Updated: Monday, May 6th 2024

Version 2.0

# Robot Programming and Scene Assets in MachineLogic

robot-machinelogic

## Contents

# Introduction

This how-to guide covers how to program your collaborative robot using MachineLogic, Vention's cloud authoring tool for creating automation programs that can be deployed on the MachineMotion controller.
MachineLogic provides both a code-free programming interface, as well as a web editor exposing Vention's Python API. Both programming interfaces produce applications that can be simulated from your web browser, or deployed on the MachineMotion Controller.
For more information on Vention's Python API definition, click here.
For more information on Vention's MachineMotion controller, click here.
For more Information on programming using MachineLogic, click here.

# 1 - Robot Configuration

In order to create robot programs in MachineLogic, you must first design requires a supported robot, as well as a MachineMotion controller, to be placed in the 3D scene. The following list all robots that currently, or that will soon support Robot Programming in MachineLogic in MachineBuilder:

- UR3e
- UR5e
- UR10e
- UR16e
- UR20
- FANUC CRX-5iA
- FANUC CRX-10iA
- FANUC CRX-10iA/L
- FANUC CRX-20iA/L
- FANUC CRX-25iA

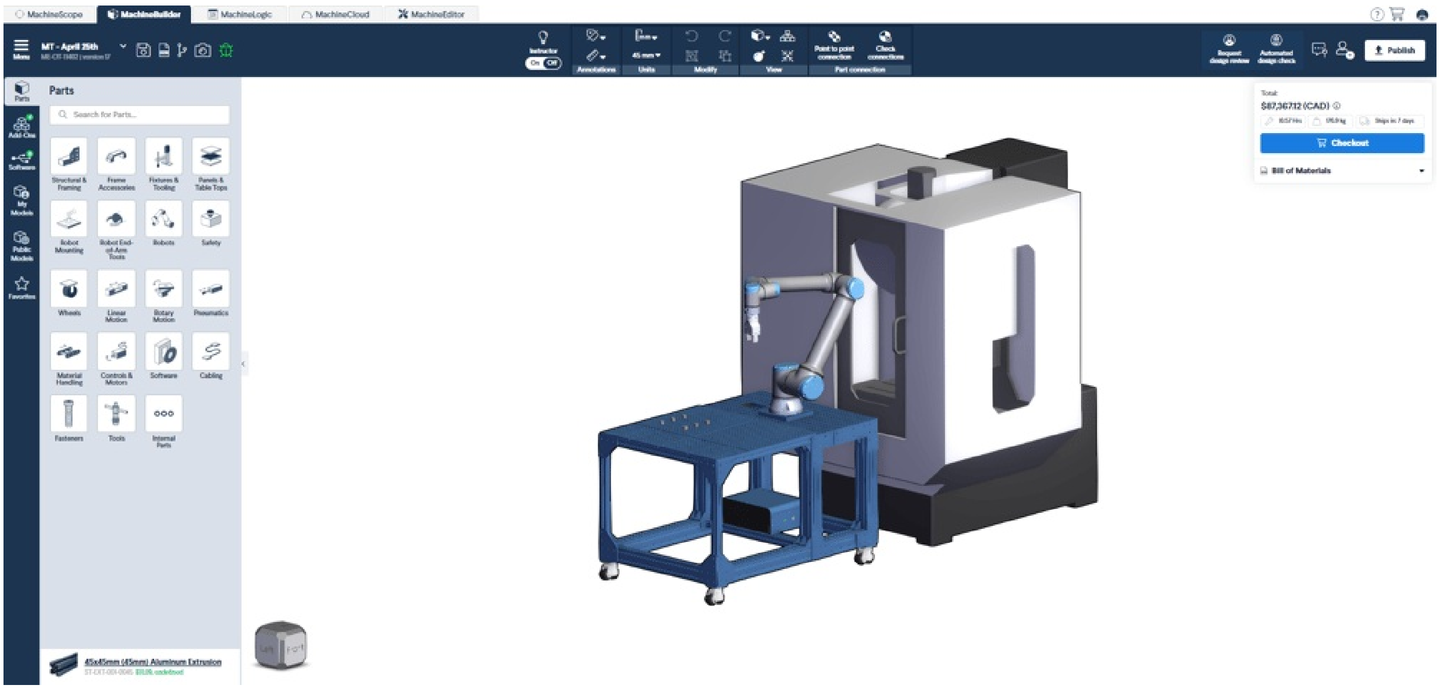
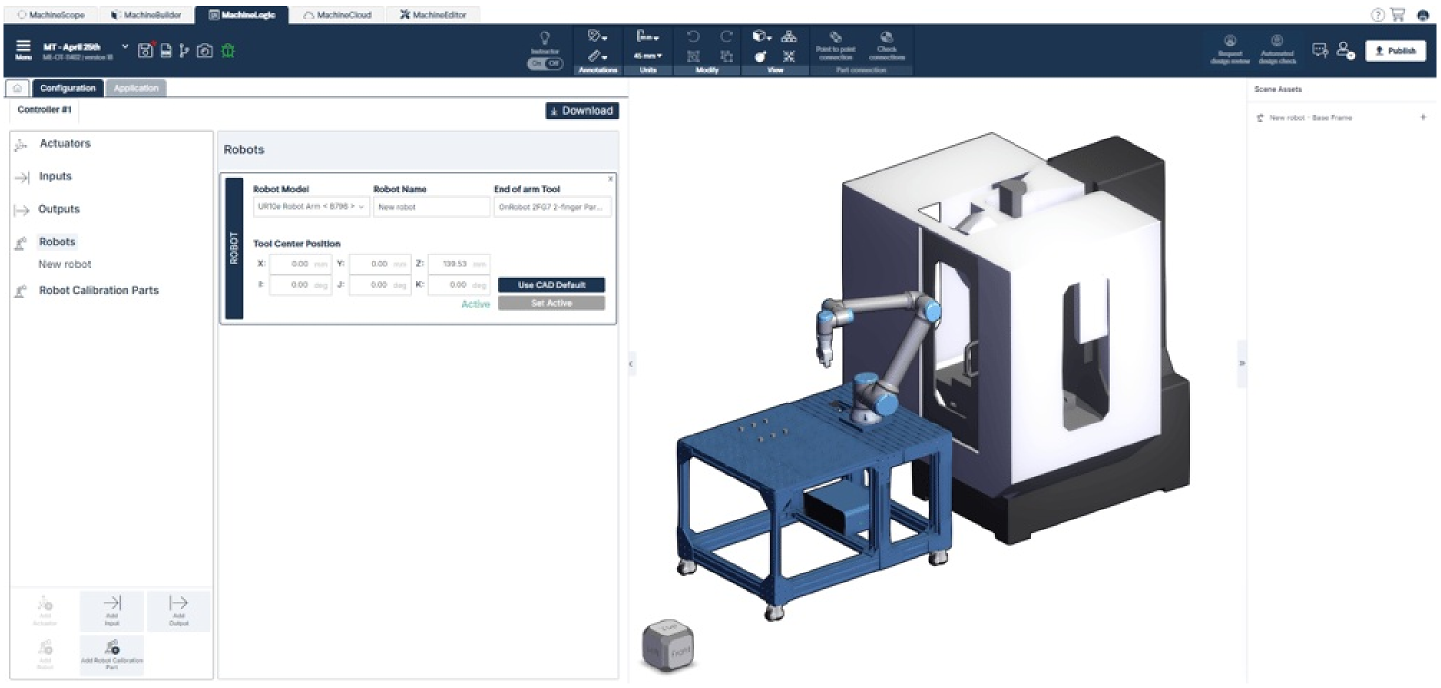


*Figure 1: Robot in MachineBuilder*

Click the MachineLogic tab to begin configuring the automated equipment.

1. From MachineLogic's Landing page, select the "**Configuration tab**"
2. Click "**Add Robot**" to add your robot configuration.
3. In the "**Robot Model**" dropdown menu, select the robot you would like to configure from your design. Please note that MachineLogic only supports a single Robot at the moment.
4. Once a robot model is selected and initialization is complete, you can input a friendly name to the robot and specify TCP values.

> **Tip:** TCP orientation dictates the pose of the robot when reaching a cartesian target. By default, the orientation of the Z-axis points directly outward from the robot flange



Robots

| Robot Model | Robot Name | End of arm Tool |
|---|---|---|
| UR10e Robot Arm < 8798 > | New robot | OnRobot 2FG7 2-finger Par... |

Tool Center Position

| | | | | | |
|---|---|---|---|---|---|
| X: | 0.00 mm | Y: | 0.00 mm | Z: | 139.53 mm |
| I: | 0.00 deg | J: | 0.00 deg | K: | 0.00 deg |

Active

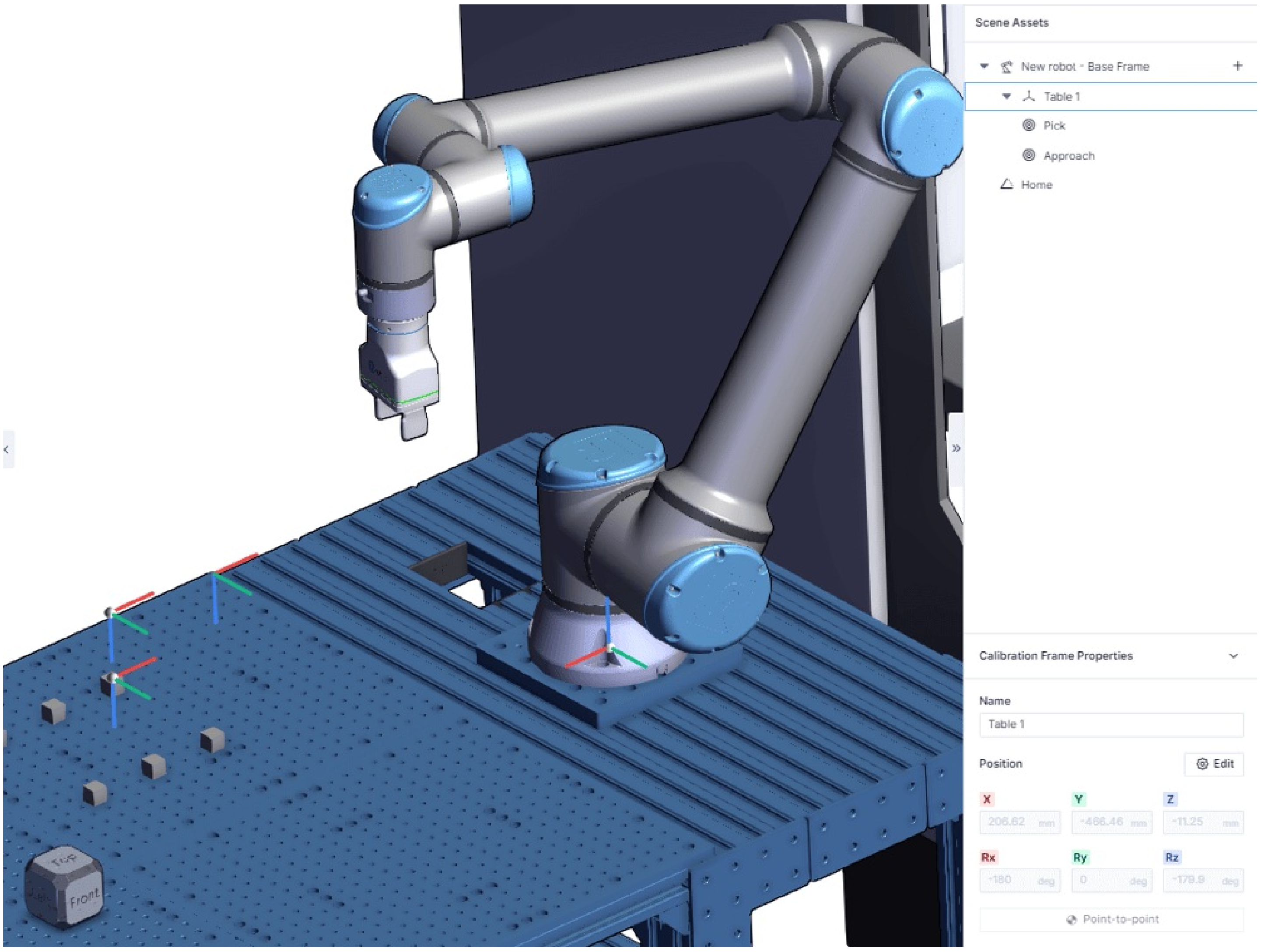
*Figure 2: Robot configuration in MachineLogic*

## 2 - Scene Assets Pane

The sections below outline all robot functionality available for you to program your next robot application.
To learn more about the basics of MachineLogic's code-free programming before getting started, click here.

The Scene Assets side-pane allows you to create assets useful for programming robot motion. These assets consists of Frames and Targets that can be reused within an application (Application-Level Scope) or across different applications (Design-Level Scope).

The Scene Assets are nested in a tree-view structure where children asset's positions and orientations are always defined relative to their parents.



Calibration Frame Properties

Name: Table 1

Position

| X | Y | Z |
|---|---|---|
| 206.62 mm | -466.46 mm | -11.25 mm |

| Rx | Ry | Rz |
|---|---|---|
| -180 deg | 0 deg | -179.9 deg |

***Figure 3: Robot in MachineLogic with Scene Assets showing***

## 2.1 - Scene Assets Type

The available assets are:

- **Calibration Frame** : A reference frame defined relative to the Robot Base Frame. The position of the Calibration Frame is nominally defined in the scene and is expected to be taught in the physical world. This frame can be used to define a coordinate system from which a set of cartesian targets are defined.
- **Reference Frame** : A reference frame defined relative to the calibration frame. This frame can be used to define a coordinate system from which a set of cartesian targets are defined.
- **Cartesian Target** : A target that the robot will attempt to reach when it is called in the Robot Move instruction. This target is defined in relation to its parents position and orientation.
- **Joint Target** : A target that the robot will attempt to reach when it is called in the Robot Move instruction. This target is defined in relation to 6 joint values.

**Tip:** When inserting Cartesian Targets, be mindful of its orientation. The direction of the Z-axis should match between Cartesian Targets and the defined TCP.

## 2.2 - Inserting new Scene Assets

Inserting new Scene Assets can be done either by selecting the “Plus” icon at the left of the Robot node, or through a context menu opened when right-clicking an existing scene asset.

### 2.2.1 - inserting a scene asset using the “Plus” Icon

Selecting the “Plus” Icon opens up the Asset’s menu. From this menu, the available Scene Assets are displayed. Note that the insertion of a Calibration Frame is required in order to insert Reference Frames or Cartesian Targets.
If an application is opened and the Asset’s menu is selected, Application level assets will be accessible.

Application level assets can only be reused within the current application. Design-Level Assets are accessible to all applications contained within the design. Calibration Frame are strictly Design-Level assets.

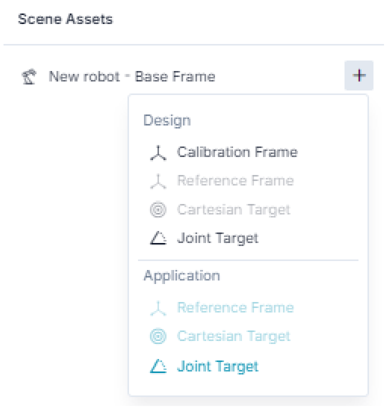


*Figure 4: Inserting Scene Assets*

### 2.2.2 - inserting a scene asset using the context menu

It is also possible to insert new Scene Assets through a context menu opened when right-clicking either an existing Calibration Frame or Reference Frame. When doing so, the context menu will display the possible child assets that can be inserted under the currently selected Calibration or Reference Frame.

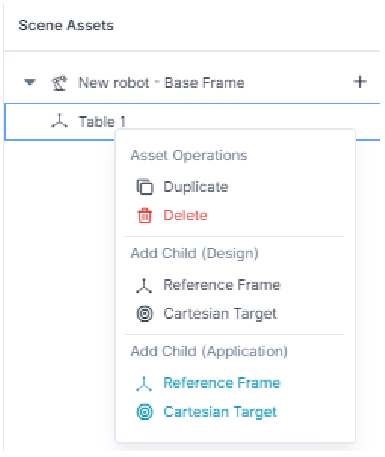


*Figure 5: Inserting Scene Assets with context menu*

## 2.3 - The Properties Pane

The Properties Pane is displayed when an Asset has been selected in the Tree. It displays the editable values of the selected asset and changes the view mode of the corresponding scene gizmo.

**Calibration Frame Properties** :

- Name: User-defined friendly name of the Asset
- Position: the position and orientation as defined with respect to the Asset's parent.

**Reference Frame Properties** :

- Name: User-defined friendly name of the Asset
- Parent reference frame: the parent of the selected asset, changing the parent will update the tree-view accordingly.
- Position: the position and orientation as defined with respect to the Asset's parent.

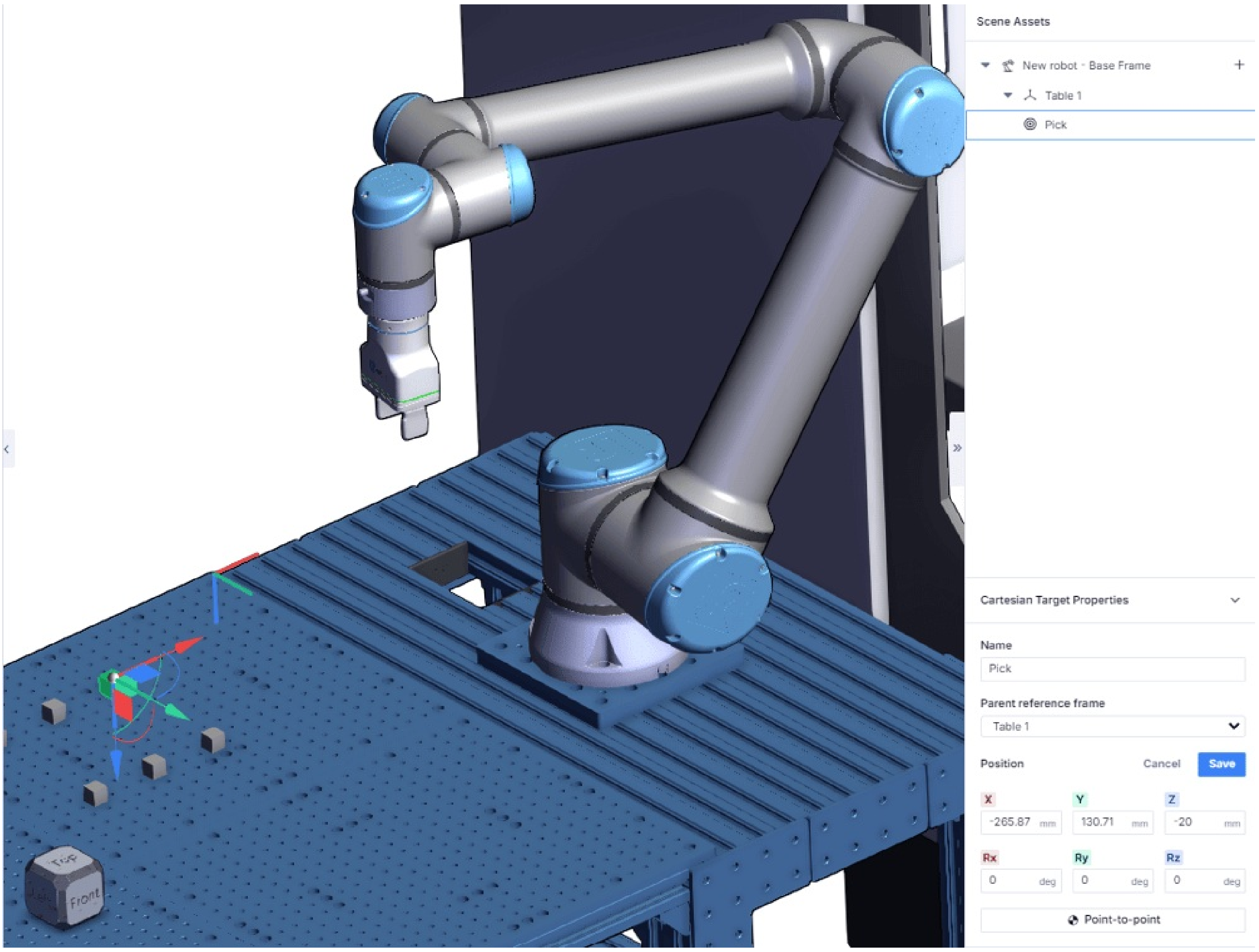
**Cartesian Target Properties** :

- Name: User-defined friendly name of the Asset
- Parent reference frame: the parent of the selected asset, changing the parent will update the tree-view accordingly.
- Position: the position and orientation as defined with respect to the Asset's parent.

**Joint Target Properties** :

- Name: User-defined friendly name of the Asset
- Joint Angles: the individual joint values of each of the robot's joints.

> **Note:** Position Values labeled Rx, Ry and RZ are described using Euler Angles



Cartesian Target Properties

Name: Pick

Parent reference frame: Table 1

Position

| X | Y | Z |
|---|---|---|
| -265.87 mm | 130.71 mm | -20 mm |

| Rx | Ry | Rz |
|---|---|---|
| 0 deg | 0 deg | 0 deg |

*Figure 6: Properties of a Cartesian Target*

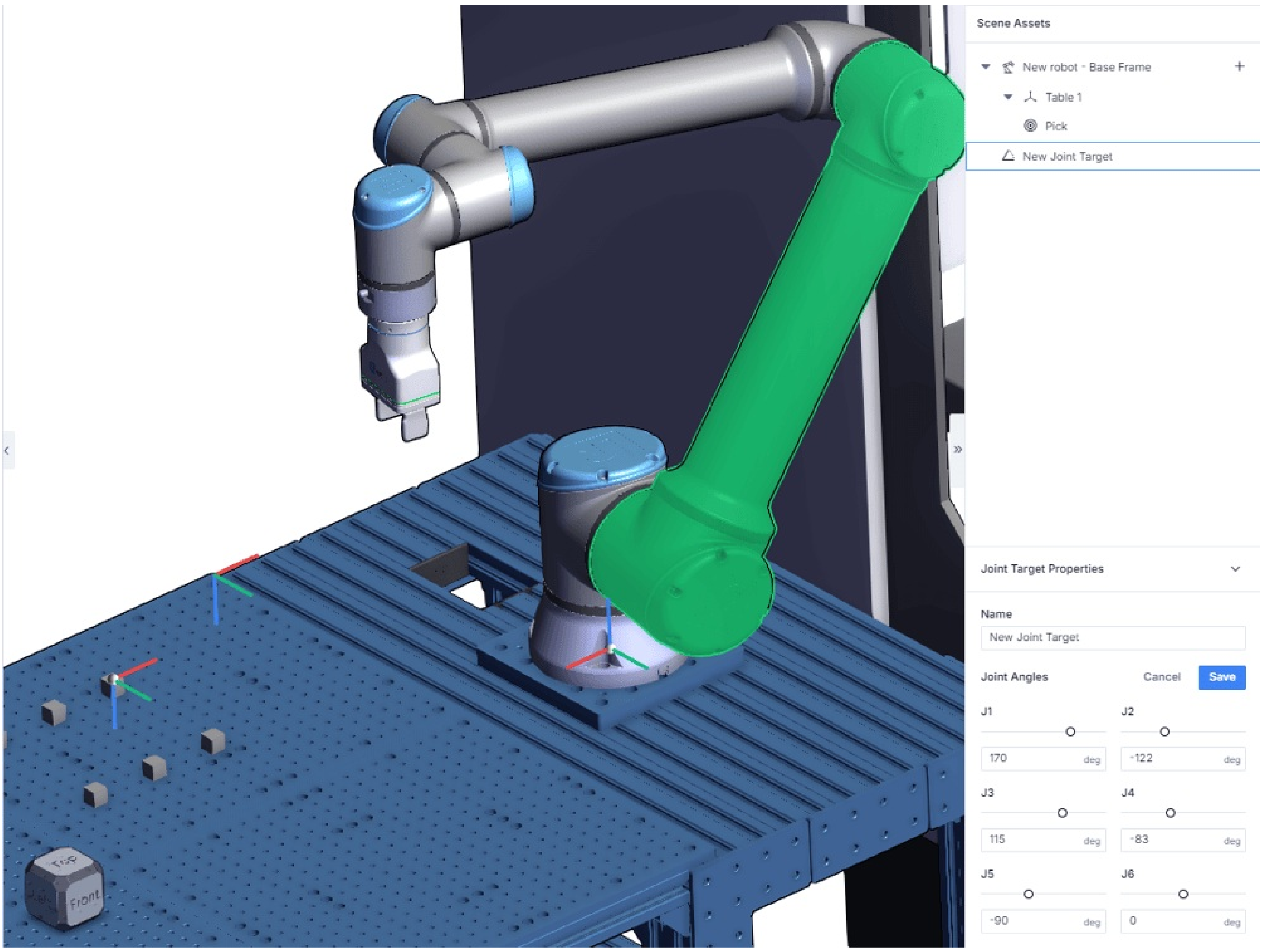


*Figure 7: Properties of a Cartesian Target*

## 2.4 - Editing the Scene Asset's Position (Edit mode).

To edit the position of an asset, the edit button must be selected from the asset's properties.

When in Edit mode, there are three ways to change the asset's position:

**Dragging the Triad in the Scene** : When in edit mode, The asset's gizmo will morph into a triad. The triad allows for linear and planar translation and elemental rotations. The position fields will update to show the asset's new positions.

**Point-to-Point** : the point-to-point feature automatically updates the position of the asset in accordance with the position of the cursor when clicking in the scene.

**Input numerical values** : It is possible to change the asset's position by inputting numerical values directly in the position fields.

> **Tip:** When using point-to-point, the Z-axis of the inserted scene asset will point downwards to match with default TCP values.

## 2.5 - Asset Operation (Context menu)

Asset operations are accessed from the context-menu and contain **Duplicate** and **Delete** Functionalities.

**Duplicate** : When duplicating a Parent, the children's assets will be duplicated as well.

**Delete** : When deleting a Parent, the children will be deleted as well.

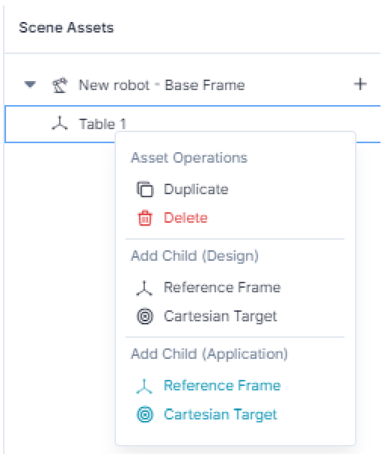


*Figure 8: Context menu showing Asset Operations*

# 3 - Robot Move Command

To create Robot Motion, Cartesian and Joint targets defined in the Scene-Assets pane must be called in the **Robot Move instruction**

## 3.1 - Inserting a Robot Move Command

The Robot Move Command can be inserted from the add command menu as shown below:

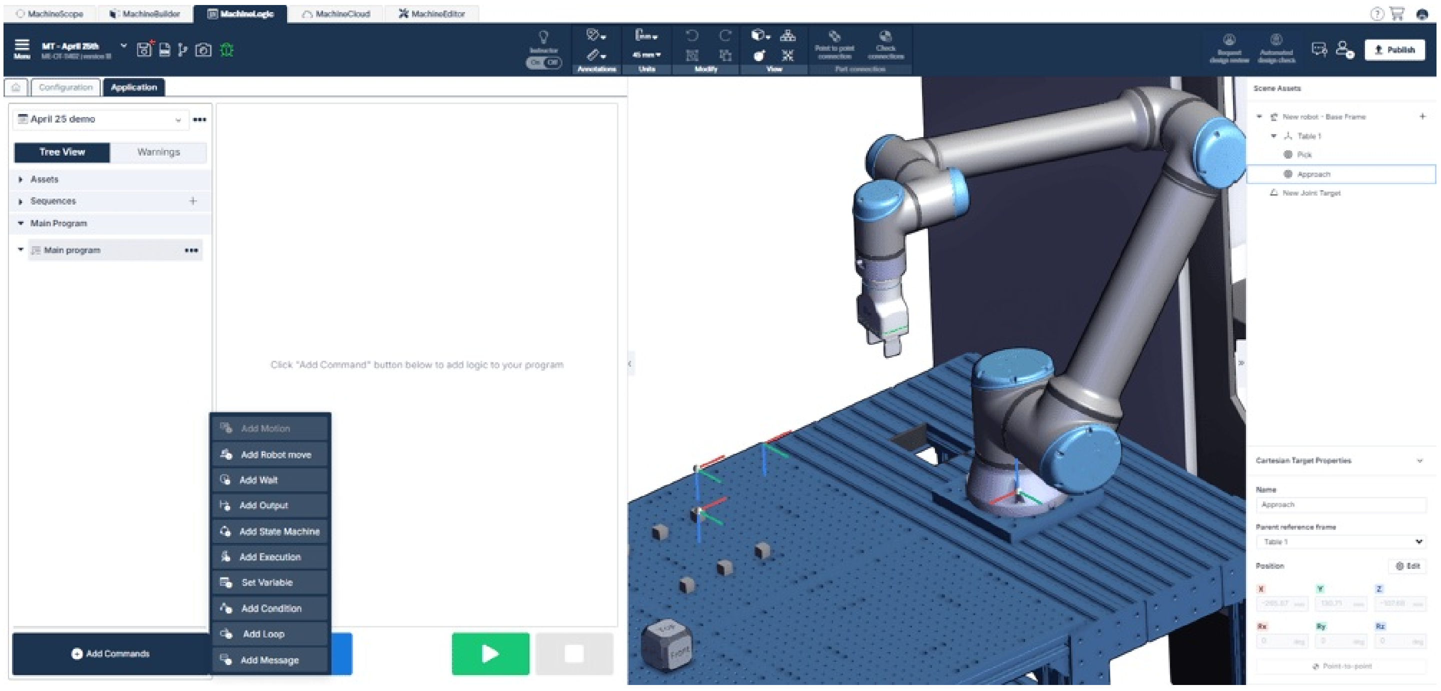
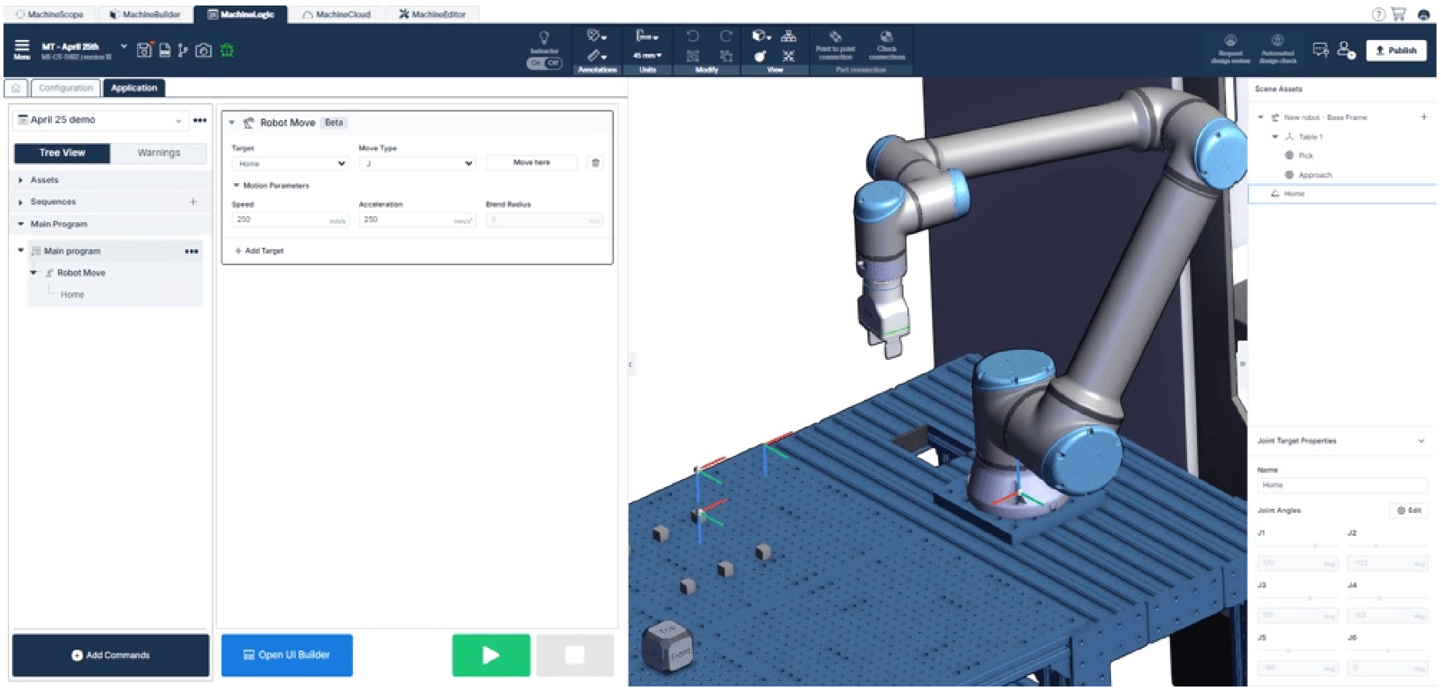


*Figure 9: Add Robot Move Instruction*

## 3.2 - Adding Targets and configuring the Robot Move Command

Once the Robot Move command has been added to a sequence select the **add target** button to access the motion parameters. multiple targets can be appended to the Robot Move command.


Robot Move Beta

Target: Home

Move Type: J

Move here

Motion Parameters

Speed: 250 mm/s

Acceleration: 250 mm/s²

Blend Radius: 0 mm

Add Target

*Figure 10: Adding targets to the Robot Move Instruction*

Once a target is selected, the following parameters will appear:

- **Target** : This drop-down field displays all available targets as defined in the Scene Assets pane.
- **Move Type**: Specifies the type of motion to be used to reach the target. Move type can be either Move J or Move L. Move L will constraint the motion to follow a linear path.

  **Tip:** As Move L imposes a linear path, it can be more challenging to find a valid solution for a Move L trajectory in opposition to a Move J. In situations where Move L is failing, try changing the move type to Move J.

- **Move Here**: This button can be used to visualize the resulting robot pose when reaching the current Target.
- **Speed**: The desired maximum TCP speed defined in mm/s.
- **Acceleration**: The desired maximum TCP acceleration defined in mm/s2
- **Blend Radius**: The Blend Radius parameter can be used to program continuous motion of the robot arm. The parameter defines a tolerance value for the path to deviate from the current target in order to conserve speed towards the next target to reach.

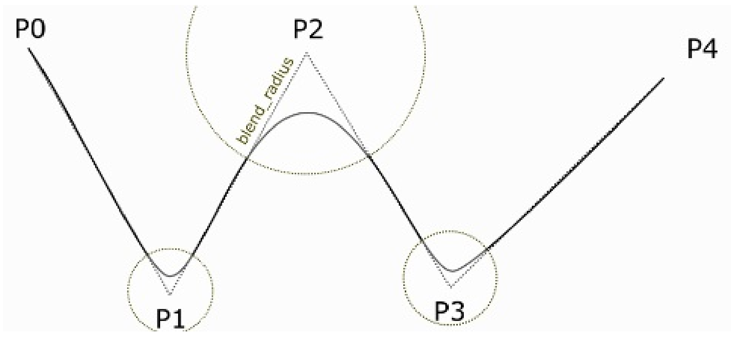


*Figure 11: Blend Radius*

# 4 - Robot State Reporting & Troubleshooting

Over the course of the robot's physical programming and operation, you may encounter certain robot states and errors. The sections below outlines those states, as well as how to troubleshoot them in the event of an error occurring. The sections will also cover best practices for programming your robot with MachineLogic.

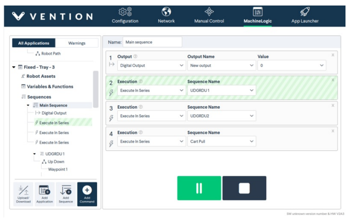
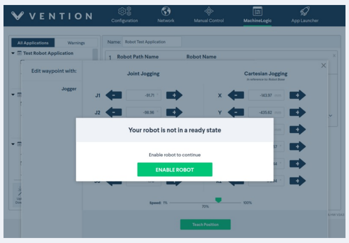
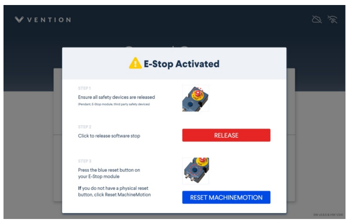
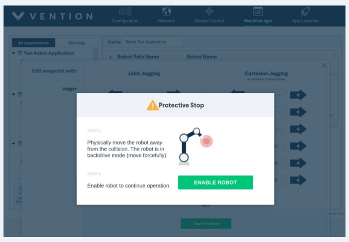
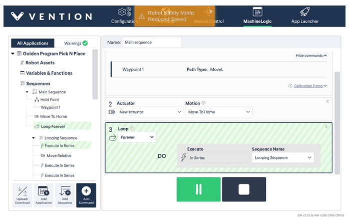
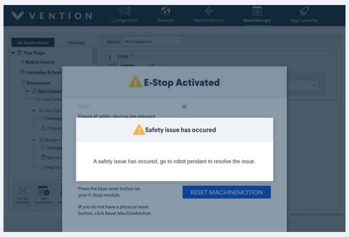
**Notes:**

- Robot Programming in MachineLogic is currently **not compatible** with Multi-Controller systems.

- Pausing a running robot program **will not** pause the Robot Motion, but rather pause the robot path command. Be careful when pausing and immediately approaching your robot cell.

## 4.1 - Robot States

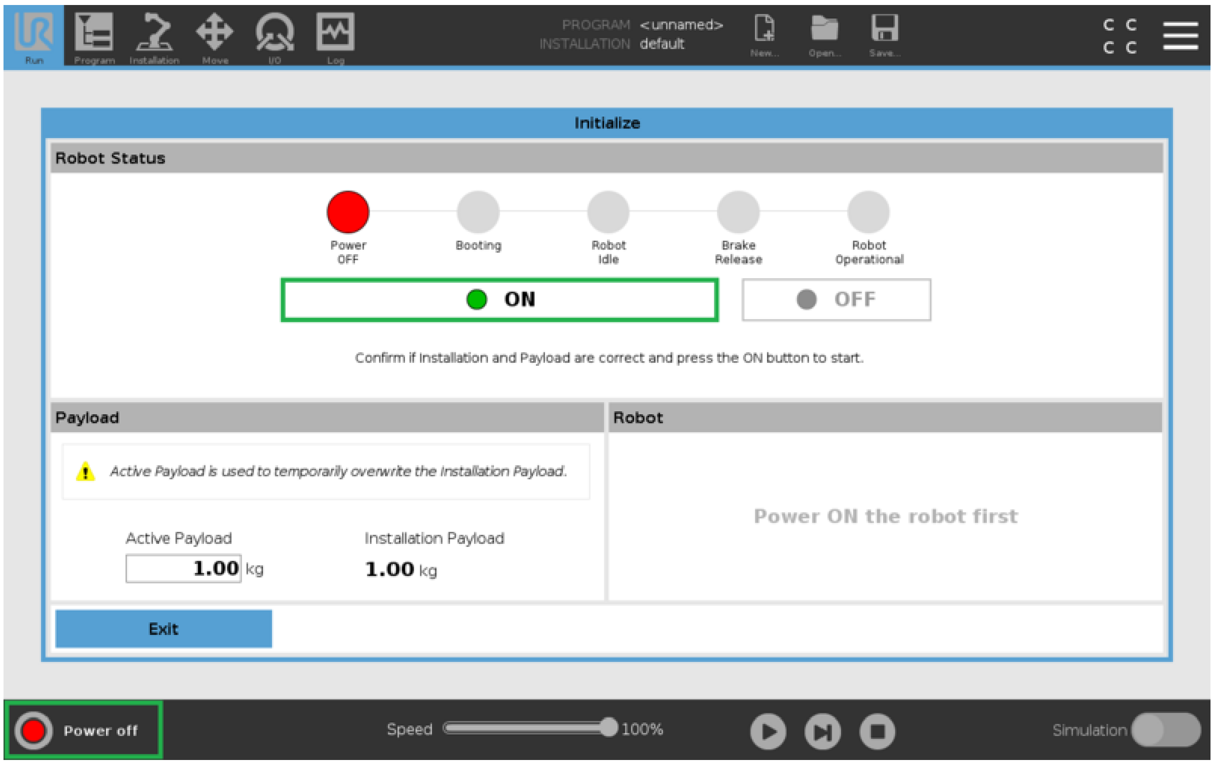
The table below outlines the various states the robot may find itself in, how it may have gotten into that state, and how to get out of it and return to normal operation.

| State | Description | Screenshot |
|---|---|---|
| **Ready** | All is good and functional. No problems or errors present. |  |
| **Not Ready** | Occurs after a reset or fault clearing event, usually brought on by an Emergency Stop or Fault state, and trying to move the robot or play a program. To return to a **Ready** state, select **Enable Robot**. |  Your robot is not in a ready state<br>Enable robot to continue<br>ENABLE ROBOT |
| **Emergency Stop** | Occurs when a physical emergency stop button is pressed. Your MachineLogic program is stopped. To return to normal operation, release all physical (and software) emergency stop buttons and select **RESET**. If the reset procedure succeeds, your robot is now in a **Not Ready** state. |  E-Stop Activated<br>Ensure all safety devices are released<br>Click to release software stop<br>RELEASE<br>Press the blue reset button on your E-Stop module<br>If you do not have a physical reset button, click Reset MachineMotion<br>RESET MACHINEMOTION |
| **Protective Stop** | Occurs when the robot collides with an external object or person. Your MachineLogic program is stopped. In this state, the robot is back-drivable. Move the robot away from any possible collision and select **Enable Robot**. The robot should now be in a **Ready** state<br>**Note:** An incorrectly set or overweight payload may trigger a Protective Stop. To re-enable the robot without retriggering the fault, manually support (lightly push up) at the robot's flange before enabling. |  Protective Stop<br>Physically move the robot away from the collision. The robot is in backdrive mode (move forcefully).<br>Enable robot to continue operation.<br>ENABLE ROBOT |
| **Reduced Mode** | Occurs when a safety device wired into the reduced pin on the robot controller is triggered (ex: walking into an area guarded by an area scanner). Your MachineLogic program continues, but at a slower speed. It returns to full speed once the triggering fault is cleared. |  |
| **Fault** | Occurs when a variation of errors not captured by the emergency or protective stop arise. Your MachineLogic program is stopped. To return to normal operation, the fault must be cleared on the robot's native pendant before continuing. Once the fault is cleared, the robot is in a **Not Ready** state. |  E-Stop Activated<br>Safety issue has occured<br>A safety issue has occured, go to robot pendant to resolve the issue.<br>RESET MACHINEMOTION |

**Note**: MachineMotion does not support **Safeguard Stop**. Consequently, any Safeguard Stop event triggers an Emergency Stop, and that flow must be used to return to normal.

### 4.1.1 Re-Enable from Robot Pendant

In certain cases (ex: heavy safety breach), the robot may required to be re-enabled from the robot controller's pendant. To get your robot back into an enabled state, simply select the **State Indicator** in the bottom left (see Figure 15a below). The robot may be completely powered down, start by powering it on (Figure 15a) and then enable it (Figure 15b). If the robot is already in an **Idle** state, enable it (Figure 15b).



Confirm if Installation and Payload are correct and press the ON button to start.

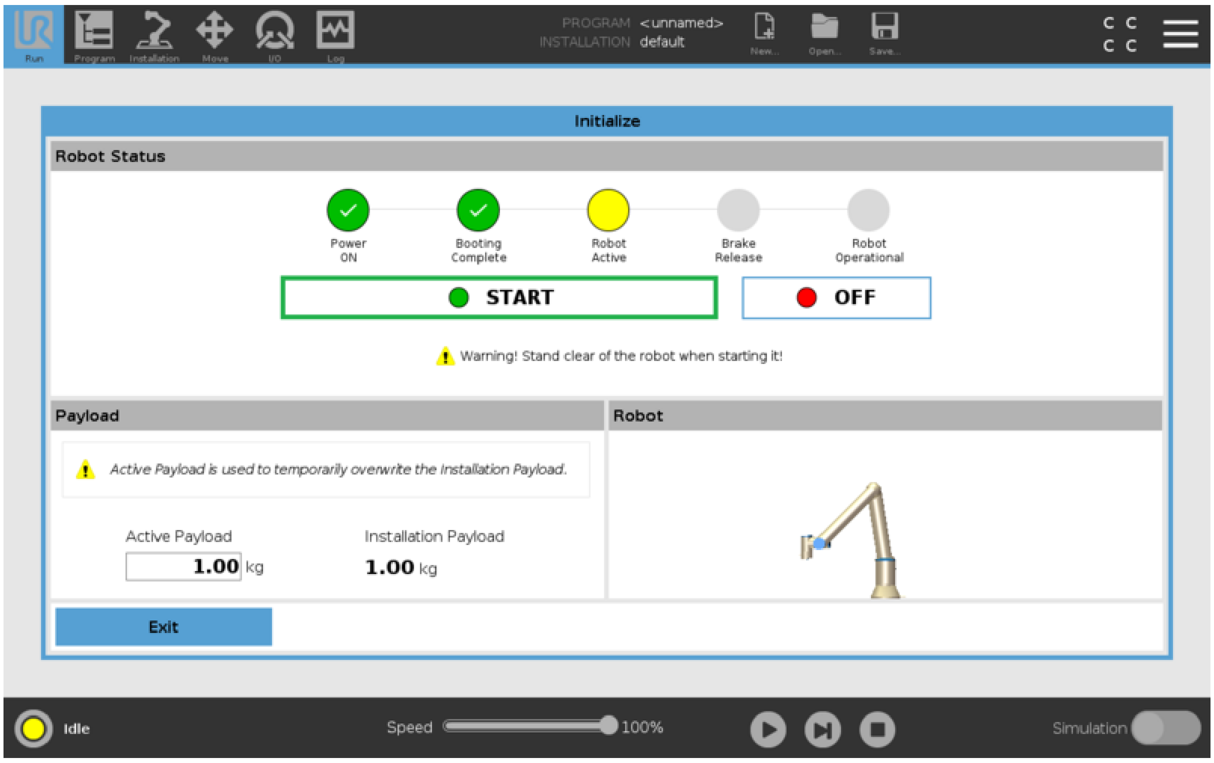
Active Payload is used to temporarily overwrite the Installation Payload.

Active Payload 1.00 kg

Installation Payload 1.00 kg

Power ON the robot first

*Figure 15a: Power on Robot*



Warning! Stand clear of the robot when starting it!

Active Payload is used to temporarily overwrite the Installation Payload.

Active Payload 1.00 kg

Installation Payload 1.00 kg

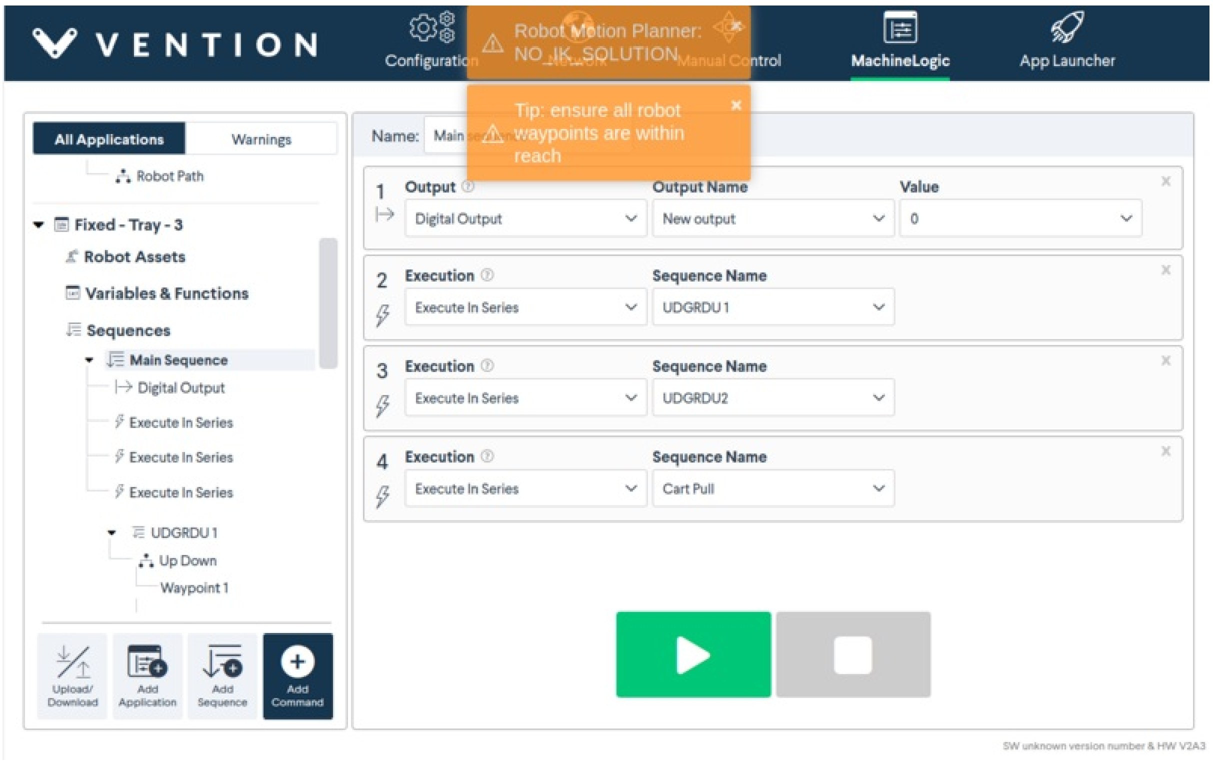
*Figure 15b: Enable Robot*

## 4.2 - Motion Planner Errors

The table below highlights some of the most common errors you may encounter when trying to move your robot and plan its path.

| Error | Error Code | Error Category | Description and Tips |
|---|---|---|---|
| PLANNING_FAILED | -1 | Overall Behaviour | This error usually occurs when certain robot movements exceed the limits set in the planner. This can be fixed by reducing the speed of the robot. |
| INVALID_MOTION_PLAN | -2 | Overall Behaviour | This error usually occurs when a robot path (generally linear - MoveL) will collide with the robot, trying to go through it to achieve the next target. |
| NO_IK_SOLUTION | -31 | Kinematics Errors | This error usually occurs when a waypoint is unatteinable in the current solution space. Try moving the targetcloser, within reach of the robot. Verify that the target is also possible with the current joint placement |

Motion planner errors, along with tips, will appear at the top of the Control Center.


Robot Motion Planner: NO_IK_SOLUTION

Tip: ensure all robot waypoints are within reach

*Figure 16: Motion Planner Error*

The table below highlights less common errors and their respective category.

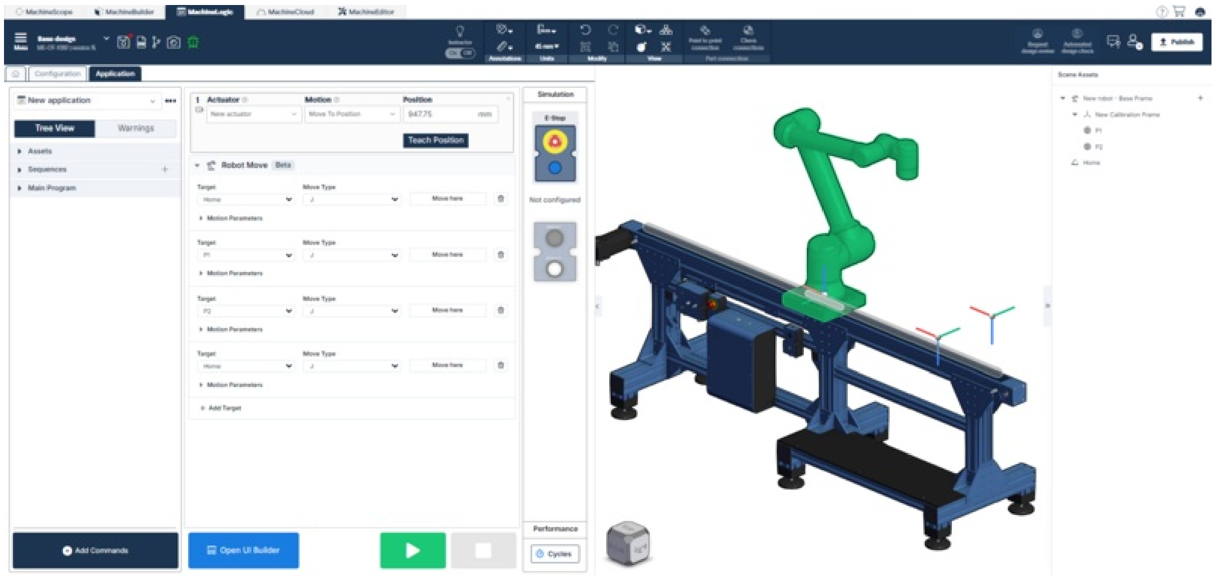
| Error | Error Code | Error Category |
|---|---|---|
| MOTION_PLAN_INVALIDATED_BY_ENVIRONMENT_CHANGE | -3 | Overall Behaviour |
| CONTROL_FAILED | -4 | Overall Behaviour |
| UNABLE_TO_AQUIRE_SENSOR_DATA | -5 | Overall Behaviour |
| TIMED_OUT | -6 | Overall Behaviour |

| Error | Error Code | Error Category |
|---|---|---|
| PREEMPTED | -7 | Overall Behaviour |
| START_STATE_IN_COLLISION | -10 | Planning and Kinematics Request Errors |
| START_STATE_VIOLATES_PATH_CONSTRAINTS | -11 | Planning and Kinematics Request Errors |
| GOAL_IN_COLLISION | -12 | Planning and Kinematics Request Errors |
| GOAL_VIOLATES_PATH_CONSTRAINTS | -13 | Planning and Kinematics Request Errors |
| GOAL_CONSTRAINTS_VIOLATED | -14 | Planning and Kinematics Request Errors |
| INVALID_GROUP_NAME | -15 | Planning and Kinematics Request Errors |
| INVALID_GOAL_CONSTRAINTS | -16 | Planning and Kinematics Request Errors |
| INVALID_ROBOT_STATE | -17 | Planning and Kinematics Request Errors |
| INVALID_LINK_NAME | -18 | Planning and Kinematics Request Errors |
| INVALID_OBJECT_NAME | -19 | Planning and Kinematics Request Errors |
| FRAME_TRANSFORM_FAILURE | -21 | System Errors |
| COLLISION_CHECKING_UNAVAILABLE | -22 | System Errors |
| ROBOT_STATE_STALE | -23 | System Errors |
| SENSOR_INFO_STALE | -24 | System Errors |

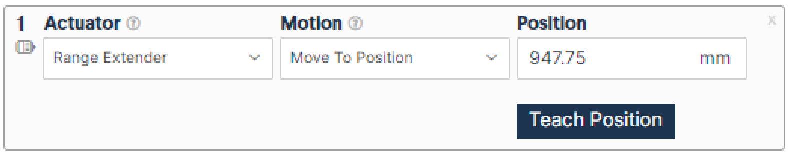
## 5.0 - Programming with a Range Extender (7th Axis)

When using a range extender in conjunction with Cartesian Targets, a few extra steps are required to make sure the robot can reach each positions:

1. in the MachineBuilder scene, move the robot to the desired positions by double-clicking on the mounting plate and dragging the cursor along the travel length of the actuator:



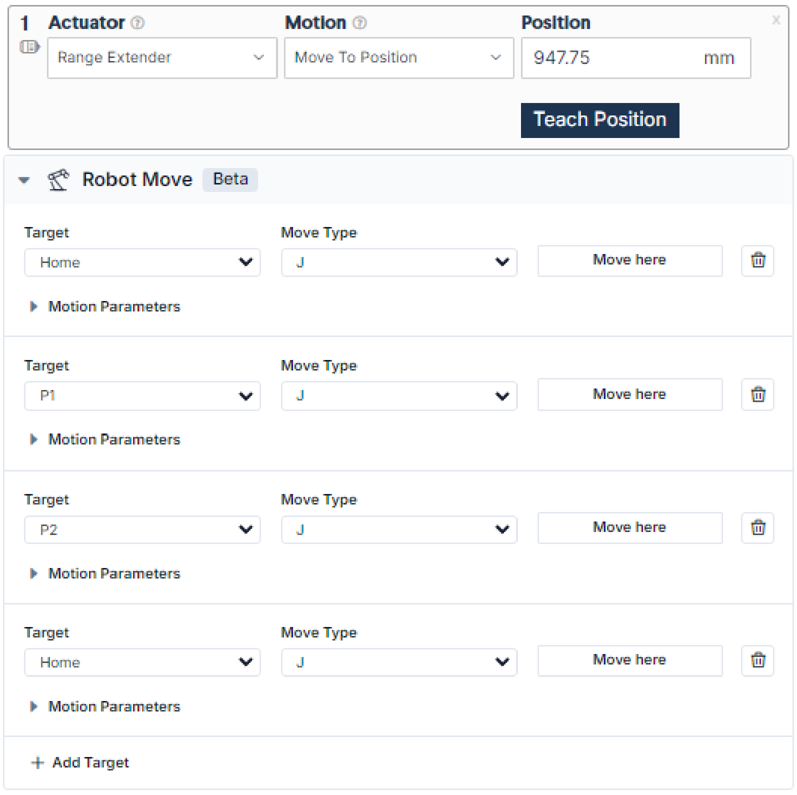
2. in the MachineLogic program insert **Move To Position** command an use the **Teach Position** button to record the current position of the range extender:



1 Actuator | Motion | Position
Range Extender | Move To Position | 947.75 mm

Teach Position

3. From the Scene Asset pane, insert new Cartesian Targets while the robot is at the desired actuator position
4. In the MachineLogic sequence, make sure to insert the **Move To Position** command immediately before the Robot Move instruction to make sure the robot is in the desired actuator position before starting motion.



1 Actuator | Motion | Position
Range Extender | Move To Position | 947.75 mm

Teach Position

Robot Move Beta

| Target | Move Type |
|---|---|
| Home | J |
| P1 | J |
| P2 | J |
| Home | J |