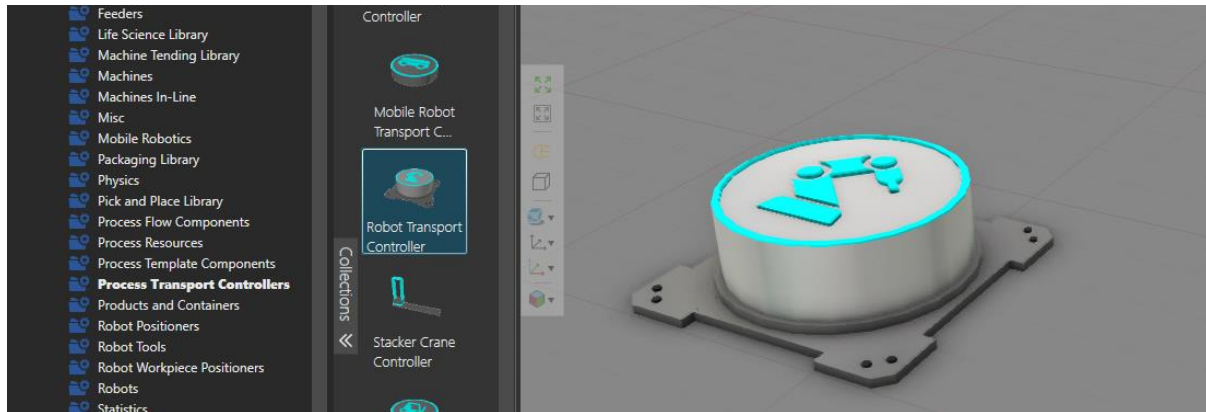


Robot Transport Controller - Manual

Visual Components 4.4 | Version: November 14, 2021



This manual describes how to use the Robot Transport Controller component. This will describe the basic use on the high level and documents all the properties found on the property panel tabs in detail. Use this document as a reference for the properties in the Robot Transport Controller component and in the associated transport links.

In this tutorial, you will learn how to:

- Use a robot as a transport resources by associating it with transport links in flow editor.
- How and why to use the properties on each tab
- How to use the properties found on the transport link property panel

This tutorial is not including the step-by-step instructions to create a layout or a test model. Please refer to other Process modeling tutorials to get the basics first.

Support

support@visualcomponents.com

Visual Components Forum

forum.visualcomponents.com

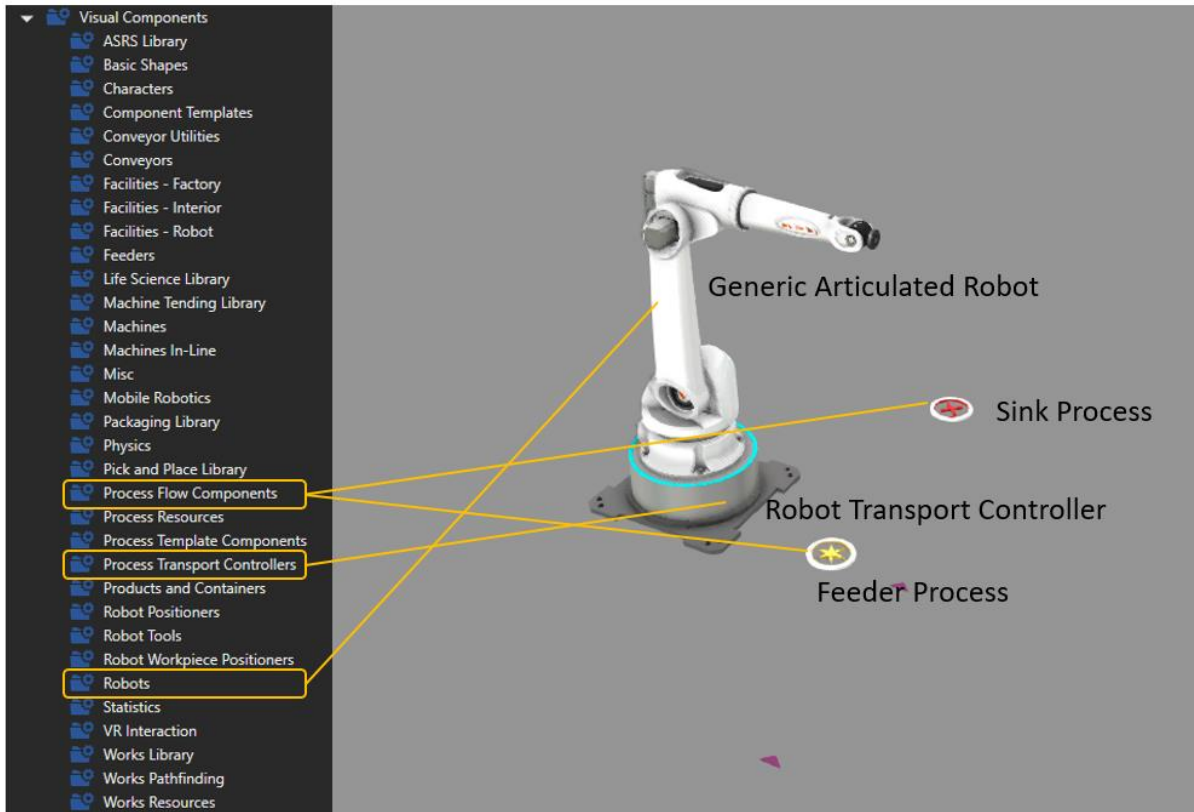
Contents

Basic use	5
Default tab properties	7
Configuration	7
Looks	7
PedestalDiameter	7
PedestalHeight	8
VisualizeStateColor	8
Speeds tab properties	8
Enabled	8
JointForce	8
JointSpeed	8
AngularAcc	8
AngularSpeed	9
CartesianAcc	9
CartesianSpeed	9
Track tab properties	9
Auto Homing tab properties	9
Enabled	9
Delay	9
Link Defaults tab properties	10
Work tab properties	11
MotionType	11
RoutineName	13
ScaleCycleTime	13
MotionDistance	13
MotionElevation	13
MotionLoops	13
Tool	13
ToolName	13
Approach	14
Priority	14
Advanced tab properties	14

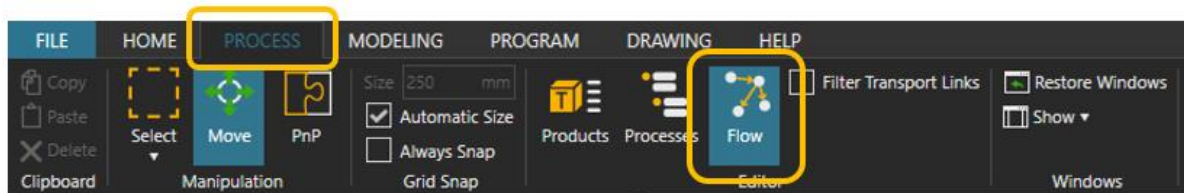
UseGripperSignals.....	14
GraspSignalPortIndex.....	14
GraspContainerName	14
ToolInterfaceName	15
Transport tab properties.....	15
Capacity.....	15
MultiPickStrategy.....	15
Transport Link properties.....	17
Priority.....	17
RobotMotion.....	17
RoutineName	18
Base.....	18
UseCustomParameters	18
Tool	18
ToolName.....	19
TcpIndex.....	19
PickTime	20
PlaceTime.....	20
PickApproach	20
PlaceApproach	20
PickApproachFrameFilter and PlaceApproachFrameFilter.....	21
PickApproachDirection	22
PlaceApproachDirection	22
SwapAtSource	22
SwapAtDestination	23
PreApproachTo	23
SwapTcpAt	23
PickPositionOffset and PlacePositionOffset	23
PositionTolerance	24
JointSpeed.....	25
RobotJointsOnMove	25
ReadCurrentJointValues	25
Implementer	25

Basic use

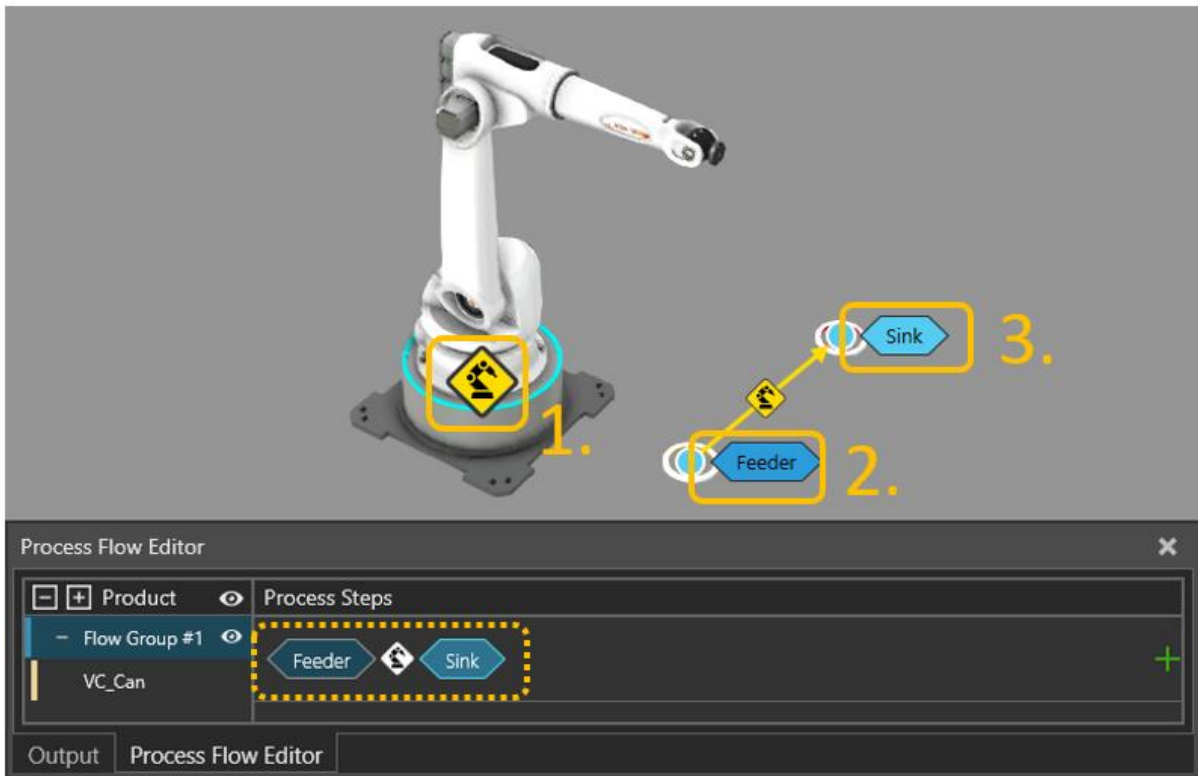
Find the components shown in the image below in the *eCatalog* and drag them to the scene. Use the *Plug and play* tool to attach the robot on top of the *Robot Transport Controller* component. Place the *Sink Process* and the *Feeder Process* components within the reach of the robot.



Go to Process tab and enable the flow editor.



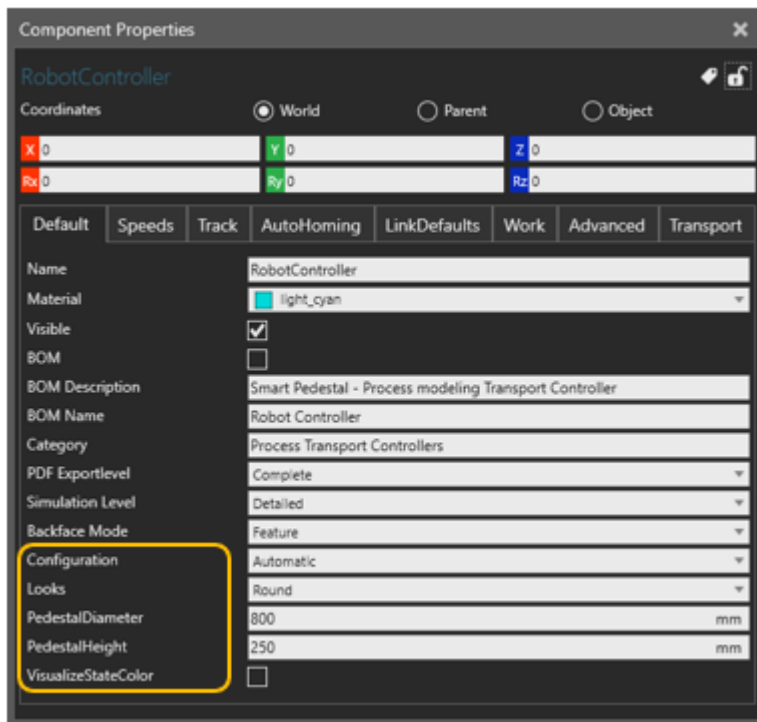
Configure the transportation from the feeder to the sink using the robot transport controller as the implementer of the transportation following these steps.



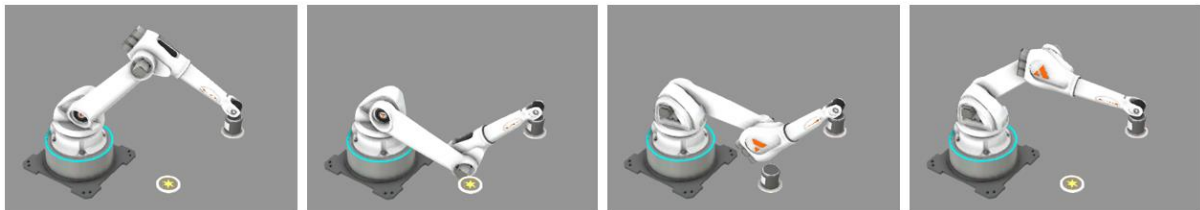
1. Set the robot controller as the active Transport Controller by selecting the icon on it
2. Hover the mouse pointer on the dot icon on the feeder and then click the “Feeder” label
3. Finally hover on the dot icon on the sink and then click on the “Sink” label

Now you can hit the play button to simulate and test it. Robot should pick the Can product that appears on the feeder and place it on the sink to be deleted and this process is repeated.

Default tab properties

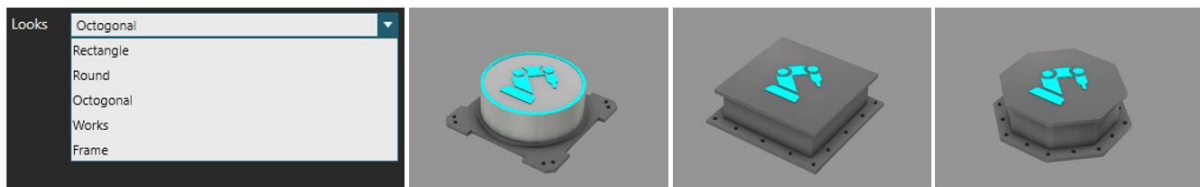


Configuration



Robot configuration defines the robot posture. Choose a desired robot configuration from a dropdown list. The options in the dropdown list depends on the robot model in use. Automatic option chooses the closest configuration on the fly but may lead to undesired configuration selection in certain circumstances.

Looks



Choose the looks of the controller component. This property has no effect on the functionality of the component and is simply an aesthetic setting.

PedestalDiameter

Defines the width / length / diameter of the component. This property has no effect on the functionality of the component and is simply an aesthetic setting.

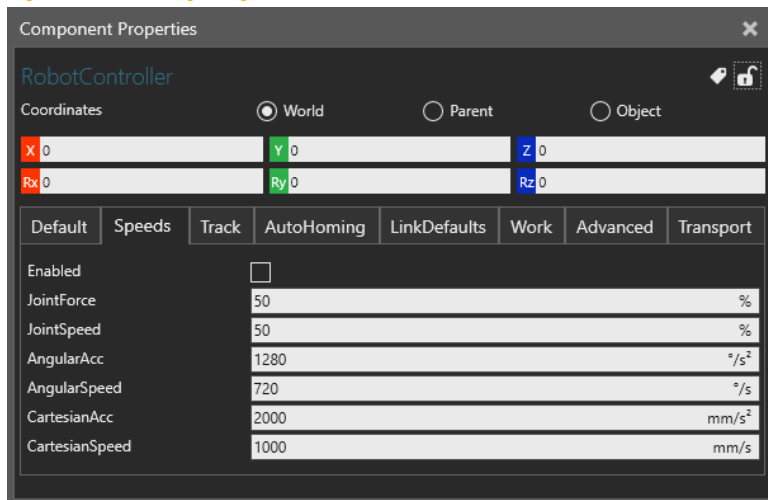
PedestalHeight

Defines the height of the component and the mount height of the robot. Changing this property will effect on the reachability of the robot in relation to other objects in the layout.

VisualizeStateColor

When enabled, changes the color of the pedestal according to the current state.

Speeds tab properties



Use these properties to control the speeds used in the automatically generated robot motions.

Enabled

If the speeds are not enabled, the controller will use the AngularAcc, AngularSpeed, CartesianAcc and CartesianSpeed that are set in the robot component and the JointForce and the JointSpeed are 100%.

If the speeds are enabled the values set in the robot controller are overwritten. Setting the enabled flag back to False will not set back the robot defaults and the overwritten values will remain.

JointForce

Defines the PTP motion acceleration and deceleration in percentage in relation to maximum joint acceleration and deceleration defined in the robot component connected on top of the controller.

JointSpeed

Defines the PTP motion speed in percentage in relation to maximum joint speeds defined in the robot component connected on top of the controller.

AngularAcc

Defines the angular (i.e. rotational) acceleration and deceleration used during linear motions. This value overwrites the values set to the robot.

AngularSpeed

Defines the angular (i.e. rotational) speed used during linear motions. This value overwrites the values set to the robot.

CartesianAcc

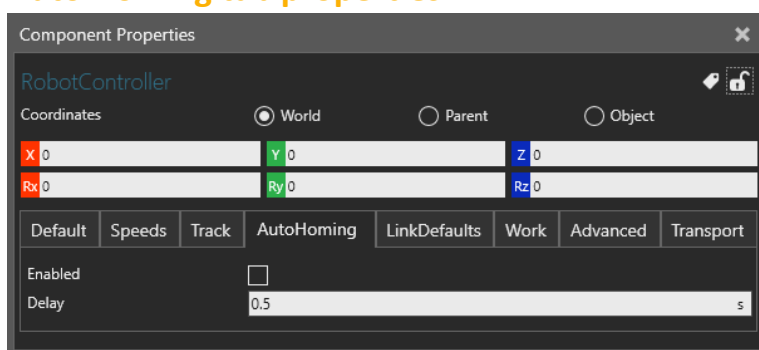
Defines the Cartesian (i.e. prismatic or linear) acceleration and deceleration during linear motions. This value overwrites the values set to the robot.

CartesianSpeed

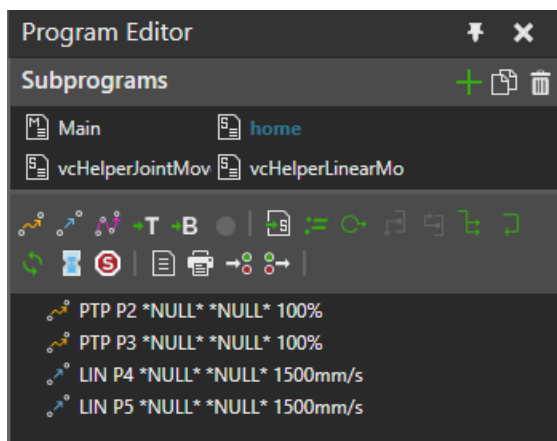
Defines the Cartesian (i.e. prismatic or linear) speed during linear motions. This value overwrites the values set to the robot.

Track tab properties

Auto Homing tab properties



Auto homing can be used to force robot to automatically to move to *home* position if it has no tasks to perform. The initial position that robot takes when simulation is reset is considered as home position unless there's a sub program home available on the program tab and it is not empty. If the home subprogram is available, it is executed always when robot needs to home instead of the init position homing.



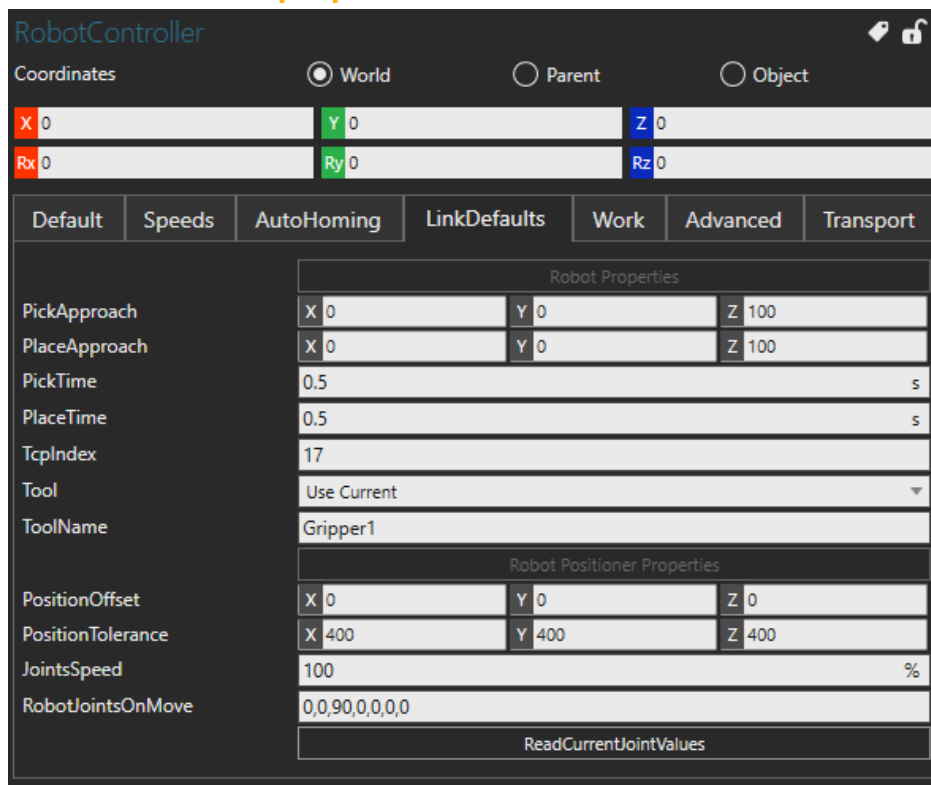
Enabled

Automatic homing is in use if the Enabled flag is checked.

Delay

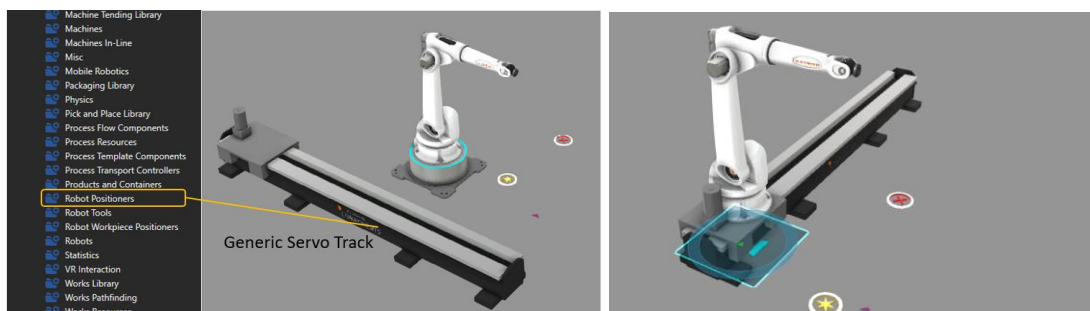
Defines the delay robot is allowed to idle before it automatically goes to home position (or the home sub program is called).

Link Defaults tab properties



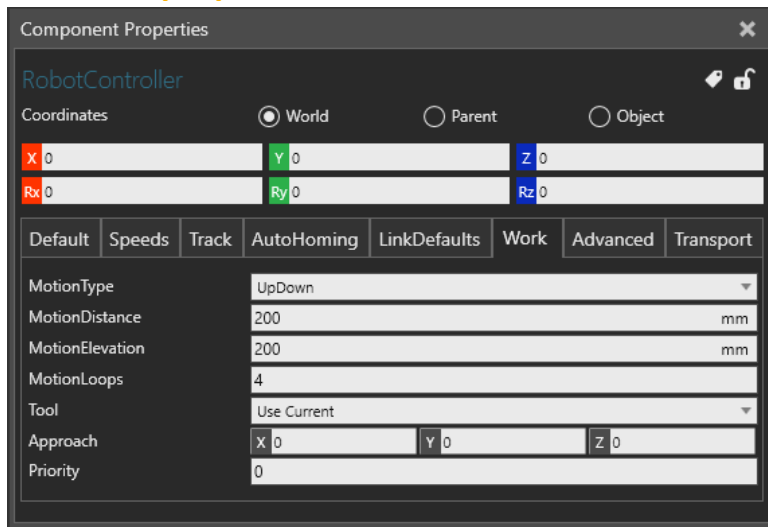
Link Default tab properties can be utilized to modify and alter the default automatic motions. These properties are also used as default values for the transport link properties that are assigned upon transport link association with this controller. If transport link property “UseCustomParameters” is disabled (i.e. unchecked) these advanced tab properties are actively being used. If “UseCustomParameters” is enabled the transport link properties will overwrite values defined in these advanced tab properties. Learn more about Transport Link Properties in a dedicated section for them later in this document.

There is two sections on this tab. Robot related properties and Robot Positioner related properties. These sections are separated with disable button properties *Robot Properties* and *Robot Positioner Properties*. Robot positioner properties configure the behavior of a robot track or a multi axis gantry. Robot positioner functionality supports up to 3 axis linear positioners. To use a Robot Positioner, connect the robot positioner first on top of the *Robot Transport Controller* and then the robot on top of the robot positioner component like shown in the image below. Track will be automatically controlled to locate the robot closer to task targets (e.g. pick, place or work location).

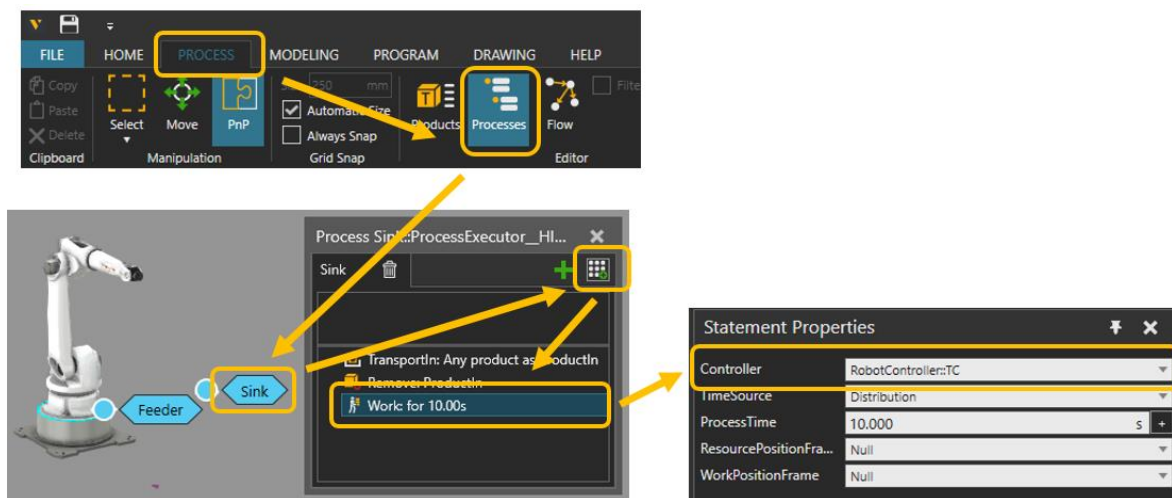


See the detailed explanation for each property later in this document in [Transport Link properties](#) section.

Work tab properties



Process Executors may call robot transport controller to execute “work”. If the Robot transport controller is associated to the Controller property of the *Work statement*. The statement simply calls robot to do some work for a specific time and the visualization of that work can be modified with these properties.



MotionType

There are four different types to visualize the work execution. Choose **Hold**, **UpDown**, **Rectangle** or **CustomRoutine**. The other properties shown on the tab depends on which type is selected.



In **Hold** mode the robot simply drives to the target frame defined in the work statement and stays still for the work process time. The motion time spent to move to the target is included in the process time, but possible time spent moving the servo track is excluded.

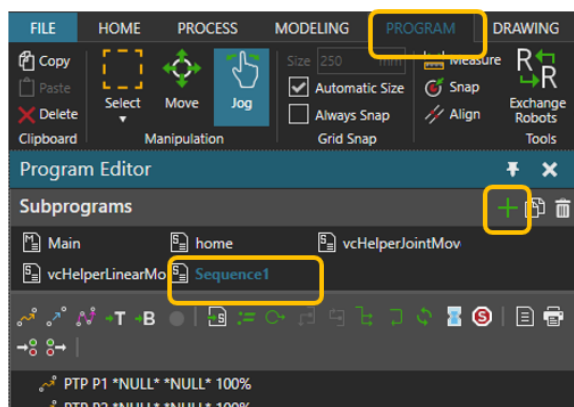
MotionType	UpDown
MotionDistance	400 mm
MotionElevation	0 mm
MotionLoops	5

In **UpDown** mode the robot moves up and down on top of the target frame and the motion speeds are adjusted to meet the given process time. The motion time spent to move to the target is included in the process time but possible time spent moving the servo track is excluded.

MotionType	Rectangle
MotionDistance	400 mm
MotionElevation	0 mm
MotionLoops	5

In **Rectangle** mode the robot moves around the target frame in square loop and the motion speeds are adjusted to meet the given process time. The motion time spent to move to the target is included in the process time but possible time spent moving the servo track is excluded.

MotionType	CustomRoutine
RoutineName	Sequence1
ScaleCycleTime	<input type="checkbox"/>



In **CustomRoutine** mode the robot will execute a routine that must be manually taught on the program tab. In this mode the given process time is not necessarily respected and the custom routine itself defines the process time (see ScaleCycleTime). In order to create a custom routine visit the Program tab and create a new subprogram for a robot.

RoutineName

Defines the name of the subprogram that will be executed. If the subprogram does not exist, a hold motion type is used instead.

ScaleCycleTime

When enabled, and when a custom routine is called, the motion statement cycle times are automatically scaled to meet the given process time. Only PTP and Linear type of motion statements are scaled, and all other execution time will be additional time on top of the given process time. That is, the time spent in other type of statements, e.g. delay and wait for binary input statements, are not taken into account.

MotionDistance

Defines the motion size in **UpDown** and **Rectangle** modes.

MotionElevation

Defines the height offset along z axis from the work target frame defined in the work statement calling the execution. Active in **Hold**, **UpDown** and **Rectangle** modes.

MotionLoops

Number of loops robot takes during the process time in **UpDown** and **Rectangle** modes. Increasing the number of loops without changing the process time will increase the motion speed.

Tool

With this option, you can define the tool that will be used to execute work tasks. The options are:

Use Current: Keep existing tool (if any used in the previous task)

Tool Name: Define the tool by its (component) name with the Property [ToolName](#).

No Tool: No tool is allowed. If the resource has any existing tool, those must be returned first.

The utilized tools will be connected through Tools interface. An error is printed to the output and simulation is paused if the tool is not found.

In case the defined tool is found but is reserved to another resource the task is kept in a priority queue until the tool is available. If other tasks are available that do not require the tool the resource will be dispatched to that task instead (skipping tasks with a higher priority).

ToolName

Name of the tool component in the scene that is used for work tasks. If this tool is not connected to the robot, the robot will move to the tool position and connect it before the actual work task. If another tool is connected robot, it will be released first.

The name does not need to be complete. A part of the name is sufficient and is used as a lookup name. In search, (*) wildcards are automatically added in front and end of the name.

For example, if there are two tools available with names “MyTool #1” and “MyTool #2”, tool name “MyTool” would accept either one.

Approach

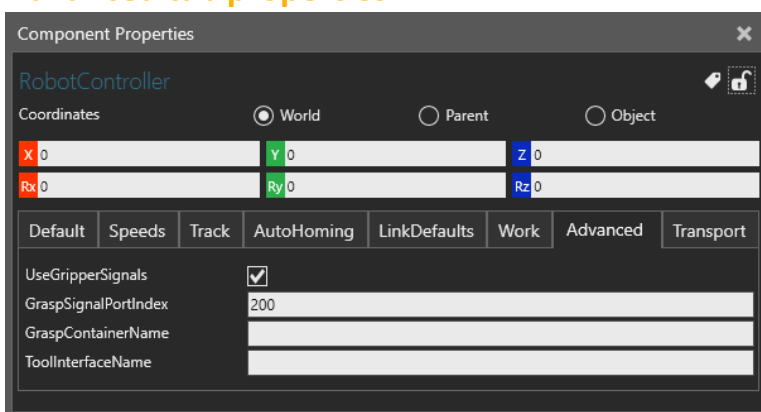
Defines a position that is used as via point when robot will move to execute the work task.

Priority

Defines the priority of work tasks among all active work and transportation tasks. Transport and work tasks share the same priority system.

The priority is a number. The lower the value, the higher the priority, e.g. 1 = high priority and 100 = low priority. Tasks with equal priorities are dealt with FIFO principle.

Advanced tab properties



UseGripperSignals

Controls connected gripper tool with signals by closing it when grasping and opening it when releasing. This is relying on the default naming of the signals.

If there is a signal named “IN_J1_Action” in the tool component, that signal is triggered **True** to close, and **False** to open. Else “IN_J1_Close” is triggered **True** to close, and “IN_J1_Close” **True** to open.

GraspSignalPortIndex

Defines port index of the robot output signal map that is automatically connected to “GraspSignal” signal of the pedestal, when robot is connected on the pedestal. This signal index is used in the recorded or manually taught robot routines (on Program tab) for grasping and releasing the product. See the [Transport Link properties](#) -section in this document to learn more about recorded routines.

Recorded routines will not use the default grasp and release actions in the robot for picking and placing. Instead, the signal is connected to a specific logic in the pedestal. This value can be only edited when pedestal is not connected to anything. In most cases this value doesn't need to be edited.

GraspContainerName

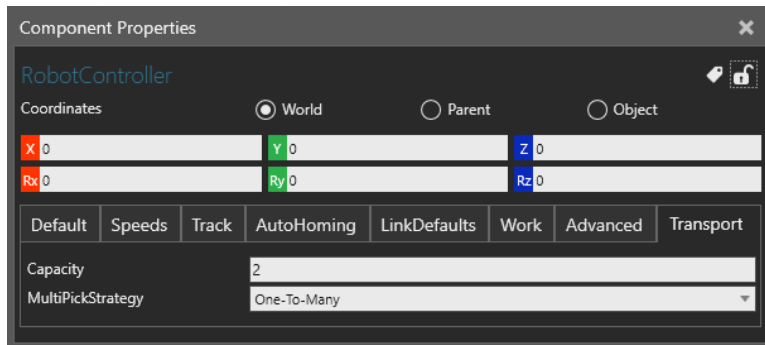
The name of the container behavior in the robot component used for grabbing components. If the value is left empty, or if the container defined by the name is not found in the robot,

the default grasp container in the flange node of the robot is used. This property is only useful in rare use cases, and it can be left empty.

ToolInterfaceName

The name of the interface behavior in the robot component for connecting end effectors. If the value is left empty, or if the interface defined by the name is not found in the robot, the default Tool interface in the flange node of the robot is used. This property is only useful in rare use cases, and it can be left empty.

Transport tab properties



The robot can collect and deliver multiple products simultaneously between processes that have active transports (matching TransportIn and TransportOut statements).

Capacity

Defines the maximum number of products that the robot tool can hold at a time. The robot collects all products first then delivers them. In case of utilizing swap action with double gripper, the capacity must be increased to 2 from the default value of 1.

MultiPickStrategy

Defines the strategy that is followed when multi-transporting. To change the value, change first the **Capacity** property value to greater than one.

One-to-One: Collecting and delivering is carried only between two processes at a time.

One-To-Many: Collects all products from one process and delivers to any number of processes at a time.

Many-To-One: Collects products from any number of processes and delivers to one.

Many-To-Many: Collects products from any number of processes and delivers to any number of processes.

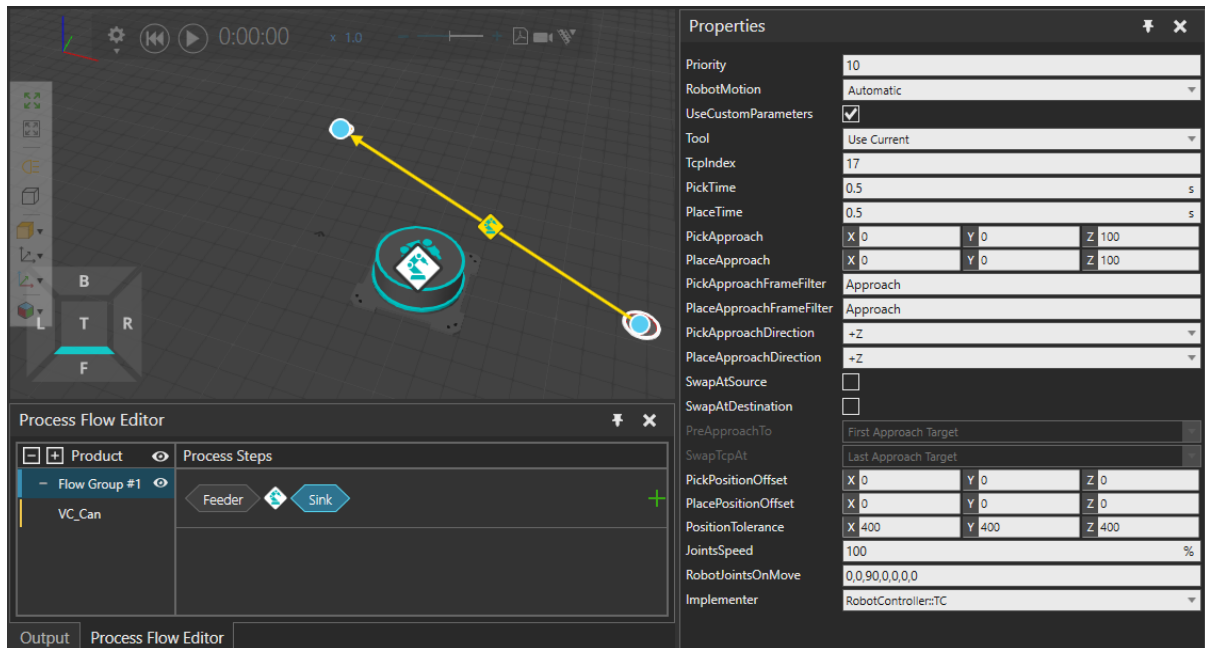
The robot controller can receive new collect tasks until its capacity is consumed or it initiates delivery. After delivery is initiated, new collect tasks are not accepted until all deliveries of the collected products are completed.

Delivering of the collected products is carried in the First-In-First-Out (FIFO) order, meaning that the products are delivered in the same order to which they were collected.

Note: Multi-transporting is greedy, which means that the robot will be dispatched to handle as many transportations (collect and delivery) tasks as possible when multiple transportations that meet the strategy are available.

Transport Link properties

In addition to the properties in the robot controller, the transports can be customized with the properties found under the transport links. Select the transport link and make sure it is associated with the robot transport controller to see the transport link properties. The properties in the controller component affect to all transports the controller will perform, but with the transport link properties the customization can be done for each transport link separately.



Priority

Defines the priority of transportation via the corresponding link among all active transportation tasks. Similarly, all work tasks are prioritized according to "Work::Priority" property in the Transport Controller. Transport and work tasks share the same priority system.

The priority is a number. The lower the value, the higher the priority, e.g. 1 = high priority and 100 = low priority. Tasks with equal priorities are dealt with FIFO principle.

Note: Link priority is applied only to picking. The placing is currently following FIFO principle. The priority is applied only to the active transport/work tasks. This means that it does not affect the transportation pairing between the processes. In other terms, processes do not have priority, only the transportation/work that the processes send/publish to the transport system.

RobotMotion

Robot motions are executed by default automatically. In certain use cases, however, the robot motions must be manually edited. This can be achieved with the help of subprograms. Typical workflow: Set the value to **RecordRoutine** and run simulation until this transport link transportation is executed once. Then reset the simulation and switch the mode to **ExecuteRoutine**.

Automatic: In this mode the python logic in the controller component controls the robot to perform the transportations.

RecordRoutine: In this mode the python logic in the controller component still controls the robot, but all the motions are recorded to the subprograms (`_PICK` and `_PLACE`) for further editing, tweaking and possible post-processing on the Program tab.

ExecuteRoutine: In this mode the controller simply executes the routine defined in the **RoutineName** property. Routine must include one set bin to true statement and one set bin to false statement to grasp and release the product. Index of the grasp signal port can be seen and modified in the controller properties on Advanced tab and is 200 by default.

RoutineName

Prefix of the robot subprogram names that are called to perform the transporting if the **RobotMotion** property is set to **ExecuteRoutine**. If the **RobotMotion** property is set to **RecordRoutine** mode, the value of **RoutineName** property will be used for the newly generated routine. Any existing routine with the same name will be overwritten.

The pick routine that is used is called with **RoutineName** + “_PICK” and the place routine is called similarly with **RoutineName** and a postfix “_PLACE”.

Base

Defines the coordinate system of the motion points when using the **RecordRoutine** mode in **RobotMotion** property. If *Attach to Target* is used, the robot motions are associated with the coordinate system (i.e. Base) that is connected to the pick up and drop of components. If those components are moved after recording the robot motion points will automatically follow. This allows easily to change the layout even after recording the robot motions. If the **Base** property is set to *Robot World*, all robot motions are associated to a coordinate system (i.e. Base) that is connected to the robot. This means that if the robot location is changed the motion points follow the robot location and may not respect the pick up and drop off locations anymore (for example CNC machine).

UseCustomParameters

Enable this property to overwrite the default values for properties defined on the *LinkDefaults* tab of the controller properties.

Tool

With this option, you can define the tool that will be used to transport the product(s). The options are:

Use Current: Keep existing tool (if any used in the previous task)

Tool Name: Define the tool by its (component) name with the Property [ToolName](#).

Product Property: Define the tool name that will be read from the given product property (string). The name lookup is the same as in [ToolName](#) option. This option is not available in the *LinkDefaults*.

No Tool: No tool is allowed. If the resource has any existing tool, those must be returned first.

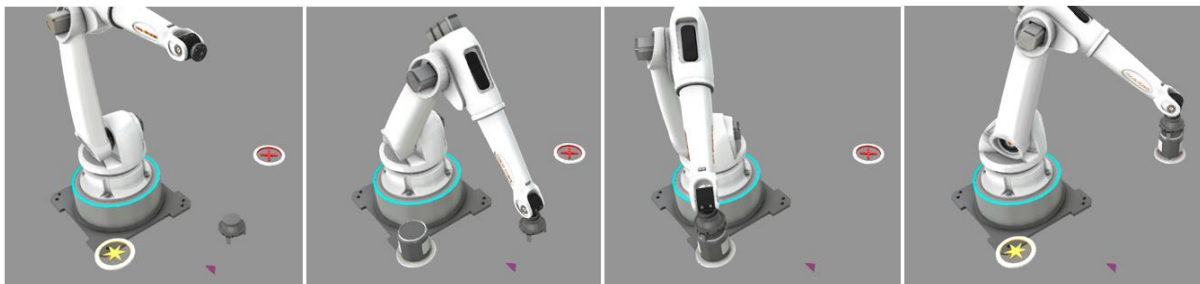
The utilized tools will be connected through Tools interface. An error is printed to the output and simulation is paused if the tool is not found.

In case the defined tool is found but is reserved to another resource the task is kept in a priority queue until the tool is available. If other tasks are available that do not require the tool the resource will be dispatched to that task instead (skipping tasks with a higher priority).

ToolName

Name of the tool component in the scene that is used for transporting. If this tool is not connected to the robot, the robot will move to the tool position and connect it before the actual transportation task. If another tool is connected robot, it will be released first.

The name does not need to be complete. A part of the name is sufficient and is used as a lookup name. In search, (*) wildcards are automatically added in front and end of the name. For example, if there are two tools available with names “MyTool #1” and “MyTool #2”, tool name “MyTool” would accept either one.



EndEffectorName Generic 3-Jaw Gripper

TcpIndex

Defines a list of tool frame indexes used for transportation tasks. To define a tool frame, go to Program tab, and then select the robot. Then click Jog panel, select the tool frame you want to edit.

All the robots in the Visual Component public library have 16 tool frames by default. When an end-effector with a predefined Tool is connected to a robot, it is automatically added to the list of tools frames with the index of 17. When the value is 17, the tool frame of the connected end-effector is used. If the defined tool frame index is not found (for example, when no end-effector is connected), the first tool frame is used.

You can list multiple tool frame indexes by separating them with commas, for example:

- “17, 18” or
- “17, 18, 19”

This is useful for multi-transporting or for double-gripping use cases. The first available tool frame index on the list is always used.

PickTime

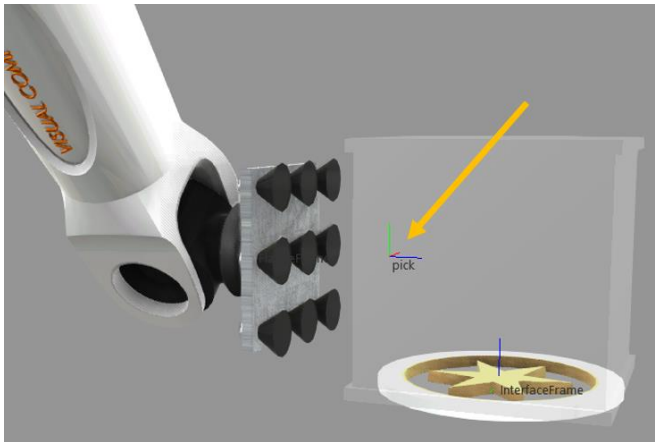
Amount of a time that the robot spends when doing grasping.

PlaceTime

Amount of a time that the robot spends when doing releasing.

PickApproach

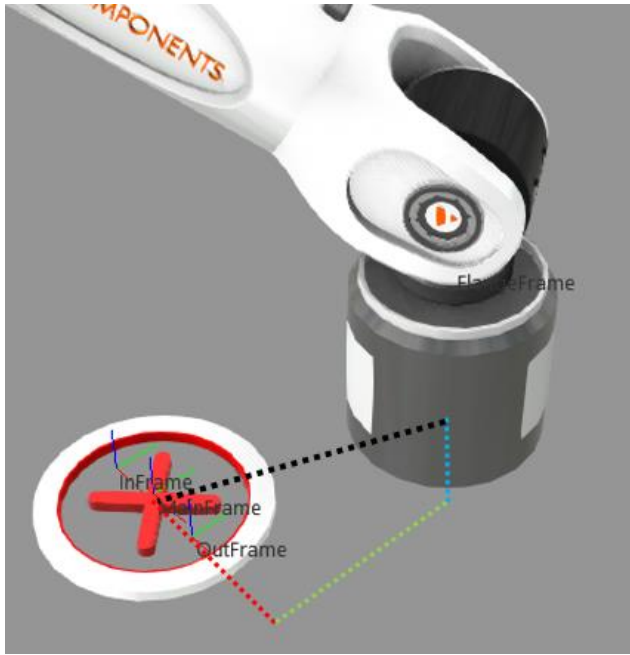
Vector value to define the approach offset from the pick location. The vector is applied in the coordinates of the pick location which is by default in on top of the product component and the vector is applied on that in product coordinates. The pick location can be defined with a frame feature in the product component named as “pick”. In that case the vector is applied in the flipped (around Y-axis) frame coordinates.



PlaceApproach

Vector value to define the approach offset when placing the part to the target location. The vector is applied in the coordinates of the target that is typically defined by a frame in the receiving process. The vector is applied in the target coordinates.

PlaceApproach	X 150	Y 200	Z 100
---------------	-------	-------	-------



PickApproachFrameFilter and PlaceApproachFrameFilter

Defines a string that is used as a keyword to filter all the frames in a process component which names (excluding an underscore and number in the end of the name) end with it. Those filtered frames are then used as approach frames in the order defined with the indexes on the name before the robot picks or places the product. After the pick or place has been done, the robot does the retract movements in the reverse order.

For example, if the filter is set to “Approach” and the process component would have two frames called “ProductLocationApproach”, “ProductLocationApproach_11” and “ProductLocationApproach_2”, those three locations would be used as a motion targets before the robot moves to the actual product location target. The frames would be sorted in the given order:

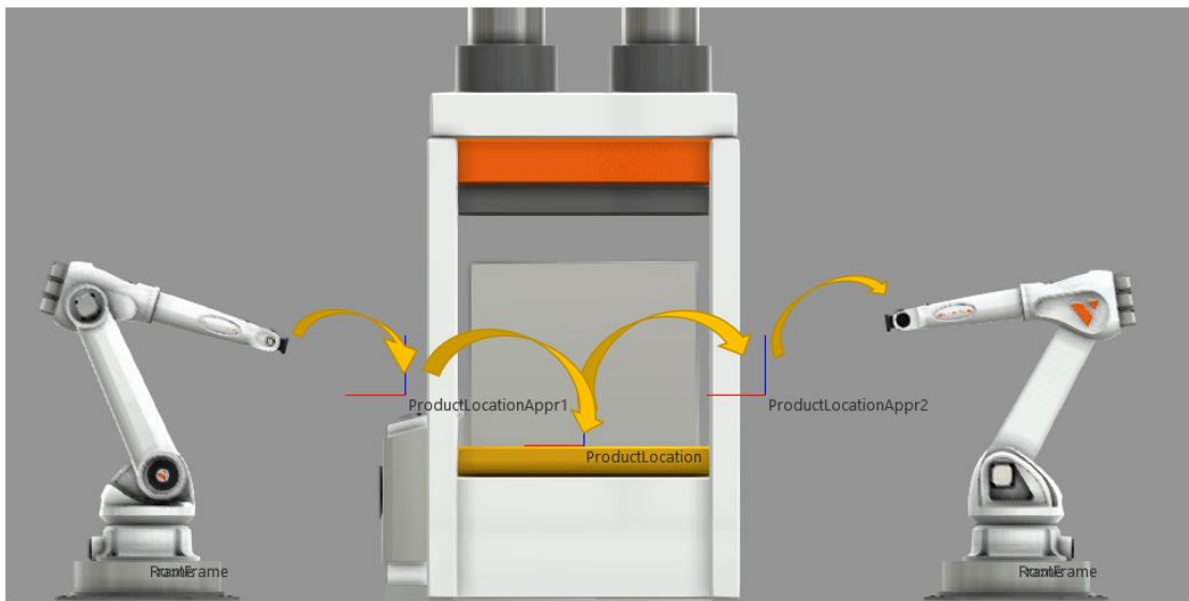
1. “ProductLocationApproach” (index is 0)
2. “ProductLocationApproach_2” (index is 2)
3. “ProductLocationApproach_11” (index is 11)

Note also that:

- If we would add fourth frame called “AnotherApproach_4”, it would be added as the first frame in the list, because the actual name before the index is sorted first.
- If we would add a frame called “ProductLocationApproach2”, it would not be used as an approach point. The reason for this is that the number in the end is not separated with an underscore, and therefore it is considered as a part of the name.
- The actual name of the frame must not contain underscores, for example “Product_Location_Approach”.
- Also, underscores after the index are not considered. For example, “ProductLocationApproach_2_1” would be sorted using the first index only.

- Note that if you want to add new frame features, or edit their locations and names, you need to have access to the Modeling tab (requires *Visual Components Professional* or *Visual Components Premium* level product).

This can be useful for example in a use case, where a product is placed with another robot than it is picked out, and when the robots are located on different sides of the process component. In the image below the robot on the left is associated with the incoming transport link and the PlaceApproachFrame property in that link is set to “Appr1” and the robot on the right is associated with the outgoing transport link and the PickApproachFrame in the link is set to “Appr2”.



PickApproachDirection

Defines the face that is used for grabbing the part when picking. If the value is +Z, the middle point of the top face is used to as the target point. This can be changed, for example, when part must be grabbed from the side.

However, this approach direction is ignored if the product has a frame called “pick”. In that case, the frame is used to as the target point.

PlaceApproachDirection

Defines the direction that is used for approaching the target point when placing. If the value is +Z, the target location is approached from the above. This can be changed, for example, when part must be placed sideways, for example on a shelf of a rack.

SwapAtSource

This option is used for a robot that has a double gripper as end effector. Change this value to True, if you want to pick a product from the source process that the link is connected from, and immediately after this place another product in the same process with the same gripper (but different Tool frame).

Hint: Remember to increase the capacity of the transport to 2 on the Transport tab when utilizing swap.

SwapAtDestination

This option is used for a robot that has a double gripper as end effector. Change this value to True, if you want to place a product to the destination process that the link is connected to, and immediately after this pick another product in the same process with the same gripper (but different Tool frame).

Hint: Remember to increase the capacity of the transport to 2 on the Transport tab when utilizing swap.

PreApproachTo

This option is used for a robot that has a double gripper as end effector. Defines whether the first or last approach frame is used as a waiting position, when the robot is swapping.

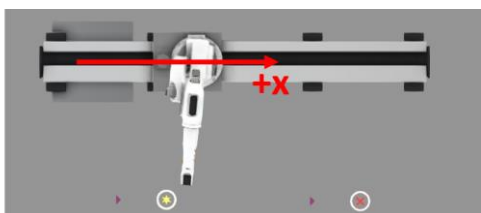
First Approach Target is usually a good option, because it is typically located outside of the machine – the machine is still busy, and its doors are closed.

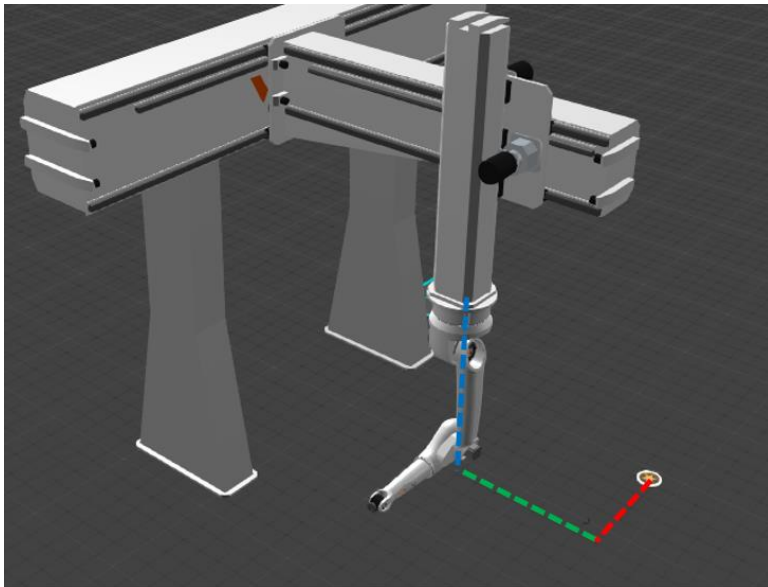
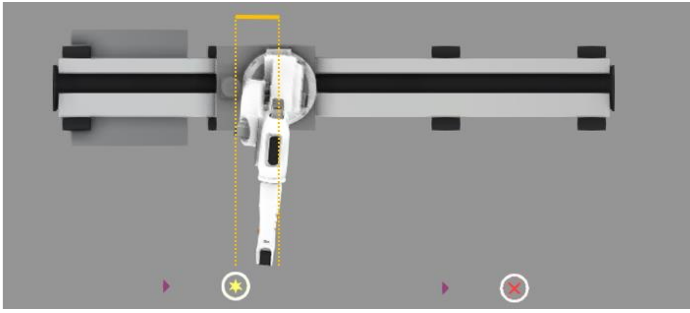
SwapTcpAt

This option is used for a robot that has a double gripper as end effector. Defines the location that is used to change the Tool frame to another. With a small product, it is possible to do the swapping closer to the process location. But to avoid collisions when swapping the tool frame with a bigger product attached to the tool, you might need to select a position that is outside of the machine.

PickPositionOffset and PlacePositionOffset

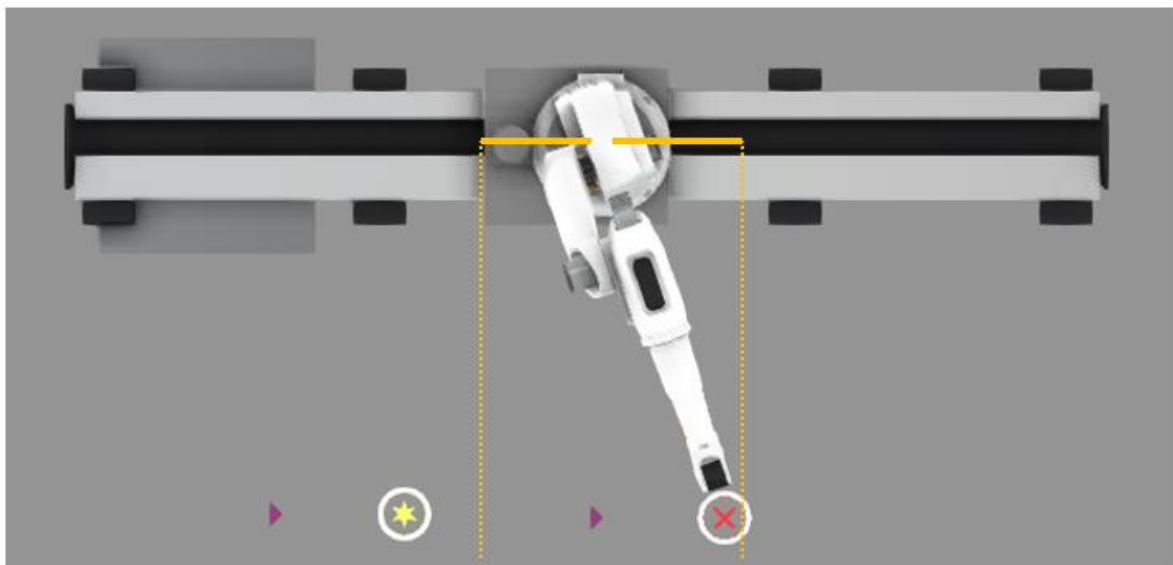
This property defines the robot location offset from the actual task (pick, place, work) location. Both pick and place position offset properties default to the same controller component LinkDefault-tab property *PositionOffset*. Use positive or negative values. The offset is given in Positioner coordinate system. For example, if a robot track axis moves the robot along the positioner's local X axis, set the offset value in X field of the vector. In case the robot positioner is a 3-axis gantry, you may utilize all x, y and z fields of the vector property.





PositionTolerance

If robot is already close to the task target, minor track movements can be filtered out. PositionTolerance property defines how close robot track must be to skip the track movement. This means if the PositionTolerance is given to be 500mm along x-axis and the robot is already within e.g. 300mm from the positioner target position, the positioner won't move to the target position but the robot will reach the target from the current position. Hint: Use larger tolerance for robots with longer reach.



JointSpeed

Track servo axis speed in percentage of the maximum speed set in the robot positioner component.

RobotJointsOnMove

Comma separated list of joint values that define the robot posture during moving the track. The default value has 6 joint values but the list length must respect the number of joints in the connected robot model. Use dash ("-") to allow joint to remain in its current value after previous task. For example "0,0,0,-,-,-" will define only the values for 1st, 2nd and 3rd joints and will leave 4th, 5th and 6th joint to their current values.

ReadCurrentJointValues

Sets the current joint values of the robot to the **RobotJointsOnMove** property.

Implementer

This is a default property in all transport links and it defines the transport controller responsible for the transport task. Set RobotController here. Note that you may have multiple robot controllers in the scene. They all have a unique component name. The robot controller is automatically set as an implementer when the link is associated with the robot controller in the 3d scene.