| | |
|---|---|
| Sekolah: **SEKOLAH KEJURUTERAAN ELEKTRIK** | |
| Nama Matapelajaran: Makmal Kejuruteraan Elektrik<br>Kod Matapelajaran : SKEM 3742 | Semakan : 1<br>Tarikh Keluaran : 2020<br>Pindaan Terakhir : 2020<br>No. Prosedur : **PK-UTM-FKE-(0)-10** |

# SKEM 3742

## SEKOLAH KEJURUTERAAN ELEKTRIK
## FAKULTI KEJURUTERAAN
## UNIVERSITI TEKNOLOGI MALAYSIA
## KAMPUS SKUDAI
## JOHOR

ROBOTICS LAB

# ADDITIONAL MATERIALS
# RobotStudio Pick & Place Exercise

# INDUSTRIAL ROBOTS

| | | | |
|---|---|---|---|
| Disediakan oleh | | Disahkan oleh | : Pengarah |
| Nama | : Dr. Mohd Ridzuan Ahmad<br>Dr. Mohamad Hafis Izran Ishak<br>En. Ahmad Ridhwan Wahab<br>Dr. Mohamad Amir Shamsudin | Nama | : P.M. Ir. Dr. Norhaliza Hj. Abd Wahab |
| Tandatangan | : | Tandatangan | : |
| Cop | : | Cop | : |
| Tarikh | : 27 April 2020 | Tarikh | : 27 April 2020 |

## Content

### Creating a basic station

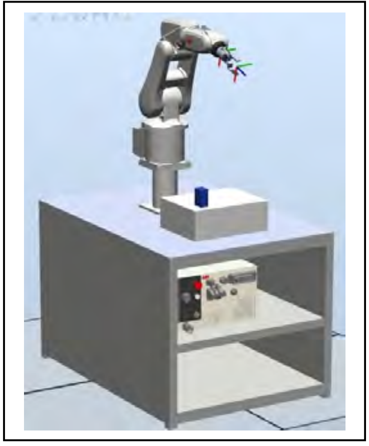
## Smart Component

## Programming and simulating I/O signals

# Creating a basic station

## Goal of the chapter

- In these exercises we will learn how to build a basic station containing a robot, a tool, fixtures and a work pieces as shown in the picture. Later we will program the robot to pick and place the work piece.



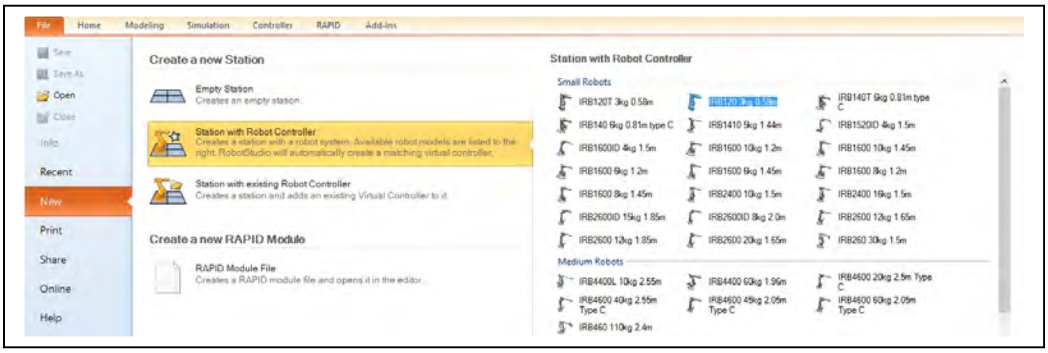
## 1.1. New station

### 1.1.1 Start new station

1. Open the **RobotStudio 5.61 (64-bit)** application

2. In the **File** tab, click **New** and then select a new **Station with Robot Controller**.

3. By default, template systems are listed in the right pane of the window. Select **IRB120_3kg 0.58m** and then click **create**.



Create a new Station

Empty Station
Creates an empty station.

Station with Robot Controller
Creates a station with a robot system. Available robot models are listed to the right. RobotStudio will automatically create a matching virtual controller.

Station with existing Robot Controller
Creates a station and adds an existing Virtual Controller to it.

Create a new RAPID Module

RAPID Module File
Creates a RAPID module file and opens it in the editor.

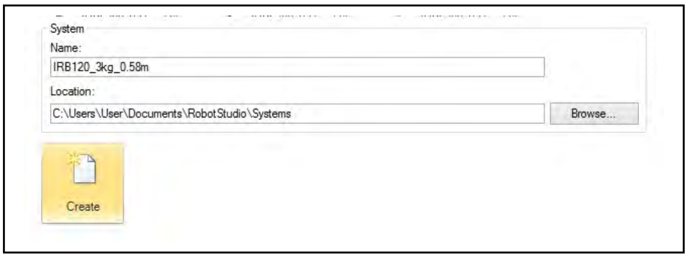
Station with Robot Controller

Small Robots

IRB120T 3kg 0.58m | IRB120 3kg 0.58m | IRB140T 6kg 0.81m type C
IRB140 6kg 0.81m type C | IRB1410 5kg 1.44m | IRB1520ID 4kg 1.5m
IRB1600ID 4kg 1.5m | IRB1600 10kg 1.2m | IRB1600 10kg 1.45m
IRB1600 6kg 1.2m | IRB1600 6kg 1.45m | IRB1600 8kg 1.2m
IRB1600 8kg 1.45m | IRB2400 10kg 1.5m | IRB2400 16kg 1.5m
IRB2600ID 15kg 1.85m | IRB2600ID 8kg 2.0m | IRB2600 12kg 1.65m
IRB2600 12kg 1.85m | IRB2600 20kg 1.65m | IRB260 30kg 1.5m

Medium Robots

IRB4400L 10kg 2.55m | IRB4400 60kg 1.96m | IRB4600 20kg 2.5m Type C
IRB4600 40kg 2.55m Type C | IRB4600 45kg 2.05m Type C | IRB4600 60kg 2.05m Type C
IRB460 110kg 2.4m

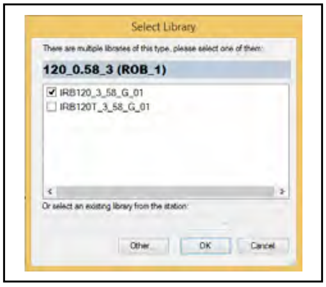


System

Name: IRB120_3kg_0.58m

Location: C:\Users\User\Documents\RobotStudio\Systems

4. When you are prompted to *select library*, choose **IRB120_3_58_G_01**



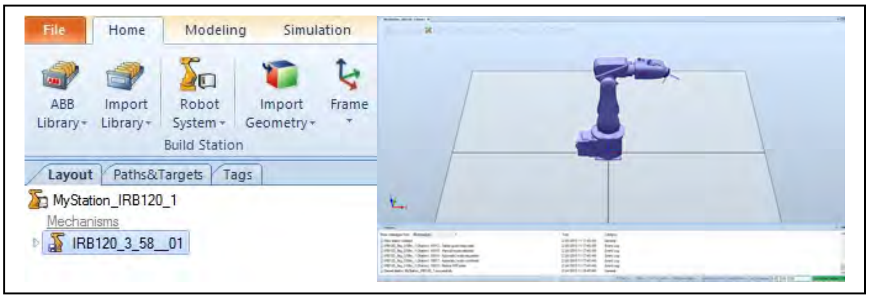
Select Library

There are multiple libraries of this type, please select one of them:

120_0.58_3 (ROB_1)

☑ IRB120_3_58_G_01
□ IRB120T_3_58_G_01

Or select an existing library from the station:



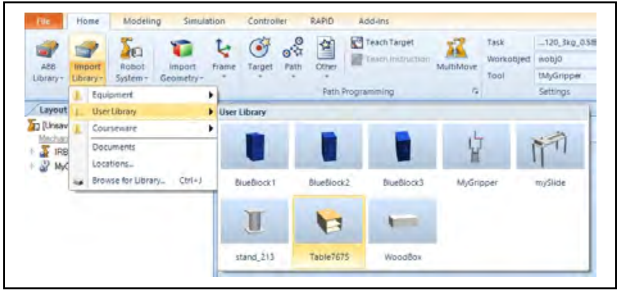
*A station with IRB120 robot have been created*

5. In the **File** tab select **Save As**.
6. Browse to the folder *C:\Users\0wner\Desktop\ABBexperiment\SxGx* and save the station as *MyABBpick&place_SxGx_s1*. (*if you are from* ***section 1*** *and* ***group 2****, save the file in* **S1G2** folder. *Create the folder if it does not exist*).

## 1.1.2 Adding a tool

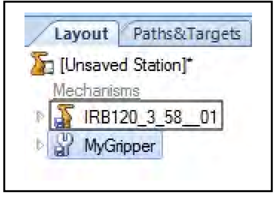
### a) Importing the tool

1. Open the station from the last exercise (*MyABBpick&place_SxGx_s1*), unless it is already open.
2. In the **Home** tab click the **Import Library** button. Click the **User Library** and select *MyGripper*.

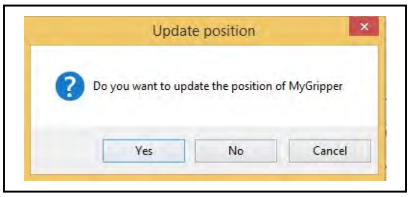


### b) Attaching the tool

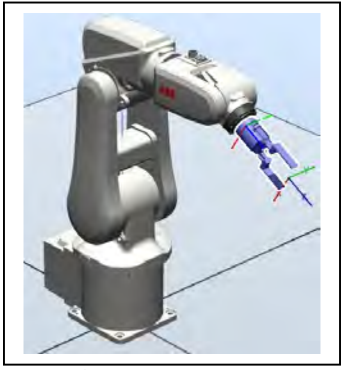
3. Inside the **Layout** browser, drag the tool *MyGripper* and drop it on the robot *IRB120*.



4. Answer the question “Do you want to update the position of MyGripper” with **Yes**.


Update position

Do you want to update the position of MyGripper



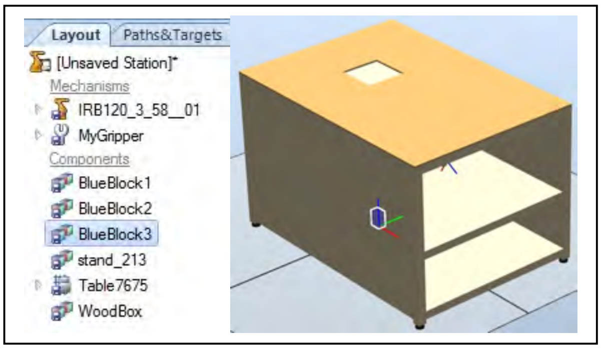
*The tool is moved to the wrist of the robot.*

5. **Save** the station as *MyABBpick&place_SxGx_s2*.

## 1.1.3 Importing and arranging

### a) Importing fixtures and work piece

1. Open the station from the last exercise (*MyABBpick&place_SxGx_s2*), unless it is already open.
2. On the **Home** tab click the **Import Library** button. Select *Table*, *Stand*, *WoodBox* and *BlueBlocks* from the **User Library**.



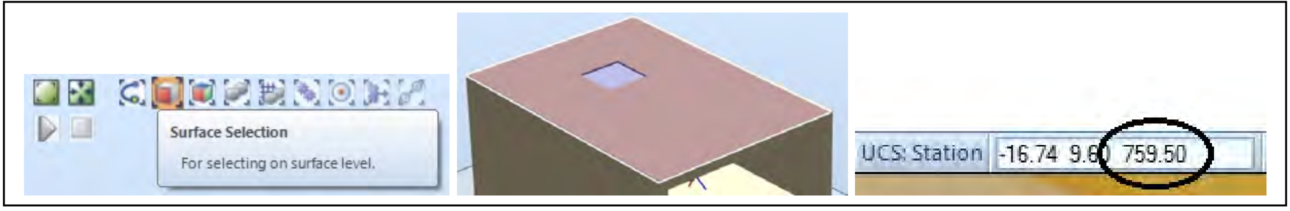
*All components have been inserted in the station*

## b) Positioning the stand

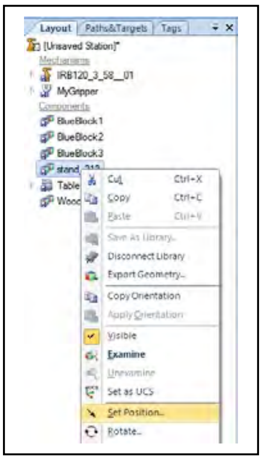
**Overview:**

- The next step is to put the stand on the table.
- The snap mode and selection level tools can be extremely useful when working with geometry.

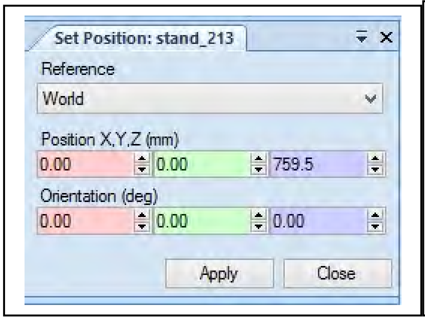
3. Set the selection level to **surface** and click on the top of the table. In the coordinates output (bottom right corner of the screen) we can see the height of the top of the table is *759.50mm.*



4. In the **Layout** browser right click the *stand* and select **Set Position**.



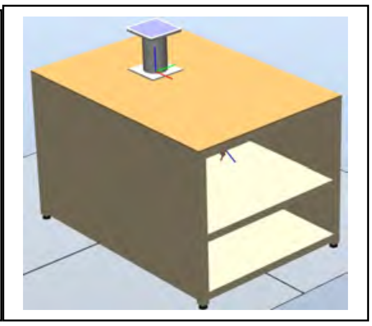
5. In the **Set position** dialog, in the **Reference** list select the **World** coordinate system.
6. In the **Position** fields enter these values **0, 0, 759.50**.
7. In the **Orientation** fields enter these values **0, 0, 0**.
8. Click **Apply**


Set Position: stand_213

Reference: World

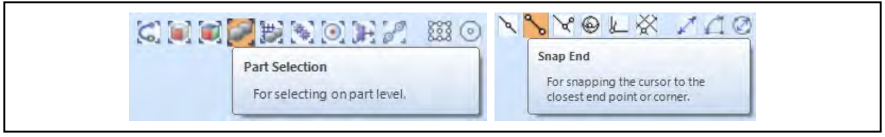
Position X,Y,Z (mm): 0.00 0.00 759.5

Orientation (deg): 0.00 0.00 0.00

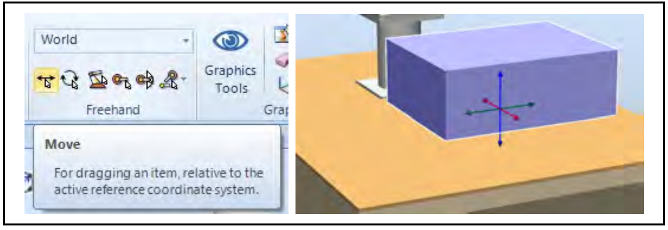


## c) Placing the wood box

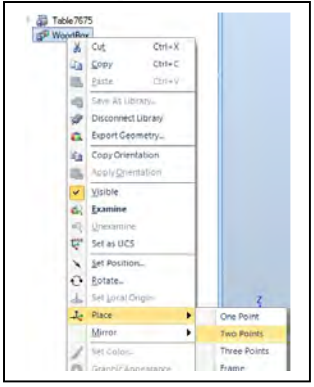
9. In the Graphics window select the **Part Select** level and the **Snap End** mode.


Part Selection
For selecting on part level.
Snap End
For snapping the cursor to the closest end point or corner.

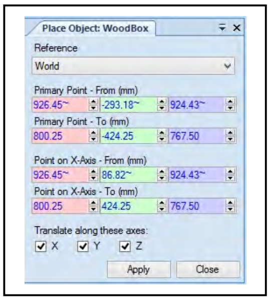
10. On **Freehand**, select **Move** to drag the table as in picture below


World
Graphics Tools
Freehand
Grap
Move
For dragging an item, relative to the active reference coordinate system.

11. Rotate and zoom the station so you get a clear view of the *wood box* and the *corners* of the *table*.
12. In the **Layout** browser right click the *Woodbox* part, point to **Place** select **Two Points**.



13. When the text insertion point is positioned in any of the **Primary point - From** boxes, click the corner of the *WoodBox* marked with **1** in the picture.
14. When the text insertion point is positioned in any of the **Primary point - To** boxes, click the corner of the *table* marked with **2**.
15. When the text insertion point is positioned in any of the **Point on X-Axis -From** boxes, click the corner of the *WoodBox* marked with **3**.
16. When the text insertion point is positioned in any of the **Point on X-Axis -To** boxes, click the corner of the *table* marked with **4**.


Place Object: WoodBox

Reference

World

Primary Point - From (mm)

926.45~ | -293.18~ | 924.43~

Primary Point - To (mm)

800.25 | -424.25 | 767.50

Point on X-Axis - From (mm)

926.45~ | 86.82~ | 924.43~

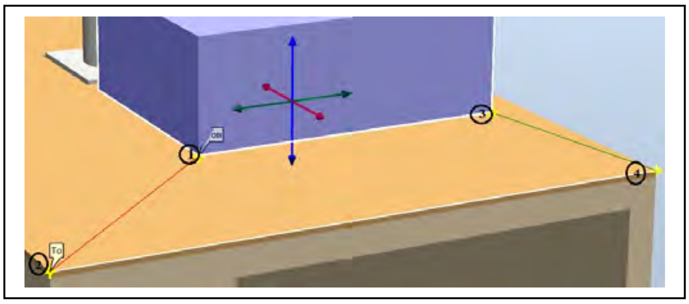
Point on X-Axis - To (mm)

800.25 | 424.25 | 767.50

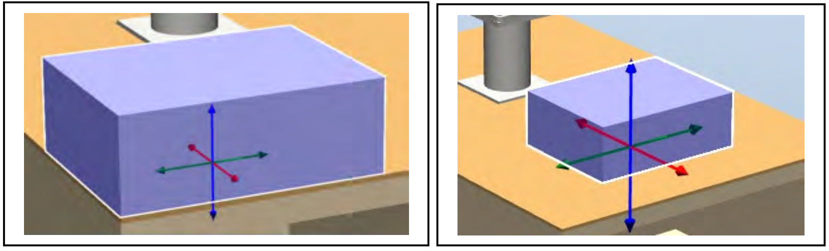
Translate along these axes:

☑ X ☑ Y ☑ Z

Apply Close



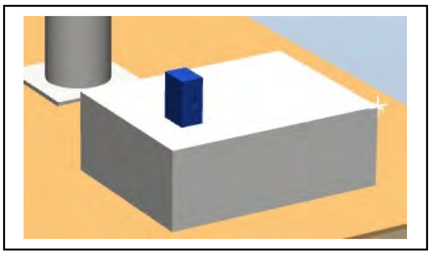
17. Click **Apply**.
18. Reposition the *WoodBox* in X and Y direction by using **Move** on On **Freehand** field.



*The primary point on the WoodBox is now moved to the primary point on the table*

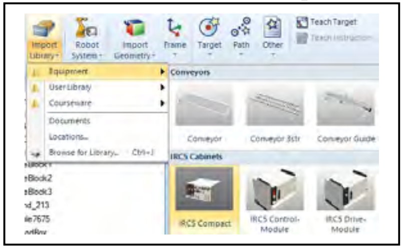
## d) Placing the blue block

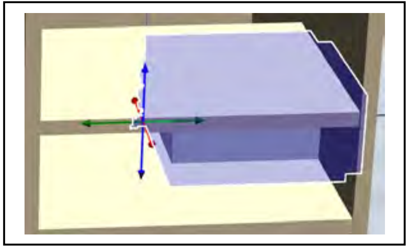
19. Repeat step 9 to 17 to **Place** *BlueBlock* on *WoodBox*



## e) Importing the controller cabinet

20. On the **Home** tab click the **Import Library** button (lower section).
21. In the **Equipment** folder, select the *IRC5 Compact* library.

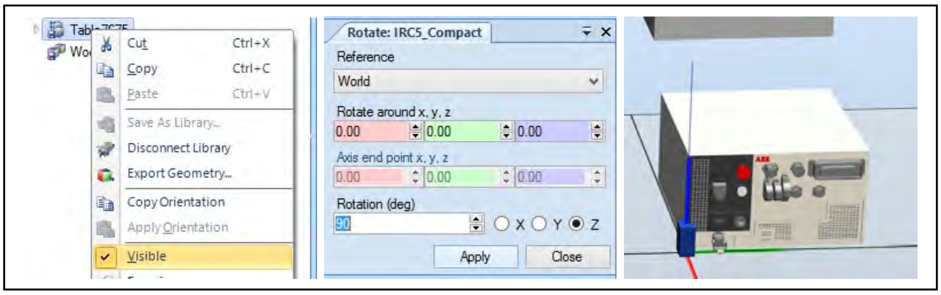




*The cabinet will be imported to the station and placed at the origin of the world coordinate system.*

## f) Placing the controller cabinet

22. Uncheck the **visible** of the *table* and rotate the *IRC5* about Z axis by $90^0$.



Rotate: IRC5_Compact

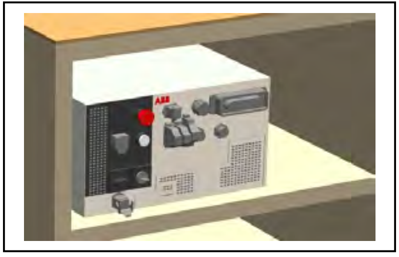
Reference: World

Rotate around x, y, z: 0.00 0.00 0.00

Axis end point x, y, z: 0.00 0.00 0.00

Rotation (deg): 90 ○ X ○ Y ● Z

23. Using the same technique as in **section 2.2.7**, place the controller cabinet on the table cabinet.



24. Save the station as *MyABBpick&place_SxGx_s3*.

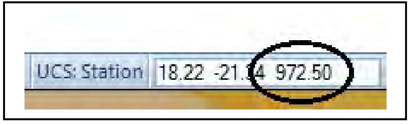
### 1.1.4 Positioning the robot

**Overview**

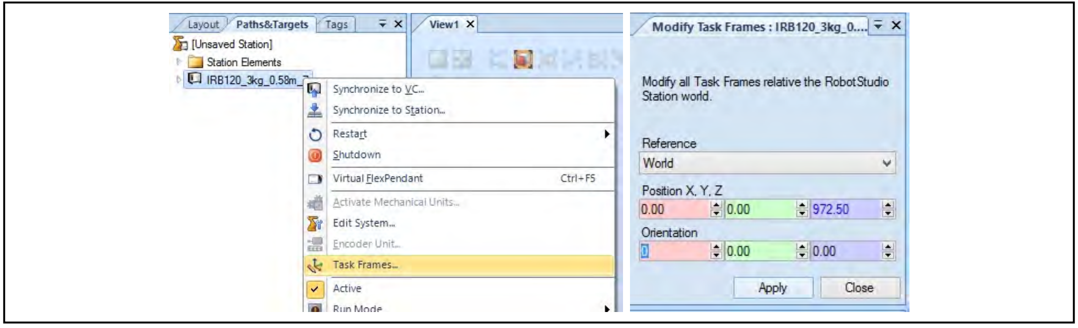
- The next step is to put the robot on the stand. To do this in RobotStudio we will move the “Task Frame”. This moves the robot system around within the RobotStudio environment without affecting the controller base frame values.
- The snap mode and selection level tools can be extremely useful when working with geometry.

1. Open the station from the last exercise (*MyABBpick&place_SxGx_s3*), unless it is already open.

2. Set the selection level to **surface** and click on the top of the *stand*. In the coordinates output (bottom right corner of the screen) we can see the height of the top of the *stand* is *972.50mm.*



3. Right click on the system in the **Paths&Targets** browser in order to bring up the **Modify Task Frames** dialog box.
4. Enter the value previously determined (**Z=972.50mm**). Click **Apply**.



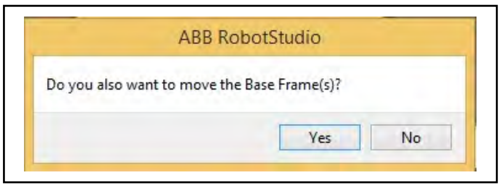
Modify all Task Frames relative the RobotStudio Station world.

Reference: World

Position X, Y, Z: 0.00 0.00 972.50

Orientation: 0 0.00 0.00

5. As you do not want to change the relationship between the controller and the base frame answer **Yes** to the question *"Do you also want to move the Base Frames(s)"?*



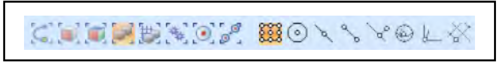
Do you also want to move the Base Frame(s)?

6. Save the station as *MyABBpick&place_SxGx_s4.*

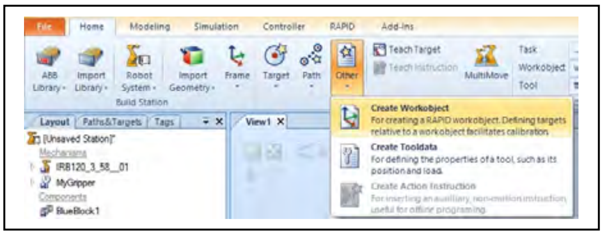
## 1.2 Programming the basic station

### 1.2.1 Creating a workobject

1. Open the station from the last exercise (*MyABBpick&place_SxGx_s4*), unless it is already open.
2. In the Graphics window select the **Part Select** selection level and the **Snap Object** snap mode.



3. Rotate and zoom the station till you get a clear view of the top of the box.
4. On the **Home** tab click on **Other** and select **Create Workobject** from the drop-down menu.

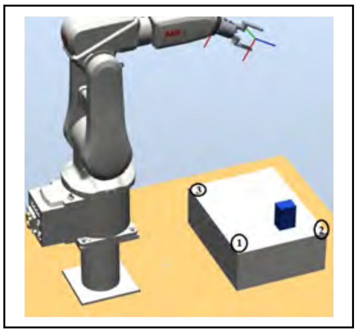


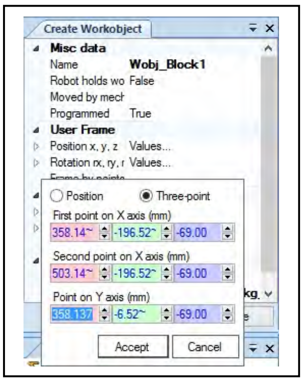
Create Workobject: For creating a RAPID workobject. Defining targets relative to a workobject facilitates calibration.

Create Tooldata: For defining the properties of a tool, such as its position and load.

Create Action Instruction: For inserting an auxiliary, non-motion instruction useful for offline programing.

5. In the **Create Workobject** dialog, in the **Name** box enter *Wobj_Block1*.
6. In the **User Frame** group click the **Frame by points** box and then click the drop-down arrow.
7. In the **Frame by points** dialog select **Three-point** as method for defining the frame.
8. Set the insertion point in one of the **First point on X axis** boxes and then click the corner of the box marked as **1** in the picture above. The coordinates of the selected point are now inserted in the boxes and the insertion point moved to the **Second point on X** axis boxes.
9. Continue clicking in corners **2** and **3** in **Second point on X axis** and **Point on Y axis**, respectively.




Create Workobject

**Misc data**

| | |
|---|---|
| Name | Wobj_Block1 |
| Robot holds wo | False |
| Moved by mech | |
| Programmed | True |

**User Frame**

| | |
|---|---|
| Position x, y, z | Values... |
| Rotation rx, ry, r | Values... |

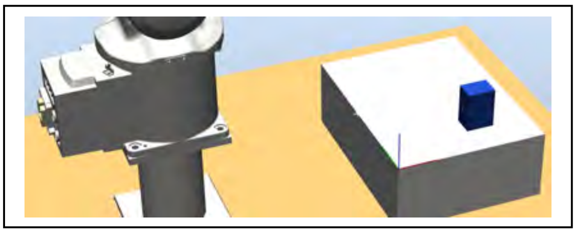
Frame by points

○ Position ● Three-point

First point on X axis (mm): 358.14~ -196.52~ -69.00

Second point on X axis (mm): 503.14~ -196.52~ -69.00

Point on Y axis (mm): 358.137 -6.52~ -69.00

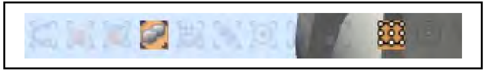


10. Click **Accept**. The Create frame by points dialog will close.
11. In the Create Workobject dialog click **Create**. A workobject, displayed as a coordinate system, is now created on the *WoodBox*. You can also see the workobject in the **Paths&Targets** browser.
12. Save the station as *MyABBpick&place_SxGx_s5*.

## 1.2.2 Programming motion

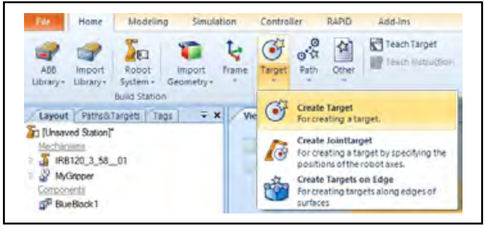
### a) Creating the targets

1. Open the station from the last exercise (*MyABBpick&place_SxGx_s5*), unless it is already open.
2. In the Graphics window select **Part Selection** level and **Snap Centre**.

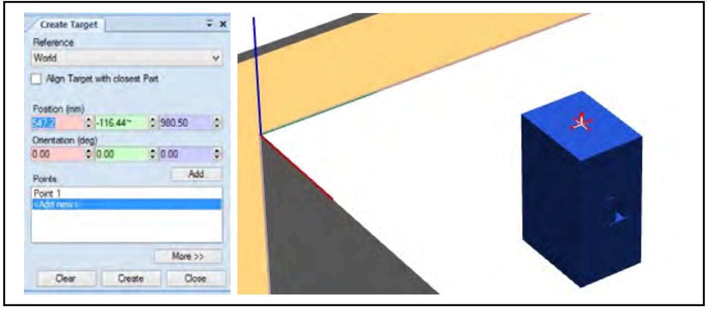


3. Zoom and rotate the station so that you get a clear view of the tool and the small *BlueBlock*.

4. On the **Home** tab click the **Target** drop-down and select **Create Target**.



5. In the **Create Target** dialog, make sure the pointer is set to the centre position on *WoodBlock*.


Create Target

Reference: World

□ Align Target with closest Part

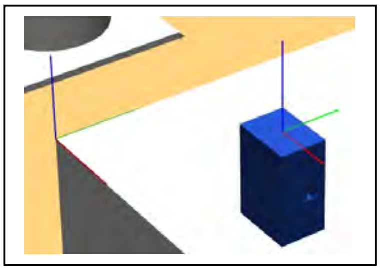
Position (mm): 547.2 -116.44~ 980.50

Orientation (deg): 0.00 0.00 0.00

Points

Point 1
<Add new>

6. In the **Create Target** dialog click the **Create** button.



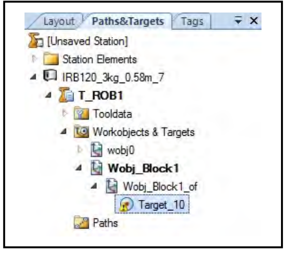
*Now 1 target is created with default orientation (0,0,0)*

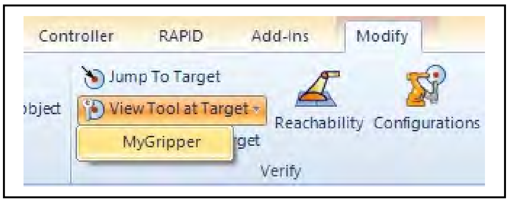
7. Save the station as *MyABBpick&place_SxGx_s6*.

## b) Adjusting the target orientation

- *The function **View Tool at Target** will give us a preview on how the tool will be oriented around the targets.*
- *The function **View Robot at Target** will give us a preview on how the robot will be oriented around the targets. If the target is reachable, then robot will automatically jump to the target.*

8. Open the station from the last exercise (*MyABBpick&place_SxGx_s6*), unless it is already open.
9. In the **Paths&Targets** browser select the first target (Target_10) and click the **Modify** tab.
10. In the **View Tool at target** drop-down, select *MyGripper*.

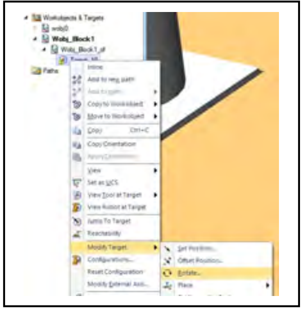




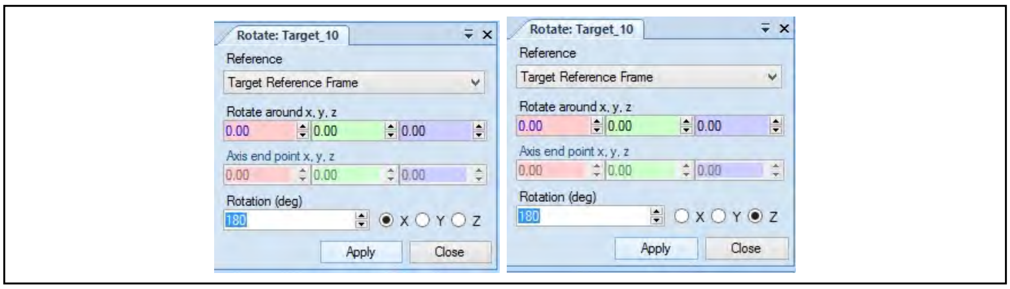
*As the orientation of our targets are zero and our TCP has Z pointing out from the tool, the preview of the tool will be hidden in the WoodBox. To be able to see this preview we need to make the WoodBox invisible*

11. In **Layout** browser un-check **Visible** in the context menu of the *WoodBox*. We will now able to see the tool on *Target_10*.

12. Right click on *Target_10*, select **Rotate** from **Modify Target** context menu



13. In **Rotate:Target_10** menu, set the **Reference** to *Target Reference Frame*.
14. Rotate the target approximately +180 degrees around the X axis by inserting $180^0$ in **Rotation(deg)** field and click **Apply**.
15. Rotate the target approximately +180 degrees around the Z axis by inserting $180^0$ in **Rotation(deg)** field and click **Apply**.


Rotate: Target_10

Reference: Target Reference Frame

Rotate around x, y, z: 0.00 | 0.00 | 0.00

Axis end point x, y, z: 0.00 | 0.00 | 0.00

Rotation (deg): 180 ● X ○ Y ○ Z

Apply | Close

Rotate: Target_10

Reference: Target Reference Frame

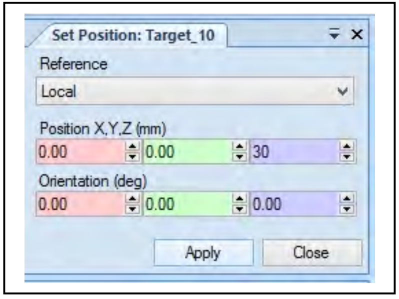
Rotate around x, y, z: 0.00 | 0.00 | 0.00

Axis end point x, y, z: 0.00 | 0.00 | 0.00

Rotation (deg): 180 ○ X ○ Y ● Z

Apply | Close

16. In the context menu of Target_10, click **Modify Target>Set Position**.
17. Select **Local** as reference and set Z position to 30mm. Press **Apply**.

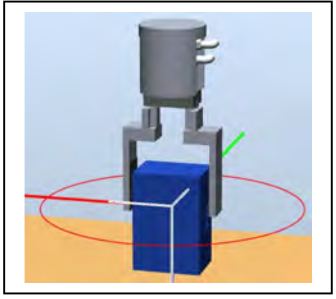

Set Position: Target_10

Reference: Local

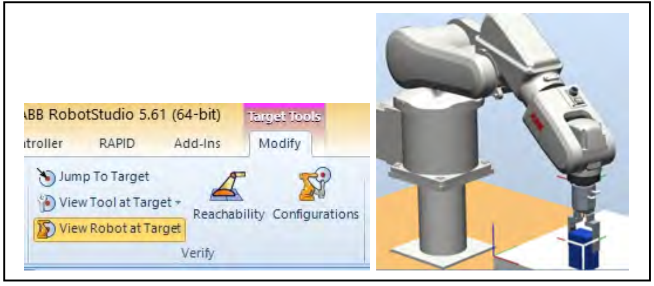
Position X,Y,Z (mm): 0.00 | 0.00 | 30

Orientation (deg): 0.00 | 0.00 | 0.00

Apply | Close



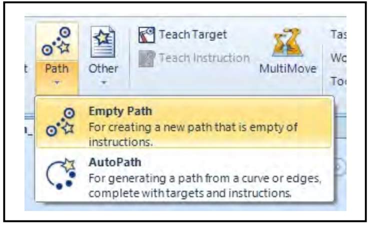
18. Make the *WoodBox* visible again by checking **Visible** from the context menu.
19. Activate **View Robot at Target**, and then ensure the robot jump automatically to the targets. If the robot not jumps to the target/s, reposition the respective target/s and block/s.



20. Disable **View Tool at Target**, **View Robot at Target.**
21. Save the station as *MyABBpick&place_SxGx_s7*.

## c) Adding the targets to a path

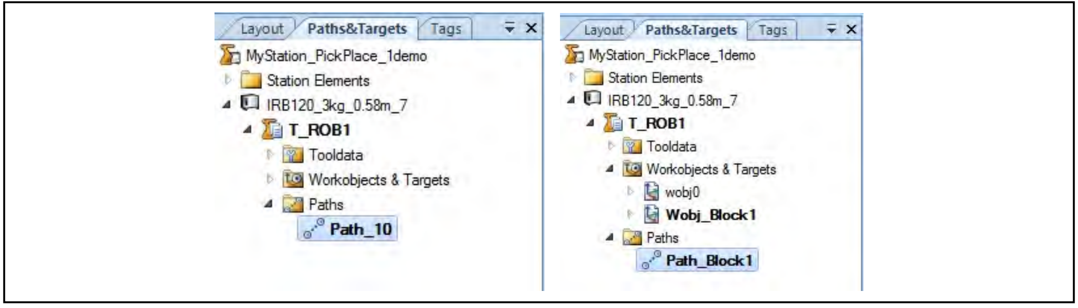
22. Open the station from the last exercise (*MyABBpick&place_SxGx_s7*), unless it is already open.
23. On the **Home** tab click **Empty Path** from the **Path** drop-down menu.



**Empty Path**
For creating a new path that is empty of instructions.

**AutoPath**
For generating a path from a curve or edges, complete with targets and instructions.

24. Right click the path, select **Rename** and change the name to Path_Block1.

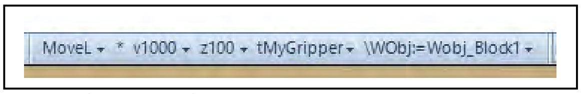

Layout | Paths&Targets | Tags

- MyStation_PickPlace_1demo
  - Station Elements
  - IRB120_3kg_0.58m_7
    - T_ROB1
      - Tooldata
      - Workobjects & Targets
      - Paths
        - Path_10

Layout | Paths&Targets | Tags

- MyStation_PickPlace_1demo
  - Station Elements
  - IRB120_3kg_0.58m_7
    - T_ROB1
      - Tooldata
      - Workobjects & Targets
        - wobj0
        - Wobj_Block1
      - Paths
        - Path_Block1

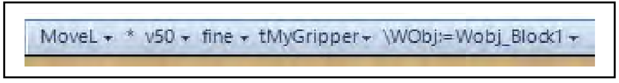
*An empty path, **Path_10**, is now created and displayed in the **Paths&Targets** browser.*



MoveL * v1000 z100 tMyGripper \WObj:=Wobj_Block1

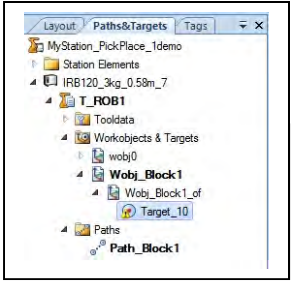
*On the* ***Status Bar*** *down at the bottom of the interface you can see the active instruction template. These are the default settings that will be used when creating the Move instructions.*

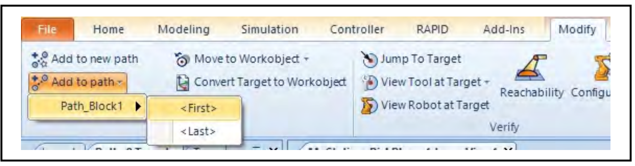
25. Change the parameters according to the picture below.



MoveL * v50 fine tMyGripper \WObj:=Wobj_Block1

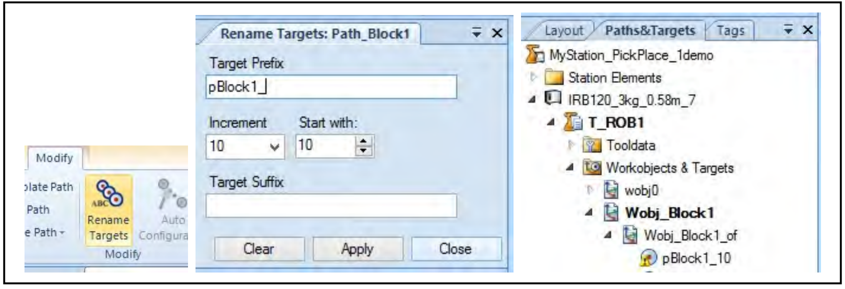
26. In the **Paths&Targets** browser select *Target_10*.
27. On the **Modify** tab click the **Add to Path** button and select **Path_Block1** and **First**.





***Tip!*** *You can also use drag&drop to create the instructions.*

28. Highlight the path (**Path_Block1**), and in the **Modify** tab select **Rename Targets**. Write *pBlock1_* as target prefix and press **Apply**. This function is also available from the context menu of the path.



Rename Targets: Path_Block1

Target Prefix: pBlock1_

Increment: 10

Start with: 10

Target Suffix:

29. Save the station as (*MyABBpick&place_SxGx_s8*).

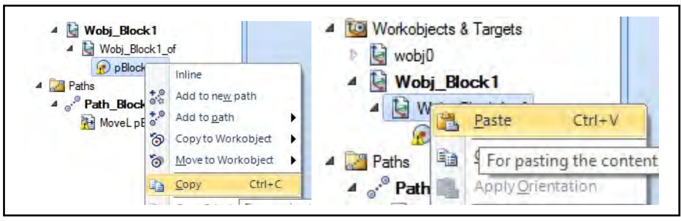
### 1.2.3 Adding a approach, home and start position

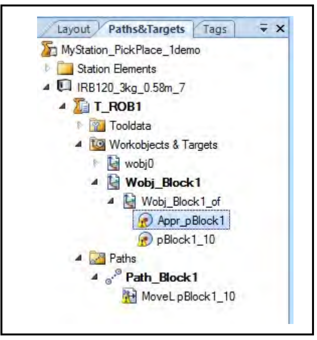
**Overview**

- To be sure that the robot can reach the target we will add a new target which we will use as start position and approach/depart target. As the path, so far, only consists of linear instructions problems will appear in situations where the actual position of the robot makes a linear movement to the first instruction impossible

#### a) Adding an approach/depart target

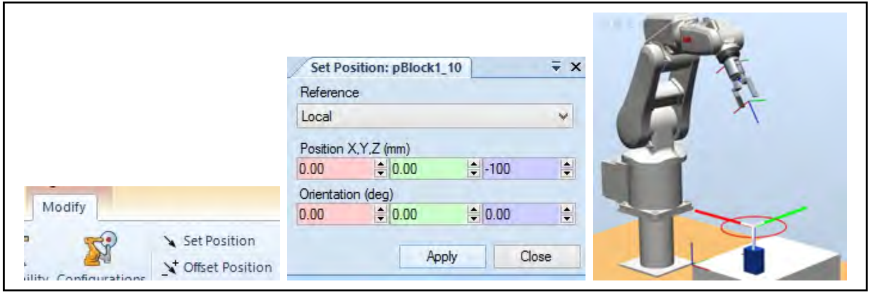
1. Open the station from the last exercise (*MyABBpick&place_SxGx_s8*), unless it is already open.
2. In the **Paths&Targets** browser select copy from the **pBlock1_10** context menu. Then click Paste from the **Wobj_Box** context menu.
3. Rename the copy target to **Appr_pBlock1**.





4. Select **Appr_Box** and click **Set Position** from the **Modify** tab.

5. Set the reference to **Local** and move the target **-**100mm in the Z direction. Press **Apply**



Set Position: pBlock1_10

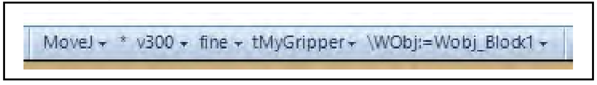
Reference: Local

Position X,Y,Z (mm): 0.00 | 0.00 | -100

Orientation (deg): 0.00 | 0.00 | 0.00

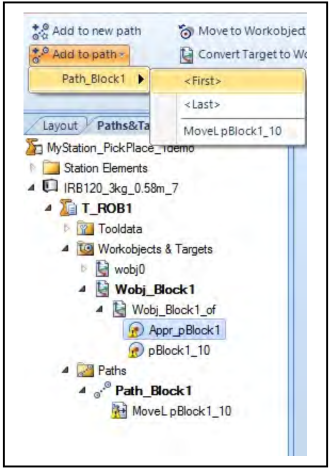
*The **Appr_pBlock1** has been shifted up*

6. In the toolbar for active templates, change to **MoveJ** and set the parameters as below.



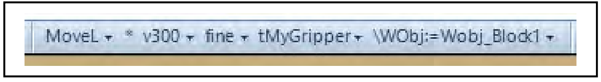
MoveJ ▾ * v300 ▾ fine ▾ tMyGripper ▾ \WObj:=Wobj_Block1 ▾

7. Highlight the target **Appr_Block1** and then add it to the top (first) of the path.



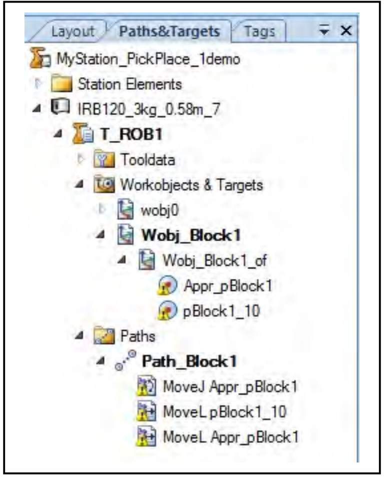
MyStation_PickPlace_1demo
- Station Elements
- IRB120_3kg_0.58m_7
  - T_ROB1
    - Tooldata
    - Workobjects & Targets
      - wobj0
      - Wobj_Block1
        - Wobj_Block1_of
          - Appr_pBlock1
          - pBlock1_10
    - Paths
      - Path_Block1
        - MoveL pBlock1_10

8. Change back to **MoveL** as active template.



MoveL ▾ * v300 ▾ fine ▾ tMyGripper ▾ \WObj:=Wobj_Block1 ▾

9. Repeat step 7 but now add the instruction last <*Last*> in the path.



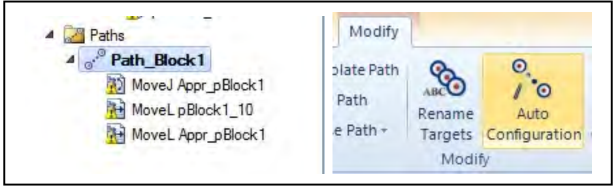
MyStation_PickPlace_1demo
- Station Elements
- IRB120_3kg_0.58m_7
  - T_ROB1
    - Tooldata
    - Workobjects & Targets
      - wobj0
      - Wobj_Block1
        - Wobj_Block1_of
          - Appr_pBlock1
          - pBlock1_10
    - Paths
      - Path_Block1
        - MoveJ Appr_pBlock1
        - MoveL pBlock1_10
        - MoveL Appr_pBlock1

## b) Setting axes configuration

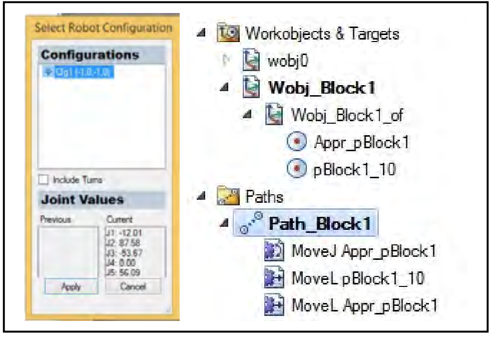
- Note that the yellow triangle warning appears on the new instructions. This is because no axes configuration is set yet. Before we can setup and run a simulation we need to define what axes configurations the robot should have for each target. This can be done manually by stepping through each instruction and clicking

**Configurations** from the context menu. In this way you will get a list of all available configurations. In cases where we have many instructions a more efficient way is to use **Auto Configuration**. In this way we will only set the start configuration and then RobotStudio will calculate the configuration for the rest of the instructions in order to get as smooth movements of the robot axes as possible.

10. Select **Path_Block1** and click **Auto Configuration** from the **Modify** tab.



11. Select the first configuration in the list and click **Apply**. You can see the robot moves on the configure path. The selected configuration will now be set to the first target and calculated for the others.



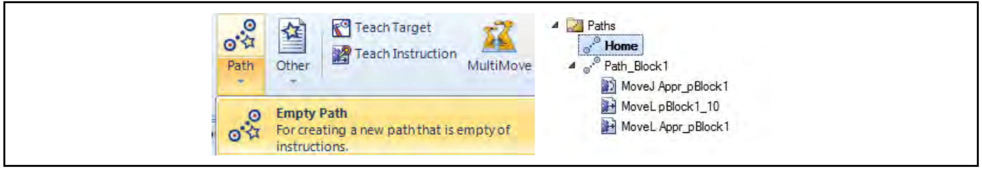
***Path_Block1** has been configured*

12. Save the station as *MyABBpick&place_SxGx_s9*.

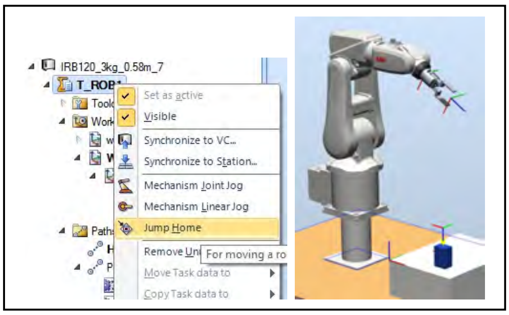
## c) Adding a Home position

Now we will also add a Home position that will be placed in a separate path.

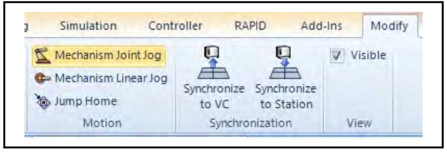
13. Open the station from the last exercise (*MyABBpick&place_SxGx_s9*), unless it is already open.
14. From **Home** tab, create a new empty path and rename it to **Home**.



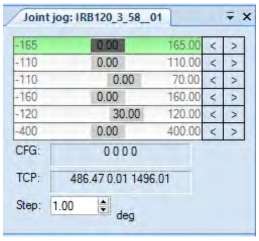
15. In **Home** tab, select **Jump Home** from the context menu of the robot. The robot will now reset the axes to default values.



16. From the **Modify** tab, select **Mechanism Joint Jog.**



17. Jog the robot or set all the values to zero.



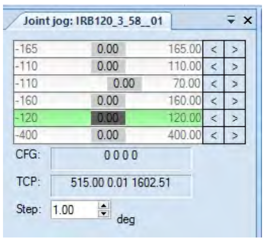
Joint jog: IRB120_3_58__01

| | | |
|---|---|---|
| -165 | 0.00 | 165.00 |
| -110 | 0.00 | 110.00 |
| -110 | 0.00 | 70.00 |
| -160 | 0.00 | 160.00 |
| -120 | 30.00 | 120.00 |
| -400 | 0.00 | 400.00 |

CFG: 0 0 0 0

TCP: 486.47 0.01 1496.01

Step: 1.00 deg



Joint jog: IRB120_3_58__01

| | | |
|---|---|---|
| -165 | 0.00 | 165.00 |
| -110 | 0.00 | 110.00 |
| -110 | 0.00 | 70.00 |
| -160 | 0.00 | 160.00 |
| -120 | 0.00 | 120.00 |
| -400 | 0.00 | 400.00 |

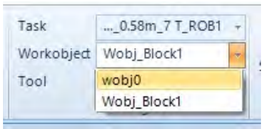
CFG: 0 0 0 0

TCP: 515.00 0.01 1602.51

Step: 1.00 deg

***Tip!** Click the separate boxes for each axis and press space on your keyboard. Now you will be able to write exact values.*

18. Change active work object to **wobj0** from the **Settings** group in **Home** tab.

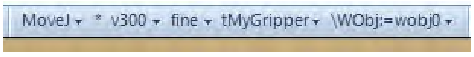


Task: ..._0.58m_7 T_ROB1

Workobject: Wobj_Block1

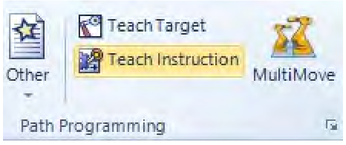
Tool

19. In the toolbar for active templates, change to **MoveJ** and set the parameters as below.



MoveJ * v300 fine tMyGripper \WObj:=wobj0

20. In **Path Programming group** of **Home** tab, click **Teach Instruction.** Press *Yes* on the message that follows.



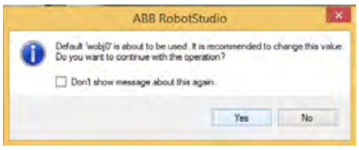


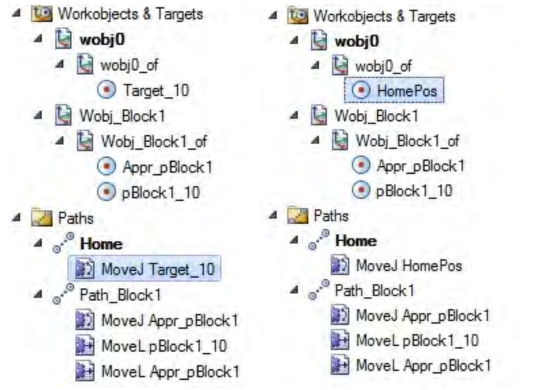
ABB RobotStudio

Default 'wobj0' is about to be used. It is recommended to change this value.
Do you want to continue with the operation?

□ Don't show message about this again.

*A new target (Target_10) will now be created in **wobj0** and a **MoveJ** instruction will be added to the new Path.*

21. Rename the new target to **HomePos**.



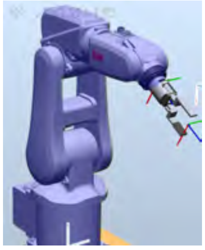
- Workobjects & Targets
  - wobj0
    - wobj0_of
      - Target_10
  - Wobj_Block1
    - Wobj_Block1_of
      - Appr_pBlock1
      - pBlock1_10
- Paths
  - Home
    - MoveJ Target_10
  - Path_Block1
    - MoveJ Appr_pBlock1
    - MoveL pBlock1_10
    - MoveL Appr_pBlock1

- Workobjects & Targets
  - wobj0
    - wobj0_of
      - HomePos
  - Wobj_Block1
    - Wobj_Block1_of
      - Appr_pBlock1
      - pBlock1_10
- Paths
  - Home
    - MoveJ HomePos
  - Path_Block1
    - MoveJ Appr_pBlock1
    - MoveL pBlock1_10
    - MoveL Appr_pBlock1

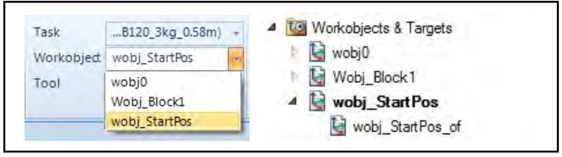
## d) Adding a Start position

This Start position is important to avoid singularity problem. The robot will move to/from this Start position from/to Home position by only rotating its joint 5.

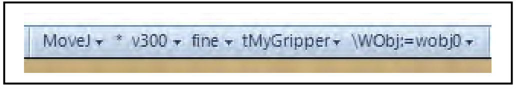
22. From **Home** tab, create a new empty path and rename it to **Path_start**.
23. In **Home** tab, select **Jump Home** from the context menu of the robot. The robot will now reset the axes to default values.



24. In **Paths&Targets** tree, select *copy* from **wobj_Block1** context menu and then select *paste* from **Workobjects & Targets** context menu. Answer '*No*' for the message.
25. Rename the new workobject as *Wobj_StartPos* and delete the available targets in ***Wobj_StartPos***.
26. Change active work object to **Wobj_StartPos** from the **Settings** group in **Home** tab.

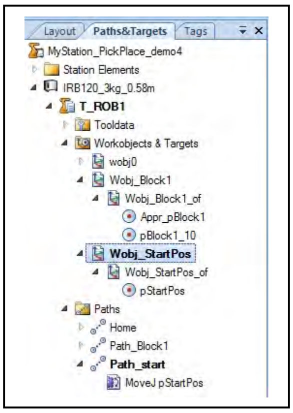


27. In the toolbar for active templates, change to **MoveJ** and set the parameters as below.



MoveJ ▾ * v300 ▾ fine ▾ tMyGripper ▾ \WObj:=wobj0 ▾

28. In **Path Programming group** of **Home** tab, click **Teach Instruction.** Press *Yes* on the message that follows.
29. Rename the new target to **pStartPos**.



30. Save the station as *MyABBpick&place_SxGx_s10.*

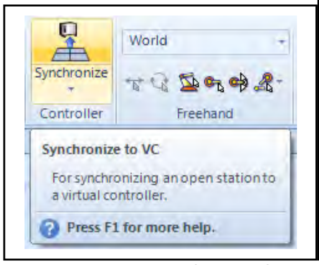
### 1.2.4 Running the Simulation

**Overview**

- Now we have completed all steps required to proceed creating a RAPID program. The strength with the virtual controller is that we use the same software as the real robot controller. This means that we are able to run a simulation where we get very close to the same behaviour as on a real robot controller. The robot program will be stored on the system running on the virtual controller, just as the program for a real robot is stored on its system.

### a) Synchronize to the Virtual Controller

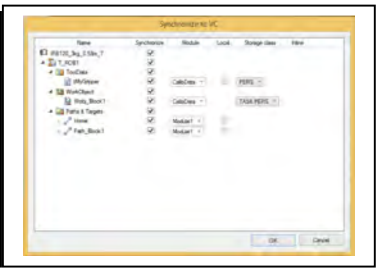
1. Open the station from the last exercise (*MyABBpick&place_SxGx_s10*), unless it is already open.
2. On the **Home** tab, click the **Synchronize** button.
3. In the **Synchronize to VC** dialog make sure all data are selected and then click **OK**.



Synchronize to VC

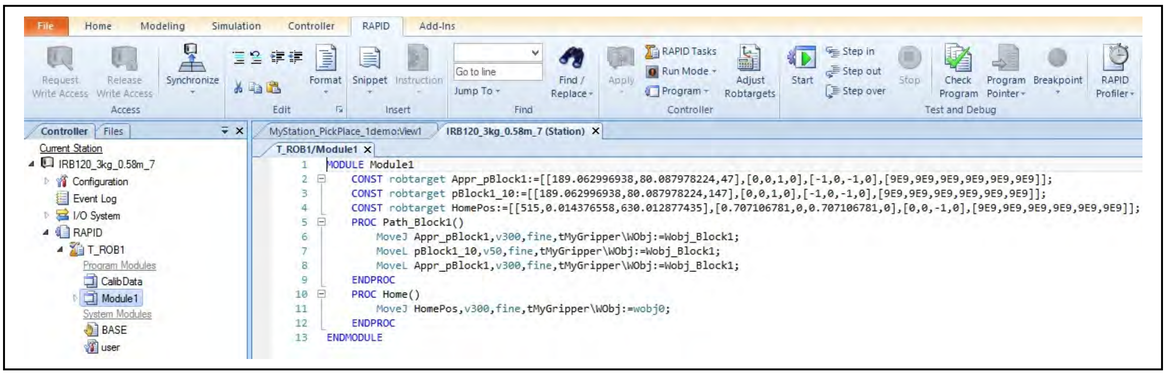
For synchronizing an open station to a virtual controller.

Press F1 for more help.



*All program data is now transferred from the RobotStudio station to the virtual controller.*

4. To get a view of the result, expand the tree structure in the Rapid tab, and double click **Module1** as shown below.

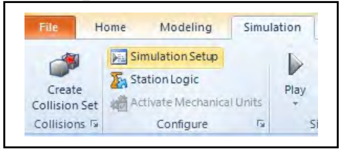


```
MODULE Module1
    CONST robtarget Appr_pBlock1:=[[189.062996938,80.087978224,47],[0,0,1,0],[-1,0,-1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget pBlock1_10:=[[189.062996938,80.087978224,147],[0,0,1,0],[-1,0,-1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget HomePos:=[[515,0.014376558,630.012877435],[0.707106781,0,0.707106781,0],[0,0,-1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
    PROC Path_Block1()
        MoveJ Appr_pBlock1,v300,fine,tMyGripper\WObj:=Wobj_Block1;
        MoveL pBlock1_10,v50,fine,tMyGripper\WObj:=Wobj_Block1;
        MoveL Appr_pBlock1,v300,fine,tMyGripper\WObj:=Wobj_Block1;
    ENDPROC
    PROC Home()
        MoveJ HomePos,v300,fine,tMyGripper\WObj:=wobj0;
    ENDPROC
ENDMODULE
```

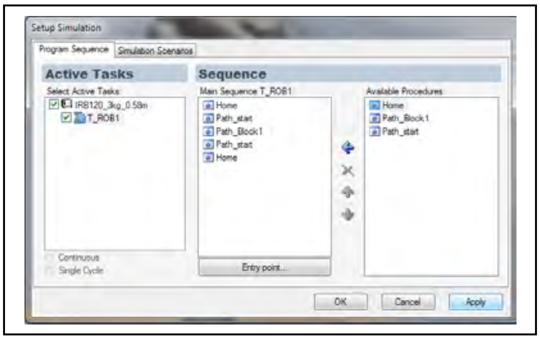
## b) Setup the Simulation

To be able to start a simulation we need to define where the robot should start the execution. This can be done by adding a main sequence directly in the Rapid Editor or we can use the Simulation Setup dialog where we get this done automatically.

5. On the **Simulation** tab click the **Simulation Setup** button.



6. Tick the **T_ROB1**. Select *Home* and *Path_Block1*, and click the arrow pointing to the left in order to add it to main procedure.

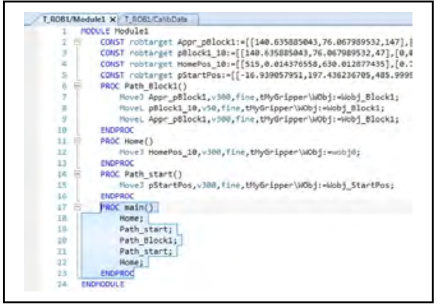


☑ IRB120_3kg_0.58m
☑ T_ROB1

Main Sequence T_ROB1:
Home
Path_start
Path_Block1
Path_start
Home

Available Procedures:
Home
Path_Block1
Path_start

7. Click the **OK** button.
8. Now go back to the Rapid editor to see the resulting main sequence.

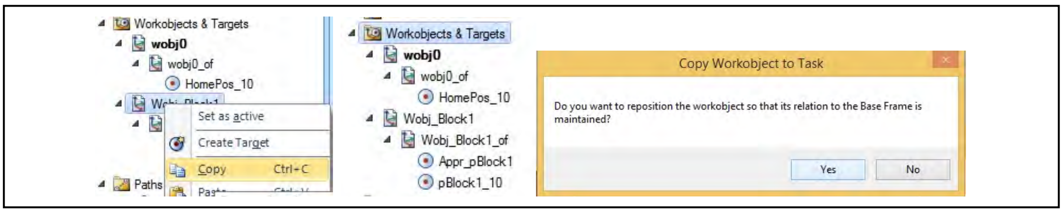


```
MODULE Module1
    CONST robtarget Appr_pBlock1:=[[140.635885043,76.067989532,147],[
    CONST robtarget pBlock1_10:=[[140.635885043,76.067989532,47],[0,0
    CONST robtarget HomePos_10:=[[515,0.014376558,630.012877435],[0.7
    CONST robtarget pStartPos:=[[-16.939057951,197.436236705,485.9999
    PROC Path_Block1()
        MoveJ Appr_pBlock1,v300,fine,tMyGripper\WObj:=Wobj_Block1;
        MoveL pBlock1_10,v50,fine,tMyGripper\WObj:=Wobj_Block1;
        MoveL Appr_pBlock1,v300,fine,tMyGripper\WObj:=Wobj_Block1;
    ENDPROC
    PROC Home()
        MoveJ HomePos_10,v300,fine,tMyGripper\WObj:=wobj0;
    ENDPROC
    PROC Path_start()
        MoveJ pStartPos,v300,fine,tMyGripper\WObj:=Wobj_StartPos;
    ENDPROC
    PROC main()
        Home;
        Path_start;
        Path_Block1;
        Path_start;
        Home;
    ENDPROC
ENDMODULE
```

9. In the Simulation tab click the **Play** button. The robot will now execute the RAPID program.
10. Save the station as *MyABBpick&place_SxGx_s11*.

### 1.2.5 Copying a workobject

1. Open the station from the last exercise (*MyABBpick&place_SxGx_s11*), unless it is already open.
2. In the **Paths&Targets** browser select *Wobj_Block* and then click copy.
3. Select **Workobjects & Targets** and then click paste. Answer ‘*No*’ for the message.

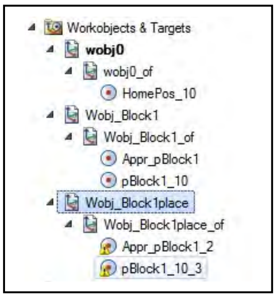


Copy Workobject to Task

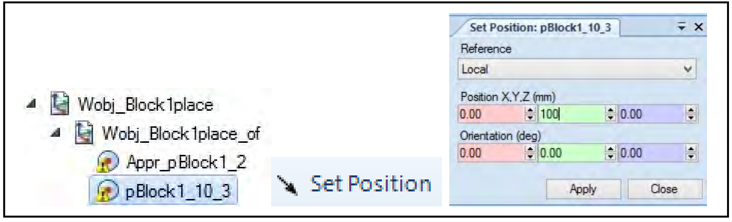
Do you want to reposition the workobject so that its relation to the Base Frame is maintained?

Yes No

4. **Rename** the workobjects to *Wobj_Block1place*.



5. In the *Wobj_Block1place* tree, select *pBlock1_10_3* and click **Set Position** from the **Modify** tab.
6. Set the reference to **Local** and move the target 100mm in the Y direction. Press **Apply**.



Set Position: pBlock1_10_3

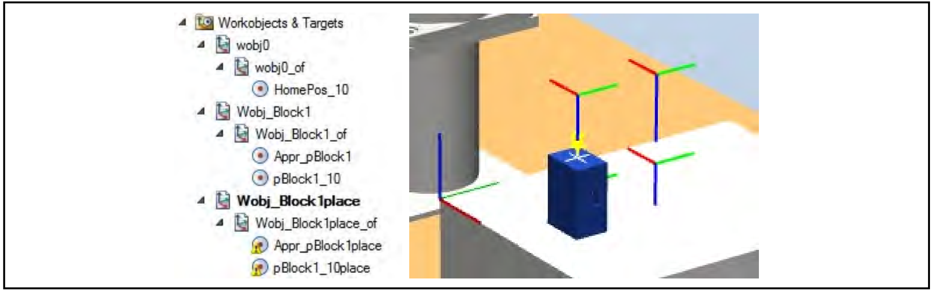
Reference: Local

Position X,Y,Z (mm): 0.00 100 0.00

Orientation (deg): 0.00 0.00 0.00

Apply Close

7. Repeat step 5-6 for *Appr_pBlock1_2*
8. **Rename** the targets *pBlock1_10_3* to *pBlock1_10place*, and *Appr_pBlock1_2* to *Appr_pBlock1place* .

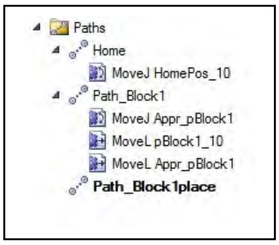


9. Save the station as *MyABBpick&place_SxGx_s12*.

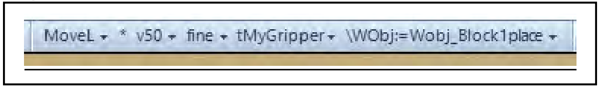
### 1.2.6 Adding programming motion

#### a) Adding the targets to a path

1. Open the station from the last exercise (*MyABBpick&place_SxGx_s12*), unless it is already open.
2. On the **Home** tab click **Empty Path** from the **Path** drop-down menu.
3. Right click the path, select **Rename** and change the name to *Path_Block1place*.

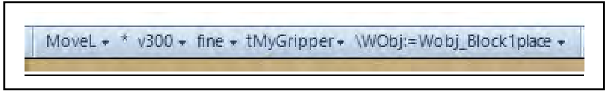


4. Change the parameters according to the picture below.



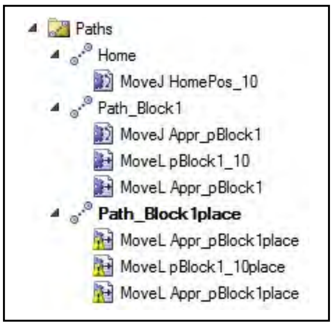
MoveL * v50 fine tMyGripper \WObj:=Wobj_Block1place

5. In the **Paths&Targets** browser select *pBlock1_10place*.
6. On the **Modify** tab click the **Add to Path** button and select **Path_Block1place** and **First**.
7. Change the active template according to the picture below.



MoveL ▾ * v300 ▾ fine ▾ tMyGripper▾ \WObj:=Wobj_Block1place ▾

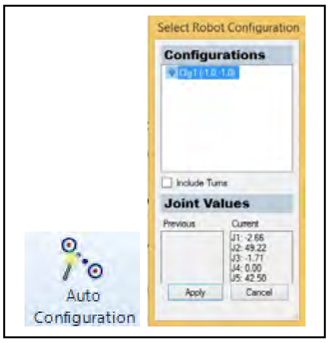
8. In the **Paths&Targets** browser select *Appr_Block1place*.
9. On the **Modify** tab click the **Add to Path** button and select **Path_Block1place** and **First**.
10. Repeat step 8-9 but now add the instruction last<Last> in the path. You will get following path for **Path_Block1place**



- Paths
  - Home
    - MoveJ HomePos_10
  - Path_Block1
    - MoveJ Appr_pBlock1
    - MoveL pBlock1_10
    - MoveL Appr_pBlock1
  - **Path_Block1place**
    - MoveL Appr_pBlock1place
    - MoveL pBlock1_10place
    - MoveL Appr_pBlock1place

*Note that the yellow triangle warning appears on the new instructions. This is because no axes configuration is set yet.*

11. Select **Path_Block1place** and click **Auto Configuration** from the **Modify** tab.
12. Select the first configuration in the list and click **Apply**. The selected configuration will now be set to the first target and calculated for the others.



Select Robot Configuration

Configurations

Cfg1 (-1,0,-1,0)

□ Include Turns

Joint Values

Previous

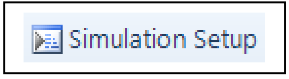
Current

J1: -2.66
J2: 49.22
J3: -1.71
J4: 0.00
J5: 42.50

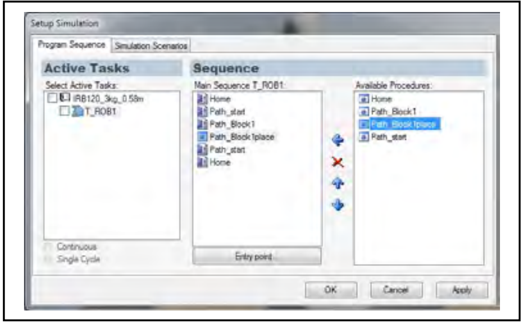
13. Save the station as *MyABBpick&place_SxGx_s13*.

## b) Synchronize and Simulation

14. Open the station from the last exercise (*MyABBpick&place_SxGx_s13*), unless it is already open.
15. On the **Home** tab, click the **Synchronize** button.
16. In the **Synchronize to VC** dialog make sure all data are selected and then click **OK**.
17. On the **Simulation** tab click the **Simulation Setup** button.



18. Using the **Available Procedures**, construct the Main Sequence according to the picture below.



Setup Simulation

Active Tasks

Select Active Tasks:

□ IRB120_3kg_0.58m
□ T_ROB1

Continuous
Single Cycle

Sequence

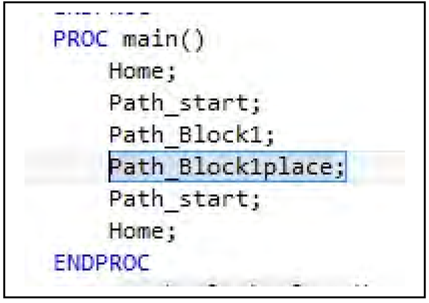
Main Sequence T_ROB1:

Home
Path_start
Path_Block1
Path_Block1place
Path_start
Home

Available Procedures:

Home
Path_Block1
Path_Block1place
Path_start

19. Click the **OK** button.
20. Now go back to the Rapid editor to see the resulting main sequence.



```
PROC main()
    Home;
    Path_start;
    Path_Block1;
    Path_Block1place;
    Path_start;
    Home;
ENDPROC
```

21. In the Simulation tab click the **Play** button. The robot will now execute the RAPID program.
22. Save the station as *MyABBpick&place_SxGx_s14.*

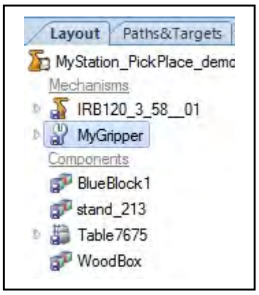
## Smart Component

**Goal of the chapter**

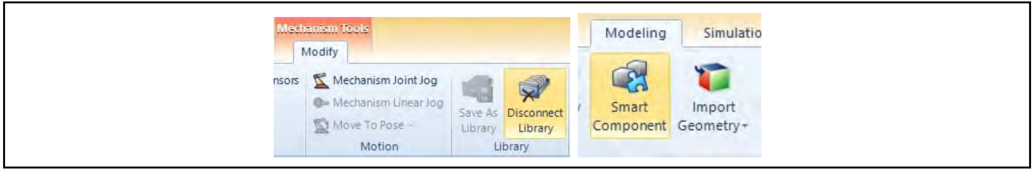
- In this exercise, we will learn how to create a Smart Component (SC) representing a gripper

### 2.1 Create Smart Component – Gripper
### 2.1.1 Preparation

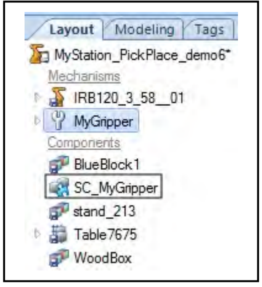
1. Open the station from the last exercise (*MyABBpick&place_SxGx_s14*), unless it is already open.
2. In Layout browser select **MyGripper**



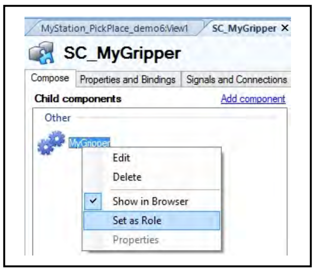
3. As we will modify the library file we need to disconnect the library.
4. In the library context menu, click **Disconnect Library**.
5. In **Modeling** tab, click **Smart Component** to create a new empty SC.



6. Rename the SC to **SC_MyGripper**.
7. In Layout browser, drag and drop the *MyGripper* to **SC_MyGripper**.

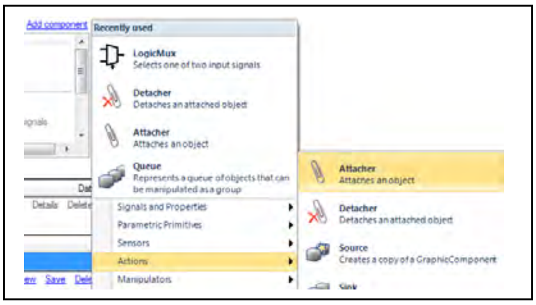


8. In order to get our new component to act as a tool, we need to set the My**Gripper** as **role**. In this way the **tooldata** will be created when we attach our **SC gripper** to a robot.
9. In the SC view, set the *MyGripper* as **role** from the context menu.

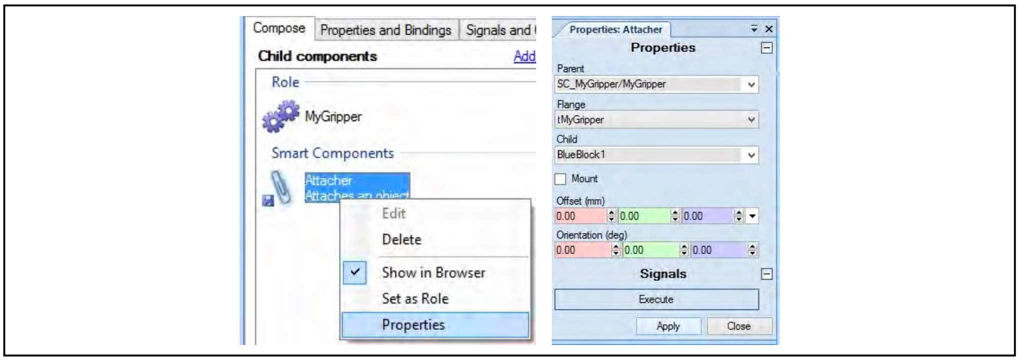


## 2.1.2 Add Base Components

1. Under **Compose**, click **Add component**. Now add an A**ttacher** 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 *MyGripper*.


Add component
Recently used
LogicMux
Selects one of two input signals
Detacher
Detaches an attached object
Attacher
Attaches an object
Queue
Represents a queue of objects that can be manipulated as a group
Signals and Properties
Parametric Primitives
Sensors
Actions
Manipulators
Attacher
Attaches an object
Detacher
Detaches an attached object
Source
Creates a copy of a GraphicComponent
Sink

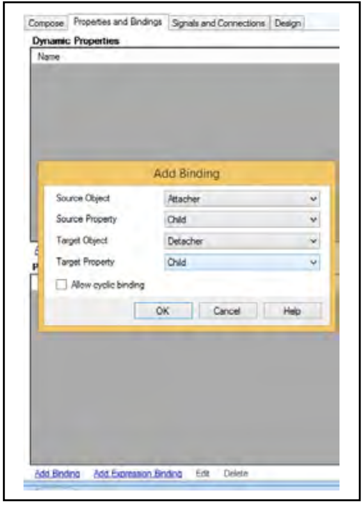
2. In the properties window of the **Attacher**, select the **SC_MyGripper/MyGripper** as parent and **BlueBlock1** as child, as shown below.



Parent: SC_MyGripper/MyGripper
Flange: tMyGripper
Child: BlueBlock1
☐ Mount
Offset (mm): 0.00 0.00 0.00
Orientation (deg): 0.00 0.00 0.00

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

3. Go back to the **Compose** tab and add a **Detacher** base component
4. Click **Add binding** in the **Properties and Bindings** tab of the SC view again and create the binding as below:

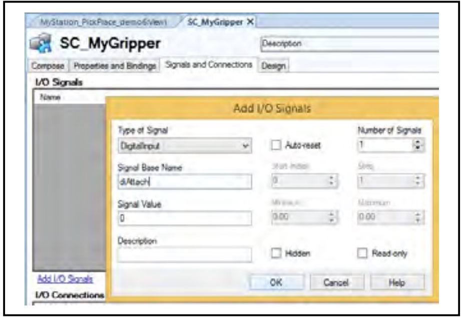


Source Object: Attacher
Source Property: Child
Target Object: Detacher
Target Property: Child
☐ Allow cyclic binding

### 2.1.3 Internal Signals

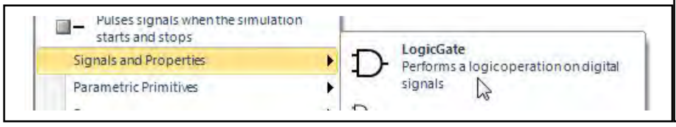
- Now we need to define internal signals in our component that later will be cross connected with the **I/O** signals in the Virtual Controller. To define the actions in our **SC**, we also need to define some **I/O** connections.

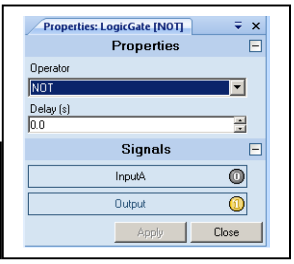
1. In the **Signals and connections** tab of the SC view, click **Add I/O Signals** and add a digital input signal, **diAttach**.



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

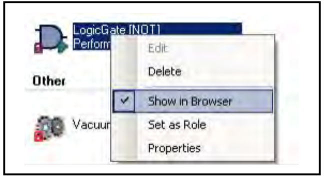
2. When the signal **diAttach** is set low, we want the **Detacher** to execute. Instead of creating a new **I/O** signal we will add a new base component, **LogicGate (NOT).** In this way we can trigger the **Detacher** when the **diAttach** signal is low (**NOT** high).
3. In the **Compose** tab of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


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

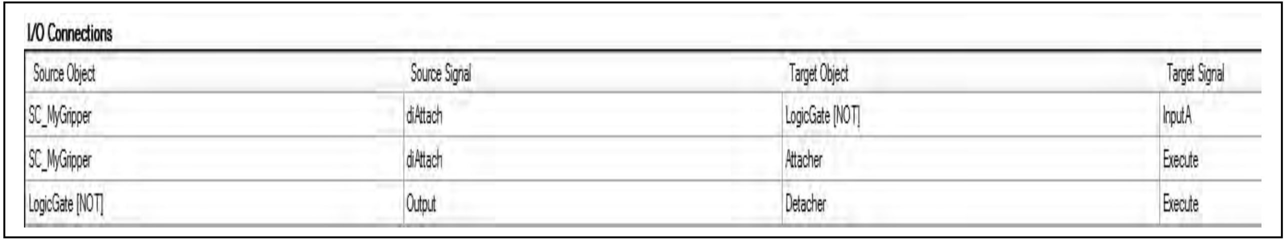


Operator: NOT
Delay (s): 0.0

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



6. Click **Add I/O Connection** which under **Signals and Connections** tab to create the following first three **I/O connections** with the following input.



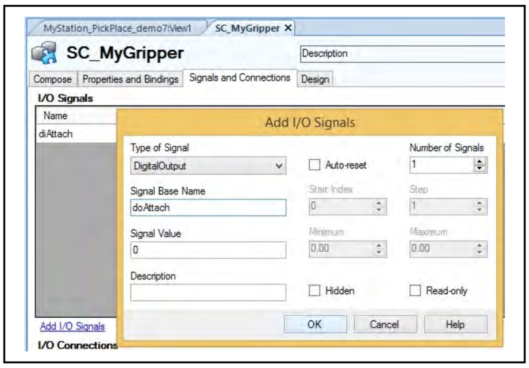
I/O Connections

| Source Object | Source Signal | Target Object | Target Signal |
|---|---|---|---|
| SC_MyGripper | diAttach | LogicGate [NOT] | InputA |
| SC_MyGripper | diAttach | Attacher | Execute |
| LogicGate [NOT] | Output | Detacher | Execute |

***diAttach** signal will be the input of **LogicGate[NOT]** and the **Attacher**, and the output of **LogicGate[NOT]** will be the input of the **Detacher***

- *The next step is to create a handshake output signal (SC internal), **doAttached** from our **SC**, which later will be connected to our real digital input signal back to the robot controller giving information if 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.***

7. 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: doAttach

Start Index: 0

Step: 1

Signal Value: 0

Minimum: 0.00

Maximum: 0.00

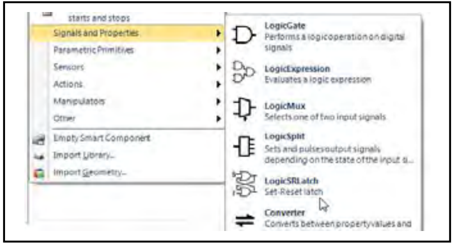
Description:

□ Hidden

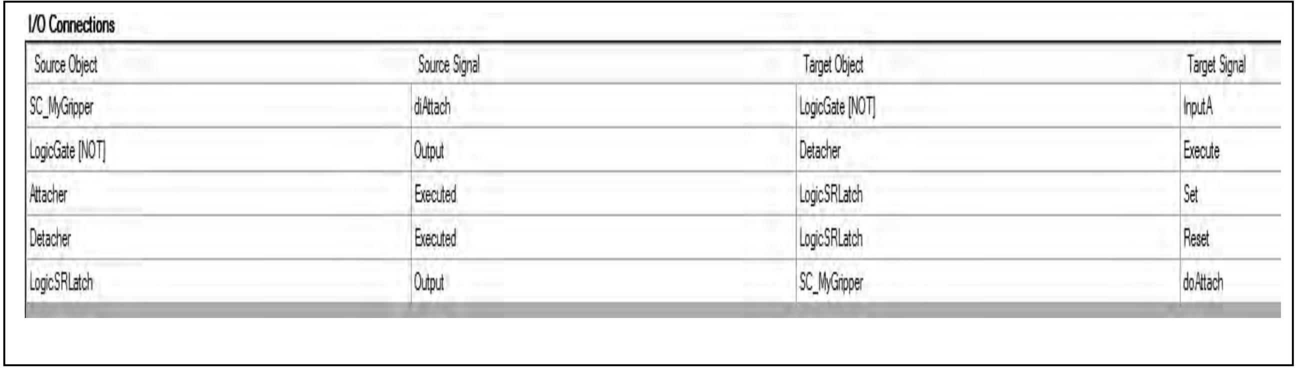
□ Read-only

OK Cancel Help

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

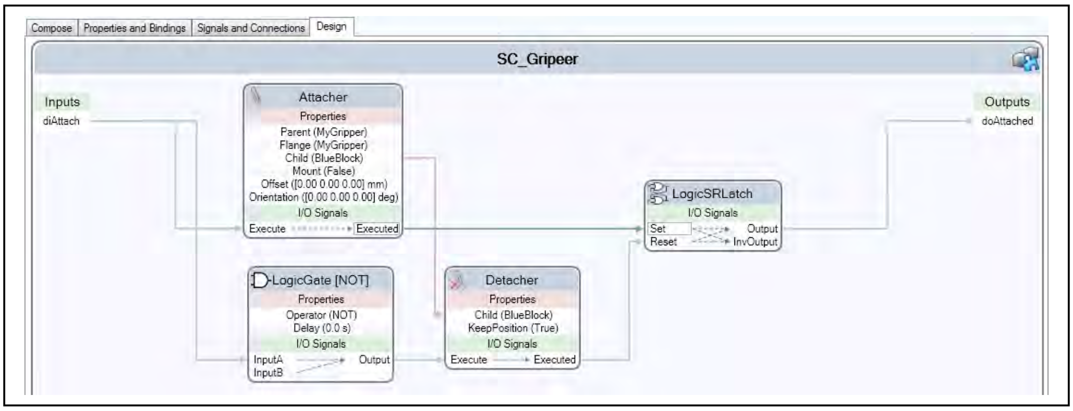


9. Then continue adding **I/O connections** according to the list below. Make sure you go through and understand each step.


I/O Connections

| Source Object | Source Signal | Target Object | Target Signal |
|---|---|---|---|
| SC_MyGripper | diAttach | LogicGate [NOT] | InputA |
| LogicGate [NOT] | Output | Detacher | Execute |
| Attacher | Executed | LogicSRLatch | Set |
| Detacher | Executed | LogicSRLatch | Reset |
| LogicSRLatch | Output | SC_MyGripper | doAttach |

10. Now we have finished our **SC**. To get an overview, 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 components around to arrange them according to your preference.



11.Save the **SC_MyGripper** as … *Libraries \SC_MyGripper.rslib.*
12. Save the station as *MyABBpick&place_SxGx_s15.*

## Programming and simulating I/O signals

### Goal of the chapter

- In this exercise we will learn how to use Smart Components in a simulation. After this is done we will use the RAPID Editor to edit the procedures to complete our program. We will learn to use functions such as *grip_block1* and *ungrip_block1*.

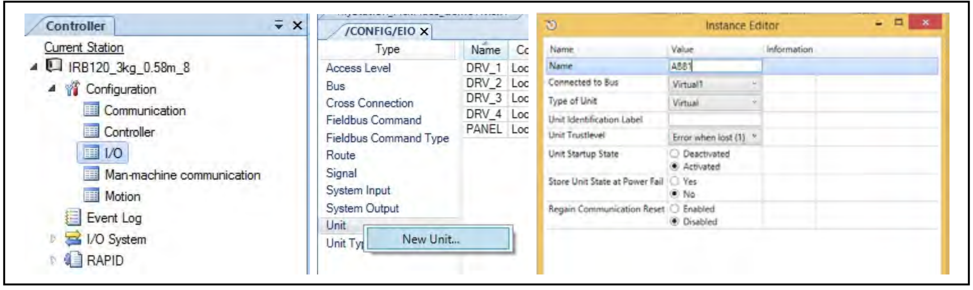
- We will also learn how to define a variable that is required by VC. This variable will be used in RAPID Editor.

## 3.1 Working with Smart Components

- Next step is to setup I/O connections between the Smart Components and the Virtual Controller to get a complete simulation

### 3.1.1 Define unit and signal

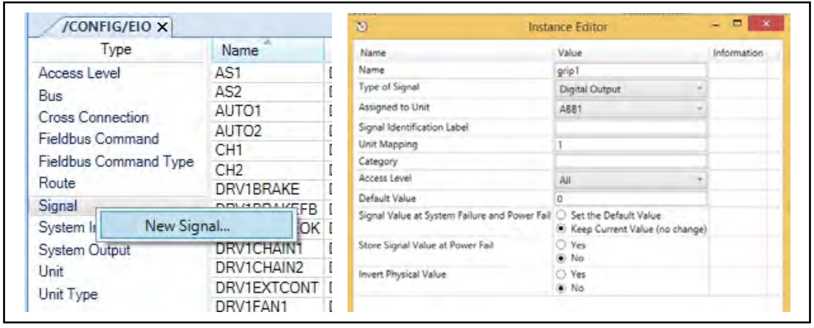
1. Open the station from the last exercise (*MyABBpick&place_SxGx_s15*), unless it is already open.
2. Under **Controller** tab, click **I/O** from **Configuration** context menu.
3. Right click on **Unit** to add *New Unit*. Then set the new unit according to the picture below



| Name | Value | Information |
|---|---|---|
| Name | A881 | |
| Connected to Bus | Virtual1 | |
| Type of Unit | Virtual | |
| Unit Identification Label | | |
| Unit Trustlevel | Error when lost (1) | |
| Unit Startup State | ○ Deactivated<br>● Activated | |
| Store Unit State at Power Fail | ○ Yes<br>● No | |
| Regain Communication Reset | ○ Enabled<br>● Disabled | |

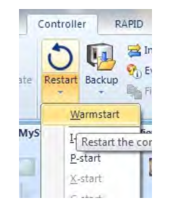
*This unit will be further used by **grip1**signal*

4. Click *OK* to the prompted message. We will do **warm-restart** later.
5. Right click on **Signal** to add *New Signal*. Then set the new signal according to the picture below



| Name | Value | Information |
|---|---|---|
| Name | grip1 | |
| Type of Signal | Digital Output | |
| Assigned to Unit | A881 | |
| Signal Identification Label | | |
| Unit Mapping | 1 | |
| Category | | |
| Access Level | All | |
| Default Value | 0 | |
| Signal Value at System Failure and Power Fail | ○ Set the Default Value<br>● Keep Current Value (no change) | |
| Store Signal Value at Power Fail | ○ Yes<br>● No | |
| Invert Physical Value | ○ Yes<br>● No | |

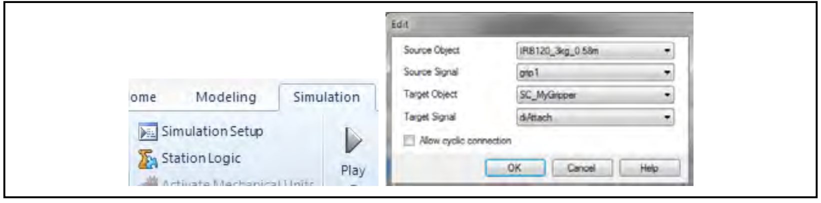
6. Do warm-restart to ensure the changes are effective.



7. Saved station as *MyABBpick&place_SxGx_s16*.

### 3.1.2 Signal and connection

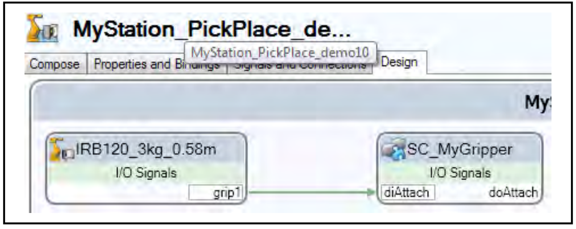
1. Open the station from the last exercise (*MyABBpick&place_SxGx_s16*), unless it is already open.
2. In the ribbon of the **Simulation** tab, click **Station Logic.**
3. On the **Signals and Connections** tab, click **Add I/O Connection**.
4. Set **Source Object** to *IRB120_3kg_0.58m_8*, **Source Signal** to *grip1*, **Target Object** to *SC_MyGripper* (the Virtual Controller) and **Target Signal** to *diAttach.* Click **OK**. We have now created our first I/O Connection.



| | |
|---|---|
| Source Object | IRB120_3kg_0.58m |
| Source Signal | grip1 |
| Target Object | SC_MyGripper |
| Target Signal | diAttach |

☐ Allow cyclic connection

5. Now we have finished our **MyStation_PickPlace**. To get an overview, click the **Design** tab.



6. Save the station as *MyABBpick&place_SxGx_s17*.

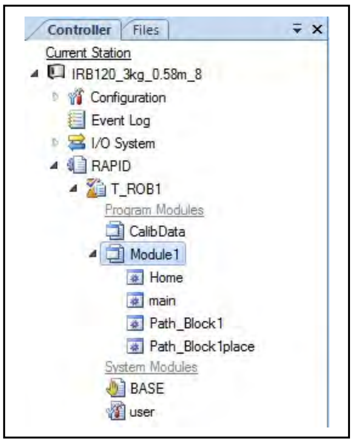
## 3.2 Using RAPID Editor

**RAPID Editor**

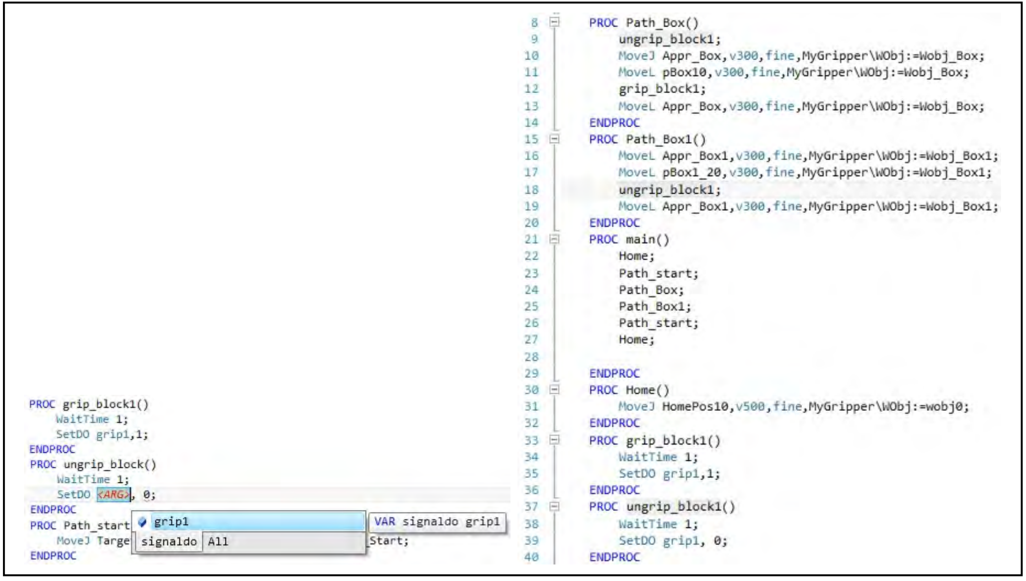
- We will now finalize the RAPID program by adding action instructions. This can be done either from the **Home** tab or manually writing the instructions in the **Rapid** editor. In this example, we will focus on the editor.

### 3.2.1 Editing program

1. Open the station from the last exercise (*MyABBpick&place_SxGx_s17*), unless it is already open.
2. Go to the **Rapid** tab and open the **RAPID Editor** for *Module1.*



3. From the last program, write up the **grip_block1** and **ungrip_block1** functions, and call them in **Path_Block1** and **Path_Block1**place function as shown in lines 8, 11 and 26.

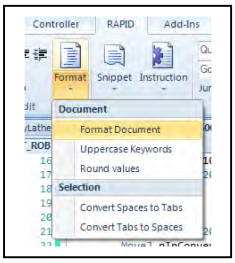


```
PROC grip_block1()
    WaitTime 1;
    SetDO grip1,1;
ENDPROC
PROC ungrip_block()
    WaitTime 1;
    SetDO <ARG>, 0;
ENDPROC
PROC Path_start
    MoveJ Targe
ENDPROC
```

```
8   PROC Path_Box()
9       ungrip_block1;
10      MoveJ Appr_Box,v300,fine,MyGripper\WObj:=Wobj_Box;
11      MoveL pBox10,v300,fine,MyGripper\WObj:=Wobj_Box;
12      grip_block1;
13      MoveL Appr_Box,v300,fine,MyGripper\WObj:=Wobj_Box;
14  ENDPROC
15  PROC Path_Box1()
16      MoveL Appr_Box1,v300,fine,MyGripper\WObj:=Wobj_Box1;
17      MoveL pBox1_20,v300,fine,MyGripper\WObj:=Wobj_Box1;
18      ungrip_block1;
19      MoveL Appr_Box1,v300,fine,MyGripper\WObj:=Wobj_Box1;
20  ENDPROC
21  PROC main()
22      Home;
23      Path_start;
24      Path_Box;
25      Path_Box1;
26      Path_start;
27      Home;
28
29  ENDPROC
30  PROC Home()
31      MoveJ HomePos10,v500,fine,MyGripper\WObj:=wobj0;
32  ENDPROC
33  PROC grip_block1()
34      WaitTime 1;
35      SetDO grip1,1;
36  ENDPROC
37  PROC ungrip_block1()
38      WaitTime 1;
39      SetDO grip1, 0;
40  ENDPROC
```

*When you add **WaitTime** or **SetDo**, you will notice that you are prompted to add the argument. By holding Ctrl and hitting the space bar while on the <ARG> you will get a dropdown menu of all configured signals present in the controller. (Notice you can also select the tab to open up all commands.)*

4. Once your main procedure is done click on the lower half of the **Format** button and select format document.



5. Click **Apply** changes.
6. Confirm the change by clicking **YES** and answer YES for the subsequent message/s.

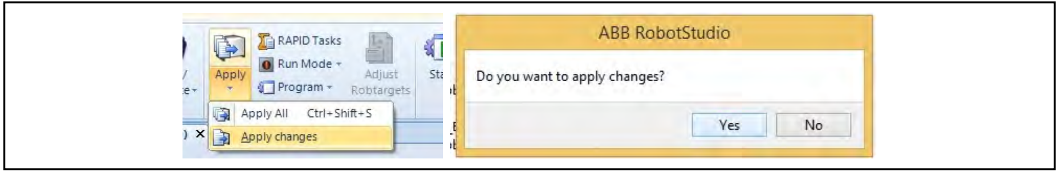
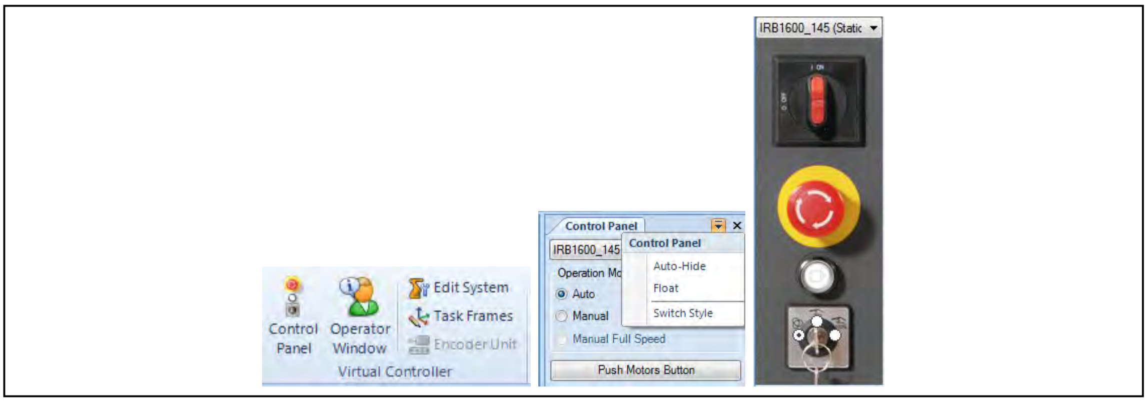


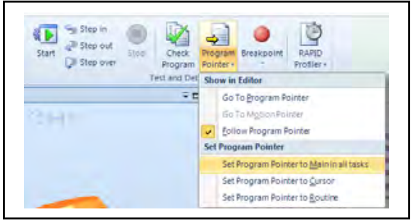
ABB RobotStudio

Do you want to apply changes?

7. In the Virtual Controller group of the **Controller** tab, open **the Control Panel** and change to **AUTO** mode and push the **Motors Button**.



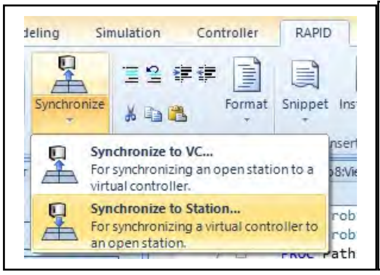
8. Back in the **Rapid** tab Set the program pointer to the **main** routine

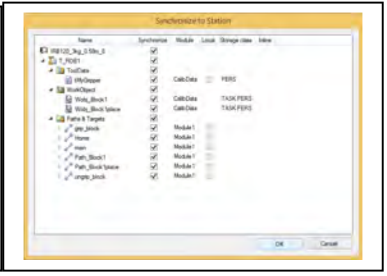


9. Use the Step in (**F11**) function Step in to step through the program.

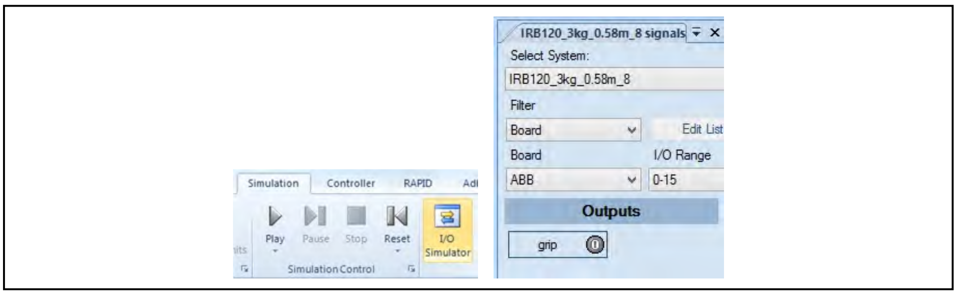
*As we now have done changes to the program directly in the* ***virtual controller****, we need to synchronize the changes back to the* ***station****.*

10. While still in the **Rapid** tab, click the lower half of the **Synchronize** button to reveal the **Synchronize to Station** feature. Synchronize to the station.
11. In the dialog, check the top node so that everything get selected and press **OK**.


deling
Simulation
Controller
RAPID
Synchronize
Format
Snippet
Synchronize to VC...
For synchronizing an open station to a virtual controller.
Synchronize to Station...
For synchronizing a virtual controller to an open station.



12. Go to the **Simulation** tab and open up the **I/O Simulator**.



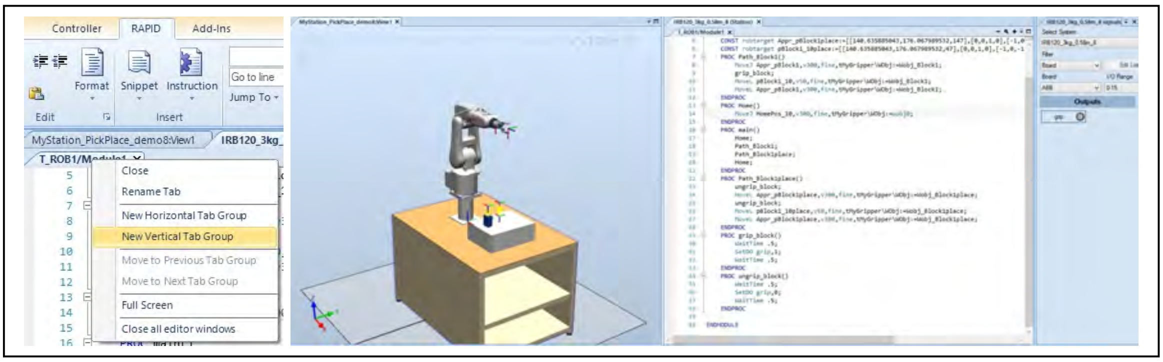
Select System: IRB120_3kg_0.58m_8

Filter: Board

Board: ABB

I/O Range: 0-15

13. Switch back to the **Rapid** tab and right click on the module tab you would like to view. Select New Vertical Tab Group.



```
CONST robtarget Appr_pBlock1place:=[[140.635885043,176.067989532,147],[0,0,1,0],[-1,0
CONST robtarget pBlock1_10place:=[[140.635885043,176.067989532,47],[0,0,1,0],[-1,0,-1
PROC Path_Block1()
    MoveJ Appr_pBlock1,v300,fine,tMyGripper\WObj:=Wobj_Block1;
    grip_block;
    MoveL pBlock1_10,v50,fine,tMyGripper\WObj:=Wobj_Block1;
    MoveL Appr_pBlock1,v300,fine,tMyGripper\WObj:=Wobj_Block1;
ENDPROC
PROC Home()
    MoveJ HomePos_10,v300,fine,tMyGripper\WObj:=wobj0;
ENDPROC
PROC main()
    Home;
    Path_Block1;
    Path_Block1place;
    Home;
ENDPROC
PROC Path_Block1place()
    ungrip_block;
    MoveL Appr_pBlock1place,v300,fine,tMyGripper\WObj:=Wobj_Block1place;
    ungrip_block;
    MoveL pBlock1_10place,v50,fine,tMyGripper\WObj:=Wobj_Block1place;
    MoveL Appr_pBlock1place,v300,fine,tMyGripper\WObj:=Wobj_Block1place;
ENDPROC
PROC grip_block()
    WaitTime .5;
    SetDO grip,1;
    WaitTime .5;
ENDPROC
PROC ungrip_block()
    WaitTime .5;
    SetDO grip,0;
    WaitTime .5;
ENDPROC

ENDMODULE
```

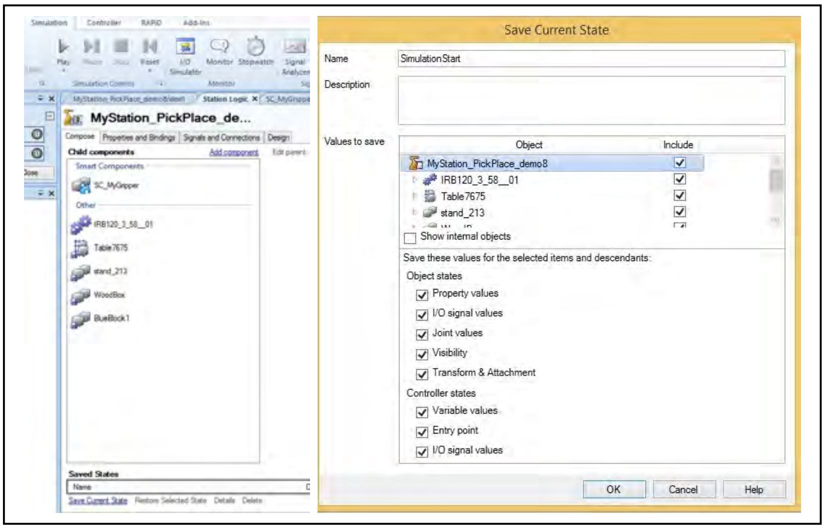
*This way you can see the simulation, the RAPID execution and handle IOs*

14. Save the station as *MyABBpick&place_SxGx_s18*.

### 3.2.2 Save initial state

Each SC that consumes time (e.g. moving an object) needs to have a simulation running. But if the RAPID Editor should be used for debugging and editing, the Virtual Controller has to be excluded from that simulation.

1. Open the station from the last exercise (*MyABBpick&place_SxGx_s18*), unless it is already open.
2. On the **Compose** tab, click **Save Current State**.
3. Name the Current State **SimulationStart**. Check all values for **Object states** and **I/O Signal values** for **Virtual Controller states** and Smart Components as shown below and press OK.



Save Current State

Name: SimulationStart

Description:

Values to save:

| Object | Include |
|---|---|
| MyStation_PickPlace_demo8 | ☑ |
| IRB120_3_58__01 | ☑ |
| Table7675 | ☑ |
| stand_213 | ☑ |

☐ Show internal objects

Save these values for the selected items and descendants:

Object states
- ☑ Property values
- ☑ I/O signal values
- ☑ Joint values
- ☑ Visibility
- ☑ Transform & Attachment

Controller states
- ☑ Variable values
- ☑ Entry point
- ☑ I/O signal values

*When **Simulation Play** is pressed, all checked objects will go back to the state it had when the **Current State** was saved, so it is important that joint values, I/O signals etc. has the correct state **before** the state is saved.*

4. In **Simulation** tab, click the **Play** button. Note that the robot now first go to the Home position we added to the main procedure from the **Rapid Editor**.
5. Save the station as *MyABBpick&place_SxGx_s19*.