Programming the robot

In the previous chapter, we learned how to define surface target points. Now, it is time to learn how to make our first program and ensure that the previously defined target points must be traversed in a specific order. This task is considered as the continuity of the task accomplished in previous chapter.

Paint Spray Operation

As the target points have been defined, instead of picking target points one by one, it is possible to define a sequence of actions using targets. This whole operation can be split into three sequences:

1. From Home position to Approach position.
2. Go through all surface points.
3. From the last target surface point to the Retract position and then back to the Home position.

From Home to Approach position

Select both Home and Approach target points with the CTRL key and select 'Create Program' from the right-click menu.

In addition to the target positions, a reference frame and a tool are also added automatically. The program can be renamed as 'Home2Approach'. Now, if you select 'Run' option from the right-click menu of the program, you will be able to see your robot in action and jumping between Home and Approach Targets.

Note

By default, the first movement in the program is defined as a movement in joint space and referred to as MoveJ. The movement space can be easily switched between Joint move and Linear move by the right-click menu option of any individual target point.

Sequence through Surface points

A similar operation is repeated for all the surface points. All the points need to be selected first, and then a program is created for these target points.

The program has been renamed as ‘Surface Paint’ to identify its operation. As expected, the reference frame and tool are automatically added. You can witness the sequence of operations by running the program.

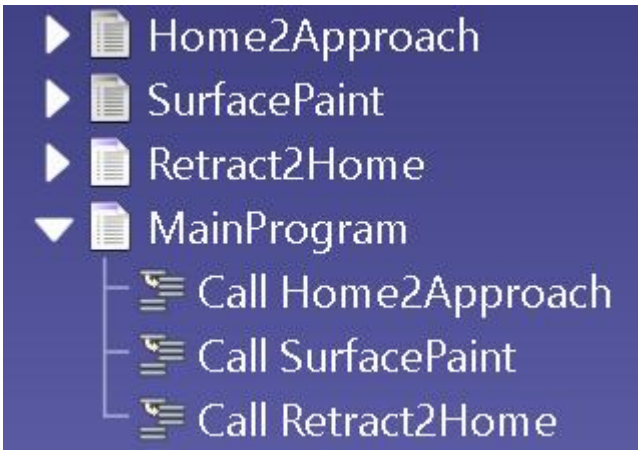
Return to the Home position

In the previous program, the robot is supposed to complete its operation at the last surface point identified as ‘Surface 8’. From there, it must jump to the defined retract target point and return to the home. For this purpose, select the retract and home target points and create a program.

If the sequence of instructions is not according to your desire, it is very easy to hold and move up and down the list to change the sequence.

Main Program

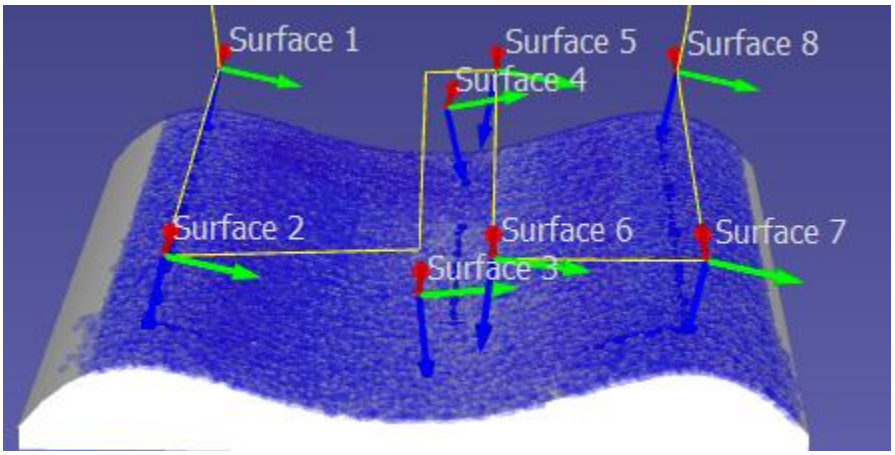
The individual programs are well defined and are working perfectly. But we are interested in defining the whole operation, starting and ending at the home position. For the purpose, select all three programs and select 'Make Main Program' from the right-click menu. A new main program is created which calls three sub programs. This concept is like the main function and calling sub-functions from the main function.



Now, it is possible to see the whole operation by just running the main program. If you like to repeat these operations indefinitely, then enable 'Loop' options from the main program's right-click menu.

Introducing Macro

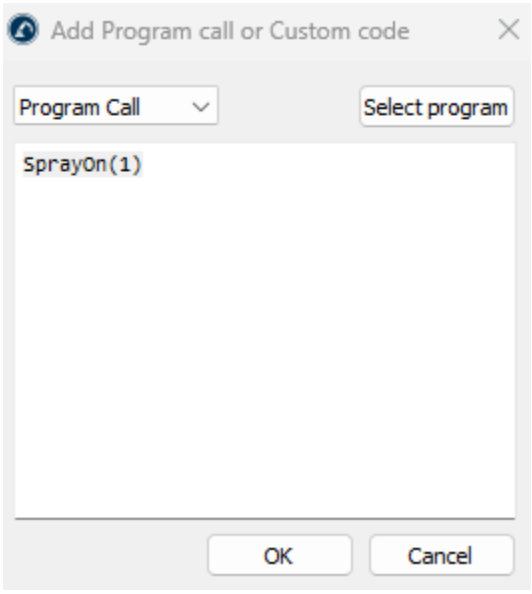
RoboDK improves visualization with the help of the scripts. These scripts can be written in Python and are introduced as Macros to the existing workstation. These macros don't interrupt the normal sequencing of the program, rather run parallel to the actual program. For the paint sparying, RoboDK has already provided a macro that can be accessed from Library → Macros → Spray on. Double clicking the macro will help to turn it on or off. The results of the spraying are displayed in the figure; the target points can be modified to perform fine tuning.



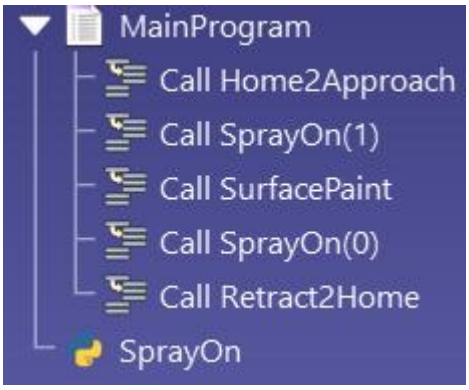
To make it more alive and respond to the situation, spray operation can be turned on/off through instructions.

Program Call Instructions

To enable/disable the paint operation, it must be controlled by means of instructions. For this purpose, RoboDK allows us to make additions to the existing program. Right click the MainProgram to access Add Instruction → Program Call Instruction. A dialog box will pop up through which spray operation can be enabled/disabled using flag 0/1 as an argument.



It is desired that the spray paint must be enabled when robot starts operating towards the first surface paint and is disabled when operation is completed and it starts retraction.



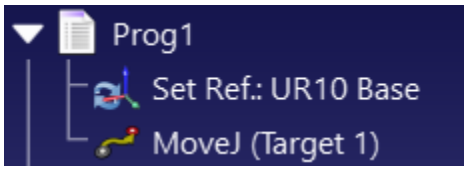
Generate a program from scratch

Programming your robot is possible in various ways with a lot of flexibility. Import into RoboDK station a robot of your interest.

Either from the toolbar  or from the program menu, select Add Program that will list the empty program in the station tree as Prog1. Same instructions related to the program can either be accessed from the Program menu or by right clicking the program name and then proceeding to the Add Instruction with further list of options.

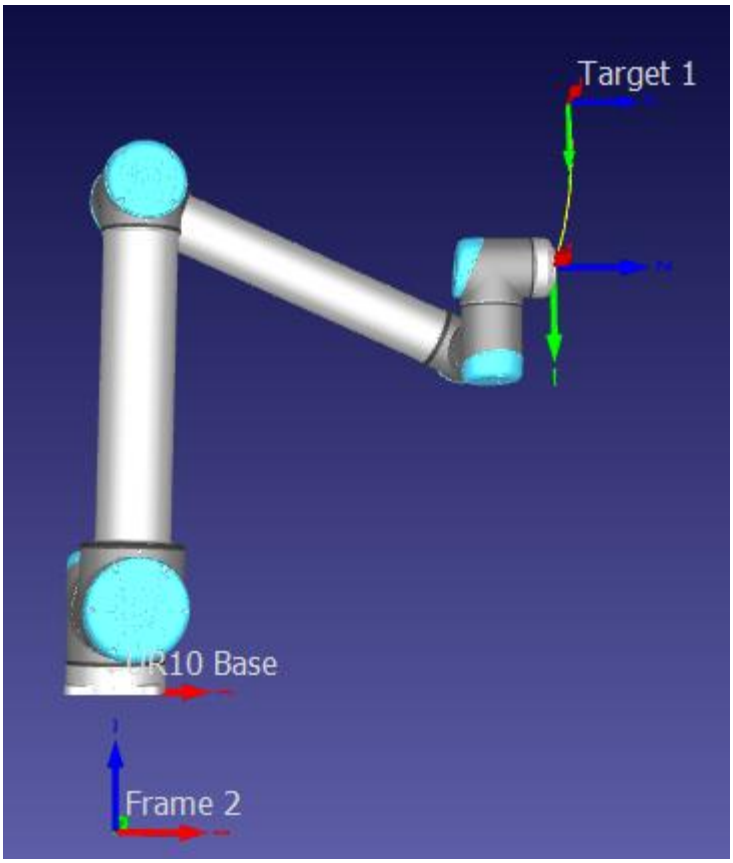
Move Joint Instruction

Move instruction is used to command a robot's tool to access desired location. If it is the first instruction in the program, then this action will add two entries under the program tree. First one is the referencing, by default, it is referenced with respect to the base of the manipulator unless modified.



Set Reference Frame

A new reference frame is introduced to the scene and placed roughly under the base reference frame. Now by changing the referencing from Set Reference Link option available in the Set Ref to the new frame just defined.



It is observed that the tool moves to a new position instead of Target 1 because a new base referencing is defined.

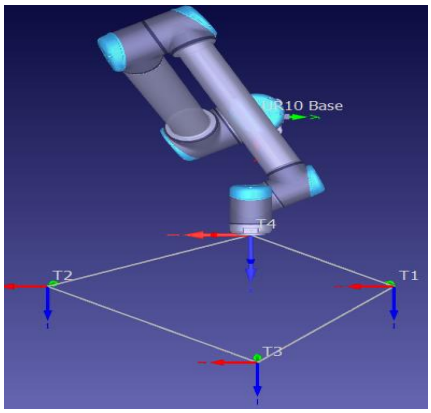
Tool with respect to reference frame = Target position with respect to frame 2 + Frame 2 with respect to base

$$^{new}T = {}_{2}Target + {}_{base}Frame2$$

$$= \begin{bmatrix} 672.919 \\ -276.648 \\ 1080.600 \\ -69.282 \\ 69.282 \\ -69.282 \end{bmatrix} + \begin{bmatrix} -9.119 \\ 112.748 \\ -224.200 \\ 0.00 \\ 0.00 \\ 0.00 \end{bmatrix} = \begin{bmatrix} 663.800 \\ -163.900 \\ 856.400 \\ -69.282 \\ 69.282 \\ -69.282 \end{bmatrix}$$

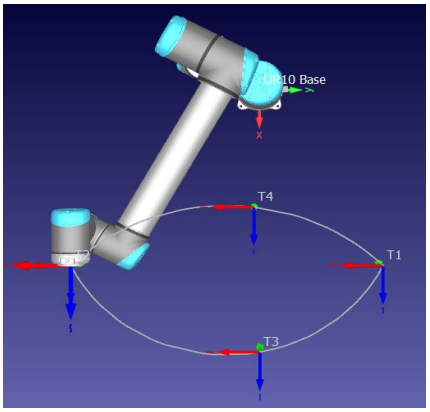
Move Linear Instruction

To obtain linear movements between two target points, a linear instruction, MoveL, can be added to the program. It just needs one target point to which a straight path is generated.



Move Circular Instruction

If the desire is to form an arc, then Move circular instruction is built for this purpose. From the existing pose, the first target is linked by its option menu through which this arc will pass through. The second linked target specifies where to end this arc. It is only possible to obtain an arc through which a semi-circle can be made. To obtain a full circle, two circular instructions need to be utilized.

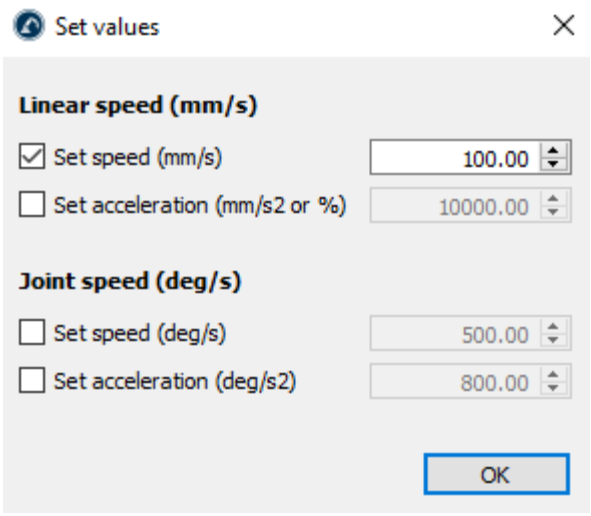


Set Tool Frame Instruction

If the manipulator is equipped with multiple tools, then in RoboDK, we have to specify the active tool. Likewise, in the program, through the set tool link, any of the available tools can be picked.

Set Speed

It is also possible to set speed/acceleration in cartesian or joint space. It is even possible to set different speeds in the same program to perform various actions.



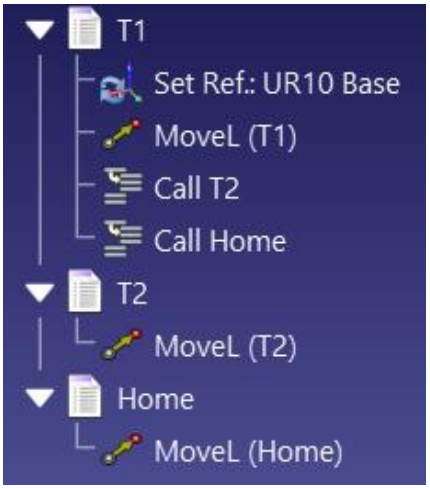
With respect to the base, Target 1 and Target 2 are specified as (895.462, -163.900, 855.086, -69.282, 69.282, -69.282) and (595.462, -163.900, 855.086, -69.282, 69.282, -69.282) respectively. To perform a complete cycle, the length measured along the X-axis is 600mm. As the speed is specified as 100 mm/s, it requires 6.0 s to execute the program.

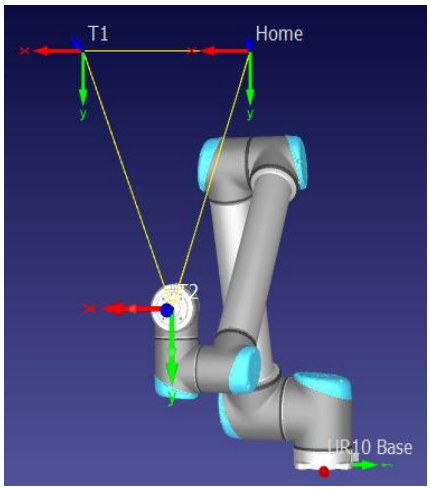
Pause Instruction

It is also possible to introduce delay or pause between two instructions. The pause time is specified in ms. After introducing pause, the whole execution time of the program increased to 6.5s.

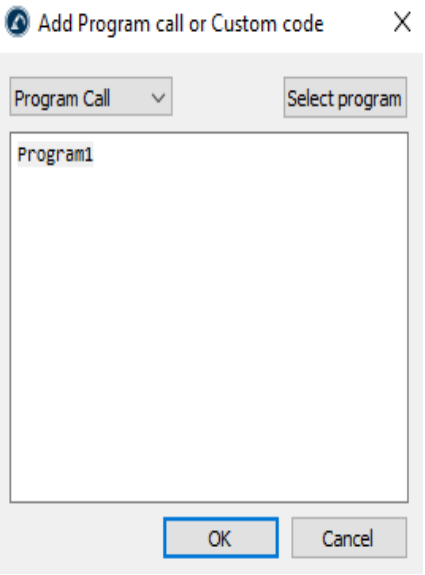
Program Call Instruction

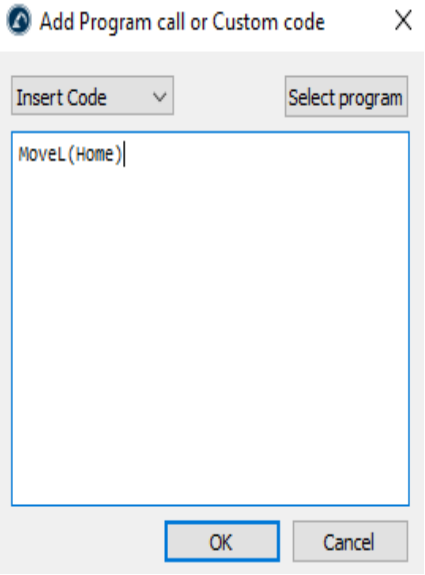
Through program call instruction, it is possible to reuse one program or sub-program at multiple instances.





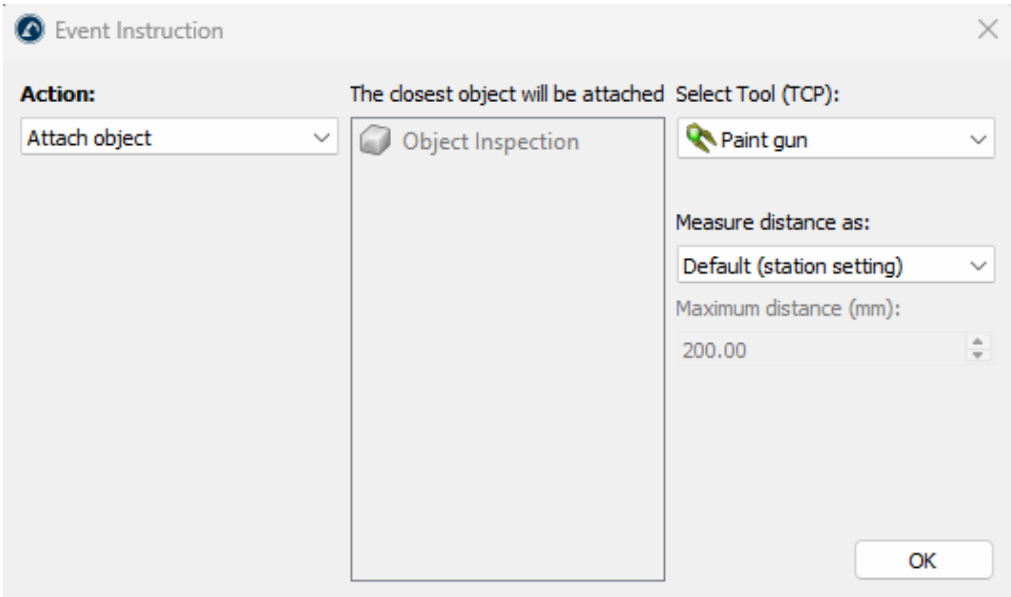
In the call instruction dialog box, two most useful options are either to select programs that already exist in the station tree or it is also possible to write instructions manually.





Simulation Event Instruction

In the simulation environment, it is possible to invoke some special actions which obviously will not affect the actual code.



Following set of actions are possible:

Attach or detach objects to robot tools

This sort of action is handy when we wish to simulate operations like pick and place. A gripper or vacuum reaches a specific object, then that specific object is attached to the robot and upon reaching the desired location, detach operation can be triggered.

Show or hide objects or tools

If a robot is equipped with multiple tools and you wish to show or hide certain tools depending upon the requirement of the operation, then this option is handy.

Change the position of objects and reference frames

During the buildup, it is quite possible that objects and reference frames get disturbed from their desired pose. To ensure that the robot is always working in a proper manner, it is best practice to replace the frames at their original pose.

Simulate Program

To run the specific program, right-click the program name and select run. The execution of instructions will be initiated one by one and the whole program will run for once.

If it is desired that the program should run indefinitely, then the **Loop** option must be enabled.

Double click on the program menu and this will open the simulation bar at the bottom of the screen. The simulation bar allows you to perform some control actions, such as **jumping** to the start or end of the simulation. **Play**, **pause** and **fast simulation** options are also available. Counterclockwise **circular arrow** sets the program to repeat whereas toggling it runs the program only once.

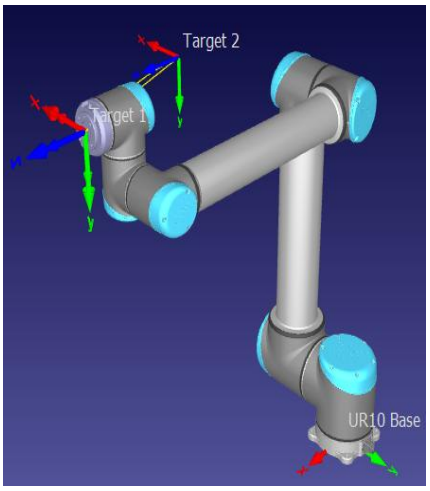
Note

By default, simulation bar has a curved arrow which on pressing replaces itself with the circular arrow.

Generate Program

Probably, RoboDK's strongest strength is its capability to generate programs for various types of robots without having any concern about the type of language specific to that robot.

A simple program is generated that guides the UR10 robot to move between two targets.



With a right click on the program's name, Generate Robot Program (F6) option can be clicked. By default, it is stored to ../RoboDK/Programs directory with the file name same as the name of the program.

This is what the generated program looks like. The program's length and the script changes with the type of robot. This program is saved as Prog1.script

```
def Prog1():
  # Global parameters:
  global speed_ms    = 0.250
  global speed_rads  = 0.750
  global accel_mss   = 1.200
  global accel_radss = 1.200
  global blend_radius_m = 0.001
  global ref_frame = p[0,0,0,0,0,0]

  #
  # TO REMOVE HEADER:
  # Go to "Program" -> "Post-Processor Editor"
  # Select "Universal Robots"
  # Set "INCLUDE_HEADER" to "False"
  #
  #--------------------------
  # Add any default subprograms here
  # For example, to drive a gripper as a program call:
  # def Gripper_Open():
  #   ...
  # end
  #
  # Example to drive a spray gun:
  def SprayOn(value):
```

```
  # use the value as an output:
  DO_SPRAY = 5
  if value == 0:
    set_standard_digital_out(DO_SPRAY, False)
  else:
    set_standard_digital_out(DO_SPRAY, True)
  end
end

# Example to drive an extruder:
def Extruder(value):
  # use the value as an output:
  if value < 0:
    # stop extruder
  else:
    # start extruder
  end
end

# Example to move an external axis
def MoveAxis(value):
  # use the value as an output:
  DO_AXIS_1 = 1
  DI_AXIS_1 = 1
  if value <= 0:
    set_standard_digital_out(DO_AXIS_1, False)

    # Wait for digital input to change state
    #while (get_standard_digital_in(DI_AXIS_1) != False):
    #  sync()
    #end
  else:
    set_standard_digital_out(DO_AXIS_1, True)

    # Wait for digital input to change state
    #while (get_standard_digital_in(DI_AXIS_1) != True):
    #  sync()
    #end
  end
end
#-------------------------


# Main program:
# Program generated by RoboDK v5.9.1 for UR10 on 22/07/2025 09:35:00
# Using nominal kinematics.
```

```
 ref_frame = p[0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000]
 movej([0.000000, -1.169432, -1.787968, -0.184190, 1.570796, -0.000000],accel_radss,speed_rads,0,0)
 movel(pose_trans(ref_frame,p[0.663800, -0.163900, 0.856400, -1.209200, 1.209200, -1.209200]),accel_mss,speed_ms,0,0.001)
 # End of main program
end

Prog1()
```

If we change the robot to PUMA560 with the new reachable target points, the program is generated as Prog1.cnc

```
; program: Prog1()
G161
G90
F15000
; Program generated by RoboDK v5.9.1 for PUMA 560 on 22/07/2025 09:39:19
; Using nominal kinematics.
; Setting reference frame:
; PUMA 560 Base: X   0.000 Y   0.000 Z   0.000 A   0.000 B   0.000 C   0.000
G00  D     0.000000 E     0.000000 F     0.000000 G     0.000000 H     0.000000 I    -0.000000
G00  D    -8.564240 E   -23.697300 F    76.837600 G     0.000000 H   -53.140400 I     8.564240
; ENDPROC
```

If you have a Main program with different subprograms, RoboDK generates a robot program for each program separately.

Transfer of Program

Once the program is generated, an actual robot can be driven in two possible ways.

1. Manually transferring the codes to the robot controller

In this option, you must do the hard work of uploading the program from your PC to the robot controller and run it later. This option can be laborious if you have to do it multiple times.

2. Automatically transferring the codes to the robot controller

First, you must configure your robot from the Connect menu. This communication is performed using File Transfer Protocol (FTP). In the robot settings, the IP address must be configured along

with the port. After configuring, it is recommended to check the stability of the connection by pinging the robot. In case of success, a connection is established which you can also verify through Get Position command. In case of failure, a connection status is displayed as Failed.

After successfully configuring the robot, from the right-click menu of the program, a program can be directly sent to the robot. After a successful transfer, a program can be run on a robot. You can enable this option from the right-click menu of the program.

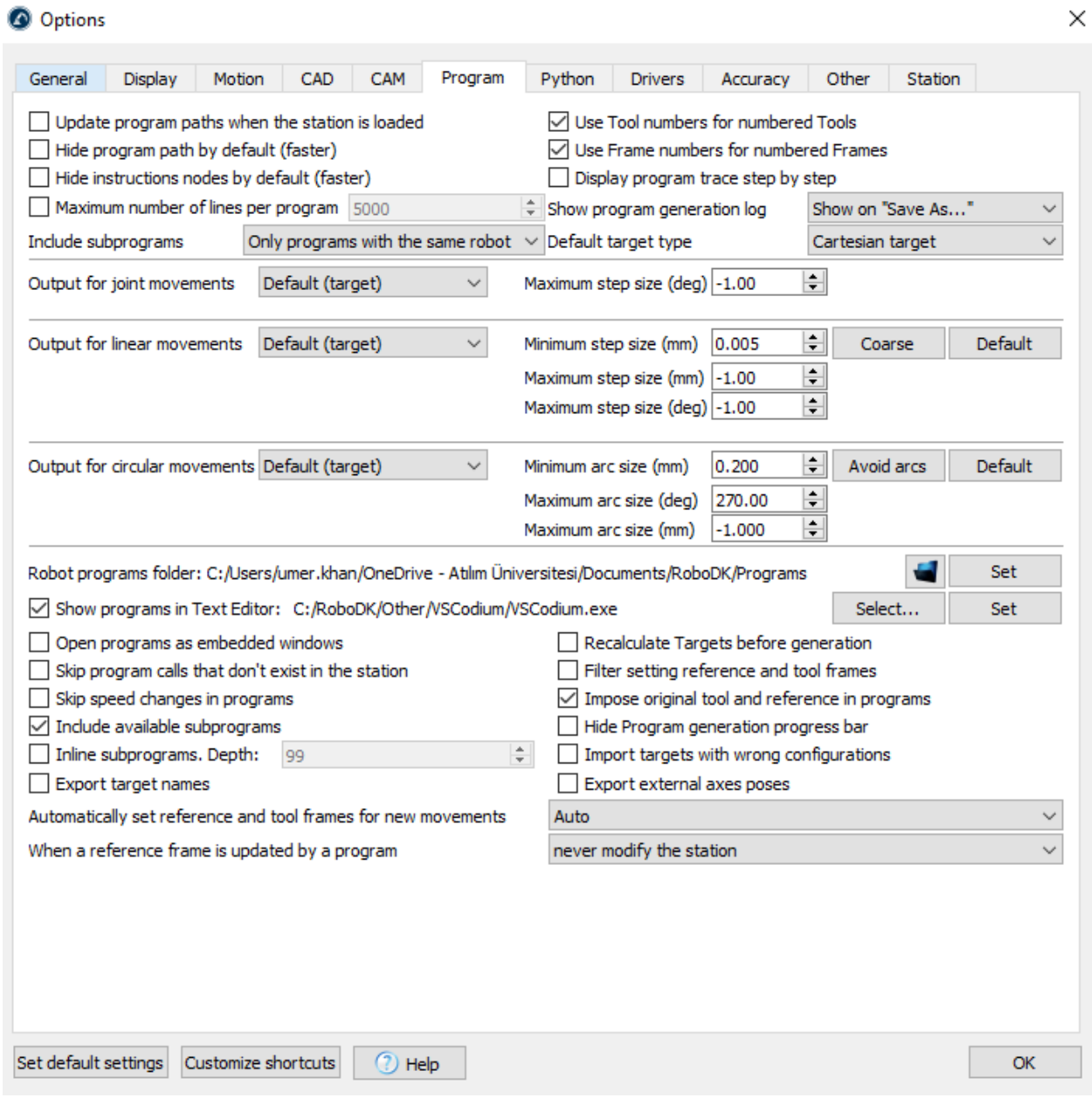
Selection of Post Processor

The post processor serves the purpose for the generation of a program that is specific to the robot. It automatically determines the post processor suitable for the robot. In case you wish to define the post processor on your own, you can either do it from the main Program menu from the toolbar or from the right-click options of the program.

Program Settings

If you wish to change the default Program generation settings, you can always override the existing settings from Tool → Options → Program. It is quite enriched and does contain very useful options. It is not possible to go through every option, just a few are discussed.

1. It is possible to limit the length of the program, then a larger program exceeding a certain number of lines will be split to multiple programs.
2. Default target type can be changed from Cartesian to Joint.
3. Minimum, maximum step sizes can be defined for joint and linear movements.
4. Minimum and maximum arc sizes can also be defined for circular movements with the option to avoid arcs.
5. A default Text Editor of your likings can also be defined for displaying the programs.
6. If you do not wish to have separate files for the sub-programs, inline subprogram option can be used to make them a part of the main program.


Options
General
Display
Motion
CAD
CAM
Program
Python
Drivers
Accuracy
Other
Station
Update program paths when the station is loaded
Hide program path by default (faster)
Hide instructions nodes by default (faster)
Maximum number of lines per program
5000
Use Tool numbers for numbered Tools
Use Frame numbers for numbered Frames
Display program trace step by step
Show program generation log
Show on "Save As..."
Include subprograms
Only programs with the same robot
Default target type
Cartesian target
Output for joint movements
Default (target)
Maximum step size (deg)
-1.00
Output for linear movements
Default (target)
Minimum step size (mm)
0.005
Coarse
Default
Maximum step size (mm)
-1.00
Maximum step size (deg)
-1.00
Output for circular movements
Default (target)
Minimum arc size (mm)
0.200
Avoid arcs
Default
Maximum arc size (deg)
270.00
Maximum arc size (mm)
-1.000
Robot programs folder: C:/Users/umer.khan/OneDrive - Atılım Üniversitesi/Documents/RoboDK/Programs
Set
Show programs in Text Editor: C:/RoboDK/Other/VSCodium/VSCodium.exe
Select...
Set
Open programs as embedded windows
Skip program calls that don't exist in the station
Skip speed changes in programs
Include available subprograms
Inline subprograms. Depth:
99
Export target names
Recalculate Targets before generation
Filter setting reference and tool frames
Impose original tool and reference in programs
Hide Program generation progress bar
Import targets with wrong configurations
Export external axes poses
Automatically set reference and tool frames for new movements
Auto
When a reference frame is updated by a program
never modify the station
Set default settings
Customize shortcuts
Help
OK