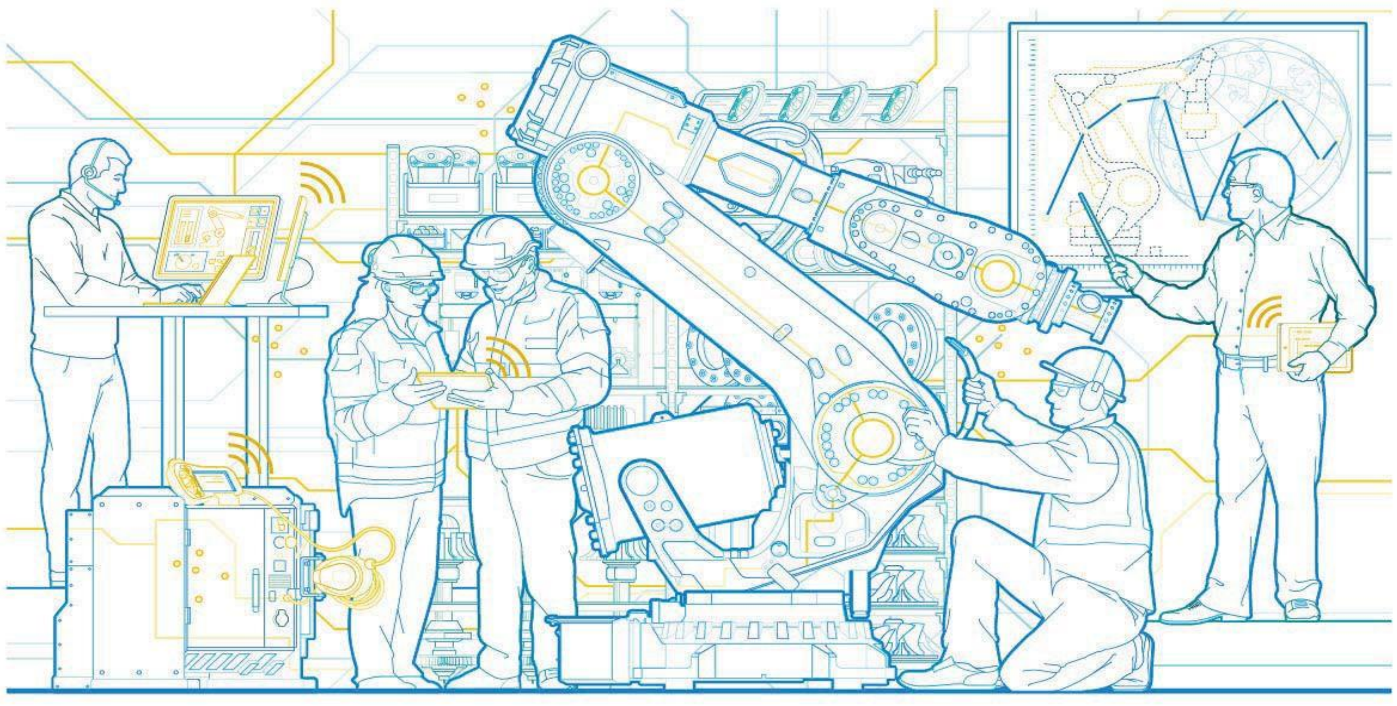


RobotStudio^{tm} 6.08

SmartComponents

The information in this manual is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this manual.
Except as may be expressly stated anywhere in this manual, nothing herein shall be construed as any kind of guarantee or warranty by ABB for losses, damages to persons or property, fitness for a specific purpose or the like.
In no event shall ABB be liable for incidental or consequential damages arising from use of this manual and products described herein.

This manual and parts thereof must not be reproduced or copied without ABB's written permission, and contents thereof must not be imparted to a third party nor be used for any unauthorized purpose.
Contravention will be prosecuted.
Additional copies of this manual may be obtained from ABB at its
then current charge.

© Copyright 2017 ABB All right reserved.
ABB AB
Robotics Products
SE-721 68
Västerås Sweden

2017-05-08 ABB

1. Smart Components

Overview

Smart Components give RobotStudio users a series of building blocks or tools which enable components within a station to have more complex behavior. Some examples are gipper motion, objects moving on conveyors, logic etc. Once created these objects can be saved as library files for use in future stations and simulations.

1.1. Create Smart Component (SC) - InFeeder

Overview

In this exercise we will learn how to create a Smart Component that simulates an in feeder. It will create dynamic objects which it will move in a straight line until they arrive at a picking position. When an object is removed from the picking position a new object will automatically be created.

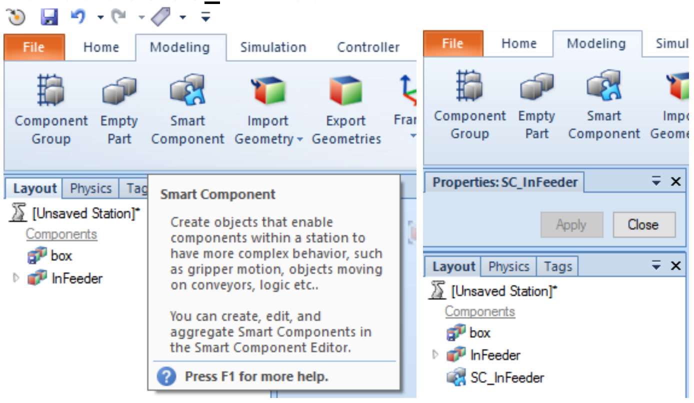
Preparing the station

We will start with an empty station and import the geometry of the conveyor, and the part which should be generated by the SC.

1. Import the geometry ....*Geometry\InFeeder.sat*
2. Import the library ...*Libraries\box.rslib*

Creating the Smart Component

1. On the **Modeling** tab click the **Smart Component** button.
2. Name the SC **SC_InFeeder**



Smart Component

Create objects that enable components within a station to have more complex behavior, such as gripper motion, objects moving on conveyors, logic etc..

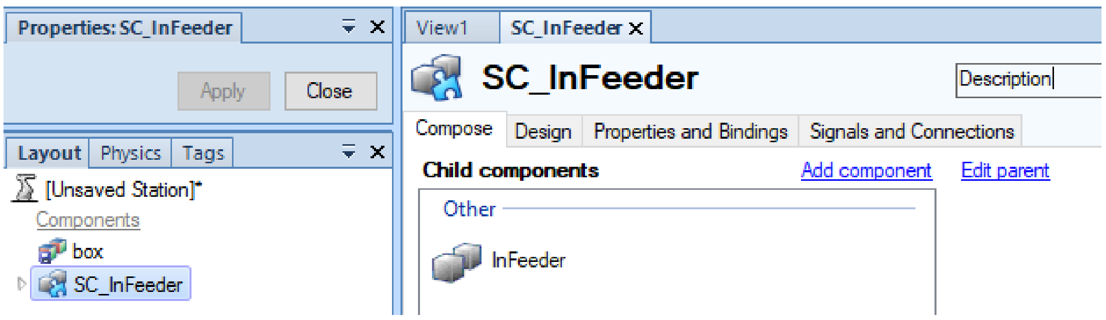
You can create, edit, and aggregate Smart Components in the Smart Component Editor.

Press F1 for more help.

Adding a part to the SC

Next we will add the part **InFeeder** to our SC as a **Child component**.

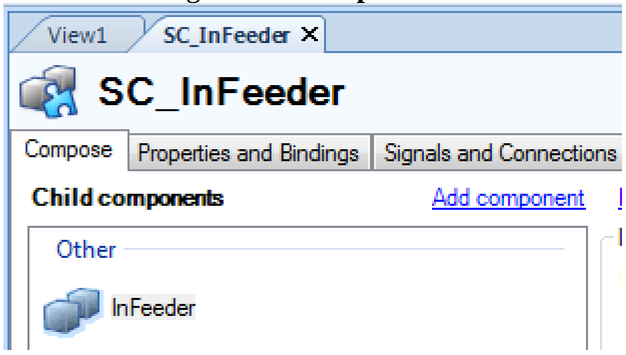
1. In the **Layout** browser drag the part **InFeeder** and drop it on **SC_InFeeder**.



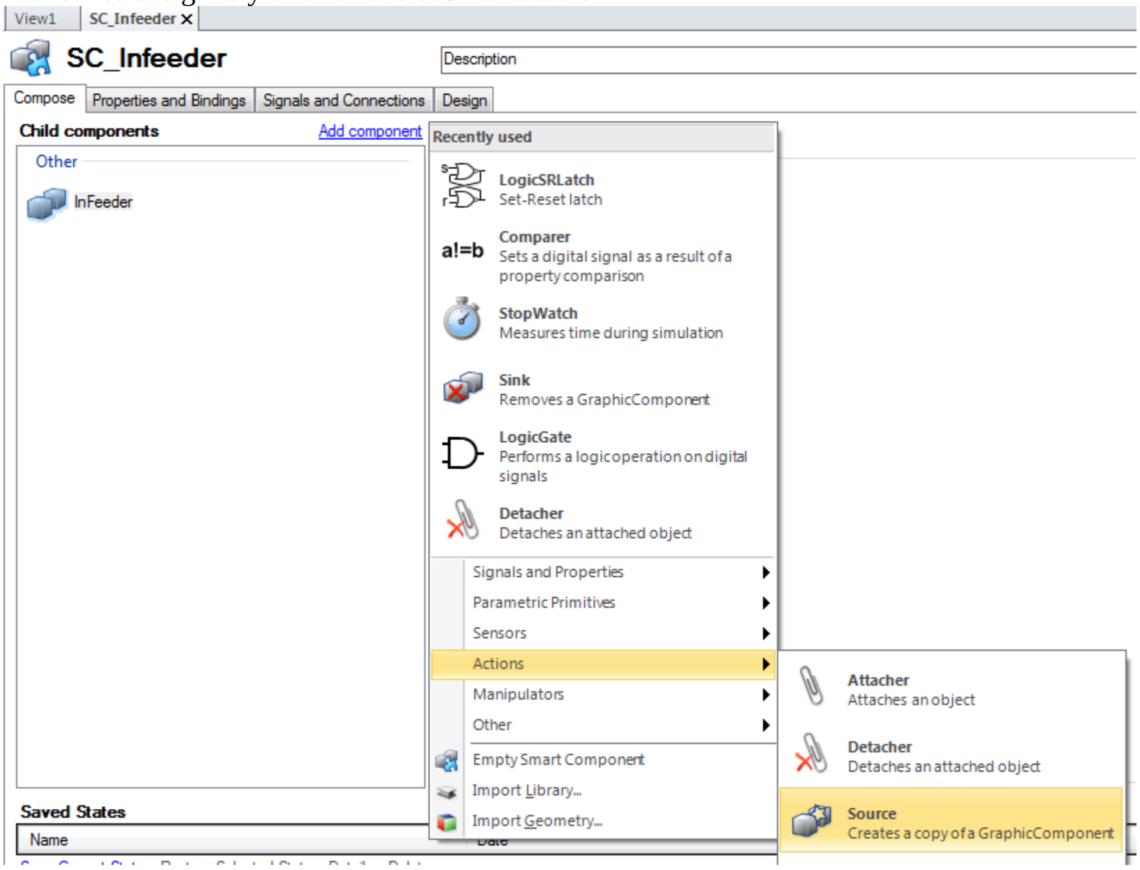
Adding base components to our SC

A Smart Component consists of one or more **base components**. A **base component** is a building block with predefined behaviors and properties.

1. In the SC view go to the **Compose** tab and click the **Add component** button.

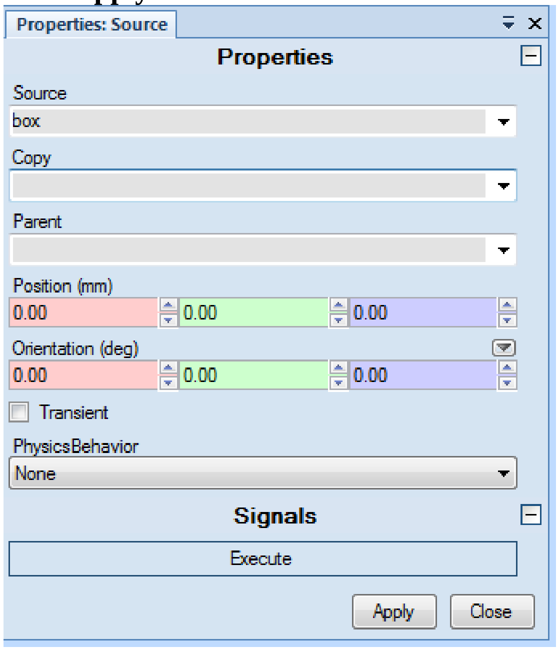


2. In the **Actions** gallery click on the **Source** button.



3. Right click on the **Source** and select **Properties** in the context menu (if the properties window not already is opened).
4. In the Properties window select the **box** in the **Source** drop down box.
   NOTE: If the Transient box is checked the part (in this case the box) will automatically be deleted when the simulation is stopped. Leave the box unchecked for this exercise.

5. Press **Apply**.


Properties: Source

Properties

Source: box

Copy:

Parent:

Position (mm): 0.00 0.00 0.00

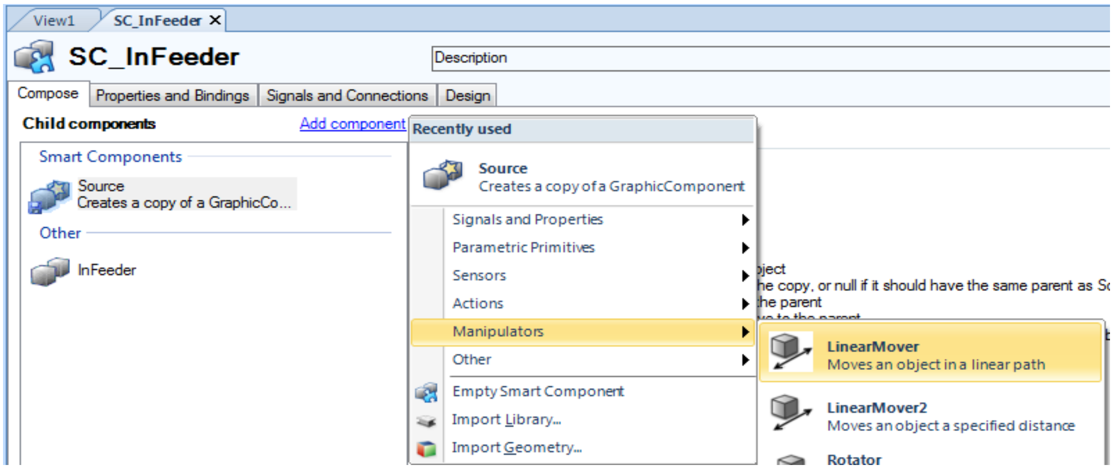
Orientation (deg): 0.00 0.00 0.00

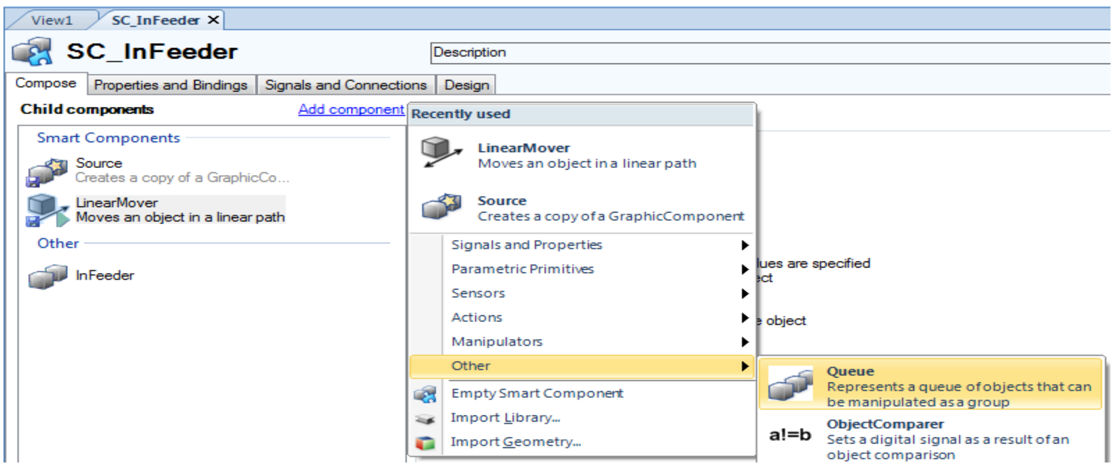
Transient

PhysicsBehavior: None

Signals

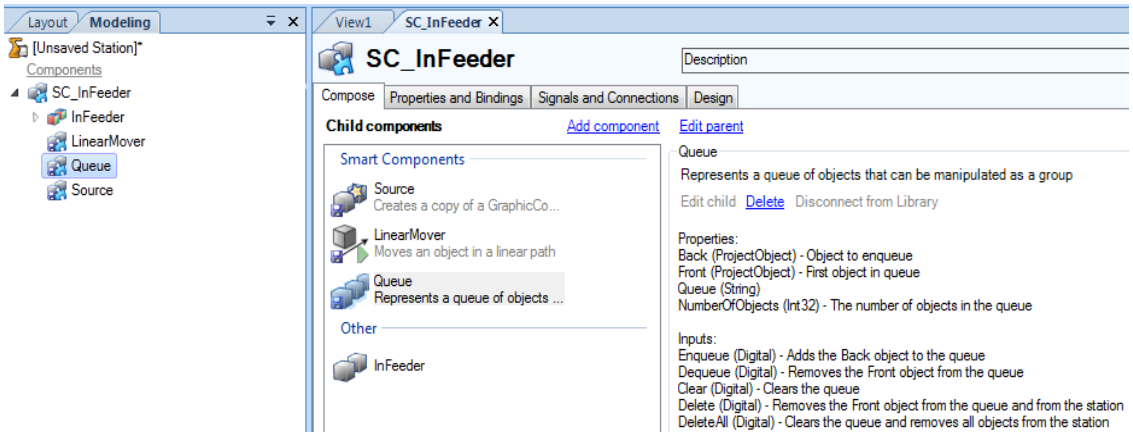
6. Add the base component **LinearMover** from the **Manipulators** gallery, and the base component **Queue** from the **Other** gallery.





Note: As additional components are added they will populate in the recently used section.

7. The SmartComponent should now have the following features present.



SC_InFeeder

Description

Child components

Source
Creates a copy of a GraphicCo...

LinearMover
Moves an object in a linear path

Queue
Represents a queue of objects ...

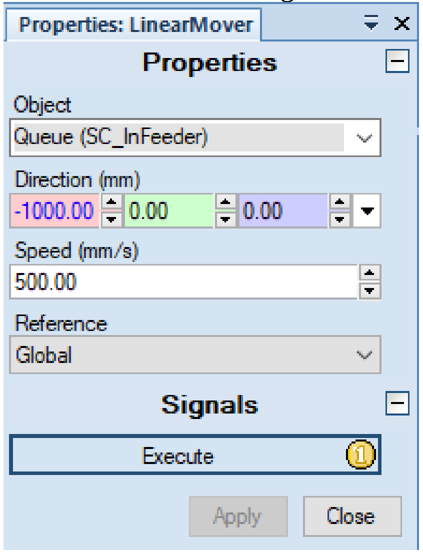
Queue

Represents a queue of objects that can be manipulated as a group

Properties:
Back (ProjectObject) - Object to enqueue
Front (ProjectObject) - First object in queue
Queue (String)
NumberOfObjects (Int32) - The number of objects in the queue

Inputs:
Enqueue (Digital) - Adds the Back object to the queue
Dequeue (Digital) - Removes the Front object from the queue
Clear (Digital) - Clears the queue
Delete (Digital) - Removes the Front object from the queue and from the station
DeleteAll (Digital) - Clears the queue and removes all objects from the station

8. Open the Properties window for the **LinearMover** by right clicking on it in the **Layout** browser. Select the **SC_InFeeder/Queue** as the Object. Change the **Direction** to X= -1000, Y=0, Z=0 (the box will move in minus X direction). Change the **Speed** to 500mm/s. Press **Apply**.
9. Activate the **Execute** signal. Close the Properties window.



Object: Queue (SC_InFeeder)

Direction (mm): -1000.00 0.00 0.00

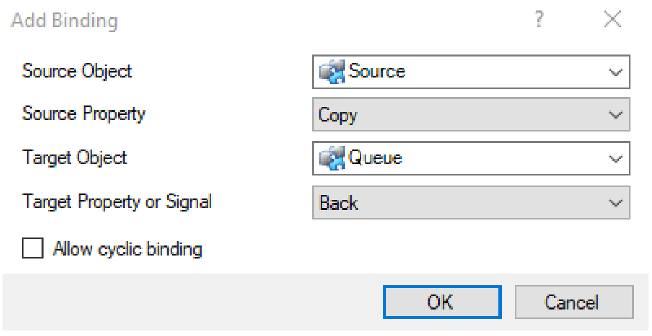
Speed (mm/s): 500.00

Reference: Global

Adding Property Bindings

The next step is to add a binding between the base component **Source** and the base component **Queue.** This will define which object will be moved by the conveyor.

1. In the SC view, go to the **Properties and Bindings** tab and click the **Add Binding** button.
2. Bind the **Copy** of the **Source** to the **Back** of the **Queue** and press **OK**:



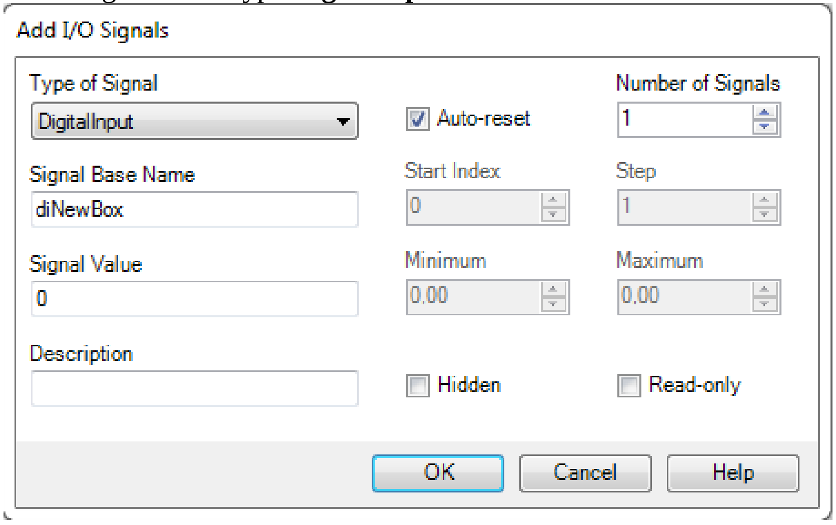
Source Object	Source
Source Property	Copy
Target Object	Queue
Target Property or Signal	Back

☐ Allow cyclic binding

Adding Signals and Connections

In order to define actions in our SC we need to define **I/O Signals** and **I/O Connections**.

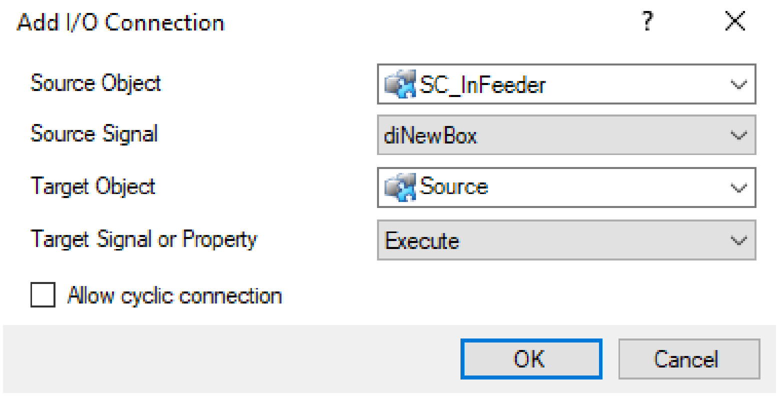
1. In the SC view, go to the **Signals and Connections** tab and click the **Add I/O Signals** button.
2. Add a signal of the type **DigitalInput** and name it **diNewBox**. Check **Auto-reset**.



Add I/O Signals

Type of Signal: DigitalInput	☑ Auto-reset	Number of Signals: 1
Signal Base Name: diNewBox	Start Index: 0	Step: 1
Signal Value: 0	Minimum: 0,00	Maximum: 0,00
Description:	☐ Hidden	☐ Read-only

3. In the SC view, go to the **Signals and Connections** tab and click the **Add I/O Connections** button.
4. Set **Source Object** to **SC_InFeeder**, **Source Signal** to **diNewBox**, **Target Object** to **Source** and **Target Signal** to **Execute**. Press **OK**.

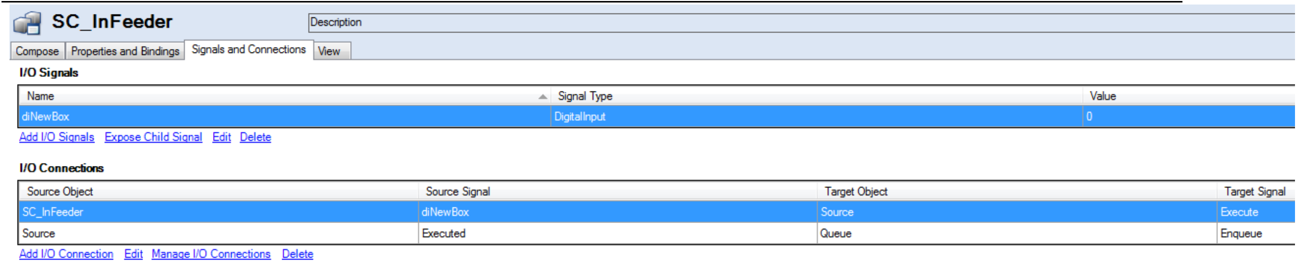


Add I/O Connection

Source Object	SC_InFeeder
Source Signal	diNewBox
Target Object	Source
Target Signal or Property	Execute

☐ Allow cyclic connection

5. Add a new **I/O Connection** and set **Source Object** to **Source**, **Source Signal** to **Executed**, **Target Object** to **Queue** and **Target Signal** to **Enqueue**. Press **OK**.



SC_InFeeder

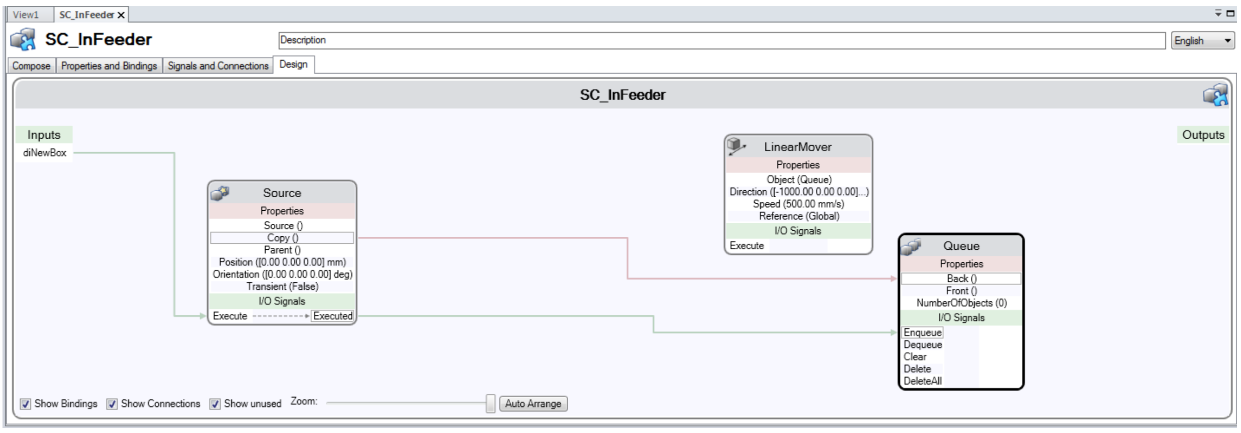
I/O Signals

Name	Signal Type	Value
diNewBox	DigitalInput	0

I/O Connections

Source Object	Source Signal	Target Object	Target Signal
SC_InFeeder	diNewBox	Source	Execute
Source	Executed	Queue	Enqueue

Note: At any point you can also look at the Design view which gives you a graphical representation of the Smart Component. I/O Connections are represented by green lines while bindings are represented by red lines. These can be toggled on and off to simplfy the view if necessary.



6. **Save** the station as *...\my_SC_InFeeder*.

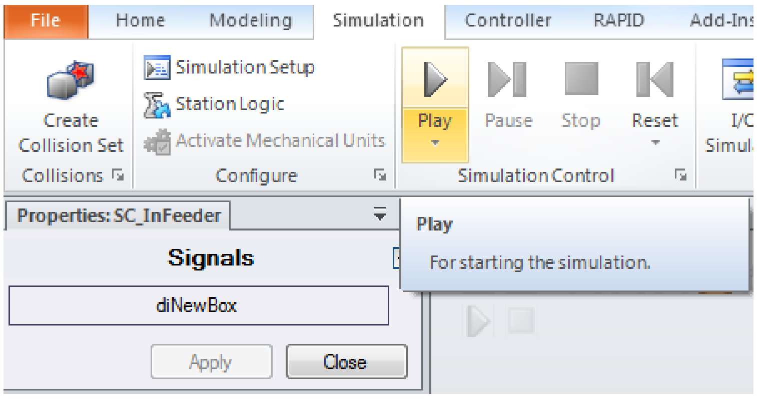
Test running the Smart Component

When you are creating **Smart Components (SC)**, you can minimize the risk of making mistakes by testing it as often as possible. For example, if a sensor is added, you can test the SC by activating the sensor and then moving an object to intersect the beam. The **SensorOut** signal should go high when the beam is intersected.

Before we add a sensor to this SC we will test the **Source** and the **Queue** to ensure it works as expected.

Note: SmartComponents that consume time (e.g. moving objects) are activated by the simulation function within RobotStudio. Therefore a simulation needs to be running in order to test/use a SC.

1. Click on the **my_SC_InFeeder:View1** tab so you can see the conveyor and the box.
2. Open the property window for **SC_InFeeder**.
3. Press **Simulation Play**.

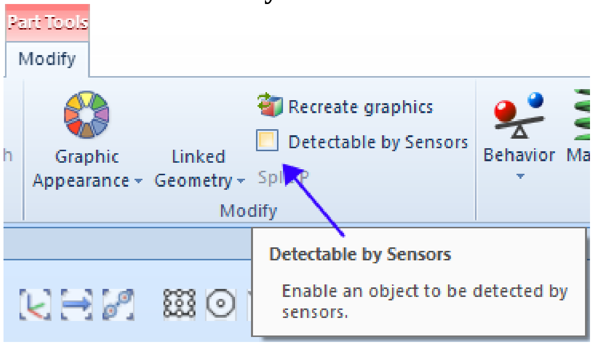


4. Click on **diNewBox** a couple of times and verify that a new box is created and starts moving for each click.
5. Stop the simulation and delete all boxes generated during the simulation.

Adding a Plane Sensor to our Smart Component

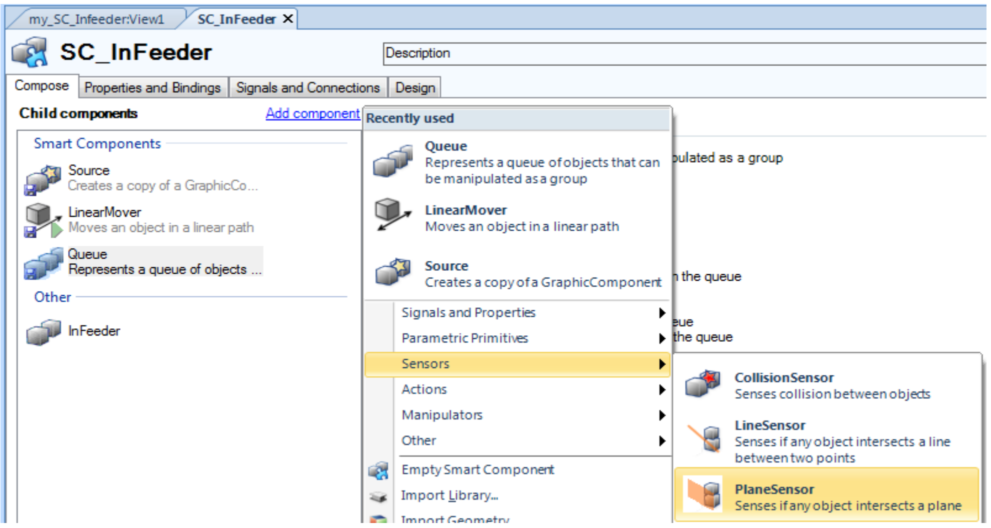
Next we will add a sensor at the end of the conveyor which will stop the box when it hits the sensor beam. We will also add an I/O signal which will remain high as long as a box is in contact with the sensor. Finally we will add an action which generates a new box when a box is removed from the sensor. (We will have the robot pick up the box.)

1. Select the *InFeeder* in the **Layout** browser, then click on **Part Tools-Modify**, and then uncheck **Detectable by Sensors**.

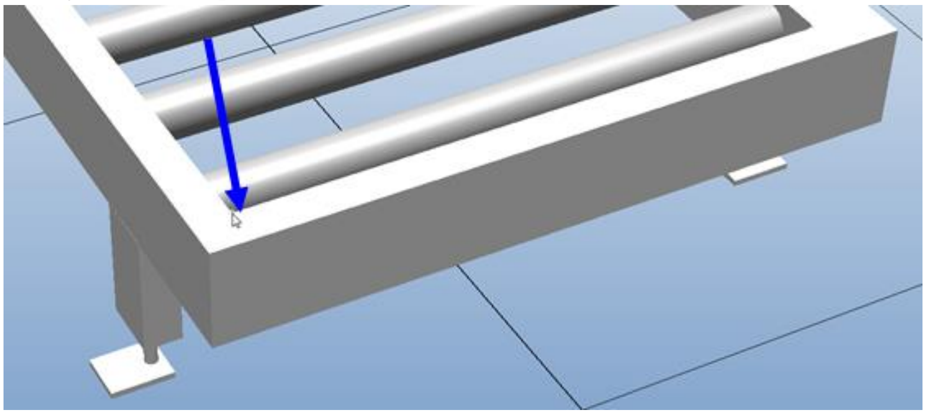


2. This is done to prevent the **PlaneSensor** from detecting the *InFeeder* geometry. It will also prevent the robot's tool from detecting and attaching to the *InFeeder*.

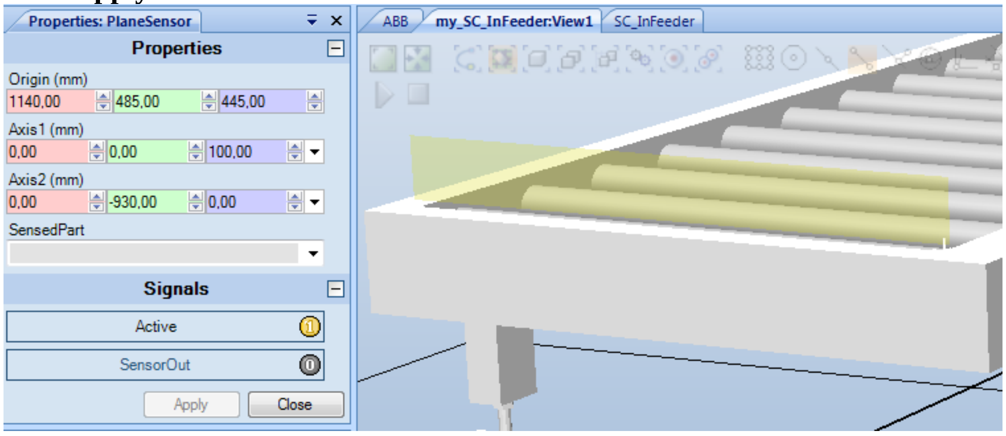
3. In the **Sensors** gallery click on the **PlaneSensor** button.



4. Place the cursor in the **Origin** field in the property window for the **PlaneSensor**.
5. Set up your **selection level** and **snap mode** to enable you to select the corner of the conveyor as in the picture below:



6. Increase the Y value with 10mm. This is to ensure that the sensor is wider than the box and that the entire edge of the box is covered by the sensor so that it is not missed.
7. In the **Axis 1** field type in Z= 100mm (height of the beam).
8. In the **Axis 2** field type in Y= -930mm (width of the beam).
9. Make sure that the sensor is **Active** and that the signal **SensorOut** is low.
10. Press **Apply**.


Properties: PlaneSensor

Properties

Origin (mm): 1140,00 | 485,00 | 445,00

Axis1 (mm): 0,00 | 0,00 | 100,00

Axis2 (mm): 0,00 | -930,00 | 0,00

SensedPart

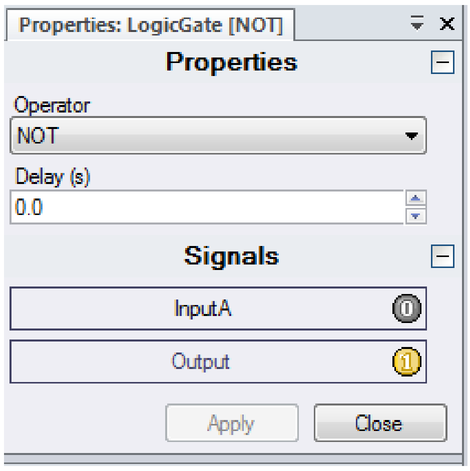
Signals

Active 1

SensorOut 0

11. In the SC view (SC_inFeeder) go to the **Compose** tab and click the **Add component** button.

12. In the **Signals and Properties** gallery click on the **LogicGate** button.
13. Change the **Operator** to **NOT** in the property window for the **LogicGate**.


Properties: LogicGate [NOT]

Properties

Operator: NOT

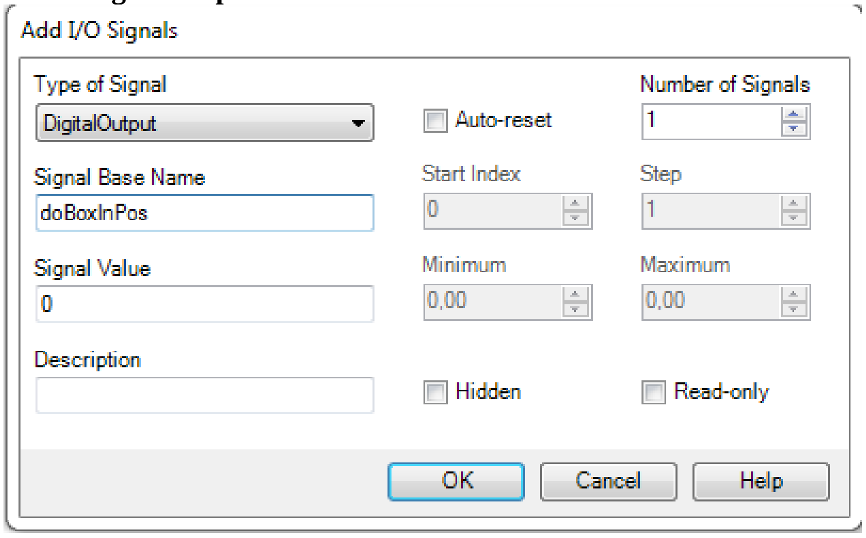
Delay (s): 0.0

Signals

InputA 0

Output 1

14. In the SC view (SC_inFeeder) go to the **Signals and Connections** tab and click the **Add I/O Signals** button.
15. Add a **DigitalOutput** with the name **doBoxInPos**. Press **OK**.


Add I/O Signals

Type of Signal: DigitalOutput

Auto-reset

Number of Signals: 1

Signal Base Name: doBoxInPos

Start Index: 0

Step: 1

Signal Value: 0

Minimum: 0,00

Maximum: 0,00

Description:

Hidden

Read-only

16. Click the **Add I/O Connection** button.
17. Add the I/O connections with the details from the table below.

Source Object	Source Signal	Target Object	Target Signal
PlaneSensor	SensorOut	Queue	Dequeue
PlaneSensor	SensorOut	SC_InFeeder	doBoxInPos
PlaneSensor	SensorOut	LogicGate [NOT]	InputA
LogicGate [NOT]	Output	Source	Execute

Test running and saving the Smart Component

Before we save the SC as a library file we should test run it again.

1. Click on the **my_SC_InFeeder:View1** tab so you can see the conveyor and the box.
2. Open the property window for **SC_InFeeder**
3. Press **Simulation Play**.
4. If a new box not is generated automatically, click on **diNewBox**.
5. Verify that the box stops when it hits the sensor and that **doBoxInPos** goes high.
6. Move the box away from the sensor with **Freehand Move** and verify that **doBoxInPos** goes low and that a new box is generated.
7. Stop the simulation and delete all boxes that were generated during the simulation.
8. Save the **SC_InFeeder** as ... *Libraries \SC_InFeeder.rslib*.

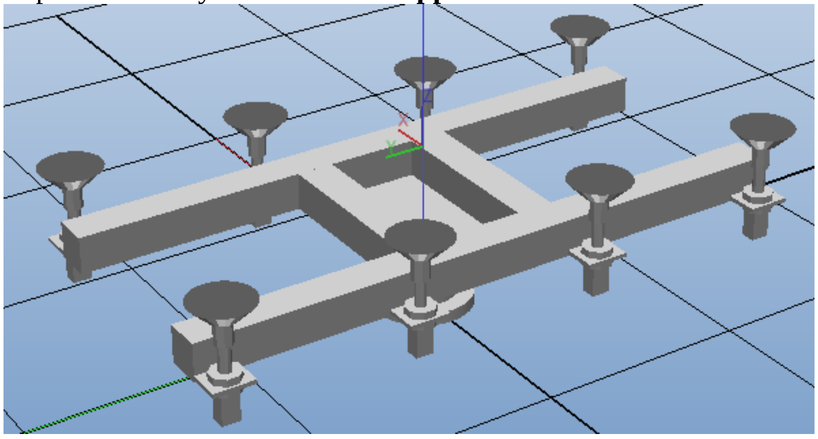
1.2. Create Smart Component - Vacuum Gripper

Overview

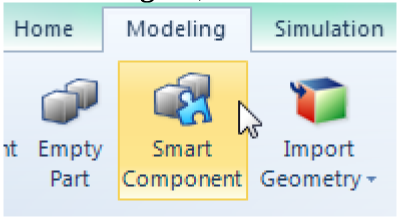
In this exercise we will learn how to create a Smart Component (SC) representing a vacuum gripper.

Preperation

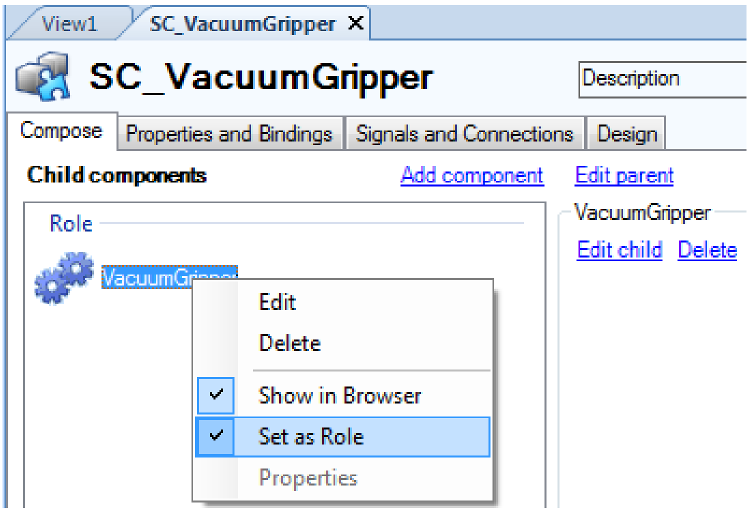
1. Create a new empty station.
2. Import the library file **VacuumGripper.rslib**.



3. As we will modify the library file we need to disconnect the library.
4. In the library context menu, click **Disconnect Library**.
5. In **Modeling** tab, click **Smart Component** to create a new empty SC.



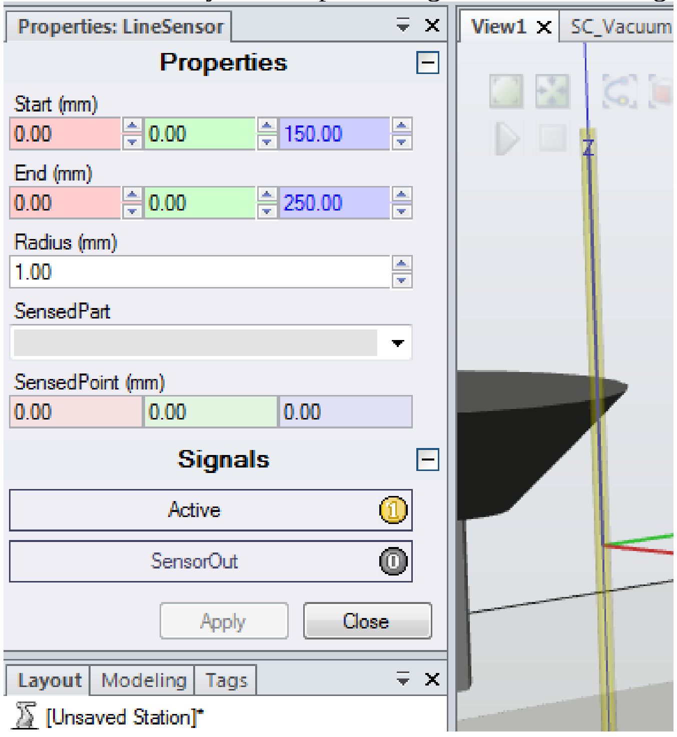
6. Rename the SC to **SC_VacuumGripper**.
7. In Layout browser, drag and drop the vacuum gripper to **SC_VacuumGripper**.
8. In order to get our new component to act as a tool, we need to set the **Vacuum_Gripper** as **role**. In this way the **tooldata** will be created when we attach our **SC gripper** to a robot. In the SC view, set the vacuum gripper as **role** from the conontext menu.



Add Base Components

Now we will start to add base components. We will start with a **sensor** that will react when an object hits the sensor beam.

1. Click **Add component** and then select the **LineSensor** from the **Sensors** gallery.
2. In the **LineSensor** properties window set values as below. After clicking **Apply,** you will see a small cylinder representing the **sensor** in the graphics view.


Properties: LineSensor

Properties

Start (mm): 0.00 | 0.00 | 150.00

End (mm): 0.00 | 0.00 | 250.00

Radius (mm): 1.00

SensedPart:

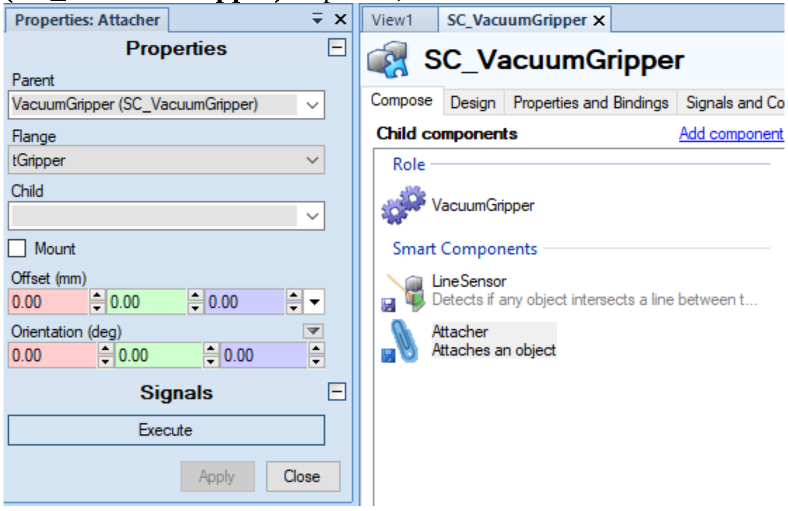
SensedPoint (mm): 0.00 | 0.00 | 0.00

Signals

Active: 1

SensorOut: 0

3. Next add an **Attacher** base component from the **Actions** gallery. When trigged, this base component will attach a **child** component to a **parent** component. The **parent** in this example is represented by the vacuum gripper.
4. In the properties window of the **Attacher**, select the **VacuumGripper (SC_VacuumGripper)** as parent, as shown below.


Properties: Attacher

Properties

Parent: VacuumGripper (SC_VacuumGripper)

Flange: tGripper

Child:

Mount: ☐

Offset (mm): 0.00 | 0.00 | 0.00

Orientation (deg): 0.00 | 0.00 | 0.00

Signals

Execute

SC_VacuumGripper

Child components

Role: VacuumGripper

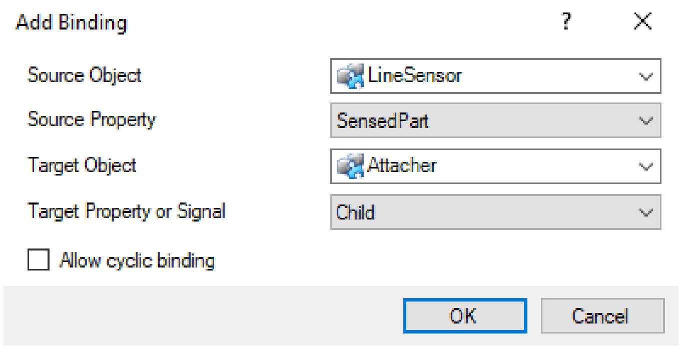
Smart Components:

LineSensor — Detects if any object intersects a line between t...

Attacher — Attaches an object

At this point we cannot state which object should represent the child in the **Attacher.** This depends upon the object that is intersected by the sensor in the simulation. Therefore a binding needs to be made between the object sensed by the **LineSensor** we created earlier.

5. Click **Add Binding** in the **Properties and Bindings** tab of the **SC** view. Create the binding as below:

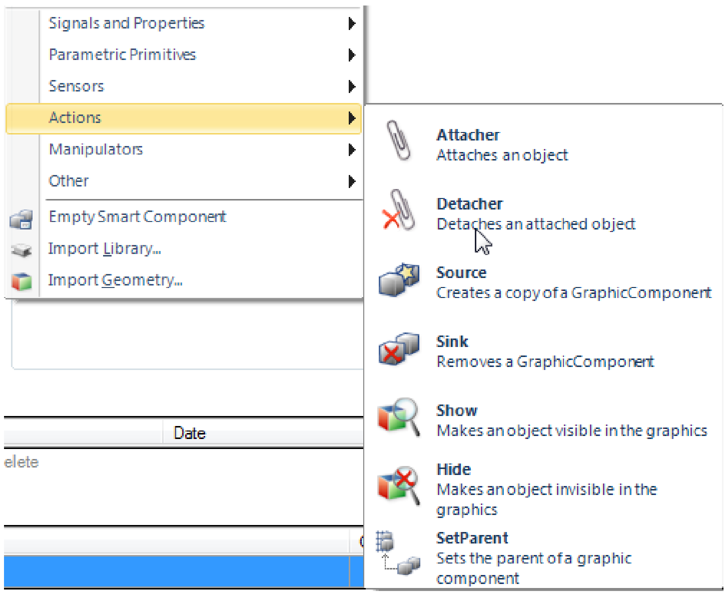

Add Binding

Source Object	LineSensor
Source Property	SensedPart
Target Object	Attacher
Target Property or Signal	Child

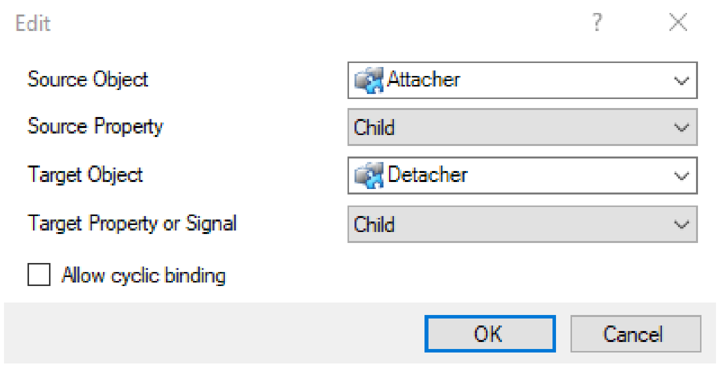
☐ Allow cyclic binding

As we also want to be able to detach the attached object we need to add a **Detacher** and a binding between the attached part and the part that will be detached.

6. From the **Compose** tab and add a **Detacher** base component.


Signals and Properties
Parametric Primitives
Sensors
Actions
Manipulators
Other
Empty Smart Component
Import Library...
Import Geometry...
Attacher
Attaches an object
Detacher
Detaches an attached object
Source
Creates a copy of a GraphicComponent
Sink
Removes a GraphicComponent
Show
Makes an object visible in the graphics
Hide
Makes an object invisible in the graphics
SetParent
Sets the parent of a graphic component
Date
elete

7. Click **Add binding** in the **Properties and Bindings** tab of the SC view again and create the binding as below:


Edit

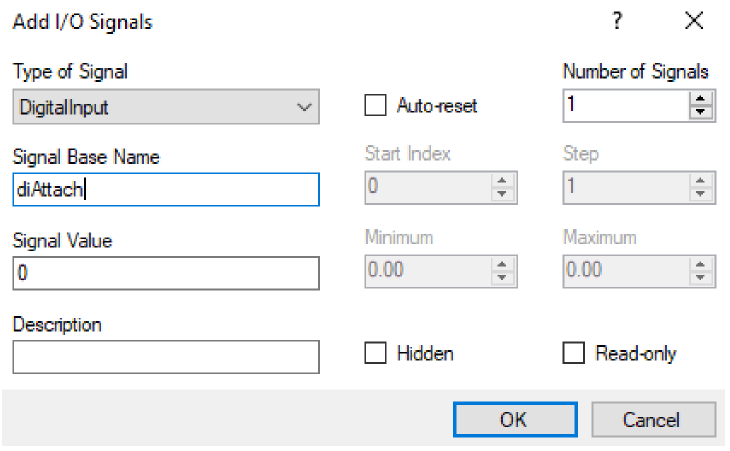
Source Object	Attacher
Source Property	Child
Target Object	Detacher
Target Property or Signal	Child

☐ Allow cyclic binding

Internal Signals

Next we will define internal signals in our component that later will be cross connected with the **I/O** signals in the Virtual Controller. To define the actions in our **SC**, we also need to define some **I/O** connections.

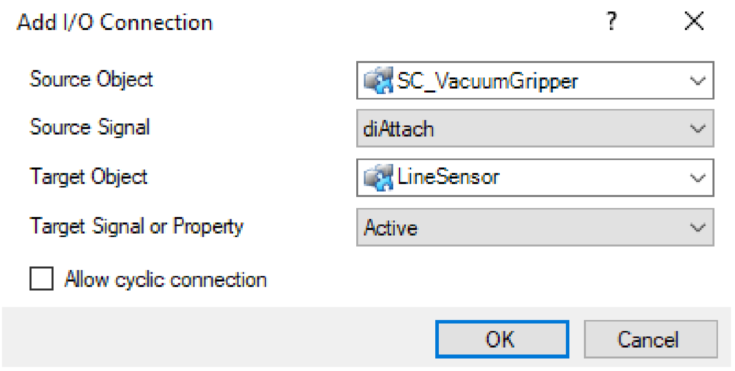
1. In the **Signals and connections** tab of the SC view, click **Add I/O Signals** and add a digital input signal, **diAttach**.



Add I/O Signals

Type of Signal: DigitalInput	Auto-reset: ☐	Number of Signals: 1
Signal Base Name: diAttach	Start Index: 0	Step: 1
Signal Value: 0	Minimum: 0.00	Maximum: 0.00
Description:	Hidden: ☐	Read-only: ☐

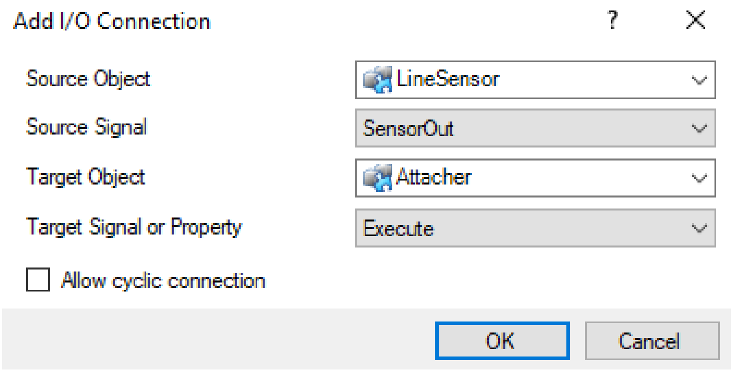
2. Create the first two **I/O connections** with the following input. When the signal is set, the sensor will become active and attach any object breaking the sensor beam.



Add I/O Connection

Source Object	SC_VacuumGripper
Source Signal	diAttach
Target Object	LineSensor
Target Signal or Property	Active

☐ Allow cyclic connection

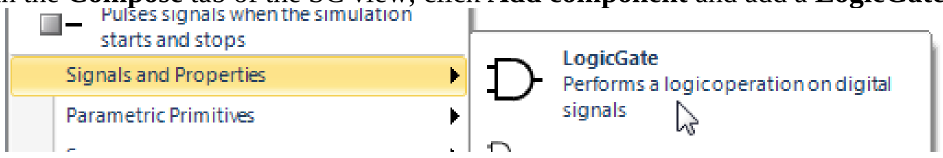


Add I/O Connection

Source Object	LineSensor
Source Signal	SensorOut
Target Object	Attacher
Target Signal or Property	Execute

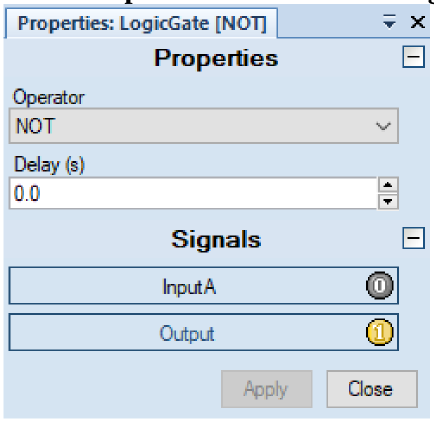
☐ Allow cyclic connection

3. When the signal **diAttach** is set low, we want the **Detacher** to execute. Instead of creating a new **I/O** signal we will add a new base component, **LogicGate (NOT).** This will trigger the **Detacher** when the signal is low (**NOT** high).
4. In the **Compose** tab of the SC view, click **Add component** and add a **LogicGate**.



Pulses signals when the simulation starts and stops
Signals and Properties
Parametric Primitives
LogicGate
Performs a logicoperation on digital signals

5. In the **Properties** window of the **LogicGate**, change the **Operator** to **NOT**.



Properties: LogicGate [NOT]

Properties

Operator: NOT

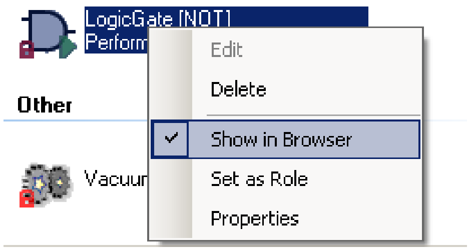
Delay (s): 0.0

Signals

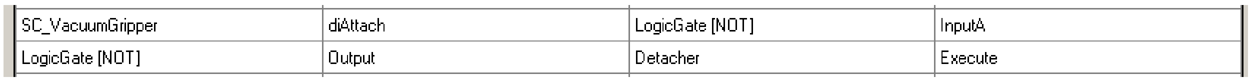
InputA 0

Output 1

6. If you want the component visible in the **Layout** browser, select **Show in Browser** from the context menu.



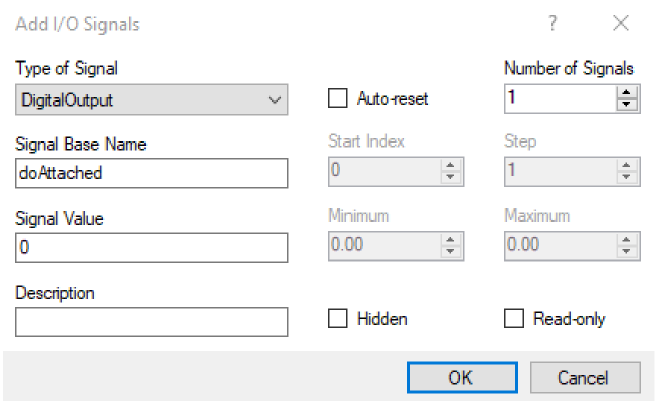
7. Add two more **I/O connections** according to the list below. (Ensure you understand the steps here as this represents the logic discussed in step 3.)



SC_VacuumGripper	diAttach	LogicGate [NOT]	InputA
LogicGate [NOT]	Output	Detacher	Execute

The next step is to create a handshake output signal (SC internal), **doAttached** from our **SC.** This will later be connected to a digital input signal in the robot controller confirming whether something is attached or not. This signal will be controlled by a **SRLatch** (Set-Reset latch) which will be set by the **Attacher** and reset by the **Detacher.**

8. In the **Signals and connections** tab of the **SC** view**,** add a digital output signal, **doAttached.**



Add I/O Signals

Type of Signal: DigitalOutput

Auto-reset: ☐

Number of Signals: 1

Signal Base Name: doAttached

Start Index: 0

Step: 1

Signal Value: 0

Minimum: 0.00

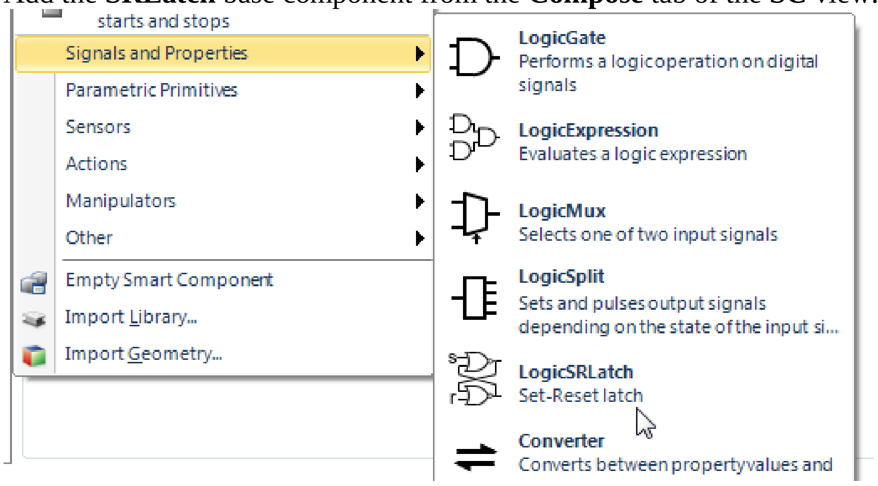
Maximum: 0.00

Description:

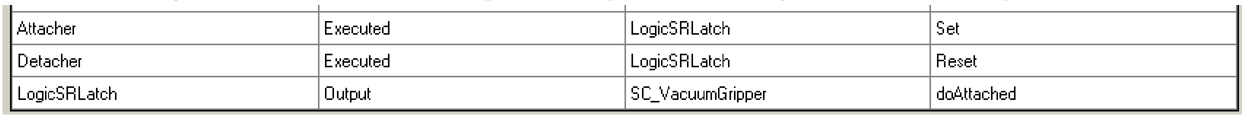
Hidden: ☐

Read-only: ☐

9. Add the **SRLatch** base component from the **Compose** tab of the **SC** view.

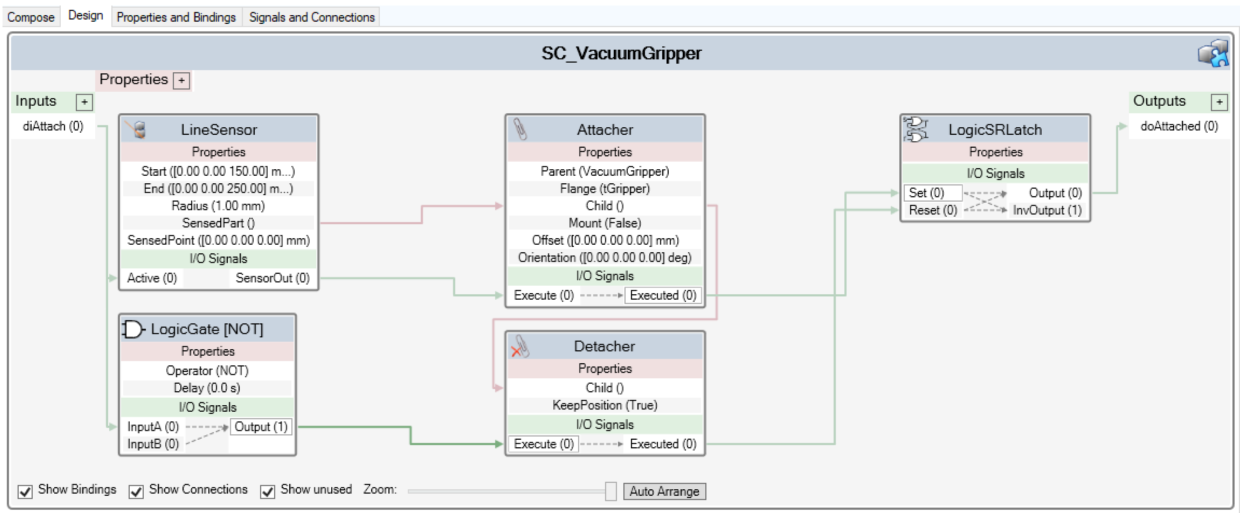


10. Then continue adding **I/O connections** according to the list below. Make sure you go through and understand each step. (This represents the logic discussed in step 7.)



Attacher	Executed	LogicSRLatch	Set
Detacher	Executed	LogicSRLatch	Reset
LogicSRLatch	Output	SC_VacuumGripper	doAttached

11. To get an overview of the completed **SC**, click the **Design** tab of the **SC** view. To get a better view click on **Auto Arrange**. Note that you can also drag the various compnents around to arrange them according to your preference. You can also make **I/O connections** and **Bindings** directly in the **Design** tab.



12. Save the **SC_VacuumGripper** as ... *Libraries \SC_VacuumGripper.rslib.*

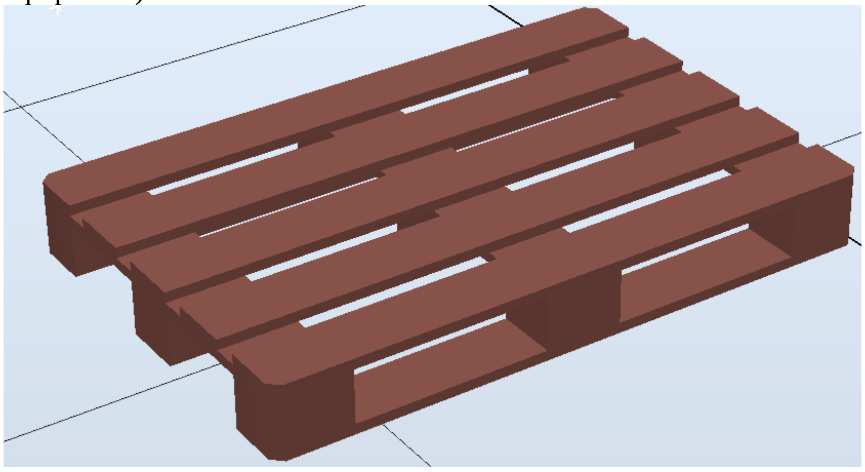
1.3. Create Smart Component - Out Pallet

Overview

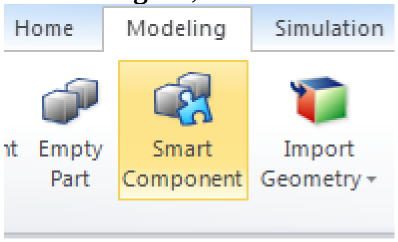
In this exercise we will learn how to create a Smart Component (SC) representing an outfeed pallet.

Preparation

1. Create a new empty station.
2. Import the library file **EuroPallet.rslib**. (Note this is in the standard ABB library under equipment.)



3. Set the pallet’s position to X=0, Y=1100, Z=0.
4. As we will modify the library file we need to disconnect the library.
5. In the library context menu, click **Disconnect Library**.
6. In **Modeling** tab, click **Smart Component** to create a new empty SC.

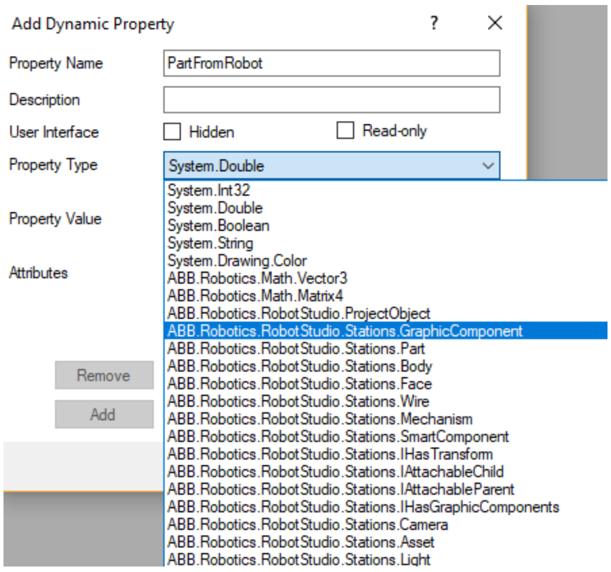


7. Rename the SC to **SC_OutPallet**.
8. In the **Layout** browser, drag and drop the Euro Pallet onto the *SC_OutPallet*.

Add Base Components

We will start with a **Sink** for our first base component. This component enable the removal of a graphic component. (In this example it will be the box the robot places on the pallet.)

1. Click **Add component** and then select the **Sink** from the **Actions** gallery.
2. Open up the properties and bindings tab and select add a **Dynamic Property.**
3. Give it the name **PartFromRobot** and under property type select **GraphicComponent.**



Add Dynamic Property

Property Name: PartFromRobot

Description:

User Interface: ☐ Hidden ☐ Read-only

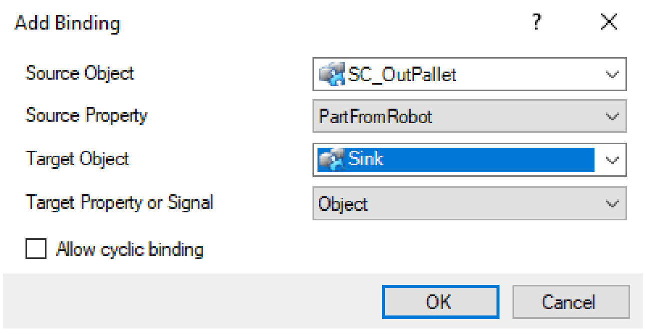
Property Type: System.Double

Property Value

Attributes

System.Int32
System.Double
System.Boolean
System.String
System.Drawing.Color
ABB.Robotics.Math.Vector3
ABB.Robotics.Math.Matrix4
ABB.Robotics.RobotStudio.ProjectObject
ABB.Robotics.RobotStudio.Stations.GraphicComponent
ABB.Robotics.RobotStudio.Stations.Part
ABB.Robotics.RobotStudio.Stations.Body
ABB.Robotics.RobotStudio.Stations.Face
ABB.Robotics.RobotStudio.Stations.Wire
ABB.Robotics.RobotStudio.Stations.Mechanism
ABB.Robotics.RobotStudio.Stations.SmartComponent
ABB.Robotics.RobotStudio.Stations.IHasTransform
ABB.Robotics.RobotStudio.Stations.IAttachableChild
ABB.Robotics.RobotStudio.Stations.IAttachableParent
ABB.Robotics.RobotStudio.Stations.IHasGraphicComponents
ABB.Robotics.RobotStudio.Stations.Camera
ABB.Robotics.RobotStudio.Stations.Asset
ABB.Robotics.RobotStudio.Stations.Light

4. Next add a binding to bind the **Sink** with the **Dynamic Property**. (In this case it will be the part dropped off by the robot.)



Add Binding

Source Object: SC_OutPallet

Source Property: PartFromRobot

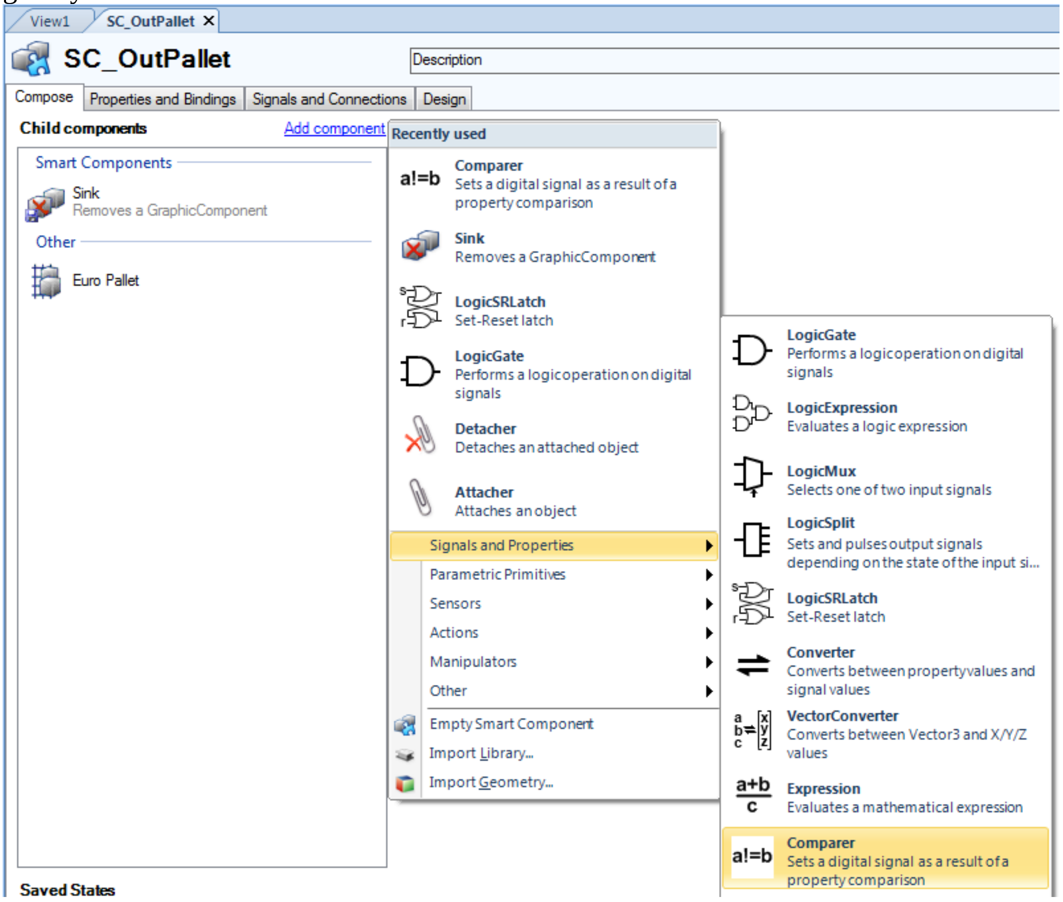
Target Object: Sink

Target Property or Signal: Object

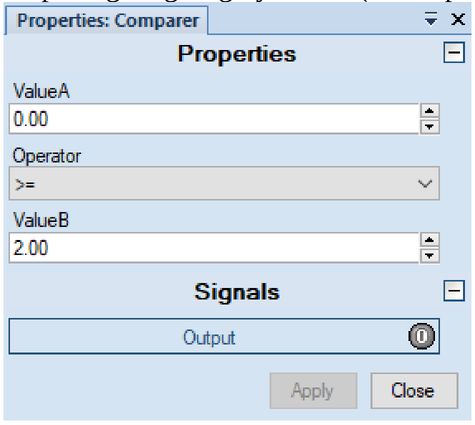
☐ Allow cyclic binding

For the next base component we will add a **Comparer**. This component enables the ability to set a signal based upon the result of a property comparison.

5. Add the base component **Comparer** which is found under the **Signal and Properties** gallery.


View1
SC_OutPallet
SC_OutPallet
Description
Compose
Properties and Bindings
Signals and Connections
Design
Child components
Add component
Smart Components
Sink
Removes a GraphicComponent
Other
Euro Pallet
Saved States
Recently used
Comparer
Sets a digital signal as a result of a property comparison
Sink
Removes a GraphicComponent
LogicSRLatch
Set-Reset latch
LogicGate
Performs a logic operation on digital signals
Detacher
Detaches an attached object
Attacher
Attaches an object
Signals and Properties
Parametric Primitives
Sensors
Actions
Manipulators
Other
Empty Smart Component
Import Library...
Import Geometry...
LogicGate
Performs a logic operation on digital signals
LogicExpression
Evaluates a logic expression
LogicMux
Selects one of two input signals
LogicSplit
Sets and pulses output signals depending on the state of the input si...
LogicSRLatch
Set-Reset latch
Converter
Converts between property values and signal values
VectorConverter
Converts between Vector3 and X/Y/Z values
Expression
Evaluates a mathematical expression
Comparer
Sets a digital signal as a result of a property comparison

6. In the **Comparer** properties window set values as below and click **Apply**. Note that the output signal gets greyed out. (see step 8)



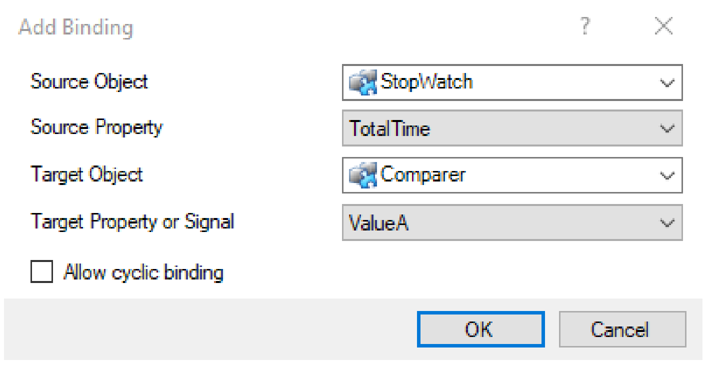
ValueA
0.00

Operator
>=

ValueB
2.00

For the next base component we will add a **Stop Watch.**

7. Add a **Stop Watch** which is also found in the **Signals and Properties** gallery.
8. Add a binding to connect the **Stop Watch** to the **Comparer**. In this example by making the binding, the **Comparer** will set an output after 2 seconds of simulation time. Ultimately we will use this to remove/delete the part that will be placed on the pallet.

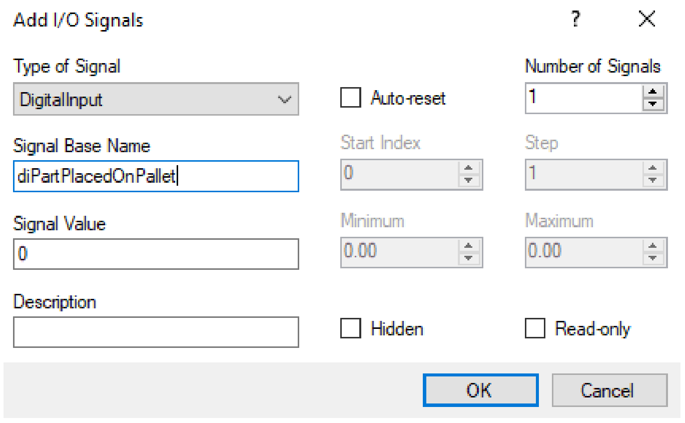

Add Binding

Source Object	StopWatch
Source Property	TotalTime
Target Object	Comparer
Target Property or Signal	ValueA

☐ Allow cyclic binding

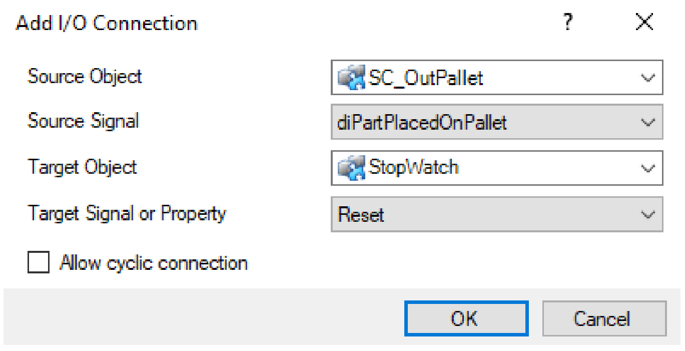
For the next base component we will add a **LogicSRLatch.**

1. Add a **LogicSRLatch** which is again found in the **Signals and Properties** gallery.
2. Next add an I/O signal of the type **DigitalInput** as shown below.


Add I/O Signals

Type of Signal	DigitalInput
Auto-reset	☐
Number of Signals	1
Signal Base Name	diPartPlacedOnPallet
Start Index	0
Step	1
Signal Value	0
Minimum	0.00
Maximum	0.00
Description	
Hidden	☐
Read-only	☐

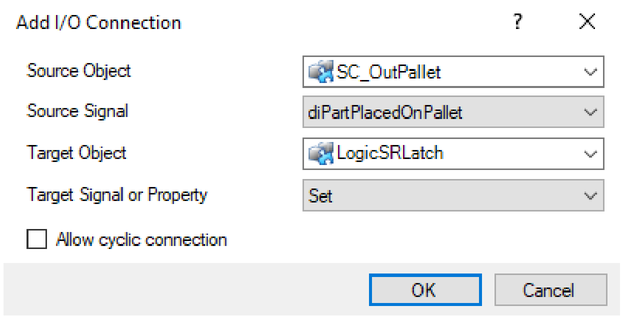
3. Add an **I/O connection** to reset the **Stop Watch** when the digital input goes high.


Add I/O Connection

Source Object	SC_OutPallet
Source Signal	diPartPlacedOnPallet
Target Object	StopWatch
Target Signal or Property	Reset

☐ Allow cyclic connection

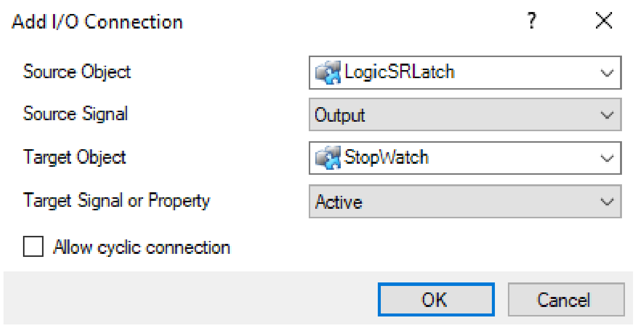
4. Add another **I/O connection** to activate the **LogicSRLatch** when the digital input goes high.


Add I/O Connection

Source Object	SC_OutPallet
Source Signal	diPartPlacedOnPallet
Target Object	LogicSRLatch
Target Signal or Property	Set

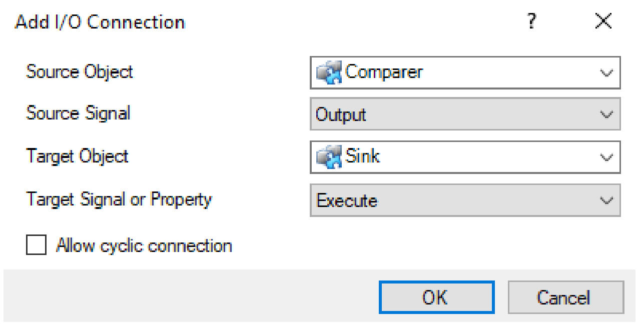
☐ Allow cyclic connection

5. Add another **I/O connection** to connect the **LogicSRLatch Output** signal to the **StopWatch Active** signal. This to activate the **StopWatch** when the digital input goes high.


Add I/O Connection

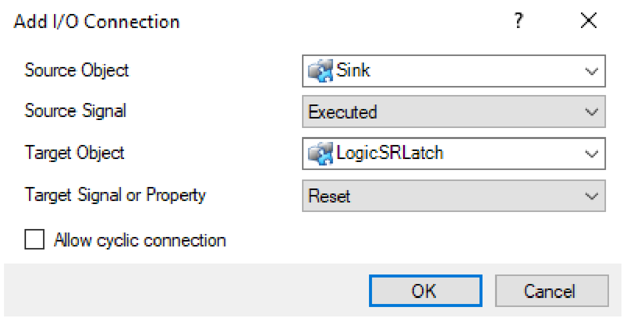
Source Object: LogicSRLatch
Source Signal: Output
Target Object: StopWatch
Target Signal or Property: Active
☐ Allow cyclic connection

6. Add another **I/O connection** to connect the **Comparer** and the **Sink.** This will delete the object when the **Comparer Output** signal goes high (after 2 seconds).


Add I/O Connection

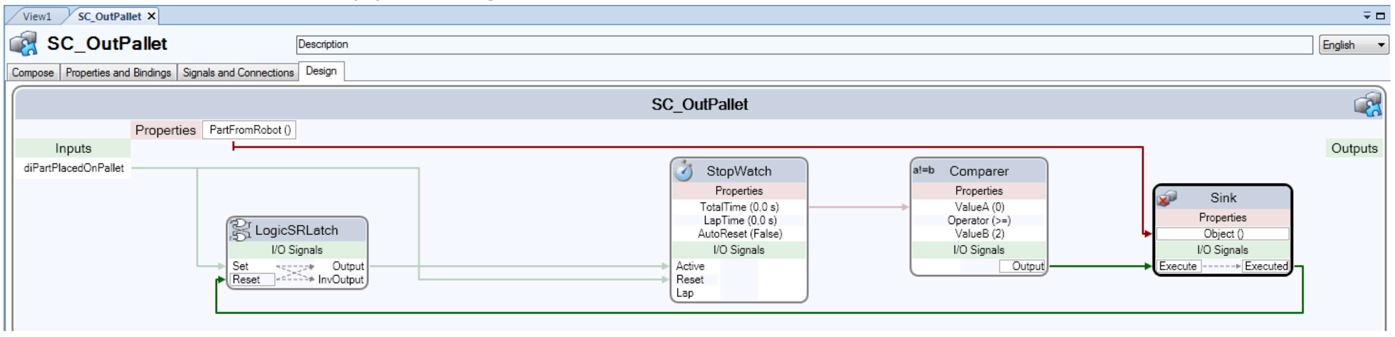
Source Object: Comparer
Source Signal: Output
Target Object: Sink
Target Signal or Property: Execute
☐ Allow cyclic connection

7. Add the final **I/O connection** to deactivate the **StopWatch** when the object is deleted**.**


Add I/O Connection

Source Object: Sink
Source Signal: Executed
Target Object: LogicSRLatch
Target Signal or Property: Reset
☐ Allow cyclic connection

8. Verify your design is as shown below.



14.Save the **SC_OutPallet** as … *Libraries \SC_OutPallet.rslib.*

1.4. Complete the Palletizing Simulation

Overview

In the next exercise we will complete a palletizing simulation by building a station with our smart components along with a robot system.

Preperation

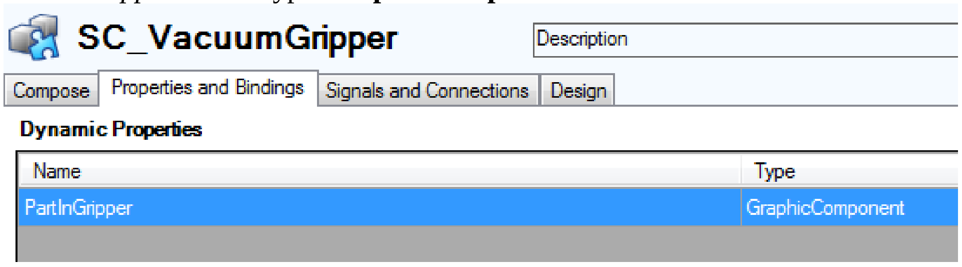
1. Unpack the file *Courseware\Stations\SC_Palletizing_Robot.rspag* to a new folder *Stations\Module_7|my_SC_Palletizing*.
2. Import the ...*\Libraries \SC_VacuumGripper.rslib* created earlier in this exercise.
3. Import the ...*\Libraries \SC_InFeeder.rslib* created earlier in this exercise.*
4. Import the ...*\Libraries \SC_OutPallet.rslib* created earlier in this exercise.
5. Import the library ...*\Libraries\box.rslib.**

***Note:** When a SC is saved as a library component any links that may have existing during its creation are removed. This gives us the ability to reuse SC in other stations and simulations. When this is done some of the base components may need to be connected, bound etc. to another component.
Hint: In the SC_InFeeder what graphical component are we looking for the **Source** to copy?

Modify the gripper

To be able to delete the object that the robot has placed on the out pallet, we need to modify the Smart Component *SC_VacuumGripper*.

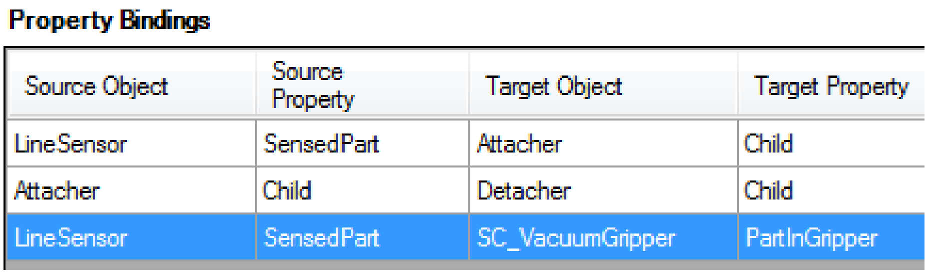
1. Disconnect *SC_VacuumGripper* from library.
2. Right click on *SC_VacuumGripper* and select **Edit Component** in the context menu.
3. In the **Properties and Bindings** tab, add a new **Dynamic Property** with the name *PartInGripper* of the type **GraphicComponent**.



Dynamic Properties

Name	Type
PartInGripper	GraphicComponent

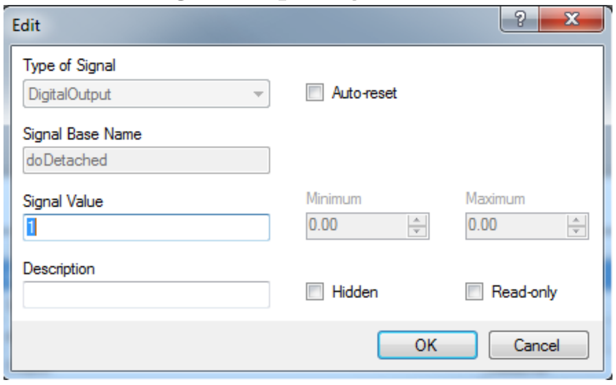
4. Bind the property *LineSensor – SensedPart* to the new property.



Property Bindings

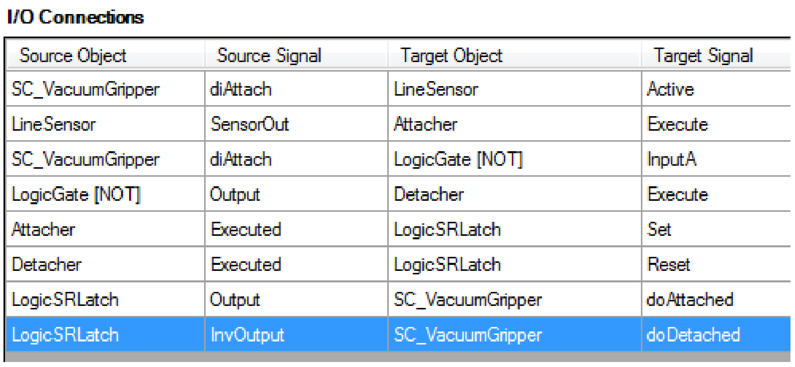
Source Object	Source Property	Target Object	Target Property
LineSensor	SensedPart	Attacher	Child
Attacher	Child	Detacher	Child
LineSensor	SensedPart	SC_VacuumGripper	PartInGripper

5. Add a new **DigitalOutput** signal with an initial value of 1 and name it *doDetached*.



Type of Signal: DigitalOutput
Signal Base Name: doDetached
Signal Value: 1
Minimum: 0.00
Maximum: 0.00
Description:

6. Make an **I/O Connection** from the *LogicSRLatch –InvOutput* to *doDetached*.



I/O Connections

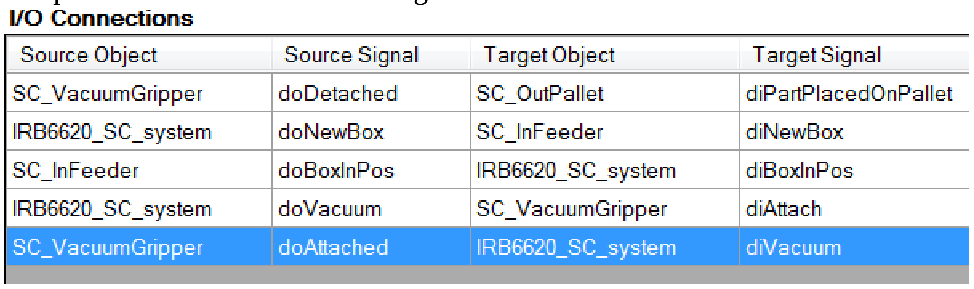
Source Object	Source Signal	Target Object	Target Signal
SC_VacuumGripper	diAttach	LineSensor	Active
LineSensor	SensorOut	Attacher	Execute
SC_VacuumGripper	diAttach	LogicGate [NOT]	InputA
LogicGate [NOT]	Output	Detacher	Execute
Attacher	Executed	LogicSRLatch	Set
Detacher	Executed	LogicSRLatch	Reset
LogicSRLatch	Output	SC_VacuumGripper	doAttached
LogicSRLatch	InvOutput	SC_VacuumGripper	doDetached

7. Close the Smart Component Editor window and save the gripper with the same name as before.
8. Attach *SC_VacuumGripper* to the robot.

Station Logic setup

The last step is to setup I/O connections between the Smart Components and the Virtual Controller to get a complete simulation. This can be thought of as “virtual wiring” between the various components in the simulation environment whereas in the real application there would be physical wiring in place between the actual components.

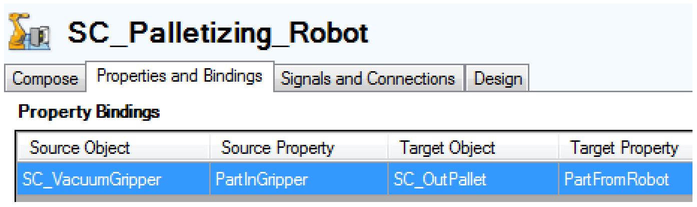
1. Setup the I/O connections according to the list below.



I/O Connections

Source Object	Source Signal	Target Object	Target Signal
SC_VacuumGripper	doDetached	SC_OutPallet	diPartPlacedOnPallet
IRB6620_SC_system	doNewBox	SC_InFeeder	diNewBox
SC_InFeeder	doBoxInPos	IRB6620_SC_system	diBoxInPos
IRB6620_SC_system	doVacuum	SC_VacuumGripper	diAttach
SC_VacuumGripper	doAttached	IRB6620_SC_system	diVacuum

2. Add a binding from *SC_VacuumGripper - partInGripper* to *SC_OutPallet - PartFromRobot*.



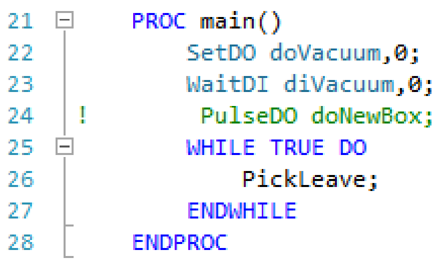
Property Bindings

Source Object	Source Property	Target Object	Target Property
SC_VacuumGripper	PartInGripper	SC_OutPallet	PartFromRobot

Run the Simulation

1. Make sure all I/O signals (including all SC) have the correct start values***** and save the current state.
2. Add the saved state in **Simulation Setup** and then run the simulation. Refer to Module 5 for assistance if required.
3. Save the station as a *station file* (.rsstn) and as a *Pack and Go* file (.rspag)

***Note:** Depending upon the exact signal values you start with multiple boxes may be created when the simulation is started. One option to correct this may be to change the Rapid code. In the main procedure an I/O is being pulsed in order to create a new box. The SC_InFeeder is also using a logic gate to create a new part when the sensor is low at the end of the conveyor.



```
PROC main()
    SetDO doVacuum,0;
    WaitDI diVacuum,0;
!    PulseDO doNewBox;
    WHILE TRUE DO
        PickLeave;
    ENDWHILE
ENDPROC
```

© Copyright 2017 ABB All right reserved.
ABB AB
Robotics Products
SE-721 68 Västerås
Sweden