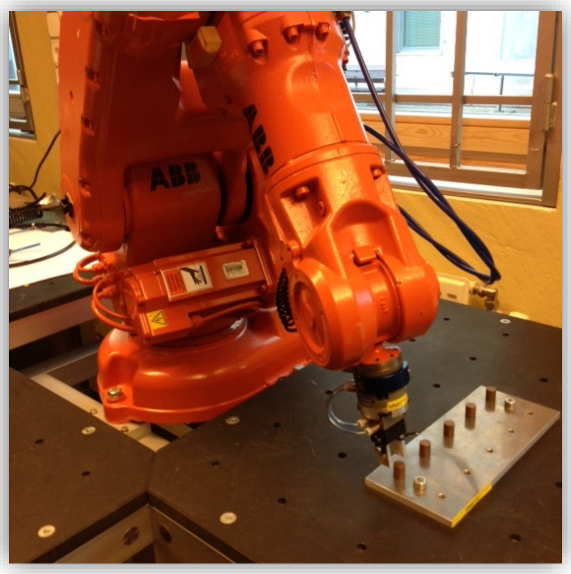
Exercise 2, Robot Studio, using a gripper tool

For RobotStudio version 6.05

Gunnar Bolmsjö, [rev Sept 2017, ARo/ERo]

The aim of this exercise is to make some simple control of the environment via digital outputs and understand the basics in moving objects around in a simulation environment. The work scenario is a robot that picks up a workpiece at a specific location and place it at another location. The robot station is the same as in Exercise 1 (ABB IRB140). The instruction of this exercise is less detailed than the first exercise, but includes very detailed information of all new issues introduced in RobotStudio.


Picture 1: An actual robot performing the program you will write in this exercise

1. Setting up the robot station

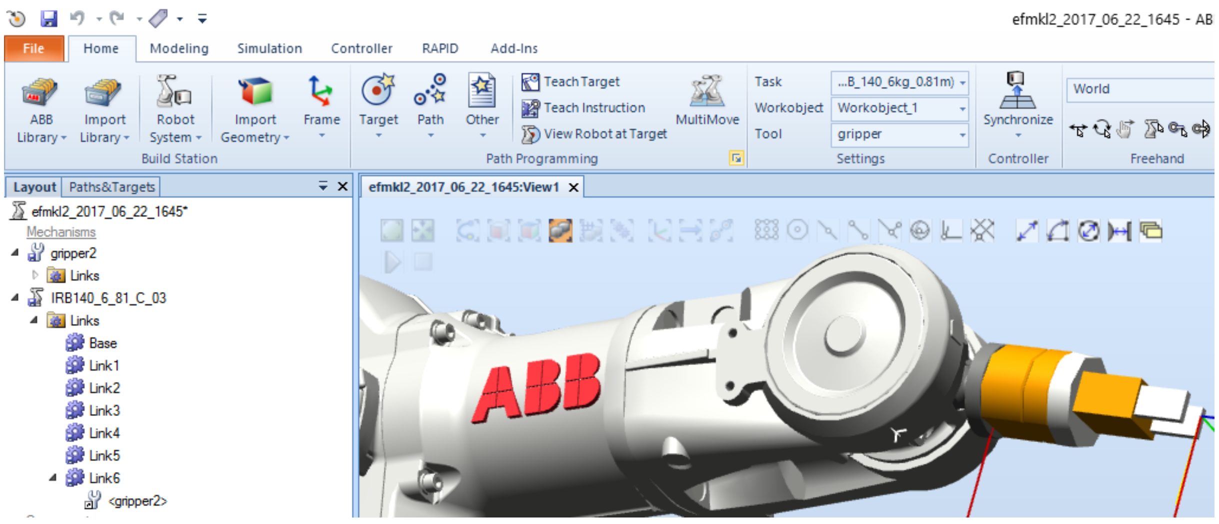
Sept 2017: Remember to work locally on the C:\-drive and afterwards save your station as Pack&Go to your home folder (see RobotStudio-exercise 1 for details of the settings).

Start RobotStudio and start a new station using a template system. Select the robot IRB140_6kg_0.81m. After some time, the system will come up with a robot (and the controller status: 1/1 turns green). Import some geometries Import Library -> Browse for library, where you select the following library folder: S:\Courses\design\MMKF15 Robotteknik\RobotStudio_exercises_2016\Lab2 Gripper
Import the following geomtries robot-table, gripper, brick and piece12.

If you zoom in to the gripper, you will notice a cylinder between the gripper fingers. The use of this is a typical “simulation fix” and will be explained later.

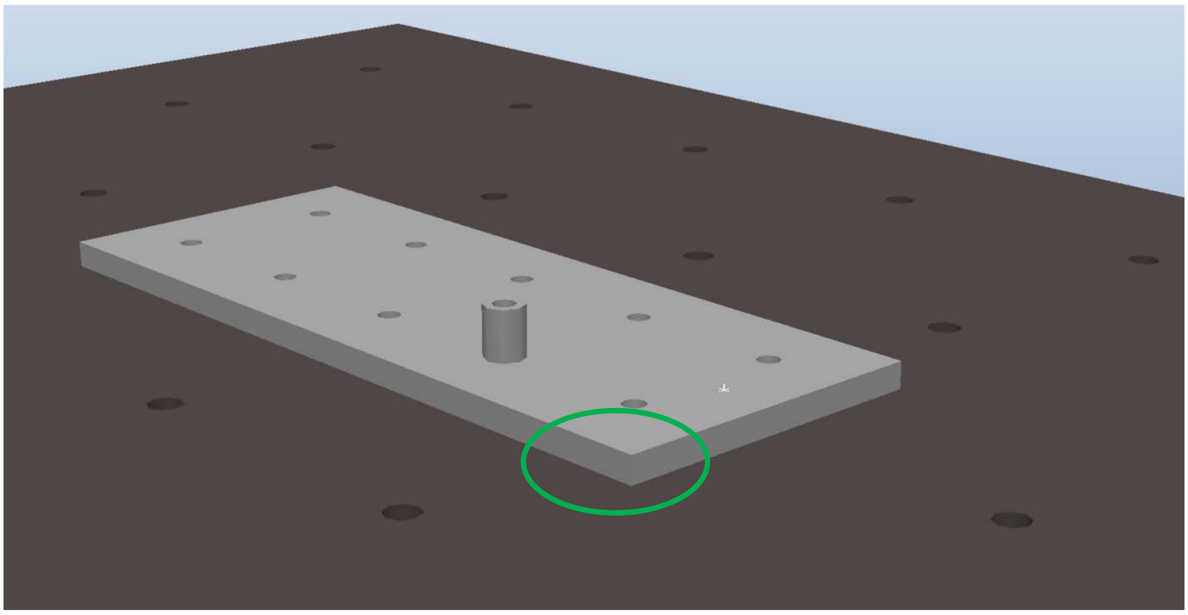
Click on the Home -> Layout -tab, right-click the robot and choose “Position” ->“Set Position”. Set the Z value to 670 mm using “World” as reference and select “Apply”. Confirm that you want to change the “Move Task Frame” by clicking “OK”. Note: You may achieve the same by choosing Task Frames in the Controller tab.

In the Home -> Layout-tab, attach the gripper to the robot (in the left menu, drag the gripper to the robot, click “Yes” to the question about updating the position, and it will snap on to the last link of the robot (see Picture 2).



Picture 2: You can verify that the gripper is attached by looking at the last link of the robot

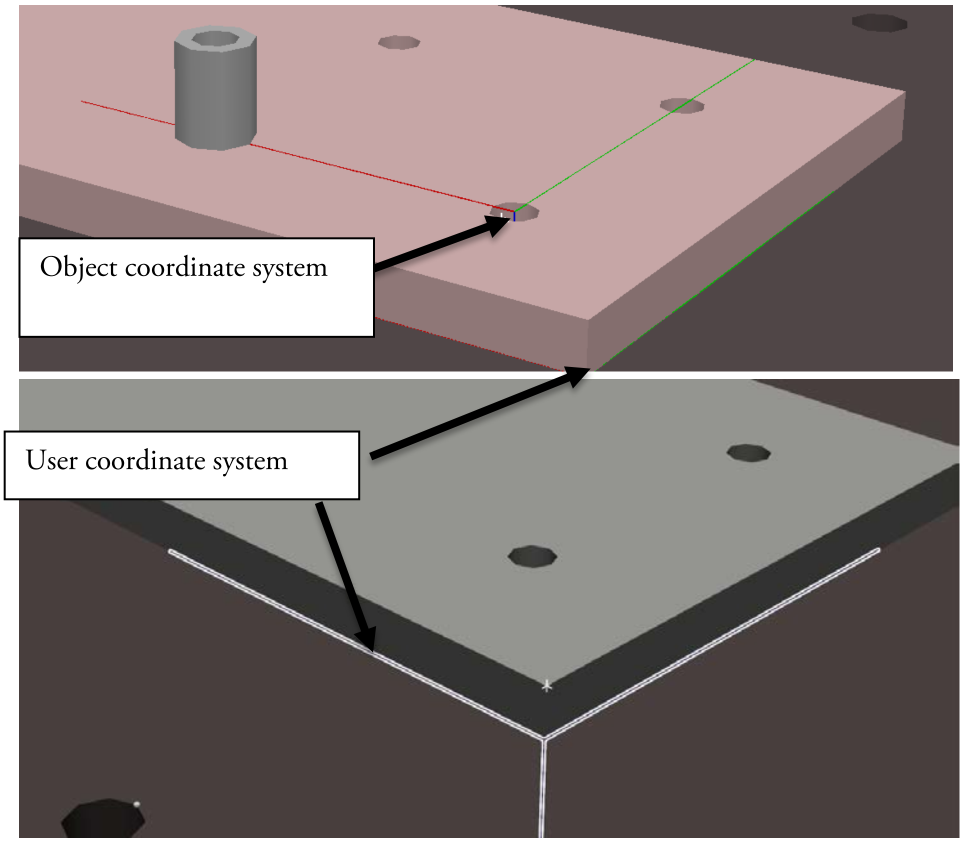
Now, we have the basic robot station set up for the task. Before we continue, we have to define a workobject coordinate frame. Zoom up to a view similar to Picture 3.



Picture 3: Define the workobject by snapping to the indicated corner

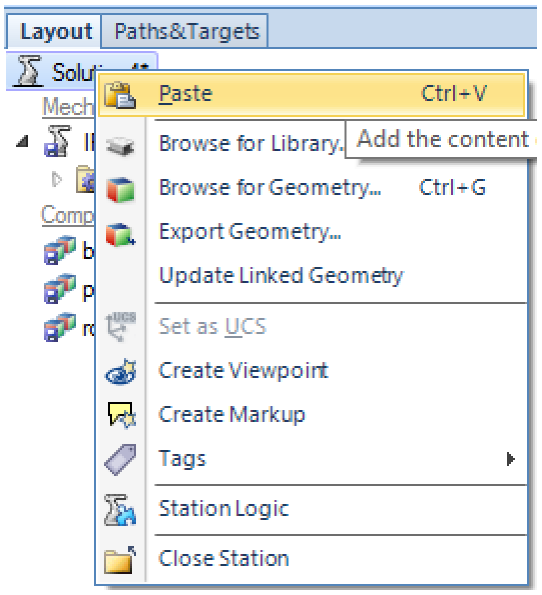
Define a workobject to closest corner in the view so we get a user coordinate system at the bottom of the plate and the workobject coordinate system located at the top of the plate at the first (nearest) hole to that corner. The distance between the holes is 50 mm in x- and y-directions and the distance from the corner to the first hole is 25 mm in x- and y-directions. The thickness of the plate is 8 mm. Also, make sure you get the z-axis pointing down and the x-axis along the long edge of the object. If you are uncertain how to define the correct rotation of the "user frame" when creating the workobject, you can always change it afterwards by right-clicking on the workobject in the "Paths & Target" - tab and choose "Rotate". This will result in frames like the following (see Picture 4).

The object "piece12" exists only at one place now. For the task, we need five such workpieces. If you select the object "piece12", you can copy it and create new by using a normal copy and paste operation. Right-click and select "Copy".



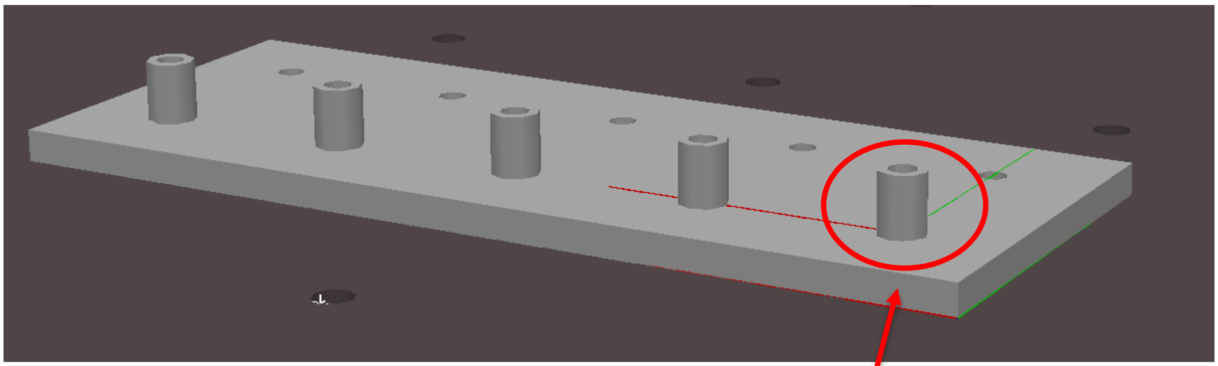
Picture 4: How the coordinate systems should be defined. The object coordinate system might not be visible as shown

Click on the station name (in the top of the "Layout" tab) and right-click, and select "Paste" four times.



Picture 5: Paste your piece four times

Move the objects using "Position"->"Set position" (right-click on a piece12 copy) and use "Local" as reference. Notice that you get a shadowy object when you enter a value for a new position, and the "Apply" executes the command. The result should look like the following:



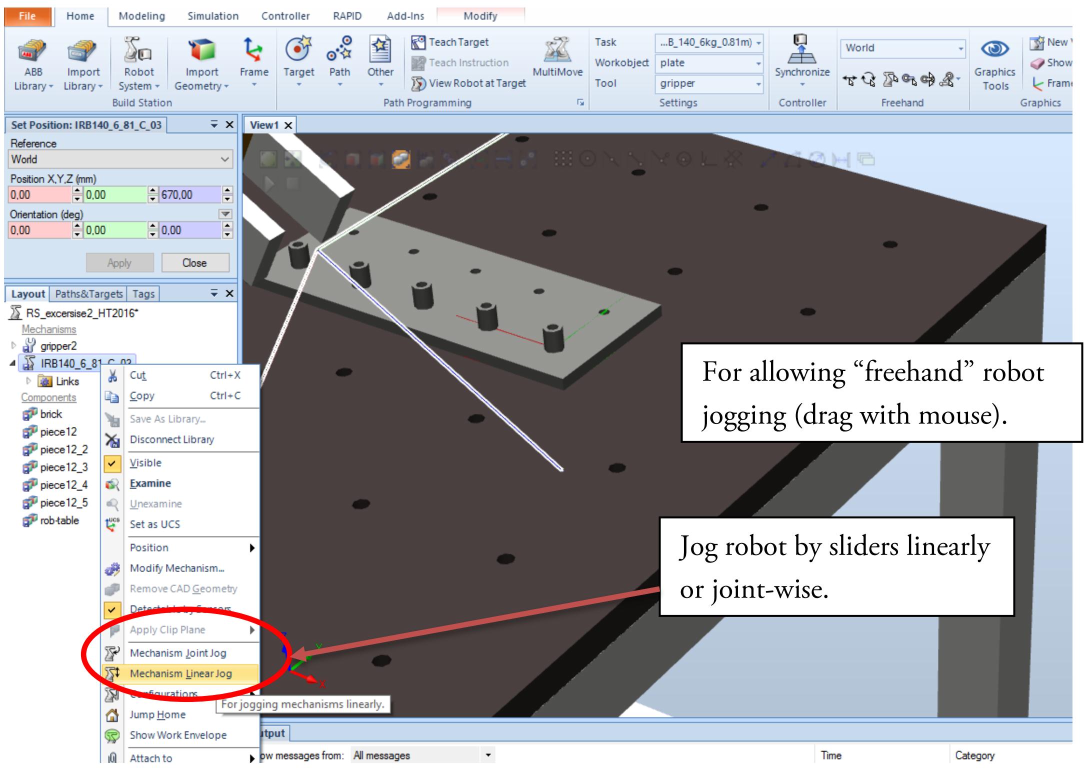
Picture 6: How the "brick" should look after the first part of the exercise

This is a good time to save the station. The next step will be to create the targets and the paths for the pick-and-place operations.

2. Targets and paths

Make a target at the home position of the robot by using Teach Target. Rename the target point to a suitable name so that you easily recognize it in the target list.

(You may also first move the robot to another suitable starting position by e.g., right-clicking on the robot in the Home -> Layout-tab and choose "Mechanism Joint Jog" or "Mechanism Linear Jog" or choosing any of the Freehand jog modes in the Home toolbar (see figure) to interact with the mouse in the graphics view.



For allowing "freehand" robot jogging (drag with mouse).

Jog robot by sliders linearly or joint-wise.

Picture 7: Where you find the "Freehand" robot jogging and the mechanisms for Linear & Joint Jog

Then, the robot should move to the first "piece12", pick it up, and place it in the nearest empty hole, move to the next "piece12", pick that up and place it next to the other one, and so on until the whole row of "piece12" have been moved to the next row.

In practice, the targets for the first "piece12" movement will be something like the following:

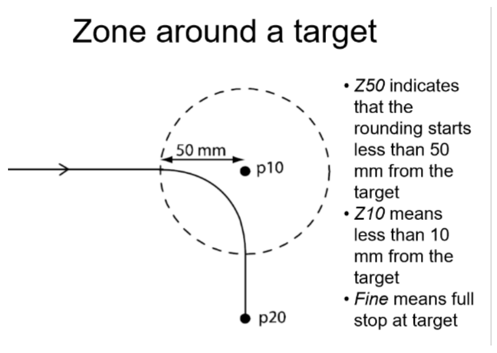
- Move to a target about 50 mm above the piece12
- ADD Rapid-command to open gripper
    - (Note: Even if the gripper might be open already during programming, it may be closed next time you run the program)
- Move down and grasp the piece12
    - Move down
    - Close Gripper
- Move up again
- Move to above the position to where to place the piece12
- Move down to the place position
    - Move down
    - Open gripper
- Move up again
- Repeat for the next "piece12" copy until all copies have been moved.
- …

Assignment:

Create the targets necessary to perform the task as described above. Create one path for the first pick-and-place operation.

Note 1: Similar and almost repeatable operations as described here can be defined by parameters and loops in many ways. However, to better understand the workflow and some other issues, we make the program with individual/different paths and targets. Possible other solutions to the assignment will be shown later.

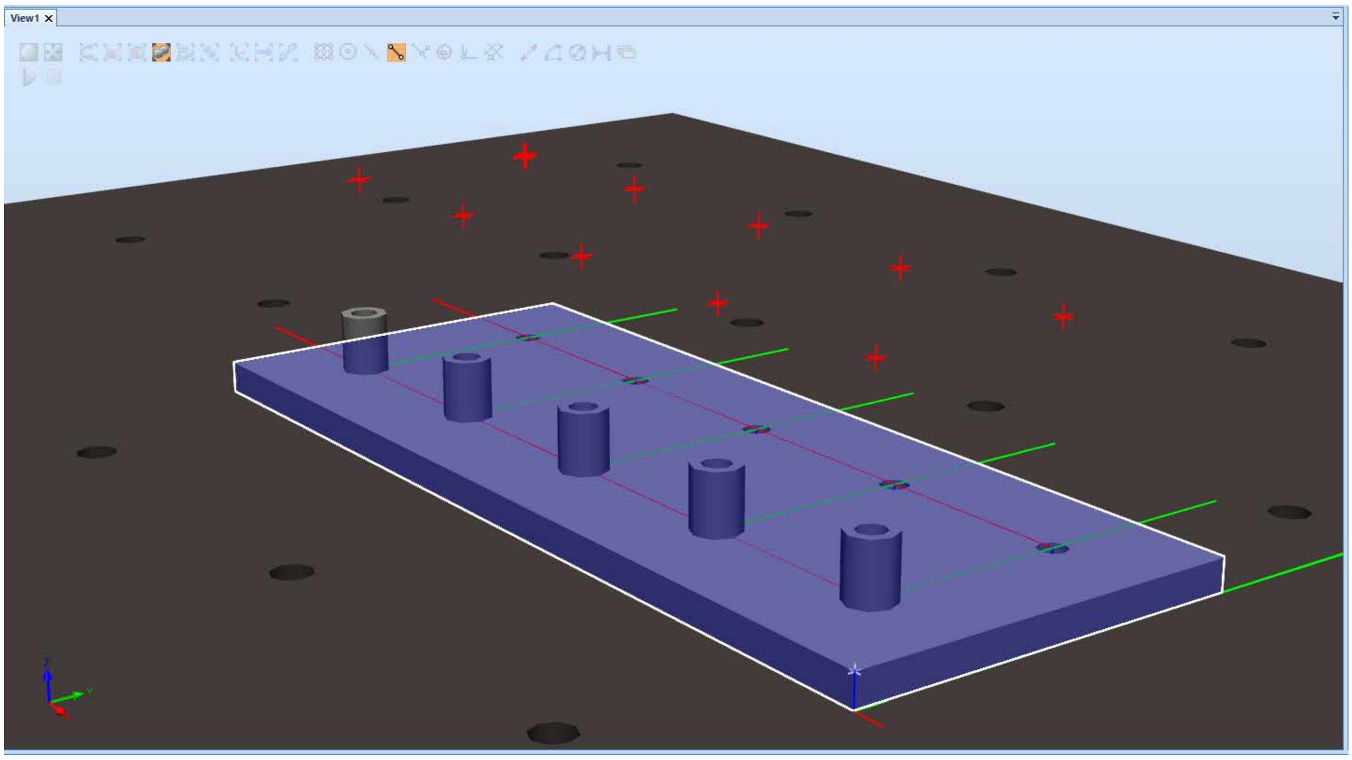
Note 2: Movements from the home position to the first target above the first piece12, as well as moving from the last piece12 to the home position should be MoveJ. Use zone value z10 for the target above the first piece12.


Zone around a target
50 mm
p10
p20
• Z50 indicates that the rounding starts less than 50 mm from the target
• Z10 means less than 10 mm from the target
• Fine means full stop at target

Picture 8: What the z-value (zone) means

Note 3: All other movements should be MoveL. Use velocity v100 and zone z10 except for the pick / place targets which should have a "fine-point" value (i.e., the robot makes a full stop in that point and does not make any "shortcuts" on its way to the next target point).

After defining the target points they will be located as the following figure illustrates, if you have defined all the targets. In this exercise, it is enough to just define the targets for the first pick-and-place. We will later copy that path in a specific manner:



Picture 9: Here we have defined the ten targets above the ten holes, and we have defined targets in the middle of the holes themselves

Now, fill the paths with targets to produce the intended task. Name targets for the respective paths, for example Pick1_10 … _40 for path1. Remember to use workobject as reference when defining coordinate values. All paths must be checked for configuration ("Auto Configuration").

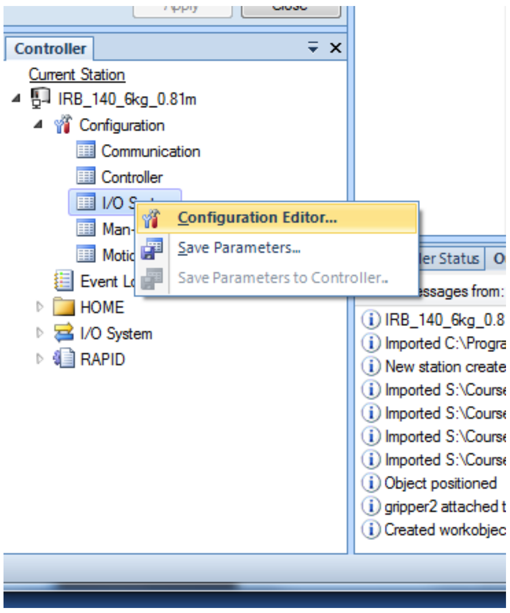
Before we continue with the other paths, we will study the first path in more detail and produce a complete working program, which can be simulated under Simulation. After the fine-point targets (pick-and-place targets), insert "Action Instructions". Right click where to insert (after an instruction), use the "Insert Action Instruction", select "WaitTime" as default, delete "END_OF_LIST" and enter "2" (seconds).

Save your station.

Make a synchronization to RAPID. Go to the Simulation tab and Simulation Setup. Include the path (Pick1 for example) to the main procedure. (In case your do not see a "main" procedure, just type one in the RAPID code and write a procedure call to your path. Apply changes and Synchronize to station). Run the simulation. If everything works fine, the robot makes the motions as it should but the piece12 is not moving.

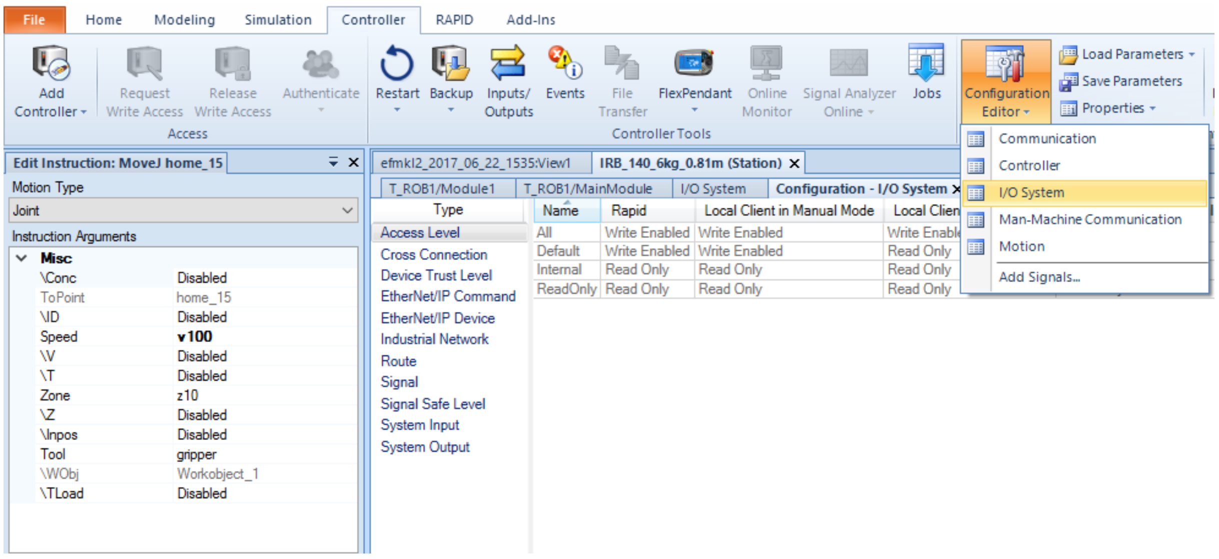
Next, we will set up the I/O connections for the robot. On the controller of the robot in the lab, there are (in our case) 16 digital outputs available (0-15), and the digital output signals DO 0 and 1 are connected to control the gripper. To close the gripper, DO 0 must be "0" and DO 1 must be "1", to open the gripper, DO 0 must be "1" and DO 1 must be "0". In a system for programming robots, we work with logical names / variables, which are mapped to physical connections as set-up data. We will use the names "DO10_0" and "DO10_1" in the RAPID-code below for controlling the gripper.

First, we configure the I/O unit on the virtual controller. (The physical robot controller in the lab already has these settings so it is IMPORTANT to use the same names for the signals to be able to move your program and run it successfully at the end of the exercise). Go to the RAPID tab, expand the "Configuration" so you see "I/O System". Right-click, and open the "Configuration Editor", see Picture 10:



Picture 10: How to open the "Configuration Editor" from the "Controller" Window

You can also open the "Configuration Editor" -> I/O System by clicking on the Configuration Editor icon, and then choosing "I/O System" as indicated in Picture 11.



Motion Type: Joint

Instruction Arguments

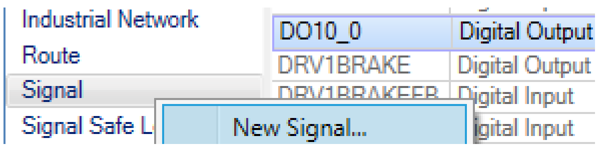
Misc	
\Conc	Disabled
ToPoint	home_15
\ID	Disabled
Speed	v100
\V	Disabled
\T	Disabled
Zone	z10
\Z	Disabled
\Inpos	Disabled
Tool	gripper
\WObj	Workobject_1
\TLoad	Disabled

Type
Access Level
Cross Connection
Device Trust Level
EtherNet/IP Command
EtherNet/IP Device
Industrial Network
Route
Signal
Signal Safe Level
System Input
System Output

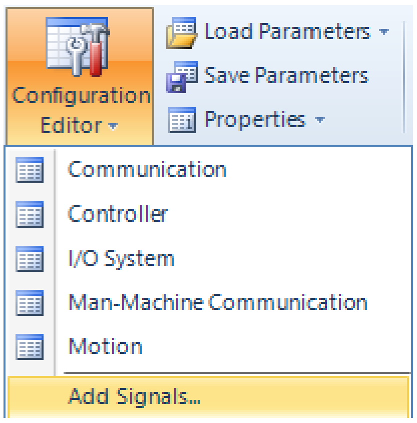
Name	Rapid	Local Client in Manual Mode	Local Clien
All	Write Enabled	Write Enabled	Write Enabl
Default	Write Enabled	Write Enabled	Read Only
Internal	Read Only	Read Only	Read Only
ReadOnly	Read Only	Read Only	Read Only

Picture 11: Another wat of opening the Configuration Editor

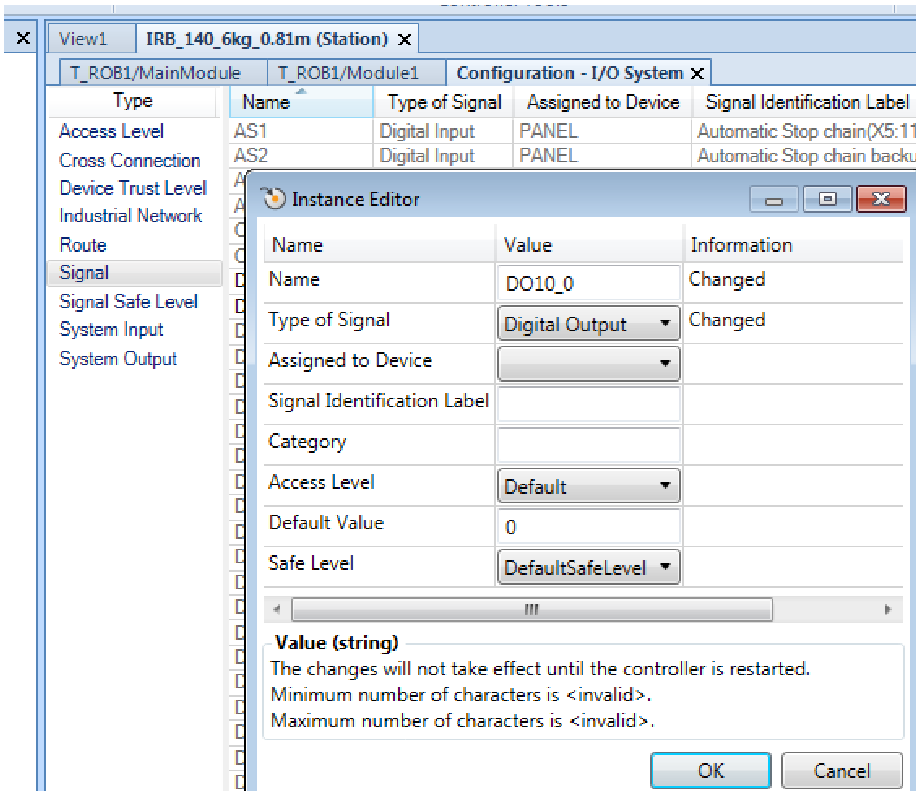
Right-click on "Signal" and add "New signal" or choose "Configuration editor-> Add signal". The different alternatives result in different pop-up windows. If you pressed "Add Signal", make sure the name is correct after you have created your signal. The name can change when the signal is created, even if entered correctly. To change the name, right-click and press "Edit Signal(s)".



Picture 12: The first option for adding a "New Signal"



Picture 13: The second option for adding a "New Signal"



Type
Access Level
Cross Connection
Device Trust Level
Industrial Network
Route
Signal
Signal Safe Level
System Input
System Output

Name	Type of Signal	Assigned to Device	Signal Identification Label
AS1	Digital Input	PANEL	Automatic Stop chain(X5:11
AS2	Digital Input	PANEL	Automatic Stop chain backu

Instance Editor

Name	Value	Information
Name	DO10_0	Changed
Type of Signal	Digital Output	Changed
Assigned to Device		
Signal Identification Label		
Category		
Access Level	Default	
Default Value	0	
Safe Level	DefaultSafeLevel	

Value (string)
The changes will not take effect until the controller is restarted.
Minimum number of characters is <invalid>.
Maximum number of characters is <invalid>.

Picture 14: Fill in this information in the pop-up window. If you are asked to change the drive unit name, leave the drive unit name empty

<u>ALTERNATIVE:</u> You may also define the signals with some more options from Controller -> Configuration Editor -> "Add Signals"

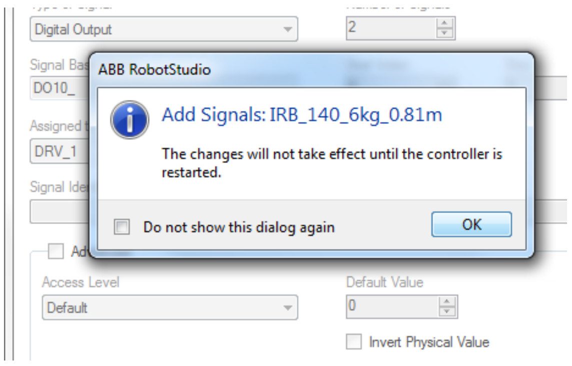


ABB RobotStudio

Add Signals: IRB_140_6kg_0.81m

The changes will not take effect until the controller is restarted.

Do not show this dialog again

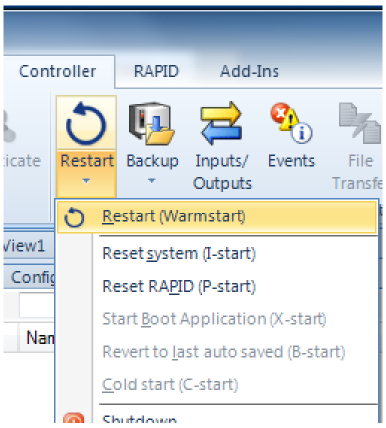
Picture 15: You will get a similar pop-up after creating your signal

Specify "Name" as "DO10_0", Type of signal as "Digital Output, ("Assign to Device" <none>) and set "Unit Mapping" as "0" (zero).

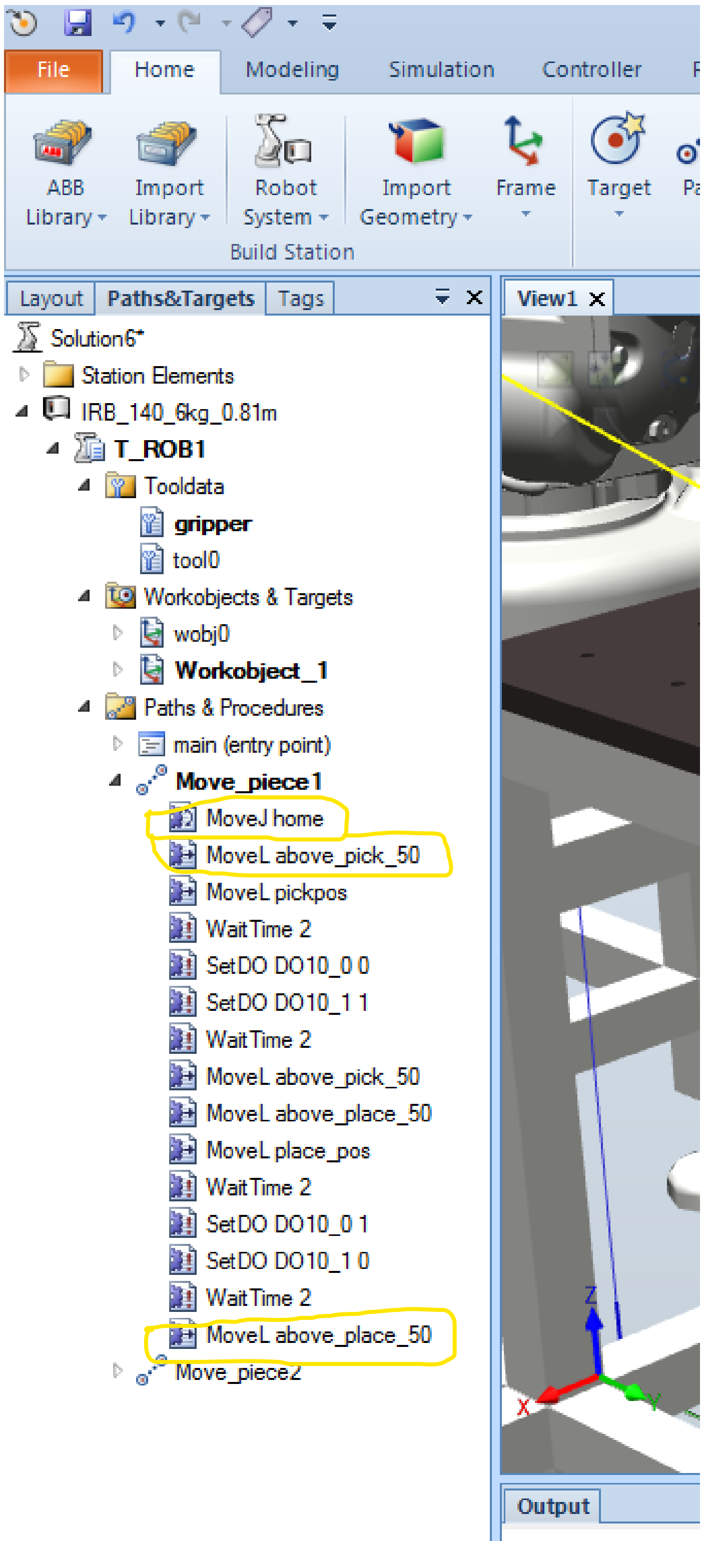
Make another signal with the name DO10_1 and mapping as "1" (meaning the physical connection of the digital output on the physical robot).

Please, make sure that your signals are named "DO10_0" and "DO10_1" and not "DO10_00" and "DO10_01" as the real robot will then not find these names in its configuration. In RobotStudio v.6.05, there is nothing called "Unit Mapping". Instead, it is called "Destination Mappping", but if you cannot assign a value to it, the leave it as not assigned and move along with the exercise.

To activate the changed configuration, we need to do a "warm start" of the controller. In the Controller tab, there is a command/icon "Restart", select "Warmstart", and wait until the Controller status (lower right corner) turns green again.



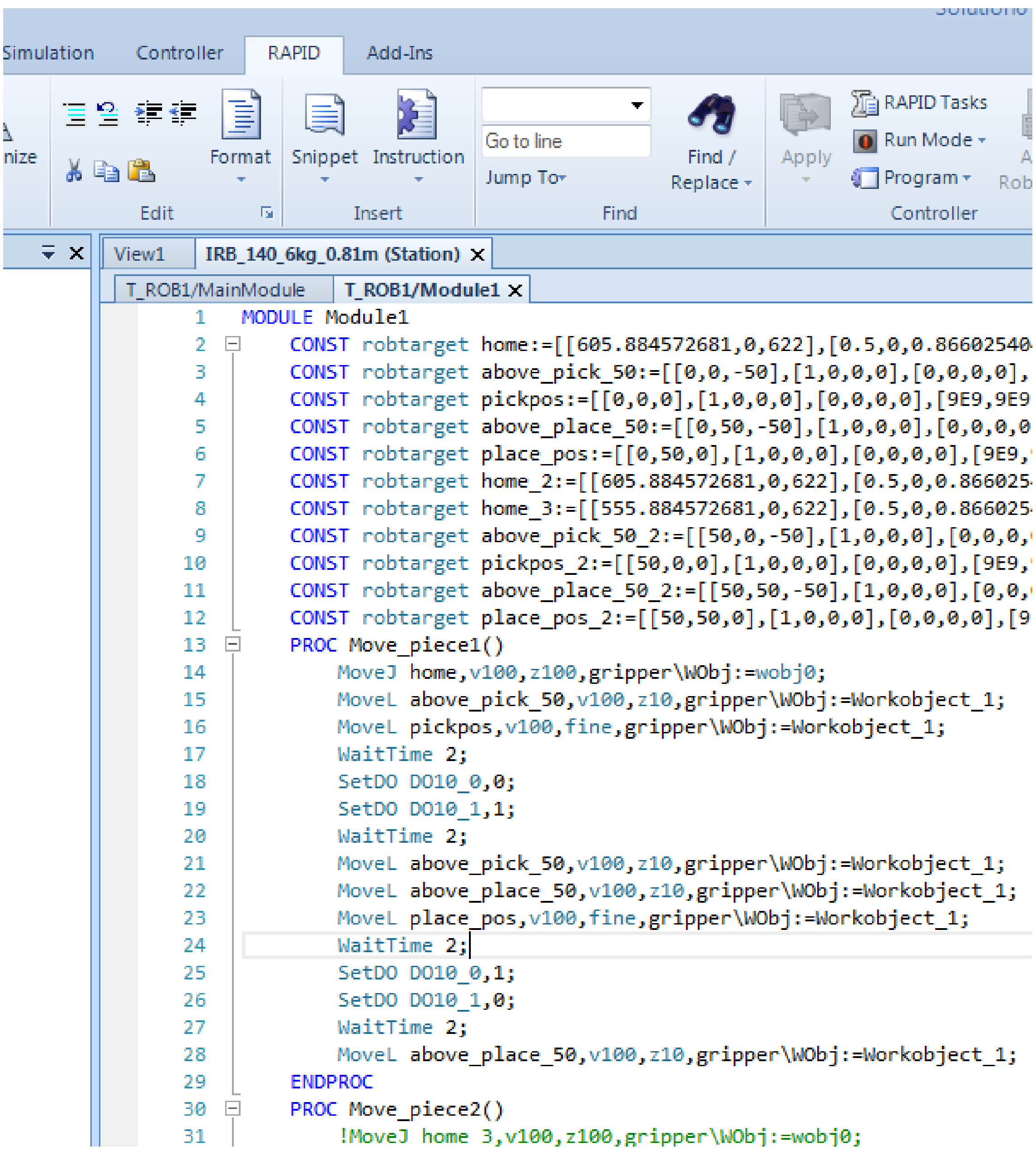
We can now use the digital outputs to control the gripper. Insert an "Action instruction" in the path for the gripper. To close the gripper, set DO10_0 to "0" and DO10_1 to "1". To open, do the vice versa. The result should be like this:



Solution6*
Station Elements
IRB_140_6kg_0.81m
T_ROB1
Tooldata
gripper
tool0
Workobjects & Targets
wobj0
Workobject_1
Paths & Procedures
main (entry point)
Move_piece1
MoveJ home
MoveL above_pick_50
MoveL pickpos
WaitTime 2
SetDO DO10_0 0
SetDO DO10_1 1
WaitTime 2
MoveL above_pick_50
MoveL above_place_50
MoveL place_pos
WaitTime 2
SetDO DO10_0 1
SetDO DO10_1 0
WaitTime 2
MoveL above_place_50
Move_piece2

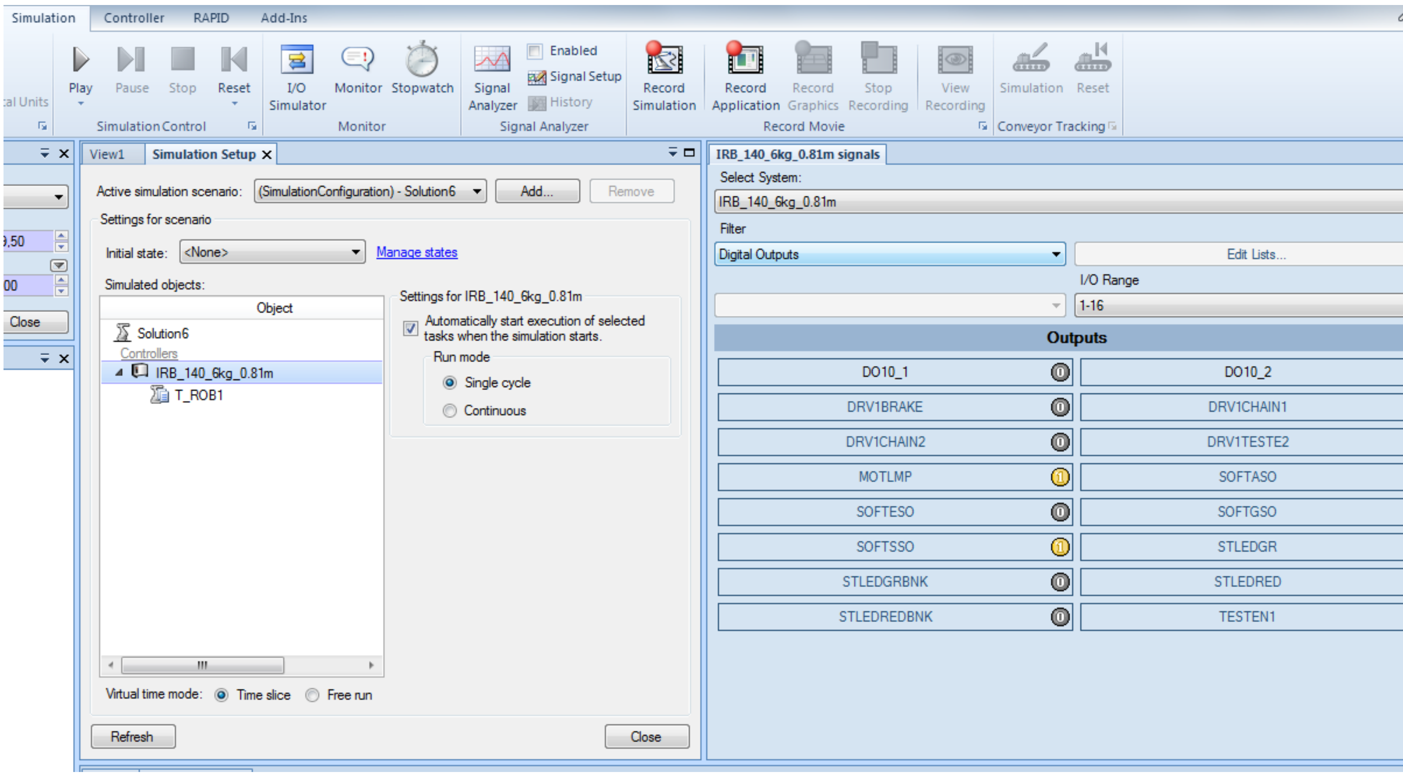
?

Picture 16: The result should look similar to the picture



```
MODULE Module1
    CONST robtarget home:=[[605.884572681,0,622],[0.5,0,0.86602540
    CONST robtarget above_pick_50:=[[0,0,-50],[1,0,0,0],[0,0,0,0],
    CONST robtarget pickpos:=[[0,0,0],[1,0,0,0],[0,0,0,0],[9E9,9E9
    CONST robtarget above_place_50:=[[0,50,-50],[1,0,0,0],[0,0,0,0
    CONST robtarget place_pos:=[[0,50,0],[1,0,0,0],[0,0,0,0],[9E9,
    CONST robtarget home_2:=[[605.884572681,0,622],[0.5,0,0.866025
    CONST robtarget home_3:=[[555.884572681,0,622],[0.5,0,0.866025
    CONST robtarget above_pick_50_2:=[[50,0,-50],[1,0,0,0],[0,0,0,
    CONST robtarget pickpos_2:=[[50,0,0],[1,0,0,0],[0,0,0,0],[9E9,
    CONST robtarget above_place_50_2:=[[50,50,-50],[1,0,0,0],[0,0,
    CONST robtarget place_pos_2:=[[50,50,0],[1,0,0,0],[0,0,0,0],[9
    PROC Move_piece1()
        MoveJ home,v100,z100,gripper\WObj:=wobj0;
        MoveL above_pick_50,v100,z10,gripper\WObj:=Workobject_1;
        MoveL pickpos,v100,fine,gripper\WObj:=Workobject_1;
        WaitTime 2;
        SetDO DO10_0,0;
        SetDO DO10_1,1;
        WaitTime 2;
        MoveL above_pick_50,v100,z10,gripper\WObj:=Workobject_1;
        MoveL above_place_50,v100,z10,gripper\WObj:=Workobject_1;
        MoveL place_pos,v100,fine,gripper\WObj:=Workobject_1;
        WaitTime 2;
        SetDO DO10_0,1;
        SetDO DO10_1,0;
        WaitTime 2;
        MoveL above_place_50,v100,z10,gripper\WObj:=Workobject_1;
    ENDPROC
    PROC Move_piece2()
        !MoveJ home_3,v100,z100,gripper\WObj:=wobj0;
```

Now, we are ready for a test run. Synchronize to Station. Open the Simulation Setup and check that the path is in the main procedure or called from it. Click on the I/O Simulator and run the simulation.



Active simulation scenario: (SimulationConfiguration) - Solution6

Settings for scenario

Initial state: <None>

Simulated objects:

Solution6
Controllers
IRB_140_6kg_0.81m
T_ROB1

Settings for IRB_140_6kg_0.81m

[✓] Automatically start execution of selected tasks when the simulation starts.

Run mode: [✓ Single cycle] Continuous

Virtual time mode: [✓ Time slice] Free run

IRB_140_6kg_0.81m signals

Select System: IRB_140_6kg_0.81m

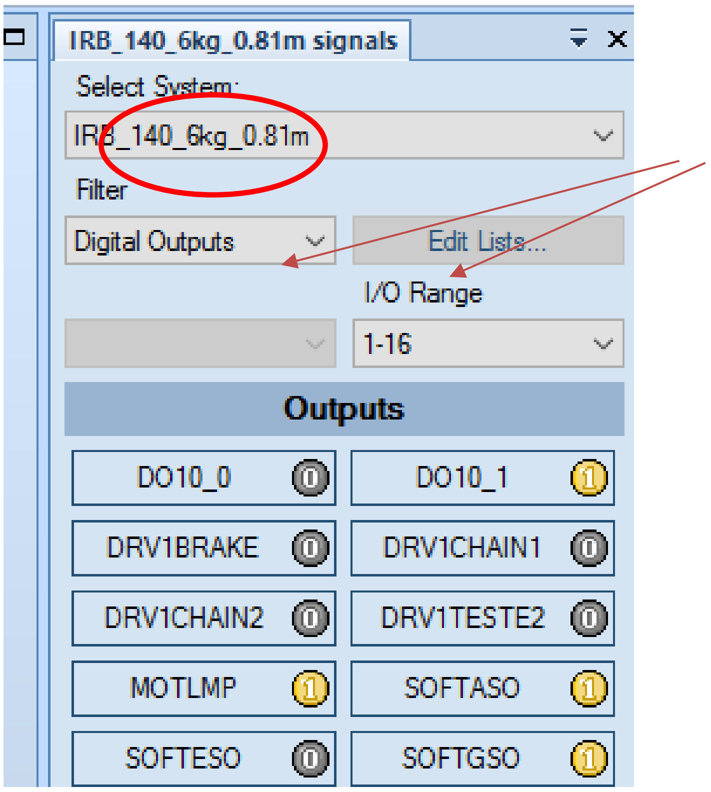
Filter: Digital Outputs

Edit Lists...

I/O Range: 1-16

Outputs

DO10_1	0	DO10_2	
DRV1BRAKE	0	DRV1CHAIN1	
DRV1CHAIN2	0	DRV1TESTE2	
MOTLMP	1	SOFTASO	
SOFTESO	0	SOFTGSO	
SOFTSSO	1	STLEDGR	
STLEDGRBNK	0	STLEDRED	
STLEDREDBNK	0	TESTEN1	



IRB_140_6kg_0.81m signals

Select System: IRB_140_6kg_0.81m

Filter: Digital Outputs

Edit Lists...

I/O Range: 1-16

Outputs

DO10_0	0	DO10_1	1
DRV1BRAKE	0	DRV1CHAIN1	0
DRV1CHAIN2	0	DRV1TESTE2	0
MOTLMP	1	SOFTASO	1
SOFTESO	0	SOFTGSO	1

Picture 17: Filter on Digital Outputs to see the signals

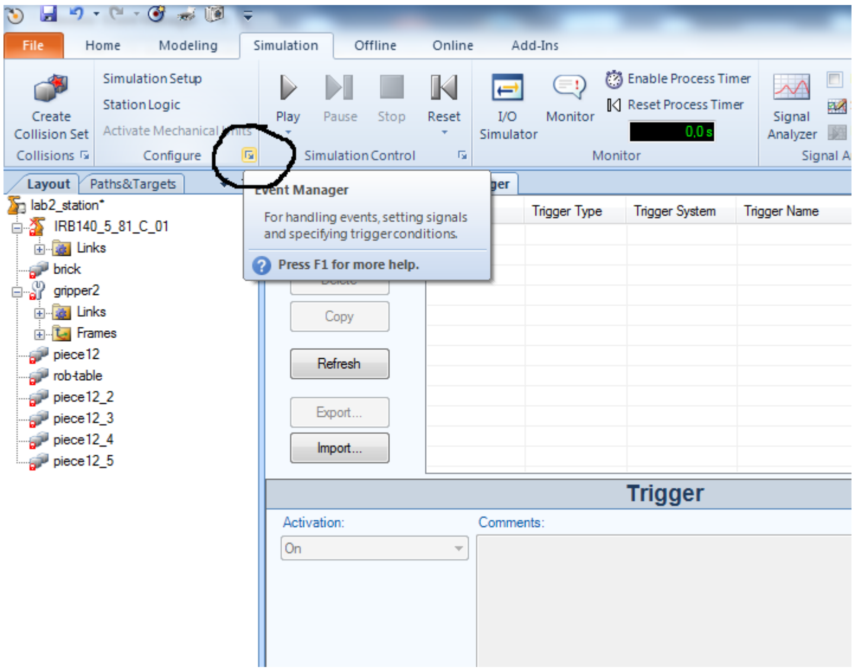
Observe the DO during the simulation. (Note that you should have a **WaitTime** both before and after the "Open"/"Close" gripper commands!) As alternative to writing the "SetDO"-commands in the code for the path, you can write separate RAPID-procedures "OpenGripper" / "CloseGripper", which embeds the necessary commands for this. This will make the program easier to read.

The workpiece "piece12" is still not moving and we will fix that now.

Even if this program would work on the real robot by opening and closing the real gripper, it does not SHOW here in RobotStudio (the gripper in no real mechanism which can move in the simulation, and the simulation environment does not know how to "pick up things" if we do not help it).

The next section is thus just for the program to show the desired behavior in RobotStudio and has nothing to do with the code we download to the robot in the end.

Click on the Event Manager in the Simulation tab (it is the tiny arrow next to the word "Configure", see Picture 18). "Add" and event, select "Activation" = On, "I/O signal changed" = DO10_1 with "Signal is true ("1") and Active Controller, "Action type" = <Attach Object>, and "Attach object:" "<Find closest object>" with "Keep position" and "Attach to:" "<gripper>", and Finish.



Picture 18: The location of the Event Manager

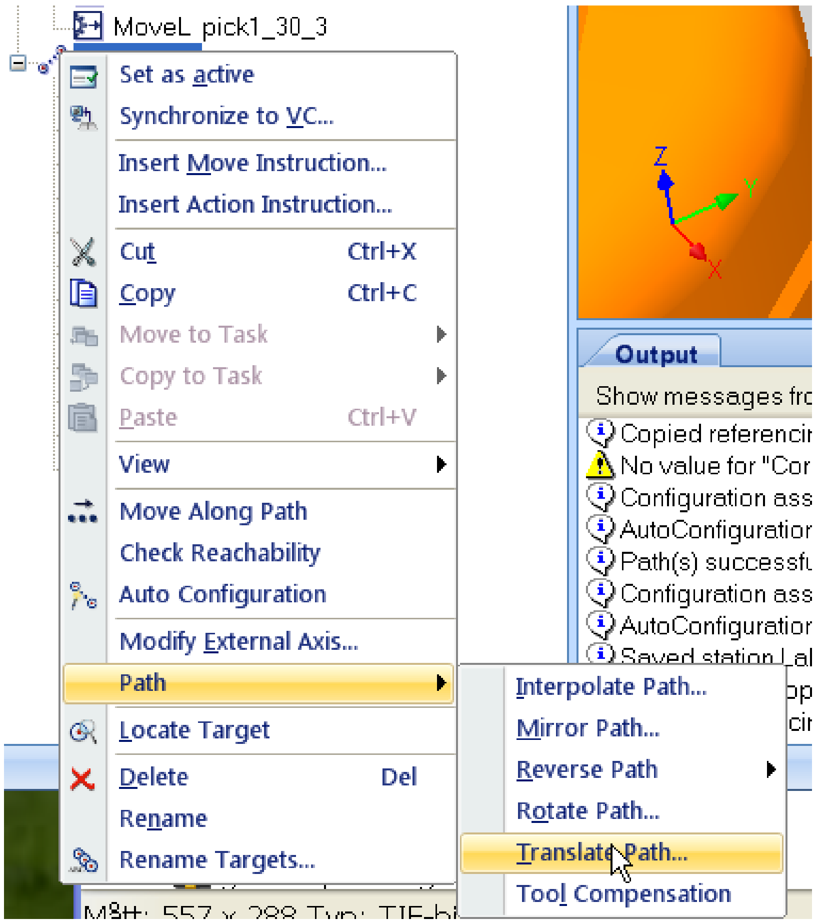
Add another event and select signal trigger DO10_0 but use "<Detach object>" instead (<Any object> from the gripper).

Try to run the simulation now and if correct, the piece12 should move to the new location.

Save your station.

We can now copy the path and duplicate it in the following way: Select the path, right-click and select "Copy". Click on "Paths & Procedures" and right-click. Select "Paste". Answer "Yes" to the question

"Should New Targets be created". The path is now at the same location as the old path. Click on the new path name you just made, right-click, select "Path" and "Translate Path".



Use "Reference Frame" a "Selected Frame" and mark the workobject at the corner of the brick workpiece (You may mark the frame in the graphics window or e.g., mark a workobject in list of the layout-tab, whichever you find is easier). Then the translation of the path is easy to make (50 mm in the x direction for each hole).

Note 1: for the first copy path, delete the first target (home position target) and change the next Move instruction to a MoveL and the velocity to 100 mm/s. Check the path with Auto Configuration.

Note 2: Add the target to the home position to the end of the last path. Use a MoveJ and velocity v1000.

The system might ask you what solution to use. In most cases, the first one is fine, but you can click through the list and see the corresponding joint angles of those configurations, see below.

(The numbers in Cfg (a,b,c,d) represents a configuration where

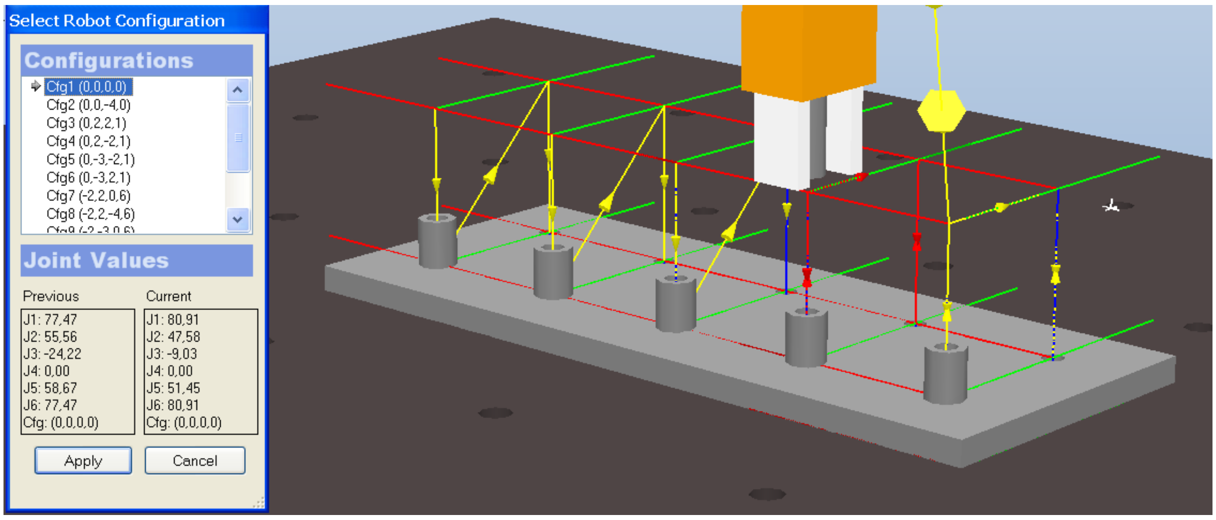
Joint 1 is in the a$^{th\ quadrant}$ (a=0 means an angle in [0,90] deg.)

Joint 4 is in the bth quadrant,

Joint 6 is in the cth quadrant

An external axis is in the dth quadrant

a = 1 means an angle in [90, 180] degrees
a = -1 means an angle in [0, -90] degrees



Select Robot Configuration

Configurations

- Cfg1 (0,0,0,0)
- Cfg2 (0,0,-4,0)
- Cfg3 (0,2,2,1)
- Cfg4 (0,2,-2,1)
- Cfg5 (0,-3,-2,1)
- Cfg6 (0,-3,2,1)
- Cfg7 (-2,2,0,6)
- Cfg8 (-2,2,-4,6)
- Cfg9 (-2,-3,0,6)

Joint Values

Previous	Current
J1: 77,47	J1: 80,91
J2: 55,56	J2: 47,58
J3: -24,22	J3: -9,03
J4: 0,00	J4: 0,00
J5: 58,67	J5: 51,45
J6: 77,47	J6: 80,91
Cfg: (0,0,0,0)	Cfg: (0,0,0,0)

Make Auto Configuration on all paths, synchronize to Station, and add the paths to the main procedure using the Simulation Setup.

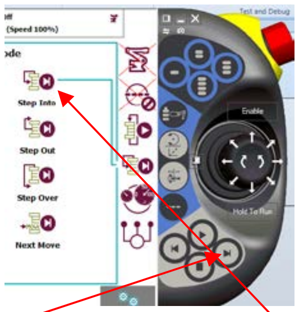
Test and run the simulation and make any necessary changes. Finally, save the station and remember to save the program to an USB-stick (right-click T-ROB1 -> “Save Program As…”.

Show the simulation result to the teacher or teacher’s assistant. Run the simulation on the real robot in the lab together with the teacher.

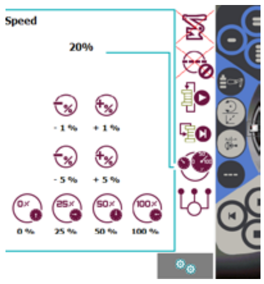
When testing your program on the real robot, it is very important that there are no collisions as this may damage both the robot and the gripper. Therefore, it is good practice to try your program in "Manual mode" where you quickly can stop the robot execution if you suspect/see a collision.

In the "Cog wheel-menu", (see Picture 19) you can reduce the programmed speed during these first tests and you should also run your program instructions "step-by-step" (step into procedure calls), which makes it easier to evaluate the correctness of programmed target points and that grippers open/close at the right occasions, which else may cause collisions.

Save your file on the C:\-drive and afterwards save your station as "Pack&Go" to your home folder (see RobotStudio-exercise 1 for details of the settings).



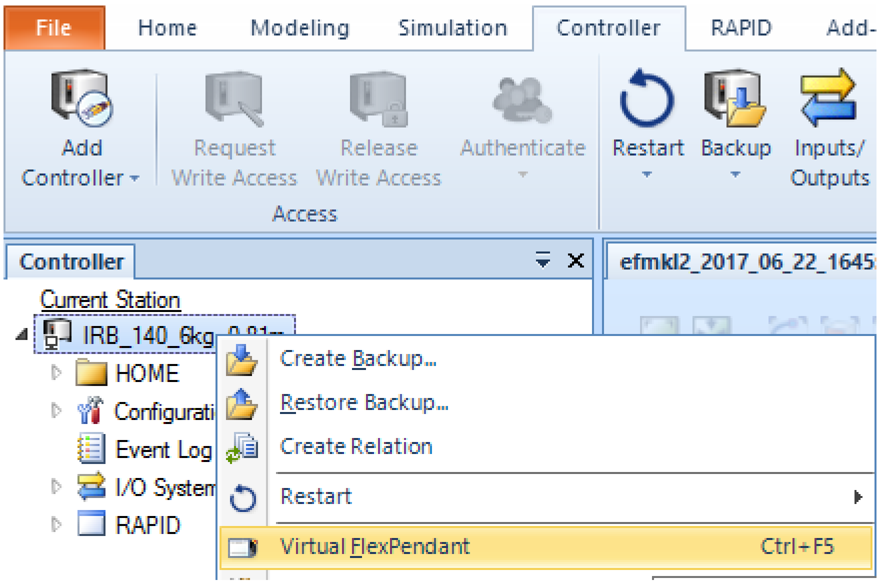
*Picture 19: Button to execute program commands step-by-step (Mode: move **into** procedures)*



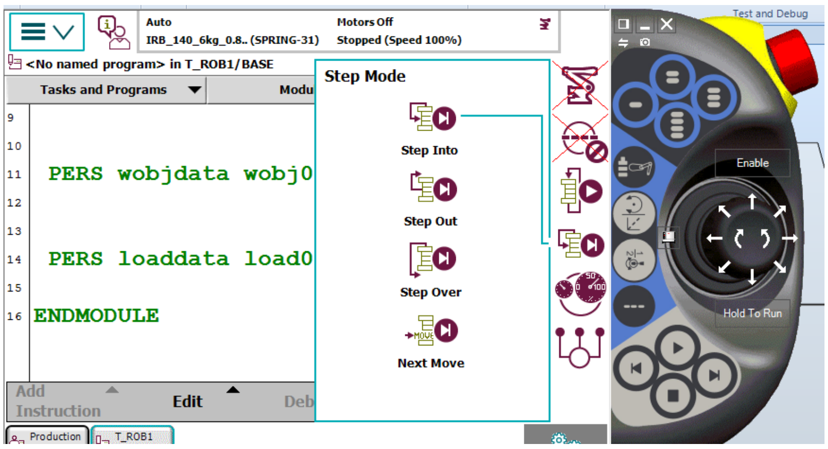
Picture 20: Menu for modifying speed during execution.

Note: You can start a Virtual FlexPendant also in RobotStudio, which looks similar and has the same menus as the real.

Controller -> Controller Tools -> FlexPendant -> Virtual FlexPendant or as indicated in Picture 21



Picture 21: Another way to open the Virtual FlexPendant



```
PERS wobjdata wobj0

PERS loaddata load0

ENDMODULE
```

Picture 22: How the Virtual FlexPendant will look

3. Reflection

Think about the following:

- Interconnection between simulation and programming
- Some issues need to be taken care of only in simulation but not in programming
- Some issues need to be taken care of only in programming but not in simulation
- Knowledge about the physical system and relate and map data and signals in the right way
- In exercises 1 and 2, we have had CAD-models and pre-defined tools, which automatically come with their tool definition (pen and gripper, respectively). If you mount a new tool on your robot for which you do not have an accurate drawing, how would you define the tooldata (doing a tool calibration)?
- The combination of selection of different arguments and parameters for instruction and wizards are almost infinite
- A tool as RobotStudio is rather flexible and a challenge to master for a general case / task. However, for most users, every task is usually similar which makes such tools efficient to use.