to which it belongs to are shown in blue. If the program is paused, the program execution indicator icon marks the last command that was in the process of being executed. Clicking the button with the ⇨ icon below the Program Tree jumps to the current executing or the last executed command in the tree. If a command is clicked while a program is running, the `Command` tab will keep displaying the information related to the selected command. Pressing the ⇨ button will make the `Command` tab continuously show information about the currently executing commands again.

14.2.3 Search Button

Tap the 🔍 to search in the Program Tree. When clicked a search text can be entered and program nodes that match will be highlighted in yellow. Additionally, navigation buttons are made available to navigate through the matches. Press the ✕ icon to exit search. Note: The Program Tree must be expanded to access the additional navigation buttons.

14.2.4 Undo/Redo Buttons

The buttons with icons and in the toolbar at the base of the Program Tree serve to undo and redo changes made in the Program Tree and in the commands it contains.

14.2.5 Program Dashboard

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* allows you to adjust the program speed at any time, which directly affects the speed at which the robot arm moves. Additionally, the *speed slider* shows in real time the relative speed at which the robot arm moves taking into account the safety settings. The indicated percentage is the maximum achievable speed for the running program without violating the safety limits.

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 arm 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 arm will do.

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. Only use the `Step` button when it is absolutely necessary. Make sure to stay outside the robot workspace when the `Step` button is pressed.
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 arm.
4. When a emergency stop or protective stop has occured, the robot program will stop. It can be resumed as long as no joint has moved more than 10°. When pressing play, the robot will move slowly back onto the trajectory, and continue program execution.

While the program is being written, the resulting motion of the robot arm is illustrated using a 3D drawing on the `Graphics` tab, described in 14.30.

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.

14.3 Variables

A robot program can make use of variables to store and update various values during runtime. Two kinds of variables are available:

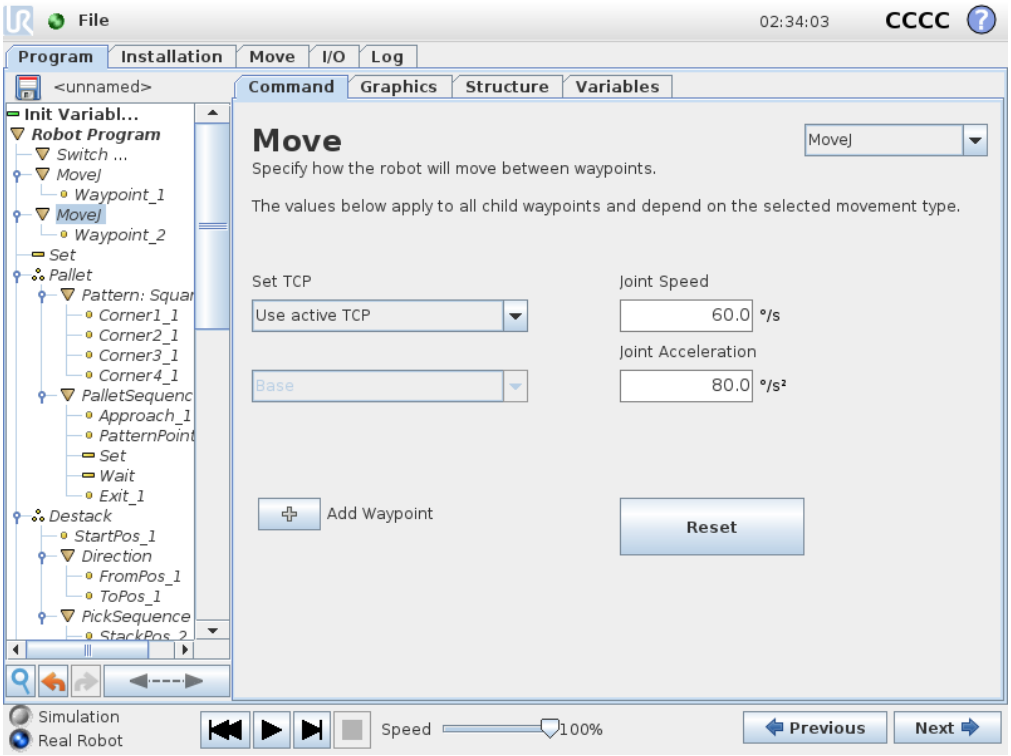
Installation variables: These can be used by multiple programs and their names and values are persisted together with the robot installation (see 13.10). Installation variables keep their value after the robot and control box has been rebooted.

Regular program variables: These are available to the running program only and their values are lost as soon as the program is stopped.

Variable types:

bool	A boolean variable whose value is either `True` or `False`.
int	A whole number in the range from −2147483648 to 2147483647 (32 bit).
float	A floating point number (decimal) (32 bit).
string	A sequence of characters.
pose	A vector describing the location and orientation in Cartesian space. It is a combination of a position vector (x, y, z) and a rotation vector (rx, ry, rz) representing the orientation, written `p[x, y, z, rx, ry, rz]`.
list	A sequence of variables.

14.4 Command: Move

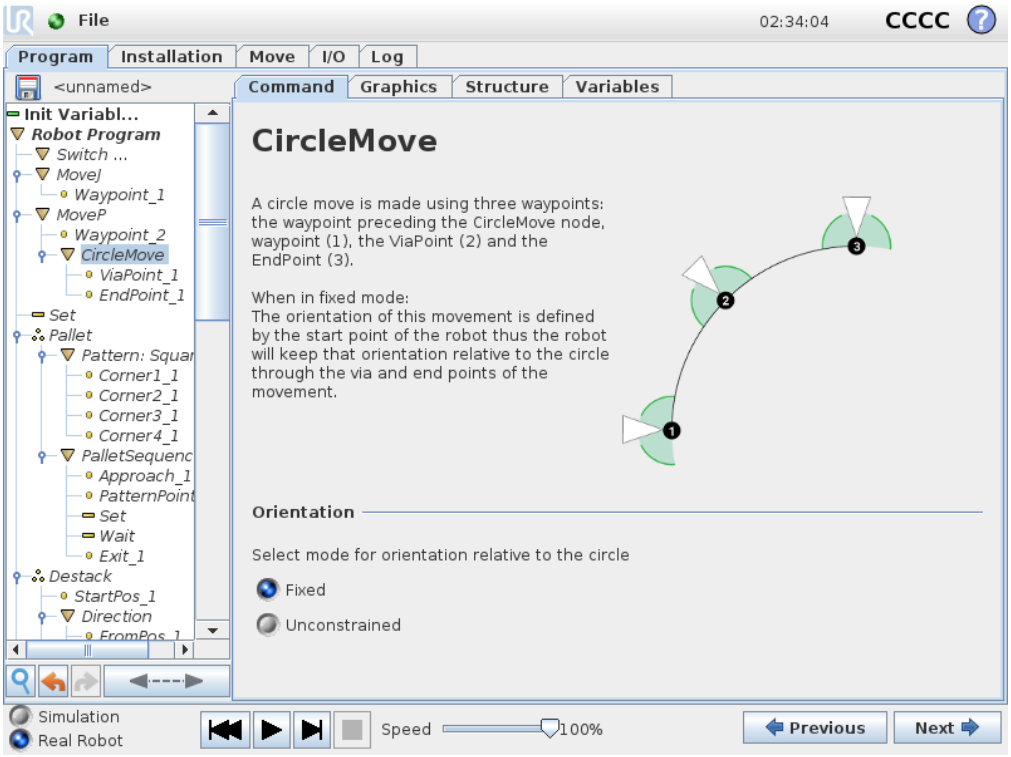


The **Move** command controls the robot motion through the underlying waypoints. Waypoints have to be under a Move command. The Move command defines the acceleration and the speed at which the robot arm will move between those waypoints.

Movement Types

You can select one of three types of movements: **MoveJ**, **MoveL** and **MoveP**. Each movement type is explained below.

- **moveJ** makes movements that are calculated in the robot arm **joint space**. 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, specified in deg/s and deg/s^2, respectively. If it is desired to have the robot arm move fast between waypoints, disregarding the path of the tool between those waypoints, this movement type is the preferable choice.
- **moveL** moves the Tool Center Point (TCP) 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.
- **moveP** moves 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 by 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 arm is moving through the waypoints with constant speed, the robot control box cannot wait for either an I/O operation or an operator action. Doing so might stop the robot arm's motion, or cause a protective stop.
- **Circle move** can be added to a **moveP** to make a circular movement. The robot starts the movement from its current position or start point, moves through a **ViaPoint** specified on the circular arc, and an **EndPoint** that completes the circular movement.
  A mode is used to calculate tool orientation, through the circular arc. The mode can be:
  - Fixed: only the start point is used to define tool orientation
  - Unconstrained: the start point transforms to the **EndPoint** to define tool orientation



Shared parameters

The shared parameters in the bottom right corner of the Move screen apply to the movement from the previous position of the robot arm 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.

TCP selection

The TCP used for the waypoints under this Move command can be selected from the drop-down menu. It is possible to select from user defined TCPs from the installation, the active TCP or simply using the tool flange. If a user defined TCP or the active TCP is selected, the motion under this Move command will adjusted be with respect to this. If **Use Tool Flange** is selected, no TCP is used and the motion under this Move command will be with respect to the tool flange (i.e. no adjustments to the motion).

If the active TCP for this motion is determined during runtime of the program, it needs to be set dynamically using the Set command (see 14.11) or by using script commands. For further information about configuring named TCPs (see 13.6).

Feature selection

The feature spaces the waypoints under the Move command, that should be represented when specifying these waypoints (see section 13.12). 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:

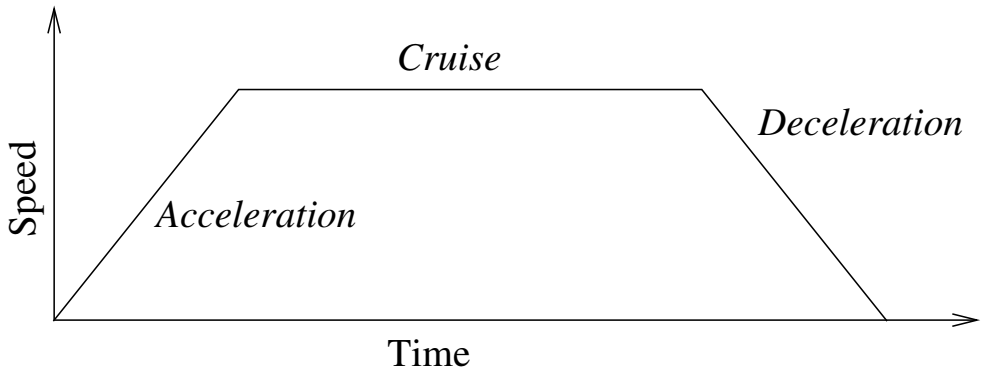


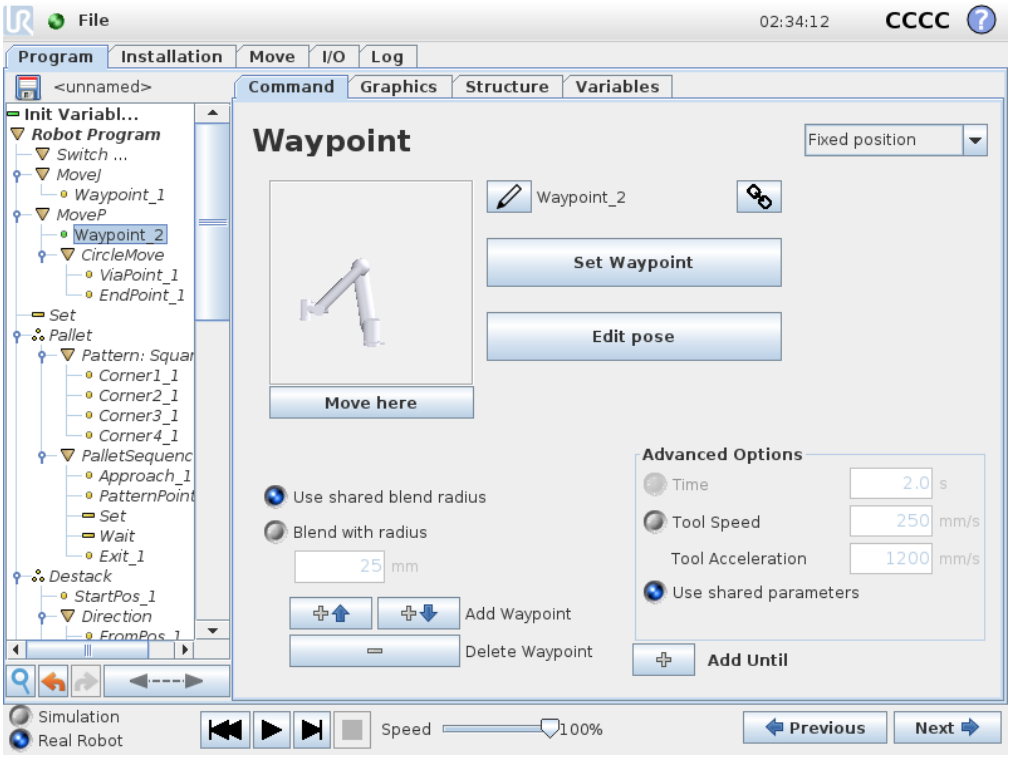
Figure 14.1: Speed profile for a motion. The curve is divided into three segments: *acceleration*, *cruise* and *deceleration*. The level of the *cruise* phase is given by the speed setting of the motion, while the steepness of the *acceleration* and *deceleration* phases is given by the acceleration parameter.

Relative waypoints: The selected feature has no effect on relative waypoints. The relative movement is always performed with respect to orientation of the **Base**.

Variable waypoints: When the robot arm moves to a variable waypoint, the Tool Center Point (TCP) is calculated as the coordinates of the variable in the space of the selected feature. Therefore, the robot arm movement for a variable waypoint changes if another feature is selected.

Variable feature: If any of the features in the currently loaded installation are selected as variable, these corresponding variables are also selectable in the feature selection menu. If a feature variable (named with the name of the feature suffixed by "_var") is selected, robot arm movements (except to **Relative waypoints**) are relative to 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 as configured in the installation. If this value is modified, then the movements of the robot change.

14.5 Command: Fixed Waypoint



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

Setting the waypoint

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

Waypoint names

Waypoints automatically get a unique name. The name can be changed by the user. By selecting the link icon, waypoints are linked and share position information. Other waypoint information such as blend radius, tool/joint speed and tool/joint acceleration is configured for individual waypoints even though they may be linked.

Blending

Blending enables the robot to smoothly transition between two trajectories, without stopping at the waypoint between them.

Example Consider a pick and place application as an example (see figure 14.2), where the robot is currently at Waypoint 1 (WP_1), and it needs to pick up an object at Waypoint 3 (WP_3). To avoid collisions with the object and other obstacles (O), the robot must approach WP_3 in the direction coming from Waypoint 2 (WP_2). So three waypoints are introduced to create a path that fulfils the requirements.

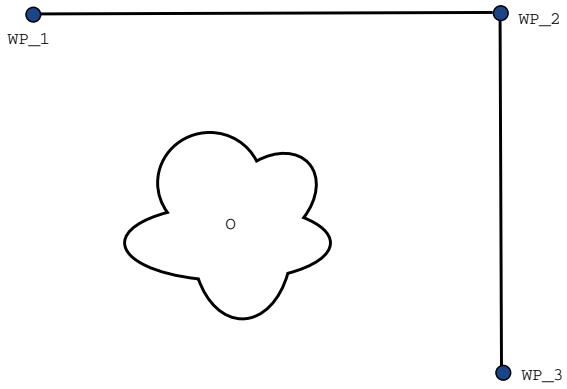


Figure 14.2: WP_1: initial position, WP_2: via point, WP_3: pick up position, O: obstacle.

Without configuring other settings, the robot will make a stop at each waypoint, before continuing the movement. For this task a stop at WP_2 is not optimal since a smooth turn would require less time and energy while still fulfilling the requirements. It is even acceptable that the robot does not reach WP_2 exactly, as long as the transition from the first trajectory to the second happens near this position.

The stop at WP_2 can be avoided by configuring a blend for the waypoint, allowing the robot to calculate a smooth transition into the next trajectory. The primary parameter for the blend is a radius. When the robot is within the blend radius of the waypoint it can start blending and deviate from the original path. This allows for faster and smoother movements, as the robot does not need to decelerate and re-accelerate.

Blend parameters Apart from the waypoints, multiple parameters will influence the blend trajectory (see figure 14.3):

- the blend radius (r)
- the initial and final speed of the robot (at positions p1 and p2, respectively)
- the movement time (e.g. if setting a specific time for a trajectory this will influence the initial/final speed of the robot)
- the trajectory types to blend from and to (MoveL, MoveJ)

If a blend radius is set, the robot arm trajectory blends around the waypoint, allowing the robot arm not to stop at the point.

Blends cannot overlap, so it is not possible to set a blend radius that overlaps with the blend radius of a previous or following waypoint as shown in figure 14.4.

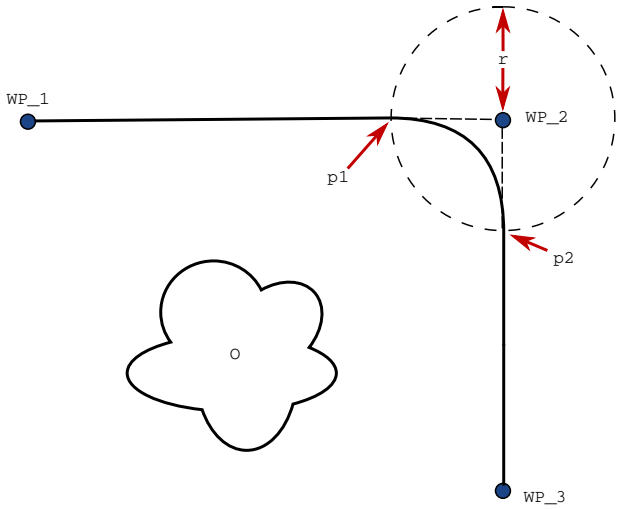


Figure 14.3: Blend over `WP_2` with radius `r`, initial blend position at `p1` and final blend position at `p2`. `O` is an obstacle.

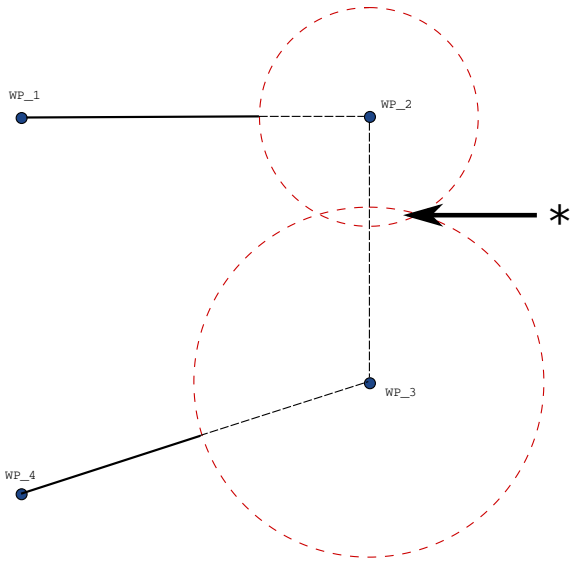


Figure 14.4: Blend radius overlap not allowed (*).

Conditional blend trajectories The blend trajectory is affected both by the waypoint where the blend radius is set and the following one in the program tree. That is, in the program in figure 14.5 the blend around `WP_1` is affected by `WP_2`. The consequence of this becomes more apparent when blending around `WP_2` in this example. There are two possible ending positions and to determine which is the next waypoint to blend to, the robot must evaluate the current reading of the `digital_input[1]` already when entering the blend radius. That means the **if...then** expression (or other necessary statements to determine the following waypoint, e.g. variable waypoints) is

evaluated before we actually reach WP_2 which is somewhat counter-intuitive when looking at the program sequence. If a waypoint is a stop point and followed by conditional expressions to determine the next waypoint (e.g. the I/O command) it is executed when the robot arm has stopped at the waypoint.

```
MoveL
    WP_I
    WP_1 (blend)
    WP_2 (blend)
    if (digital_input[1]) then
        WP_F_1
    else
        WP_F_2
```

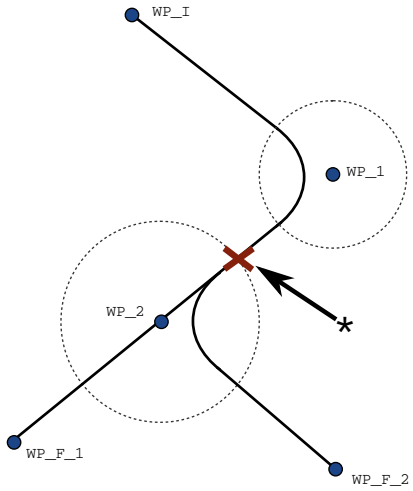


Figure 14.5: WP_I is the initial waypoint and there are two potential final waypoints WP_F_1 and WP_F_2, depending on a conditional expression. The conditional if expression is evaluated when the robot arm enters the second blend (*).

Trajectory type combinations It is possible to blend between all four combinations of trajectory types of **MoveJ** and **MoveL**, but the specific combination will affect the computed blend trajectory. There are 4 possible combinations:

1. **MoveJ** to **MoveJ** (Pure Joint space blend)
2. **MoveJ** to **MoveL**
3. **MoveL** to **MoveL** (Pure Cartesian space blend)
4. **MoveL** to **MoveJ**

Pure joint space blending (bullet 1) vs. pure Cartesian space blending (bullet 3) is compared in figure 14.6. It shows two potential paths of the tool for identical sets of waypoints.

Of the different combinations, bullets 2, 3 and 4 will result in trajectories that keep within the boundaries of the original trajectory in Cartesian space. An example of a blend between different trajectory types (bullet 2) can be seen in figure 14.7.

Pure joint space blends (bullet 1), however, may behave in a way that is less intuitive, since the robot will try to achieve the smoothest possible trajectory in Joint space taking velocities and time requirements into account. Due to this, they may deviate from the course specified by the waypoints. This is especially the case if there are significant differences in a joint's velocity between the

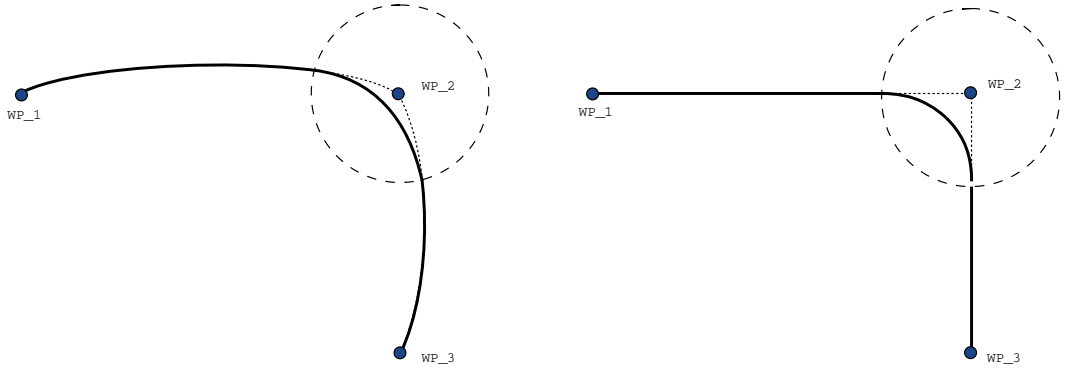


Figure 14.6: Joint space (MoveJ) vs. cartesian space (MoveL) movement and blend.

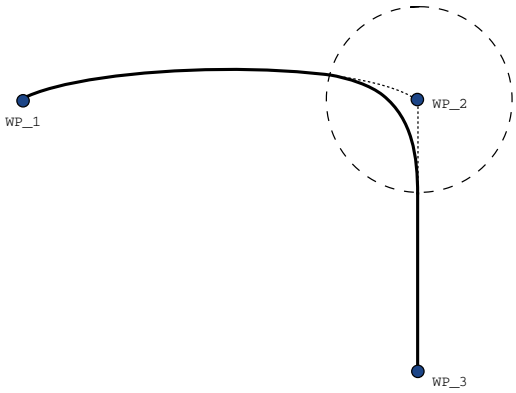


Figure 14.7: Blending from a movement in joint space (MoveJ) to linear tool movement (MoveL).

two trajectories. *Caution:* if the velocities are very different (e.g. by specifying advanced settings - either velocity or time - for a specific waypoint) this can result in large deviations from the original trajectory as shown in figure 14.8. If you need to blend between different velocities and cannot accept this deviation consider a blend in Cartesian space using **MoveL** instead.

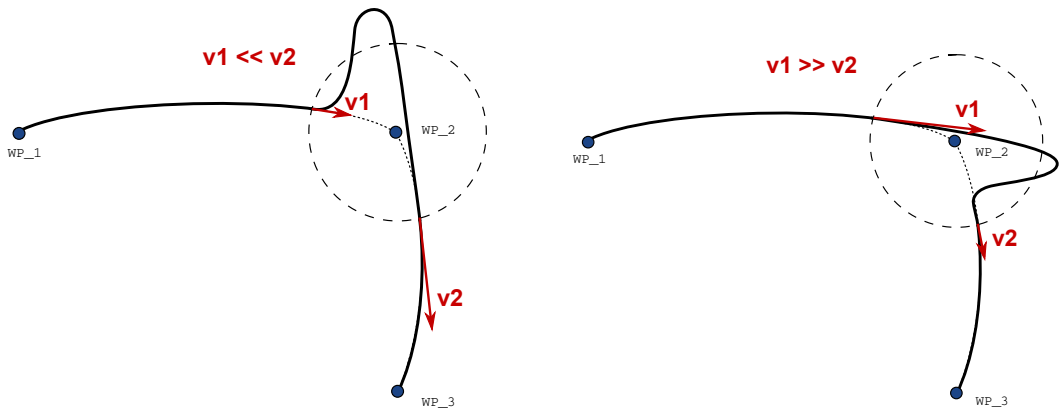
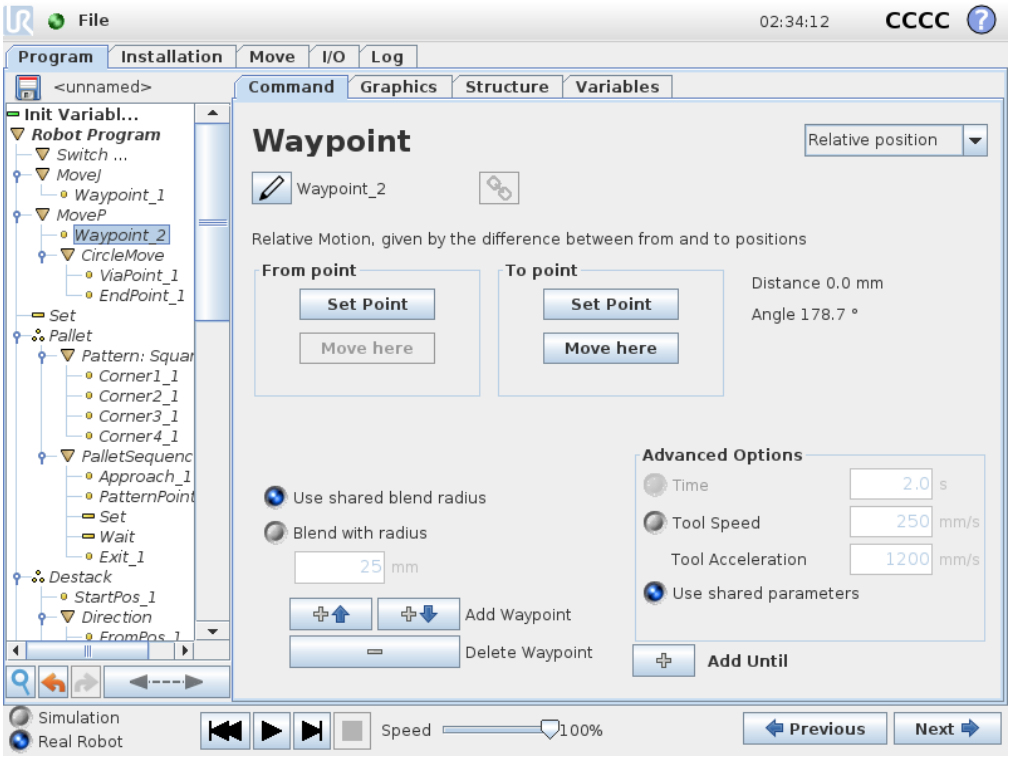


Figure 14.8: Joint space blending when initial velocity `v1` is significantly smaller than final velocity `v2` or the opposite.

14.6 Command: Relative Waypoint

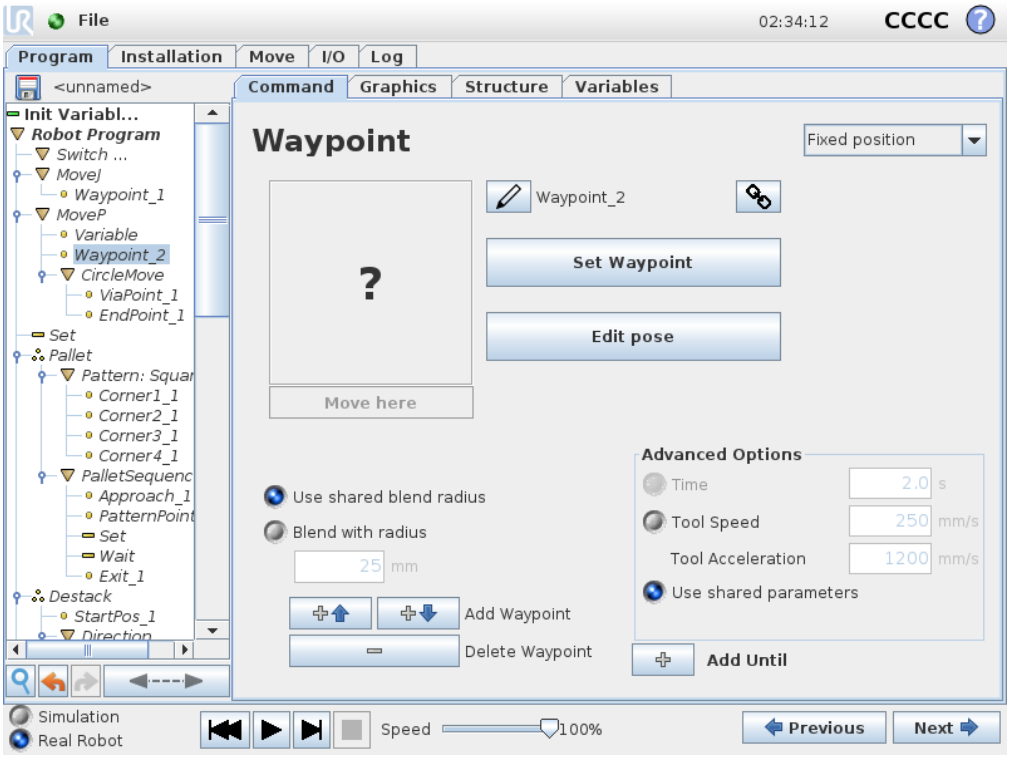


A waypoint with the position given relative to the robot arm's previous position, such as "two centimeters to the left". The relative position is defined as the difference between the two given positions (left to right).

Note: repeated relative positions can move the robot arm 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.

14.7 Command: 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. If a blend radius is set on a fixed waypoint and the waypoints preceding and succeeding it are variable or if the blend radius is set on a variable waypoint, then the blend radius will not be checked for overlap `(see 14.5)`. If, when running the program, the blend radius overlaps a point, the robot will ignore it and move to the next one.

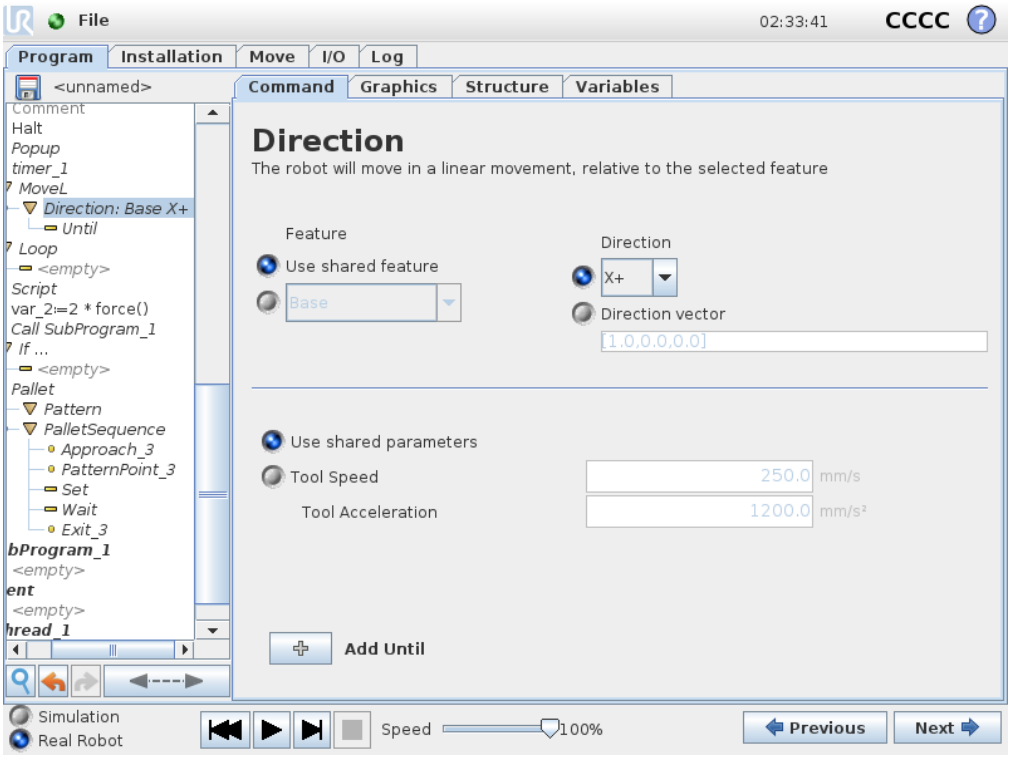
For example, to move the robot 20 mm along the z-axis of the tool:

```
var_1=p[0,0,0.02,0,0,0]
Movel
  Waypoint_1 (variable position):
    Use variable=var_1, Feature=Tool
```

14.8 Command: Direction

The program node *Direction* specifies a motion relative to feature axes or TCPs. The robot moves in along the path specified by the Direction Program Node until that movement is stopped by an

Until condition. You must have Until conditions for stopping a direction movement, tap the *Add Until* button to define the stop criteria.



Stopping a Direction Movement

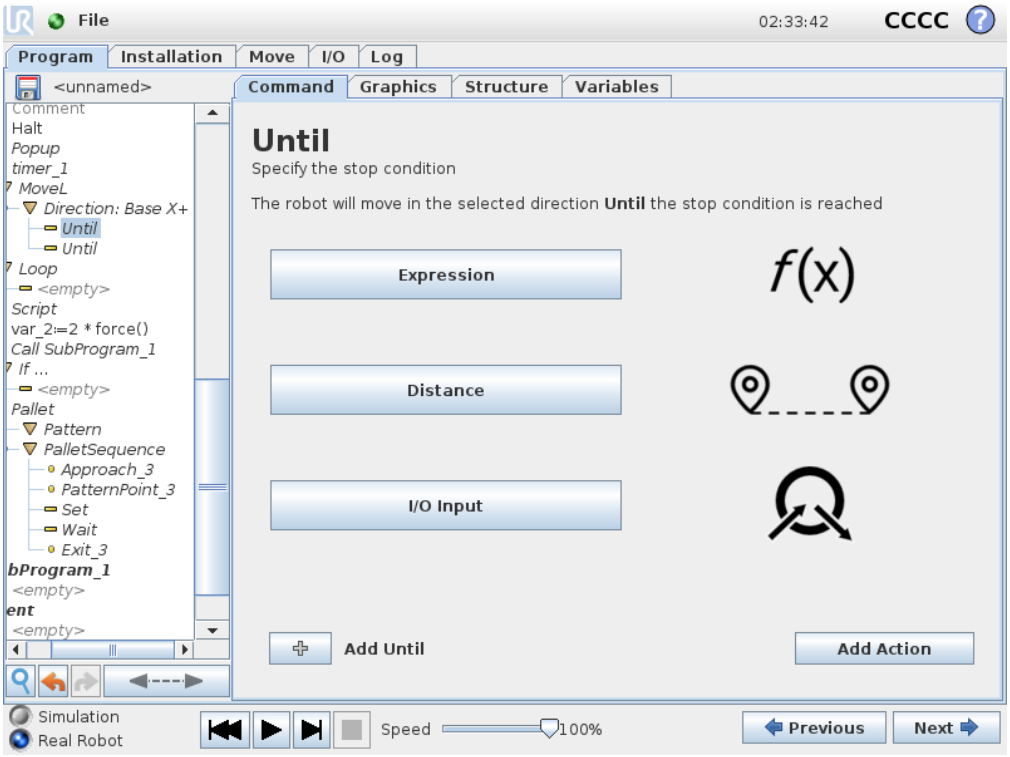
You can add Direction Vector settings, for **Tool Speed** and **Tool Acceleration**, to define the vector direction for linear motion, allowing for advanced uses as:

- defining linear motion relative to multiple feature axes
- computing the direction as a mathematical expression

The Direction Vectors defines a custom code expression that is resolved to a unit vector. For example, for a vector of [2,1,0] the robot moves two units in the x direction for every unit in the y direction, relative to the specified speed.

14.9 Command: Until

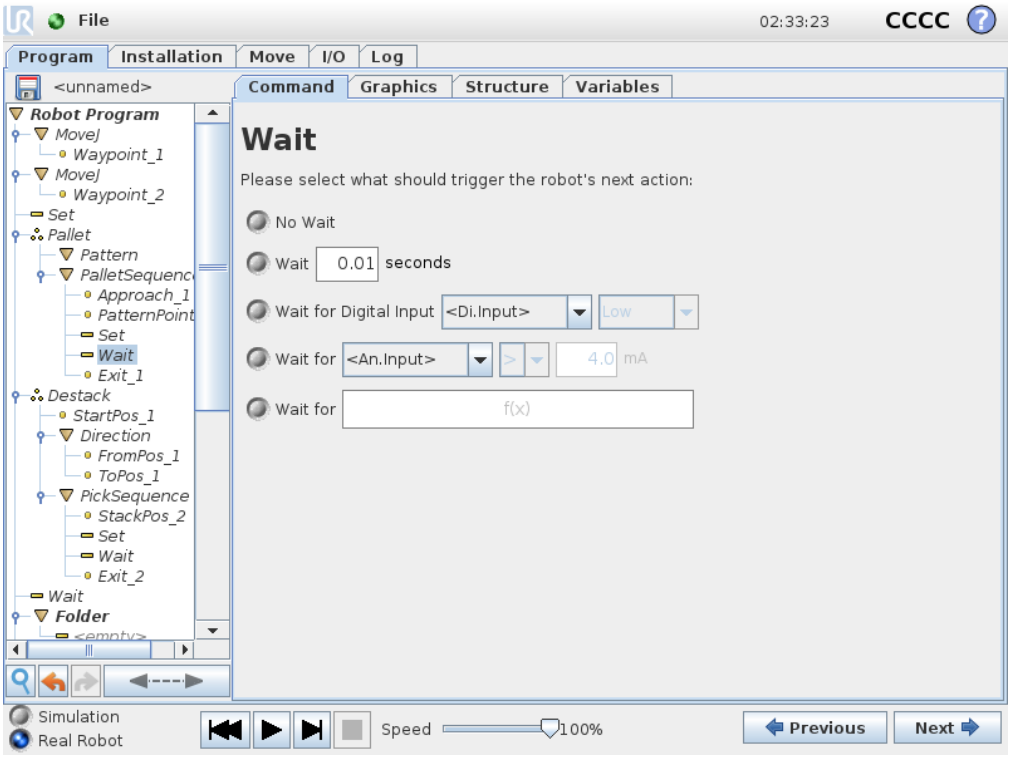
The program node **Until** defines a stop criterion for a motion. The robot moves along a path and stops when contact is detected.



In the **Until** field, you can define the following stop criteria:

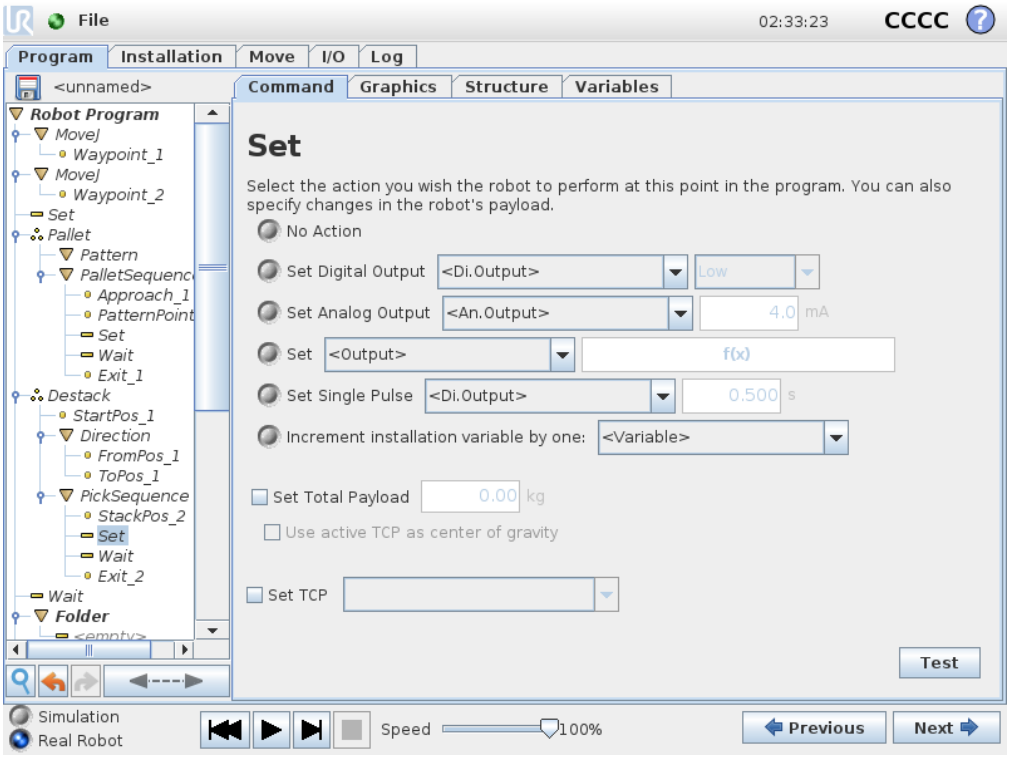
- **Add Action** Add program nodes if a specific until condition is met. For instance if an error state is detected, the program can be stopped with a Popup node.
- **Distance** This node can be used to stop a Direction move when the robot has moved a certain distance. The velocity is ramped down so the robot stops exactly at the distance.
- **Expression** This node can be used to stop the motion due to a custom program expression. You can use I/Os, variables or script functions to specify the stop condition.
- **I/O Input** You can use this node to stop a signal controlled motion on an I/O Input.

14.10 Command: Wait



Wait pauses I/O signal, or expression, for a given amount of time. If **No Wait** is selected, nothing is done.

14.11 Command: Set



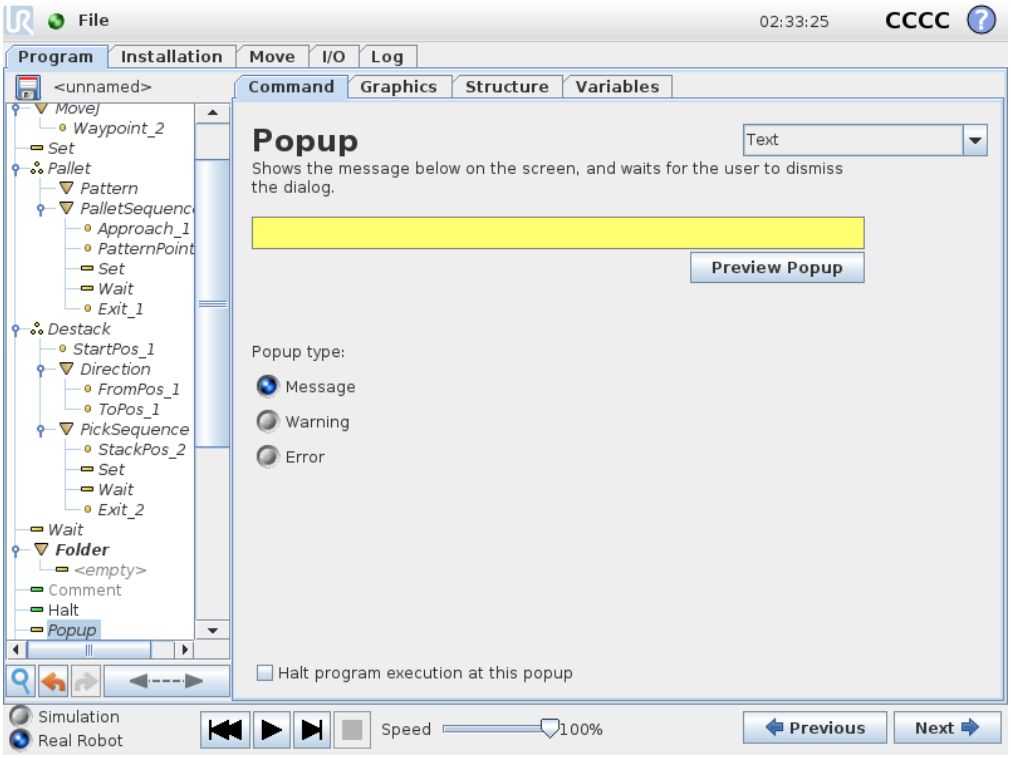
Sets either digital or analog outputs to a given value. Digital outputs can also be set to send a single pulse.

Use the Set command to set the payload of the Robot Arm. You can adjust the payload weight to prevent the robot from triggering a protective stop, when the weight at the tool differs from the expected payload. If the active TCP should not be used as the center of gravity the checkbox must be unchecked.

The active TCP can also be modified using a **Set** command, by selecting the check box and choosing one of the TCP offsets from the menu.

If the active TCP for a particular motion is known at the time of writing of the program, consider using the TCP selection on the **Move** card instead (see 14.4). For further information about configuring named TCPs (see 13.6).

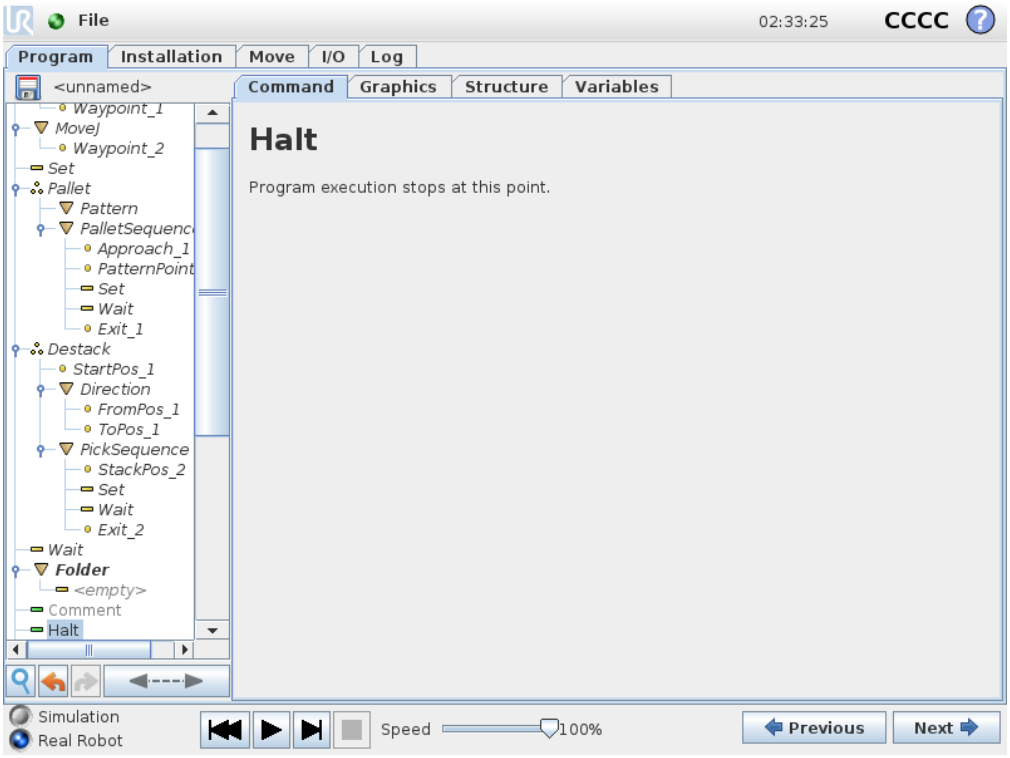
14.12 Command: Popup



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.

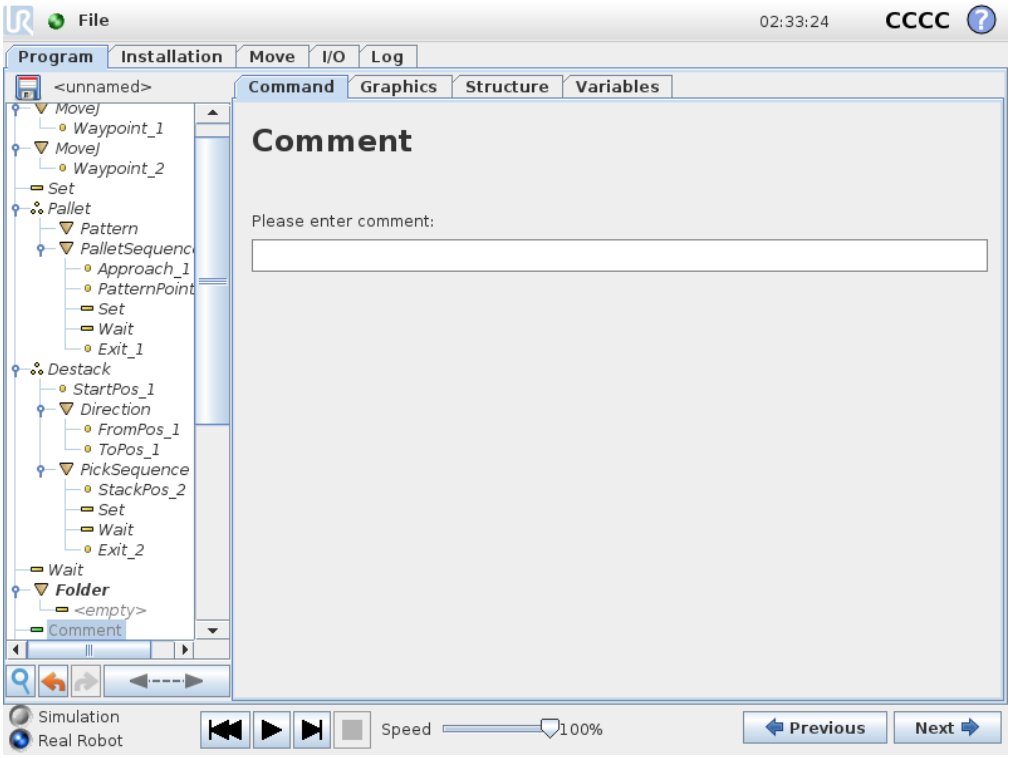
Note: Messages are limited to a maximum of 255 characters.

14.13 Command: Halt



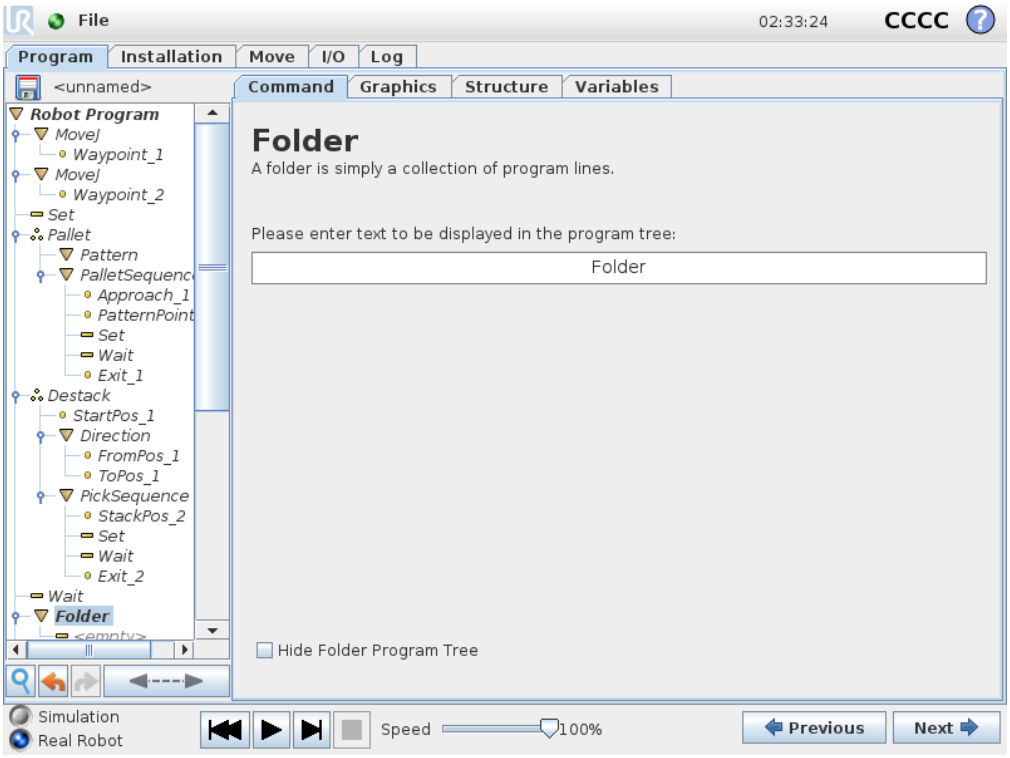
The program execution stops at this point.

14.14 Command: 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.

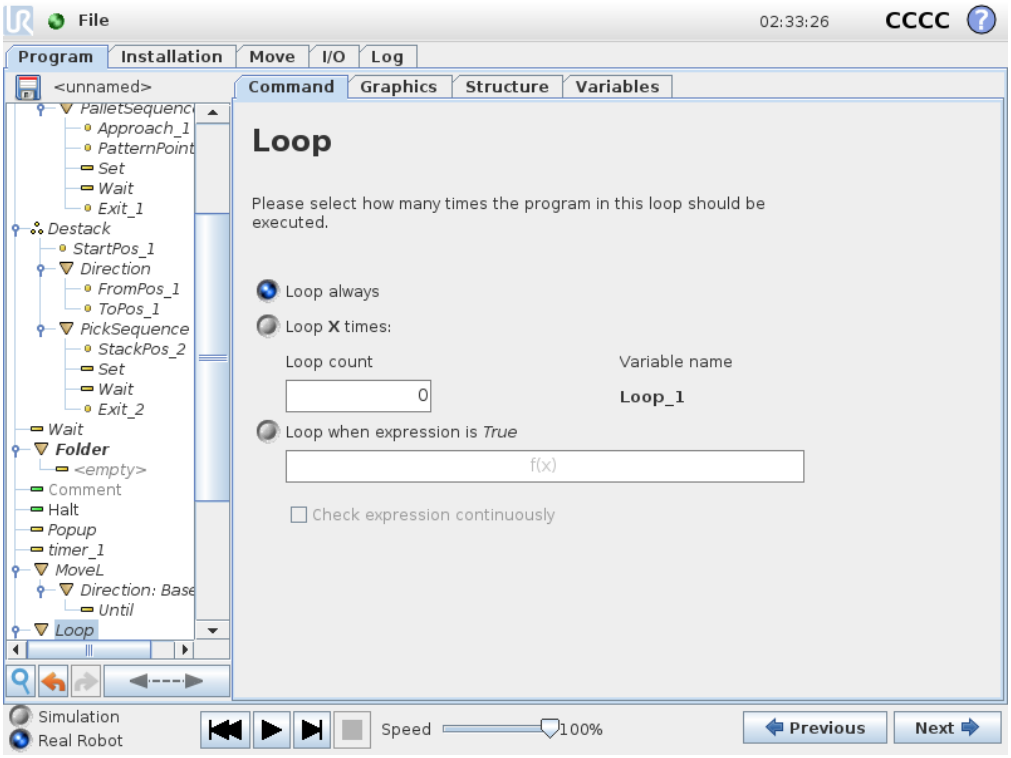
14.15 Command: Folder



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.

Folders have no impact on the program and its execution.

14.16 Command: Loop

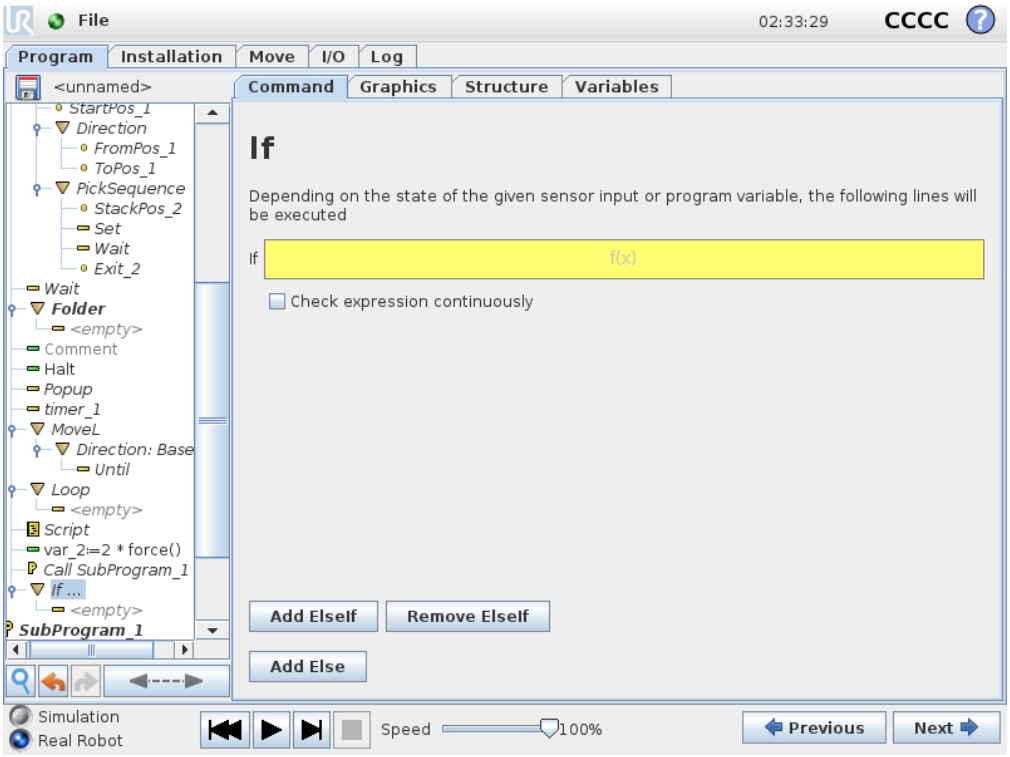


Loops the underlying program commands. Depending on the selection, the underlying program commands are either looped infinitely, a certain number of times or as long as the given condition is true. When looping a certain number of times, a dedicated loop variable (called `loop_1` in the screen shot above) 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.

14.17 Command: If

`If` and `If...Else` statements change the robot's behavior based on sensor inputs or variable values.



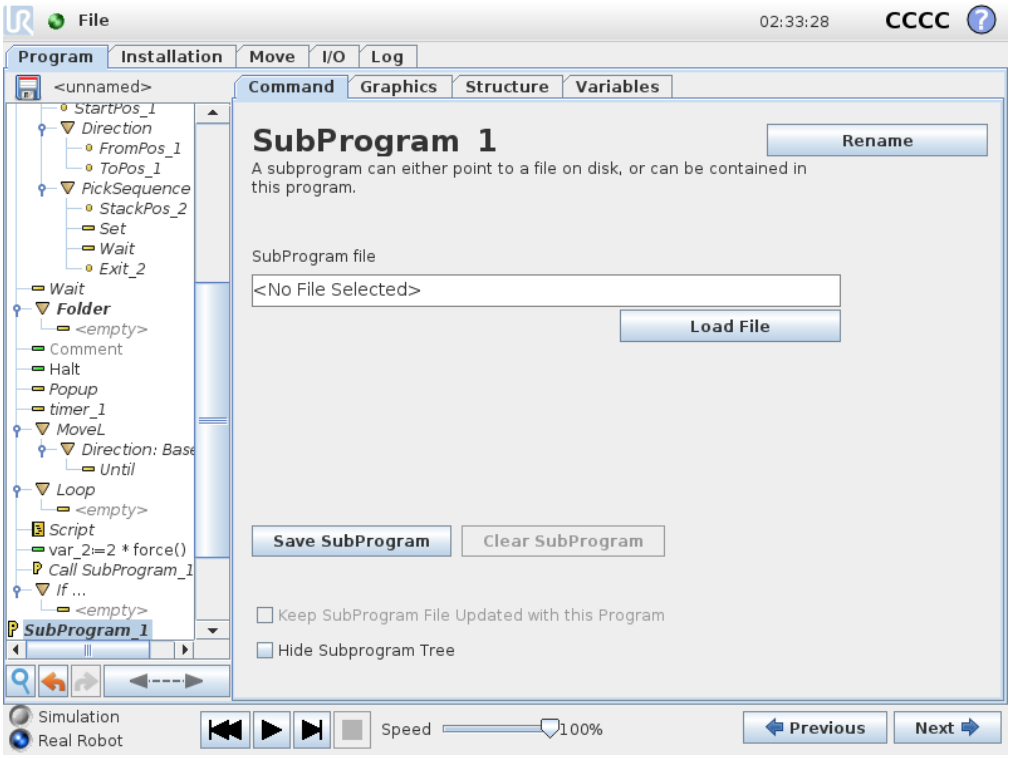
Select conditions in the Expression Editor that make up expressions using an `If` statement. If a condition is evaluated as True, the statements within this `If` command are executed. An `If` statement can have only one `Else` statement. Use `Add ElseIf` and `Remove ElseIf` to add and remove ElseIf expressions.

Select `Check Expression Continuously` to allow `If`, `ElseIf` and `Loop` statements to be evaluated while the contained lines are executed. If an expression inside an `If` statement is evaluated as False, the `ElseIf` or `Else` statements are followed.

NOTE:

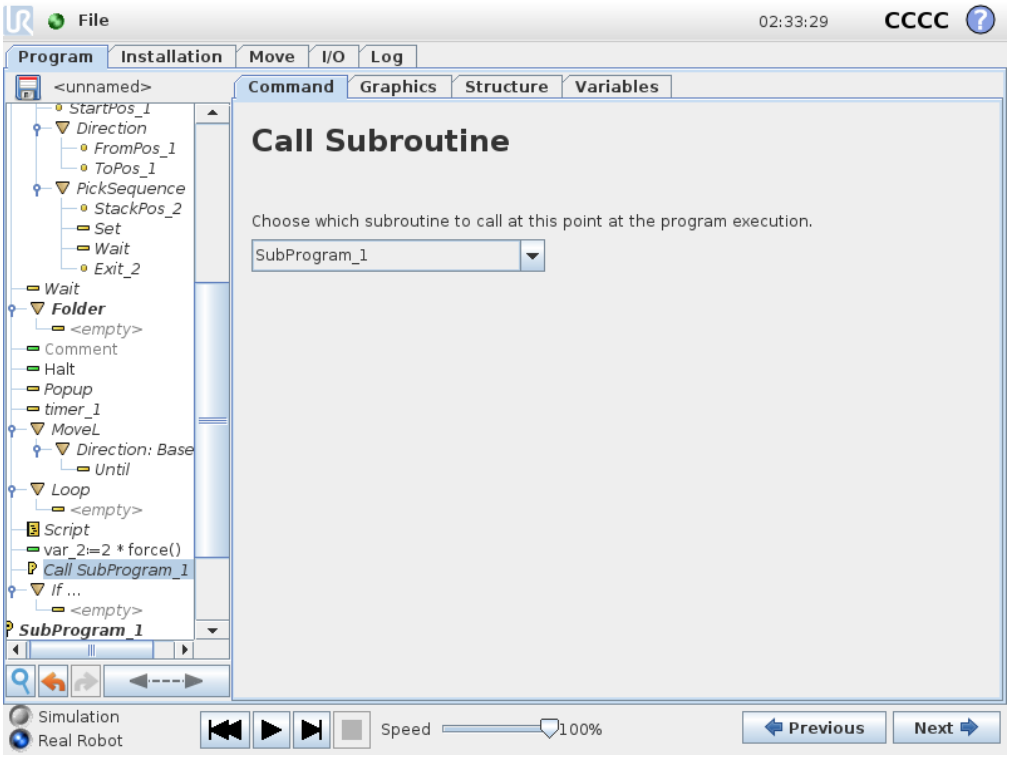
If there are waypoints inside an `If` expression or inside a `Loop` expression with the `Check Expression Continuously` option, you can add a `stopj()` or a `stopl()` after the expression to gently decelerate the robot arm. This is valid for both If and Loop Commands `(see section 14.16)`.

14.18 Command: SubProgram



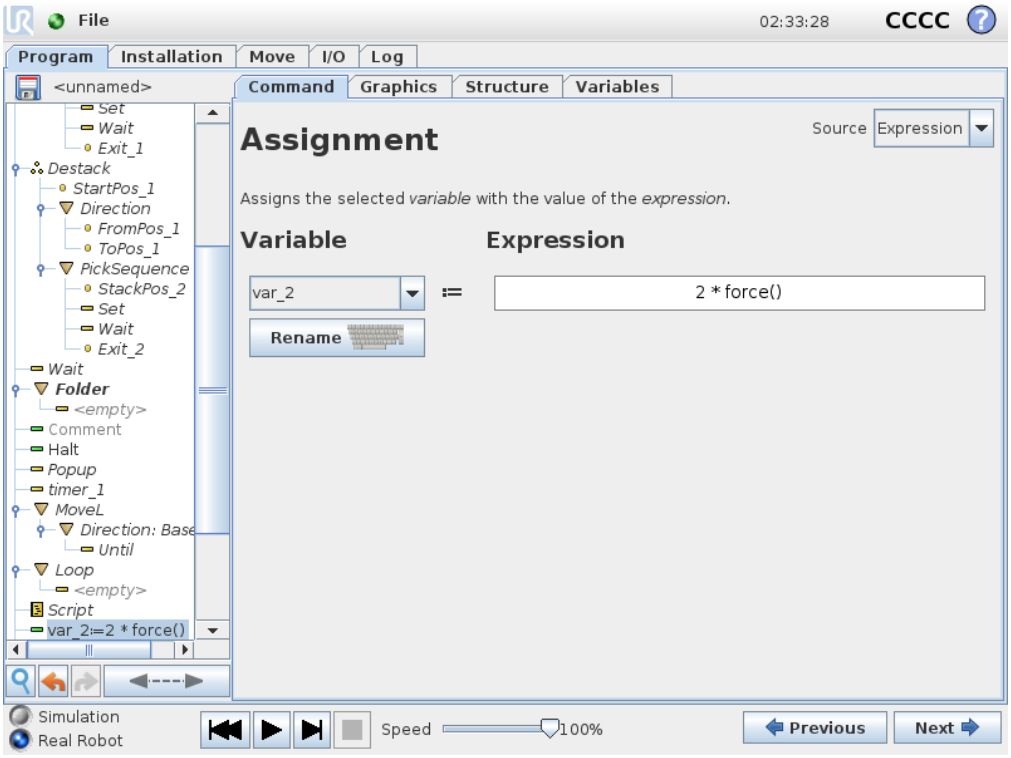
A Sub Program can hold program parts that are needed several places. A Sub Program can be a separate file on the disk, and can also be hidden to protect against accidental changes to the SubProgram.

Command: Call SubProgram



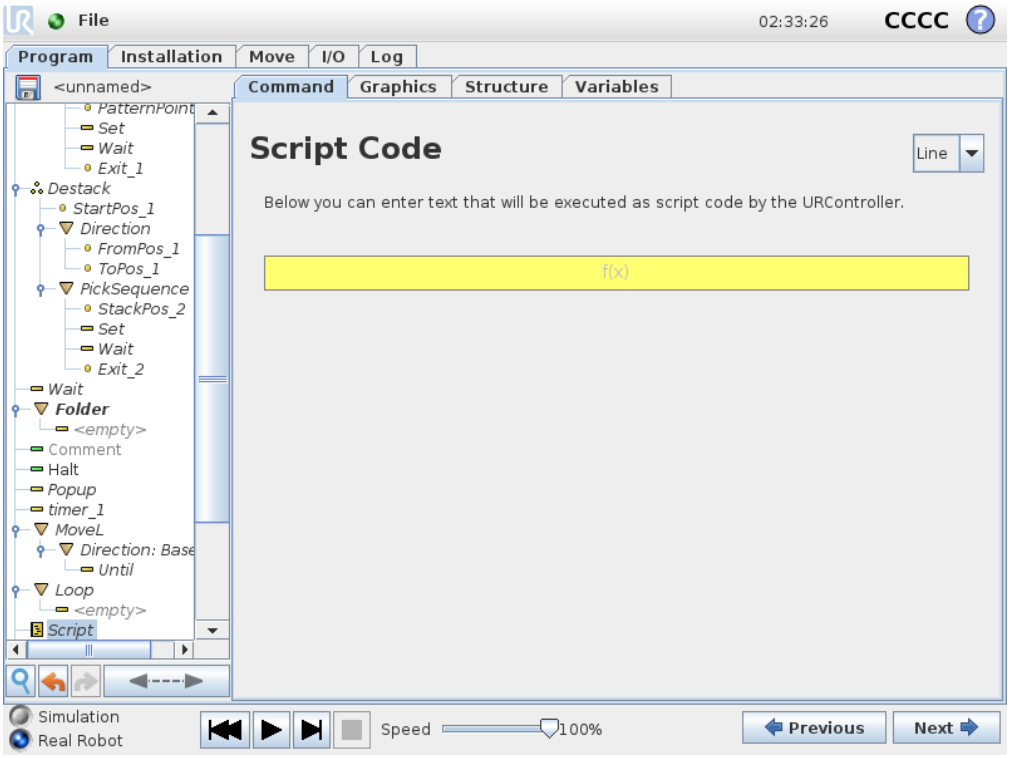
A call to a sub program will run the program lines in the sub program, and then return to the following line.

14.19 Command: 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.

14.20 Command: Script



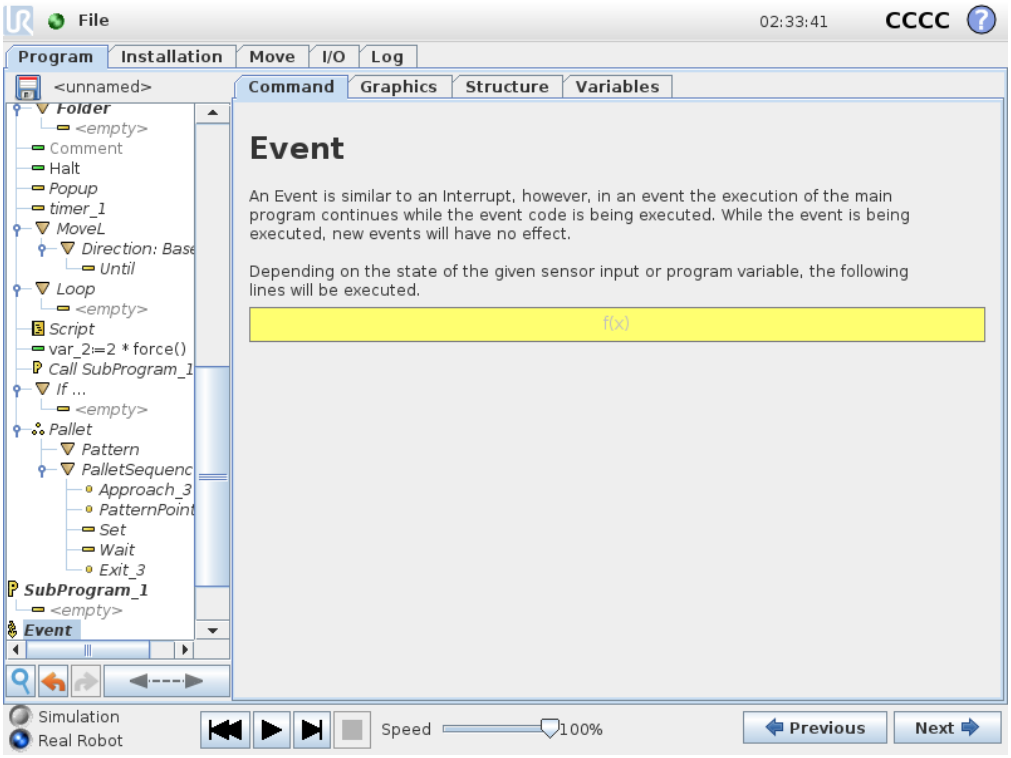
The following options are available in the drop down list under Command:

- **Line** allows you to write a single line of URscript code, using the Expression Editor (12.1)
- **File** allows you to write, edit or load URscript files.

You can find instructions for writing URscript in the Script Manual on the support website (http://www.universal-robots.com/support).

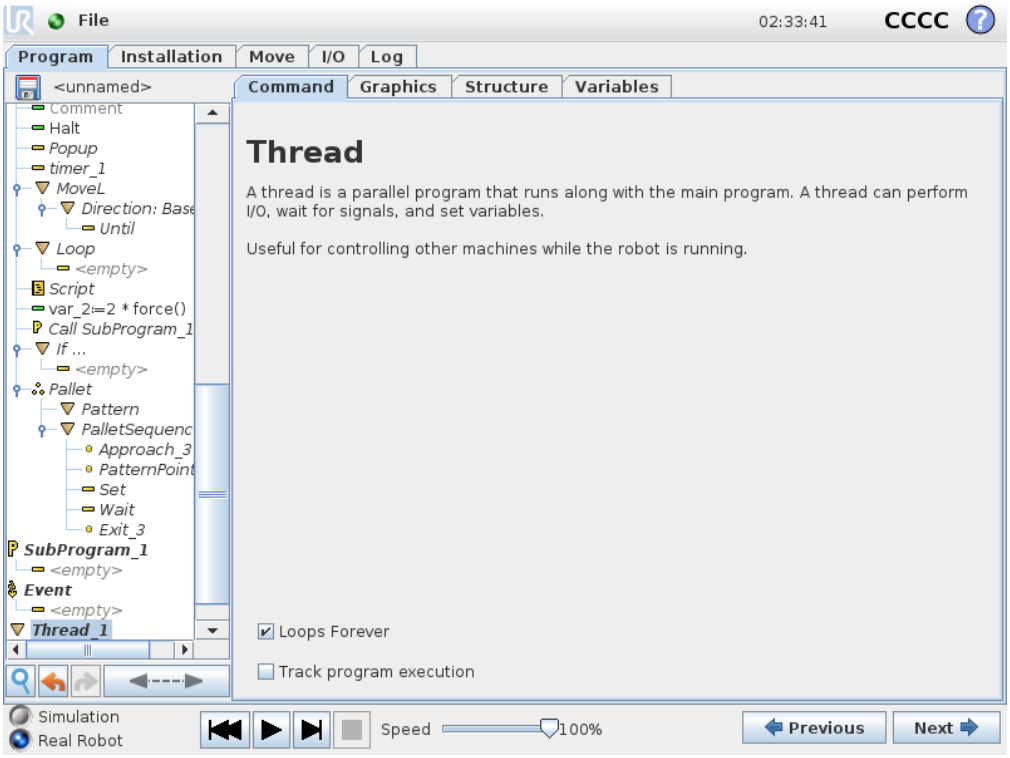
Functions and variables declared in a URscript file are available for use througout the program in the PolyScope.

14.21 Command: Event



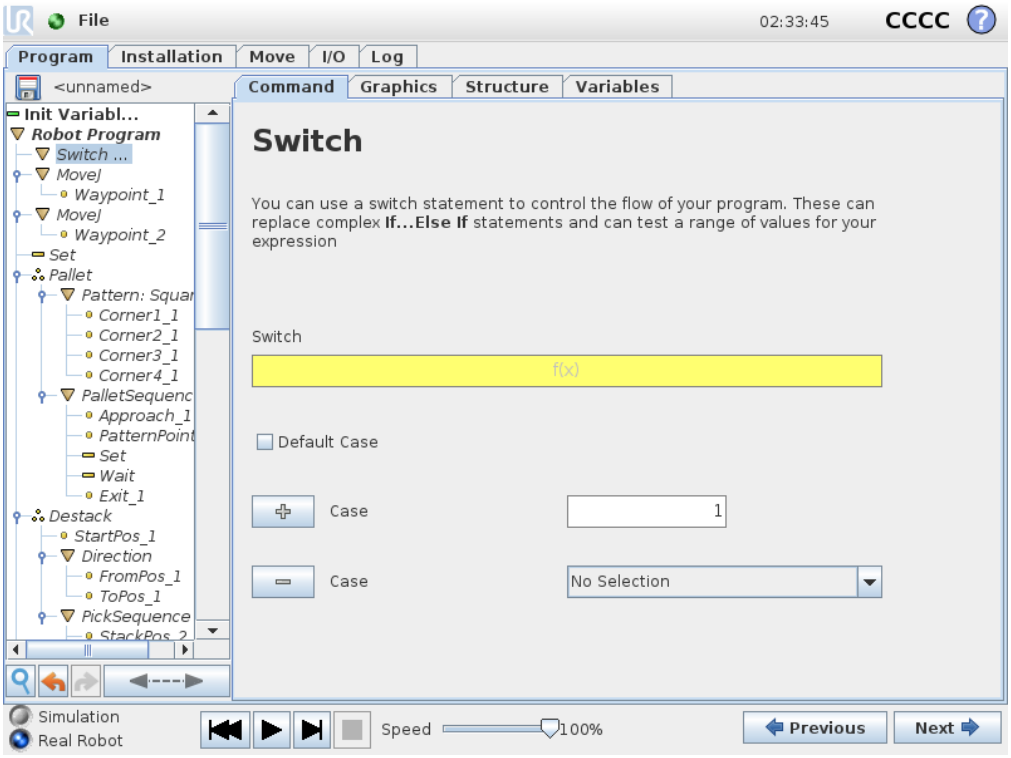
An event can be used to monitor an input signal, and perform some action or set a variable when that input signal goes high. For example, in the event that an output signal goes high, the event program can wait for 200ms and then set it back to low again. This can make the main program code a lot simpler in the case on an external machine triggering on a rising flank rather than a high input level. Events are checked once every control cycle (8ms) .

14.22 Command: 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.

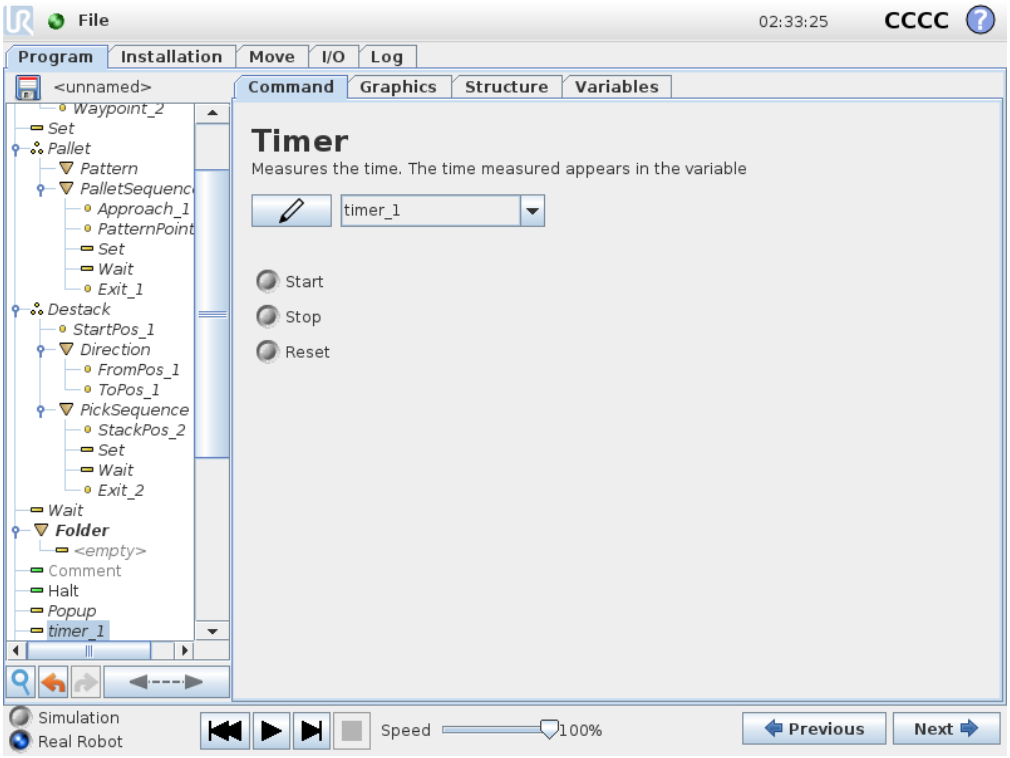
14.23 Command: Switch



A **Switch Case** construction can make the robot change behavior based on sensor inputs or variable values. Use the **Expression Editor** to describe the base condition and define the cases under which the robot should proceed to the sub-commands of this `Switch`. If the condition is evaluated to match one of the cases, the lines inside the `Case` are executed. If a `Default Case` has been specified, then the lines will be executed only if no other matching cases were found.

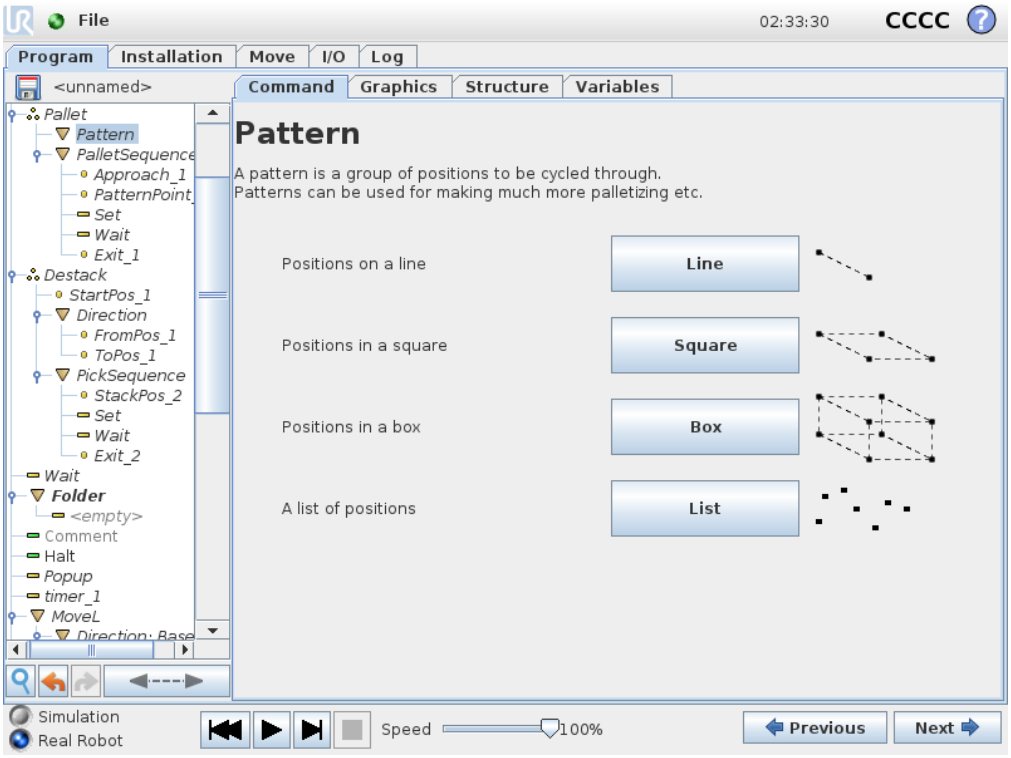
Each `Switch` can have several `Cases` and one `Default Case`. `Switches` can only have one instance of any `Case` values defined. `Cases` can be added using the buttons on the screen. A `Case` command can be removed from the screen for that switch.

14.23.1 Timer



A Timer measures the length of time it takes for specific parts of the program to run. A program variable contains the time passed since a Timer started, and can be seen in the Variables Tab and in the Run Tab.

14.24 Command: Pattern



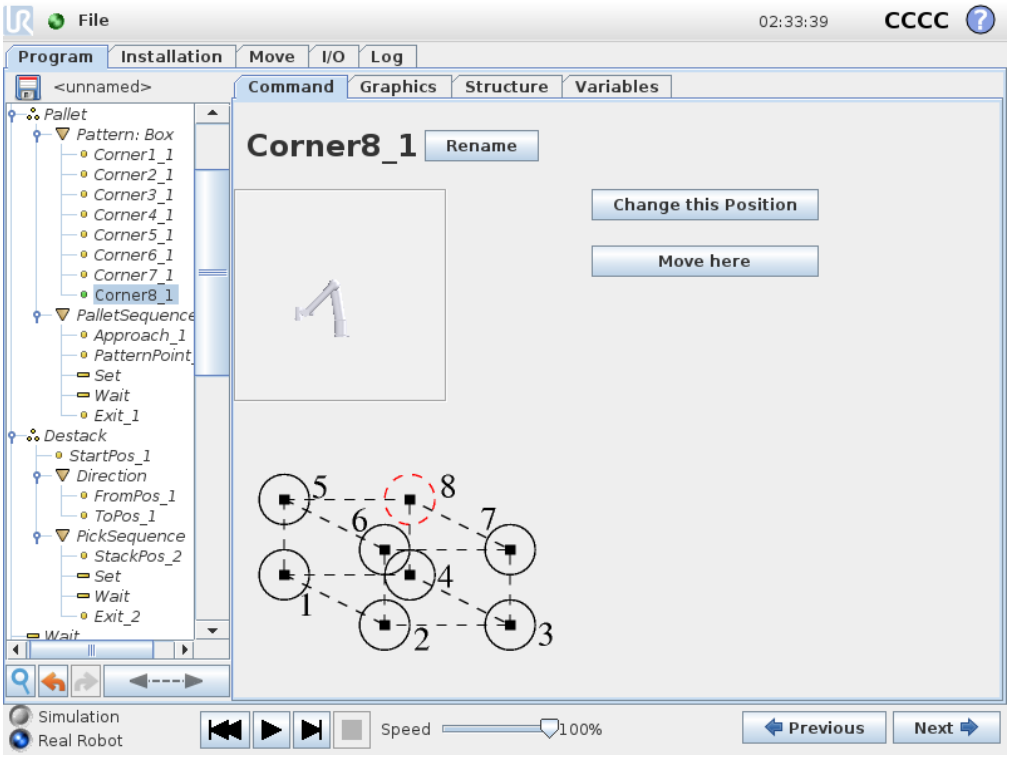
The **Pattern** command can be used to cycle through positions in the robot program. The **Pattern** command corresponds to one position at each execution.

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.

14.25 Command: Force

In the robot workspace **Force mode** allows for compliance and force in selectable axes. All robot arm movements under a **Force** command are in **Force mode**. When the robot arm is moving in **Force mode**, it is possible to select one or more axes there the robot arm is compliant. The robot arm complies with the environment along a compliant axes. This means the robot arm automatically adjusts its position in order to achieve the desired force. It is also possible to make the robot arm itself apply a force to its environment, e.g. a workpiece.

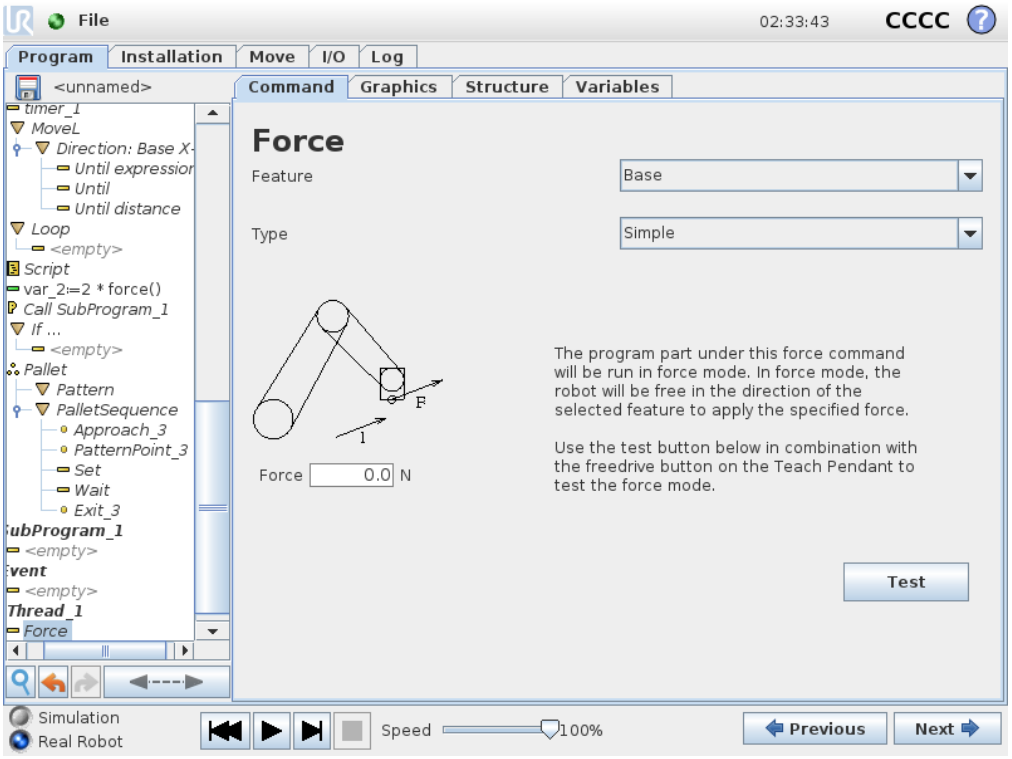
Force mode is suited to applications where the actual TCP position along a predefined axis is not important, but instead a desired force along that axis is required. For example if the robot TCP rolls against a curved surface, pushes or pulls a workpiece. **Force mode** also supports applying certain torques around predefined axes.

Note: if no obstacles are met in an axis where a non-zero force is set, the robot arm attempts to accelerate along that axis.
Although an axis is selected to be compliant, the robot program still tries to move the robot along that axis. However, force control assures that the robot arm still approaches the specified force.

NOTE:
If there is a Force node inside an `If`, `ElseIf` or `Loop`, and the `Check Expression Continuously` option is selected, you can add an `end_force_mode()` script at the end of the expression to exit force control.

WARNING:

1. Avoid high deceleration just before entering force mode.
2. Avoid high acceleration in force mode, since it decreases force control accuracy.
3. Avoid movements parallel to compliant axes before entering 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 13.12).

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 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 arm's motion, and in the x-y plane of the selected feature. This can be useful when de-burring along a complex path, where a force is needed perpendicular to the TCP motion.
  Note: when the robot arm is not moving: If force mode is entered with the robot arm standing still, there will be no compliant axes until the TCP speed is above zero. If later, while still in force mode, the robot arm 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 (see 14.30), when the robot is operating in force mode.

Force value selection

- Force or torque value can be set for compliant axes, and robot arm adjusts its position to achieve the selected force.
- For non-compliant axes robot arm will follow the trajectory set by the program.

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

NOTE:
You must do the following:

- Use `get_tcp_force()` script function in separate thread, to read actual force and torque.
- Correct wrench vector, if actual force and/or torque is lower than requested.

Limits selection

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

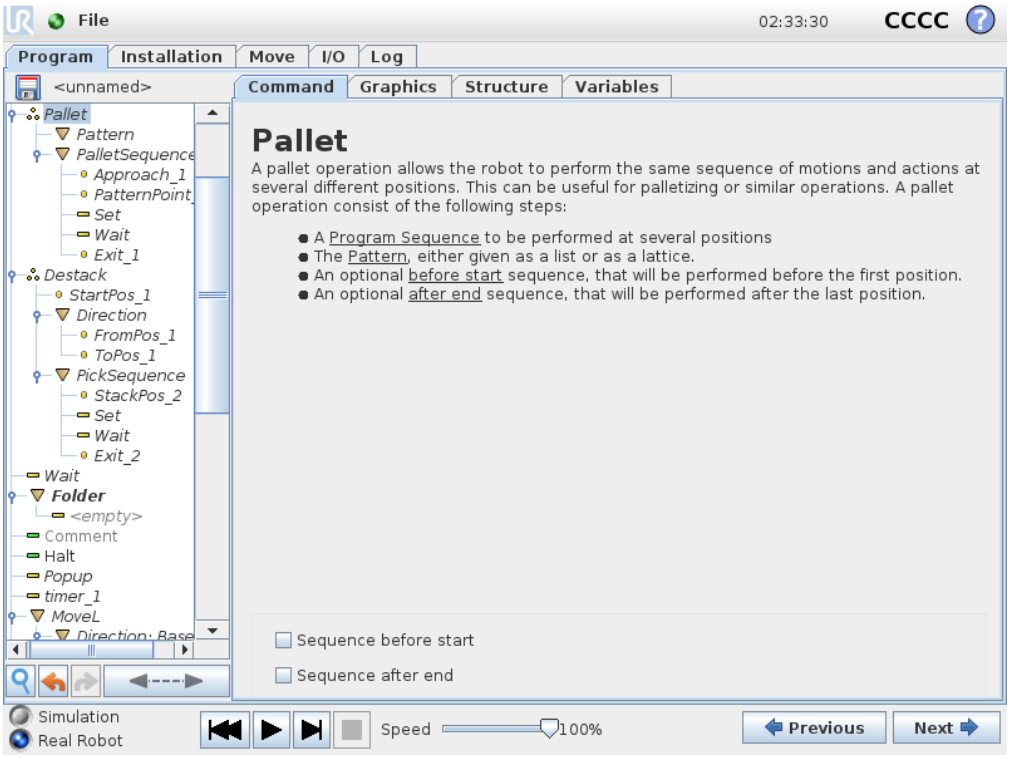
- **Compliant**: The limit is the maximum speed the TCP is allowed to attain along/about the axis. Units are [mm/s] and [deg/s].
- **Non-compliant**: The limit is the maximum deviation from the program trajectory which is allowed before the robot protective stops. Units are [mm] and [deg].

Test force settings

The on/off button, labelled **Test**, toggles the behavior of the **Freedrive** button on the back of the Teach Pendant from normal Freedrive mode to testing the force command.

When the **Test button** is on and the **Freedrive** 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 **Freedrive** button with the other, and notice in which directions the robot arm can/cannot be moved. Upon leaving this screen, the Test button automatically switches off, which means the **Freedrive** button on the back of the Teach Pendant is again used for regular **Freedrive** mode.
Note: The **Freedrive** button will only be effectual when a valid feature has been selected for the Force command.

14.26 Command: Pallet



A pallet operation can perform a sequence of motions in a set of places given as a pattern (see 14.24). At each of the positions in the pattern, the sequence of motions will be run relative to the pattern position.

Programming a Pallet Operation

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 a **Pallet Sequence** node, the motions of the robot arm are relative to the pallet position. The behavior of a sequence is such that the robot arm 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"

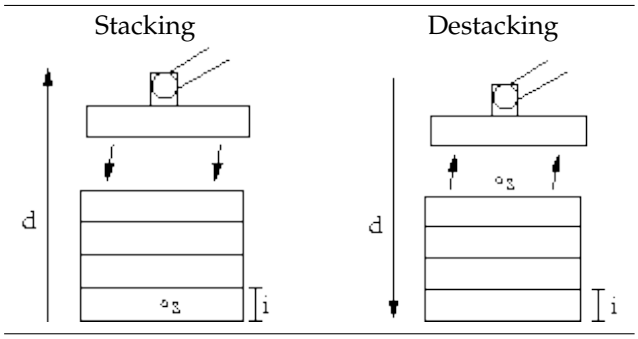
The optional **BeforeStart** sequence is run just before the operation starts. This can be used to wait for ready signals.

"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.

14.27 Command: Seek

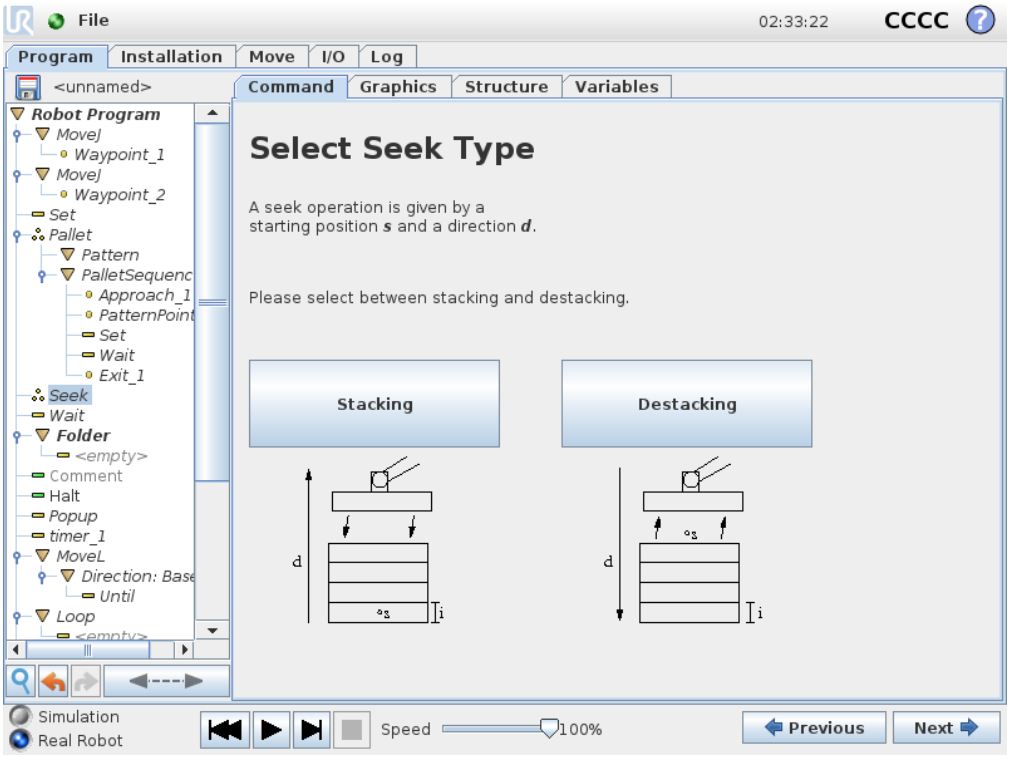
A seek function uses a sensor to determine when the correct position is reached to grab or drop an item. The sensor can be a push button switch, a pressure sensor or a capacitive sensor. This function is made for working on stacks of items with varying item thickness, or where the exact positions of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

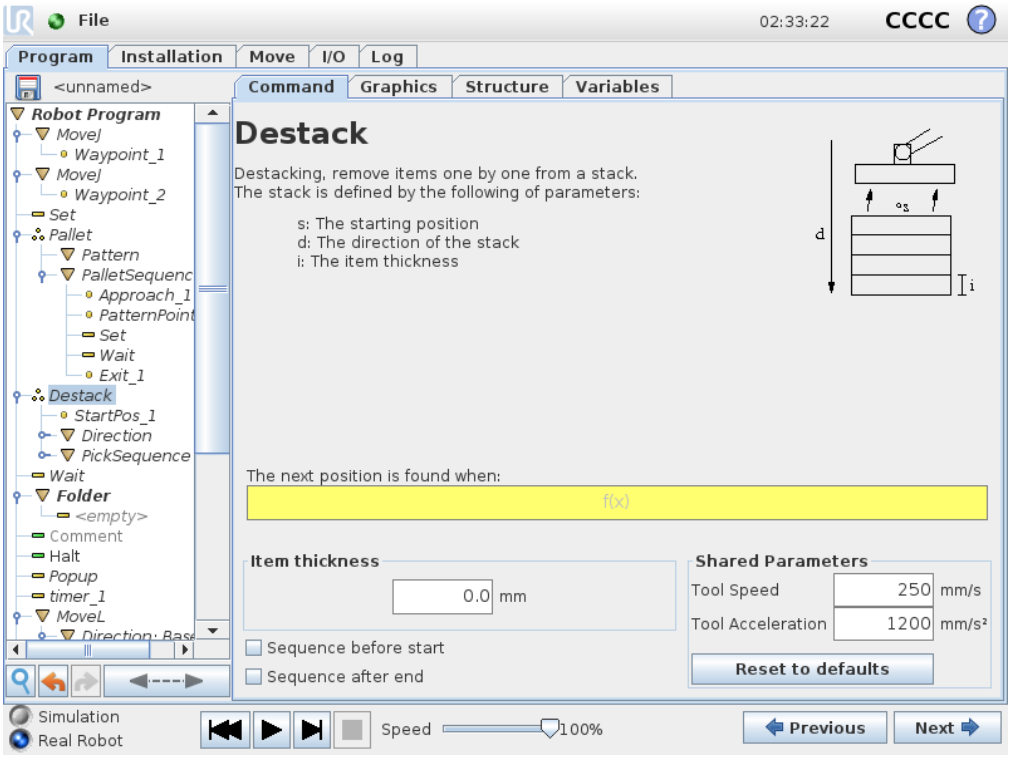
On top of this, one must define the condition for when the next stack position is reached, and a special program sequence that will be performed at each of the stack positions. Also speed and accelerations need to be given for the movement involved in the stack operation.

Stacking



When stacking, the robot arm 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

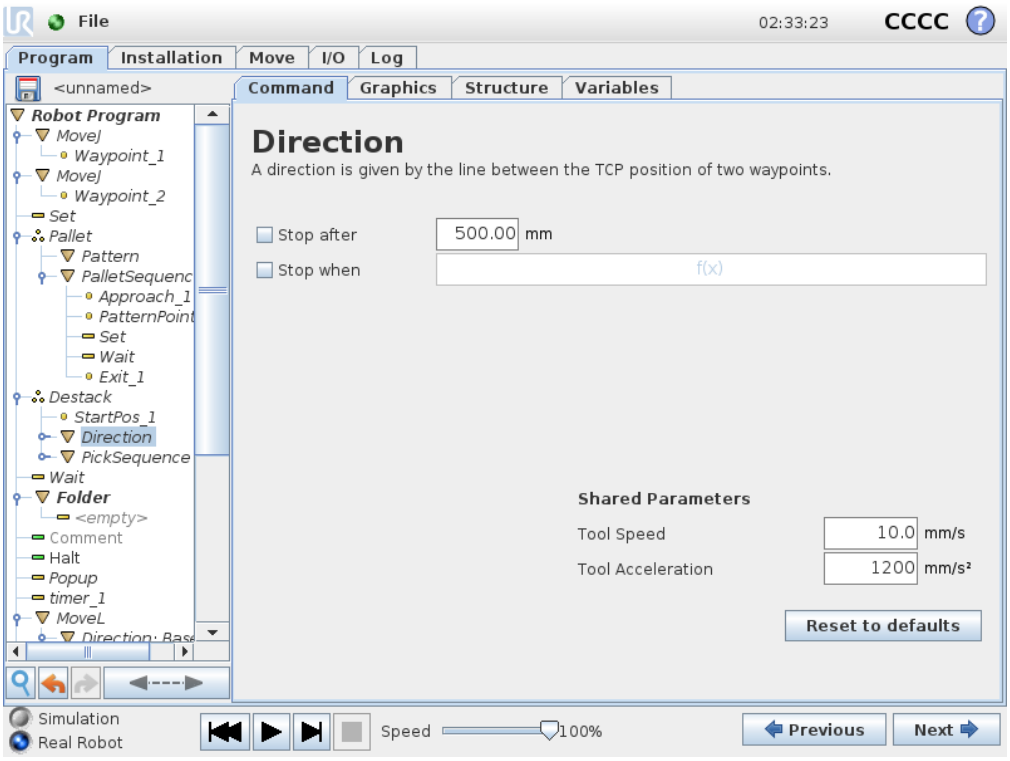


When destacking, the robot arm moves from the starting position in the given direction to search for the next item. The condition on the screen determines when the next item is reached. When the condition becomes satisfied, the robot remembers the position and performs the special sequence. The next 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 robot arm's 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 arm 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

The Pick/Place Sequence is a special program sequence performed at each stack position, similar to the Pallet operation `(see 14.26)`.

14.28 Command: Conveyor Tracking

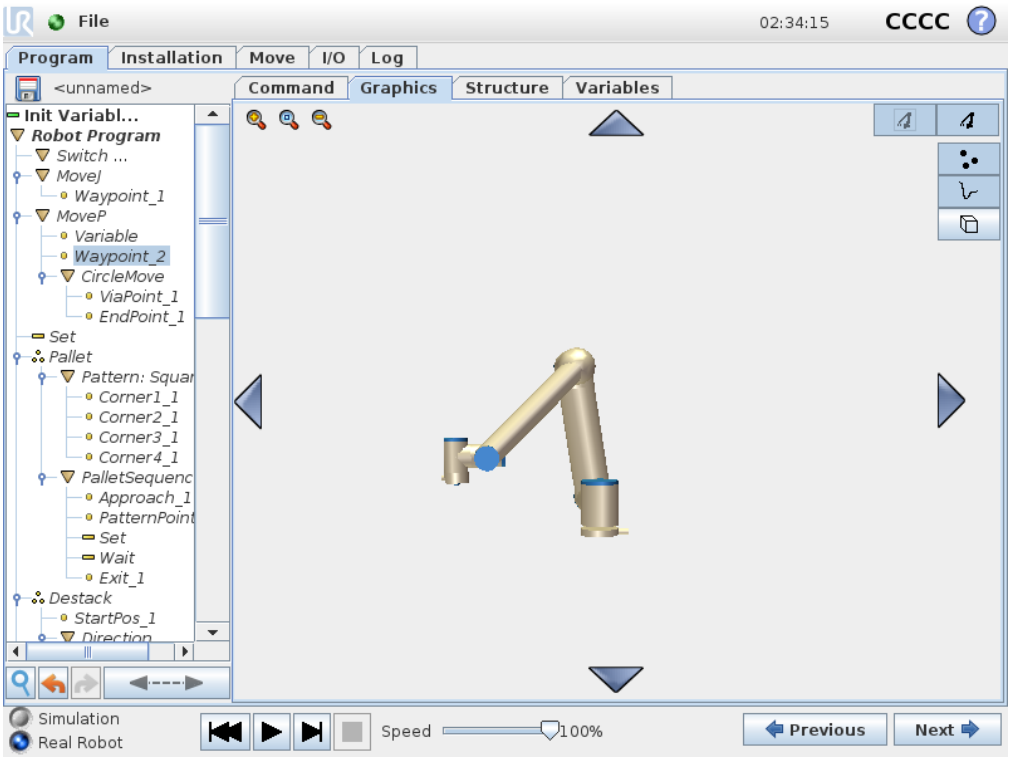
The robot can be configured to track the movement of one configured conveyor (Conveyor 1). When the **Conveyor Tracking** defined in the installation is configured correctly, the robot adjusts the movements to follow the conveyor. The **Conveyor Tracking** program node is available from the **Wizards** tab under the `Structure` tab. All movements under this node are allowed while tracking the conveyor, but they are relative to the motion of the conveyor belt. The **Conveyor Tracking** setup under the **Installation** tab `(see section 13.13)` provides options for configuring the robot to work with absolute and incremental encoders, as well as, linear and circular conveyors.

NOTE:
The Control Box can only accomodate one incremental encoder that must be used with one conveyor (Conveyor 1) .

14.29 Command: 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.

14.30 Graphics Tab



Graphical representation of the current robot program. The path of the TCP is shown in 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 arm shows the current position of the robot arm, and the *shadow* of the robot arm shows how the robot arm intends to reach the waypoint selected in the left hand side of the screen.

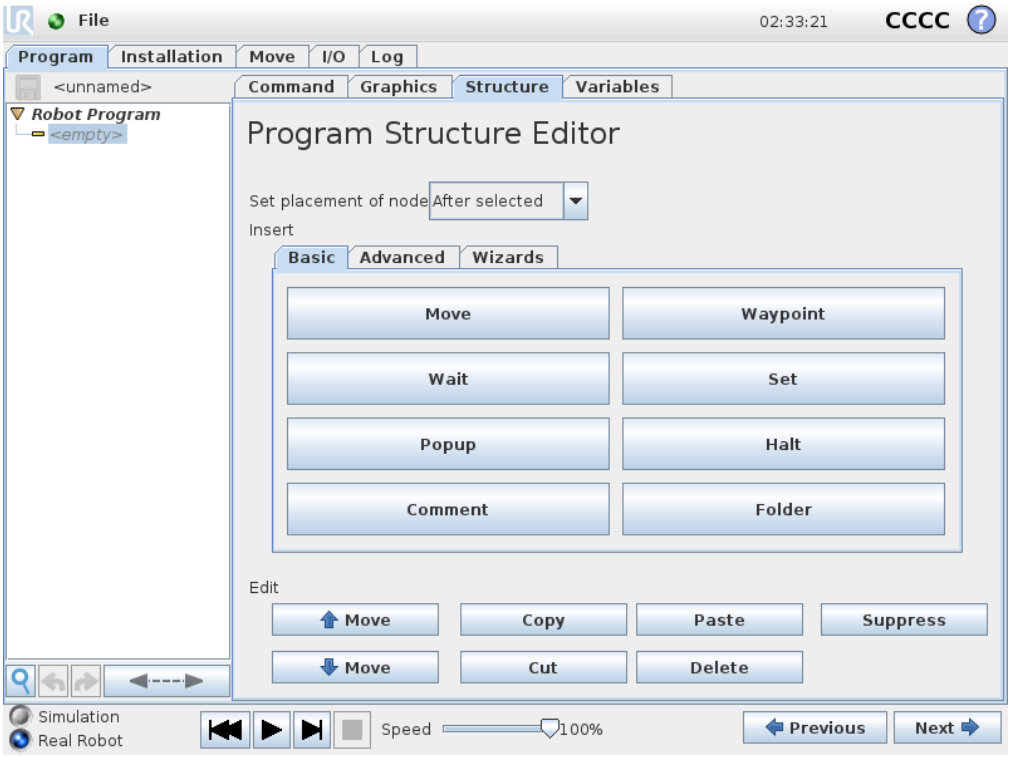
If the current position of the robot TCP comes close to a safety or trigger plane, or the orientation of robot tool is near the tool orientation boundary limit (see 10.12), a 3D representation of the boundary limit is shown.
Note: when the robot is running a program, the visualization of boundary limits will be disabled.

Safety planes are visualized in yellow and black with a small arrow representing the plane normal, which indicates the side of the plane on which the robot TCP is allowed to be positioned. Trigger planes are displayed in blue and green and a small arrow pointing to the side of the plane, where the **Normal** mode limits (see 10.6) are active. The tool orientation boundary limit is visualized with a spherical cone together with a vector indicating the current orientation of the robot tool. The inside of the cone represents the allowed area for the tool orientation (vector).

When the target robot TCP no longer is in the proximity of the limit, the 3D representation disappears. If the TCP is in violation or very close to violating a boundary limit, the visualization of the limit turns red.

The 3D view can be zoomed and rotated to get a better view of the robot arm. The buttons in the top-right side of the screen can disable the various graphical components in 3D view. The bottom button switches on/off the visualization of proximate boundary limits. The motion segments shown depend 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.

14.31 Structure Tab



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

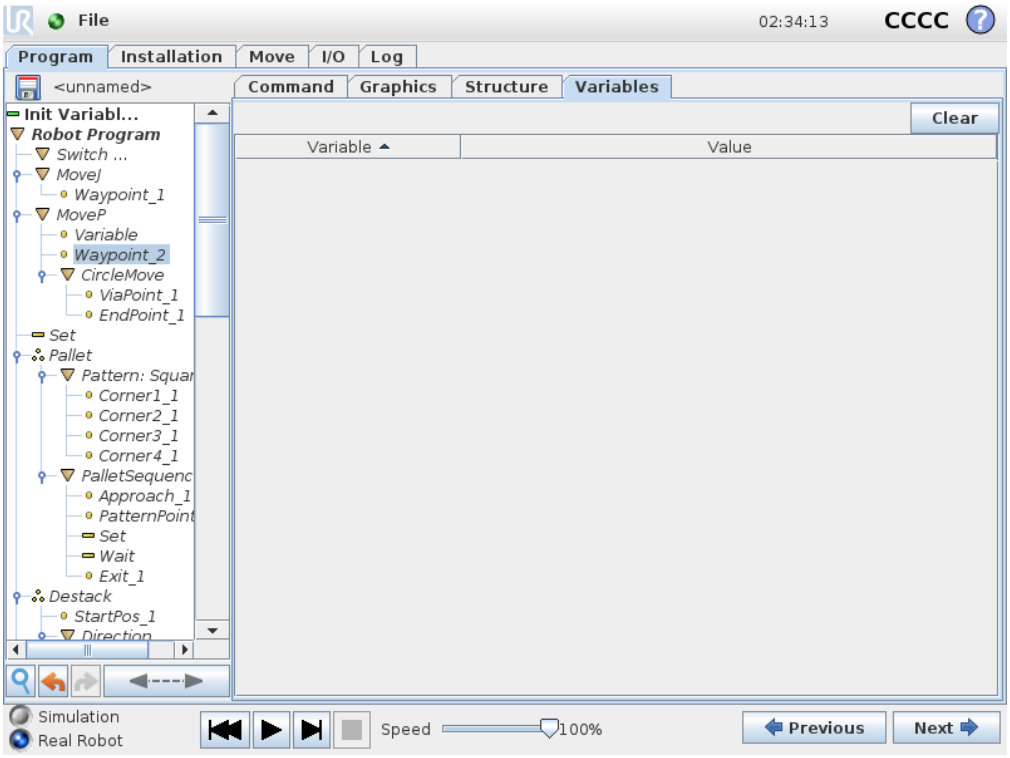
To insert new commands, perform the following steps:

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.

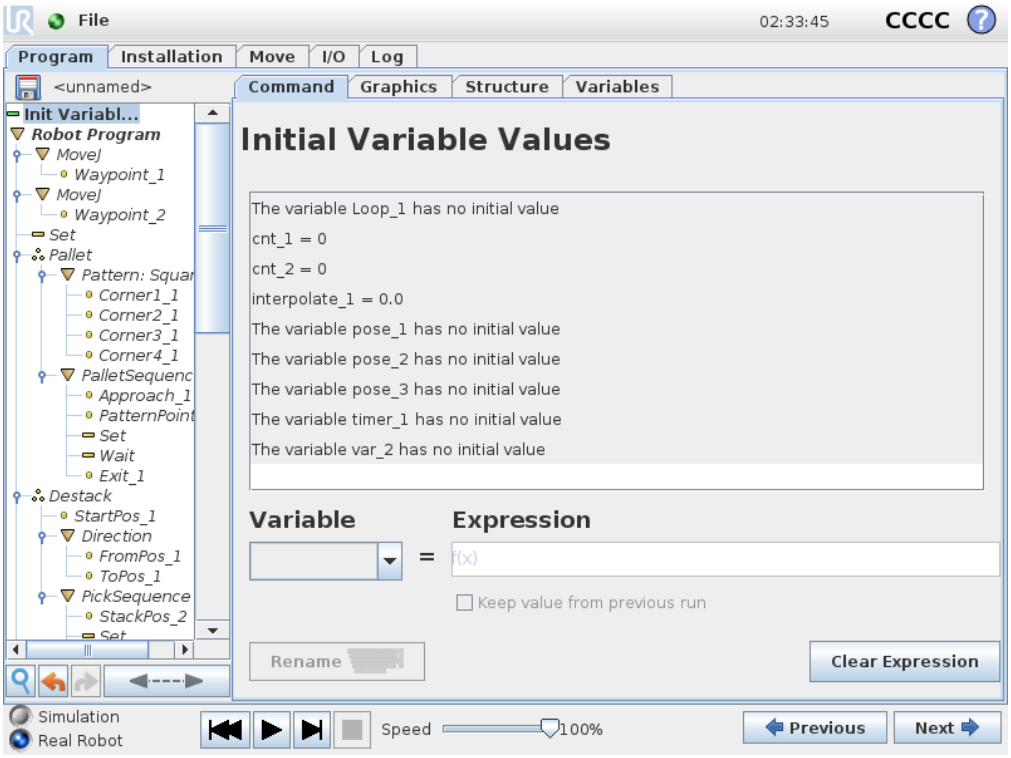
Not all commands fit at all places in a program. `Waypoints` must be under a Move command (not necessarily directly under). `ElseIf` and `Else` commands are required to be after an `If`. In general, moving `ElseIf` commands around can be messy. Variables must be assigned values before being used.

14.32 Variables Tab



The **Variables** tab shows the live values of variables in the running program, and keeps a list of variables and values between program runs. It only appears when it has information to display. The variables are ordered alphabetically by their names. 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.

14.33 Command: Variables Initialization



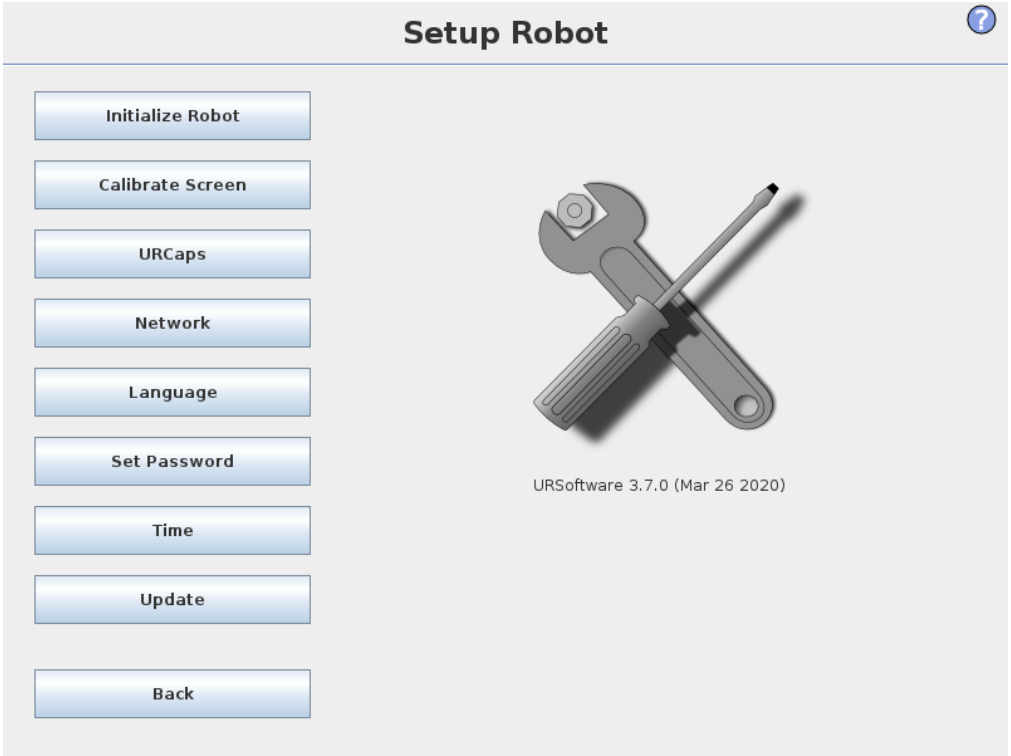
This screen allows setting variable values before the program (and any threads) start executing.

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 **Keep value from previous run** check-box is selected, the variable will be initialized to the value found on the **Variables** tab (see 14.32). 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.

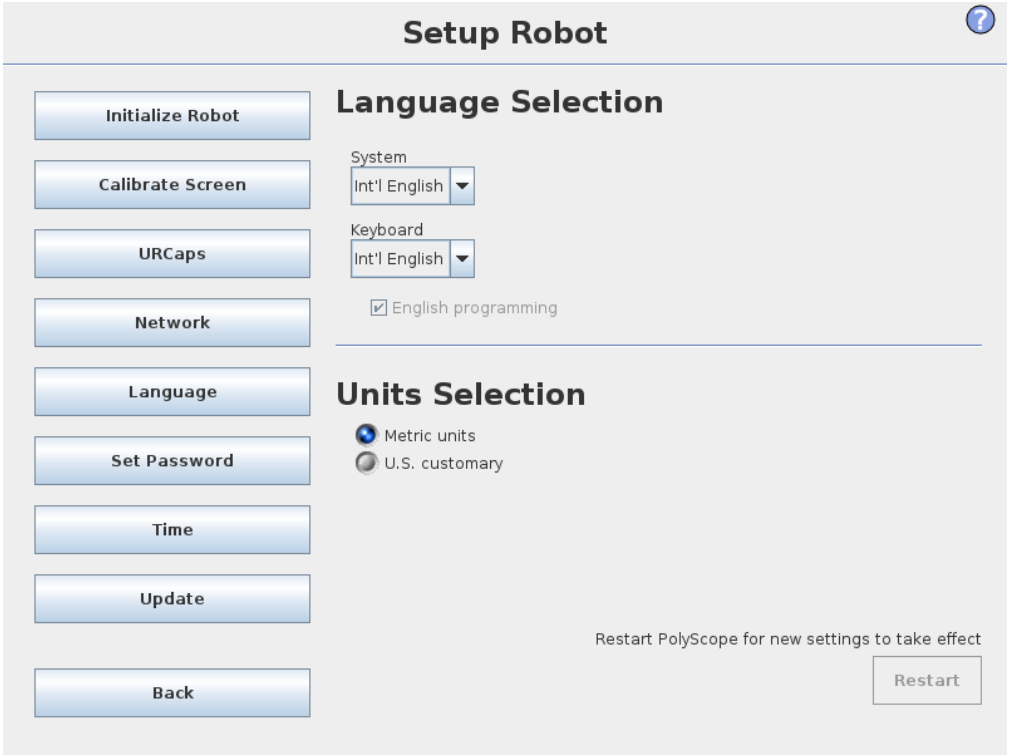
A variable can be deleted from the program by setting its name to blank (only spaces).

15 Setup Screen



- **Initialize Robot** Goes to the initialization screen, see 11.5.
- **Language and Units** Configure the language and units of measurements for the user interface, see 15.1.
- **Update Robot** Upgrades the robot software to a newer version, see 15.2.
- **Set Password** Provides the facility to lock the programming part of the robot to people without a password, see 15.3.
- **Calibrate Screen** Calibrates the "touch" of the touch screen, see 15.4.
- **Setup Network** Opens the interface for setting up the Ethernet network for the robot control box, see 15.5.
- **Set Time** Set the time and date for the system and configure the display formats for the clock, see 15.6.
- **URCaps Setup** Overview of installed URCaps as well as options for installation and uninstallation, see 15.7.
- **Back** Returns to the Welcome Screen.

15.1 Language and Units



Language, units and keyboard language used in PolyScope can be selected on this screen.

The selected language will be used for the text visible on the various screens of PolyScope as well as in the embedded help. Tick off “English programming” to have the names of commands within robot programs written in English. PolyScope needs to be restarted for changes to take effect.

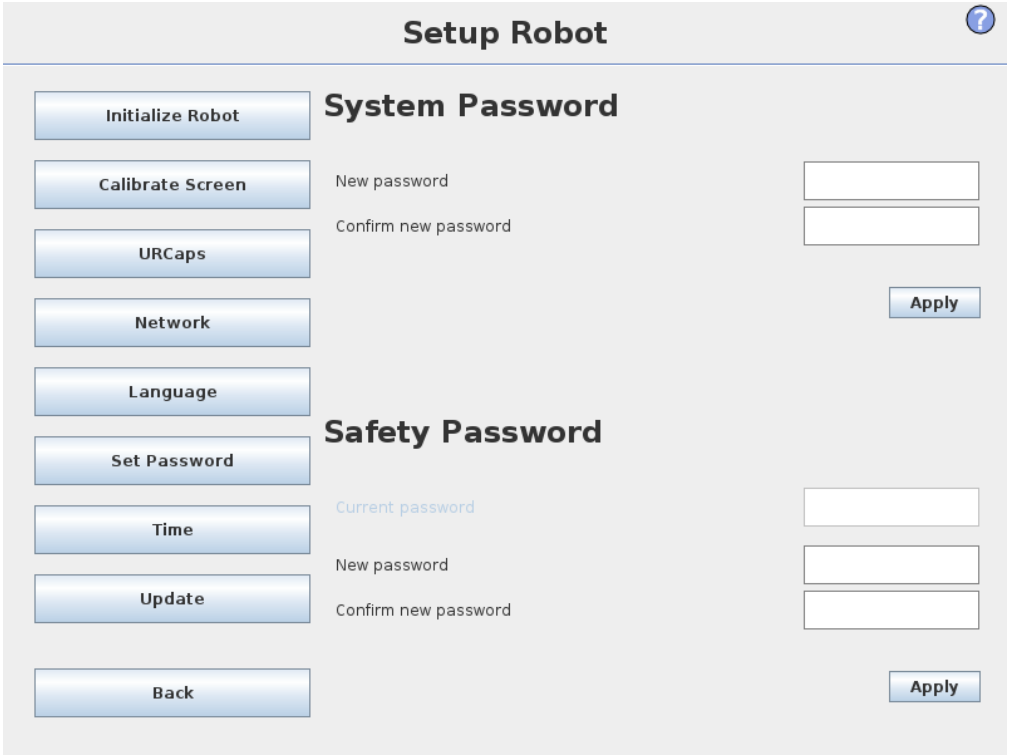
The selected keyboard language will be used in all pop-up keyboards in PolyScope.

15.2 Update 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.

WARNING:
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.

15.3 Set Password



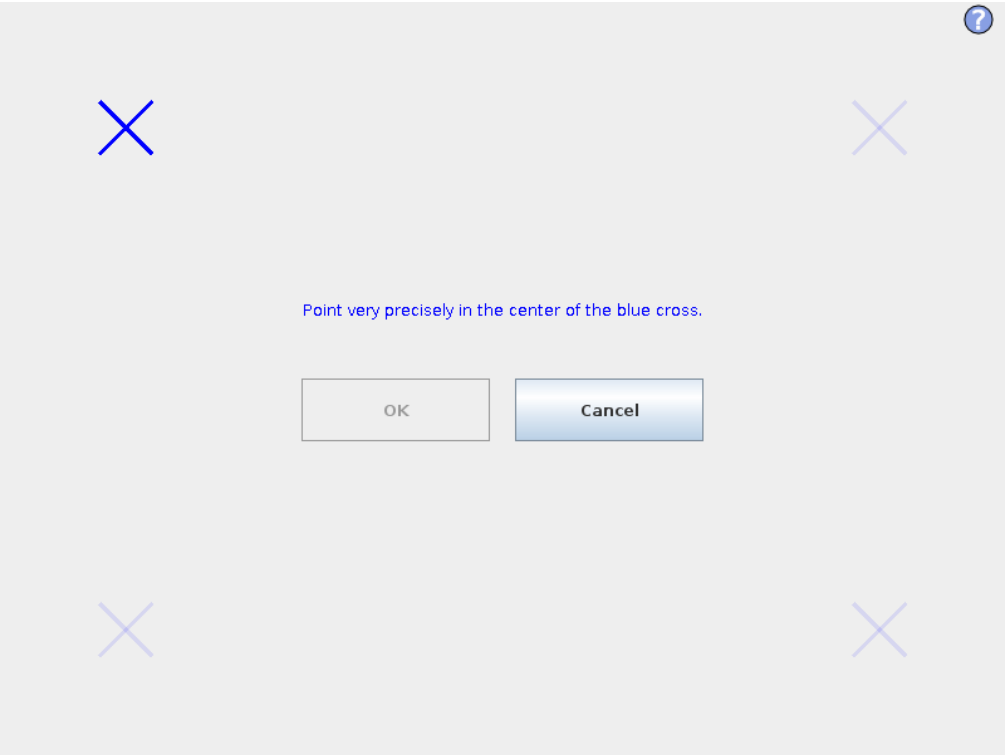
Two passwords are supported. The first is an *optional* System password which prevents unauthorized modification of the setup of the robot. When the System password is set, programs can be loaded and executed without the password, but the user must enter the correct password in order to create or change programs.

The second is a *required* Safety password which must be entered correctly in order to modify the safety configuration.

NOTE:
In order to change the safety configuration, the Safety password must be set.

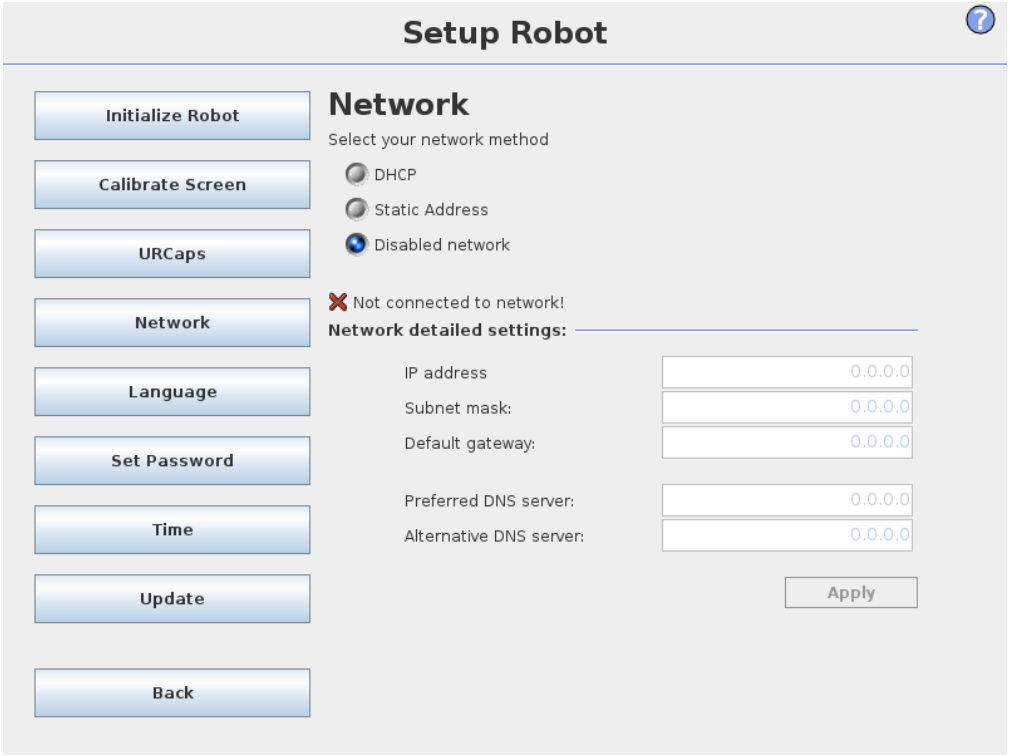
WARNING:
Add a System password to prevent non-authorized personnel from changing the robot installation.

15.4 Calibrate Screen



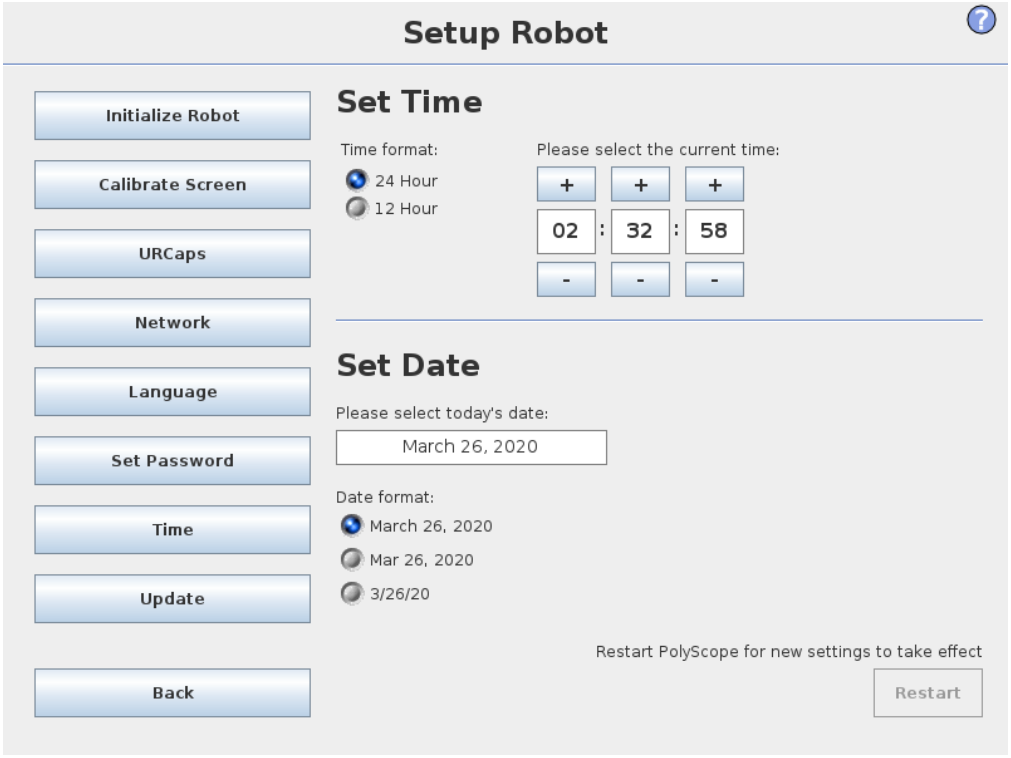
Calibrating the touch screen. Follow the on-screen instructions to calibrate the touch screen. Preferably use a pointed non-metallic object, such as a closed pen. Patience and care help achieve a better result.

15.5 Setup Network



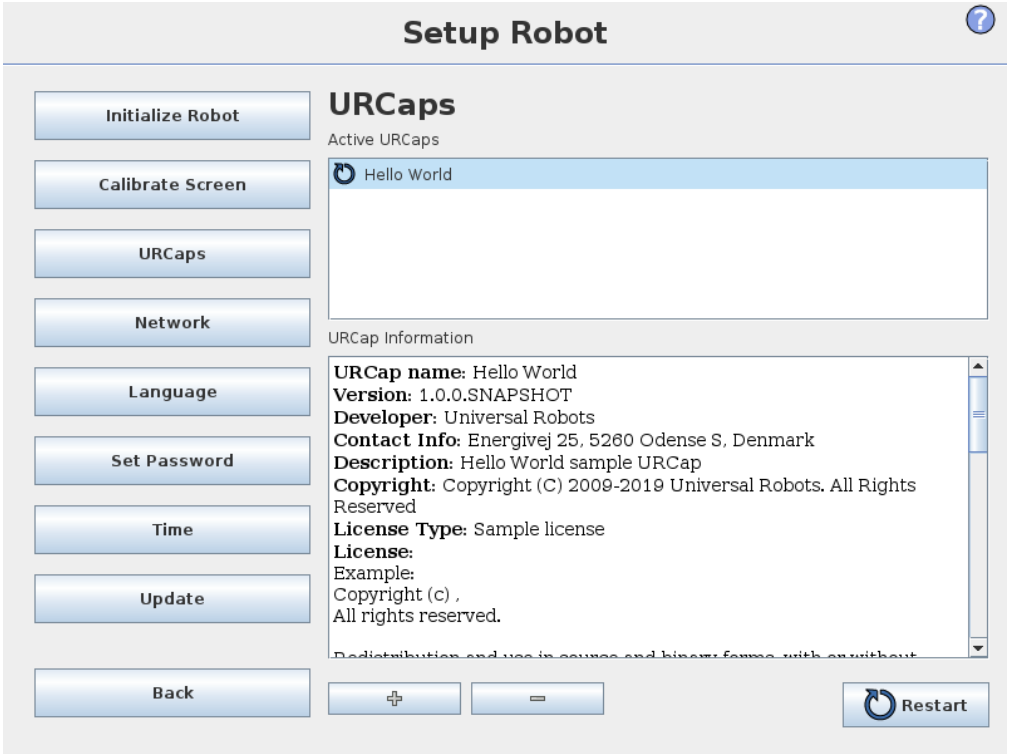
Panel for setting up the Ethernet network. An Ethernet connection is not necessary for the basic robot functions, and is disabled by default.

15.6 Set Time



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.

15.7 URCaps Setup



In the top list an overview of all installed *URCaps* is presented. Clicking on a URCap displays its meta information (including the name of the URCap, the version, license etc.) in the `URCap Information` area below the list.

Click the + button in the bottom of the screen to install a new URCap. A file chooser is shown where a `.urcap` file can be selected. Click `Open` and PolyScope will return to the setup screen. The selected URCap will be installed and a corresponding entry will appear in the list shortly after. Newly installed or uninstalled URCaps require PolyScope to be restarted and the `Restart` button will be enabled.

To uninstall a URCap, simply select the URCap in the list and click the − button. The URCap will disappear from the list, but a restart is still required.

In the list, the icon shown next to an entry indicates the state of the URCap. The different states are defined below:

- *URCap ok*: The URCap is installed and running normally.
- *URCap fault*: The URCap is installed but unable to start. Contact the URCap developer.
- *URCap restart needed*: The URCap has just been installed and a restart is required.