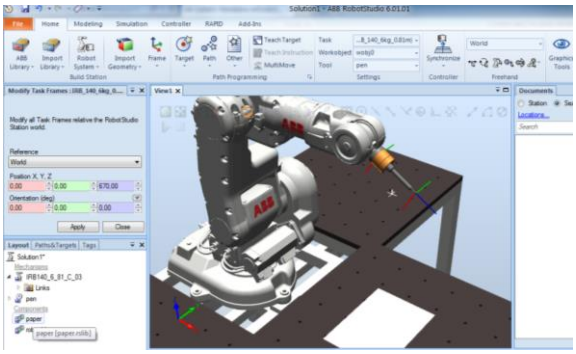
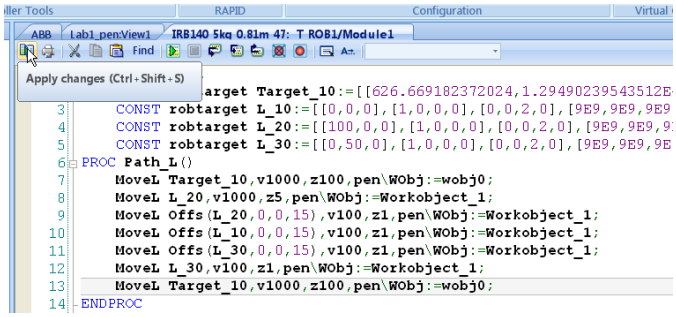
Exercise 1, Robot Studio, using a pen tool (MMKF15 Applied Robotics), LTH, Lund University

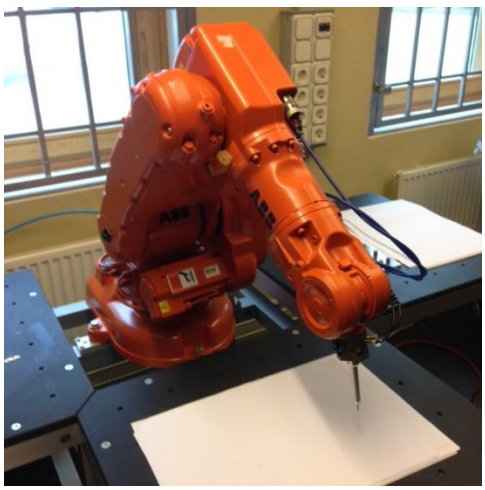


1 ***Modeling and simulation*** *in program RobotStudio*



2 ***RAPID-program*** *auto-generated from your specified target-points and paths in RobotStudio*

(or handwritten in text-editor or programmed at Flexpendant): instructions for robot motion



3 ***Download and test your RAPID-program*** *on a real robot (ABB IRB140)*

Exercise 1, Robot Studio, using a pen tool

For RobotStudio version 6.09

Gunnar Bolmsjö, [rev Sept 2019 ARo\ERo]

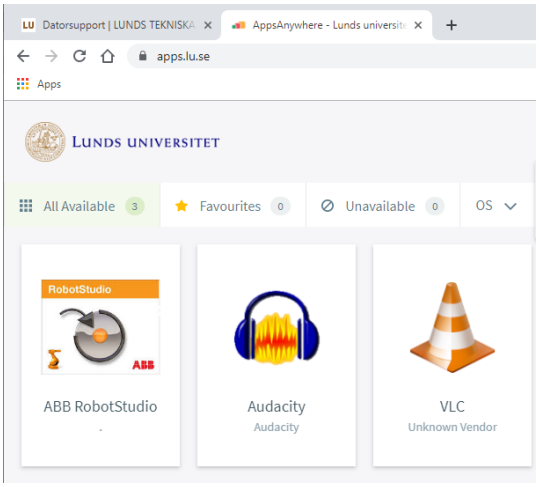
The aim of this exercise is to familiarize yourself with **RobotStudio** and the general workflow to create a **workstation**, define **actions** such as move instructions, make a **simulation** and **edit a robot program** (RAPID code). You will finally upload and run the generated robot code (the RAPID program) on the **target robot** (either via an USB-pendrive or directly via a network-connection to the Robot controller).

The instruction for this exercise is very detailed. The instructions for exercises 2 and 3 will be less detailed as you get more used to the software.

1. Launch RobotStudio

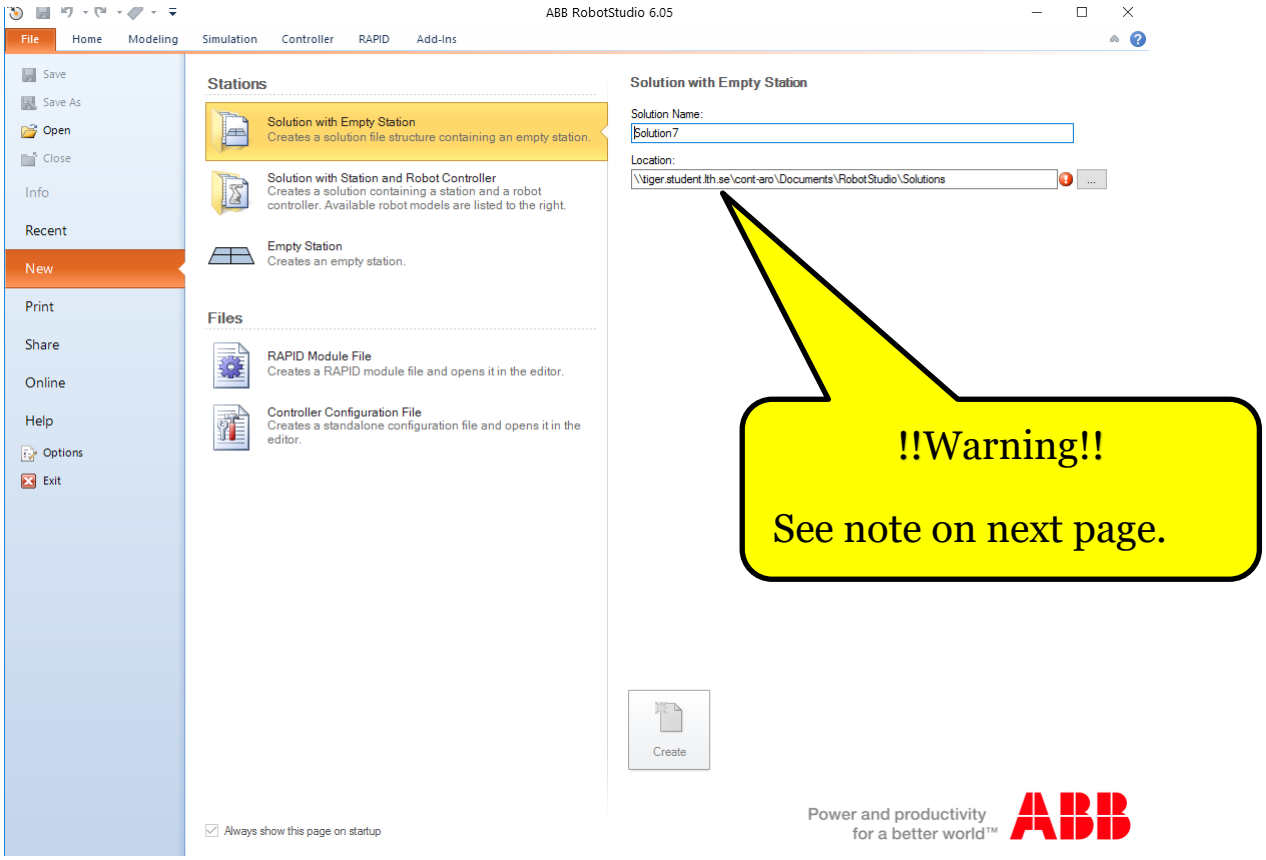
Before launching RobotStudio, please **create the folders**

- C:\temp\Documents*myUserName*\RobotStudio
- C:\temp\Documents*myUserName*\RobotStudio\Solutions



Picture 1. Start RobotStudio2019 from a Google-chrome browser at https://apps.lu.se/ (use your STiL-account)

After start, you will then get a screen like in Picture 2. Note that it will take a while as the software also starts and runs a **virtual controller** in the background.



Picture 2: Startup screen of RobotStudio 6.05 (64-bit)

NOTE [2019-09-13]: RobotStudio does unfortunately **NOT** support the networked-mounted folders correctly (in particular not UNC paths, starting with \\tiger, \\puma etc.), so we **strongly suggest** that you instead of using your network-mapped home-drive (H:\), run your RobotStudio-exercises in a folder on the local computer (C:\) until this issue is solved. You should therefore **NOT** only save your station, but also make a complete **Pack&Go-station** (.rspag-file), see Appendix. (A Pack&Go station not only saves links to components used in the simulation, but also actually makes a local copy and puts all components to a complete station in one folder, so that you after the exercise is finished can copy the Pack&Go-station to your home folder and easily transfer it between different computers without any path-problems.

[Release notes RobotStudio2019]

RobotStudio on computers with roaming user profiles

RobotStudio may fail on PC configurations with roaming user profiles, i.e., when the users' documents folder resides on a server and not on the local disk.

How to work around the problem:

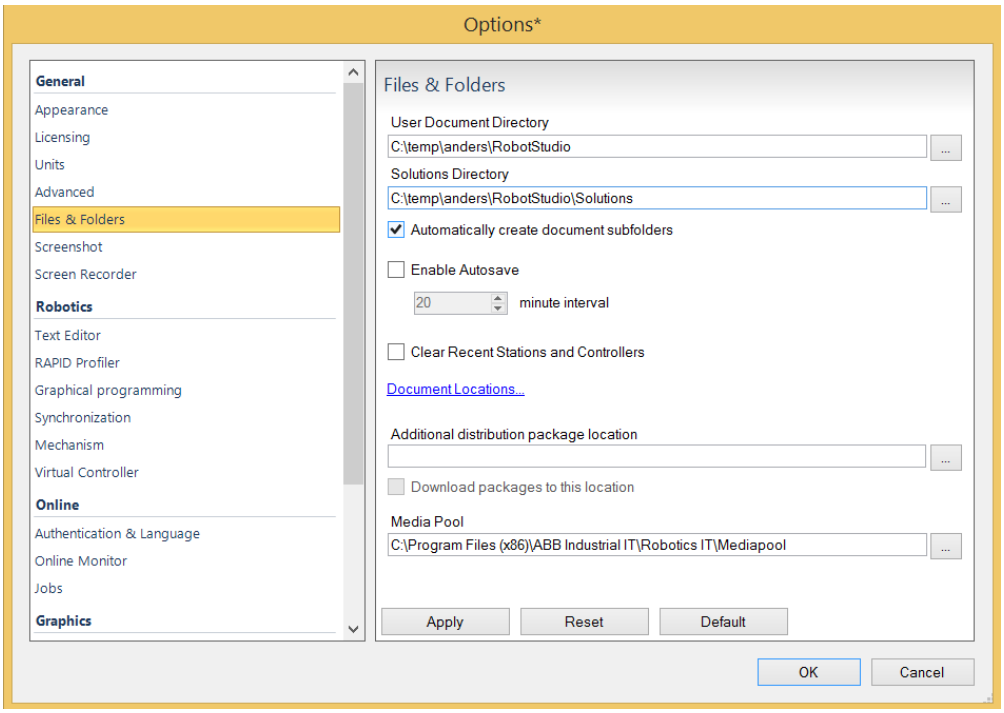
Redefine the "User Documents location" to a folder on the local disk (File → Options → General → Files & Folders → RobotStudio Documents).

Select browse and enter

C:\temp\Documents*myUserName*\RobotStudio

&

C:\temp\Documents*myUserName*\RobotStudio\Solutions for the User Document Directory and Solutions Directory respectively as indicated in Picture 3.



Picture 3: Set working folder to local disk to avoid problem while running RobotStudio.

Remember to copy saved **Pack&Go stations** to your home folder after the exercise!

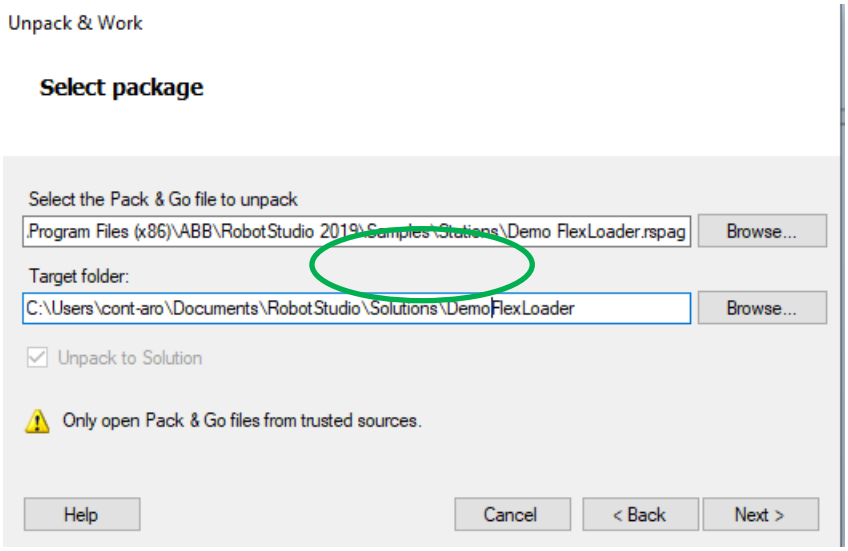
Try out some premade stations

Next, click Open in RobotStudio's File screen and browse to

C:\Program Files (x86)\ABB\RobotStudio 2019\Samples\Stations
where all the sample stations are located in the computer lab. The sample stations are **rspag files**, so make sure that you can see those files by selecting **Pack&Go files** above the "Open" and "Cancel" buttons.

Open any of these. We will describe the "FlexLoader" station.

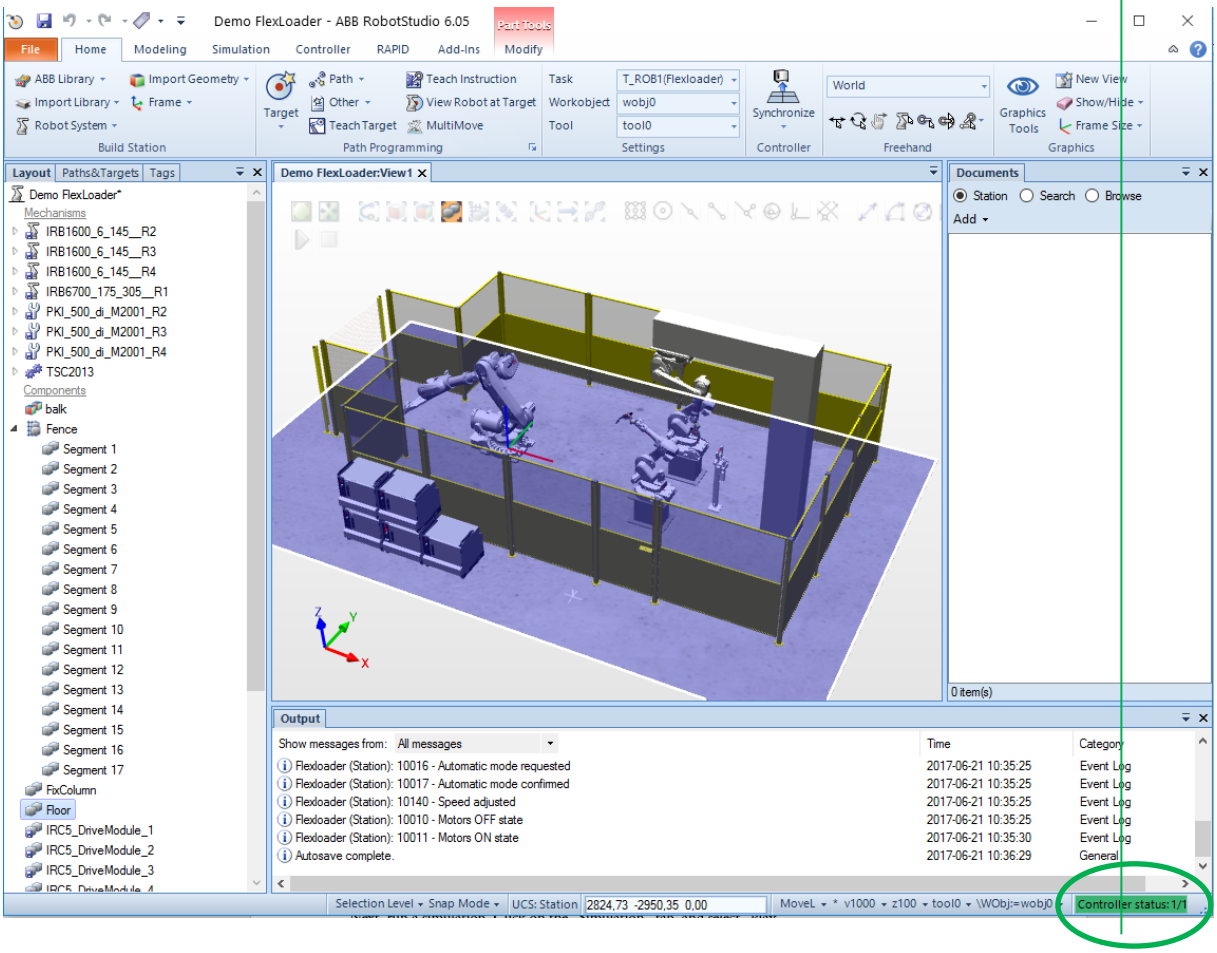
Click "Next" on the wizard that pops up. Type in the address **and a new folder** in the "Target Folder" where the files will be unpacked. (Note: Use your temporary storage folders to avoid the network-mapping problem), see below in **Error! Reference source not found.**.



Picture 4: Make sure to create a new folder (you may get an error if unpacking in non-empty folder).

Click “Next” and “Finish” to execute the wizard. The station will look like the following after a while. Have patience while the startup bar in the lower right corner changes from

Red-->Yellow--> Green!



Picture 5: Look in the right-down corner so that Controller status 1/1 turns green.

Try out the **mouse functions** as follows:

(**LM** = Left Mouse button, **MM** = Middle Mouse button, MB = Middle Mouse button pressed down, **RM** = Right Mouse button)

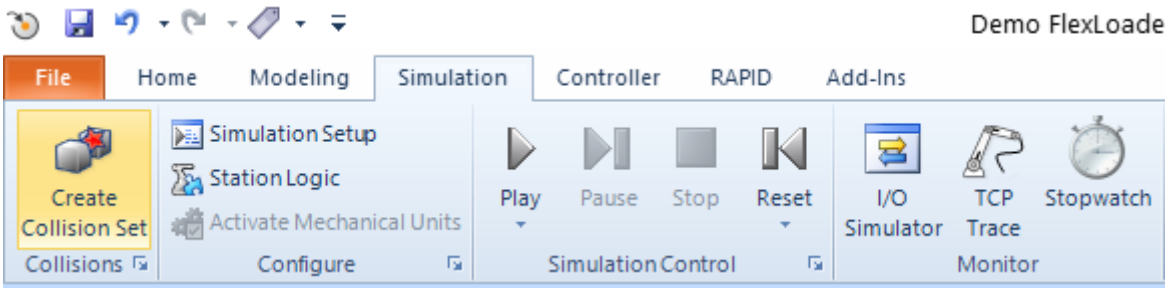
Pan: Hold down Ctrl + **LM** and drag.

Zoom: Use the scrolling function of the **MM**, or use **MB** and drag horizontal, or Shift + **RM** and drag a rectangle.

Note: you can right click anywhere and use “View All” (zoom out), “View Center” or predefined views as needed.

Rotate the scene: Use Ctrl + Shift + **LM**, **MM** + **RM** or **MM** + **LM** and move the mouse

Next, **run a simulation**. Click on the **Simulation-tab**, and select **Play**.



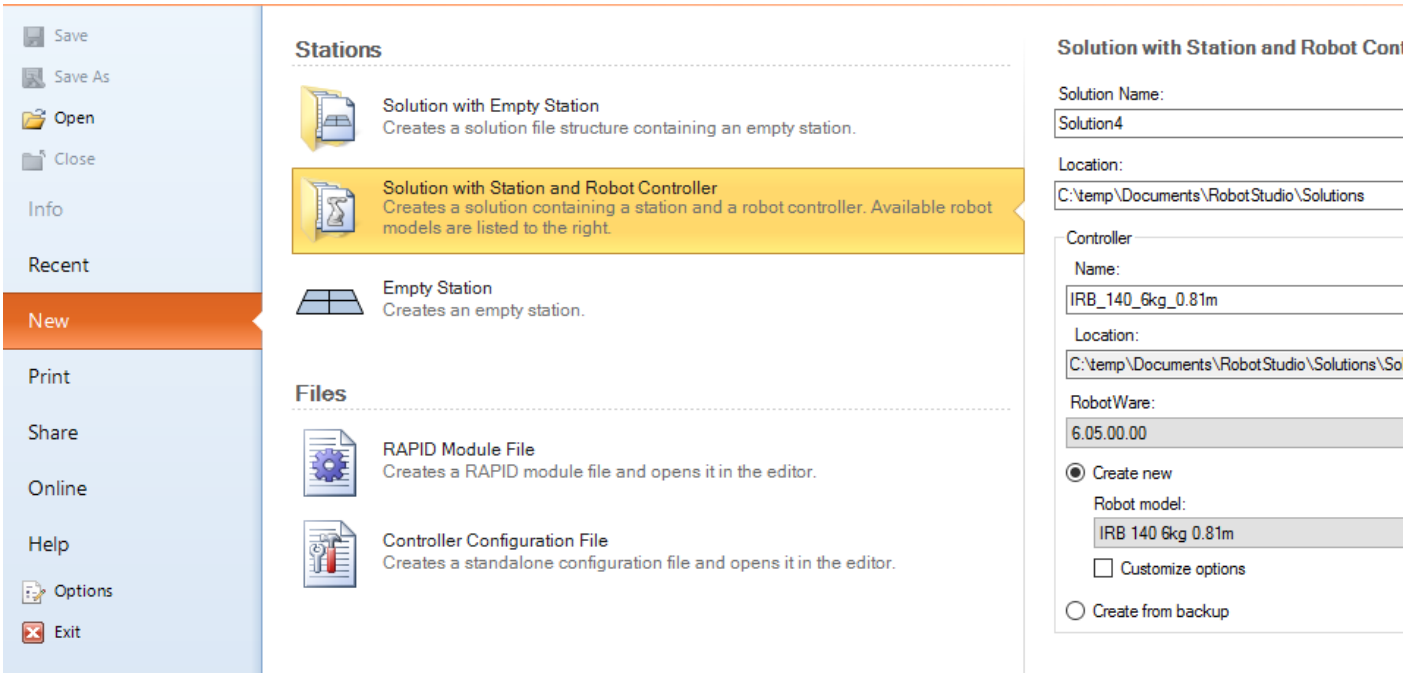
Picture 6: How the Simulation-tab looks in RobotStudio.

During the simulation, you can use the rotate, zoom and pan commands to get a better view of the simulation.

2. Create a pen drawing station

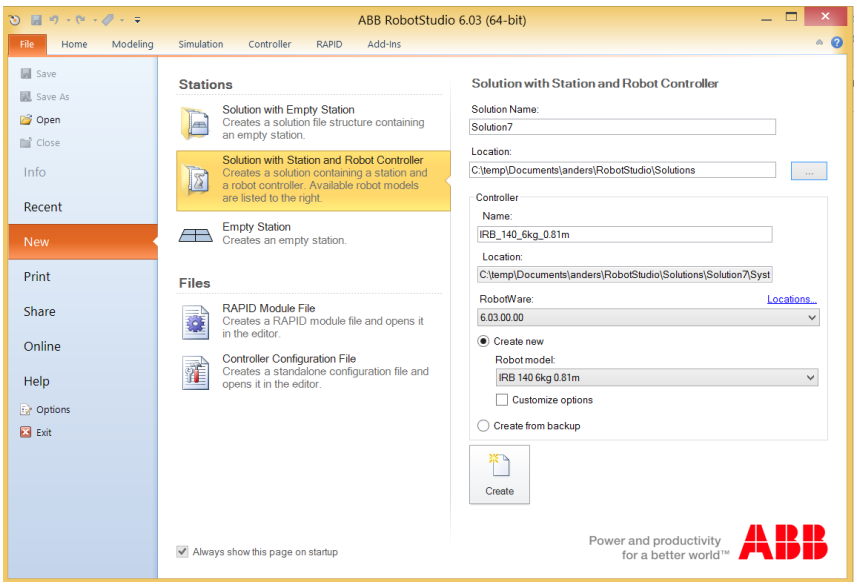
Now, you are going to create a new station similar to the robot station used at the hands-on lab. We will use a model of the ABB 140 robot and build a station with the table and a pen tool.

In the file tab, select New and then select Solution with Station and Robot Controller. Change the robot model to the IRB140 6kg type C robot, as in the picture below. Type in the address of your created folder in the location field and click "Create". If you are asked whether to save the station, you are currently working on or not, choose to discard the changes.



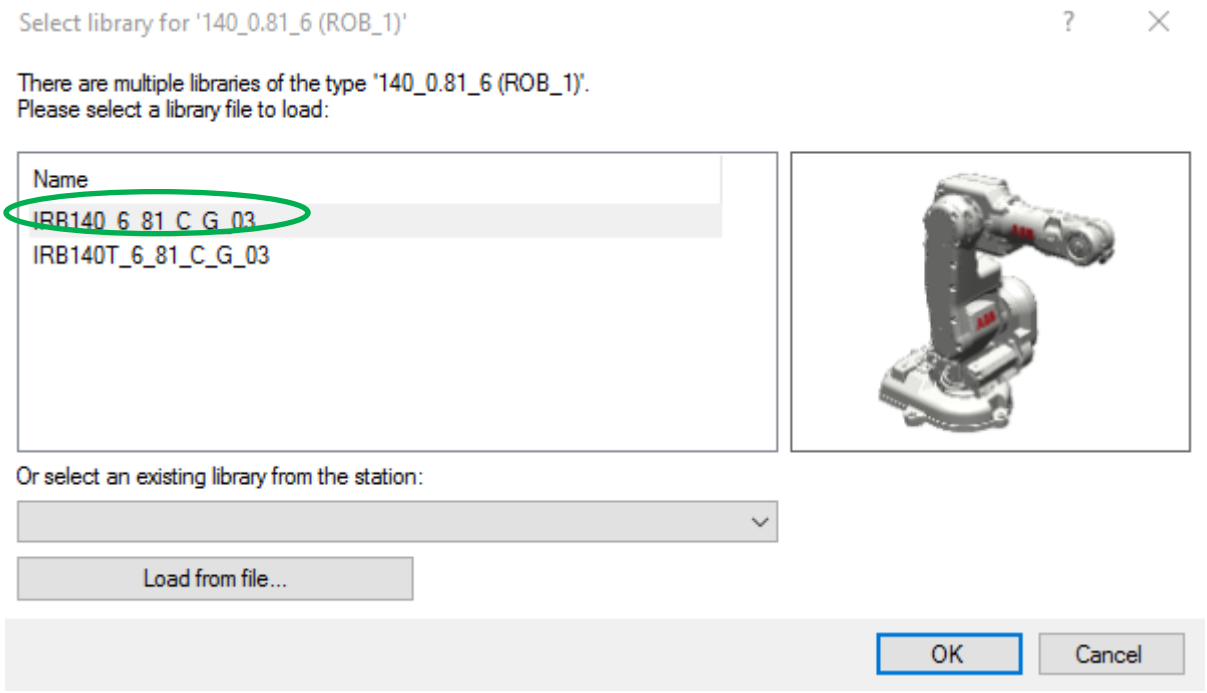
Picture 7: Under New, select Solution with Station and Robot Controller. Choose the correct Robot Model without any "T"!

(If a red exclamation mark shows up to the right of your “Location”, it is often due to that UNC-folders are not supported, please correct the folder address according to instructions above in Picture 3).



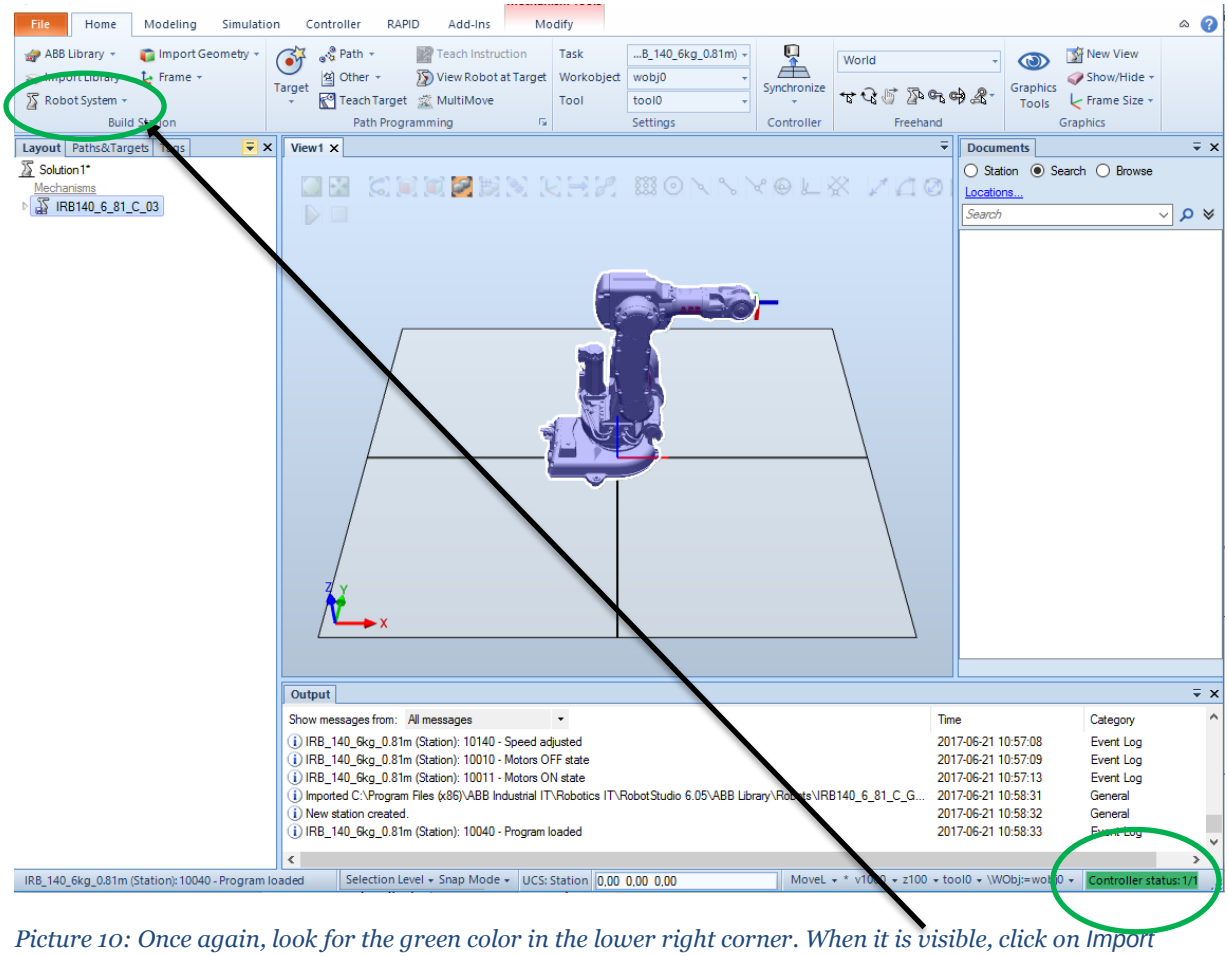
Picture 8: How the information should look before you create your “Solution”

Choose the floor mounted IRB140-robot version (without T in the name).



Picture 9: Be SURE not to choose the version with a T in the name

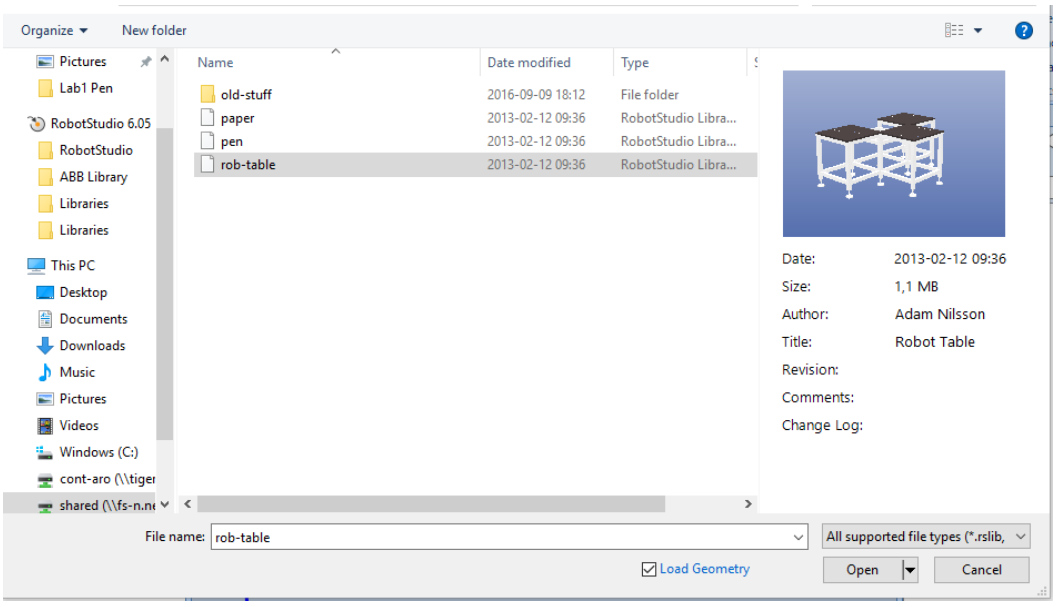
It will take some time to start the system and the rectangle in the right lower corner of RobotStudio will shift from red, via yellow to finally green when everything is finished and the robot controller has started. A robot will appear after a while as shown in the figure below.



Picture 10: Once again, look for the green color in the lower right corner. When it is visible, click on Import Library-> Browse for Library and find the rob-table among the files.

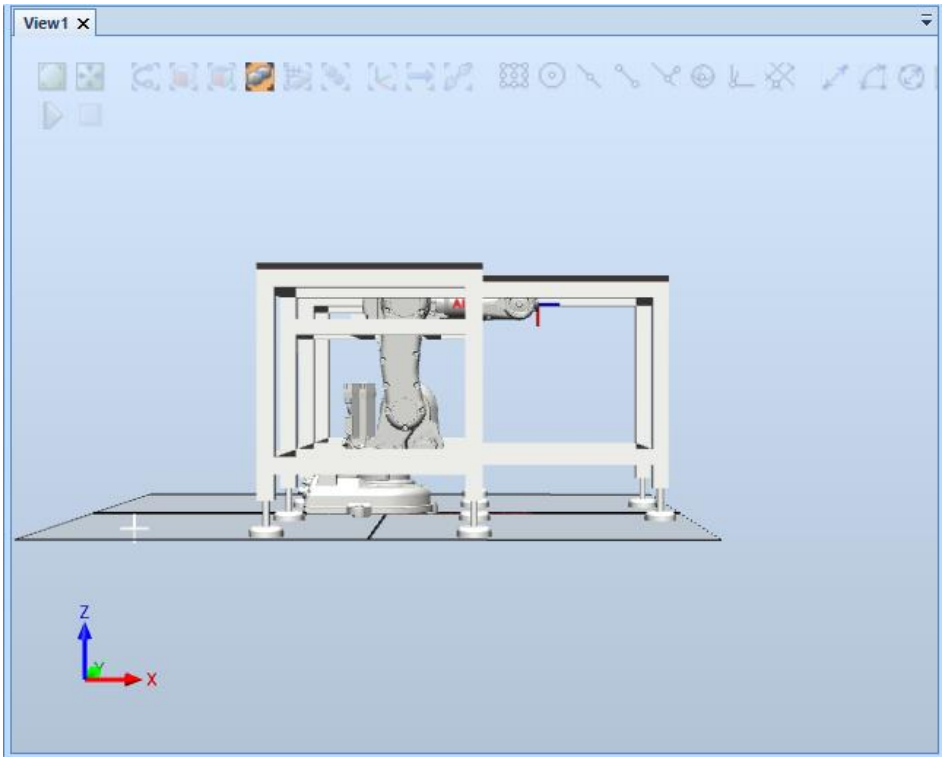
Import the "**rob-table**" from the library (**Home tab**): **Import Library -> Browse for Library**

The address: This PC->shared->Courses\control\FRTF20_Applied_Robotics\RobotStudio_exercises_2019\Lab1 Pen



Picture 11: The picture shows how your "Open" window should look. Select and open "rob-table"!

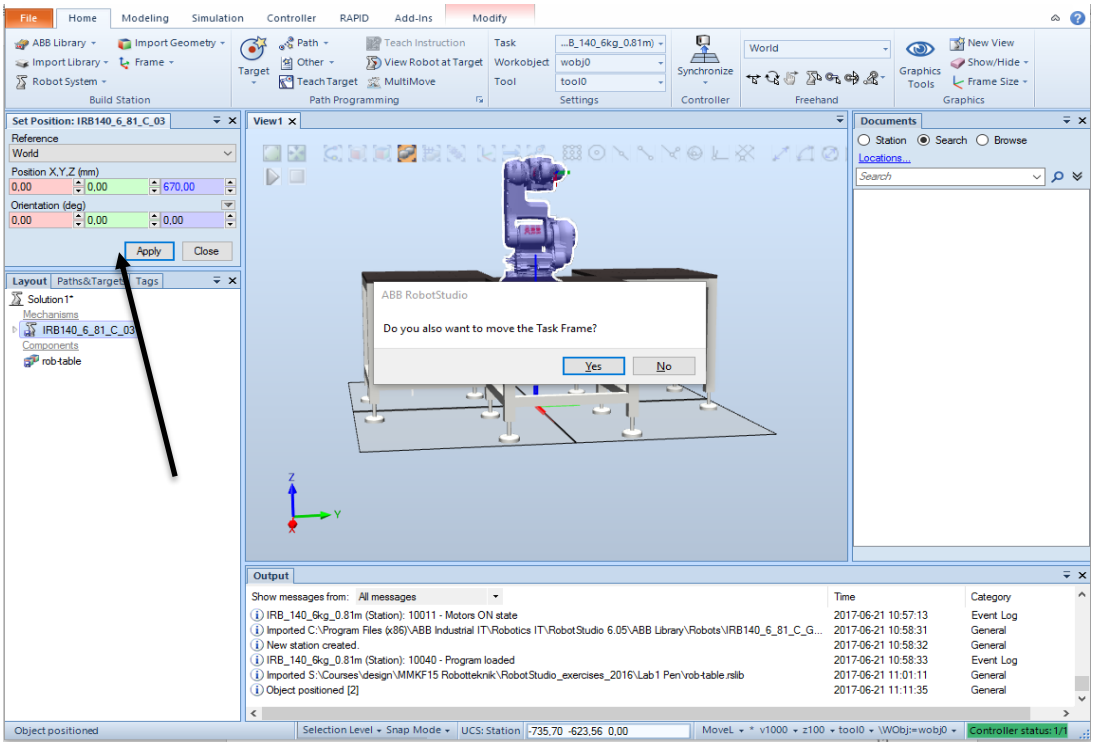
Click on the "rob-table" and then on "Open".



Picture 12: How the window will look after the "rod-table" has been imported

The robot and the table are both on the floor level. To place the robot on the table, we need to change the **Task coordinates**. Click on the **Home-tab**, right-click the robot icon, which will turn the robot blue to indicate that the robot is selected (in the layout section on the left side, called "**IRB140_6_81_C_03**") and choose "**Set Position**". A window pops up where you can set the frame data.

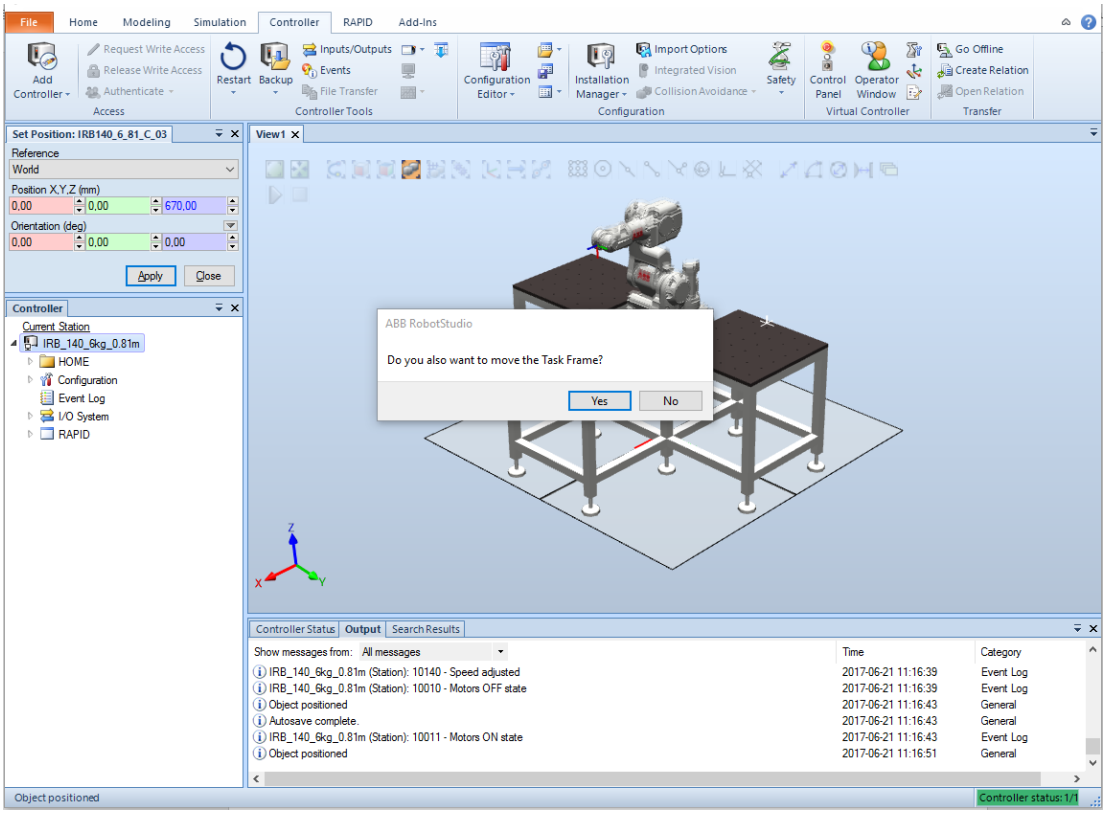
Click in the Position Z column and select 670 (millimeters), which means that the robot will be lifted 670 mm and placed on the table. Leave all other data and click on Apply. Confirm that you want to move the "**Task frame**" in a pop-up after applying the **offset**.



Picture 13:: Select "Yes" to move the robot up upon the table

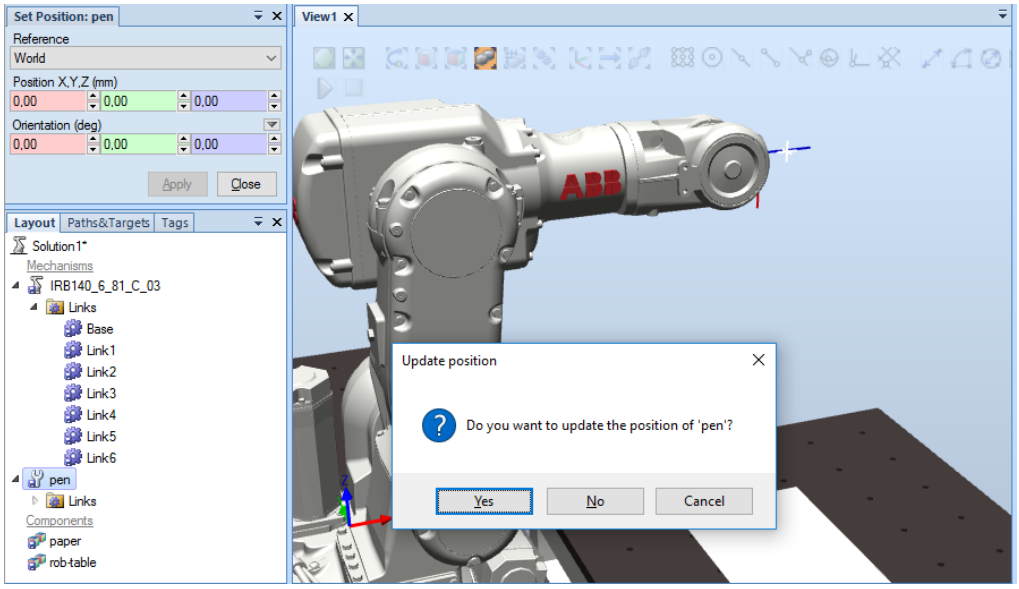
Note: You may achieve the same by choosing "**Task Frames**" in the Controller tab (see below in Picture 14).

Close the window. The robot station now looks like the following figure.

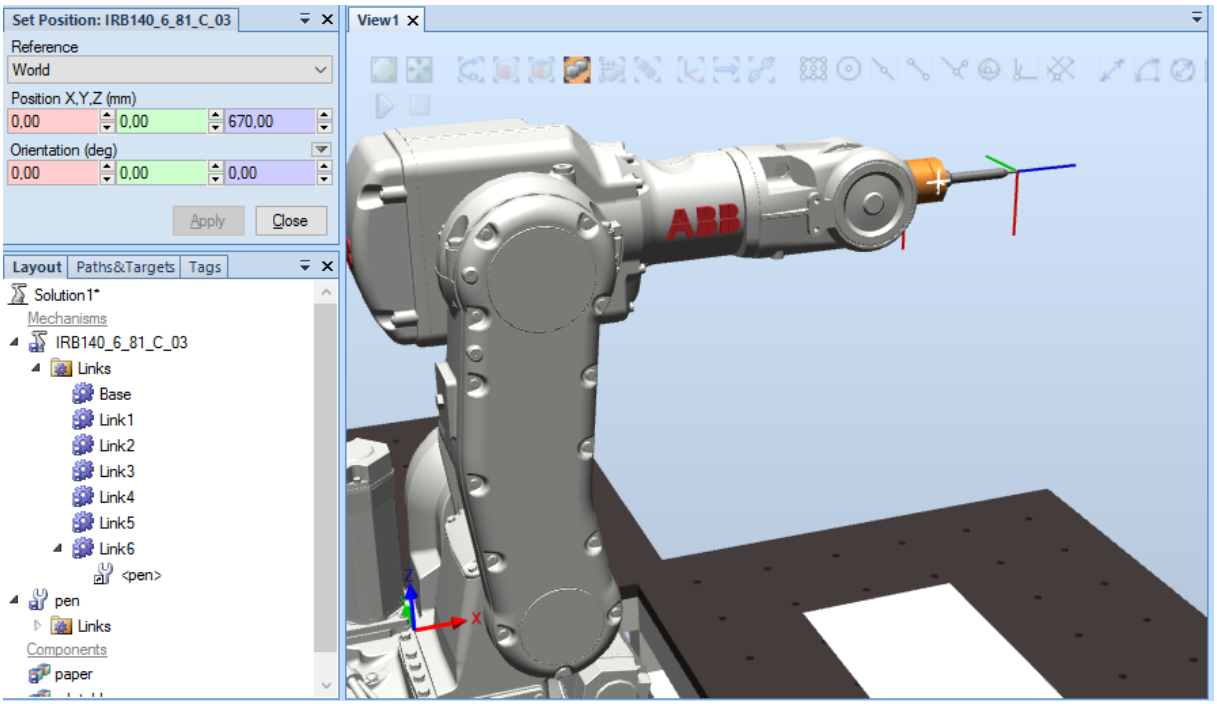


Picture 14: How it looks if you change the position from the Controller tab

Now, import through **Import Library** the "pen" and the "paper" from the same address as before. In the "**Layout**" view, you will now see the pen and the paper in the same way as you imported the rob-table. In the left layout tab, drag the pen tool to the robot (**IRB140_6_81_C_03**) and confirm that you want to move and attach the pen to the robot. The pen will now be attached to the 6th link of the robot with the TCP at the tip of the pen tool.



Picture 15: This pop-up will appear after you have dragged the pen to the robot

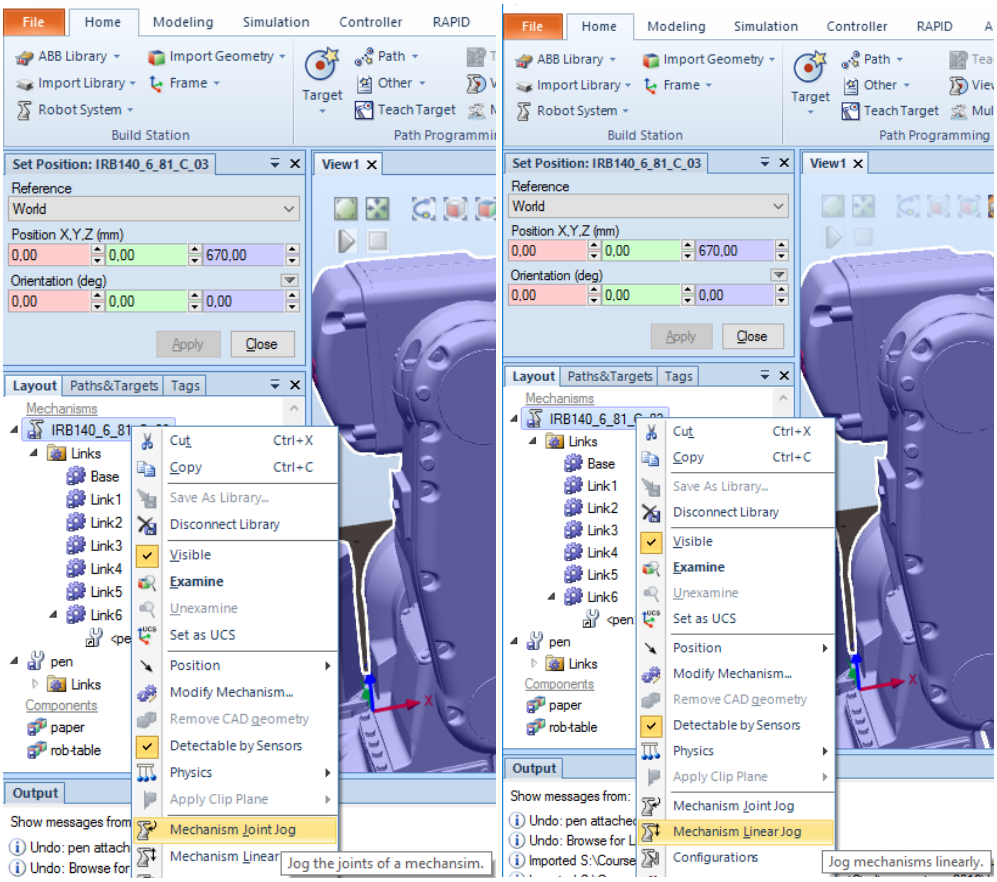


Picture 16: The pen has been successfully attached and the paper is lying on the table.

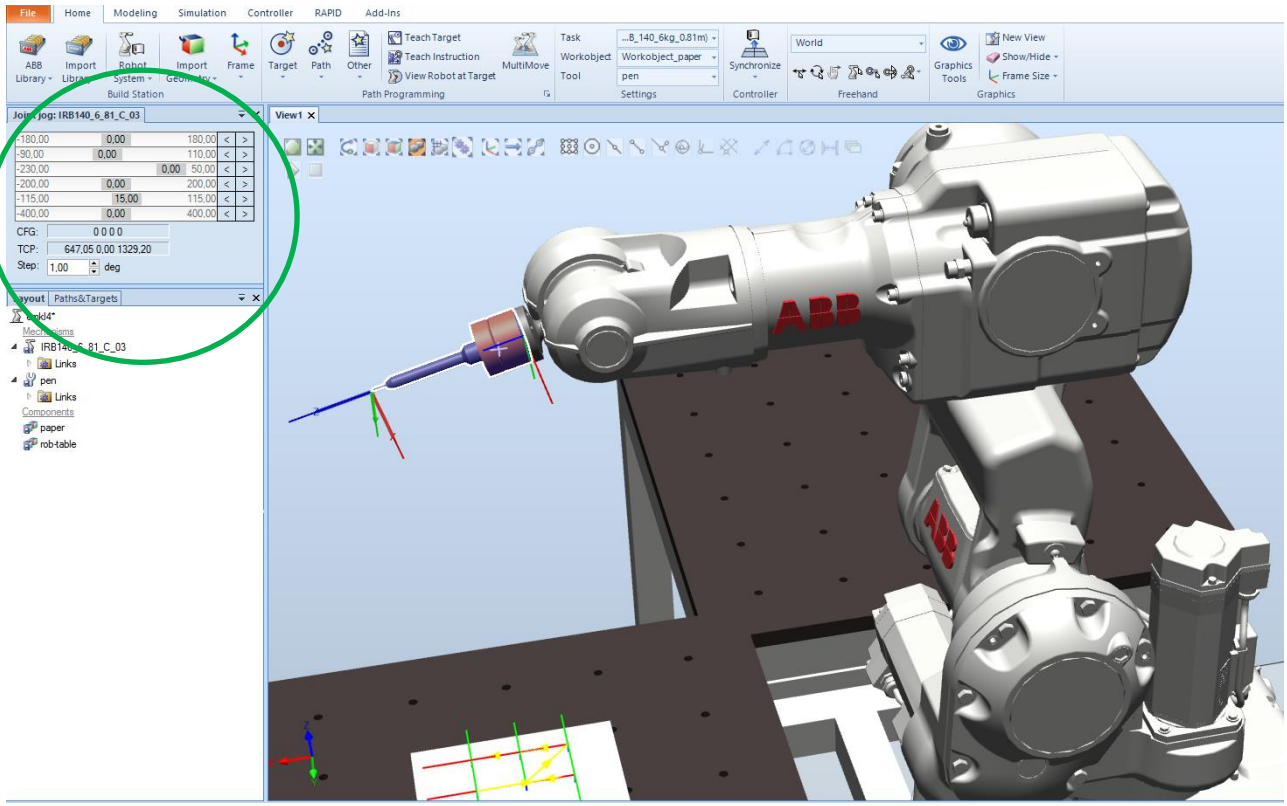
If you expand the tree in the "**Layout**" view of the robot you will see that the pen is attached to the last link of the robot.

Now, the all geometrical parts are included in the station and the next step will be to define some **target points**, which will be used to program the robot and **define paths**.

Moving the robot ("jogging") in the simulation:

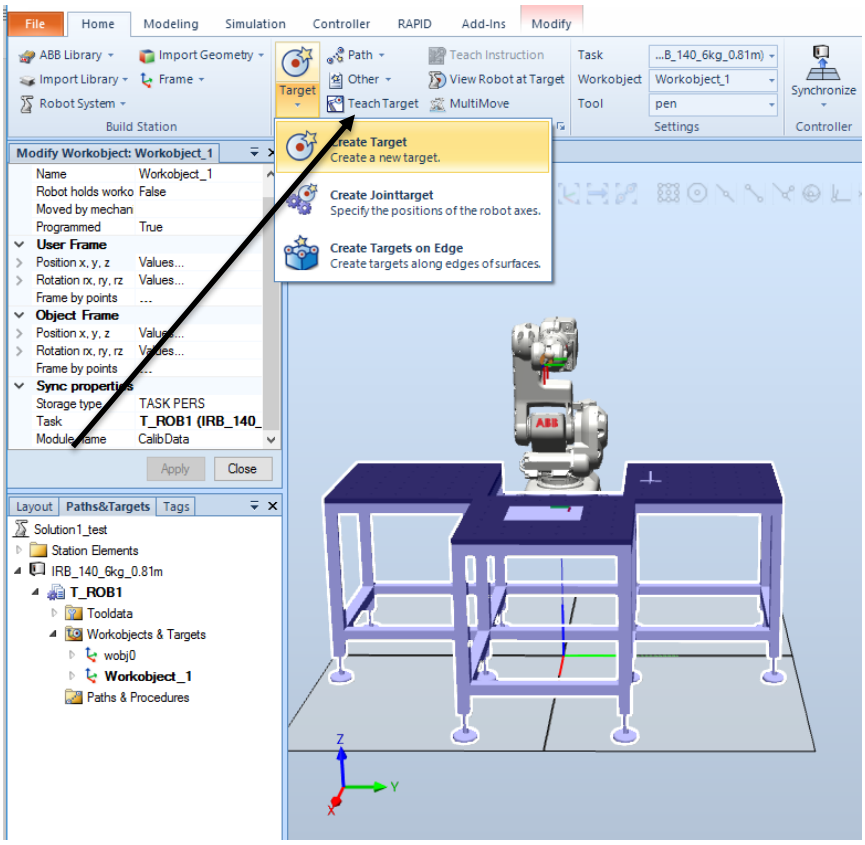


Note that the robot's so called ***HOME-position*** (where all joint values are 0) is actually a ***manipulator singularity*** (see lecture notes), so you can **NOT** move linearly (**MoveL**) to/from that point. First, use the ***Mechanism Joint Log*** to log the robot into the following position (see Picture 17).



Picture 17: Joint log the robot into the following position and compare your view to the picture

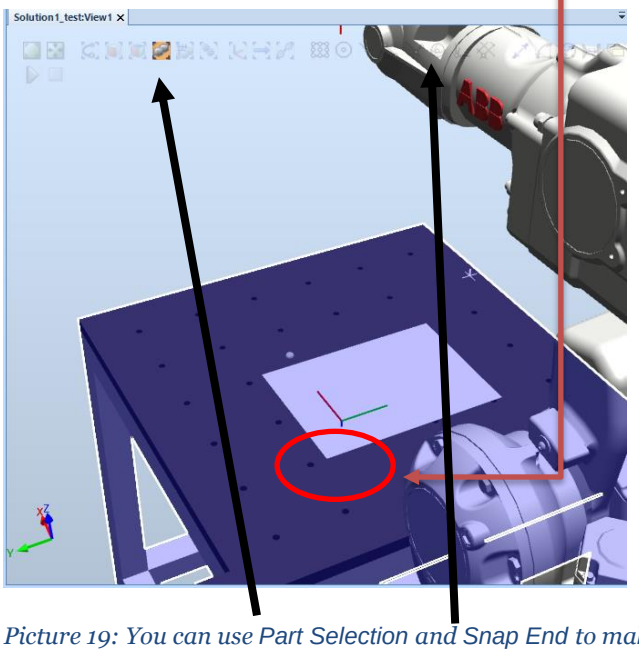
Then, in the **Home tab**, click on the **Teach Target-button** to create a target point where the robot is. (The name will nominally be "**Target_10**" if it is the first target, but the target should be renamed to a more informative name, for example "**home15**" as we set **joint_5** to 15).



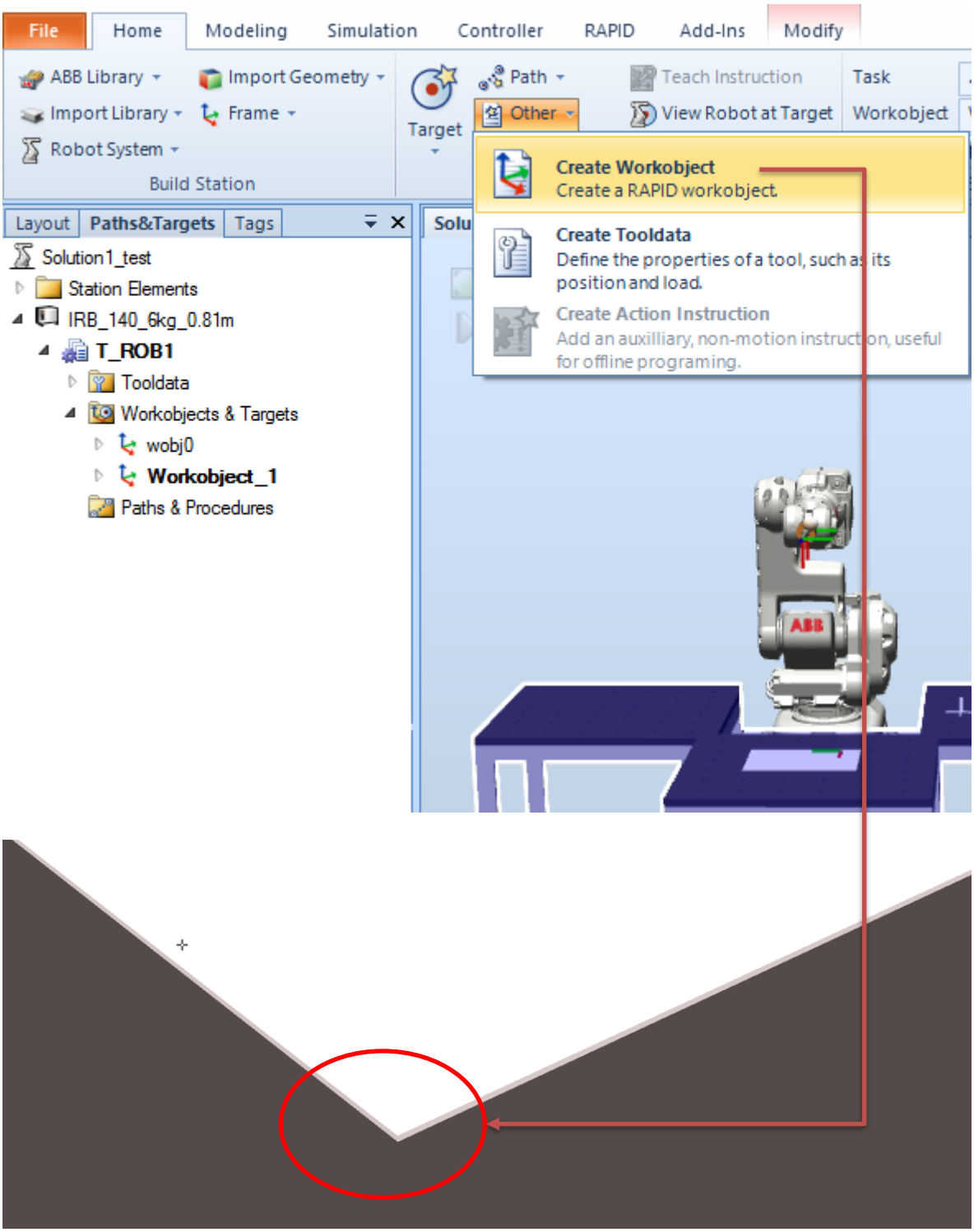
Picture 18: Press Teach Target to make a point at the tip of the pen

Note that we now work with the **pen tool** and the **workobject** is "**wobj0 (default)**". In other words, the target specified is **the point at the tip of the pen**.

Now, define the workobject as the **paper location**. Orient the view so you can click on the paper. In order for the instructions that follow should be accurate, make sure that you pick the same corner as shown in the pictures below (**lower left in the picture**)



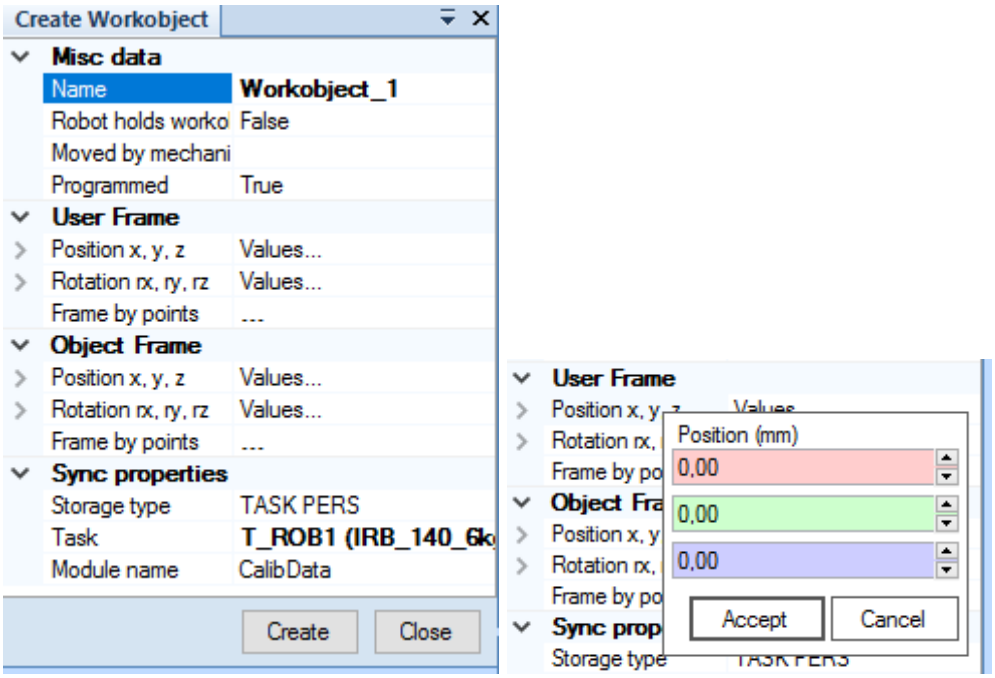
Picture 19: You can use Part Selection and Snap End to make it easier to snap the corner of the workobject



Picture 20: Make sure to pick the indicated corner when creating your "Workobject"!

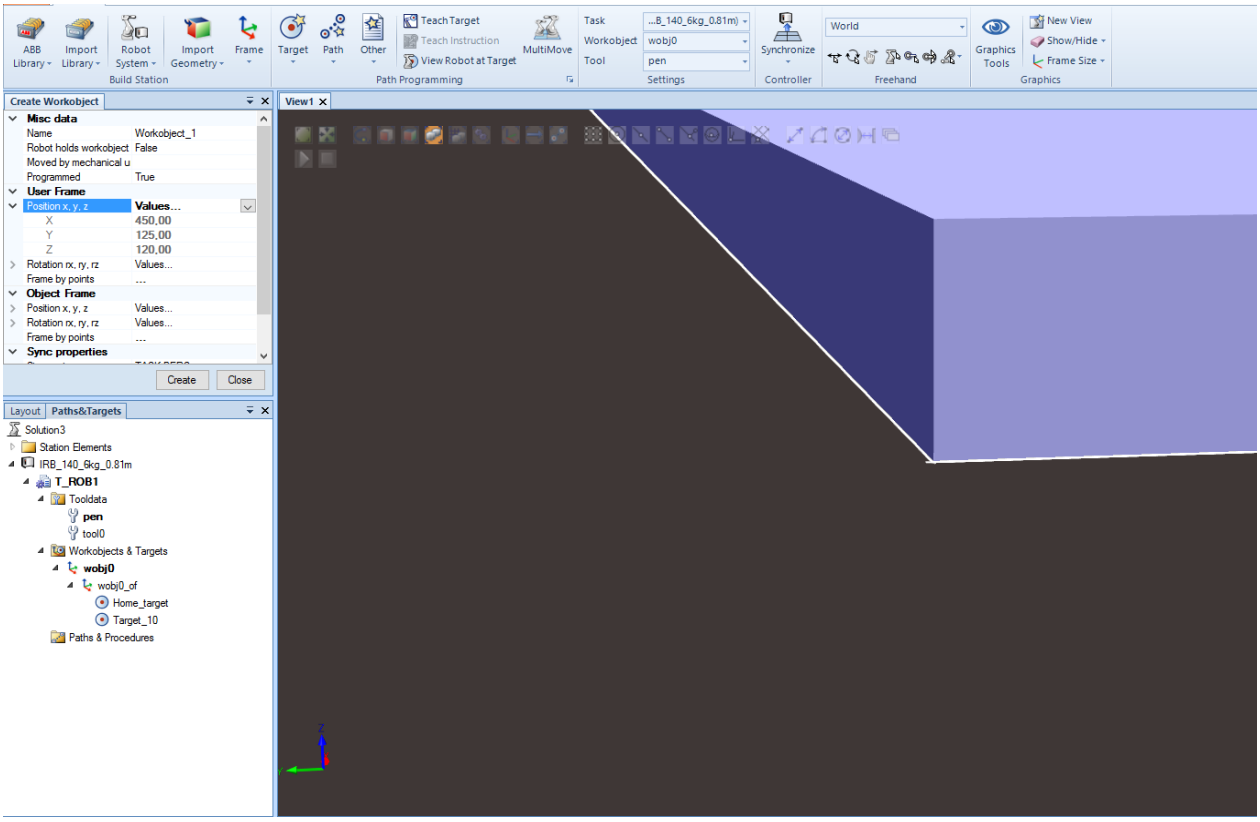
Define a workobject by clicking on the Other -> Create Workobject.

A window pops up to fill in some data for the new workobject. Click on the "Values" at the "User Frame", "Position x, y, z":



Picture 21: Click on "User Frame" and then in the red square before clicking on the corner of the white paper

Use the mouse buttons to zoom up the paper and click to indicate where to snap the **workobject frame**. Click on the read column for the x value, and then on the edge of the white paper. Make sure to point on the edge of the paper.

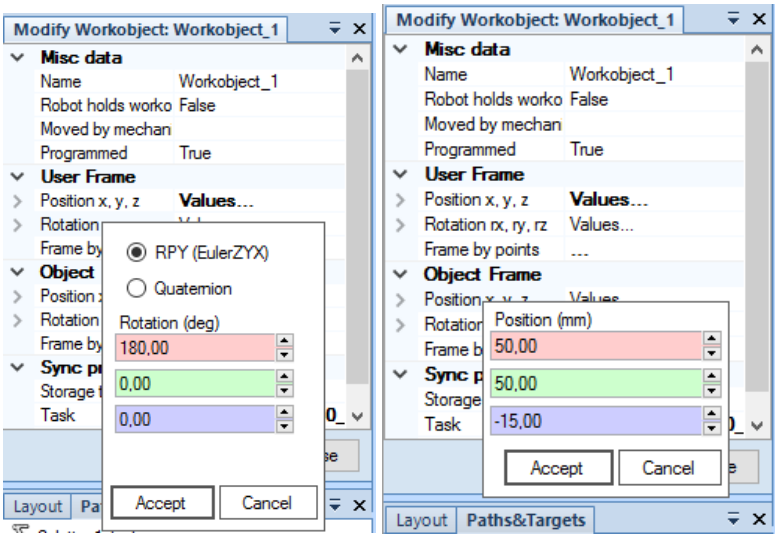


Picture 22: The data values should be as the following. If not, fill in the right values or redo the procedure

The data values should be as the indicated in the picture (450, 125, 120). If not, fill in the right values or do the procedure again.

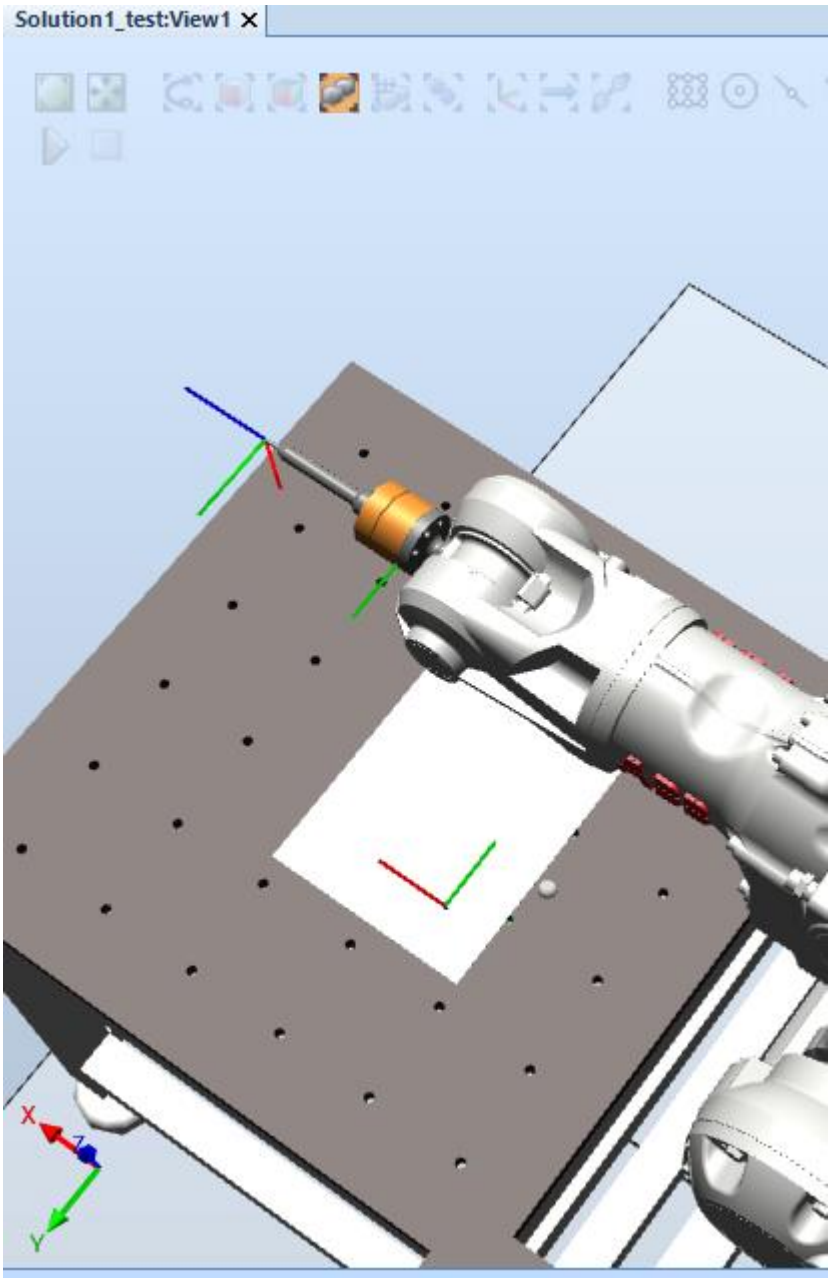
By default, the Z-axis is pointing up. The pen tool's Z-axis is pointing in the direction of the pen, which will be down when drawing. Therefore, the Z-axis of the "**User Frame**" should point down. To do this, make a rotation around the X-axis of the "**User Frame**" (rotation rx values, 180 degrees). The position of the "**Object Frame**" can now be set to 50 mm in X- and Y-axis, respectively, **and -15** mm in Z-axis (note minus-sign!), meaning a position above the paper area (and a bit above the table surface).

Note: The Z value of the "**User Frame**" (**120 mm**) is just on the surface level and will not work for drawing. Therefore, change the Z value of the "**User Frame**" to **116 mm** instead. The pen is **spring-loaded** and this value will work for drawing with a safe margin of a few mm.



Picture 23: Set your values accordingly!

Create the workobject and close the window. If you made a mistake, you can always change the data using the command **Modify workobject** (right-click on the created workobject). The view from above the robot will now look like Picture 24. The frame shown is the workobject and the "**User Frame**" is to the lower left corner below the surface. The red axis is the X direction and the green axis is the Y direction.



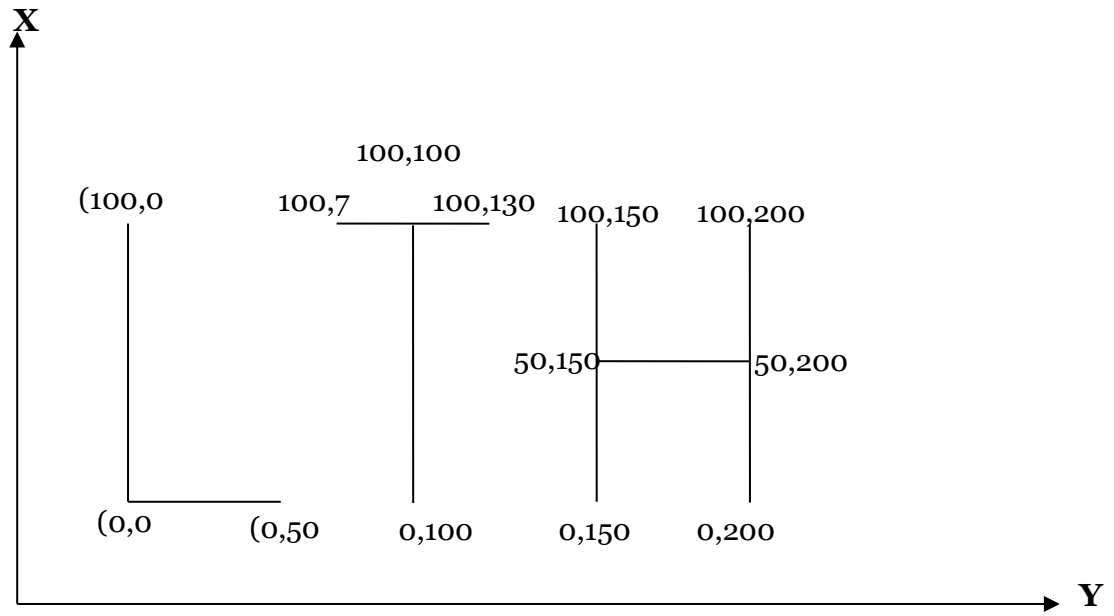
Picture 24: If your "View" does not look similar to the picture or if your values are incorrect, revisit previous instruction

This is a good time to save your station (**File -> Save Station As...**). (Due to the current problem with network-mounted disks, select a folder on the C:\-drive.

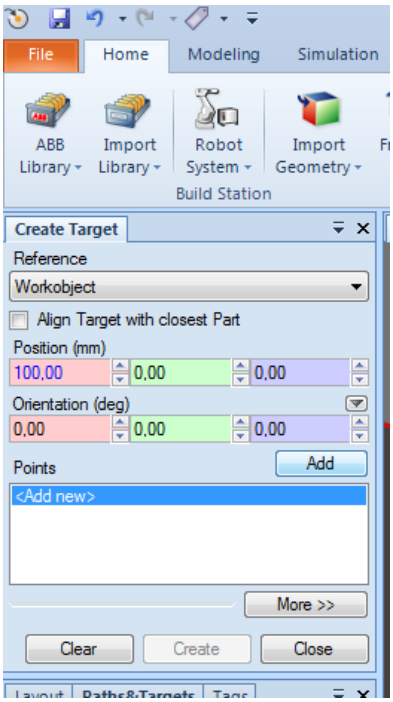
Create targets and paths for some pen drawing

You can make the drawing as you like and this example is just to illustrate how to define the targets. We will define all targets as coordinates in the **workobject frame** we defined before and assume that we write on a **paper of A3 format** (297 mm x 420 mm) but as we start 50 mm inside we can assume that we can use about 200 mm in the X-direction and 300 mm in the Y-direction.

To set up the coordinate values for the letters “LTH” we can define the targets as in the figures below. Click on **Target -> Create Target**. A window pops up where you can define the coordinate values.



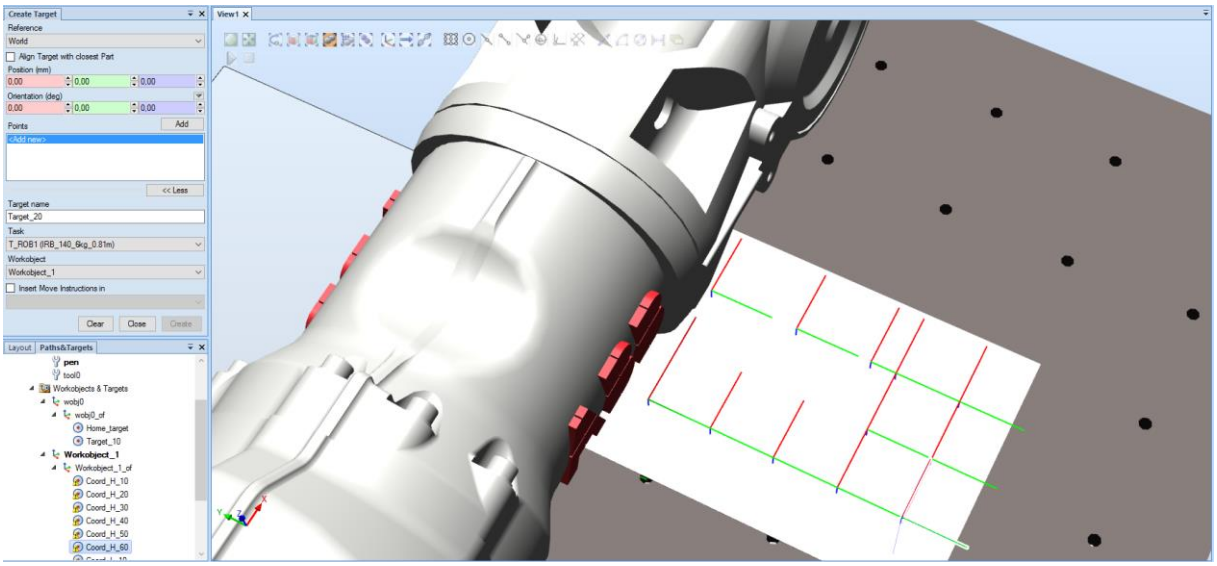
Make the following selections for the letter “**L**”. Set the reference to “**Workobject**” and write the X, Y coordinates:



Picture 25: Be sure to set “Reference” to “Workobject”

Click on “**Add**” for each new target and “**Create**” to create all three targets. The target points can automatically get new names, but it is good to rename them to something easily identifiable.

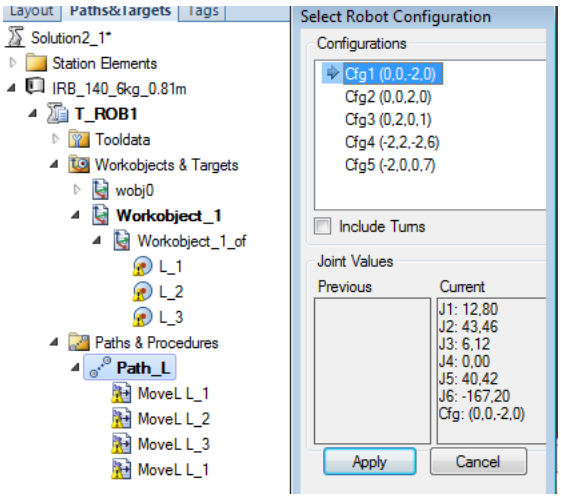
Click on “**Close**” to close the window.



Picture 26: When you start to add targets to your workobject, the x-axis, and the y-axis should start to become visible in your view.

Create a path by right click on the **Paths -> Empty path**; rename the path to "**Path_L**" (or something suitable). Drag your target point (located under your workobject) to the created path, i.e., select the three targets you made with the paper as workspace and drag them to the path. You can add the target point corresponding to the **home position** (nominally "**Target_10**", which you should rename) at the end of the path.

Check the path by right-click on the path and use "**Auto Configuration**" (under "**Configurations**"), choose "**Cfg1(0 , 0, -2 , 0)**", as indicated by the picture below.

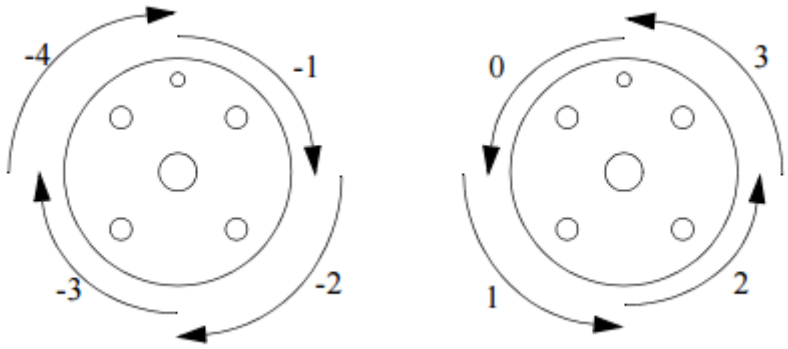


Picture 27: Under "Current", you can see the rotation of each joint for each configuration. When moving between different letters or drawings, if you can choose a configuration where the angles are similar, that is usually best.

The robot will go through the path and learn and remember each configuration for the respective targets in the path.

You can now click on the targets one by one and the robot will go to each target. Click on one of the targets. Right-click and select "**Configurations**". You can now examine the different configurations available for the robot. (The first three values correspond to the

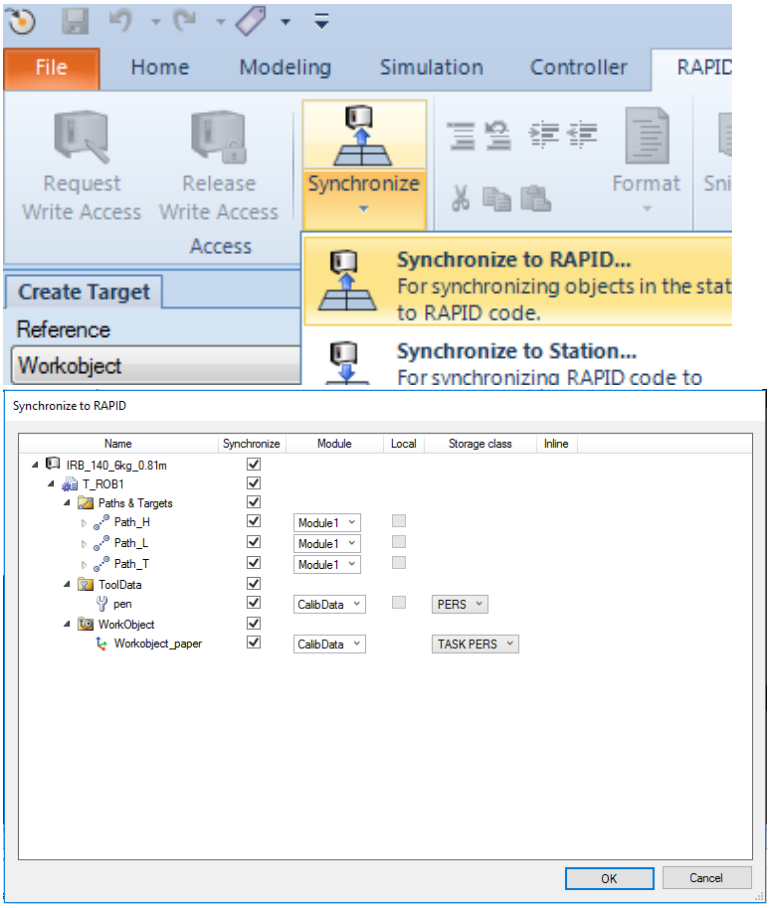
quadrant of angles for robot **joint_1**, **joint_4** and **joint_6**, respectively. The fourth value could be used for an external axis, but it is not relevant in this case).



Picture 28: The configuration quadrants for axis 6 (joint_6)

Picture 28 is taken from "**RAPID Reference Manual System Data Types and Routines On-line**", written by ABB. If you wish to learn more, read the description found at p.13 in the "RAPID Reference Manual System Data Types and Routines On-line", which is available online.

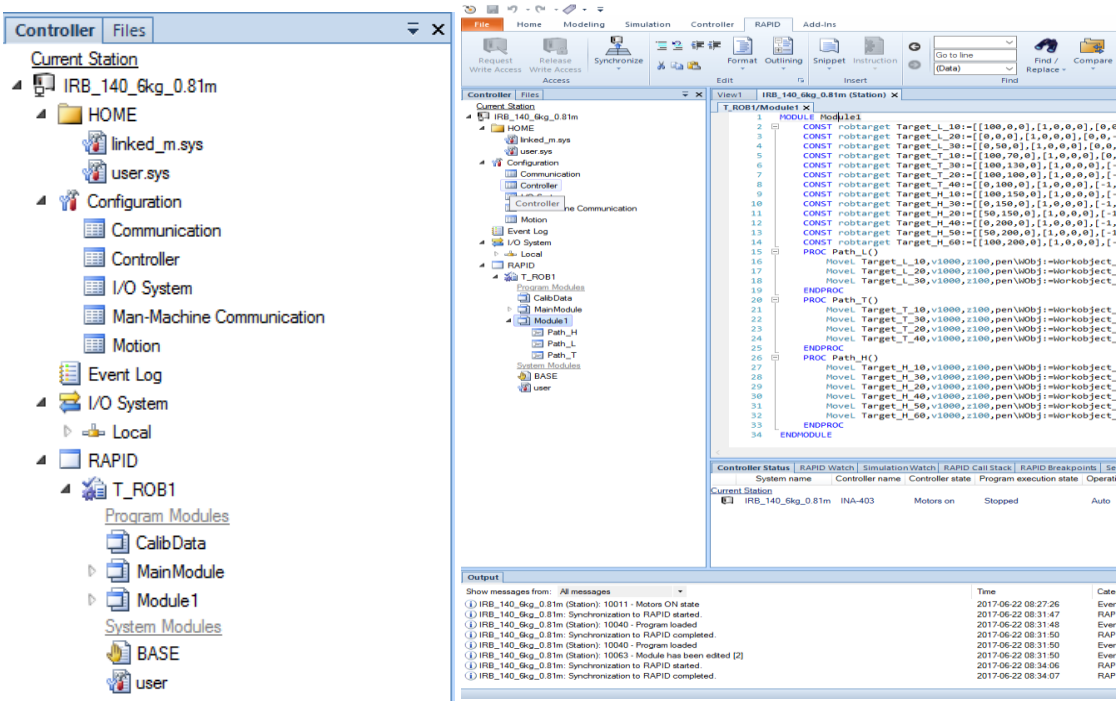
Now, we are ready to make a **RAPID program** from the work made so far. Click on the **RAPID-tab**, and **Synchronize -> Synchronize to RAPID**.



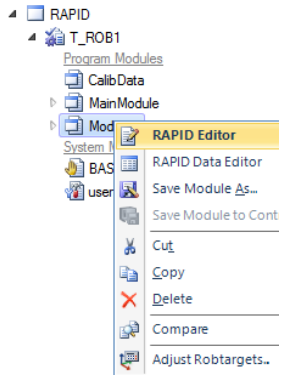
Picture 29: If you have only made small changes to a specific part of your program, you can choose only synchronizing that change.

Click “**OK**” on the popup window. By doing this you will **create/update** the programming code RAPID with the definitions of your target points and paths. Furthermore, in case you change numerical values of target points etc. in the RAPID-code in a “**RAPID-editor**” you then want **to choose** Apply **and then** Synchronize to Station for these changes to be updated.

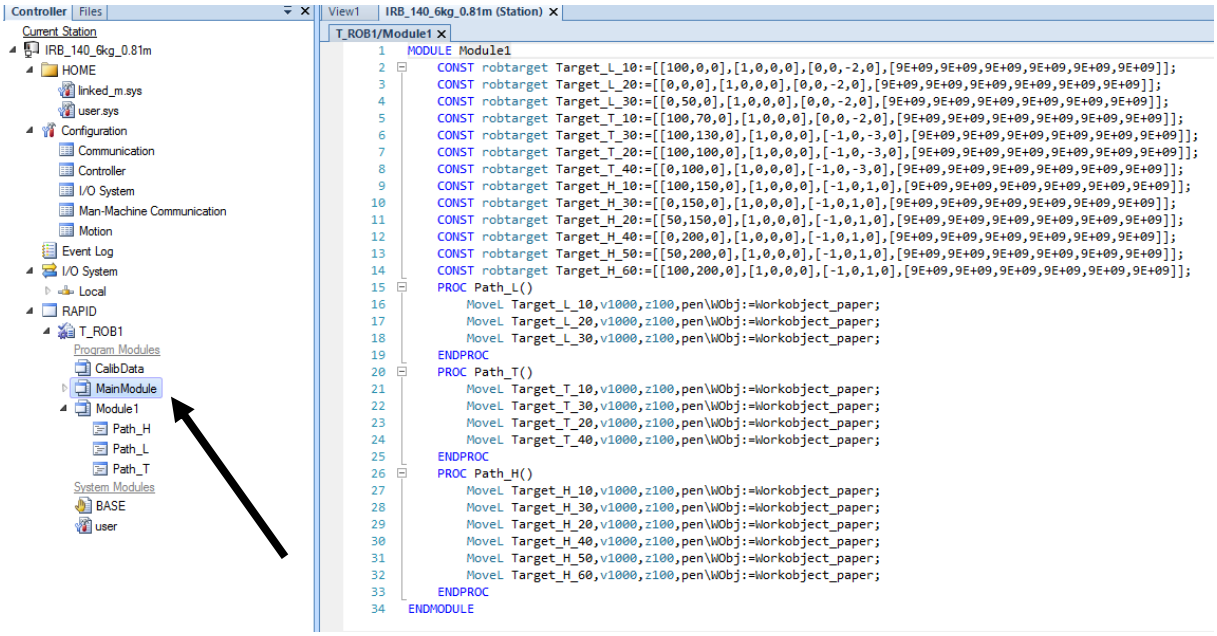
Expand the tree of the robot task we are producing so you can see the “**Program Modules**”:



Select “Module1”, and click on “RAPID Editor” icon.

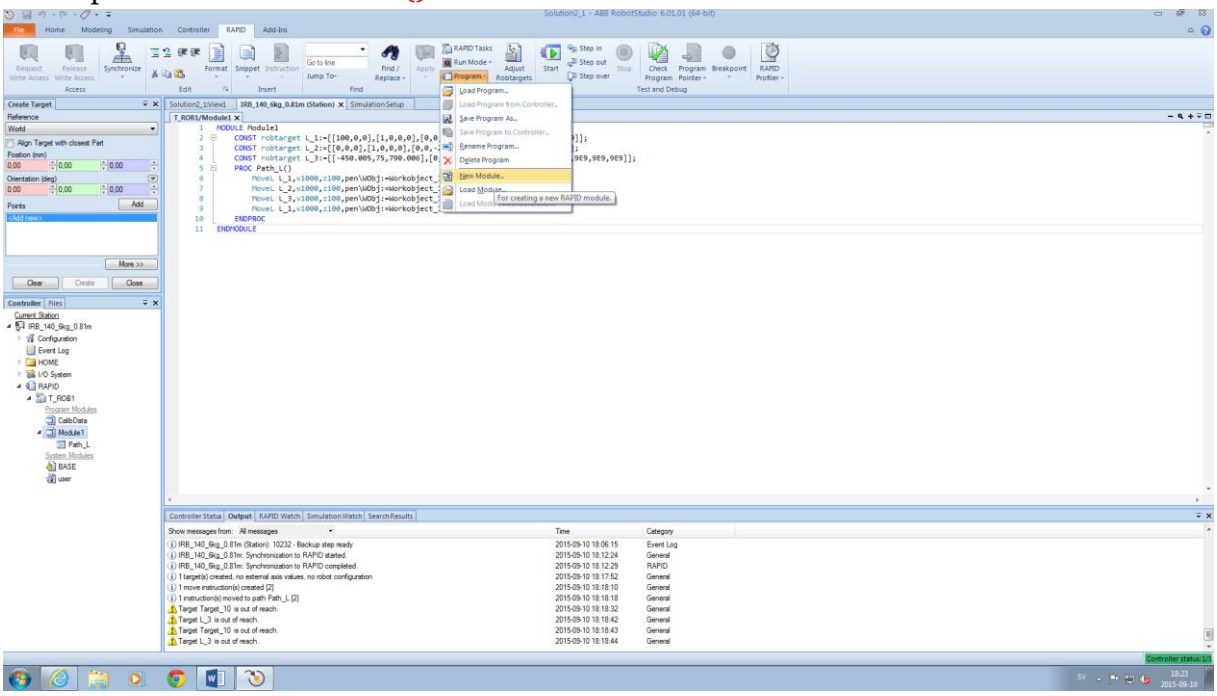


The "**RAPID editor**" will pop up and the code will look similar to this.



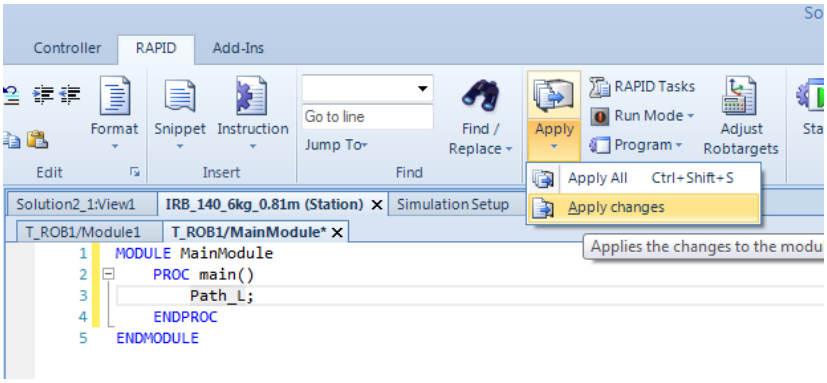
Picture 30: How the code looks in "Rapid Editor"

In some cases, as indicated by the arrow in Picture 30, there is already a "**MainModule**" with a procedure called "**main()**".



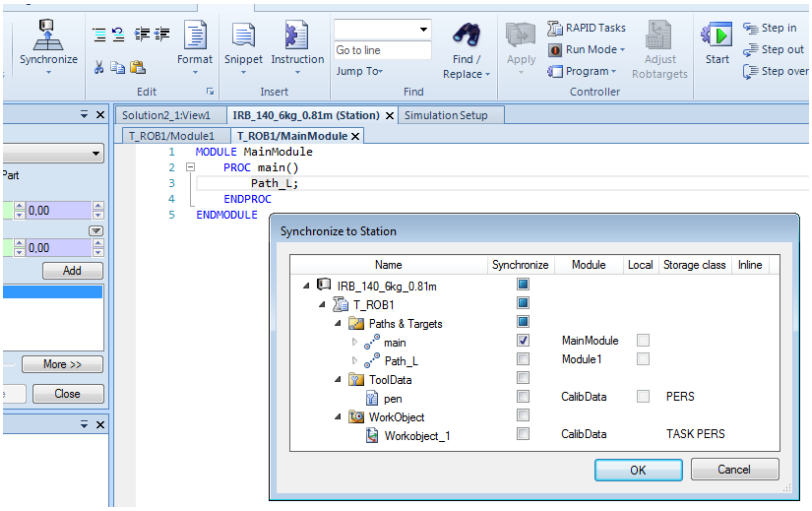
(In case you do not already have a "**MainModule**" you create by choosing **Program Control -> New module**). Name the new module "**MainModule**" and type according to the figure below.

When your code corresponds to the picture, remember to click Apply -> Apply changes



Picture 31: Go to Apply -> Apply Changes after you have written the code

After applying the changes to the **RAPID-code** we now synchronize this code with the station **Synchronize -> Synchronize to Station** (you only need to mark "**main**" as this is the only we have changed but selecting all works fine).



Click on "**OK**" and go to the "**RAPID editor**" for the module with your paths (e.g. "**Module 1**", which contains "**Path_L**").

Note that the "**main()**"-procedure can be in the same or a different module as "**Path_L**" and from within "**main()**" you can call procedures from other modules as long as they are loaded to the controller.

We can now do some preliminary editing. To make a **drawing**, we must **move down to Z value 0**. This can be done using an **offset instruction**. In the editor, you can copy and paste as usual and if we rearrange the drawing we start at "**Target_L_10**", draw to "**Target_L_20**", and finish at "**Target_L_30**". Look at the picture to see what name corresponds to which coordinate. If your "**Target_L_20**" corresponds to the guide's "**Target_L_10**", you can either rename your target or change the lines accordingly. Be sure to synchronize to RAPID if you decide to change the names of the targets.

Change the **v** (**velocity**) and **z** (**zone**) arguments so that the velocity during drawing is 100 and z is 1, except for the **home position**, if you have that, which can have **z100**. The final code will then look similar to Picture 32.

View1 | IRB_140_6kg_0.81m (Station) ×

T_ROB1/Module1 × | T_ROB1/MainModule

```
1  MODULE Module1
2      CONST robtarget Target_L_10:=[[100,0,0],[1,0,0,0],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
3      CONST robtarget Target_L_20:=[[0,0,0],[1,0,0,0],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
4      CONST robtarget Target_L_30:=[[0,50,0],[1,0,0,0],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
5      CONST robtarget Target_T_10:=[[100,70,0],[1,0,0,0],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
6      CONST robtarget Target_T_30:=[[100,130,0],[1,0,0,0],[-1,0,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
7      CONST robtarget Target_T_20:=[[100,100,0],[1,0,0,0],[-1,0,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
8      CONST robtarget Target_T_40:=[[0,100,0],[1,0,0,0],[-1,0,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
9      CONST robtarget Target_H_10:=[[100,150,0],[1,0,0,0],[-1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
10     CONST robtarget Target_H_30:=[[0,150,0],[1,0,0,0],[-1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
11     CONST robtarget Target_H_20:=[[50,150,0],[1,0,0,0],[-1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
12     CONST robtarget Target_H_40:=[[0,200,0],[1,0,0,0],[-1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
13     CONST robtarget Target_H_50:=[[50,200,0],[1,0,0,0],[-1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
14     CONST robtarget Target_H_60:=[[100,200,0],[1,0,0,0],[-1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
15     PROC Path_L()
16
17         MoveL Target_L_10,v100,z5,pen\WObj:=Workobject_paper;
18         MoveL Offs(Target_L_10,0,0,15),v100,z1,pen\WObj:=Workobject_paper;
19         MoveL Offs(Target_L_20,0,0,15),v100,z1,pen\WObj:=Workobject_paper;
20         MoveL Offs(Target_L_30,0,0,15),v100,z1,pen\WObj:=Workobject_paper;
21         MoveL Target_L_30,v100,z1,pen\WObj:=Workobject_paper;
22
23     ENDPROC
```

Picture 32: Look at "PROC Path_L()" and write your code accordingly if the names of your targets correspond to the picture.

Synchronize to Station

Name	Synchronize	Module	Local	Storage class	Inline
IRB_140_6kg_0.81m	☑				
T_ROB1	☑				
Paths & Targets	☑				
main	☑	MainModule	☐		
Path_H	☑	Module1	☐		
Path_L	☑	Module1	☐		
Path_T	☑	Module1	☐		
ToolData	☑				
pen	☑	CalibData	☐	PERS	
WorkObject	☑				
Workobject_paper	☑	CalibData		TASK PERS	

OK | Cancel

Picture 33: How the Synchronize to Station pop-up window looks

Click on **Apply changes** as indicated in Picture 31. Any errors will be indicated and you need to correct these. When all indicated error messages have been corrected,

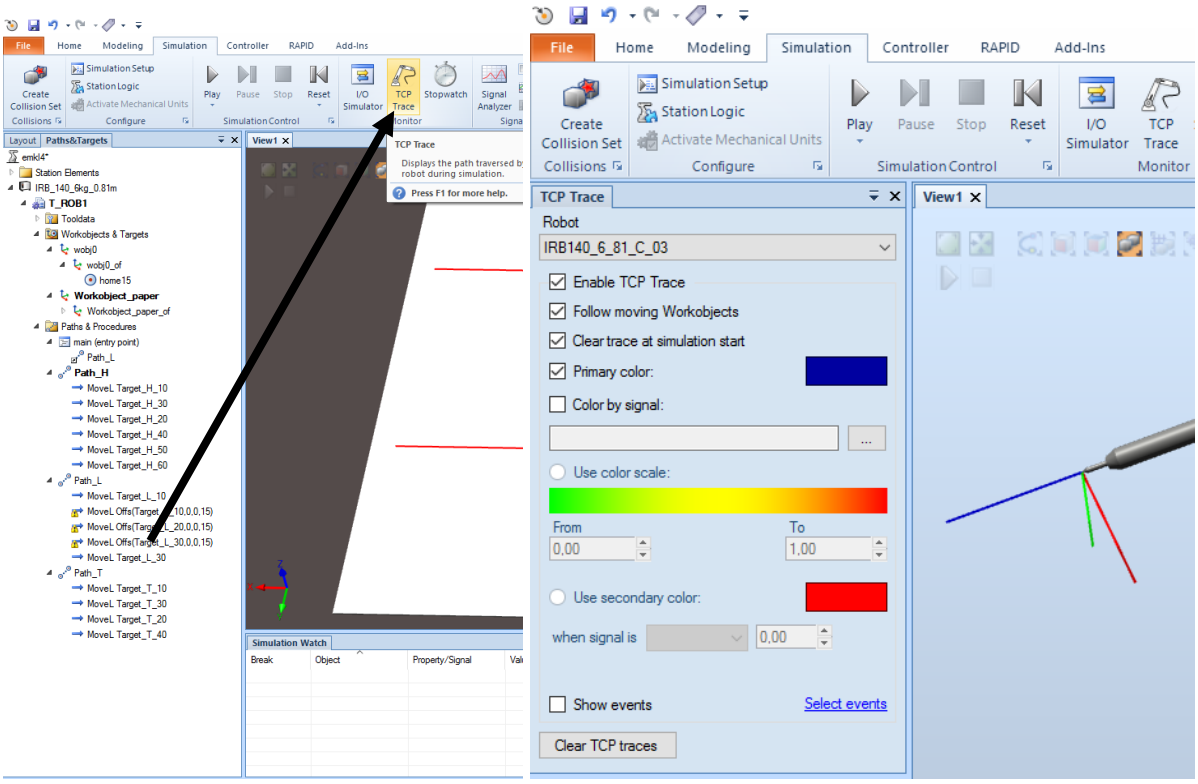
click **Synchronize -> Synchronize to Station** (in the **RAPID-tab**) and click "OK" in the popup window. Click on the **Simulation tab** and run the simulation by clicking the **Play button**.

You might get an error during the simulation. This is related to a common problem when making long linear movements. Change the move instruction to **MoveJ** between the FIRST and SECOND target, and between the SECOND target and the LAST target one as well. For information on the difference between MoveL & MoveJ, see p.241 and 249 in "RAPID Reference Manual System Data Types and Routines On-line", which is available online.

Go to the **RAPID tab** and **Synchronize -> Synchronize to RAPID**. Notice how the RAPID code changes. Run the simulation again (possibly need to make a Reset before, which is done by pressing Reset). Change your view and make a close-up to see the movement on the paper.

Additional Assignment: Your job is to add more targets to your "Workobject". The targets should represent at least one or a few more drawings or letters. Create a new path for each new drawing. Edit the RAPID code with the "RAPID Editor" with a call of the new procedures in the "main()" procedure or use the simulation setup command as described above.

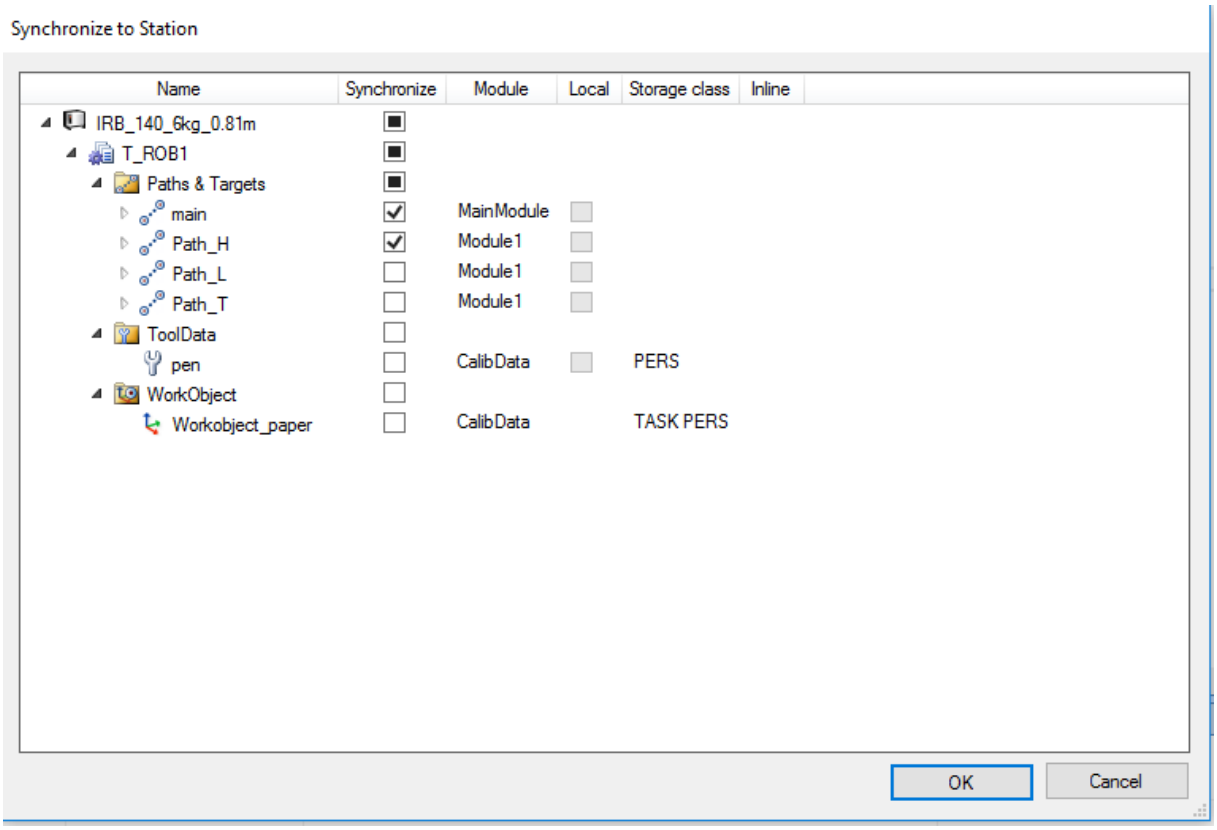
To better view your drawing, you can activate a trace of where the TCP has moved: **Simulation -> Monitor -> TCP Trace** (see the pictures below).



Picture 34: Find TCP Trace and set it to a color of your own choosing.

Note: Having problems with network drives, the program can stop working at an unexpected time. Save your work **FREQUENTLY** during the program editing process. Synchronization

can stop working and there are some work-arounds to this. Try Restart (P-start) and sync only the "main" and one of the paths.



Picture 35: If you have a problem synchronizing your work, try to synchronize only the "main" and one of the paths.

In most cases there is an error somewhere, for example a configuration error (use "**Auto Configuration**" of the paths, or check reachability). If it still does not work, try to take away any instructions like "**Offs**", which we have found "sometimes may create problems".

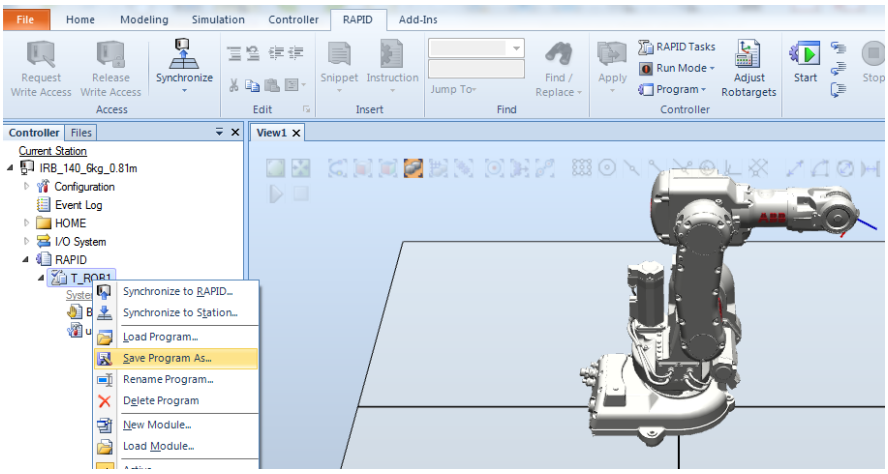
Check the "**Auto Configuration**" again and if no errors, go for a P-start. Then try to do Synchronization to Station. Then add paths and check that it works. The **best rule** is always: **do not make any errors**☺!

Finally, **SHOW** the resulting simulation for the teaching assistant.

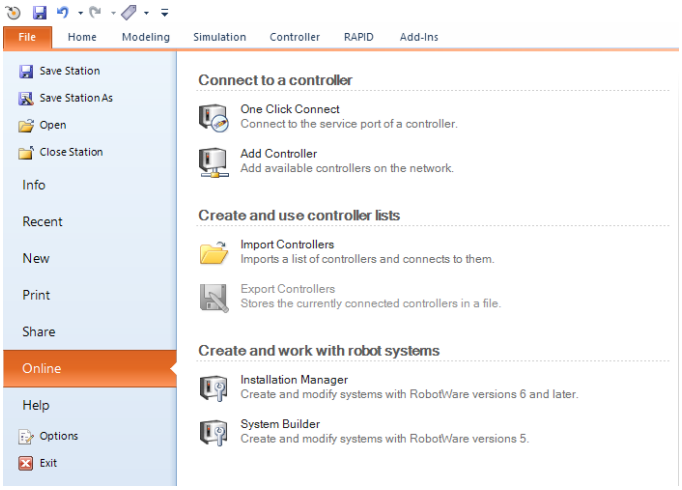
Testing your program on the real robot

Before testing on the real robot, **show** and **demonstrate** your program for the teaching assistant. You can either save the complete robot program to a USB-stick from which you can load the program via the **FlexPendant** of the robot controller IRC5. Make sure you save **ALL** modules needed, not only the main program.

To do this right-click on "**T_ROB1**" and "**Save Program As…**"(see the following picture).



Another alternative: If you have a computer connected to the robot, you can download the RAPID-program to the robot from RobotStudio. You can connect with the robot via File -> Online -> One Click Connect



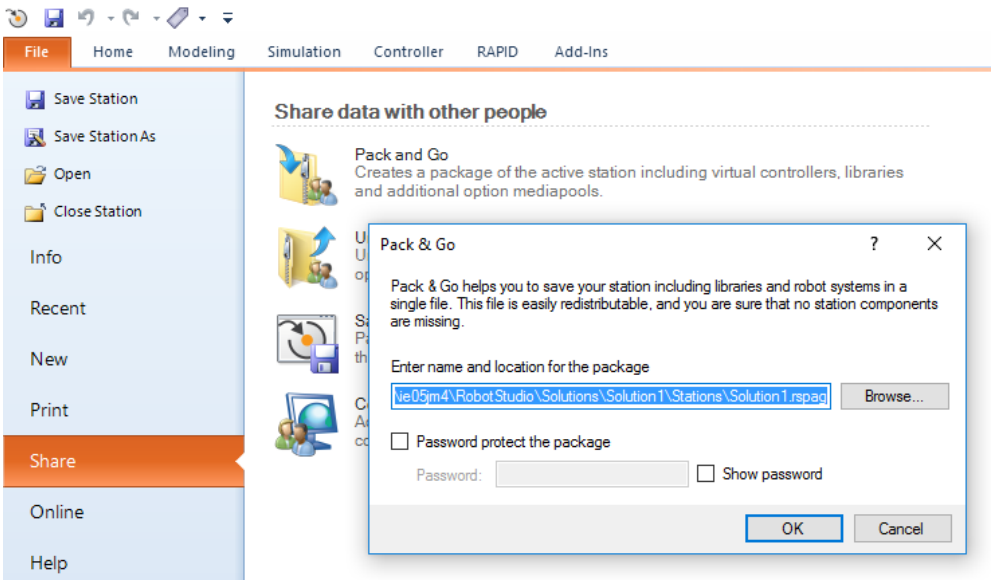
LUNDS UNIVERSITET
Lunds Tekniska Högskola

Appendix - Save stations with Pack&Go

To transfer stations and solutions between different computers with possibly different setups of RobotStudio (different versions of Robotware or differences in the file system where components and programs are located) the safest way is to create a so-called complete Pack&Go-file (.rspag) which means that when saving the station all components are copied and all components collected instead of linking to them.

You create such a pack-and-go-file and unpack such a station via the Share-menu, see figure below.

Do not forget to copy the Pack&Go file FROM the TEMPORARY local disk to your HOME-FOLDER before leaving the exercise!



Saving a station with Pack&Go is the best way to ensure that you get all components if you are moving stations/solutions between different computers.

Possible error messages if not correct network drive. Similar error messages can arise when reopening a Pack&Go file. Copy the Pack&Go file to your temporary local disk and select that

version when you try to open the file. You will once again have to create a new subfolder to open your file (see above in **Error! Reference source not found.**).