Izmir
Lubin
Pescara
Ribadeo
2023

Using A Simulator For Programming In Robotic

INDUSTRIAL ROBOTS

COLLABORATIVE ROBOTS

SIMULATOR SOFTWARE

PROGRAMMING LANGUAGE FOR ROBOTS

MAINTENANCE PRINCIPLES

KA210VET Erasmus+ Project

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY

Funded by
the European Union

INDEX

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY

Funded by
the European Union

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

PREFACE

Industrial robots have a great role in the Industry 4.0 industrial revolution. Many process operations are carried out under industrial robot control. It is important for our students to understand and learn about industrial robot systems. One of the biggest needs of our students in this regard is the need for applied source books suitable for second level technical education.

As the SIMPROTIC KA210VET project team, this booklet has been created to meet this need of our students. This booklet will guide our students and they will realize their shortcomings in order to improve themselves. This booklet is based on the about industrial robots and their controls. A general narrative language is preferred. All robots can also be adapted. Coppeliasim Edu, RoboDK softwares has been chosen for simulationing the industrial robot systems. Robotic applications can be built in a software or simulated in the same software. Thus, all experiments can be done in the computer environment and can be seen as a contribution to the Digital Transformation in the Education system.

In addition, a section is devoted to software languages such as Python, which are used in robot programming. Thus, the software side of industrial robot systems is aimed to be shown.

The problems to be encountered in industrial robot applications are mentioned in the last section. Information to solve these problems is given in this section.

We are happy to share the knowledge and experience of the teachers who came together for the Erasmus+ project KA210VET - Using a Simulator for Programming in Robotic. We hope that the book will be useful to all technical staffand technical students.

ITO VAKFI S. TASTEKIN MTAL ZESPOL SZKOL NR.1 IIS A. VOLTA PESCARA APAGA GALICIA

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

INDUSTRIAL ROBOTS

Robotics is a multi-disciplinary modern science for general-purpose programmable machine systems consisting of a combination of fields such as machinery, electrical-electronics and computers in general. The first foundations of robotics in the world were laid by Al-Jazari, who lived between 1136-1208. However, the word robot was first derived by the Czech writer Karel Capek in 1922 from the Slavic word crobota, which means worker, slave, captive. The word robotic was first used by Isaac Asimov.

There is no doubt that the field where robots are used the most is industry. The main philosophy of the industrialists is to produce more by making the production line shift faster. It means a faster shift of the production line; it means speeding up the assembly process by replacing the few workers working along the production line with a single "mechanical robot".

The capacities of the robots, which used to stand next to the production line and assemble some parts with mechanical movements (to the bodies that slide on that line), have been increased by using the feedback process. These robots also control the part after assembling it, if there is a faulty operation, it pulls that part away from the production line. Sends it back to fix the faulty state. For this reason, in some places, the definition of "Feed-Back Path" is used instead of "Assembly Line".

Based on this, the definition of industrial robots; It can be made as a reprogrammable, versatile manipulator (arm) designed to move materials, tools or special parts for different tasks with programmed movements. (American Robotics Institute, RIA)

The most comprehensive definition of industrial robot and classification of robot types are determined in ISO 8373 standard. According to this standard, a robot is defined as: "It is an auto-controlled, reprogrammable multi-purpose manipulator with three or more programmable axes, which can be fixed or mobile, used in industrial applications."

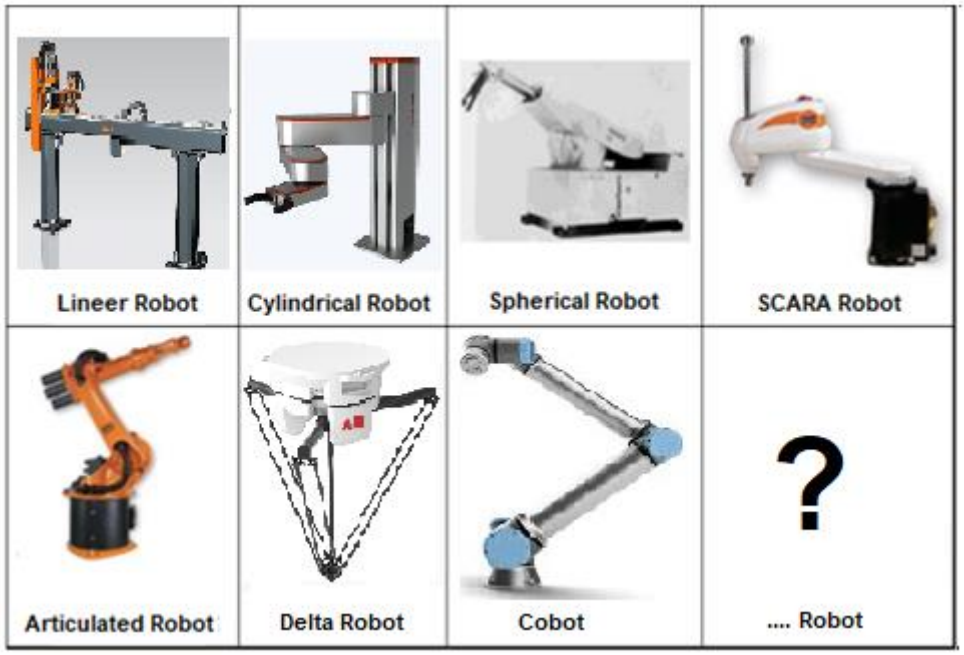


Figure 1. Examples of industrial robots

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

1. HISTORY OF INDUSTRIAL ROBOTS

In 1956, a company called Unimation (Universal Animation) was founded by George Devol and Joseph Engelberger. The first industrial robot was developed in 1959 as a result of the work done by the Unimation company. The first application of the Industrial Robot in the world was Unimate, which was integrated into a conveyor at General Motors by the Unimation company in 1961 and was tasked with picking up hot and heavy workpieces from a metal press machine and stacking them on pallets. With the technology at that time, the program of the industrial robot was recorded on a magnetic drum.

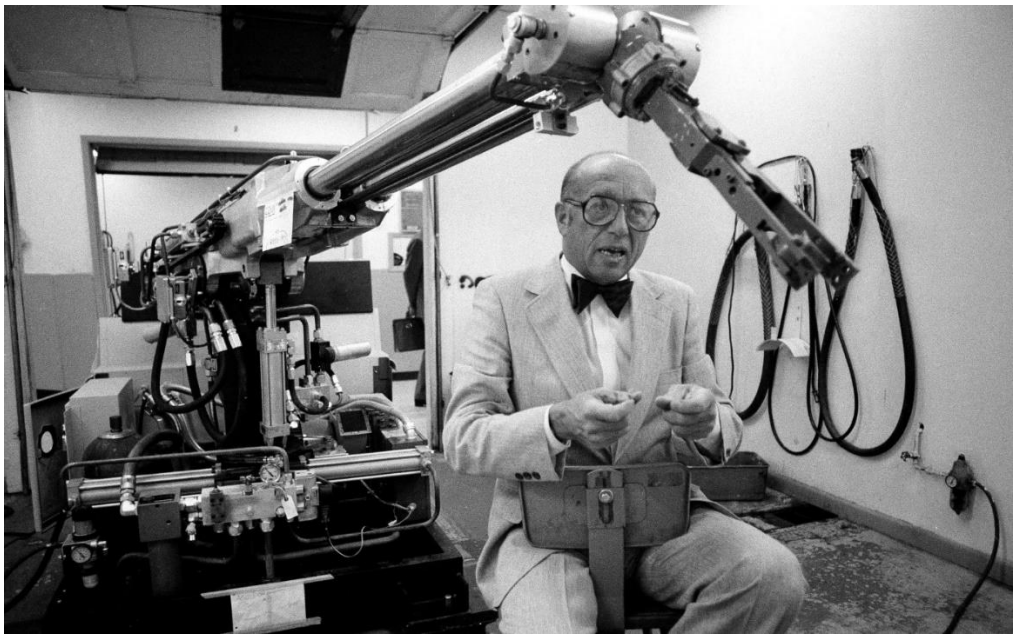


Figure 1.1. First industrial robot (Unimate, 1961) – Joseph F. Engelberger

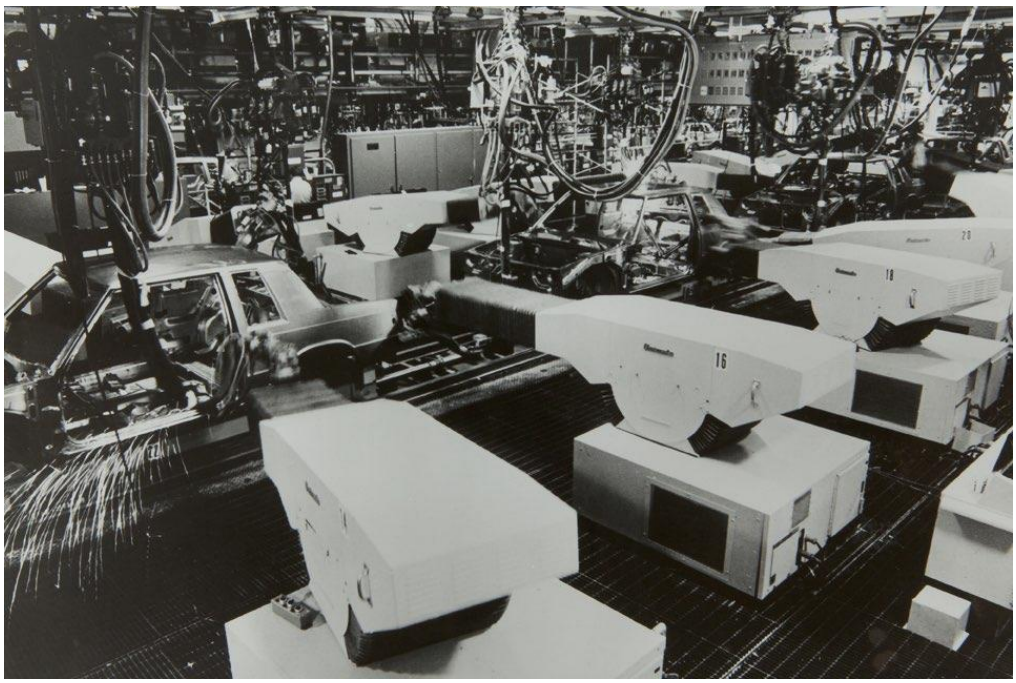


Figure 1.2. The Unimate performing spot-welding at a car factory in the United States.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

2. ADVANTAGES AND DISADVANTAGES OF INDUSTRIAL ROBOTS

The advantages of using Industrial Robots can be listed as follows:

• They can work in heavy and large jobs that force people's physical characteristics,

• Steps to workplace safety as they can work in unfavorable and dangerous conditions for human health,

• They maintain product quality standards thanks to their high precision and repeatability,

• By reducing the amount of defective production, waste of raw materials is prevented and they reduce the production cost,

• They can be easily adapted to a new job thanks to reprogramming,

• They can do more work by working continuously in monotonous, boring and tiring works without reducing the efficiency and product quality,

• They have remote access, management and control features,

• They can work together smoothly and quickly in the same environment,

• Workplace safety, health, education, insurance, etc. they save cheap labor with the reduction of expenses,

In addition to the advantages provided by the use of industrial robots, there may also be some disadvantages. These can be listed as follows:

• They may cause unemployment problems due to the cheapening of the labor force,

• They can cause unwanted harmful results in programming problems,

• A calculation error made in repetitive works can be reflected on all manufactured products.

Regarding the unemployment problems mentioned above; In fact, technological developments cause people to deal with tasks that force their minds, not their bodies, on the way to becoming an information society. Thus, in addition to the design, development and programming of new robots due to need, maintenance-repair and repair of robot and peripheral equipment, installation of robot production facilities, etc. creates new job opportunities

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

3. APPLICATION FIELDS OF INDUSTRIAL ROBOTS

The use of Industrial Robots in the industry is becoming more and more widespread day by day in many fields, especially in handling (handling: holding, carrying and releasing), welding-soldering, assembly-disassembly, painting and cutting.

Due to the advantages they have, industrial robots are widely used, especially in the automotive, electrical-electronics, chemistry, plastic machinery, metal, food-beverage sectors. The main fields of use of Industrial Robots can be listed as follows:

- Handling (material selection, transport, sorting, placing, etc.) applications,
- In Mounting and Disassembly applications,
- In Spot Welding, Arc Welding and Rotary Welding applications,
- In Adhesive/Sealants applications
- In material processing (Milling etc.) applications,
- Deburring, polishing, painting etc. in applications,
- In packaging, stock and loading applications,
- Casting, pressing, forging, etc. in applications,
- In measurement and control applications

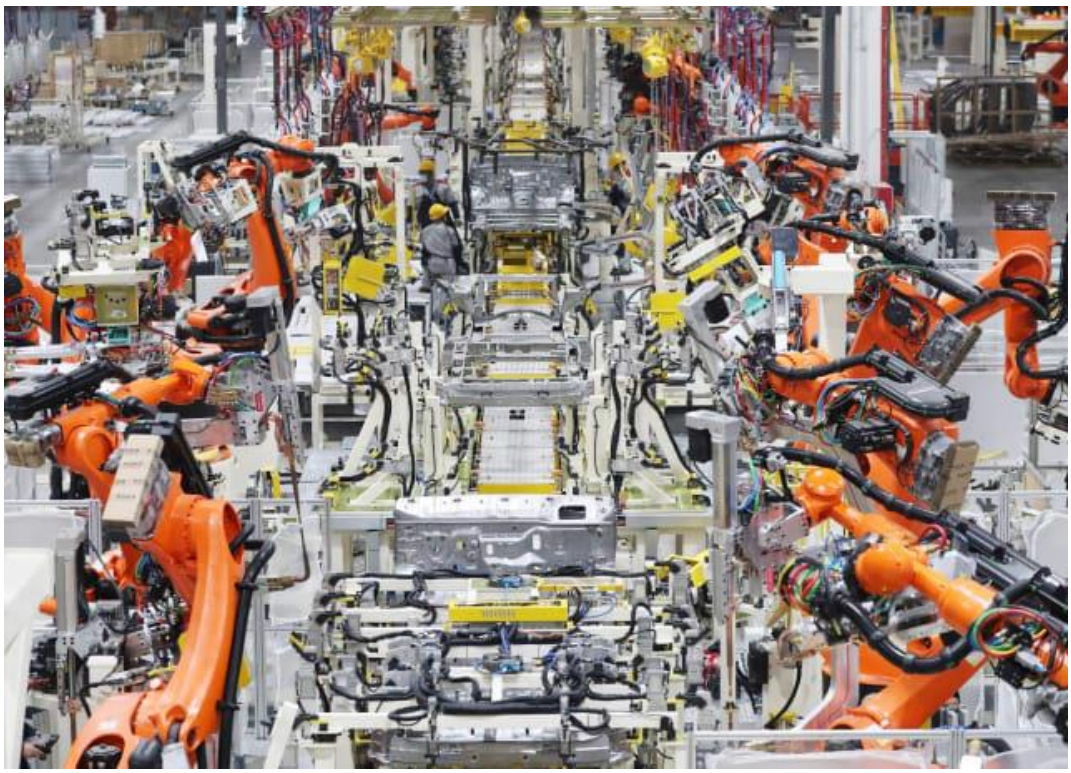


Figure 3.1. Industrial robots are at a car factory.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

4. STRUCTURE OF INDUSTRIAL ROBOTS

The main parts of a robot are the manipulator, power supply and controller. Manipulator; It is used to remove parts, materials and tools necessary for production. The power supply is used to move the manipulator. The controller controls the power supply. Thus, the manipulator arranges its own task.

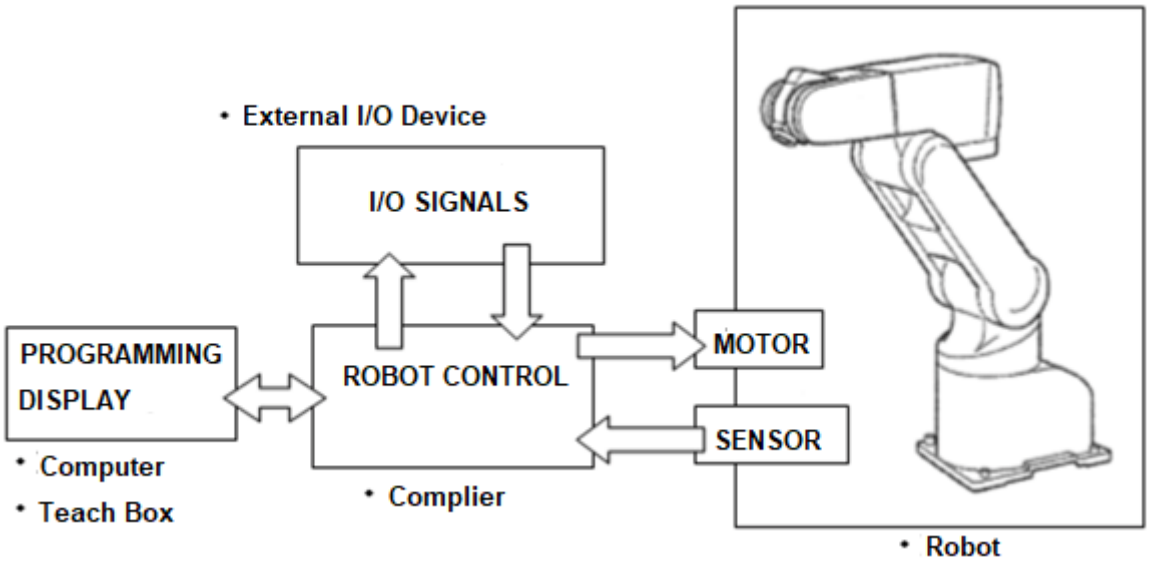


Figure 4.1. Robotic system example

Industrial Robot System, also known as Robot Cell, is basically a hardware and software based system, as seen in Figure 4.2.

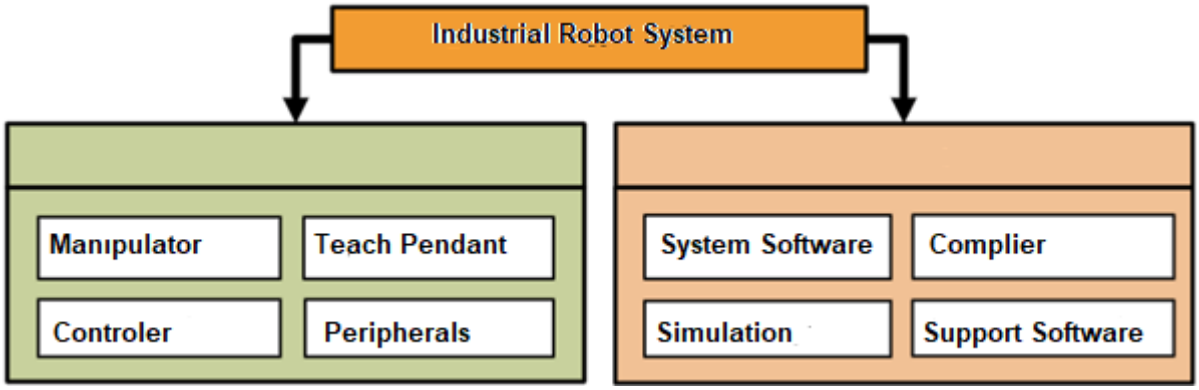


Figure 4.2. Industrial Robot System

4.1. Mechanical Structure

It includes the main body or column, the mechanical arms and the instruments placed inside.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

4.1.1. Manipulator

In general, when the manipulator is mentioned, the robot arm that creates the actual mechanical order comes to mind. The manipulator can be expressed as the robot arm formed by the mechanical and electronic parts, together with a large number of interconnected moving parts that form the kinematic chain of the robot and enable it to move in the direction of its axes.

4.1.2. Instruments/Effectors

Grippers, measuring sensorsand instruments used in robotics are defined as effectors. In addition, other processing elements that move in the working area according to the program and serve the robot to manipulate the robot's environment are also included in this definition.

4.1.3. Working Envelopes of industrial robots

The Working Envelope (Working Volume - Access Space) describes the space that covers all the points that the manipulator can reach in its environment depending on its mechanical movement capability. The Working Envelope of the manipulator changes depending on the axes and degrees of freedom in the design of the robot. A robot's Working Envelope is critical to its interactions with other machines and systems.

Joints play a major role in determining the working area of the industrial robot. Thanks to the joints, the robot gains the ability to move in many directions. Therefore, the mobility is directly related to the determination of the working area of the robot. In the design of industrial robots, two basic joint types are generally used, namely Revolute and Prismatic. Also in the industry cylindrical, spherical, screw, etc. A variety of joint types are also available.

Each joint of a robot has a limited range of motion. The axis formed by the first 3 joints of the Industrial Robot and providing the determination of the wrist position is called the Major Axis, and the axis formed by the next 3 joints and allowing to determine the direction of the hand is called the Minor Axis. The Working Envelope is determined depending on the Major axis hand structure of the Industrial Robot.

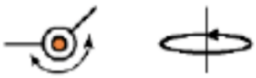
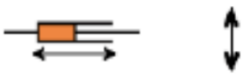
Joint Type	**Symbolic Representation**		**Joint Description**
	Literal	Figural	
Rotational – Revolute	R		Rotational motion around the axis
Prismatic – Translational	P (veya T)		Linear motion along the axis

Table 4.1. Robot joint types

Funded by the European Union

4.1.4. Axis numbers and properties

In the industrial robot system, the manipulator's mobility changes depending on the number of axes and the characteristics of a manipulator. Table 4.1 shows the manipulator axis types and their properties.

Axis	Type	Feature	Description
1-3	Major Axis	Determine wrist position	The working envelope of the industrial robot is determined
3-6	Minor Axis	Tool orientation determination	The tool at the tip of the manipulator is guided in 3D space.
7-n	Redundant Axis	Avoiding obstacles	Allows the manipulator to avoid unwanted areas or access around obstacles in the workspace.

Table 4.2. Industrial robot axes

The robot has to fulfill the task given to it, just like a working human. In order to perform this task, a robot must have the levers in his bones and the bending and twisting systems in his muscles.

The direction of industrial robots is measured by Roll, Pitch and Yaw. Human and robot wrists have similar mobility. Although it varies from person to person, the mobility of the human wrist is shown in Table 4.3. In Picture 4.1, Row (Rotation), Pitch and Yaw movements for the wrist (minor axis) of an industrial robot are seen.

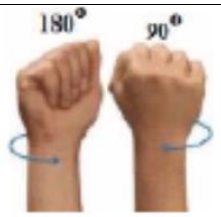
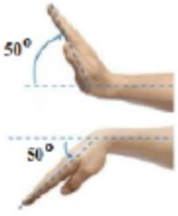
Movement Type	Description	Exemplary ability
ROLL	Moving the wrist clockwise and counterclockwise with the right arm straight and palm down in the 0° position	 ROLL=180°+90°
PITCH	Moving the wrist up and down with the right arm straight and palm down in the 0° position.	 PITCH=50°+50°
YAW	Moving the wrist left and right with the right arm straight and palm down in the 0° position	 YAW=20°+45°

Table 4.3. Mobility of the human hand

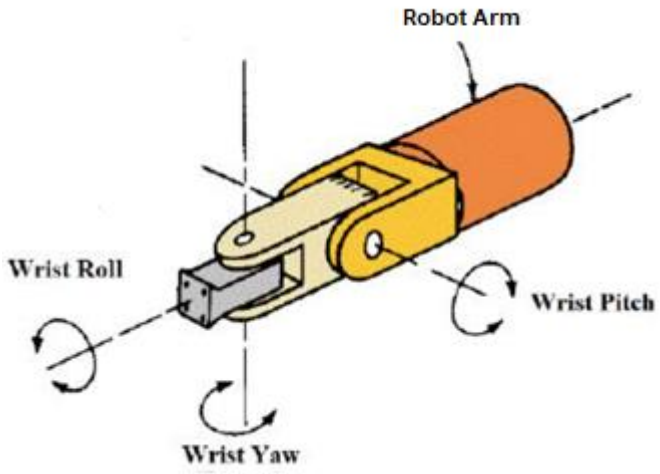


Figure 4.3. Robot wrist movements (minor axis)

The second part of the human arm has two links with three degrees of freedom; two degrees of freedom shoulder and one degree of freedom elbow. But the robot has one shoulder with one degree of freedom. The waist of the robot takes over the second movement of the human shoulder.

There are five fingers at the end of the arm. If the fingers squeeze an object, the finger joints are not independent and do not affect the position and direction of the object being held. If the fingers are used singularly and independently, they provide mobility.

Thumb, first and middle fingers are used in handling robots (pick and place type).

One of the important features of the arm structure is the ratio of the upper arm to the forearm. This is around 1:1 to 1:2. This ratio means that the robot's forearm is equal to or shorter than the upper arm. If this ratio is not met, an irregularity will occur in the movement of the robot. When evaluated mechanically, it is seen that the human arm is a hierarchical structure consisting of linear and non-linear elements.

Degree of Freedom (DOF): The number of independent movements an object can make is the number of degrees of freedom. A free body has six degrees of freedom when it moves freely in space. Three of them are for "place" and the other three are for "orientation".

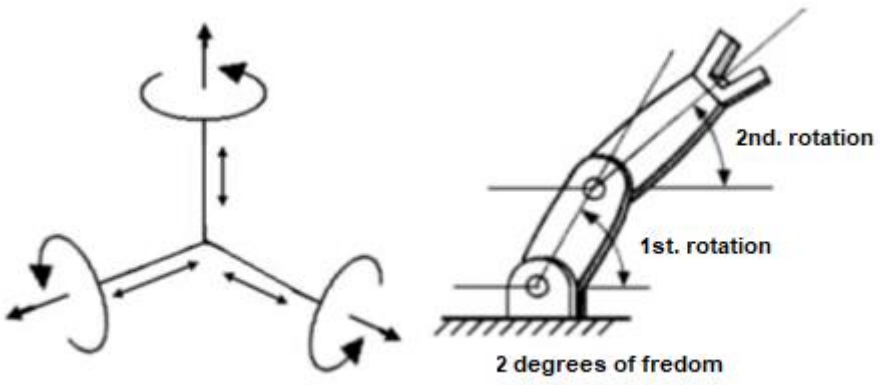


Figure 4.4. degrees of freedom

As the number of joints in the arm decreases, the working space of the arm still decreases in volume even if the physical dimensions of the fittings remain the same, and the flexibility of the arm to reach any point in this space decreases.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

When some operations require this flexibility to be high, it is necessary to choose a high degree of freedom of the arm. In such cases, arm structures with more than six, nine or ten joints are used. Increasing the degrees of freedom will increase the cost of the robot arm. However, it is possible to find the movements in the joints of robot arms that are similar to the movements in the waist, shoulders, elbows, wrists and fingers of humans in the ordinary structure.

In addition, even if the efficiency increases in robots with more than six joints, it causes programming difficulties in the calculation of coordinate transformations.

Figure 4.5. DOF

4.2. Control System

It consists of a collection of digital electronic circuits.

4.2.1. Energy Supply Methods

In general, two methods are used to provide the necessary energy to the robots and transmit the signals:

• External energy supply; The energy required in this method is provided by means of hoses or cable packages, independently of the robot, over arms or similar devices. Care must be taken to make a proper connection so that the hose or cable packages are not damaged due to robot movement.

• Integrated energy supply to the robot; The energy required in this method is provided from the robot building groups.

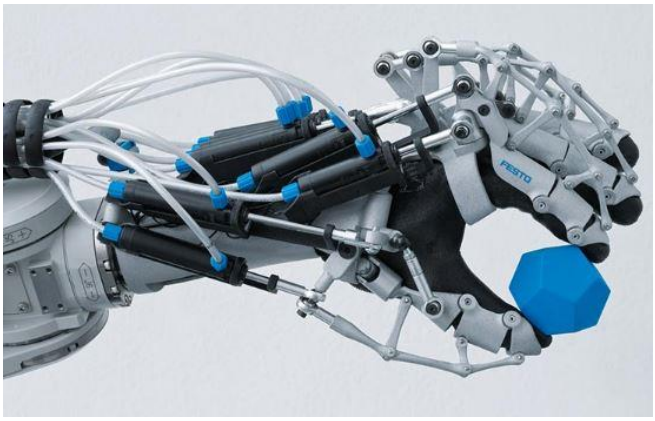


Figure 4.6. Example of external air source

4.2.2. Input and output numbers and features

There are three types of input and output connections: Emergency Stop Button entry and exit point, Robot arm enclosure cabinet door sensor entry point and parallel input and output unit.

The number of inputs and outputs at the Parallel Input Output port varies according to the type of robot used. Some robots have 16 inputs and 16 outputs, and some have 32 inputs and 32 outputs.

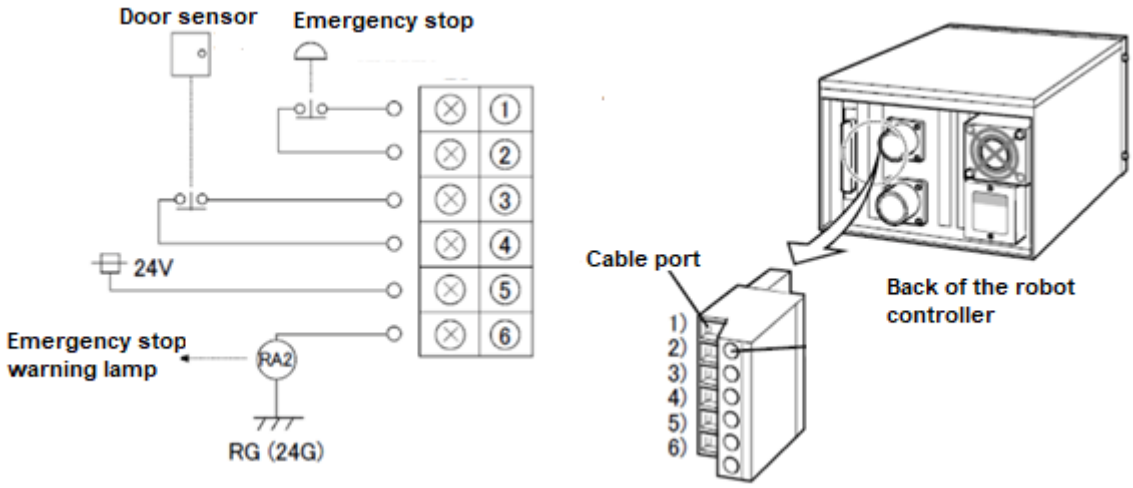


Figure 4.7. Robot input/output connections (KUKA)

4.2.3. Peripherals Connections

Various methods are used to establish communication between the industrial robot and its peripherals. These;

• Integrated inputs/outputs

• Bus systems

• It is Ethernet.

4.3. Power Unit

The drive systems used to provide the necessary power for robot manipulator joint movements are as follows:

• Electrical Drive Systems

• Hydraulic Drive Systems

• Pneumatic Drive Systems

Industrial robots with hydraulic drive systems are generally used in heavy industry, but also in molten steel processing, auto part, etc. They provide high speed and strength for large loads. The robot needs to be connected to the base. Although hydraulic drives are large and bulky, they can cause noise, oil leakage and cleaning problems. Although they can produce high torque at low power, they are difficult to control because their performance is not linear.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

Today, DC servo motors and stepper motor drivers are used for most robot manipulators. Electric drive systems are clean, but they are better in accuracy and repeatability. However, electric drive systems are both slower and less powerful than hydraulic drives. The robot needs to be connected to the base. Electric motors can be made more powerful and sensitive with the help of reducers. The fact that DC servo motors produce high torque at low power has been an important reason for preference. Stepper Motors are generally used in simpler applications such as holding, carrying and placing that do not require high torque.

Pneumatic drive systems are used especially for small robots with several degrees of freedom (DOF). They usually store quick execution of simple operations such as Hold-Release. Although the energy efficiency of pneumatic drive systems is better, feedback control is difficult. In addition, the control process becomes difficult due to the shortage of air pressure equipment that will quickly eliminate the inertia of the moving robot pistons. For this reason, they are generally preferred in simple applications. Generally, manipulator effectors are pneumatic.

4.4. Sensors

Sensors in robots are used in very wide areas. These are generally;

- Touch sensors,
- Proximity and rate sensors,
- Various sensors and basic sensor systems,
- They are grouped as automatic vision systems.

4.4.1. Touch Sensors

Touch sensors are devices that show the connection between some solid objects and themselves. We can divide these sensors into two classes: They are contact and force sensors. Contact sensors give out connections between objects as binary output signals. Force sensors (sometimes called tension sensors) also only show the magnitude of the connecting force between objects.

<u>Contact Sensors</u>

Contact sensors are used to show whether there is a connection between two objects, regardless of the magnitude of the connection force. This category includes simple devices such as limit switches, microswitches etc. takes place. Contact sensors are periodically used in the docking systems of robots. It can be used, for example, to show the acceleration of a point along conveyors. Figure 44 shows a robot system with contact sensors.

Another place where touch sensing is used is in material surface probe controls. A robot with 6 degrees of freedom has the capacity to reach part surfaces. Machine measurements in triaxial coordinates are difficult.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

Force Sensors

The measurable force capacity is as much as the number of tasks assigned to the robot allows. This capacity includes things like handling different sized items, loading machinery, setting up jobs, applying an appropriate level of force.

Force sensing in robots can be achieved in many ways. The generally used technique is the force sensing crankpin technique. It consists of a load cell mounted between the wrist and the gripper. Another technique is to measure the moments that occur in each connection. This is usually achieved by sensing the motor currents at all motor connections.

4.4.2. Proximity and Rate Sensors

Proximity sensors are activated when one object obscures another. It examines how the object is closed and makes it work uniquely. The detection distance can be from a few millimeters to 15-20 cm. Some of these sensors measure the distance between the object and the sensors. These are also called ratio sensors. Proximity and rate sensors are placed in the hand and end effectors, which are the moving parts of the robot. A practical use of proximity sensors in robots is to detect the presence or absence of workpieces or other objects. Ratio sensors are used to determine the position of the object associated with the robots. It is suitable for the construction of high-tech proximity and rate sensors. These; optical materials, acoustics, electric field and others.

4.4.3. Miscellaneous Sensors and Basic Sensor Systems

These miscellaneous categories include other types of sensors and transducers. These are devices that can detect changes in the hearing ability of working areas in robots. These hearing abilities include temperature, force, fluid flow, and electrical properties.

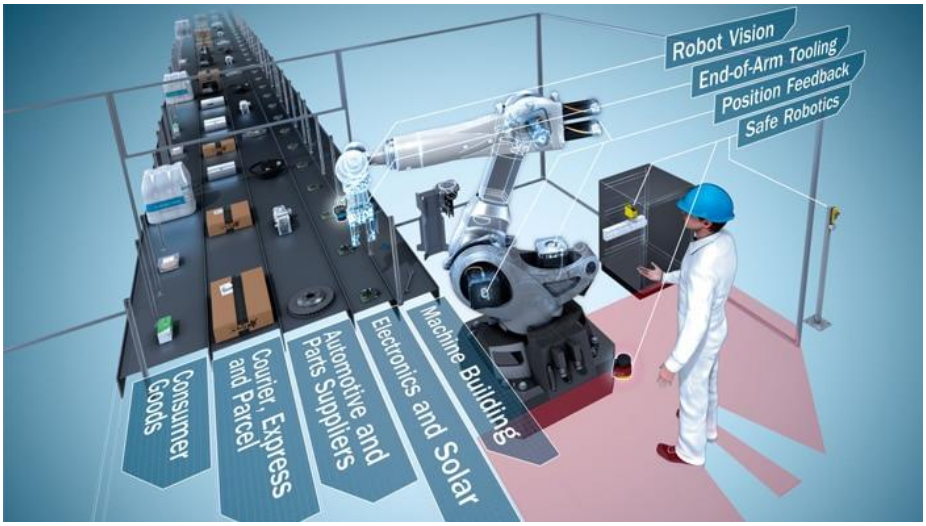


Figure 4.8. Sensor Applications

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

5. CLASSIFICATION OF INDUSTRIAL ROBOTS

Industrial robots; Although they have different structures and especially technology from past to present, they can be classified as follows in general terms:

- Cartesian Robots
- Cylindrical Robots
- Sphere Robots
- SCARA Robots
- Articulated Robots
- Delta Robots

5.1. Cartesian Robots

The Cartesian Robot is the simplest robot with all 3 major axes prismatic (PPP). All robot movements take place at right angles to each other. In Cartesian robots, moving parts move parallel to X, Y and Z cartesian coordinate system axes. Cartesian robots have the robot design with the most limited freedom of movement. The Working Space Envelope of Cartesian Robots is in the form of a rectangular prism. The features and application example for Cartesian robots can be seen in Picture 5.1. Cartesian robots can be mounted on the floor or ceiling for material handling or surface work. Cartesian robots are especially used in the assembly, transportation and processing of materials such as marble, glass and wood.

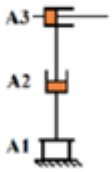
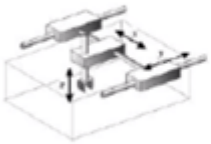
Major Axis Type	Kinematic Structure	Working Area	Application Sample
P-P-P (3P)	 cartesic coordinates	 prismatic	 Lineer Robot

Figure 5.1. Cartesian robots

5.2. Cylindrical Robots

The Cylindrical Robot is a robot with the first joint Revolute (R) and the second joint Prismatic (P) (RPP). In cylindrical robots, the robot arm moves in the form of a cylinder or a cylindrical part. The Working Envelope of Cylindrical Robots is in the form of a cylindrical piece. Features and application example for cylindrical robots can be seen in Picture 5.2.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

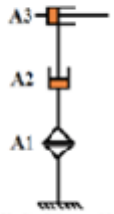
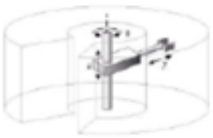
Major Axis Type	Kinematic Structure	Working Area	Application Sample
R-P-P (R2P)	 cylindrical coordinates	 cylindrical	

Figure 5.2. Cylindrical robots

5.3. Syphere Robots

The Spherical Robot is a robot with the first 2 joints Revolute (R) and the 3rd joint Prismatic (P) (RRP). The Working Envelope of Spherical Robots is spherical. Features and application example for spherical robots are shown in Picture 5.3.

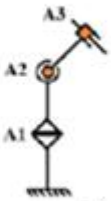
Major Axis Type	Kinematic Structure	Working Area	Application Sample
R-R-P (2RP)	 sphere coordinates	 spherical	

Figure 5.3. Syphere Robots

5.4. SCARA Robots

SCARA (Selective Compliance Assembly Robot Ann) Robot, with the first 2 joints Revolute (R) and the 3rd joint Prismatic (P), such as Robot, Spherical Robot (RRP), or the first 3 axis Revolute (R) and the 4th axis Prismatic (P) (RRRP) robot. The Revolute (R) joints of the SCARA Robot move horizontally. SCARA robots have important features in terms of accuracy, high speed and easy assembly. The features and application example for SCARA robots are shown in Picture 5.4.

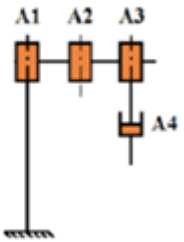
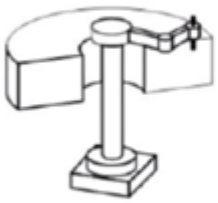
Major Axis Type	Kinematic Structure	Working Area	Application Sample
R-R-R-P (3RP)	 joint coordinates	 Seçici Uyumluluk Montaj Robot Kolu	

Figure 5.4. SCARA robots

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

5.5. Vertically Articulated Robots

Artlenlated Robots are (RRR) robots with all three of their major joints revolute (R), resembling human arm anatomy. Vertical Articulated robots are also called Anthropomorphic or Revolute Robots. The Working Area of Vertical Articulated Robots is exactly like the land. Articulated robots are more capable robots due to their mobility. Articulated robots are widely used, especially in the fields of welding and painting. The features and application example for vertically articulated robots are shown in Picture 5.5.

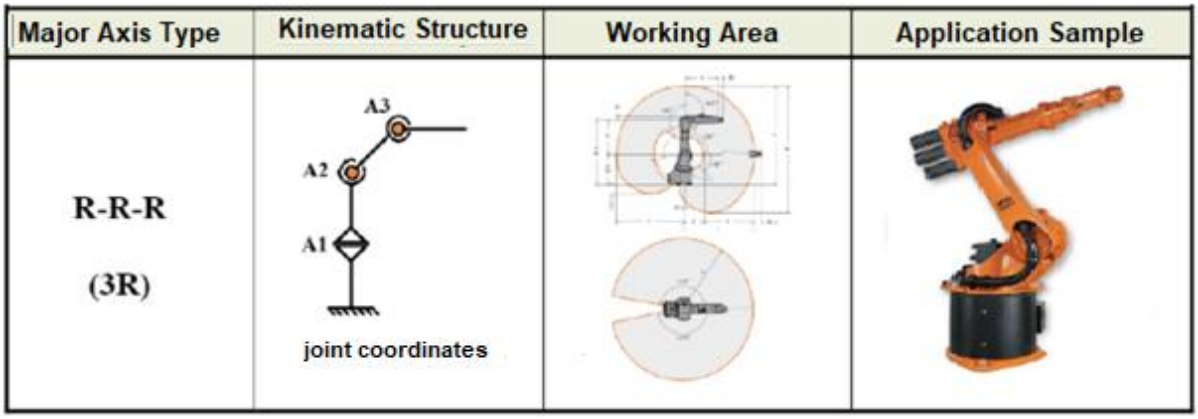


Figure 5.5. Vertical articulated robots

5.6. Delta Robots

A delta robot is a type of parallel robot[2] that consists of three arms connected to universal joints at the base. The key design feature is the use of parallelograms in the arms, which maintains the orientation of the end effector.[3]

Delta robots have popular usage in picking and packaging in factories because they can be quite fast, some executing up to 300 picks per minute

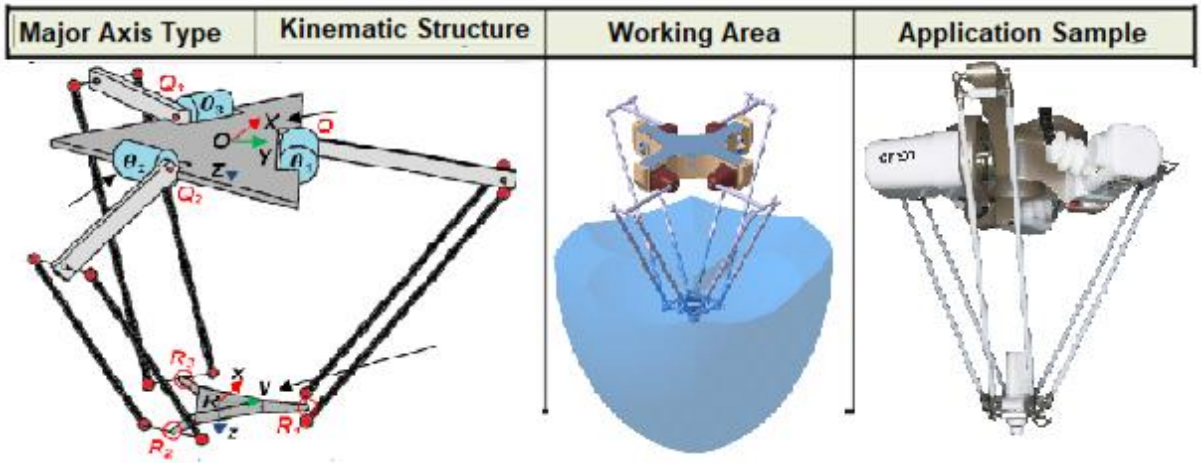


Figure 5.6. Delta robots

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

6. PERFORMANCE METHOD OF INDUSTRIAL ROBOTS

Various factors such as precision, speed, load carrying capacity, reaction time, stability are used in determining the performance of industrial robots.

6.1. Precision

Precision is defined as the smallest amount of change that can be measured. The motion sensitivity of industrial robots is one of the most important performance indicators of the robot. One of the performance criteria of the robot is precision; It is defined as a function of accuracy, resolution, and repeatability.

6.2. Accuracy

Accuracy is expressed by how close the measurements made are to the true value. In this respect, measurement and measurement error play an important role in expressing accuracy. Measuring is the process of comparing an unknown quantity with a known quantity in its own terms and accepted as a unit. The measurement error is the difference between the value obtained as a result of the measurement and the actual value. A robot's accuracy is its ability to position the robot's TCP at any point within the workspace envelope.

6.3. Resolution

Resolution is the smallest input change interval value that produces an observable change in output value. In industrial robots, resolution is related to the range of motion of the axes. In this way, as the axis range of motion decreases, the resolution of the robot increases inversely.

6.4. Repeatability

Repeatability is defined as the ability to give the same output in repeated applications of the same input value under the same conditions. The repeatability of the industrial robot is defined as the robot's ability to repeatedly position the TCP to a previously taught point within the workspace envelope. In this way, the repeatability of an industrial robot is determined by the maximum amount of error between the robot TCP and the taught point as a result of the robot's repeated movements.

6.5. Reaction Time

Response Time is the time required to obtain a noticeable change in the output of the system in response to a change in input. For Industrial Robots, the reaction time is expressed as the ability of the robot to move to the next state in a short time in relation to its movement speed. It is desirable that industrial robots preferably have a fast reaction time.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

6.6. Stability

Stability is the ability to measure a fixed input to give the same output over a period of time. For Industrial Robots, stability is generally expressed as the measure of the oscillations of the robot arm during the movement from one position to the other. An industrial robot with good stability should not oscillate at all during movement.

6.7. Load Carrying Capability and Speed

Payload and speed capacities of industrial robots; The system design of the robots varies depending on the technological elements such as the size, coordinate and drive systems, as well as the size and shape of the transported materials. For Industrial Robots, Maximum and Nominal load carrying capacities usually come to the fore:

Maximum Load Carrying Capacity; It is expressed as the maximum load value that a robot can carry within the limits of repeatability at minimum speed.

Nominal Load Carrying Capacity: It is expressed as the maximum load that a robot can carry while at maximum speed, within the limits of repeatability.

Industrial Robot speed describes the time it takes to complete a given business cycle. The fact that the Industrial Robot is fast means that the desired work is done in a shorter time.

! In the next section, we will describe the collaborative robots that have been introduced in recent years. Let's take a look at the differences between them for preparation.

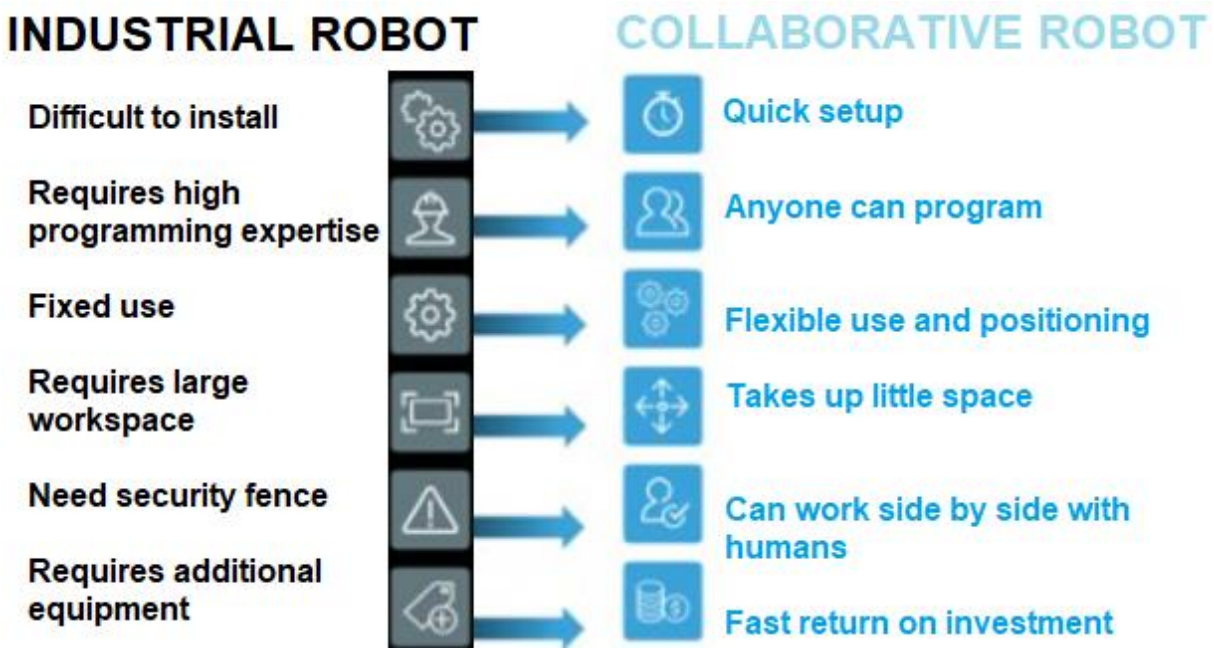


Figure 6.1. Robot vs Cobot comparison chart

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

COBOTS (COLLABORATIVE ROBOTS)

7. WHAT IS COBOT?

Collaborative robots, also known as Cobots, are robots that enable them to work safely by interacting side by side with people in a collaborative space. The word cobot, which first appeared in 1999, is a combination of the English words collaboration and robot.

The most important and distinguishing feature that distinguishes cobots from robots is their ability to work side by side with humans in interaction and in confidence. In shortly, Cobot is an industrial robot arm that can work side by side without the need for any safety barriers.

The International Federation of Robotics (IFR) defines four types of collaborative manufacturing practices:

Coexistence: Human and robot work together without a shared workspace.

Sequential collaboration: The human and robot share all or part of a workspace, but do not work on a part or machine at the same time.

Collaboration: Robot and human working simultaneously on the same part or machine and both are in motion.

Responsive collaboration: The robot responds to employee movement in real time.

Most cobots today share space with humans in industrial applications, but complete tasks independently or sequentially (coexistence or sequential cooperation). Collaboration or responsive collaboration is less common now. In the publication published by the International Federation Robotics (IFR) in 2018, this situation is shown in the figure below.

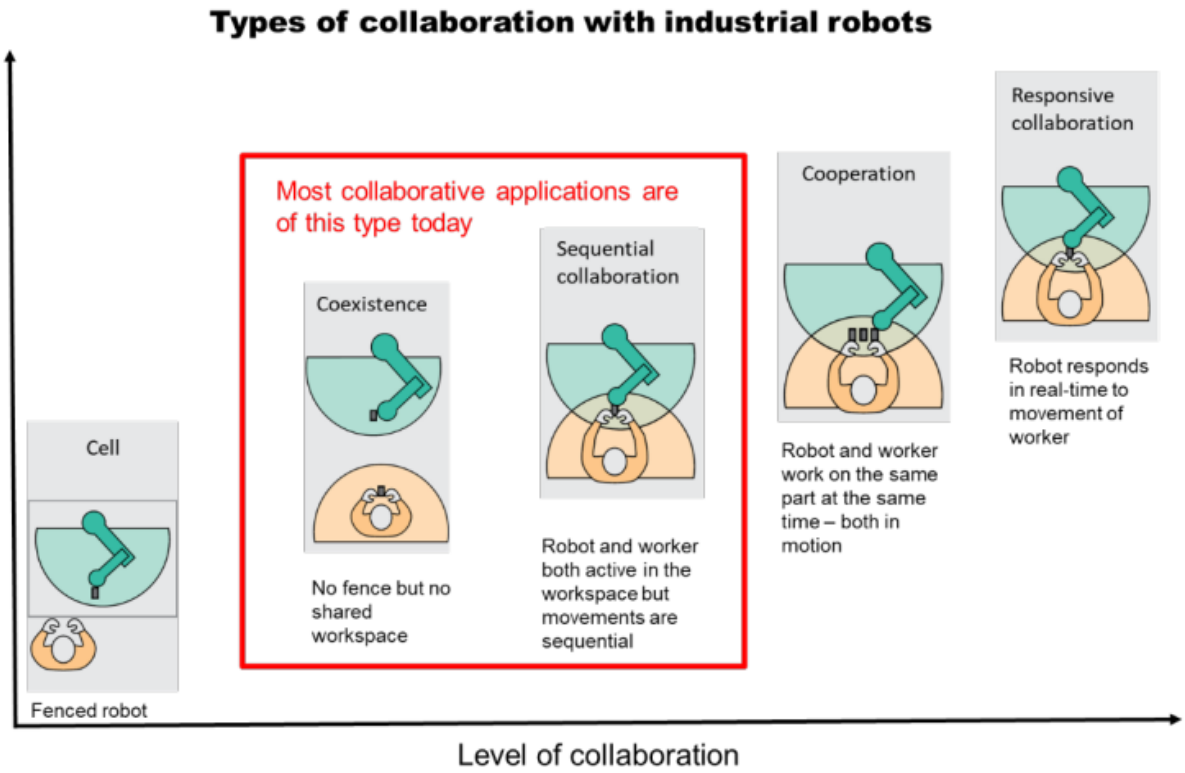


Figure 7.1. Collaborations Level Chart

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

8. HISTORY OF COBOTS

Cobots were invented in 1996 by J. Edward Colgate and Michael Peshkin,[8] professors at Northwestern University. Their United States patent entitled, "Cobots"[9] describes "an apparatus and method for direct physical interaction between a person and a general purpose manipulator controlled by a computer." Founded by these professors, Cobotics company produced various cobots until 2003.

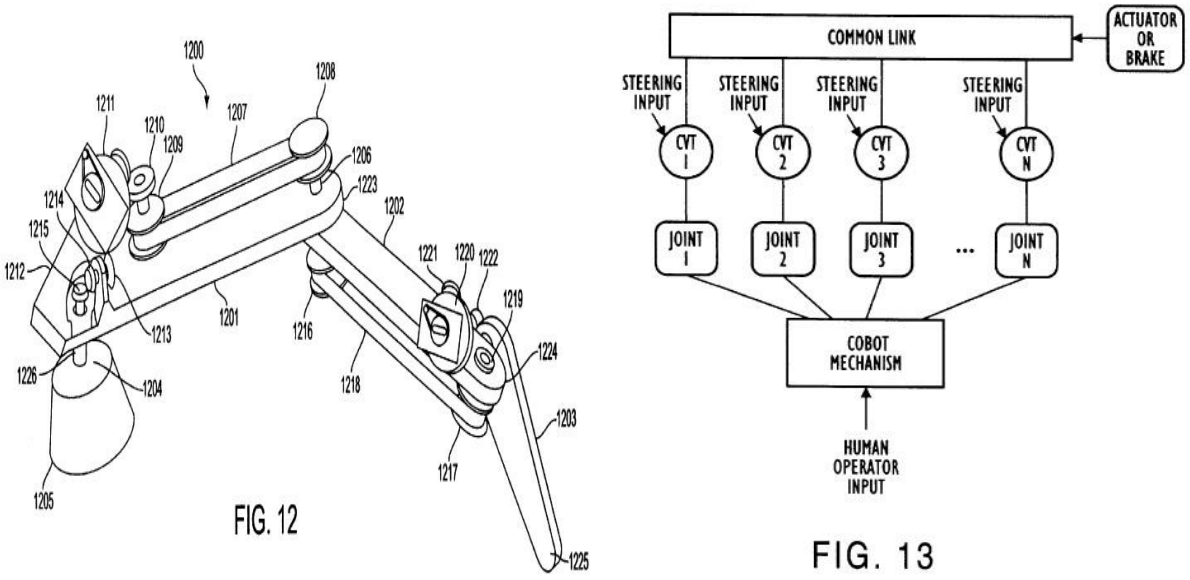


Figure 8.1. Articulated arm and architectural diagrams in the patent application of the first cobot in 1999.

KUKA company launched the first cobot in 2004, and Universal Robots company launched its first cobot in 2008. FANUC and ABB companies produced their first cobots in 2015.

According to the research of Grand View Research, the companies in the cobot market are as follows;

ABB Group DENSO Robotics Epson Robots EnergidTechnologies Corporation

F&P Robotics AG Fanuc Corporation KUKA AG MRK-Systeme GmbH

Precise Automation, Inc Rethink Robotics, Inc Robert Bosch GmbH

Universal Robots A/S Yaskawa Electric Corporation MABI Robotic AG

Techman Robot Inc. Franks Emika Gmbh AUBO Robotics Comau S.p.A.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

9. FEATURES OF COBOTS

9.1. Technical Specifications Usage Features

Generally speaking, cobots are extremely light compared to their carrying capacity. The weight of a robot carrying 10 kg is around 150 kg. But at the same payload, the cobot weighs only 30 kg. The supply voltage is the home user city mains system. This means AC 220 volts for Europe.

Cobots are 6-axis and have 360-degree rotation on all axes. Some important technical specifications of the cobots are given below.

Payload: It is the maximum payload that the collaborative robot can carry. It starts from 0.5 kg and goes up to 35 kg.

Reach (Maximum reach): It is the furthest distance that the collaborative robot can reach with its arms. It starts from 250 mm and goes up to 1813 mm.

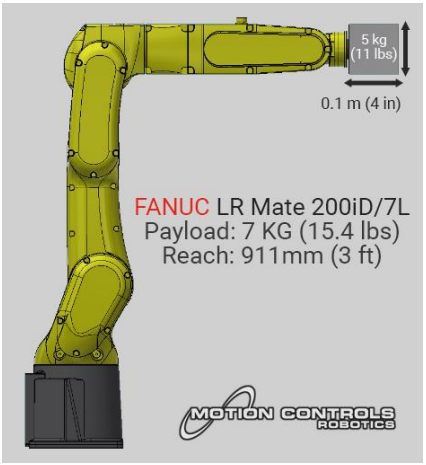


Figure 9.1. Payload and maximum reach data of Fanuc cobot.

DoF – Degree of Freedom: Indicates how many axes the collaborative robot can rotate and/or move. As this increases, the robot can make more flexible movements, and its ability to reach hard-to-reach points increases. It starts from 4 and goes up to 14.

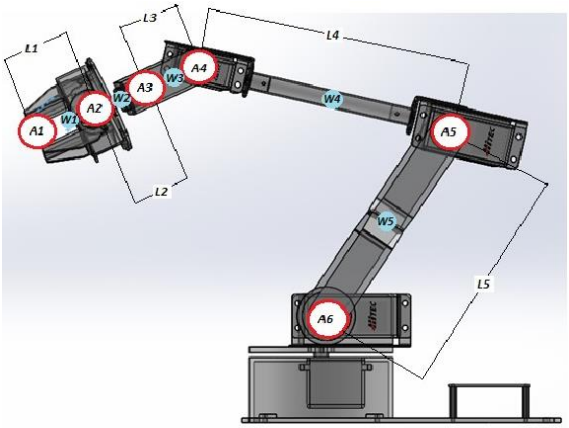


Figure 1:

Figure 9.2. Design of 5 DoF cobotic arm

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

Repeatability: An indication of how accurately the collaborative robot does the repeated work.

For example, we want him to constantly paint the center point of the dartboard. If the robot constantly goes to that exact spot, its repeatability is flawless. However, this is almost impossible. Therefore, the cobot can go up and down, like 0.01 mm or 0.1 mm, from exactly where it needs to go. This maximum deviation value indicates how uniformly (repeatably) the cobot can perform the operation. The higher the value, the lower the repeatability, which is often undesirable depending on the operation. This value is 0.01 mm today.

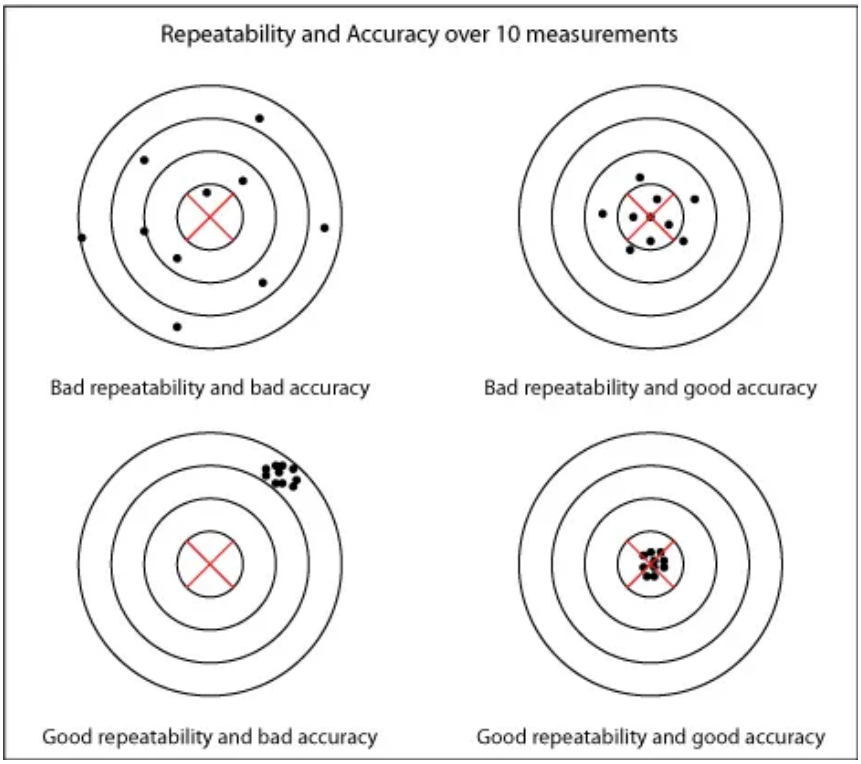


Figure 9.3. Codiac for Co-Packers cobot repeatability and accuracy testing

9.2. Usage Features of Cobots

The usage features of the cobots are briefly as follows;

Easy programming for manufacturers not found in traditional robots

Flexible and fast installation process

Use in a wide variety of fields

Ability to be installed in small-scale production

Collaborative safe working

No private protected work cells

Small footprint, low energy consumption

Low cost

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

10. COBOT APPLICATIONS

In manufacturing, there are business models or units that require attention and are frequently repeated. People in these units are more likely to make mistakes. While this increases work accidents, it also reduces productivity.

'Cobot', which is based on cooperation with people, overcomes these problems and raises occupational safety to the highest levels and increases productivity in production.

Today, cobots are used in more than 50 countries of the world. It can also be used in many processes, thanks to its high-precision arms and easy programming;

trimming-polishing,

machine feeding,

quality control,

transport,

assembly,

sticking-spreading,

bolting,

sanding,

take-drop,

packaging-palletizing,

labeling,

injection molding,

CNC,

welding,

laboratory analysis,

testing and sampling.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY

Funded by
the European Union

11. COBOT SYSTEM STRUCTURE

Cobots stand out with their collaborative technology, their approach that puts people at the center of the production process, and their design to integrate into any production facility and any application.

Cobots are prepared with the concept of Plug and Play and Start Production. Therefore, they are extremely easy to install, operate and program. A standard cobot system structure is shown below.

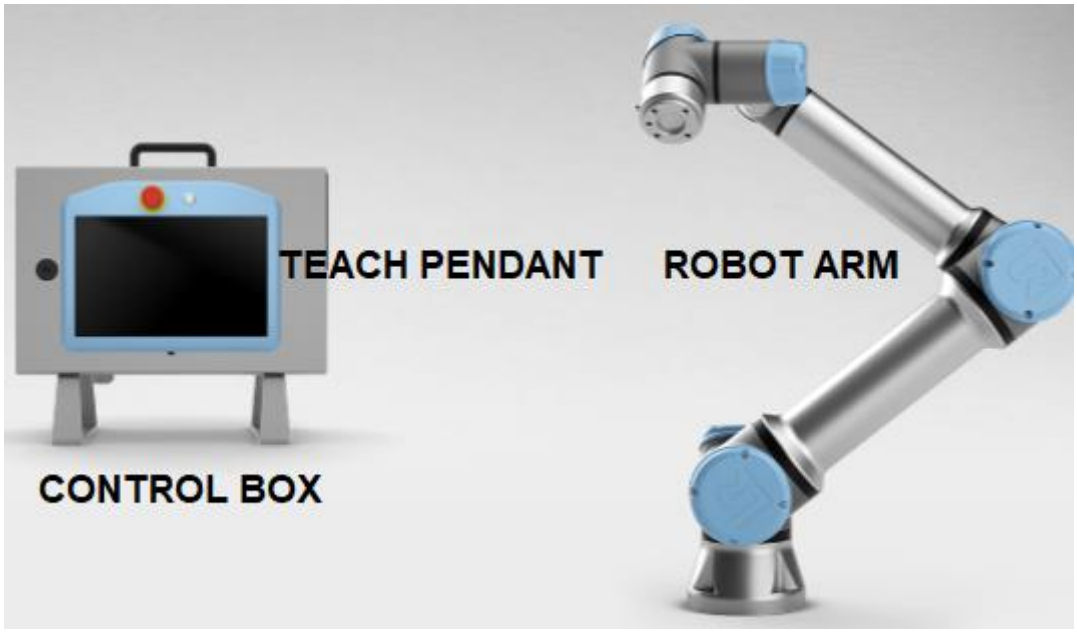


Figure 11.1. UR Cobot system

First of all, there is a 6-axis robot arm in the cobot system. 360 degree rotation is possible on all 6 axes. Their definitions are given in the picture below.

Figure 11.2. UR Cobot axis definitions

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

In the Cobot system, there is a "Control Box" besides the robot arm. This control box contains the motherboard, memory card and security control cards. All inputs and outputs are connected here. This control box provides the connection with all peripheral equipment. For example teach pendant, switches, sensors, conveyors, cameras etc.

The control box is a Linux-based microprocessor system. It communicates with each cobot company's own special software. The circuit elements in its content usually have plug-and-play connection points.

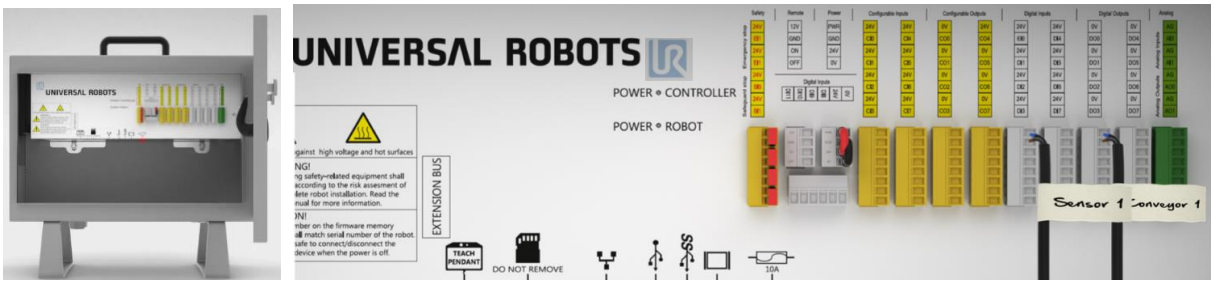


Figure 11.3. UR Cobot control box

The teach pendant is required for programming the cobot. Starting and stopping the cobot is done through this teach pendant. At the same time, programming functions such as departure points, motion patterns, waiting times, entry and exit operations are performed through this teach pendant. Security warnings are also monitored from this teach pendant.

Figure 11.4. UR Cobot teach pendant

As a result, the cobot system structure produced by each manufacturer is different. It includes different robot arm, different teach pendant panel and different control box. The software used may differ from each other. But the basic system structure and basic programming logic are the same in all of them.

! In the next section, we will describe the simulation software with which you can program all brands of cobots. Let's take a look at the these softwares for preparation.

CoppeliaSim
from the creators of V-REP
RoboDK
TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

SIMULATOR – RoboDK

12. INTRODUCTION OF THE SOFTWARE

RoboDK is a powerful and cost-effective simulator for industrial robots and robot programming. Founded by Albert Nubiola in January 2015, RoboDK Robot Development Kit is a spin-off company from the CoRo laboratory at ETS University in Montreal, Canada, one of the most prestigious robotics labs in Canada.

You can simulate any industrial robot with RoboDK. You can generate robot programs for any robot controller directly from your PC. No programming skills are required with RoboDK's intuitive interface. You can easily program any robot offline with just a few clicks. RoboDK has an extensive library with over 800 robot arms.

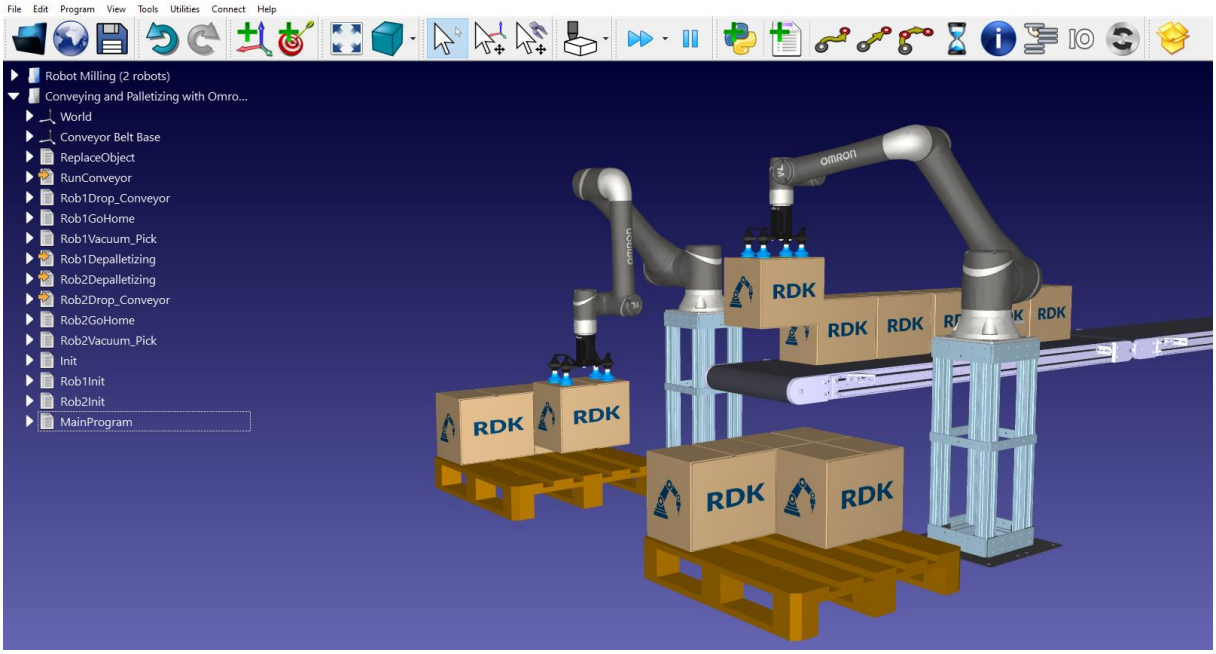


Figure 12.1. General view of the program

RoboDK key benefits:

- The advantage of using RoboDK's simulation and offline programming tools is that it allows you to program robots outside the production environment.
- With RoboDK you can program robots directly from your computer and eliminate production downtime caused by shop floor programming.

RoboDK technical specifications:

- Robot Machining
  Use your robot arm like a 5-axis milling machine (CNC) or a 3D printer. Simulate and convert NC programs to robot programs (G-code or APT-CLS files). RoboDK will automatically optimize the robot path, avoiding singularities, axis limits and collisions.

- Offline Programming Software
  Simulation and Offline Programming of industrial robots has never been easier. Create your virtual environment to simulate your application in a matter of minutes.
  Easily generate robot programs offline for any robot controller. You don't need to learn vendor-specific programming anymore.

- Robot Library
  Access an extensive library of industrial robot arms, external axes and tools from over 50 different robot manufacturers. RoboDK now offers an extensive library of over 600 robot arms from 50 robot manufacturers, more than 50 partners around the world and thousands of active users.
  Easily use any robot for any application, such as machining, welding, cutting, painting, inspection, deburring, and more!

- Robot Accuracy
  Calibrate your robot arm to improve accuracy and production results. Run ISO9283 robot performance tests.

- Export Programs to your Robot
  RoboDK Post Processors support many robot controllers, including:

    ABB RAPID (mod/prg)
    Fanuc LS (LS/TP)
    KUKA KRC/IIWA (SRC/java)
    Motoman Inform (JBI)
    Universal Robots (URP/script)
    ...and much more!

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

13. BASIC GUIDE OF THE SOFTWARE

You should see the RoboDK shortcut on your desktop when RoboDK is installed from robodk website.

https://robodk.com/download Double click the  shortcut to start RoboDK.

The RoboDK window contains a Main Menu, a Toolbar, a Status Bar and the Main Screen. The Station Tree in the Main Screen contains all the items available in the station, such as robots, reference frames, tools, programs, etc.

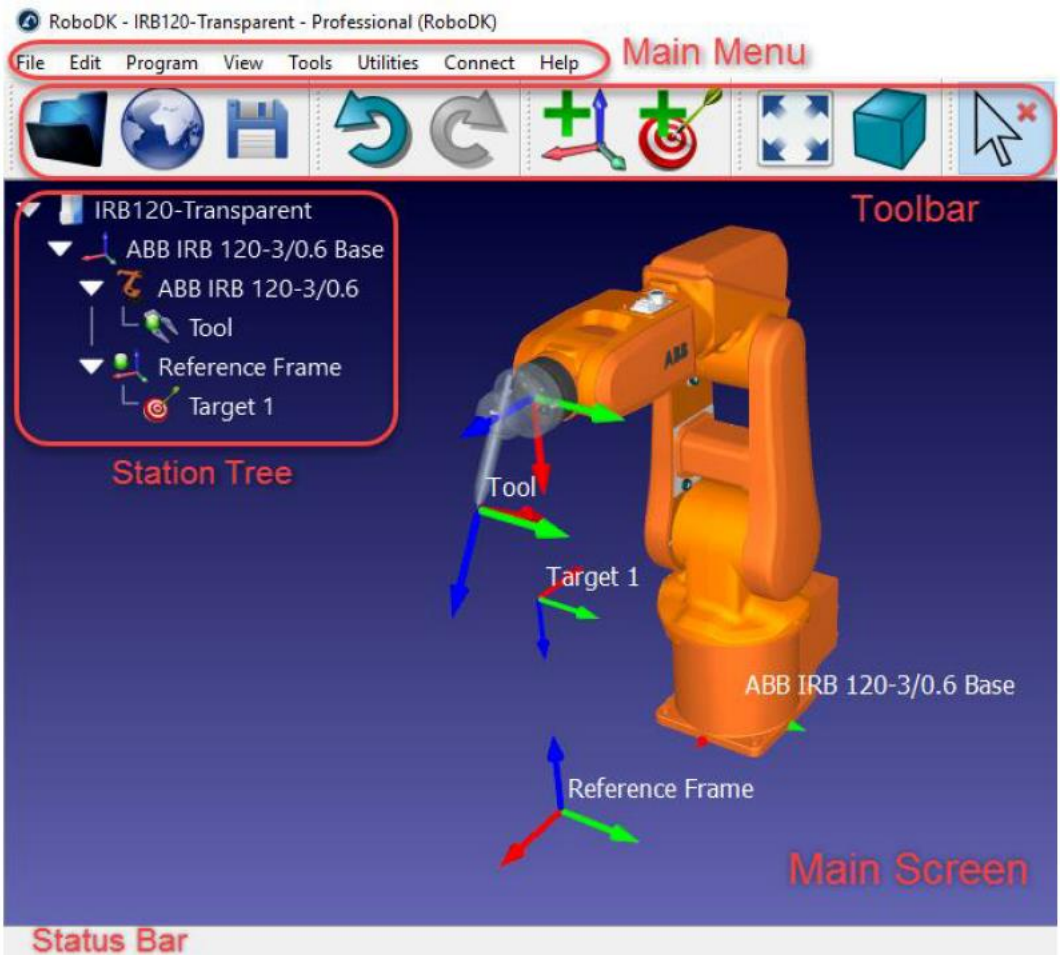


Figure 13.1. RoboDK window

Select	Pan	Rotate	Zoom
Left click	Hold mid button	Hold right click	Move mouse wheel

Figure 13.2. Mouse movements

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

14. MAIN MENU

14.1. Toolbar Menu

The RoboDK Toolbar contains graphical icons that allow quick access to frequently used actions in the menu.

Tip: Select **Tools→Toolbar Layout→Set Default Toolbar** to set up the default toolbar.

The following commands are available in the toolbar by default.

	**Open** Load a new file (RoboDK RDK Station) or a supported file type (robot, tool, STEP, IGES, STL, …)
	**Open online library** Show the online library (robots, tools and sample objects)
	**Save Station** Save the RoboDK station (RDK file)
	**Undo** Undo the last command (Ctrl+Z)
	**Redo** Redo the last command (Ctrl+Y)
	**Add a reference frame** Reference frames allow placing objects with respect to each other
	**Add a new target** Robot targets record robot positions with respect to a reference frame or in joint coordinates
	**Fit All** Update the 3D view to display all items
	**Isometric View** Display the default 3D isometric view
	**Move reference Frames** Move a reference frame by dragging it on the screen (hold Alt)
	**Move TCP (robot tool)** Move a robot TCP by dragging it on the screen (hold Alt+Shift)
	**Check collisions** Activate or deactivate collision checking. More information available regarding collision checking in the Collisions section
	**Fast simulation** Accelerate the simulation speed (hold the space bar)
	**Pause simulation** The simulation can be resumed by pressing the space bar
	**Add Program** Add a new robot program for simulation and program generation
	**Add Python Program** Add a new Python macro
	**Move Joint Instruction** Add a new joint movement instruction
	**Move Linear Instruction** Add a new linear movement instruction
	**Export Simulation** Export a program or simulation as a 3D PDF or 3D HTML file.

Figure 14.1. Commands available in the RoboDK Toolbar Menu

Funded by the European Union

14.2. File Menu

It is possible to Open, Save or export documents from the File menu.

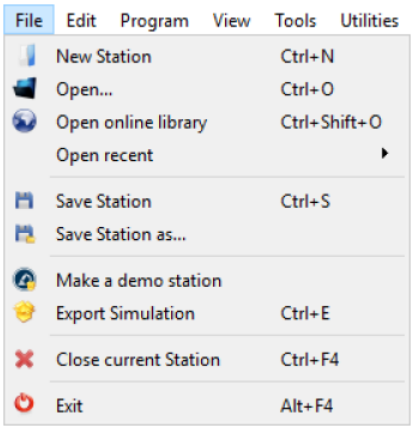


Figure 14.2. RoboDK File menu

New Station will add a new station in the tree. One station can be loaded or saved as one RDK file. The RDK file (RDK extension) holds all the information about the robots and objects so it is not required to keep a separate copy of the imported items.

Open will load a new RoboDK file (RDK Station) or import any other recognized file formats, such as .robot for robot files, STEP/IGES/STL for objects, .tool for tool files, etc.

Open online library will show a new window with the library available online.

Save Station will save the RDK file. Select Save Station as… to provide the file location.

Make a demo station will export the station as an EXE file with a simplified version of RoboDK.

Export Simulation will export a specific program or simulation as a 3D PDF or 3D HTML file. Example.

14.3. Edit Menu

Undo (Ctrl+Z) and Redo (Ctrl+Y) actions are accessible from the Edit menu. The history of undo actions is also available and allows reverting changes, backwards or forward, to a specific state by selecting the action.

It is also possible to cut (Ctrl+X), copy (Ctrl+C) or paste (Ctrl+V) one item or a group of items from the station tree. If an item is copied, all the items attached to it are also copied.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

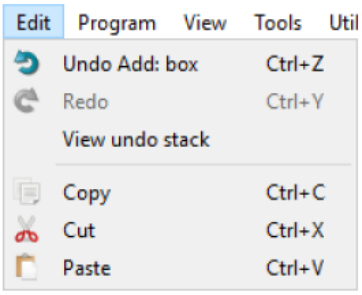


Figure 14.3. RoboDK Edit menu

14.4. Program Menu

The program menu contains all the components related to Offline Programming (OLP) and program generation. It is possible to add new programs, reference frames, targets or tools to robots. These Offline Programming components (reference frames, tools, targets, etc.) appear on all programs generated offline.

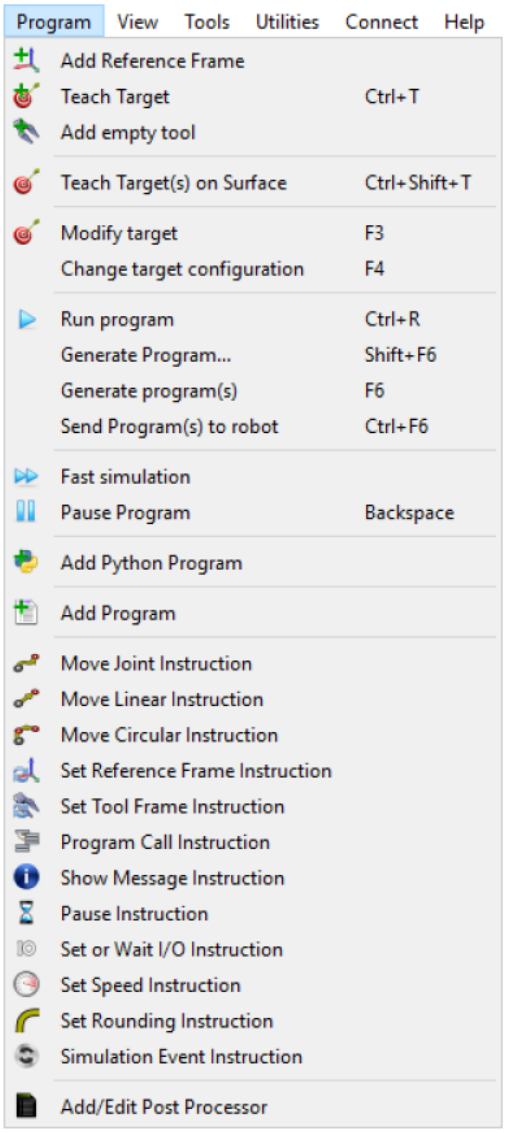


Figure 14.4. RoboDK Program menu

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

 **Add Reference Frame** will add a new reference frame attached to the station root or attached to another reference frame if that reference frame was selected.

 **Add empty tool** will add a new TCP to a robot. No geometry is required to add a new tool. Multiple Tools allow referencing different parts of the same geometry linked to one tool.

 **Teach Target (Ctrl+T)** will add a new target to the Active reference frame for the Active robot tool. The active reference frame and active tool can be selected in the robot panel. It is also possible to right click a reference frame or a tool to make them active.

 **Teach Targets on Surface (Ctrl+Shift+T)** will allow the user to select points of an object to easily create targets. An example is available in this section.

 **Add Program** will add a new program that can be created using the RoboDK Graphical User Interface (GUI). No programming experience is required to create or modify this type of robot program. The robot program can be simulated and generated for a specific robot, automatically and easily.

The Program Instructions section of the Offline Programming document provides more information about available program instructions through the GUI.

 **Add Python Program** option will include a sample Python program/macro/script/module in the station that links to the RoboDK API. A Python program using the RoboDK API allows creating robot programs from generic programming code (Python). A Python program is like a text file embedded in the station and contains Python code to automate specific tasks in RoboDK.

 It is possible to **Add or Edit Post Processors**. Post Processors define the way programs are generated for a specific robot controller, allowing to accommodate vendor-specific syntax. Post Processors are final component of the offline Programming Process.

14.5. View Menu

Most options required to navigate in 3D are available from the View menu. It is possible to Rotate, Pan and Zoom from this menu (as well as by right clicking the 3D view). This is useful for navigating in 3D using a laptop touchpad (instead of a mouse).

To allow a free rotation in any direction uncheck the option: View→Align rotation. Otherwise, RoboDK locks the station reference to keep the XY plane horizontal by default.

It is possible to show or hide the robot workspace by selecting the asterisk key (*). It is also possible to switch between visible and invisible items by selecting the F7 key.

Tip: It is possible to make the reference frames bigger or smaller by pressing the + or – key multiple times. If a lot of items are visible this is useful to adjust the size of the reference frames and properly grab them if they need to be moved from the 3D view (by holding the ALT key for example).

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

14.6. Tools Menu

Generic tools are available in the Tools menu, such as taking snapshots of the 3D view, activating the robot trace, activate collision checking or measuring point coordinates.

Activating the **Trace** will show the trace of all robots as they move.

 **Check collisions** will activate or deactivate collision checking. When collision checking is activated, objects that are in a collision state will be displayed in red. The Collision map allows specifying what object interactions are being checked.

 **Change color tool** will display a small window that allows changing the color of robots and objects. It is also possible to flip the normal vectors of surfaces.

 **Measure** will display a window that allows measuring points in 3D with respect to a local reference frame or the station reference frame (absolute measurements).

It is possible to specify the language of the RoboDK application by selecting **Tools→Language** and select the preferred language. RoboDK will be displayed in the selected language immediately.

Toolbar Layout allows setting up the default toolbar. Alternatively, it is possible to specify a toolbar for a more basic or more advanced usage.

Select  **Options** to open the main options menu. More information available in the Options Menu section.

14.7. Utilities Menu

The utilities menu allows performing specific tasks:

 **Calibrate Tool frame (TCP)** allows calibrating a robot TCP by providing data from the real setup, such as the joint configurations to reach a point using different orientations. This procedure is usually available from most robot teach pendants. RoboDK allows calibrating a TCP with as many configurations as desired. Using more configurations allows obtaining a more accurate TCP value.

 **Calibrate Reference Frame** allows identifying a reference frame with respect to a robot base frame. This allows accurately matching the part from the real setup to the virtual environment.

 **Calibrate Robot** allows setting up a robot calibration project to improve robot accuracy and find robot error parameters. A calibrated robot can be used in any RoboDK Offline Programming project. Robot calibration requires using measurement systems to take robot measurements. Robot accuracy and repeatability can be tested with ISO9283 before and/or after calibration.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

14.8. Connect Menu

It is possible to connect to a robot and enter the connections parameters, such as the robot IP, FTP username and FTP password. Setting up a robot connection allows transferring programs through FTP or running programs directly from the PC.

14.9. Help Menu

 **Help (F1)** opens this documentation online. A PDF version of the documentation is available for download at the top of each section. When you press F1, RoboDK displays the Help topic related to the item currently selected.

14.10. Options Menu

Select **Tools**  **Options (Shift+O)** to open the main RoboDK options window.

General tab

The main tab contains general options such as customizing your theme, 3D mouse navigation settings, the appearance of the tree, activate automatic backups or customize the decimal places displayed in the forms.

Station tab

The station parameters are the only parameters in the options menu that are saved with the RoboDK project (RDK file), instead of the user account settings.

Display tab

The display tab allows you to customize settings related to the appearance of the 3D view.

Motion tab

The motion tab allows you to customize the behavior of robot simulations and the tolerances used by RoboDK to display or prevent robot singularities and collisions.

CAD tab

The CAD section (Computer-aided design) allows you to specify settings related to importing parametric files (STEP/STP and IGES/IGS) and displaying these files in the 3D environment.

CAM tab

The CAM section (computer-aided manufacturing) shows all the settings related to robot manufacturing operations, such as robot machining or 3D printing, and how to import robot toolpaths created using CAM software.

Program tab

The Program tab allows you to customize settings related to robot programs and how program files are generated.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

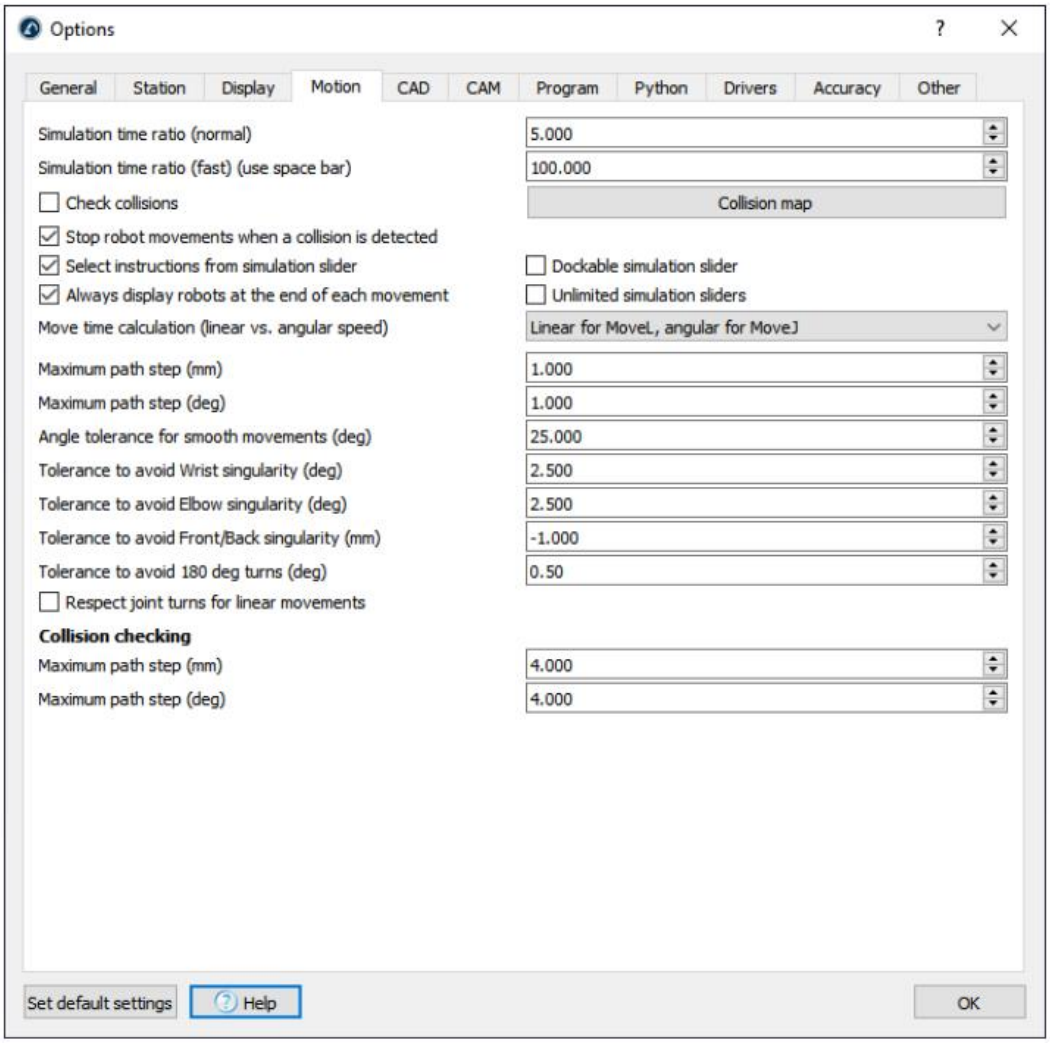


Figure 14.5. RoboDK Options/Motion menu

Python tab

The Python tab allows you to setup the path of the Python interpreter and the Python editor used by RoboDK. Most post processors require Python to allow you to generate brand-specific robot programs. Also, some examples that use the RoboDK API require Python to be installed. RoboDK installs Python 3.7 by default.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

15. GETTING STARTED

This getting started guide will help you create a simple project in RoboDK for robot simulation and offline programming. All robots, objects and tools used in a RoboDK project are saved as a RoboDK station (RDK file). A RoboDK station contains all settings related to robots, tools, reference frames, targets, objects and other parameters. The RoboDK station is stored in one file (RDK extension).

15.1. New Project

Select **File→New Station (Ctrl+N)** to start a new project

15.2. Select a Robot

New robots can be added to your project from your PC or from RoboDK's online library.

Select **File→Open robot library** (Ctrl+Shift+O). It is also possible to select the corresponding button in the toolbar.  Select **Download.** The robot should automatically appear in the station in a few seconds. The online library can be closed once the robot is loaded. Moreover you can add stations and Add-ins to your Project from roboDK's online library.

15.3. Create a Tool

New robot tools (TCPs) can be loaded or created in RoboDK from previously loaded 3D geometry.

Follow these steps to load an object and set it up as a robot tool:

Select **File→**  **Open.** Select the tool file to add it as an object (it will be added at the robot base frame). Drag & drop the object to the robot item inside the station tree. New tools can be loaded or saved as a .tool format.

15.4. Robot Panel

Double click a robot to open the robot panel (you can double click it in the tree or the 3D view). It is possible to jog the robot axes using the Joint axis jog section and enter specific joint axis values in the text boxes. The joint values and the robot coordinates should match with the values displayed by your robot controller.

You can double click the joint limits to modify the robot axis limits. By default, RoboDK uses the same joint limits used by the robot controller (physical hardware limits).

The Cartesian Jog section displays all the information related to the robot kinematics:

The Tool Frame (TF) with respect to the Robot Flange (FF) defines where the selected Tool Frame is located with respect to the Robot Flange. The Robot Flange is always the same, however, the Tool Frame changes depending on the tool that is mounted on the robot. This relationship is also known as UTOOL, ToolData or just Tool in most robot controllers. The Robot Tool is also known as the TCP (Tool Center Point). The Selected Tool becomes the "Active" tool. The active tool is used when creating new targets and programs. The selected tool displays a green mark in its icon: 

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

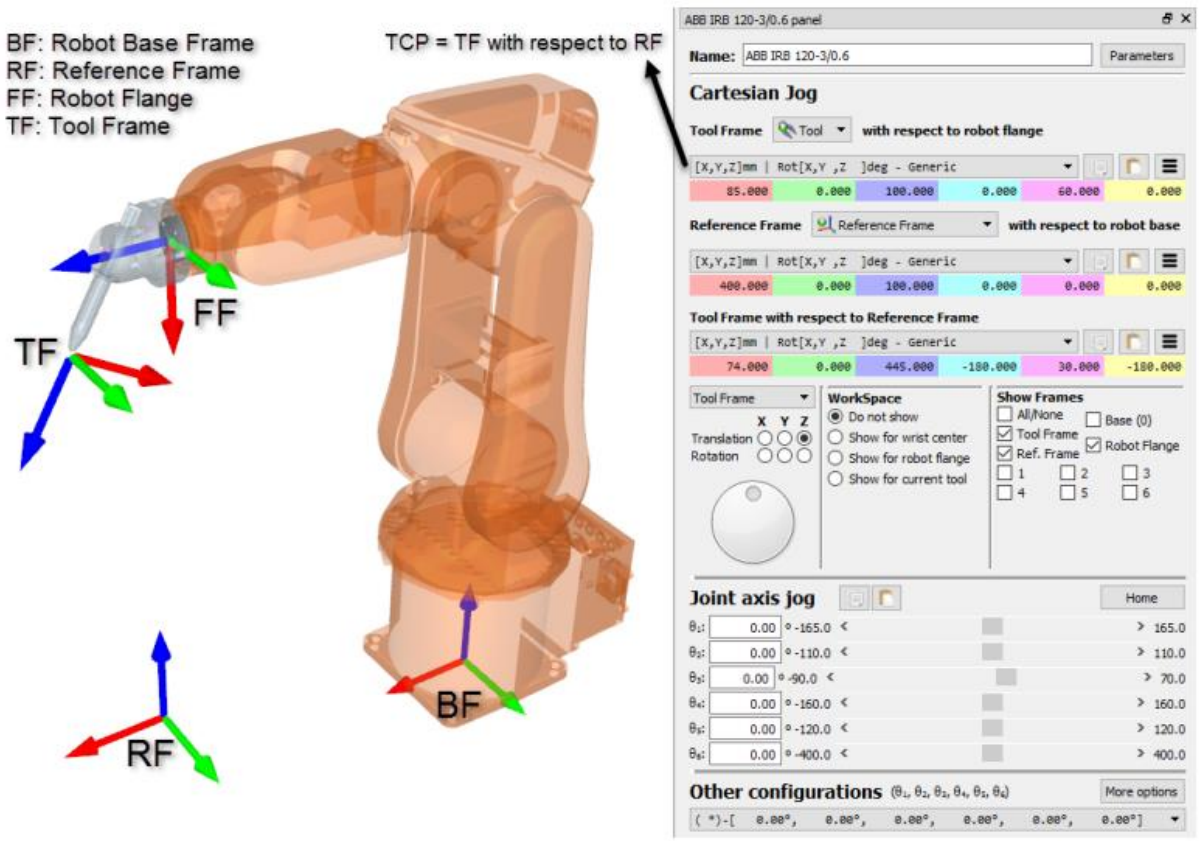


Figure 15.1. Robot panel and frame definitions

The Reference Frame (RF) with respect to the Robot Base (BF) defines where the Reference Frame is located with respect to Robot Base Frame. The Robot Base Frame never moves, however, different Reference Frames can be used to position any objects with respect to the same Robot Base Frame. This relationship is also known as UFRAME, WorkObject MFRAME or Reference in most robot controllers. The selected reference frame in the robot panel becomes the "Active" reference frame. The active reference frame is used as a reference for new targets and robot programs. The selected reference frame displays a green mark in its icon:  .

The Tool Frame (TF) with respect to the Reference Frame (RF) shows the position of the active TCP with respect to the active reference frame for the current position of the robot. Modify this value to move the robot. The joint axes are recalculated automatically. These Cartesian coordinates are recorded when a new target is created (Program→Teach Target), together with the robot axes. The target is also attached to the Active reference frame.

15.5. Save Station

Select **File→**  **Save Station** (Ctrl+S). Save the file as helloworld.rdk. The Window title and the Station name will be updated.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

WORKSHEET 1

MOVE THE INDUSTRIAL ROBOT ON THE MAIN SCREEN

The aims at the end of this worksheet are:

The student can bring an industrial robot to the study area.

The student can control the robot in the study area.

The student can operate the industrial robot through the robot panel.

Student can save their work.

Process Steps:

1- Open the program by double-clicking the program icon on the desktop 

.

2- Select the ABB IRB 120-3/0.6 robot from the File / Open Robot Library menu and bring it to the worksheet.

3- Select the Generic Pencil Tool from the File / Open Robot Library menu and bring it to the worksheet.

4- Move your robot on the Main Screen by using the right mouse button, scroll button and pressed scroll button.

5- Open the Robot Panel by double-clicking on the robot or the robot's name in the Station Tree.

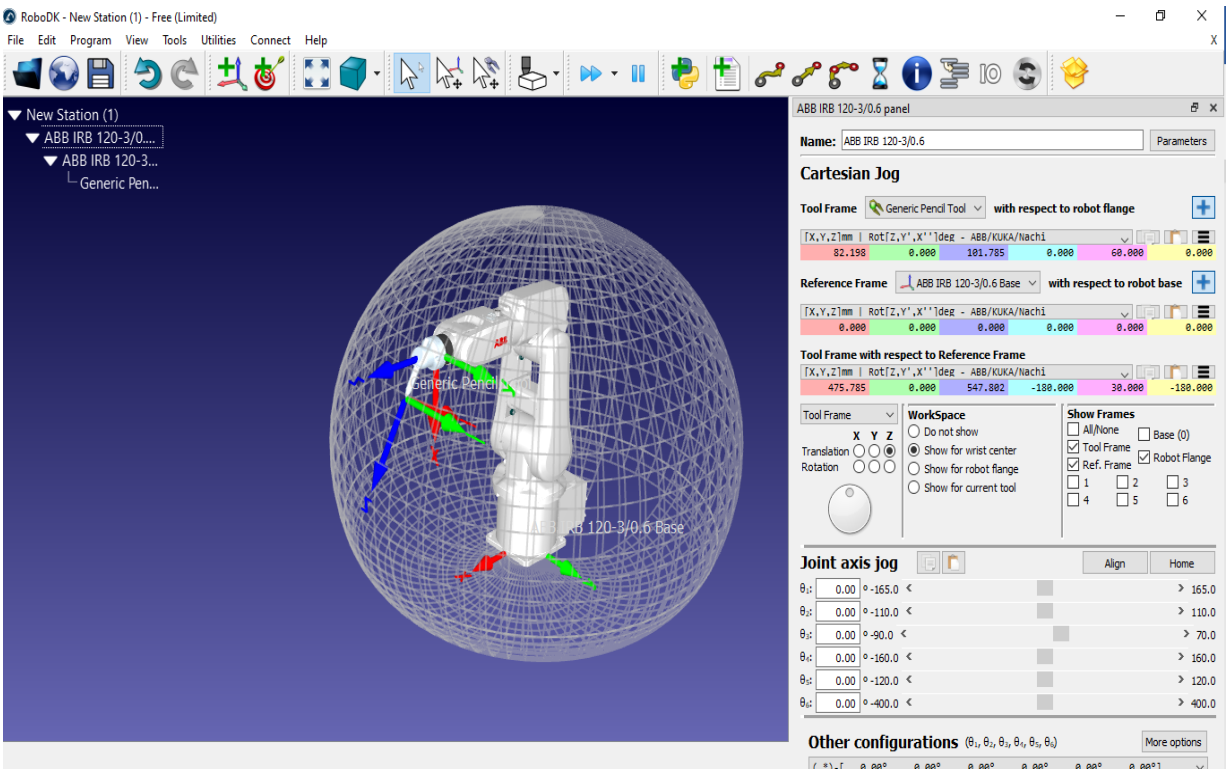


Figure 15.2. WorkSpace of the ABB IRB 120-3/0.6 robot and the Robot Panel

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

6- Click on the options in the WorkSpace area to see the workspace of your robot.

7- Change the frame visibility of your robot by clicking the options in the Show Frames field.

8- Move the joints of your robot by changing the adjustment bars in the Joint axis jog area. Press the Home button again to return to the initial state.

9- Hide product names with Shift + / keys.

10- Move your robot's joints with the frame arrows that appear on the screen by pressing the  button.

11- Save your work as worksheet 1 by pressing the icon. 

12- Please repeat the same steps for the Universal Robot UR-10.

13- Save this station you made with Cobot under the name worksheet 1-1.

Study Question:

Please, research the meanings of the words payload, work space and reach.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

15.6. Create Targets

Robot positions are recorded as Targets. A Cartesian target defines the position of the tool with respect to a coordinate system. A Joint target defines the position of the robot given robot joint values.

Note: RoboDK creates Cartesian targets by default (red targets). You can right click a target and set it as a Joint target to convert it to a Joint target (green targets).

Select **Program→**  **Teach Target (Ctrl+T)**, or the corresponding button in the toolbar. The target will automatically remember the current robot position (cartesian and joints axes).

It is common practice to use joint targets to reach a first approach position close to the working area, then, Cartesian targets ensure that the toolpath is not altered if the reference frame or the tool frames are modified.

Right click a target, then select More Options… (F3) to see the recorded pose and joint values.

The following colors are used by default:

X coordinate → Red

Y coordinate → Green

Z coordinate → Blue

1st Euler rotation → Cyan

2nd Euler rotation → Magenta

3rd Euler rotation → Yellow

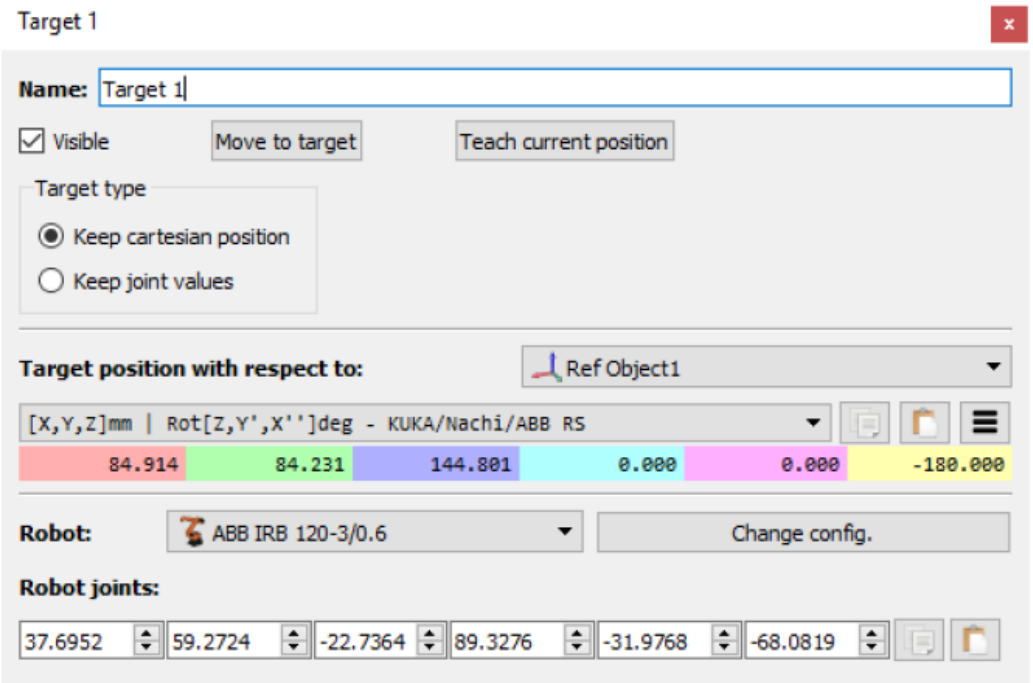


Figure 15.3. Target options window

WORKSHEET 2

CREATE TARGETS FOR ROBOT

The aims at the end of this worksheet are:

The student can bring some targets for robot to the study area.

The student can organize coordinates of targets for robot in the study area.

The student can use  button or 

button for design the targets of robots

Process Steps:

1- Open the program by double-clicking the program icon on the desktop 

.

2- Select the ABB IRB 120-3/0.6 robot from the File / Open Robot Library menu and bring it to the worksheet.

3- Select the Generic Pencil Tool from the File / Open Robot Library menu and bring it to the worksheet.

4- Move your robot by pressing the  button. When you reach the desired point, press the  button to save the target.

5- Move the robot in another direction. Double-click on the Target1 line in the Station Tree and observe that the robot returns to its Target1 point.

6- Create 3 more target coordinates to form a rectangle on the screen, then go to the destinations in the Station Tree and press the F2 key. Change the names of the targets to Corner1, Corner2, Corner3, Corner4.

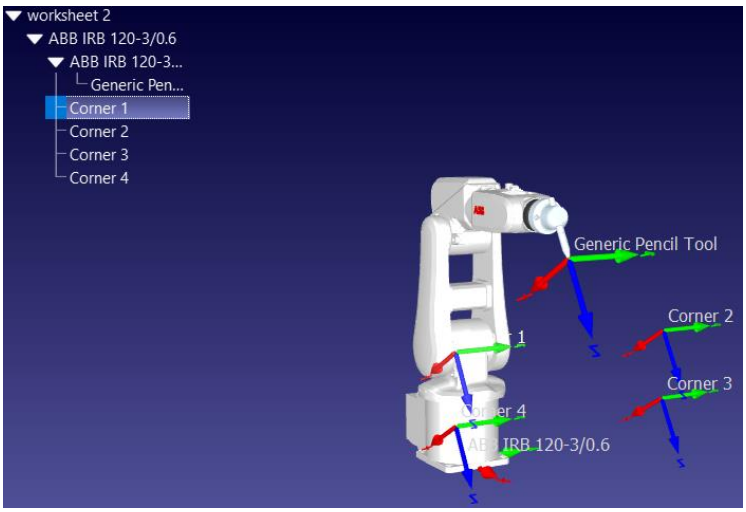


Figure 15.4. The ABB IRB 120-3/0.6 robot with created targets

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

7- Open the Options window by highlighting the Corner 1 in the Statio Change config. ing the F3 key. Change the joint settings of the robot with the Change Button from the menu.

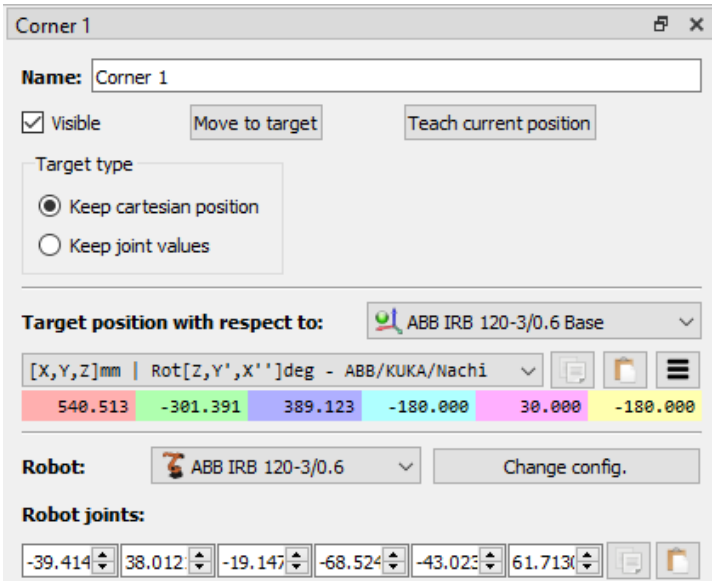


Figure 15.5. Target options window for the Corner 1

8- Enter the target coordinate values below to make a complete rectangle.

Set Corner 1 to 540, -300, 390, -180, 30, -180.

Set Corner 2 to 540, 200, 390, -180, 30, -180.

Set Corner 3 to 540, 200, 240, -180, 30, -180.

Set Corner 4 to 540, -300, 240, -180, 30, -180.

9- Save your work as worksheet 2 by pressing the icon.

10- Please repeat the same steps for the Universal Robot UR-10.

11- Save this station you made with Cobot under the name worksheet 2-1.

Study Question:

Please create the target coordinates given above using the Robot Panel for the ABB robot.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

16. ROBOT PROGRAMS

RoboDK is a simulator focused on industrial robot applications. This means that robot programs can be created, simulated and generated offline for a specific robot arm and robot controller. In other words, RoboDK is software for Offline Programming.

An extensive library of industrial robots is available. Industrial robots are modelled in RoboDK the same way they behave using vendor-specific controllers, including axis limits, sense of motion and axis linking. The RoboDK API can be used to complement these programs or to completely create a robot program.

16.1. Offline Programming

Offline Programming (or Off-Line Programming) means programming robots outside the production environment. Offline Programming eliminates production downtime caused by shop floor programming (programming using the teach pendant).

Simulation and Offline Programming allows studying multiple scenarios of a robot cell before setting up the production cell. Mistakes commonly made in designing a work cell can be predicted in time.

Offline Programming is the best way to maximize return on investment for robot systems and it requires appropriate simulation tools. The time for the adoption of new programs can be cut from weeks to a single day, enabling the robotization of short-run production.

16.2. Create a Program

A simulation can be accomplished by adding a sequence of instructions in a program. Each instruction represents specific code for a specific controller, however, RoboDK offers a Graphical User Interface (GUI) to easily build robot programs, in a generic way, without the need to write code.

The code specific to a robot controller will be generated automatically when the program is generated. To create a new empty program using the RoboDK Graphical User Interface:

1. Select **Program**→  **Add Program.** Alternatively, select the corresponding button in the toolbar.

2. Select **Tools→Rename item… (F2)** to rename the program

This action will create an empty program and will allow adding new instructions.

16.3. Program Instructions

It is possible to add new instructions by right clicking a program or selecting an instruction from the Program menu.

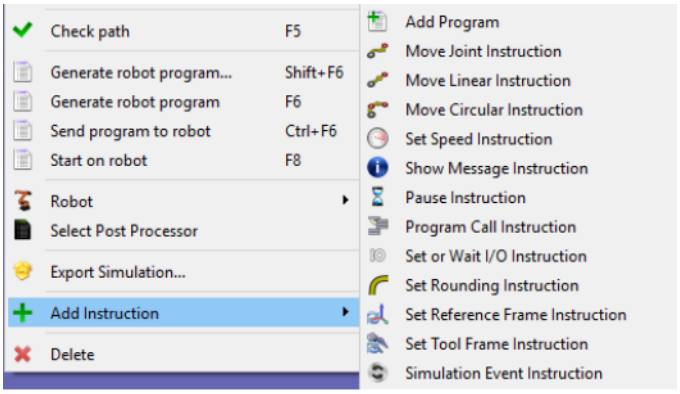


Figure 16.1. Program instructions

16.3.1. Joint Move

Select **Program**→ **Move Joint Instruction** to add a new joint movement instruction. Alternatively, select the corresponding button in the toolbar. Unless a target is selected before adding the instruction, the movement instruction will create a new target and they will be linked. If the target is moved the movement is also modified.

If this is the first instruction that is added to the program, two more instructions will be added before the movement instruction: a Reference Frame selection and a Tool Frame selection. This will make sure that when the program reaches the movement instruction the robot is using the same reference and tool frames used to create this new target.

16.3.2. Linear Move

Select **Program**→ **Move Linear Instruction** to add a new linear movement instruction. Alternatively, select the corresponding button in the toolbar.

Important: It is recommended to keep the first movement of each program as a Joint Move using a Joint target. This will properly set up the desired configuration from the first movement and make sure that the real robot is moving the same way it was simulated.

Contrary to Joint Movements, Linear Movements are sensible to robot singularities and axis limits. For example, 6-axis robots can't cross a singularity following a linear move. The following image shows an example saying *Joint 5 is too close to a singularity (0 degrees). […] Consider a Joint move instead*. As shown in the following image.

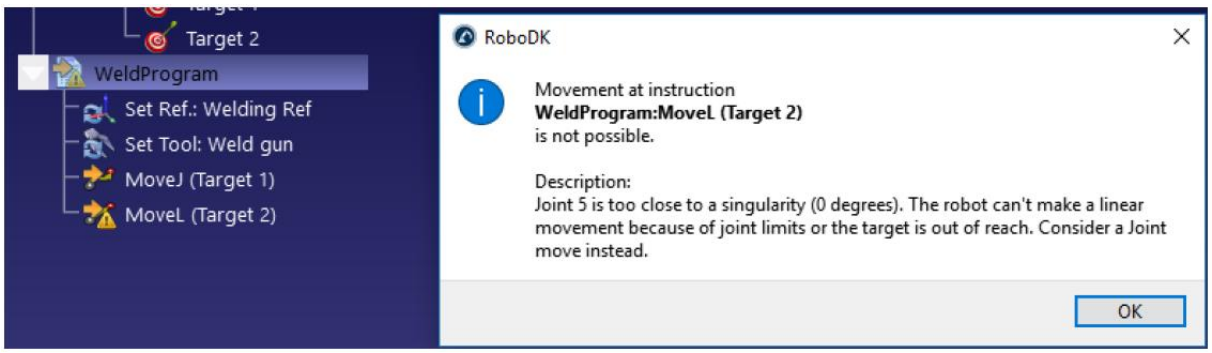


Figure 16.2. Linear movement warning information

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

16.3.3. Set Reference Frame

Select **Program**→  **Set Reference Frame Instruction** to use a specific reference frame. This will update the given reference frame on the controller for the following movement instructions and will change the **Active** reference frame of the robot in RoboDK for simulation purposes. That means that movement instructions to specific targets (Cartesian targets) will be made with respect to the last reference frame set.

16.3.4. Set Tool Frame

Select **Program**→  **Set Tool Frame Instruction** to use a specific tool frame (TCP). This will update the given tool frame on the program for the following movement instructions and will change the **Active** tool frame of the robot in RoboDK for simulation purposes. That means that movement instructions to specific target (Cartesian targets) will be made with respect to the last tool frame set.

16.3.5. Circular Move

Select **Program**→  **Move Circular Instruction** to add a new circular movement instruction. Alternatively, select the corresponding button in the toolbar.

Unless two targets are selected before adding the instruction, the movement instruction will create no new targets. It is required to add two more targets separately and link them from the circular move instruction, as shown in the next image.

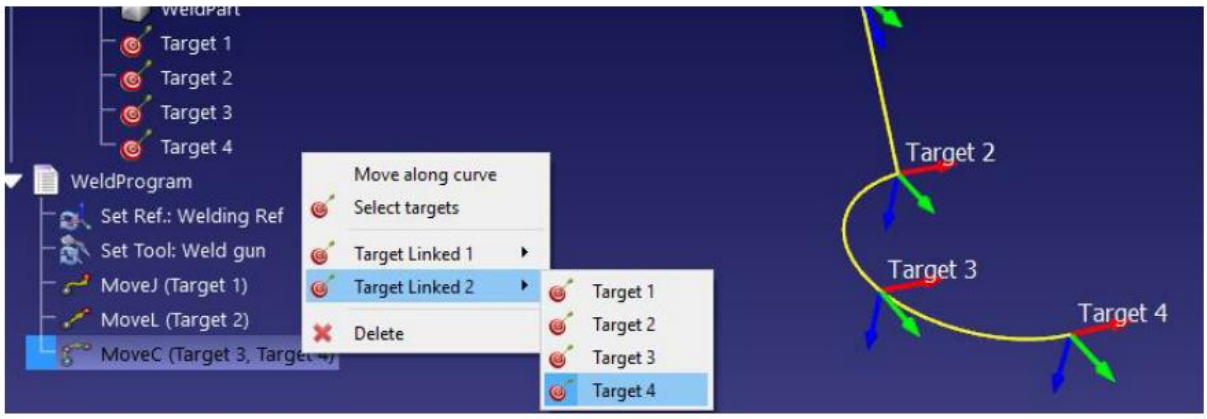


Figure 16.3. Circular movement instruction

The circular path is an arc created from the point where the robot is located, passing through the first circular point (Target Linked 1) and ending at the end point (Target Linked 2). It is not possible to accomplish a full circle with only one circular instruction. A full circle must be split into two separate circular moves.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

16.3.6. Set Speed

Select **Program**→  **Set Speed Instruction** to add a new instruction that changes the speed and/or the acceleration. It is possible to specify speed and accelerations in the joint space and in the cartesian space.

Activate the corresponding cases to impose a specific speed and/or acceleration in the program. The robot speed is applied from the moment this instruction is executed.

The robot speed can also be changed in the robot parameters menu: Double click the robot, then, select parameters.

Note: Not all robot controllers support setting accelerations accurately.

16.3.7. Show Message

Select **Program**→  **Show Message Instruction** to add a new instruction that will display a message on the teach pendant.

16.3.8. Pause

Select **Program**→  **Pause Instruction** to add a new instruction that will pause the program execution for some time or stop the program until the operator desires to resume the program.

Set the pause delay value to -1 to pause the program until the operator desires to resume the program. In that case, the instruction will be automatically named *Stop*. In the simulation, a 5 second pause will take 1 second to simulate for the default simulation ratio of 5.

16.3.9. Program call

Select **Program**→  **Program Call Instruction** to add a call to a sub program from the current program.

By default, this is a blocking call to a specific program. However, it is possible to switch to **Insert Code** to enter code specific at the location of this instruction. This might be useful for a specific application and a specific controller.

Tip: Select **Select program** to automatically fill the text field. Otherwise, a text match should also work. If there is a name match with the sub program used in the instruction, this subprogram will be simulated in RoboDK.

Tip: Enter multiple lines to automatically set up multiple program call instructions in a row.

Switch from **Program Call** to **Start Thread** to provoke a non-blocking call to a sub program. In this case, the controller will start a new thread. This option is only available for certain controllers and only works for specific operations.

16.3.10. Set/Wait IO

Select **Program→** **Set or Wait I/O Instruction** to change the state of Digital Outputs (DO). By default, this instruction is set to **Set Digital Output**. This instruction also allows waiting for a specific Digital Input (DI) to switch to a specific state.

The IO Name can be a number or a text value if it is a named variable. The IO Value can be a number (0 for False and 1 for True) or a text value if it is a named state.

Note: This instruction also supports setting Analog Outputs (AO) or waiting for Analog Inputs (AI) on some robot controllers. In that case, it is possible to provide decimal numbers or specific text instead of numbers.

16.3.11. Set Rounding value

Select **Program→** **Set Rounding Instruction** to alter the rounding accuracy. The rounding accuracy used to smooth the edges between consecutive movements. This change takes effect from the moment it is executed inside a program (same as with all the other instructions), so it is typical to set this value at the beginning of a program.

Without a rounding instruction, the robot will reach the speed of 0 at the end of each movement (unless the next movement is tangent with the previous movement). This will provoke high accelerations and quick speed changes to ensure the best accuracy for each movement.
A high rounding value will ensure a constant speed through the robot path in exchange of losing accuracy on the path edges. Depending on each application, it is common to find a good compromise between accuracy and a smooth speed.

16.3.12. Simulation event

Select **Program→** **Simulation Event Instruction** to provoke a specific simulation event. Simulation events have no impact on generated code and are used only to provoke a specific event for simulation purposes. Simulation events using the graphical user interface allow you to:

- attach or detach objects to robot tools,
- show or hide objects or tools,
- change the position of objects and reference frames .

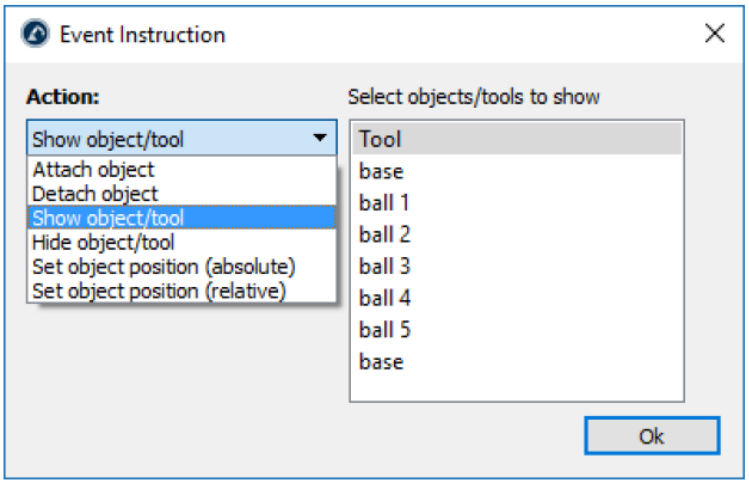


Figure 16.4. Simulation event instruction window

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

17. SIMULATE PROGRAM

Double click the program  to start the program simulation.

Alternatively:

1. Right click the program
2. Select **Run**

A simulation bar will appear at the bottom if the program is double clicked. It is possible to slide the simulation to move the simulation forward or backwards using the simulation bar.

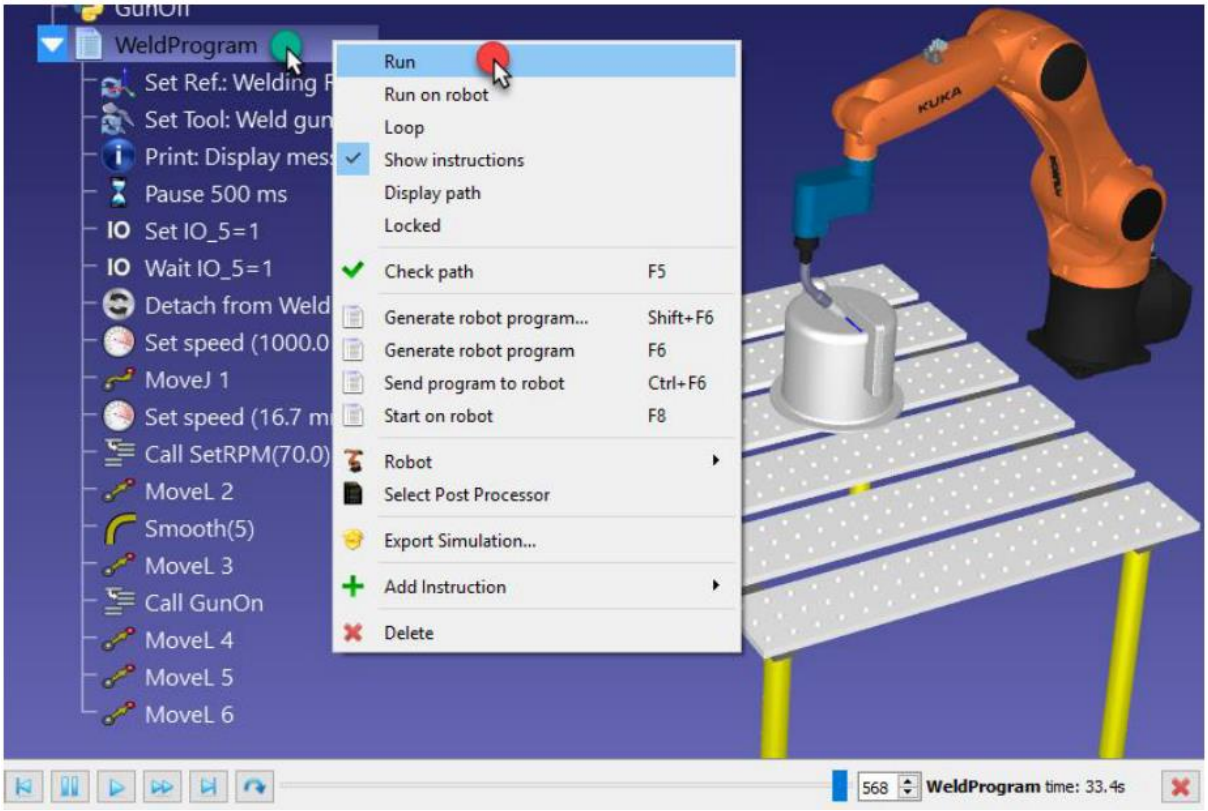


Figure 17.1. Program simulation in RoboDK

Select Program→  Fast simulation to speed up the simulation (or hold the space bar). This option is also available in the toolbar.

RoboDK simulates 5 times faster than real time by default. That means that if a program takes 30 seconds to execute it will be simulated in 30/5=6 seconds. Speeding up the simulation increases this ratio to 100. Normal and fast simulation speeds can be changed in the Tools→Options→Motion menu.

Select **Program**→  **Pause** to pause the simulation (or the Backspace key). Select Esc key to stop the simulation or double click the program again. Double click each instruction individually to execute them one by one. Right click a movement instruction and select **Start from here** to resume the program execution from that instruction.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

WORKSHEET 3

ROBOT PROGRAMMING

The aims at the end of this worksheet are:

The student can move the robot arm between targets.

The student can learn the types of movements between targets.

The student can understand the importance of the study area.

Process Steps:

1- Open the program by double-clicking the program icon on the desktop 

.

2- Press the  button to open the previous worksheet you created.

3- Add a program block to the robot by pressing the  button. See that the program you added comes to the Station Tree as Prog1. Point to the Prog1 line. Rectangle the name of the program by pressing the F2 key.

4- We will add movement to our robot by using the  movement buttons. Press the  Joint Move button. Target5, Set Ref, Set Tool and MoveJ lines will appear in the Station Tree.

5- Target5 is undesirable. Because there are 4 targets in our move list. So we're deleting Target5. We fix this situation as follows: We select the Corner1 target and the Rectangle program with the CTRL key. Press the Joint Move button. And only MoveJ (Corner1) command comes to the tree. We repeat the same for the other corners.

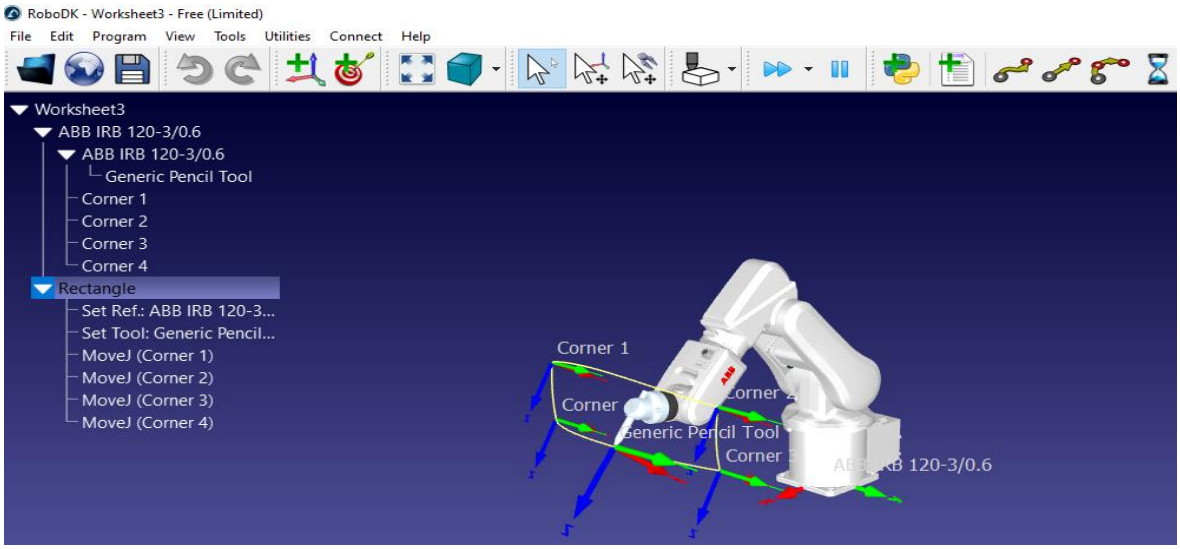


Figure 17.2. The Rectangle program with MoveJ commands

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

6- When we double click on the Rectangle program, we see that our robot makes a rectangle on the screen. The movement of the robot arm between targets is curved.

7- We select all four MoveJ commands and press the right button. We choose the Set Move Linear command from the menu that appears. Our commands became MoveL. Thus, the movement between targets will linear.

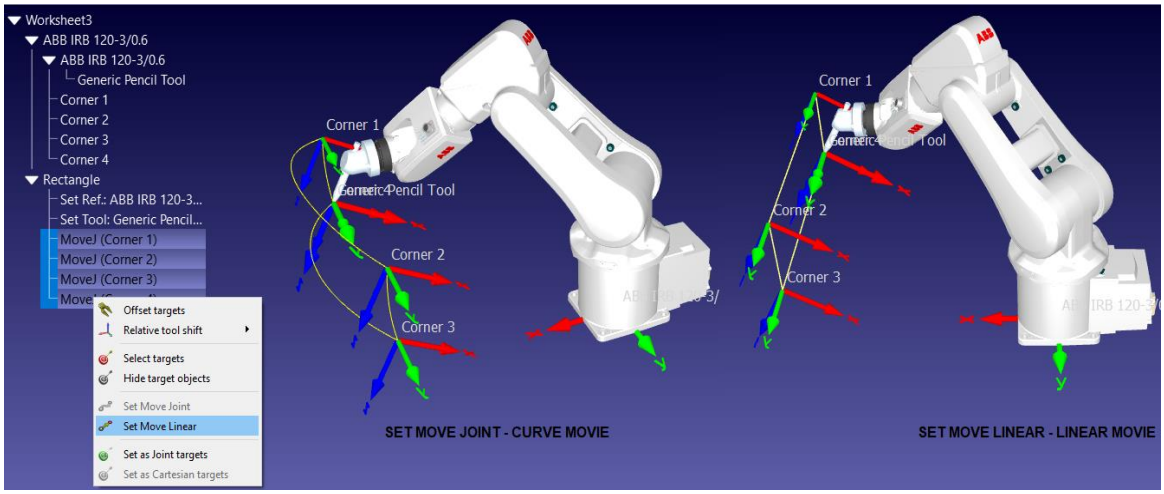


Figure 17.3. Comparison of MoveJ and MoveL commands

8- Select Corner Targets, Pencil Tool and ABB robot respectively in the Station Tree and press the right button. Let's remove the Visible check from the menu that appears. Thus, a robot will appear on the screen that draws a very clean rectangle.

9- Select the Loop command from the menu that appears when right-clicking on Rectangle. Thus, the operation will enter an infinite loop.

10- Select **File→Save station as...** to save your work as worksheet 3.

11- Please repeat the same steps for the Universal Robot UR-10.

12- Save this station you made with Cobot under the name worksheet 3-1.

Study Question:

Please create the target coordinates of EU word for the ABB robot.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

18. REFERENCE FRAMES

A Reference Frame defines the location of an item with respect to another item with a given position and orientation. An item can be an object, a robot or another reference frame. All Offline Programming applications require defining a reference frame to locate the object with respect to a robot to update the simulation accordingly. Drag & drop any reference frame or object within the Station Tree to define a specific relationship.

Hold the **Alt** key to move reference frames with respect to each other. Alternatively, select the corresponding button in the toolbar:  Then, drag the reference with the mouse on the screen. As the reference is being moved, the corresponding coordinate values will be updated.

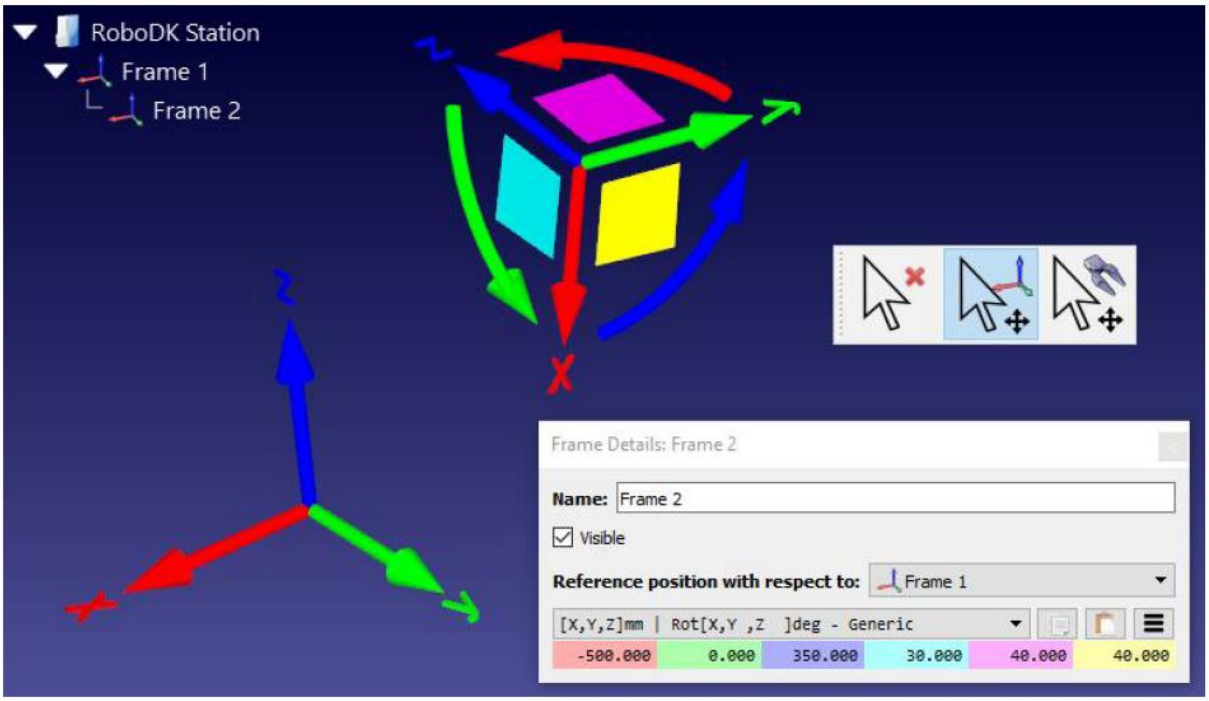


Figure 18.1. Reference Frames and the Frame Details window in RoboDK

The relationship of one reference frame with respect to another reference frame is also known as **pose** (position and orientation). A **pose** can be represented by the XYZ position and Euler angles for the orientation, by the XYZ position and Quaternion values or by a 4x4 matrix.

The following colors are used by default:

X coordinate → Red
Y coordinate → Green
Z coordinate → Blue

1st Euler rotation → Cyan
2nd Euler rotation → Magenta
3rd Euler rotation → Yellow

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

19. ROBOT CONFIGURATION

One robot configuration defines a specific state of the robot. Changing the configuration requires crossing a singularity. Robot controllers can't cross a singularity when a linear movement is being made (a joint movement would be required for that).

In other words, to accomplish a linear movement between two targets the robot configuration must be the same for the complete movement, including the first and last points.

Right click a robot and select Change configuration to open the robot configurations window. It is also possible to open this window by selecting More options in the robot panel.

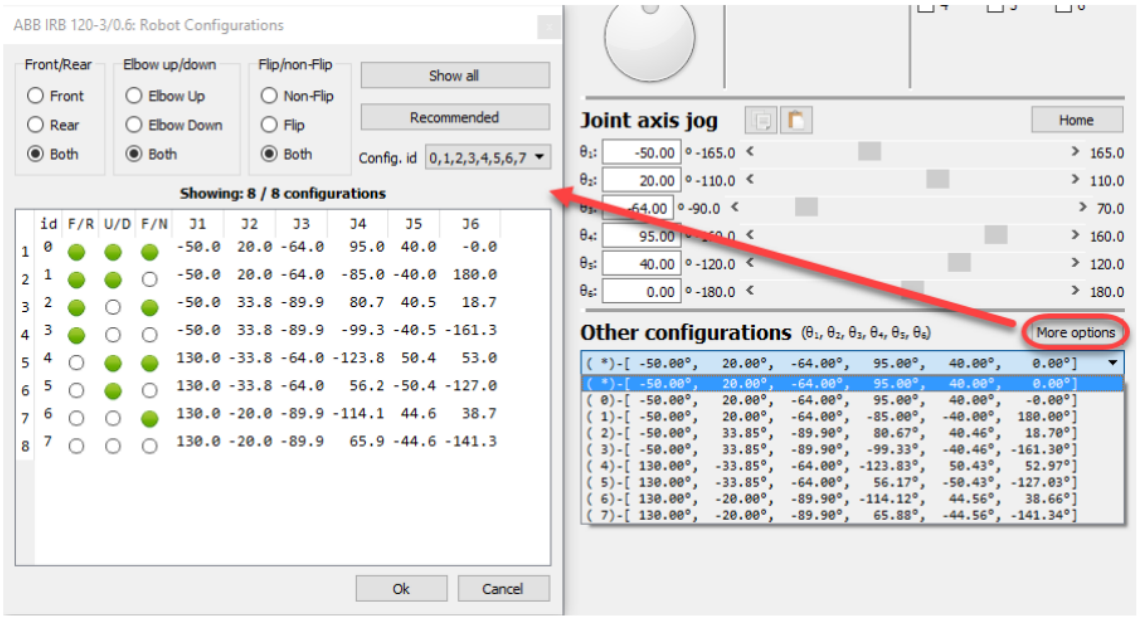


Figure 19.1. The robot panel with the robot configurations window

For a standard 6-axis robot there are typically 8 different configurations for any position of the robot if we assume each robot axis can move one full turn. In practice, joint limits can be more or less constrained depending on the robot. Therefore, it may be possible to have from 1 to more than 100 different robot configurations for a specific location depending on the robot.

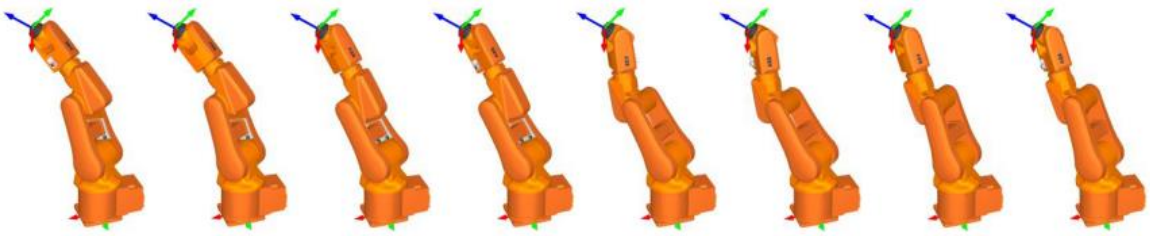


Figure 19.2. Different configurations for the same robot position

One robot configuration defines a specific way (assembly mode) of reaching a position with the robot. For example, the robot can have the elbow up or the elbow down (Up vs. Down, or U/D), at the same time it can be facing the target or the base can rotate 180 degrees to reach the target backwards (Front vs. Rear, or F/R). Finally, joint 5 can flip by switching the sign at the same time axis 4 and axis 6 compensate for that move (Flip vs. Non-Flip, or F/N). In total, this provides the 2*2*2=8 configurations.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

WORKSHEET 4

DEFINE FRAME FOR ROBOT

The aims at the end of this worksheet are:

The student can create frame for the robots.

The student can define frame in the working area.

The student can understand the importance of frame.

Process Steps:

1- Open the program by double-clicking the program icon on the desktop 

.

2- Press the  button to open the previous worksheet you created.

3- Let's make all our targets a frame. First, let's bring a Frame to the screen by pressing the  Frame button. It appears in the Station Tree as Frame2.

4- Let's bring the Frame2 formed on the worksheet to the bottom of our robot in the Station Tree by holding it with the mouse. Then let's rename Frame2's name Frame Rectangle using the F2 key.

5- After creating the frame, let's define it now. Let's choose Corner 1, Corner 2, Corner 3 and Corner 4. Let's choose the Change support command from the menu that opens with the right button. Let's click on the Frame Rectangle line.

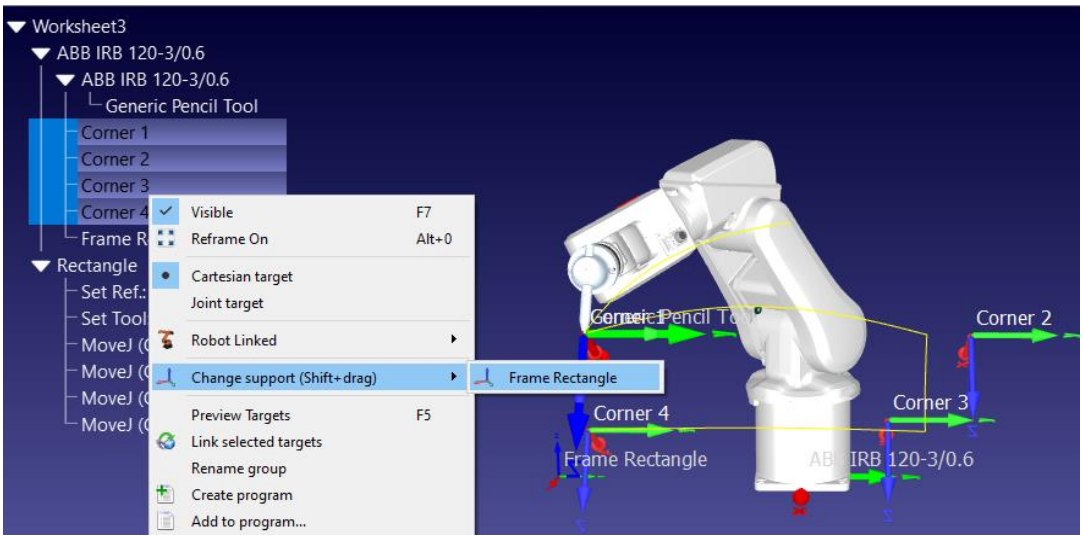


Figure 19.3. Moving the targets to the Frame Rectangle

- All Corner Targets are created under the Frame Rectangle. Now by moving the frame, you move these four corners. You can try this using the  button.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

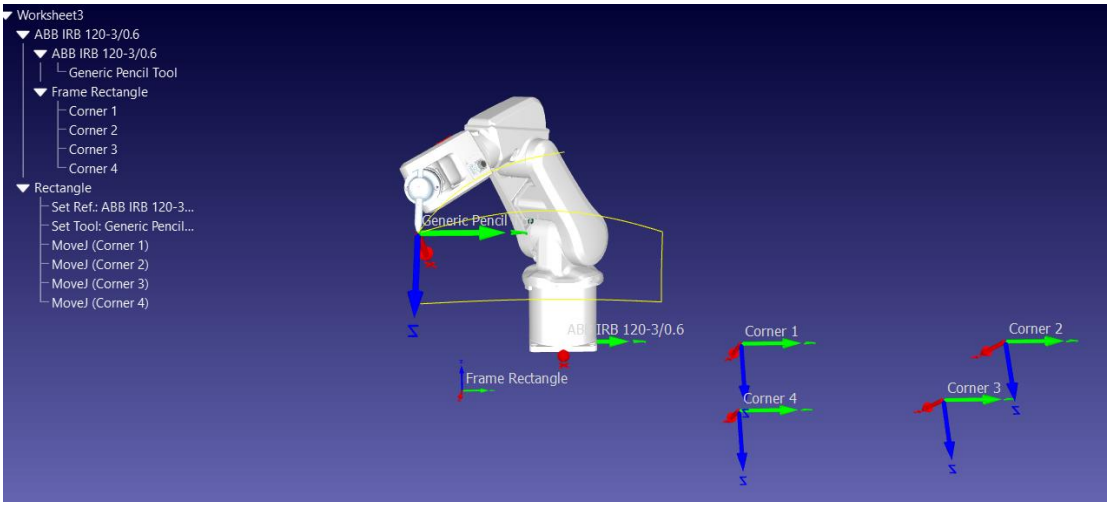
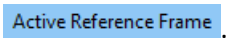
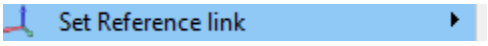


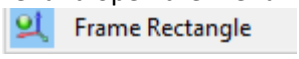
Figure 19.4. The targets moved with the Frame Rectangle

6- Double click the ABB robot from the Station Tree to open the ABB IRB 120-3/0.6 panel. Select the Frame Rectangle as Reference Frame: 

. Then open the menu with a right click on the ABB IRB 120-3/0.6 Frame in the Station Tree. Choose from here 

. Repeat the same process for Frame Rectangle.

7- The program will not run when you run it. Go to Set Ref and open the menu with a right click. Select 

and 

from the drop-down menu.

8- Thus, we introduced the Frame2 (Frame Rectangle) you created to all the elements in the station. By moving Frame2 (Frame Rectangle), you have run the robot at the 4 coordinates you want.

9- Right click the ABB robot on the Main Screen and select **Change configuration** to open the robot configurations window. It is also possible to open this window by selecting **More options** in the robot panel. Select the other configurations.

10- Select **File→Save station as...** to save your work as worksheet 4.

11- Please repeat the same steps for the Universal Robot UR-10.

12- Save this station you made with Cobot under the name worksheet 4-1.

Study Question:

Please read the Frame titles in the Help menu of the program. Bring sample applications.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

20. IMPORT 3D OBJECTS

RoboDK supports most standard 3D formats such as STL, STEP (or STP) and IGES (or IGS) formats, so you can create your own 3D objects using for example free Tinkercad software, and import them to RoboDK's project. Other formats such as WRML, 3DS or OBJ are also supported (STEP and IGES are not supported on Mac and Linux versions).
Follow these steps to load a new 3D file:

1. Select **File→ Open**
2. Select file with your 3D object on your PC.
3. Alternatively, drag & drop files into RoboDK's main window to import them automatically.
4. Drag & drop the object by holding the right click to reorder items inside station tree.

You can also add new objects, such as a table, a box, a bottle, a floor to your project from RoboDK's default library on your PC (C:/RoboDK/Library) or from RoboDK's online library.

Select **File→Open robot library** (Ctrl+Shift+O) or select the corresponding button in the toolbar
Select **Object** from **Type** menu on the left to see objects available in RoboDK's online library.

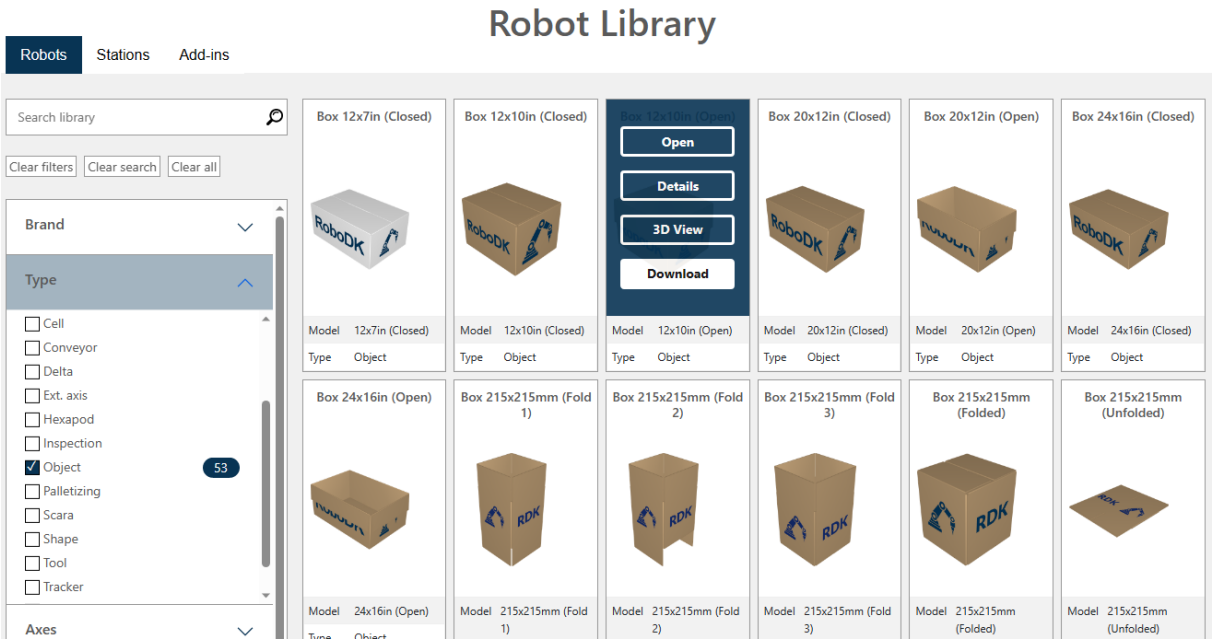


Figure 20.1. RoboDK's online library

Select object and click **Download.** Selected object should automatically appear in your project in a few seconds. The online library can be closed once the object is loaded.

Each object has an object frame that helps to place object relative to another object, robot or another reference frame. If you want, you can change position of object frame in imported object. Double click on the object to open the **Object Details** window and click **More options**. You can then enter coordinate values using **Move geometry** option to change position of the object frame. Next click **Apply move** to confirm changes and redefine the origin position of the object frame.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

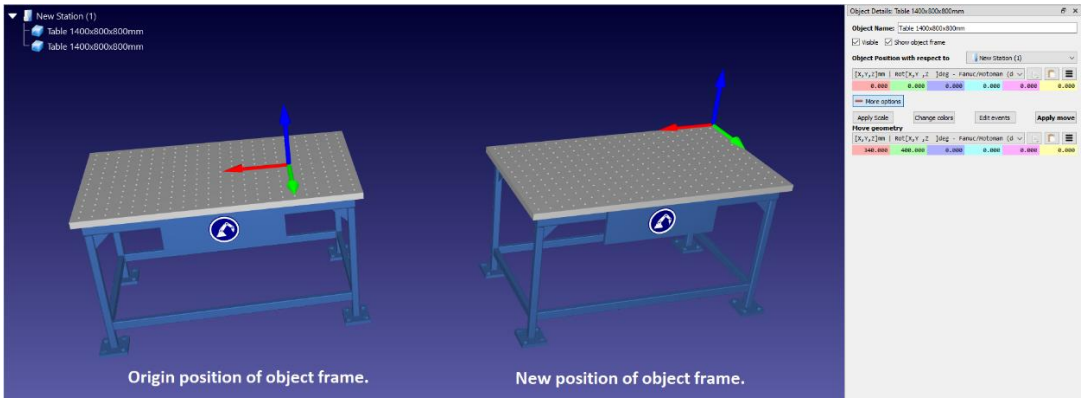


Figure 20.2. Different object frame positions for the same table

It is also possible to change scale or color of imported object in the same Object Details window. Click **Apply Scale** to change dimensions (scale) of imported object. Apply scale window will appear. Then enter scale ratio in the window and confirm changes by clicking OK.

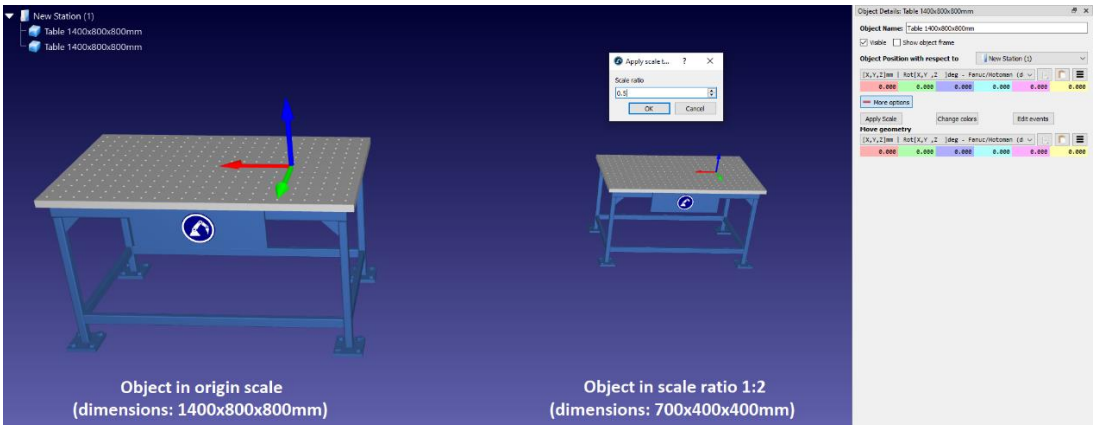


Figure 20.3. The same table in different scales

Click **Change colors** to change color of an imported object. Change colors window will appear. Then click the object and select color which you want to change in Change colors window, because it is possible to change each color separately. Select a color window will appear. Then select new color and confirm changes by clicking OK.

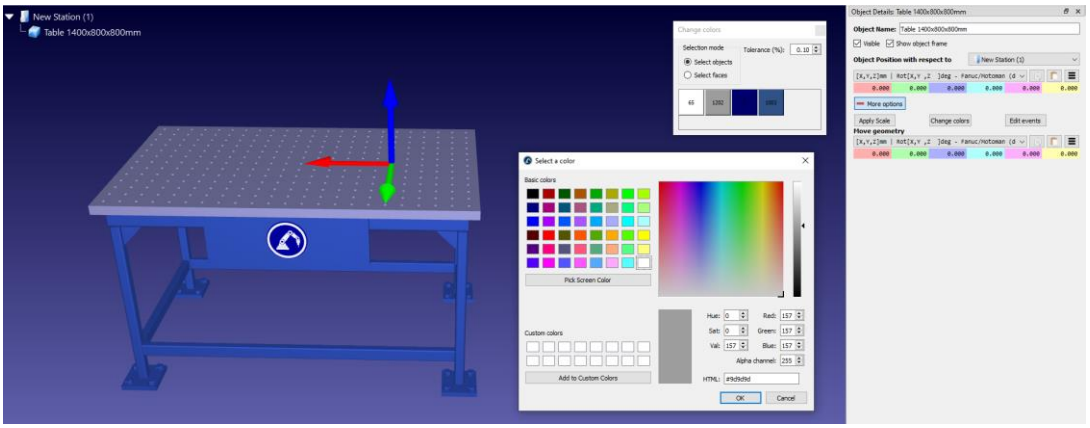


Figure 20.4. Color change steps

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

WORKSHEET 5

CREATE A WORKING AREA FOR ROBOT

The aims at the end of this worksheet are:

The student can add 3D objects to the Main Screen.

The student can change 3D object position with respect to reference frame.

The student can change scale and color of 3D objects.

Process Steps:

1- Open the program by double-clicking the program icon on the desktop 

.

2- Select the UR10 robot from the File / Open Robot Library menu and bring it to the worksheet.

3- Select the RobotiQ EPick Vacuum Gripper (1 Cup) from the File / Open Robot Library menu and bring it to the worksheet. The robot tool will be added at the UR10 Base frame automatically.

4- Let's bring a new frame to the screen by pressing the  Frame button. It appears in the Station Tree as Frame2. Then let's rename Frame2's name to World using the F2 key.

5- Select the Table 2000x1200x800mm from the File / Open Robot Library menu and bring it to the worksheet.

6- Let's choose the Table 2000x1200x800mm in the Station Tree and choose the **Change support** command from the menu that opens with the right button. Let's click on the World frame line. From now the Table 2000x1200x800mm is under the World frame.

7- Double click on the Table 2000x1200x800mm in the Station Tree to open the Object Details window. Then change coordinate values XYZ to 0 to centre the Table frame position with reference to the World frame, as shown in the figure below.

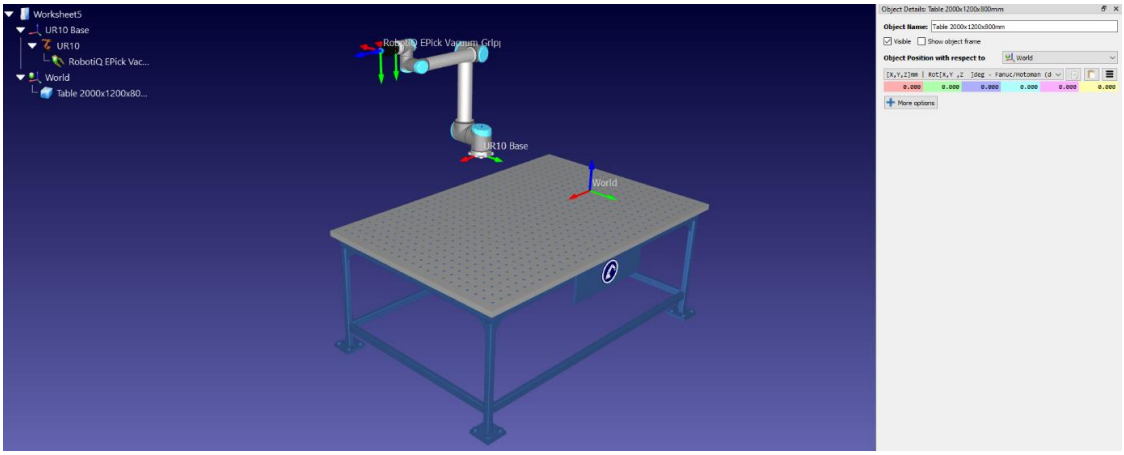


Figure 20.5. The Table moved to the World frame

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

8- Select the Floor from the File / Open Robot Library menu and bring it to the worksheet.

9- Let's select the Floor in the Station Tree and choose the **Change support** command from the menu that opens with the right button. Let's click on the World frame line. From now the Floor is under the World frame.

10- If the Floor appears in incorrect position, for example on the Table, as shown in the figure below, it will be necessary to change the position of the Floor with reference to the World frame.

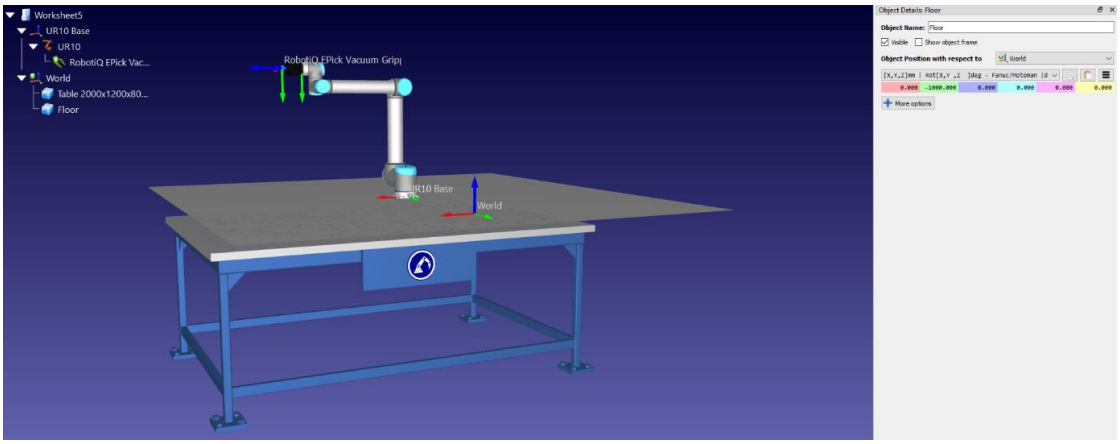


Figure 20.6. An example of incorrect position of the Floor – above the Table

Double click on the Floor in the Station Tree to open the Object Details window. Then change coordinate values XYZ to move the Floor to correct position.

Tip: the height of the Table is 800mm.

11- Let's increase the Floor surface. Click **More options** in the Object Details window and choose the **Apply Scale** option to change scale of the Floor from 1.0 to 1.5.

12- Let's change the Floor color. Click **More options** in the Object Details window and choose the **Change colors** option to change color of the Floor.

13- Let's place the UR10 robot on the Table. Double click on the UR10 Base frame in the Station Tree to open the Frame Details window. Then change coordinate values XYZ to move the UR10 robot to the position shown in the figure below.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

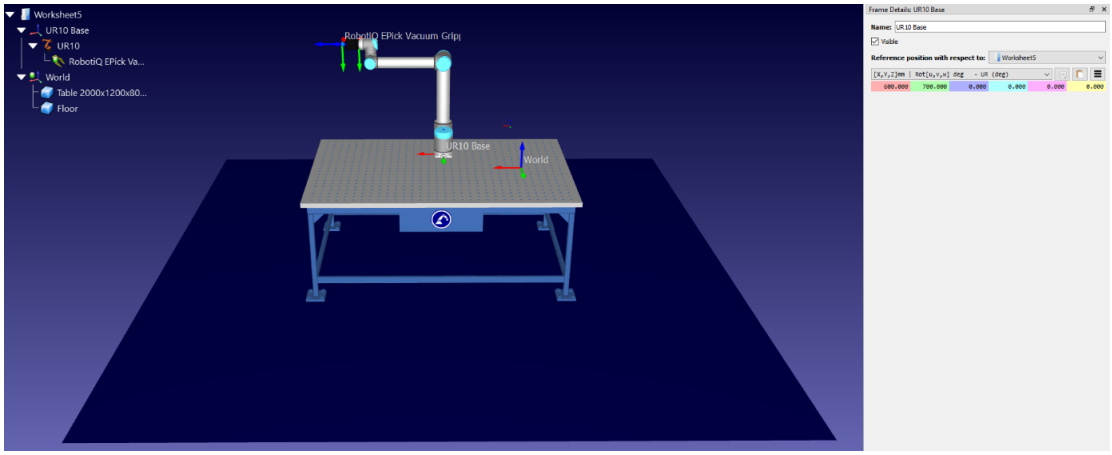


Figure 20.7. Placing the UR10 robot on the Table

14- Let's bring a new frame to the screen by pressing the  Frame button. It appears in the Station Tree as Frame3. Then let's rename Frame3's name to Parts using the F2 key.

15- Let's place the Parts frame on the Table. Double click on the Parts frame in the Station Tree to open the Frame Details window. Then change coordinate values XYZ to move the Parts frame to the position shown in the figure below.

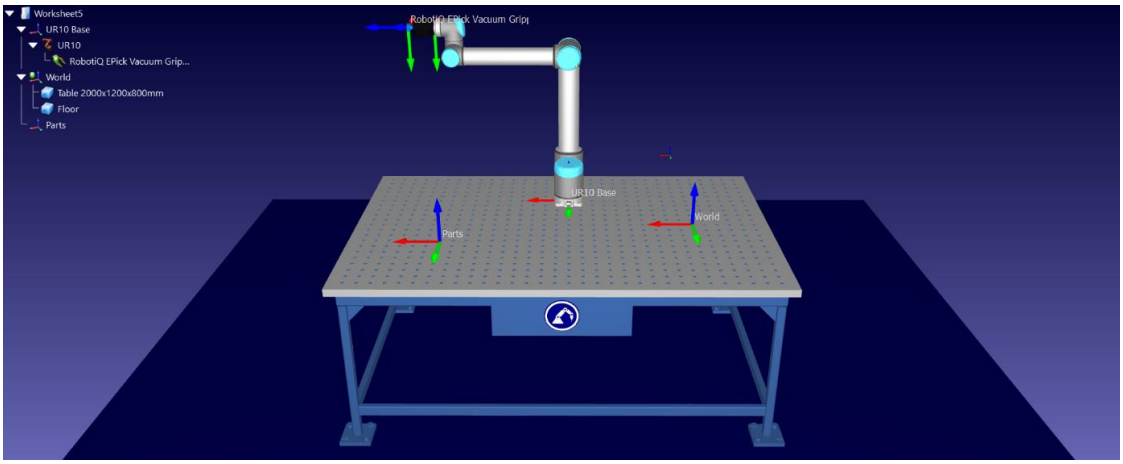


Figure 20.8. The Parts frame placed on the Table

16- Select **File➔**  **Open** to add new object from RoboDK's default library on your PC (C:/RoboDK/Library). Then select the **box** and bring it to the worksheet. Rename box's name to Box1 using the F2 key.

17- Let's select the Box1 in the Station Tree and choose the Change support command from the menu that opens with the right button. Let's click on the Frame Parts line. From now the Box1 is under the Parts frame.

18- Let's place the Box1 on the Table. Double click on the Box1 in the Station Tree to open the Object Details window. Then change coordinate values to [X, Y, Z] = [0, 0, **50**] mm to place the Box1 on the Table, in the Parts frame position.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

Tip: Check **Show object frame** option in the **Object Details** window to make the Box1 frame visible. As you can see in the figure below, the Box1 frame is located in the central point of the object, so you have to take that into account when entering coordinate values XYZ. You can also use **Move geometry** option to move position of the Box1 frame to the base of the object (the Box1 dimensions are 100x100x100mm).

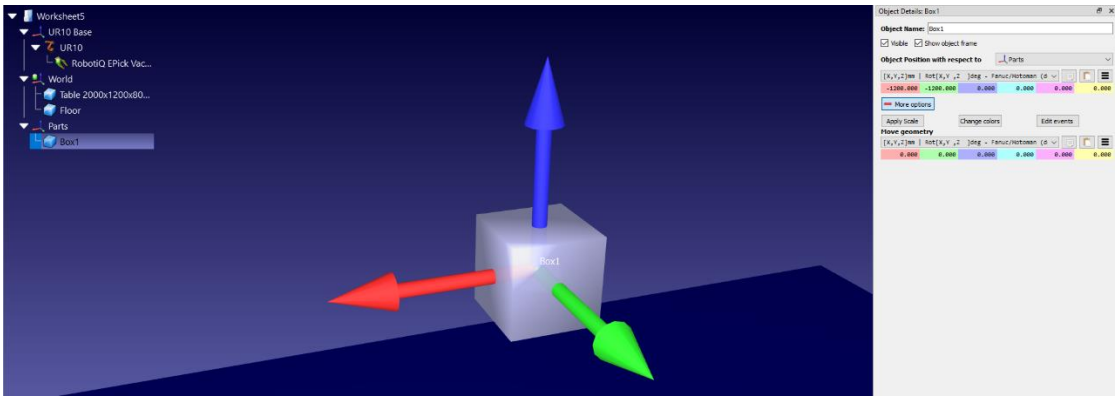


Figure 20.9. The Box1 frame location

19- Save your work as worksheet 5 by pressing the icon. 

20- Please add a new box to your project, rename it to Box2, place the Box2 next to the Box1 and change color of the Box2.

21- Save station you made with the Box2 under the name worksheet 5-1.

Study Question:

Please, check what other objects are available in the RoboDK Library.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

WORKSHEET 6

PICK AND PLACE APPLICATION PROGRAMMING

The aims at the end of this worksheet are:

The student can attach object to robot tool.

The student can detach object from robot tool.

The student can add a call to a sub program from the main program.

Process Steps:

1- Open the program by double-clicking the program icon on the desktop 

.

2- Press the  button to open the previous worksheet you created (worksheet 5). Your station should look like in the figure below.

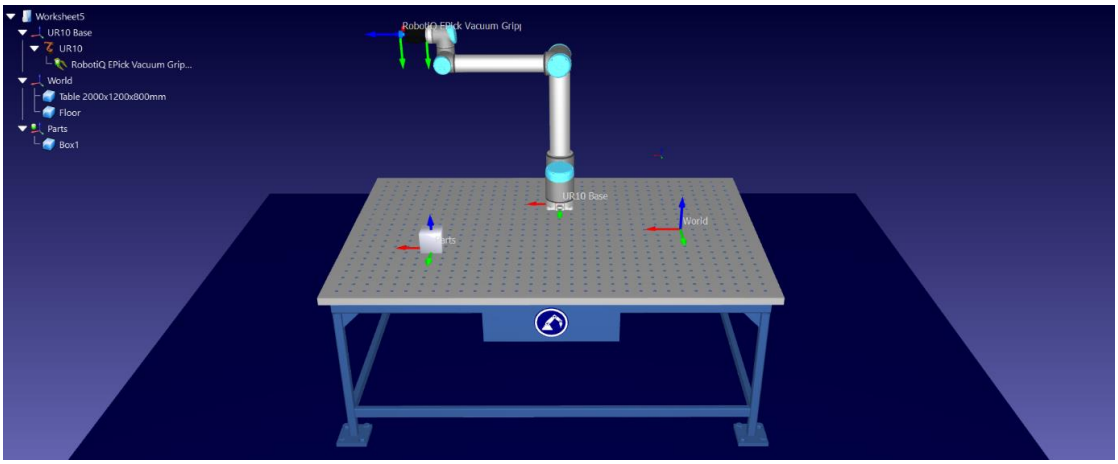


Figure 20.10. The station from worksheet 5

3- At the beginning please add a new program by pressing the  button. See that the program you added comes to the Station Tree as Prog1. Point to the Prog1 line and rename it to **Replace object** by pressing the F2 key. Program Replace object will help us to bring the Box1 back to origin position at any time.

Right click on the Replace object line in the Station Tree. Replace object program menu will appear. Choose **Add Instruction** from the menu, then click **Simulation Event Instruction**. The Event Instruction window will appear. Select **Set object position (relative)** as **Action** in the window, then choose the Box1 from the list of objects and confirm changes by clicking OK. From now double-clicking on the Replace object line in the Station Tree will bring the Box1 back to origin position.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

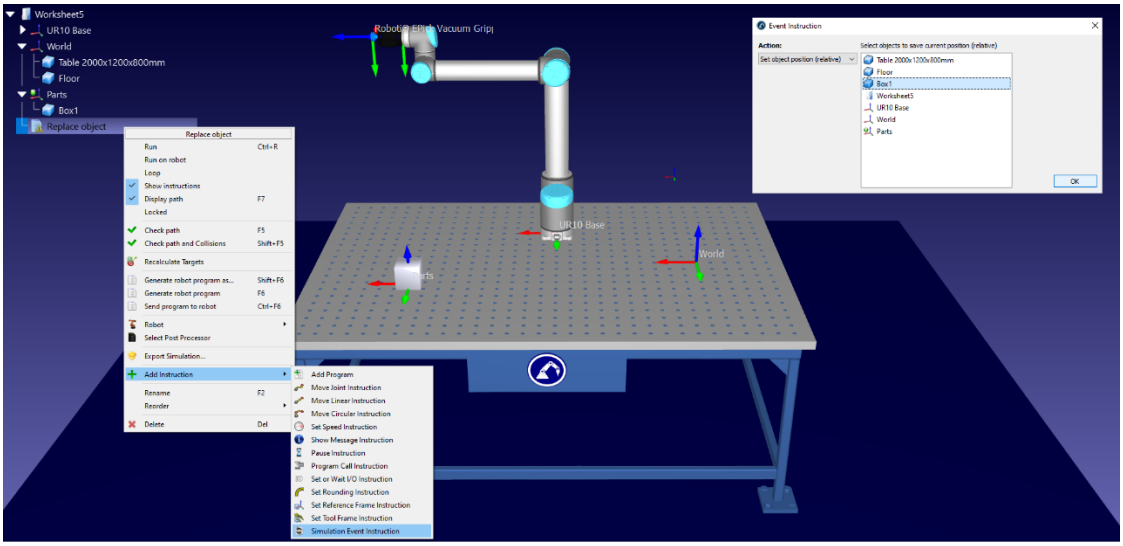


Figure 20.11. Adding the Simulation Event Instruction to the Replace object program

4- Let's create some targets for pick and place application. Move UR10 robot's joints with the frame arrows that appear on the screen by pressing the  button.

First try to move the robot so that the RobotiQ EPick Vacuum Gripper is placed on the center of the top of the Box1, as shown in the figure below.

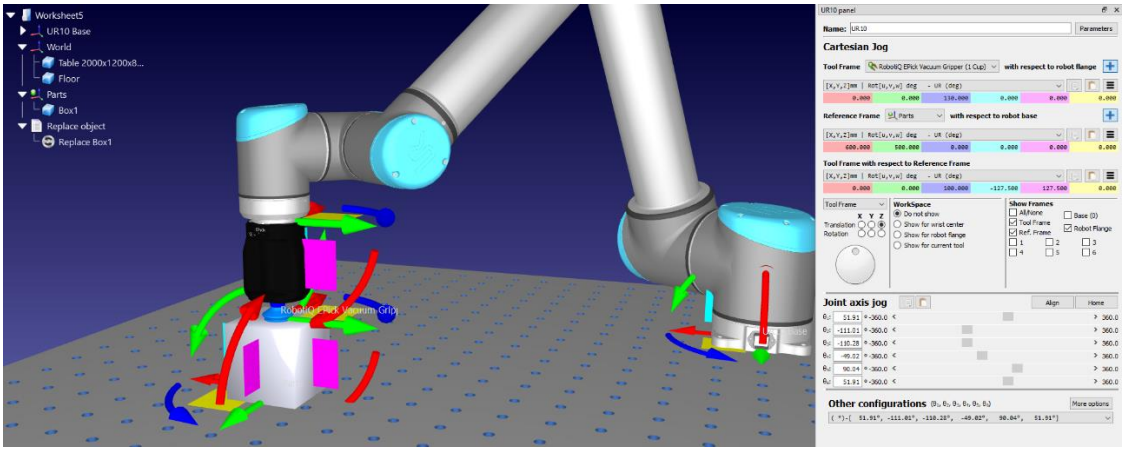


Figure 20.12. The vacuum gripper positioning

Use the UR10 panel to correct the position of the RobotiQ EPick Vacuum Gripper (double click on the robot to open the robot panel). Select the **Parts** frame as a **Reference Frame with respect to robot base** in the UR10 panel, then change coordinate values of **Tool Frame with respect to Reference Frame**.

When you reach the desired point, press the  button to save the target. Target 1 will appear in the Station Tree. Change the name of the target from Target 1 to **Pick**.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

5- Create next 3 targets using the UR10 panel and the button:

Approach_Pick – moved 200mm in Z axis from the Pick target.

Place – moved -1000mm in X axis from the Pick target.

Approach_Place – moved 200mm in Z axis from the Place target.

All 4 targets should be placed as shown in the figure below.

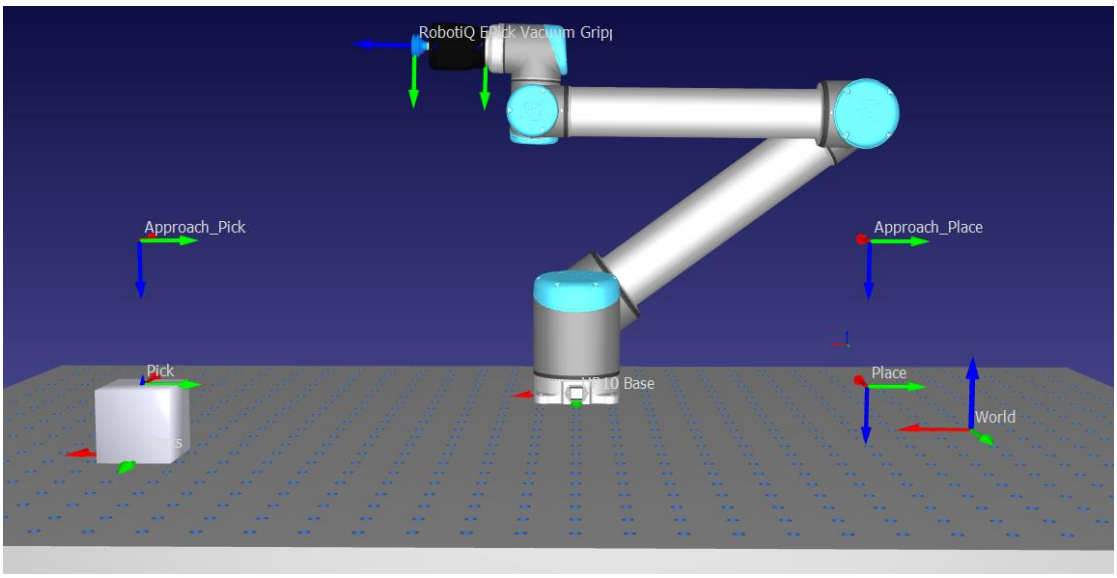


Figure 20.13. Location of the targets

6- Double click on the Approach_Pick target in the Station Tree, then add a new program by pressing the button and rename added program to **Pick_Program** by pressing the F2 key. Set Ref, Set Tool and MoveJ lines will appear automatically in the Pick_Program.

7- Select the Pick target and the Pick_Program in the Station Tree with the CTRL key. Press the Linear Move button. After this only MoveL (Pick) command comes to the Pick_Program.

8- When we double click on the Pick_Program, we see that our robot moves to Approach_Pick and then to the Pick targets. The RobotiQ EPick Vacuum Gripper should then grab the Box1. We will do this using Simulation Event Instruction, so press the button to open the Event Instruction window. Select **Attach object** as **Action** in the window, then select **TCP vs. Object Surface (list)** as **Measure distance** and choose the Box1 from the list of objects and confirm changes by clicking OK. From now the Box1 is attached to The RobotiQ EPick Vacuum Gripper.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

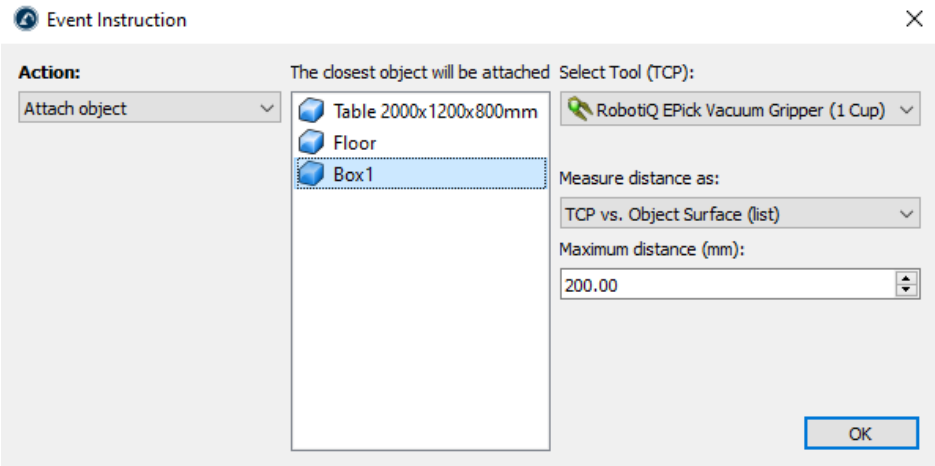


Figure 20.14. The Event Instruction window for the Attach object action

9- Now our robot should pick the Box1 up to Approach_Pick position. Select the Approach_Pick target and the Pick_Program in the Station Tree with the CTRL key. Press the Linear Move button. After this only MoveL (Approach_Pick) command comes to the Pick_Program. The Pick_Program is completed and should look like in the figure below.

Figure 20.15. The Pick_Program

10- Double click on the Approach_Place target in the Station Tree, then add a new program by pressing the button and rename added program to **Place_Program** by pressing the F2 key. Set Ref, Set Tool and MoveJ lines will appear automatically in the Place_Program.

11- Select the Place target and the Place_Program in the Station Tree with the CTRL key. Press the Linear Move button. After this only MoveL (Place) command comes to the Place_Program.

12- When we double click on the Pick_Program and then double click on the Place_Program, we see that our robot moves the Box1 from origin position to the Approach_Place and then to the Place targets. The RobotiQ EPick Vacuum Gripper should then leave the Box1 in Place position. We will do this using Simulation Event Instruction again, so press the button to open the Event Instruction window. This time select **Detach object** as **Action** in the window, then select **Parts** as **Attach to parent** and confirm changes by clicking OK. From now the Box1 is detached form the RobotiQ EPick Vacuum Gripper and attached to the Parts frame.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

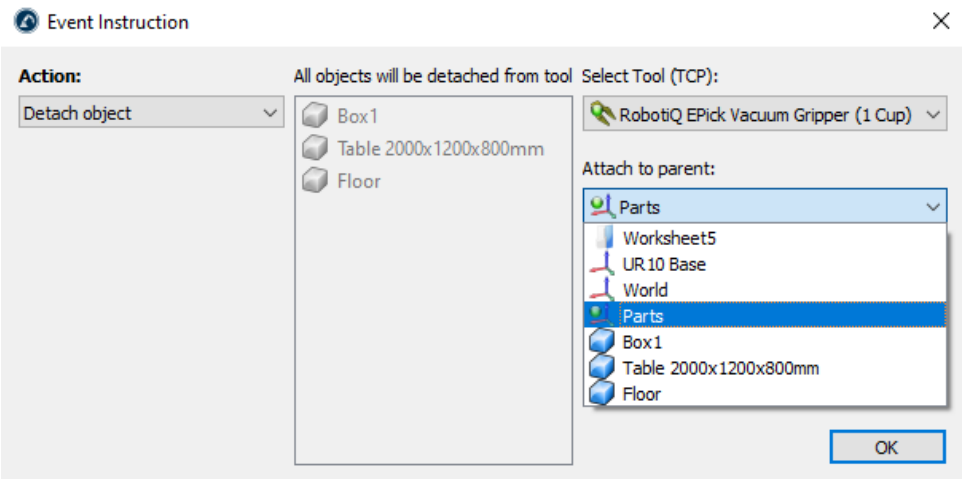


Figure 20.16. The Event Instruction window for the Detach object action

13- At the end of our pick and place application the UR10 robot should retreat to the Approach_Place position. Select the Approach_Place target and the Place_Program in the Station Tree with the CTRL key. Press the  Linear Move button. After this only MoveL (Approach_Place) command comes to the Place_Program. The Place_Program is completed and should look like in the figure below.

Figure 20.17. The Place_Program

14- Let's create a main robot program that executes all our programs sequentially. Add a new program by pressing the  button and rename added program to **Main_Program** by pressing the F2 key.

15- Press the  button to open the Program Call Instruction window and click **Select program** in the window. Then select the Replace object as a program which we want to add as a first to the Main_Program. Confirm all changes by clicking OK.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

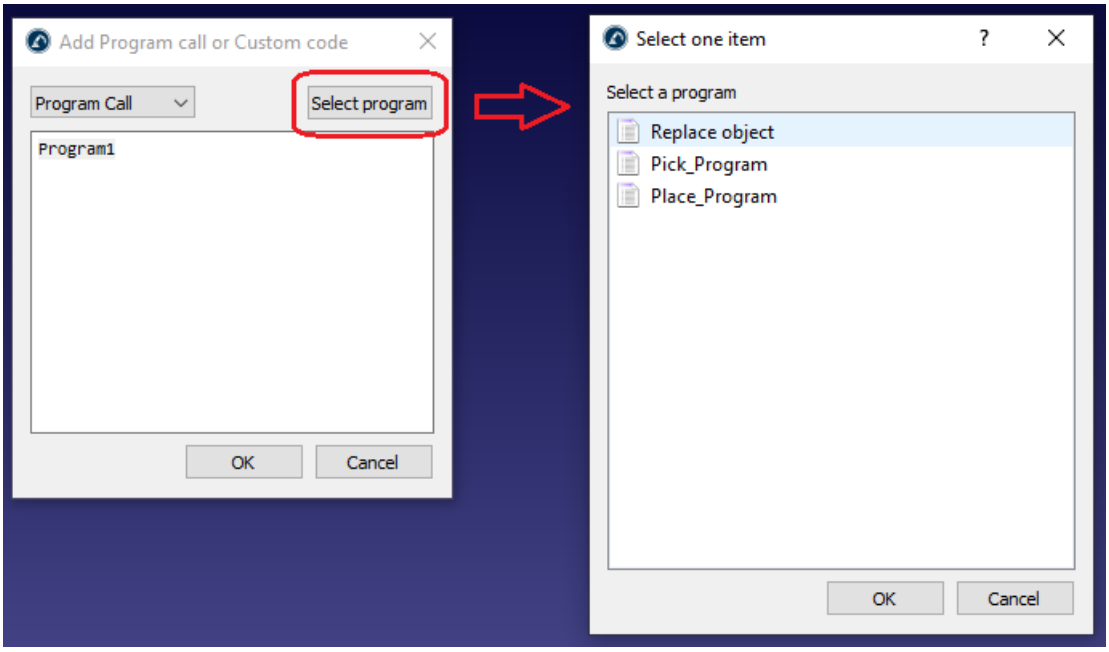


Figure 20.18. The Program Call Instruction window

Repeat the previous steps for the Pick_Program and then for the Place_Program. The Main_Program should look like in the figure below.

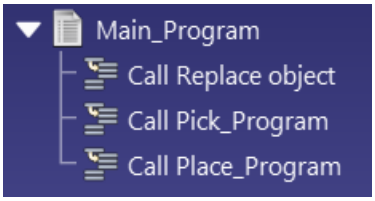


Figure 20.19. The Main_Program

16- Double-clicking the Main_Program will run the complete simulation. Right click the Main_Program and select **Loop** to make it simulate in a loop.

Tip: Uncheck the **Display path** option from the menu that appears when right-clicking on selected program in the Station Tree to hide the path which is creating by moving robot.

17- Select **File→Save station as...** to save your work as worksheet 6.

18- Please open worksheet 5-1 and try to program pick and place application with two boxes.

19- Save programmed pick and place application with two boxes under the name worksheet 6-1.

Study Question:

How to change the speed of the robot for selected movement from the program?

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

WORKSHEET 7

OPENING AND CLOSING OF THE ROBOTIQ GRIPPER MECHANISM

The aims at the end of this worksheet are:

The student can use the mechanical gripper in RoboDK projects.

The student can set the gripper opening level.

The student can create sub programs to open and to close the mechanical gripper.

Process Steps:

1- Open the program by double-clicking the program icon on the desktop 

.

2- Press the  button to open the previous worksheet you created – worksheet 5.

3- Let's remove the old robot tool from our project. Click the RobotiQ EPick Vacuum Gripper in the Station Tree and delete it by pressing the Del key.

4- Select the **RobotiQ-2F-140-Gripper-Mechanism** from the File / Open Robot Library menu and bring it to the worksheet.

5- RobotiQ 2F-140 Gripper (Mechanism) Base will appear in the Station Tree. Drag & drop it to the UR10 robot inside the Station Tree as shown in the figure in point 6.

6- Double click the RobotiQ 2F-140 Gripper (Mechanism) Base in the Station Tree to open Frame Details window. Then change coordinate values XYZ to 0 to move the RobotiQ 2F-140 Gripper (Mechanism) Base to the UR10 flange as shown in the figure below.

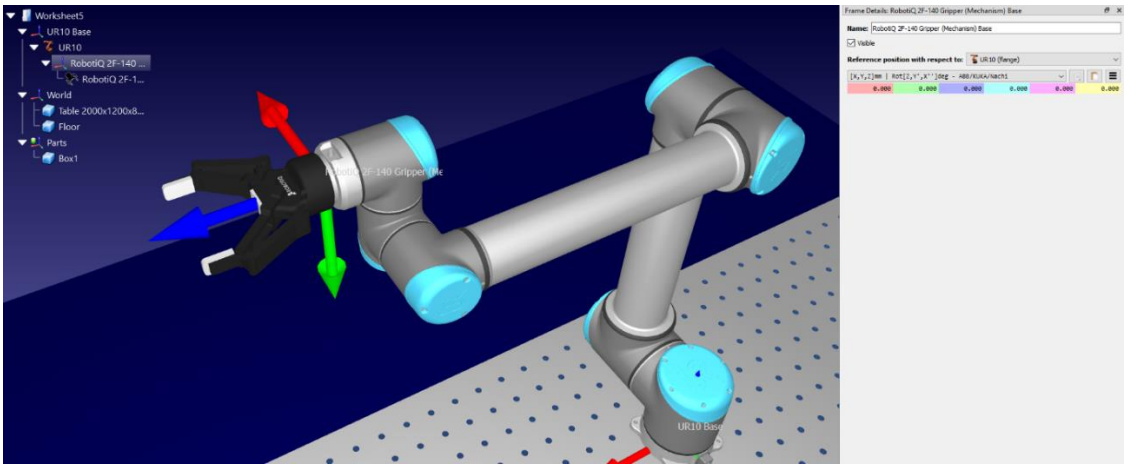


Figure 20.20. The RobotiQ 2F-140 Gripper-Mechanism mounted on the UR10 robot

7- Now we need to define the tool for the RobotiQ 2F-140 Gripper (Mechanism). Right click on the RobotiQ 2F-140 Gripper (Mechanism) line in the Station Tree. The RobotiQ 2F-140 Gripper

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

(Mechanism) menu will appear. Choose **Add Tool (TCP)** from the menu. **Tool 1** will appear in the Station Tree, and **Tool 1 TCP** will appear on the Main Screen as shown in figure below.

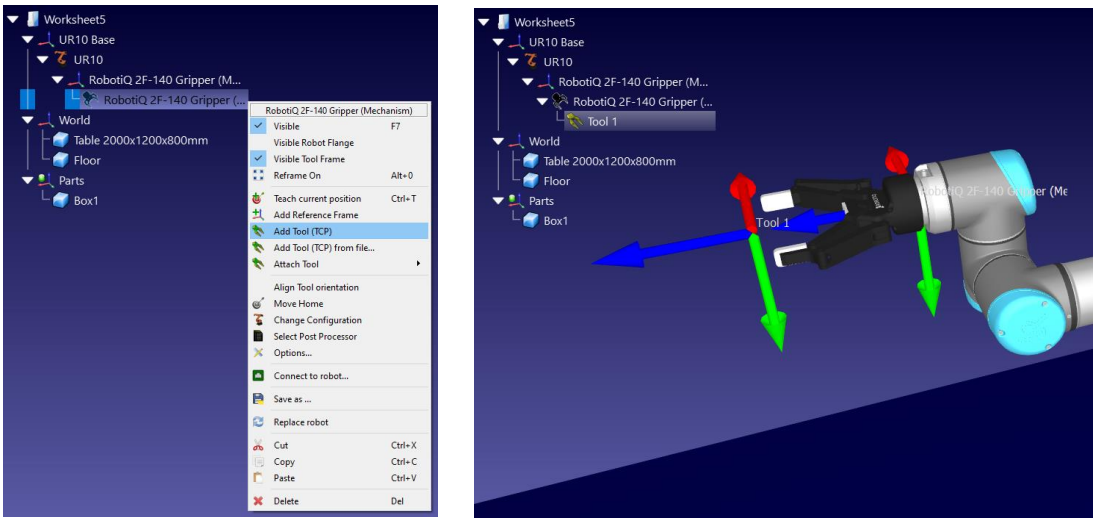


Figure 20.20. Tool definition for the RobotiQ 2F-140 Gripper (Mechanism)

8- By default, RoboDK will define the TCP at the position [X, Y, Z] = [0, 0, 200] mm. This can be changed by entering the coordinates manually. Double click the Tool 1 in the Station Tree to open the Tool Details window. Then change coordinate values to [X, Y, Z] = [0, 0, 155] mm to move the **Tool 1 TCP** a little closer to the RobotiQ 2F-140 Gripper (Mechanism) flange. In the same window, change the **Tool Name** form **Tool 1** to **Gripper**, uncheck **Show TCP** option and close the window.

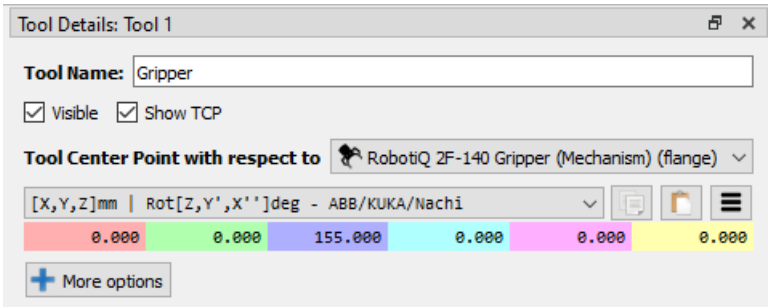


Figure 20.21. The Tool Details window for Tool 1

9- Right click on the RobotiQ 2F-140 Gripper (Mechanism) Base line in the Station Tree. The RobotiQ 2F-140 Gripper (Mechanism) Base menu will appear. Choose **Active Reference Frame** from the menu, as shown in the figure below.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

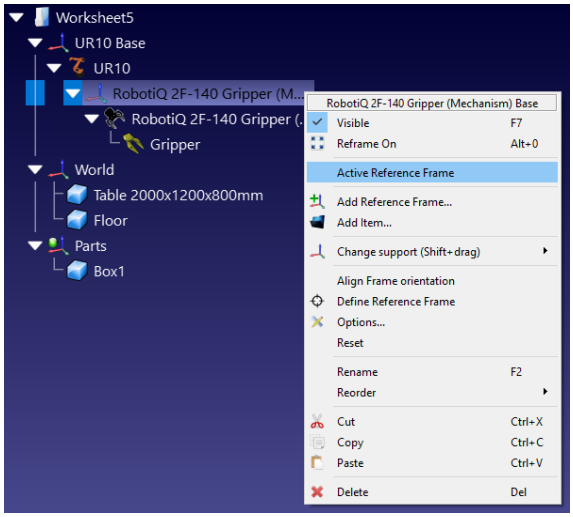


Figure 20.22. The RobotiQ 2F-140 Gripper (Mechanism) Base menu

10- Double click the RobotiQ 2F-140 Gripper (Mechanism) in the Station Tree to open the RobotiQ 2F-140 Gripper (Mechanism) panel window. In the window we can change the gripper opening level. For the RobotiQ 2F-140 Gripper (Mechanism) it is possible to change from 0 to 140 mm.

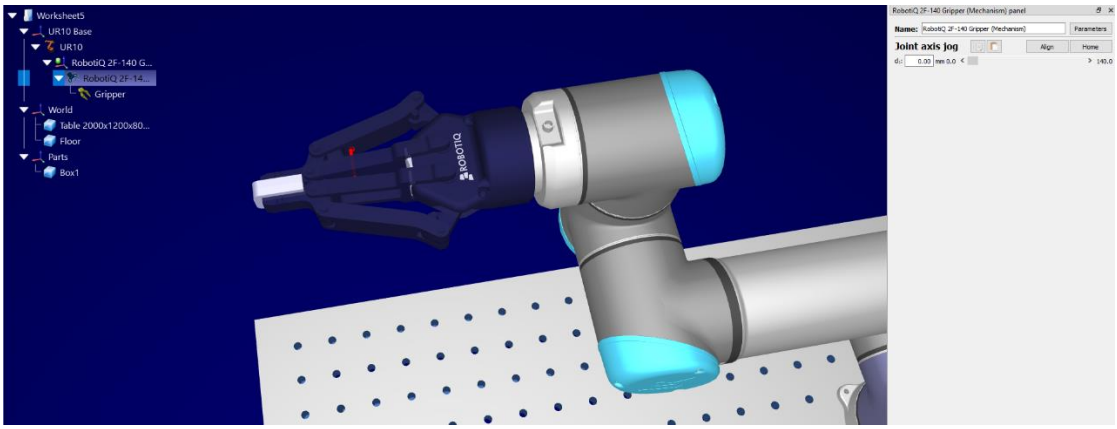


Figure 20.23. The RobotiQ 2F-140 Gripper (Mechanism) panel window

11- Set the gripper opening level to **140 mm** in the RobotiQ 2F-140 Gripper (Mechanism) panel window and then press the  button to create the target position for the open gripper. Target 1 should appear in the Station Tree, under the Gripper. Change the name of the target from Target 1 to **Open_Gripper**.

IMPORTANT: Close RobotiQ 2F-140 Gripper (Mechanism) panel window! If we create another target without closing the window first, only the first of created targets will work properly.

12- Open the RobotiQ 2F-140 Gripper (Mechanism) panel window again. We will pick and place the Box1 in next steps, so now set the gripper opening level to **100 mm** (the Box1 dimensions are 100x100x100mm) and then press the  button to create the target position for the

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

close gripper. Target 2 should appear in the Station Tree, under the Gripper. Change the name of the target from Target 2 to **Close_Gripper**.

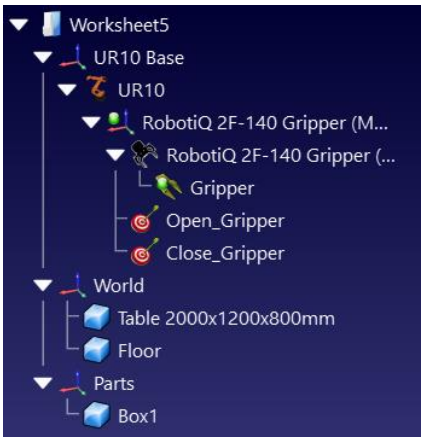


Figure 20.24. The Station Tree with the Open_Gripper and the Close_Gripper targets

13- Double click on the Open_Gripper target in the Station Tree, then add a new program by pressing the  button and rename added program to **Open_Gripper_Program** by pressing the F2 key. Set Ref, Set Tool and MoveJ lines will appear automatically in the Open_Gripper _Program.

14- Double click on the Close_Gripper target in the Station Tree, then add a new program by pressing the  button and rename added program to **Close_Gripper_Program** by pressing the F2 key. Set Ref, Set Tool and MoveJ lines will appear automatically in the Close_Gripper _Program.

Figure 20.25. The Open_Gripper_Program and the Close_Gripper_Program

15- Double-clicking the Close_Gripper_Program will run the gripper closing simulation and double-clicking the Open_Gripper _Program will run the gripper opening simulation. If it works correctly, we can add a call to the Close_Gripper _Program and the Open_Gripper _Program in any other program in our project.

16- Let's program pick and place application with created gripper.

At the beginning please create the **Replace object** program to bring the Box1 back to origin position at any time (see Worksheet 6, point 3).

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

17- Double click on the robot to open the robot panel. Select the **Parts** frame as a **Reference Frame with respect to robot base** in the UR10 panel, then create 4 targets by entering coordinate values of **Tool Frame with respect to Reference Frame** as below:

Pick	0,	0,	100,	180,	0,	0.
Approach_Pick	0,	0,	300,	180,	0,	0.
Place	-1000,	0,	100,	180,	0,	0.
Approach_Place	-1000,	0,	300,	180,	0,	0.

When you reach the desired point, press the  button to create the target and then rename created target.

18- Double-click on the Approach_Pick target in the Station Tree, then add a new program by pressing the  button and rename added program to **Pick_Program** by pressing the F2 key. Set Ref and MoveJ lines will appear automatically in the Pick_Program.

19- Select the Pick target and the Pick_Program in the Station Tree with the CTRL key. Press the  Linear Move button. After this only MoveL (Pick) command comes to the Pick_Program.

20- Press the  button to open the Program Call Instruction window and click **Select program** in the window. Then select Close_Gripper_Program as a program which we want to add to Pick_Program. Confirm all changes by clicking OK.

21- Press the  button to open the Event Instruction window. Select **Attach object** as **Action** in the window, then select **TCP vs. Object Surface (list)** as **Measure distance** and choose the Box1 from the list of objects and confirm changes by clicking OK.

22- Select the Approach_Pick target and the Pick_Program in the Station Tree with the CTRL key. Press the  Linear Move button. After this only MoveL (Approach_Pick) command comes to the Pick_Program. The Pick_Program is completed and should look like in figure below.

Figure 20.26. The Pick_Program

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

23- Double click on the Approach_Place target in the Station Tree, then add a new program by pressing the button and rename added program to **Place_Program** by pressing the F2 key. Set Ref and MoveJ lines will appear automatically in the Place_Program.

24- Select the Place target and the Place_Program in the Station Tree with the CTRL key. Press the Linear Move button. After this only MoveL (Place) command comes to the Place_Program.

25- Press the button to open the Program Call Instruction window and click **Select program** in the window. Then select the Open_Gripper_Program as a program which we want to add to Place_Program. Confirm all changes by clicking OK.

26- Press the button to open the Event Instruction window. This time select **Detach object** as **Action** in the window, then select **Parts** as **Attach to parent** and confirm changes by clicking OK.

27- Select the Approach_Place target and the Place_Program in the Station Tree with the CTRL key. Press the Linear Move button. After this only MoveL (Approach_Place) command comes to the Place_Program. The Place_Program is completed and should look like in figure below.

Figure 20.27. The Place_Program

28- Let's create a main robot program that executes all our programs sequentially. Add a new program by pressing the button and rename added program to **Main_Program** by pressing the F2 key.

29- Press the button to open the Program Call Instruction window and click **Select program** in the window. To ensure that the gripper is open at the start of the Main_Program, select the Open_Gripper_Program as a program which we want to add as a first to the Main_Program. Confirm all changes by clicking OK.

Repeat the previous steps for the Replace object program, the Pick_Program and the Place_Program.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

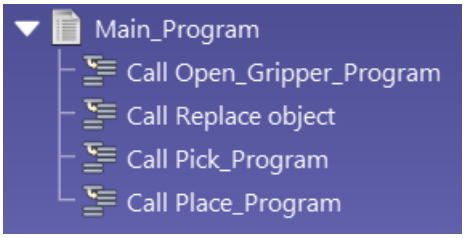


Figure 20.28. The Main_Program

30- Double-clicking the Main_Program will run the complete simulation. Right click the Main_Program and select **Loop** to make it simulate in a loop.

Tip: Uncheck the **Display path** option from the menu that appears when right-clicking on selected program in the Station Tree to hide the path which is creating by moving robot.

31- Select **File→Save station as...** to save your work as worksheet 7.

32- Try to program pick and place application with horizontal position of the RobotiQ-2F-140-Gripper-Mechanism as shown in the figure below.

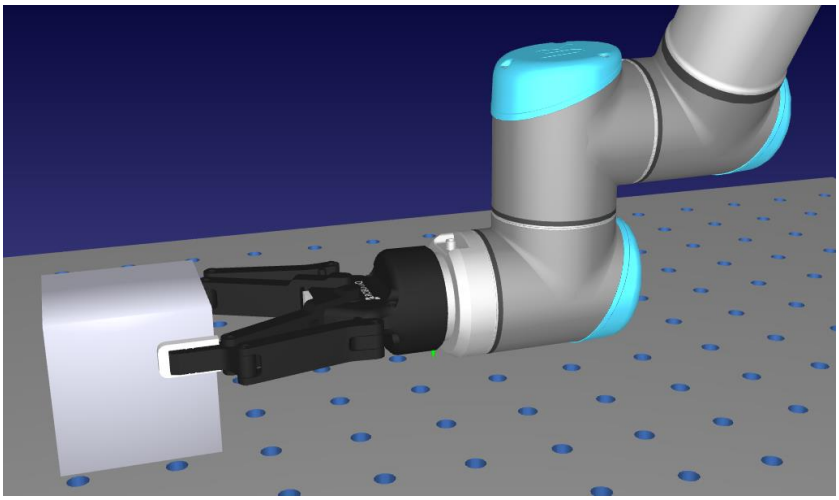


Figure 20.29. The RobotiQ-2F-140-Gripper-Mechanism in horizontal position

33- Save programmed pick and place application with horizontal position of the RobotiQ-2F-140-Gripper-Mechanism under the name worksheet 7-1.

Study Question:

Is there any other method to close and open the mechanical grippers used in RoboDK projects? Look at **Example-06.a-Pick and place – Mecademic** from RoboDK's default library on your PC (C:/RoboDK/Library).

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

21. COLLISION DETECTION

Collision checking with RoboDK can help us prevent collisions in our real setup. Collision checking can be used in different ways such as visually checking collisions, automatically avoid collisions for robot machining projects or generate a collision-free map to automatically create collision-free programs. In this section we will describe only visually checking collisions. **Remember that** virtual environment in RoboDK may not perfectly represent real setup. Therefore, it is recommended to account for a tolerance to safely prevent collisions. We can do so by loading larger and more simplified 3D models of our setup. For example, we could model spindle as a simple cube (used for collision checking only).

Select **Tools➔**  **Check collisions** to turn collision detection On or Off. If collision detection is activated, all programs and robot movements will stop when a collision is detected. All objects, tools and robot links in a collision state will be highlighted in red when the simulation is in a collision state.

If we prefer to continue simulating a program even if a collision is detected we can go to **Tools➔Options➔Motion** menu and uncheck the option **Stop robot movements when a collision is detected**.

Follow these steps to safely check a program for collisions:

1. Right click a program in the Station Tree.
2. Select **Check path and Collisions (Shift+F5)**. This option quickly checks if the path is feasible (same as **Check path – F5**) and then validates that there are no collisions.

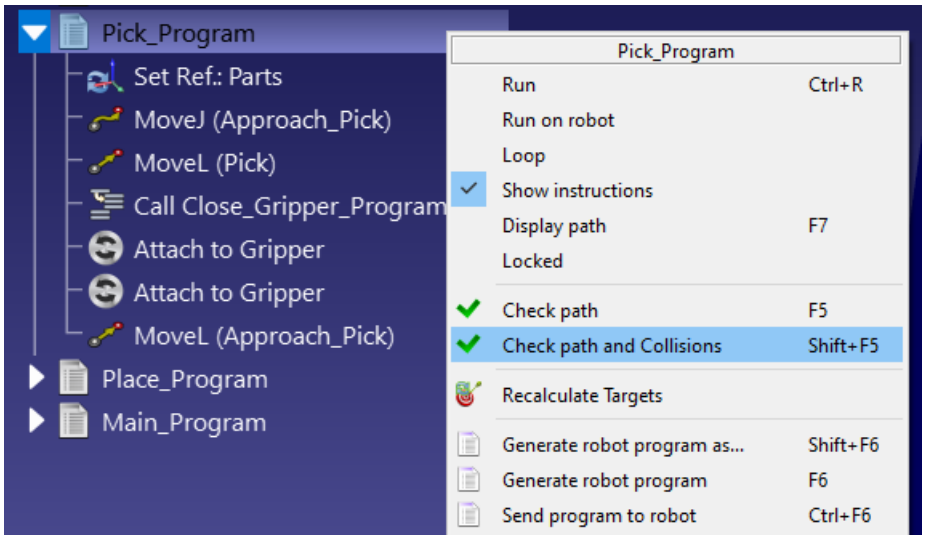


Figure 21.1. Activating the Check path and Collisions option from the program menu

We can specify if the interaction between any pair of objects needs to be checked for collisions.

Select **Tools➔**  **Collision map** to display the relationship between all moving objects in our cell and the collision check state. Double click a cell to activate or deactivate collision checking for that relationship. Select **Set default selection** to automatically set up a conservative selection.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

By default, RoboDK checks collisions between all moving objects in the station, including all robot links, objects and tools. As an exception, consecutive robot joints are not checked for collisions as they may always in contact.
Double click a cell in the diagonal to turn ON or OFF collision check for a specific object against all the other objects.

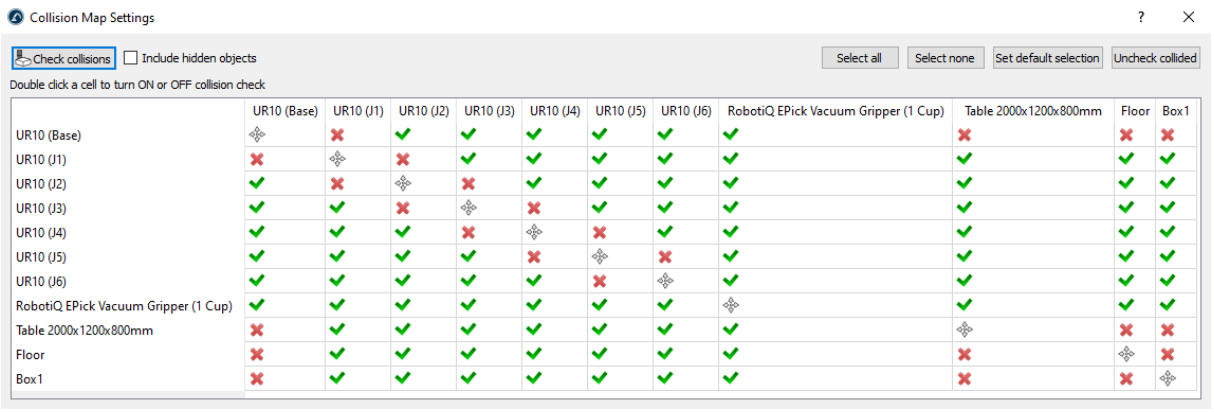


Figure 21.2. Collision Map Settings window

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

WORKSHEET 8

USING THE COLLISION DETECTION TOOL

The aims at the end of this worksheet are:

The student knows the meaning of the collision.

The student knows the types of collisions detected by the RoboDK collision detection tool.

The student can use the RoboDK collision detection tool.

Process Steps:

1- Open the program by double-clicking the program icon on the desktop 

.

2- Press the  button to open the previous worksheet you created – worksheet 6.

3- Let's add the second robot. Select the the ABB IRB 120-3/0.6 robot from the File / Open Robot Library menu and bring it to the worksheet.

4- Let's add a tool for the ABB robot. Select the Mirka AIROP 312NV Sander from the File / Open Robot Library menu and bring it to the worksheet.

5- Double click the ABB IRB 120-3/0.6 Base in the Station Tree to open Frame Details window. Then change coordinate values to [X, Y, Z] = [-100, 1200, 0] mm to move the ABB IRB 120-3/0.6 robot on the Table, as shown in the figure below.

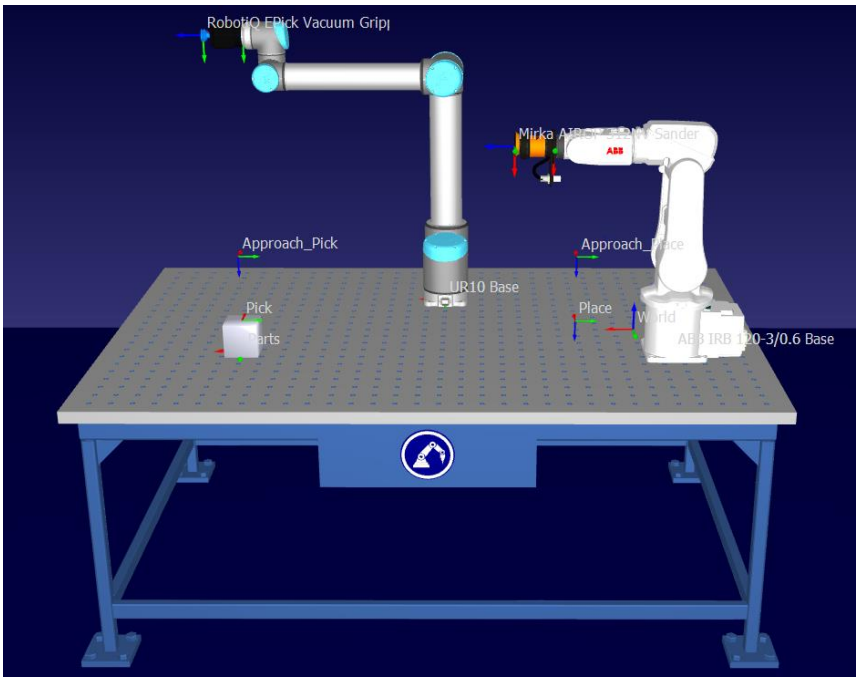


Figure 21.3. The station from worksheet 6 with added the ABB IRB 120-3/0.6 robot

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

6- Select **Tools➔Options➔Motion** menu and check the option **Stop robot movements when a collision is detected** if the option is unchecked.

7- Select **Tools➔**  **Check collisions** to turn collision detection On. From now all objects, tools and robot links in a collision state will be highlighted in red when the simulation is in a collision state.

8- Run the Main_Program and watch for potential collisions. The program will stop immediately, because two collisions were detected at the beginning of program, as shown in the figure below.

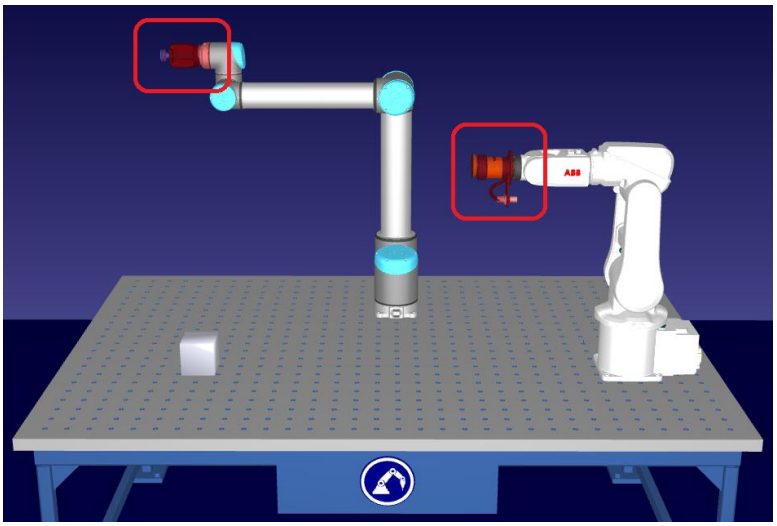


Figure 21.3. The first two collisions detected after running the Main_Program

9- Select **Tools➔**  **Collision map** to display the actual collision check state. Collisions are marked with an exclamation mark  .

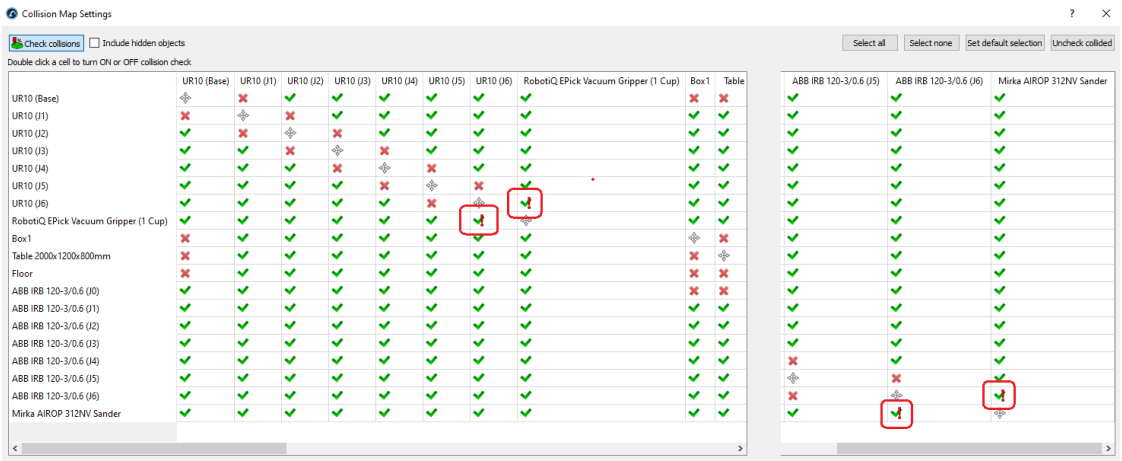


Figure 21.4. The first two collisions marked on the Collision Map

As you can see in the figure above, there are two collisions between:

- the RobotiQ EPick Vacuum Gripper and UR10 (J6),
- the Mirka AIROP 312NV Sander and ABB IRB 120-3/0.6 (J6).

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

These types of collision are obvious and should not be checked by RoboDK collision detection tool, so double click the cells to deactivate collision checking for these relationships. Then we can close the Collision Map Settings window.

10- Run the Main_Program again and watch for potential collisions. The program should stop when the RobotiQ EPick Vacuum Gripper touches the Box1, as shown in the figure below.

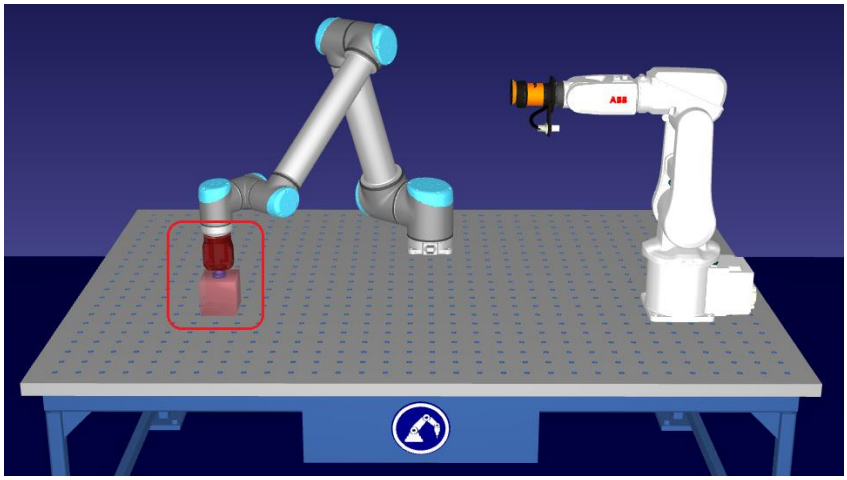


Figure 21.5. The third collision detected after running the Main_Program

11- Select **Tools➔ Collision map** again to display the actual collision check state.

There is collision between the RobotiQ EPick Vacuum Gripper and the Box1. This type of collision is obvious in pick and place application and should not be checked by RoboDK collision detection tool, so double click the cell to deactivate collision checking for this relationship. Then we can close the Collision Map Settings window.

12- Run the Main_Program again and watch for potential collisions. The program should stop when the UR10 robot hits the Mirka AIROP 312NV Sander, as shown in the figure below.

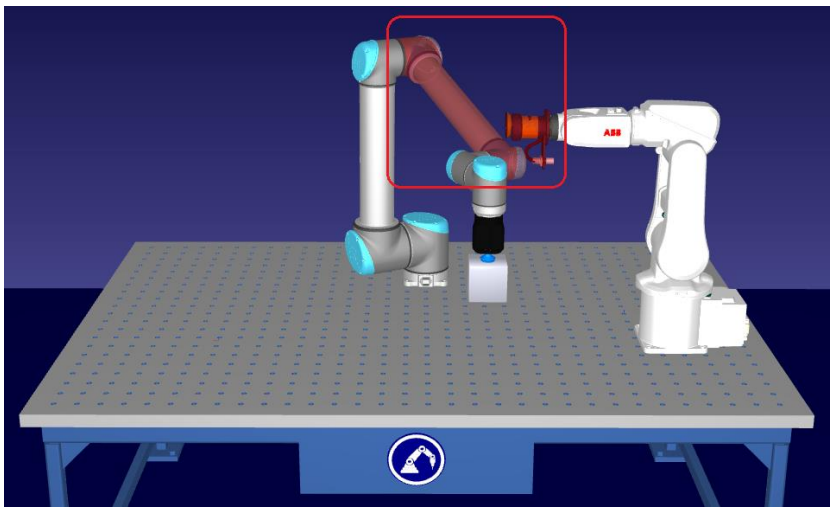


Figure 21.6. The fourth collision detected after running the Main_Program

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

13- Select **Tools➔** **Collision map** again to display the actual collision check state.

There is collision between the Mirka AIROP 312NV Sander and the UR10 (J3). This type of collision is unexpected and should be checked by the RoboDK collision detection tool, so we should not deactivate collision checking for this relationship. Close the Collision Map Settings window without making any changes.

14- Avoid collision between the Mirka AIROP 312NV Sander and the UR10 (J3) by moving the ABB IRB 120-3/0.6 robot to different position.

Double click the ABB IRB 120-3/0.6 robot to open the ABB IRB 120-3/0.6 panel. Then change position of Joint 1 from 0° to 90°, as shown in the figure below.

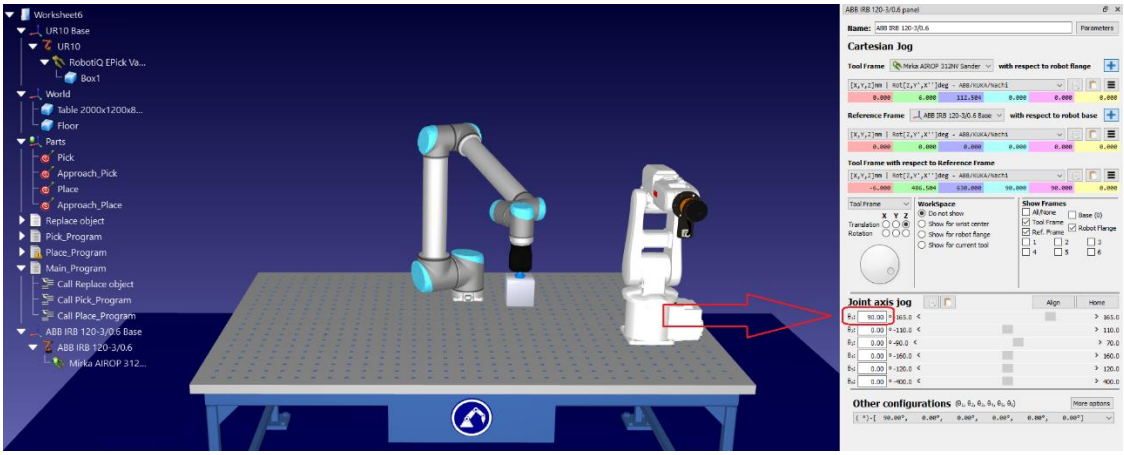


Figure 21.7. Moving the ABB IRB 120-3/0.6 robot to the new starting position

15- Run the Main_Program again. No more collision should be detected.

16- Select **File→Save station as...** to save your work as worksheet 8.

Study Question:

Take a look at our project and find another types of collisions which are obvious and should not be checked by the RoboDK collision detection tool. The Collision Map Settings window will help you find some examples.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

WORKSHEET 9

USING THE HIDE AND SHOW SIMULATION EVENT INSTRUCTIONS

The aims at the end of this worksheet are:

The student can create and program a more complex station in RoboDK.

The student can use the hide and show simulation event instructions in RoboDK to make the simulation more realistic.

Process Steps:

1- Open the program by double-clicking the program icon on the desktop 

.

2- Press the  button to open the previous worksheet you created – worksheet 8.

3- If collision detection is active press the  button to turn collision detection Off.

4- In the next steps, let's create a program in which the UR10 robot will pick up the part from the Pick point and move it to the Place point. The ABB robot will then sand the part, and then the UR10 will move the part back to the Pick Point. Of course, the part before sandering and after sanding should look different. To do this, we will use the hide and show simulation event instructions in our program.

5- Select **File➔**  **Open** to add a new object from RoboDK's default library on your PC (C:/RoboDK/Library). Then select the **box** and bring it to the worksheet. Rename box's name to Box2 using the F2 key.

The **Box2** will simulate a dirty part, which should be visible before sandering process.
The **Box1** will simulate a shining part, which should be visible after sandering process.

6- Let's choose the Box2 in the Station Tree and choose the **Change support** command from the menu that opens with the right button. Let's click on the Parts frame line. From now the Box2 is under the Parts frame.

7- Let's change the Box2 color to dark gray. Double click on the Box2 in the Station Tree to open the Object Details window. Click **More options** in the Object Details window and choose the **Change colors** option to change color of the Box2.

8- Let's place the Box2 on the Table, exactly in the same place as the Box1. Double click on the Box2 in the Station Tree to open the Object Details window. Then change coordinate values to [X, Y, Z] = [0, 0, 50] mm with respect to the Parts frame position.

Tip: If the boxes are in the same position, only one box will be visible on the Main Screen. Select the Box1 in the Station Tree and then select the Box2 in the Station Tree to see the actual position of the boxes and make sure they are in the same position.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

9- As we remember, the Replace object program in the Station Tree helps us to bring the Box1 back to origin position at any time. Now we have two boxes, so add the Box2 position to the Replace object program.

Select the Replace object line in the Station Tree, then press the button to open the Event Instruction window. Select **Set object position (relative)** as **Action** in the window, then choose the Box2 from the list of objects and confirm changes by clicking OK.

10- At the beginning of our main program the Box2 should be visible for us and the Box1 should be invisible. Let's do this by adding the show and hide simulation event instructions to the **Replace object** program.

Select again the Replace object line in the Station Tree, then press the button to open the Event Instruction window. Select **Show object/tool** as **Action** in the window, then choose the **Box2** from the list of objects and confirm changes by clicking OK.

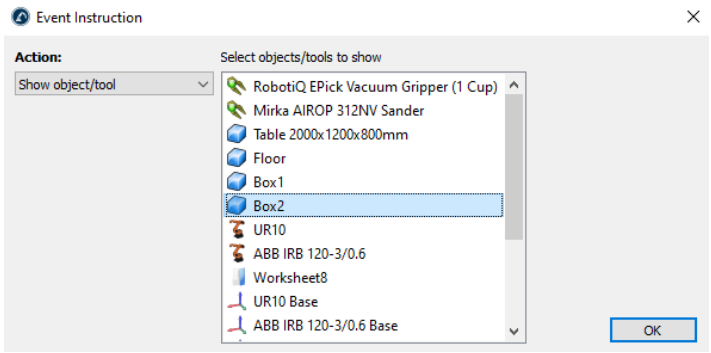


Figure 21.8. The Event Instruction window for the Show object/tool action

Select again the Replace object line in the Station Tree, then press the button to open the Event Instruction window. Select **Hide object/tool** as **Action** in the window, then choose the **Box1** from the list of objects and confirm changes by clicking OK.

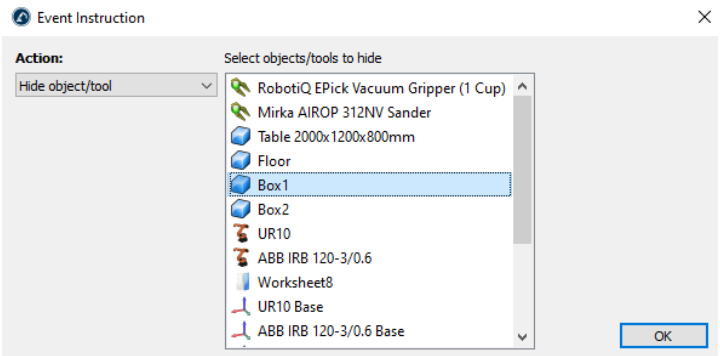


Figure 21.9. The Event Instruction window for the Hide object/tool action

The Replace object program should look like in the figure below.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

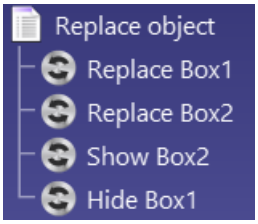


Figure 21.10. The Replace object program

From now double-clicking on the Replace object line in the Station Tree will bring the Box1 and the Box2 back to origin position and also show the Box2 and hide the Box1. Please check it – only the dark gray box should be visible on the Table, as shown in the figure below.

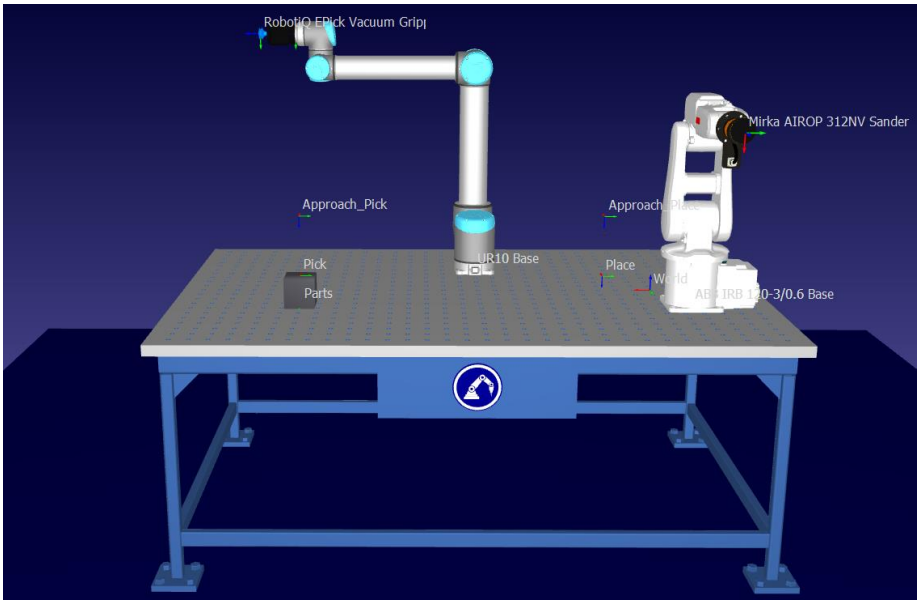


Figure 21.11. The station after running the Replace object program

11- In our new program, the RobotiQ EPick Vacuum Gripper has to grab the Box1 and the Box2 at the same time, so we need to change the Pick_Program and attach the Box2 to the RobotiQ EPick Vacuum Gripper also.

Select the **Attach to RobotiQ Epick Vacuu...** instruction line in the Pick_Program, then press the  button to open the Event Instruction window. Select **Attach object** as **Action** in the window, then select **TCP vs. Object Surface (list)** as **Measure distance** and choose Box2 from the list of objects and confirm changes by clicking OK. The new Attach instruction should appear in the Pick_Program.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

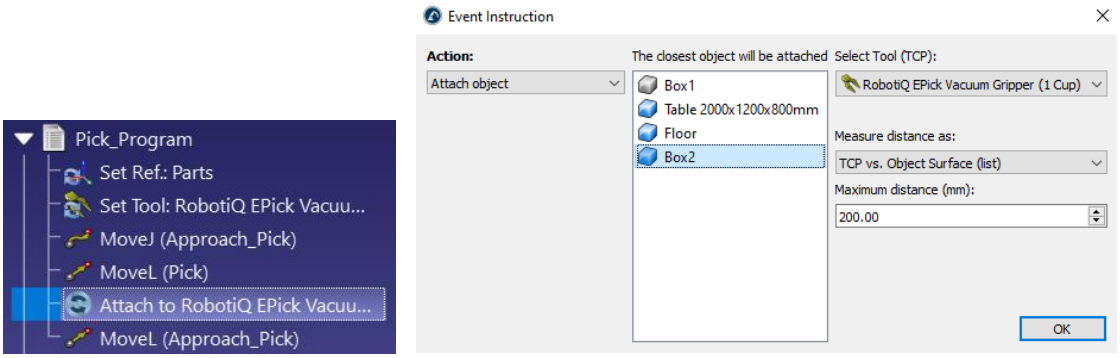


Figure 21.12. Adding the Attach object instruction for the Box2 to the Pick_Program

12- Run the Pick_Program. The UR10 robot should pick up the boxes. Only the Box2 is visible on the Main Screen, but if we look at the RobotiQ Epick Vacuum Gripper in the Station Tree we will see that there are two boxes attached to the gripper, as shown in the figure below.

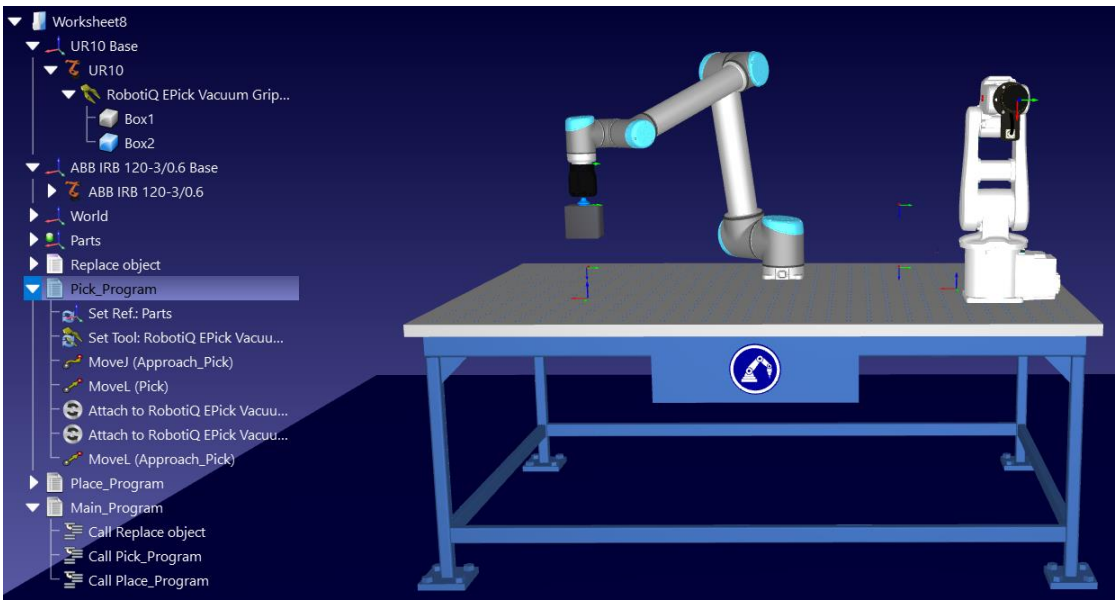


Figure 21.13. The station during the Pick_Program execution

13- There is the **Detach from RobotiQ Epick Vacuum** simulation event instruction in the Place_Program which we have added to detach the Box1 from the gripper. Detach instruction applies to all elements, which are currently attached to the gripper, so we don't have to change the Place_Program to detach the Box2 from the RobotiQ EPick Vacuum Gripper.

14- Run the following programs one by one: the Replace object, the Pick_Program and the Place_Program. At the end of this sequence, the UR10 robot should be in the Approach_Place target position and the boxes should be in the Place target position.

15- Move the UR10 robot to safe position before the ABB robot starts working to avoid collision. Double click the UR10 robot to open the UR10 panel. Then change position of Joint 1 to 90°

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

and press the  button to create the target position, as shown in the figure below. The Target 5 should appear in the Station Tree under the Parts frame. Rename the Target 5 to **UR10_wait** by pressing the F2 key.

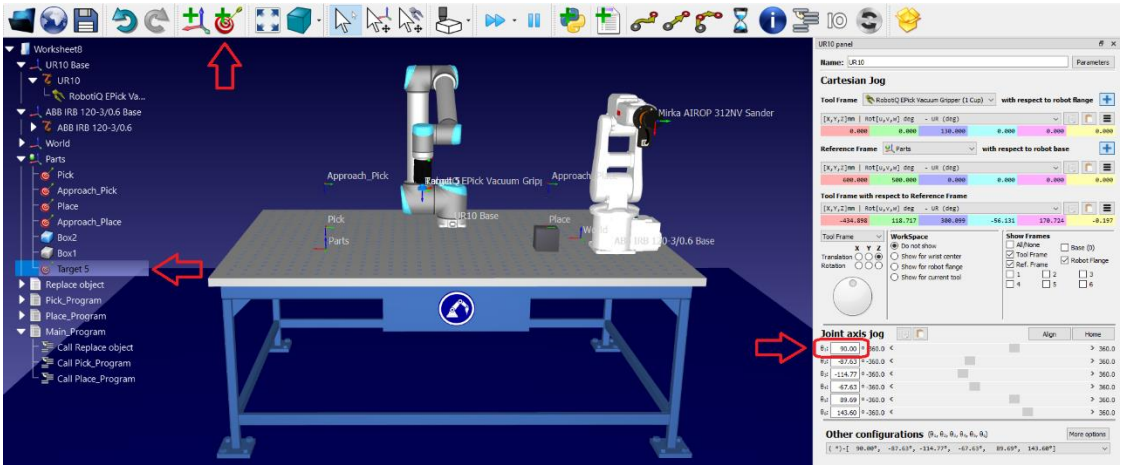


Figure 21.14. Creating the new target position (UR10_wait) for the UR10 robot

16- Let's add the target position UR10_wait to the end of the Place_Program. Select the UR10_wait target and the Place_Program in the Station Tree with the CTRL key.

Press the  Joint Move button. After this MoveJ (UR10_wait) command comes to the Place_Program.

17- Let's create the targets for the ABB robot. Double click on the ABB robot to open the ABB IRB 120-3/0.6 panel. Select the **Parts** frame as a **Reference Frame with respect to robot base** in the ABB IRB 120-3/0.6 panel.

18- First press the  button to create the target for the current position of the ABB robot. The Target 6 should appear in the Station Tree under the Parts frame. Rename the Target 6 to **ABB_Home** by pressing the F2 key.

19- Create 5 more targets by entering coordinate values of **Tool Frame with respect to Reference Frame** in the ABB IRB 120-3/0.6 panel as below:

ABB_Approach	-1000,	0,	200,	90,	180,	0.
Sander1	-980,	-30,	100,	90,	180,	0.
Sander2	-1030,	-30,	100,	90,	180,	0.
Sander3	-1030,	20,	100,	90,	180,	0.
Sander4	-980,	20,	100,	90,	180,	0.

When you reach the desired point, click the ABB robot on the Main Screen and only then press the  button to create the target and then rename created target.

IMPORTANT: If we do not click the ABB robot before pressing the button, the target may be assigned to the UR10 robot.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

20- Let's create a program for the ABB robot. Double-click on the target ABB_Approach in the Station Tree, then add a new program by pressing the button and rename added program to **ABB_Program** by pressing the F2 key. Set Ref, Set Tool and MoveJ lines will appear automatically in the ABB_Program.

21- Select the ABB_Approach target and the ABB_Program in the Station Tree with the CTRL key. Press the Joint Move button to add MoveJ (ABB_Approach) command to the ABB_Program.

IMPORTANT: The boxes are very close to the ABB robot, so do not use Linear Move button. The robot may not be able to make a linear movement for some targets bacause of joints limit.

22- Repeat the steps from the previous point for the next targets to create the ABB_Program as in the figure below.

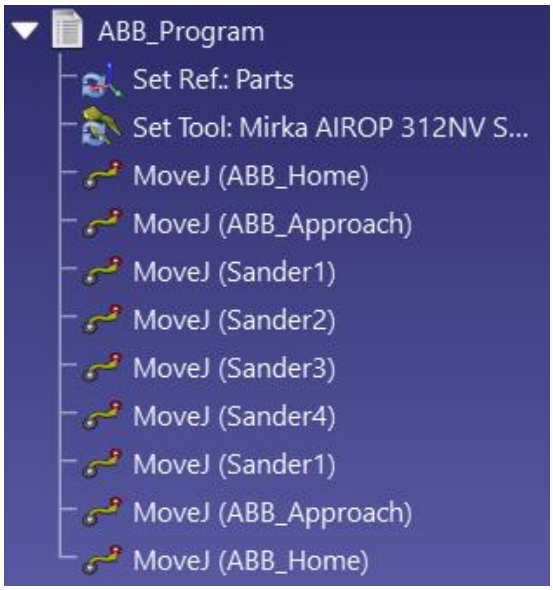


Figure 21.15. The ABB_Program with added targets

23- Select the MoveJ(Sander4) in the ABB_Program, then press the button to open the Event Instruction window. Select **Show object/tool** as **Action** in the window, then choose the **Box1** from the list of objects and confirm changes by clicking OK. The **Show Box1** instruction should appear after the MoveJ(Sander4) line in the ABB_Program.

24- Select the Show Box1 line in the ABB_Program, then press the button to open the Event Instruction window. Select **Hide object/tool** as **Action** in the window, then choose the **Box2** from the list of objects and confirm changes by clicking OK. The **Hide Box2** instruction should appear after the Show Box1 line in the ABB_Program.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

25- The ABB_Program is completed and should look like in the figure below. Let's add the ABB_Program to the Main_Program. Select the Main_Program in the Station Tree, then press the  button to open the Program Call Instruction window and click **Select program** in the window. Then select the ABB_Program as a program which we want to add to the Main_Program. Confirm all changes by clicking OK.

Figure 21.16. The ABB_Program with added the show/hide instructions

26- Let's create a next program for the UR10 robot to take the Box1 and the Box2 back to the Pick target position. Double click on the Approach_Place target in the Station Tree, then add a new program by pressing the  button and rename added program to **Back_Program** by pressing the F2 key. Add next instructions to create the Back_Program as shown in the figure below.

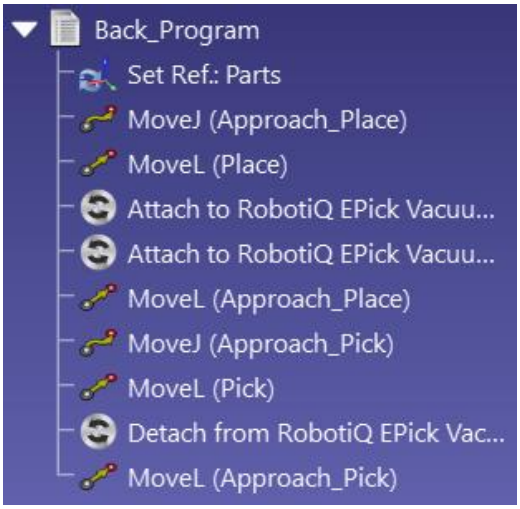


Figure 21.17. The Back_Program

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

27- Finally, add the Back_Program to the Main_Program.

28- Run the Main_Program and enjoy it ☺.

29- Select **File→Save station as...** to save your work as worksheet 9.

30- Select **Tools➔**  **Check collisions** to turn collision detection On.

31- Run the Main_Program again and watch for potential collisions.

Study Question:

Where else can we use the hide and show event instrictions? Give some examples.
Look at **Example-06.f-Pick and Place CNC and Dual RobotiQ Gripper-UR10** from RoboDK's default library on your PC (C:/RoboDK/Library).

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

22. GENERATE ROBOT PROGRAM

Once you have the simulation ready in RoboDK you can easily generate the robot program, so you can execute the program on the robot controller without having to write a single line of code. The conversion from the RoboDK simulation to a specific robot program is done by a Post Processor. The Post Processor defines how robot programs should be generated for a specific robot. Each robot has a specific/default post processor by default in RoboDK.

You can export any program individually or the main program including the sub programs.

Follow these steps to generate the robot program required by the robot controller:

1. Select the **Program**
2. Select **Program→Generate Program(s)** (F6)

Alternatively, right click a program and select **Generate robot program (F6)** to generate the program, as shown in the figure below.

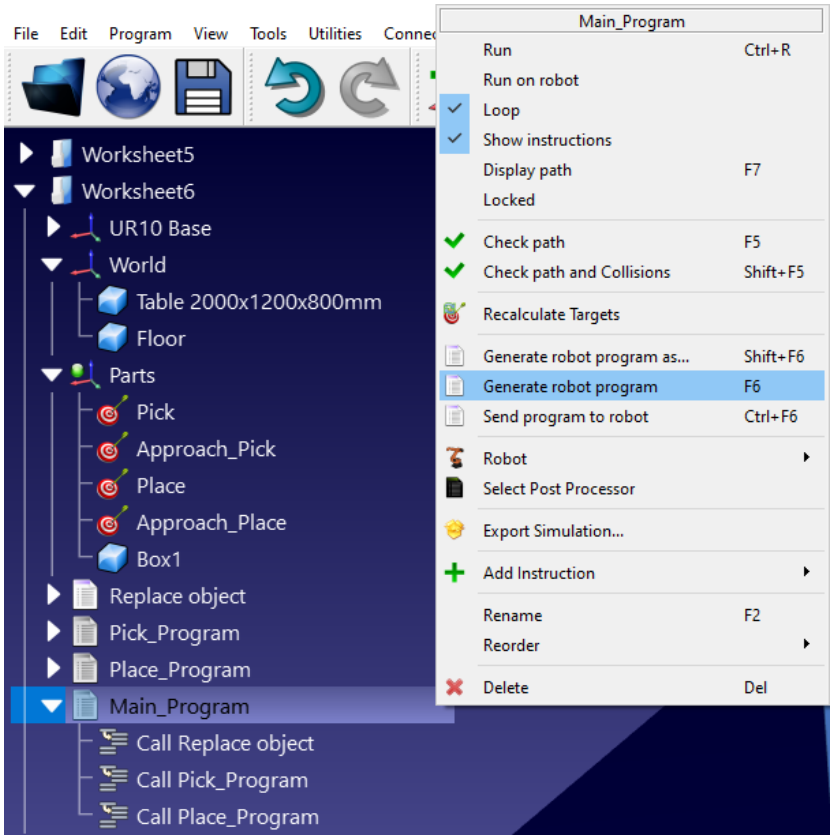


Figure 22.1. Generating robot program from the program menu

Multiple programs can be selected to generate more than one program at a time. Hold the Ctrl key to select more than one program. Selecting the **Generate Program(s)… (Shift+F6)** option will open a window asking the user to provide a location to save the program.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

The file you obtain is the result of generating the program offline. The file can be sent to the robot controller to run the same movements that were simulated in RoboDK.

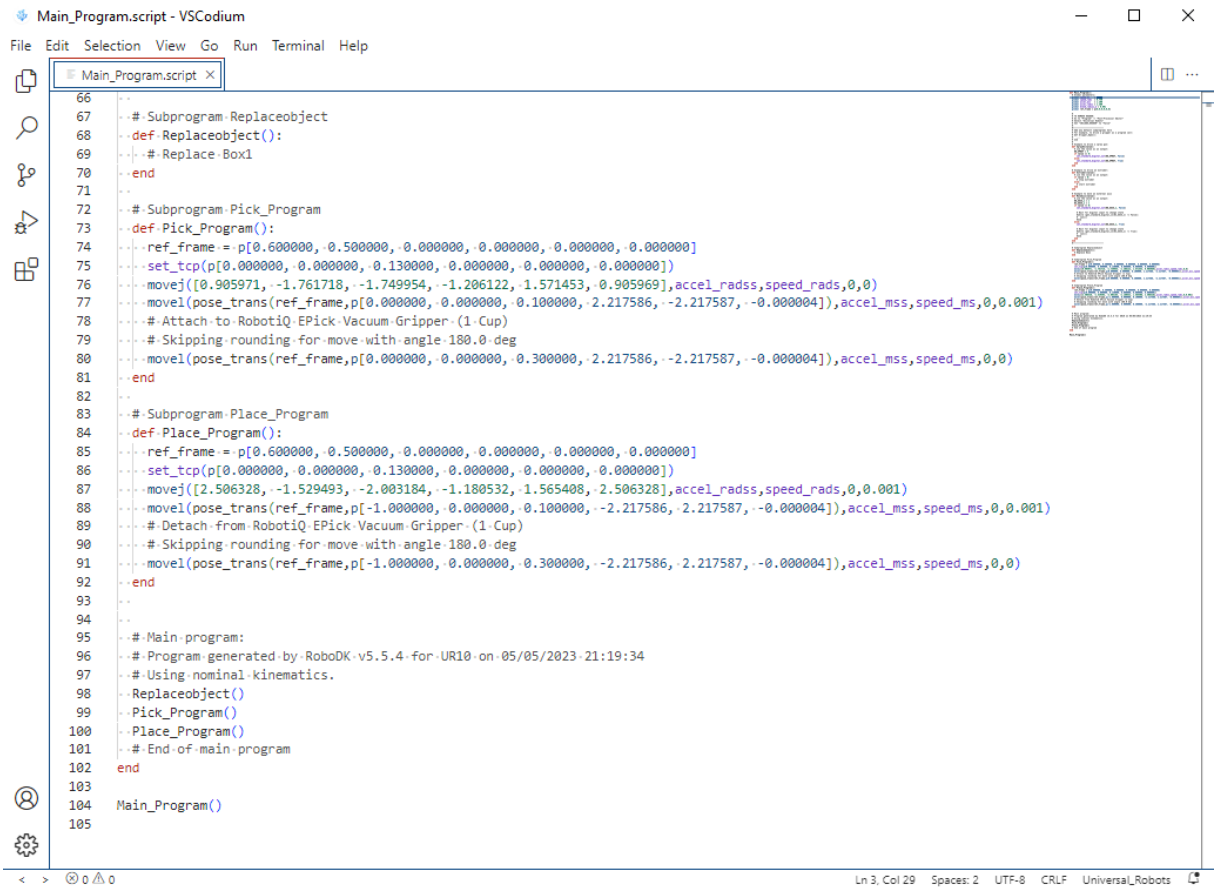


Figure 22.2. Generated robot program example

If we are properly connected to the robot we can also select one of the following options in the program menu:

1. Select **Send program to robot (Ctrl+F6)** to send the program through FTP (Offline Programming)
2. Check the option **Run on Robot** to run the program step by step each time we run the program (Online Programming). This allows executing the program on the robot as it is simulated at the same time. Robot drivers are required for Online Programming.

The Run on Robot option requires robot drivers to work properly. These drivers may require additional software options on the robot controller and/or a specific setup on the robot controller (this is not the case for UR robots).

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

PYTHON PROGRAMMING LANGUAGE

PYTHON WORKSHEETS STRUCTURE

WORKSHEET 10	
The aims at the end of this worksheet are:	• Becoming familiar with RoboDK API (Application Program Interface) • Learn how to use the first Python instructions to move the robotic arm from one position to another
Process steps:	Construction of a rectangle knowing its vertices Construction of a rectangle in the space from its four vertices: instructions for improving the Worksheet 3 (Robot Programming)
Study Question	

Python Pills:	• RoboDK API: https://robodk.com/doc/en/RoboDK-API.html#RoboDKAPI
	• Python API: https://robodk.com/doc/en/RoboDK-API.html#PythonAPI
	• Import libraries: https://robodk.com/doc/en/PythonAPI/robodk.html
	• Study of the instructions:
	  ◦ Item()
	  ◦ MoveJ()
	  ◦ MoveL()

Math Pills:	Introduction to matrix algebra
Reference file (.RDK):	**Worksheet_PY_1**
WORKSHEET 11	
The aims at the end of this worksheet are:	• Understanding **pose** data: position and orientation of the active tool with respect to the active reference system (reference frame) • Know how to apply the translation matrix
Process steps:	Construction of a square in the space from a target point Instructions for constructing a square in the space from a starting point
Study Question	

Python Pills:	Study of the instructions:
	  ◦ Pose()
	  ◦ transl()

Math Pills:	• Reference systems in the plane • Reference systems in space • Cartesian coordinates • Translation matrix
Reference file (.RDK):	**Worksheet_PY_2**
WORKSHEET 12	
The aims at the end of this worksheet are:	• Knowing how to apply the translation and rotation matrix to move in space • Knowing how to use the “for” loop to make iterations

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

Process steps:	Construction of a pentagon in space using two target points Instructions for building a pentagon in the space from its center and a final positioning point
Study Question	
Python Pills:	Study of the "for" loop Study of the instructions: • rotz()
Robotic Pills:	• Rotation matrix • Roto/translation transformations
Reference file (.RDK):	**Worksheet_PY_3**
WORKSHEET 13	
The aims at the end of this worksheet are:	• Knowing how to apply the translation and rotation matrix to move in space • Knowing how to use the "for" loop to make generalized iterations • Knowing how to create interactive dialog boxes to enter data by the user
Process steps:	Construction of a regular polygon in space given two target points Instructions for constructing a polygon in the space from its center and having the possibility to choose the number of sides and its radius
Study Question	
Python Pills:	Study of the instructions: • InputDialog()
Robotic Pills:	• Rotation matrix • Roto/translation transformations
Reference file (.RDK):	**Worksheet_PY_4, Worksheet_PY_4.1**
WORKSHEET 14	
The aims at the end of this worksheet are:	• Knowing how to use polar coordinates • Knowing how to use the "for" loop to make generalized iterations • Knowing how to create interactive dialog boxes to enter data by the user
Process steps:	Construction of a regular hexagon in the space from its center and its radius Instructions for building a hexagon from its center and its radius using the placement of points in space from a reference system to another
Study Question	
Python Pills:	Study of the instructions: • Pos() • setPos()
Math Pills:	Reference system translation, Polar coordinates
Reference file (.RDK):	**Worksheet_PY_5, Worksheet_PY_5.1**

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

23. PYTHON IN RoboDK

Python is an object-oriented, interpretive, modular and interactive high-level programming language.

Simple syntax based on indentation makes the language easier to learn and remember. It is especially preferred by those who are new to programming because there is no need to pay attention to syntax details.

Its modular structure supports class system (system) and all types of data field entry. It can run on almost any platform (Unix, Linux, Mac, Windows, Amiga, Symbian).With Python, you can develop software in many areas such as system programming, user interface programming, network programming, web programming, application and database software programming. (wikipedia.org)

We will use the Python language on the RoboDK platform to ensure that the robot arms move the way we want.

The robolink sub-module is the bridge between RoboDK and Python. Every object in the RoboDK item tree can be retrieved and it is represented by the object Item. An item can be a robot, a reference frame, a tool, an object or any other item visible in the station tree.

https://robodk.com/doc/en/RoboDK-API.html

classrobodk.robolink.Item(link, ptr_item=0, itemtype=- 1)

The Item class represents an item in RoboDK station. An item can be a robot, a frame, a tool, an object, a target, … any item visible in the station tree. An item can also be seen as a node where other items can be attached to (child items). Every item has one parent item/node and can have one or more child items/nodes.

Item(*name*, *itemtype=None*)

Returns an item by its name. If there is no exact match it will return the last closest match. Specify what type of item you are looking for with itemtype. This is useful if 2 items have the same name but different type. (check variables ITEM_TYPE_*)

Parameters

- **name** (*str*) – name of the item (name of the item shown in the RoboDK station tree)
- **itemtype** (*int*) – type of the item to be retrieved (avoids confusion if there are similar name matches). Use ITEM_TYPE_*.

Available Item types

```
ITEM_TYPE_STATION=1             # station item (.rdk files)
ITEM_TYPE_ROBOT=2               # robot item (.robot files)
ITEM_TYPE_FRAME=3               # reference frame item
ITEM_TYPE_TOOL=4                # tool item (.tool files or tools without geometry)
ITEM_TYPE_OBJECT=5              # object item (.stl, .step, .iges, ...)
ITEM_TYPE_TARGET=6              # target item
ITEM_TYPE_PROGRAM=8             # program item (made using the GUI)
ITEM_TYPE_PROGRAM_PYTHON=10     # Python program or macro
```

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

23.1. MoveJ Command

MoveJ(*target*, *blocking=True*)

Move a robot to a specific target (“Move Joint” mode). This function waits (blocks) until the robot finishes its movements. If this is used with a program item, a new joint movement instruction will be added to the program. Important note when adding new movement instructions to programs: only target items supported, not poses.

Parameters

- **target** (`Mat`, list of joints or `Item`) – Target to move to. It can be the robot joints (Nx1 or 1xN), the pose (4x4) or a target (item pointer)
- **blocking** (*bool*) – Set to True to wait until the robot finished the movement (default=True). Set to false to make it a non blocking call. Tip: If set to False, use robot.Busy() to check if the robot is still moving.

23.2. MoveL Command

MoveL(*target*, *blocking=True*)

Moves a robot to a specific target (“Move Linear” mode). This function waits (blocks) until the robot finishes its movements. This function can also be called on Programs and a new movement instruction will be added at the end of the program. If this is used with a program item, a new linear movement instruction will be added to the program. Important note when adding new movement instructions to programs: only target items supported, not poses.

Parameters

- **target** (`Mat`, list of joints or `Item`) – Target to move to. It can be the robot joints (Nx1 or 1xN), the pose (4x4) or a target (item pointer)
- **blocking** (*bool*) – Set to True to wait until the robot finished the movement (default=True). Set to false to make it a non blocking call. Tip: If set to False, use robot.Busy() to check if the robot is still moving.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

WORKSHEET 10

CONSTRUCTION OF A RECTANGLE KNOWING ITS VERTICES

The aims at the end of this worksheet are:

- Becoming familiar with RoboDK API (Application Program Interface)
- Learn how to use the first Python instructions to move the robotic arm from one position to another

Process Steps:

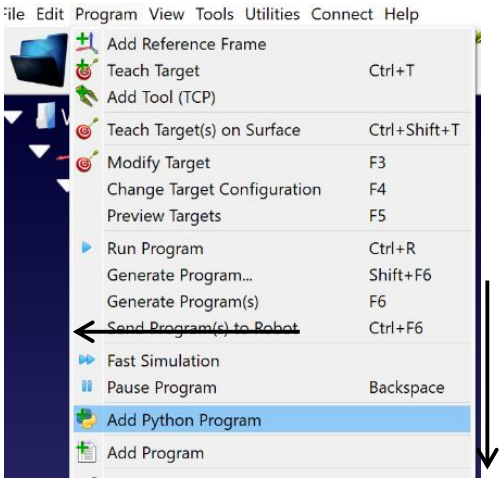
1- Open the program by double-clicking the program icon on the desktop 

2- Press the  button to open the **Worksheet 3** you created.

3- Please **save station as** Worksheet_PY_1

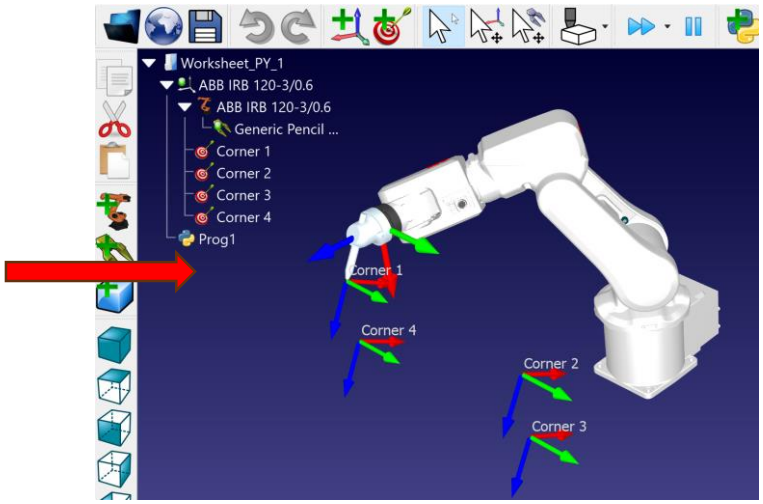
4- Right click on Rectangle and choose **delete**

5- Select Program →Add Python Program:



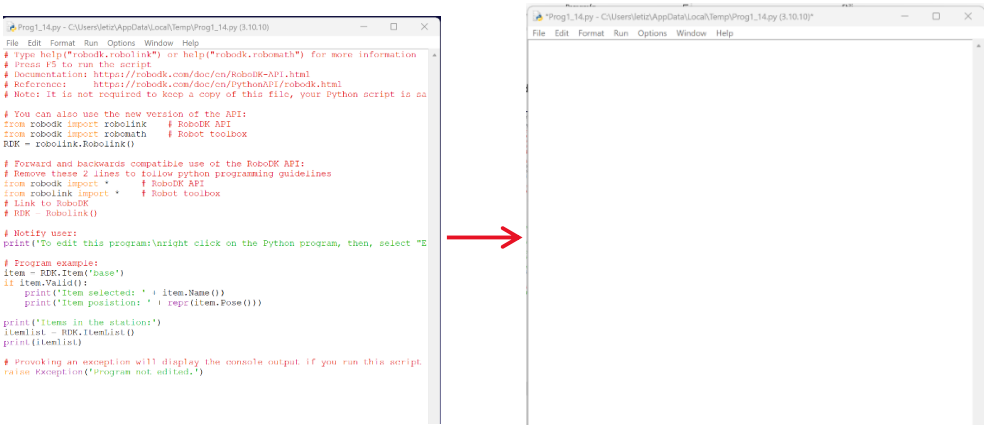
Otherwise you can use the icon in the Toolbar menu 

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union



6- **Right click** on Prog1 and select **Edit Python Script**

7- A new window appear, select the test and erase all:



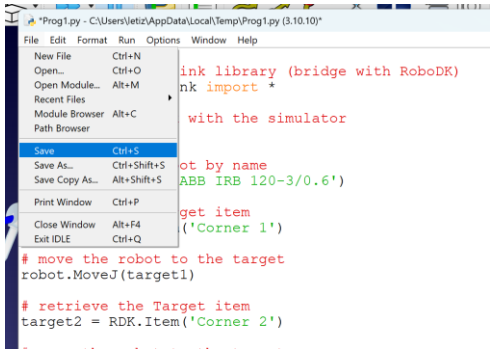
8- Begin to write this Python code:

```
# import the robolink library (bridge with RoboDK)
from robodk.robolink import *

# establish a link with the simulator
RDK = Robolink()

# retrieve the robot by name
robot = RDK.Item('ABB IRB 120-3/0.6')

# retrieve the Target item
target1 = RDK.Item('Corner 1')

# move the robot to the target 1
robot.MoveJ(target1)

# retrieve the Target item
target2 = RDK.Item('Corner 2')

# move the robot to the target 2
```

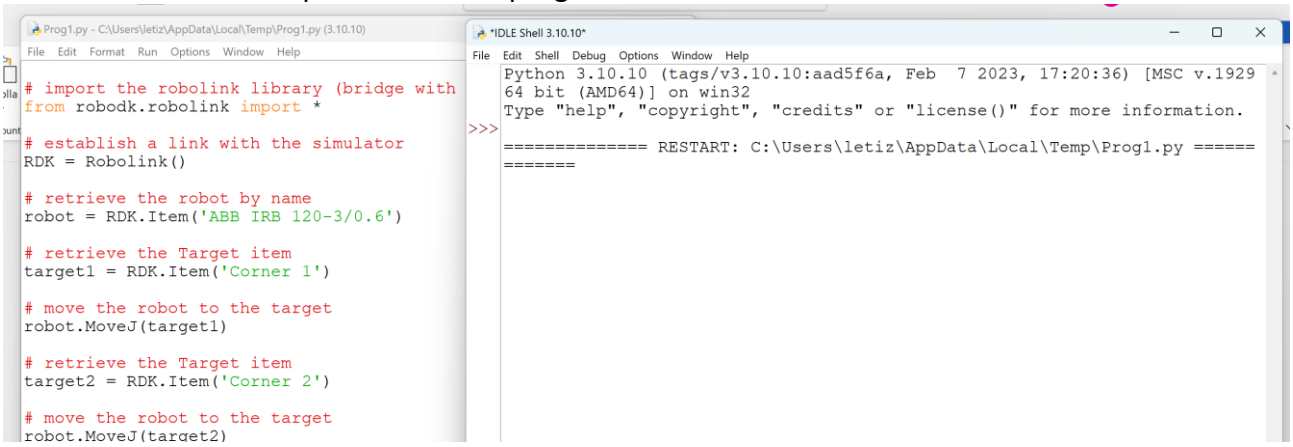
TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

```
robot.MoveJ(target2)

# retrieve the Target item
target3 = RDK.Item('Corner 3')

# move the robot to the target 3
robot.MoveJ(target3)

# retrieve the Target item
target4 = RDK.Item('Corner 4')

# move the robot to the target 4
robot.MoveJ(target4)

# retrieve the Target item
target1 = RDK.Item('Corner 1')

# move the robot to the target 1
robot.MoveJ(target1)
```

9- Save the programm:

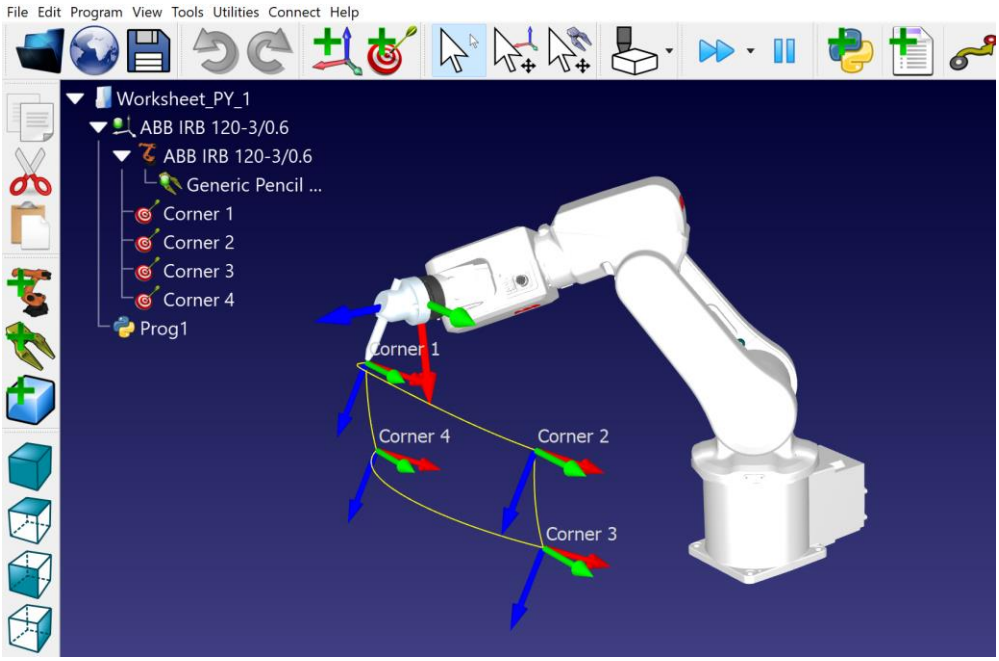


10- Select **Run**, e choose **Run Module**.
The shell window opens to run the program:



11- Go back to the main screen, by double clicking on Prog1 the robot follows the path described in the worksheet3, describing the outline of a rectangle:

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union



12- Please repeat the same steps for Universal Robot UR10 and save this station under the name Whorksheet_PY1.1

13- Please modify the program using MoveL() instruction instead of MoveJ().

Funded by the European Union

24. PYTHON VARIABLES AND VARIABLES RULES

There is no special command to declare variables in Python. All you have to do is declare the variable name and assign a value to it.

But there are some rules we need to pay attention to when naming variables. These:

a) Variable names are case sensitive. For example; The fact that the variable name is ***address*** or ***Address*** indicates that these variables are two different variables.

b) Both letters and numbers can be used when naming a variable. But the numbers don't add up. For example, while ***number1*** is a correct naming but ***1number*** is not a correct naming.

c) Underscore (_) can be used when naming a variable. However, spaces and other special characters (?,%,!, ., + etc.) are not used. For example, variable names such as ***home address*** (there is space) or ***ID%no*** (there is special character) are against the rules and will cause an error.

d) While naming the variable, you can't use commands in the Python programming language, such as if, for, true, etc.

Below are some examples of correctly defined variables within the framework of these rules:

```
live_city="Izmir"

exam_grade=80

rate3=5.7

RDK = Robolink()

robot = RDK.Item('ABB IRB 120-3/0.6')

target1 = RDK.Item('Corner 1')
```

As defined above; live_city, exam_grade, rate3, RDK, robot and target1 are variables and a value is assigned to them.

24.1. Python Operators

Operators are symbols that enable the production of new values by performing operations on variables and data. For beginners in the Python programming language, it is extremely important to learn arithmetic, assignment, comparison and logical operators.

24.1.1.Arithmetic Operators

They are operators used to perform mathematical operations on values stored in variables.

Operators	Definition	Example
+	Collection	a+b
-	Extraction	a-b
*	Impact	a*b
/	Divide	a/b
%	Getting a mode	a%b
**	exponentiation	a**b
//	Integer division (Only the whole part is taken in division.)	a//

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

24.1.2. Assignment Operators

They are operators used to transfer the value in one variable to another variable or to transfer the result of an operation to another variable.

Operators	Definition	Example
=	a=2	The variable a is assigned the value 2.
+=	a+=2	By adding the value 2 to the variable a, it is assigned to the variable a again. It means a=a+2.
-=	a-=2	The value 2 is subtracted from variable a and assigned to variable a again. It means a=a-2.
*=	a*=2	Variable a is multiplied by 2 and assigned to variable a again. It means a=a*2.
/=	a/=2	The variable a is divided by the value 2 and assigned to the variable a again. It means a=a/2.
%=	a%=2	The mode of the variable a is taken with the value 2 and it is assigned to the variable a again. It means a=a%2.

24.1.3. Comparison Operators

These are operators used when you want to compare the values of two variables and take action accordingly.

Operators	Definition	Example
==	Equal	a==b
!=	does not equal	a!=b
<	is small	a<b
>	is greater than	a>b
<=	less than equal	a<=b
>=	greater than equals	a>=

24.1.4. Logical Operators

They are operators used to decide the program flow by controlling the values of two or more variables.

Operators	Definition	Example
and	a<3 and b>=5	Returns True if two or more conditions are all true. In the example here, if variable a is less than 3 and variable b is equal to or greater than 5, the value True is returned.
or	a<3 or b>4	Returns True if at least one of two or more conditions is true. In the example here, it is sufficient for variable a to be less than 3 or variable b to be greater than 4 to return the True value.
not	not(a<3)	It is used to reverse the situation (False if True; True if False). In this example, the result of the logical test in parentheses is reversed. Assuming that the expression will return true when written without the not command, it will return false.

24.2. Python Datatypes

In Python, there are generally data types such as string (textual), numbers (numeric), boolean, list (list), tuple (tuple), dictionary (dictionary) and set (set). The types of variables we will use most at the beginning are listed below.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

24.2.1. String (Textual) Data Type

They are strings of characters written inside single or double quotes. Here, characters can be letters (a,b), numbers (1,9,2,3) or special symbols (&,/). String data types are written in single or double quotes

```
name="Mert"
surname="Yılmaz"
```

24.2.2. Numbers Data Type

It is the name given to data types that hold numerical data. Numerical data types in Python are generally int, float and complex data types.

```
number=1919
pi_value=3.14
```

24.2.3. Python Lists

Collections where different data are kept in a series are called lists. You can keep different data types such as int, float, string in a single list. Lists are used to keep multiple data in an ordered and changeable structure. In the Python programming language, lists are defined with two square brackets.

```
ilk_liste=["Ankara", 312, 0.6]
```

24.2.4 Dictionary (dict)

Stores key-value pairs. Keys must be unique. It is defined as key:value in {}.

```
thisdict = {
  "brand": "BMW
  "model": "X3",
  "year": 2020
}
```

➔ **Python in RoboDK:**

Pose()

Returns the relative pose of an object, target or reference frame. For example, the position of an object, target or reference frame with respect to its parent (the item it is attached to in the tree). For robot items, this function returns the pose of the active tool with respect to the active reference frame.

It returns the pose as **Mat**.

Tip: Use a Pose_2_* function from the robodk module (such as **robomath.Pose_2_KUKA**) to convert the pose to XYZABC (XYZ position in mm and ABC orientation in degrees), specific to a robot brand.

robodk.robomath.transl(tx, ty=None, tz=None)

Returns a translation matrix (mm)

$$T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Parameters

- **tx** (*float*) – translation along the X axis
- **ty** (*float*) – translation along the Y axis
- **tz** (*float*) – translation along the Z axis

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

WORKSHEET 11

CONSTRUCTION OF A SQUARE IN THE SPACE FROM A TARGET POINT

The aims at the end of this worksheet are:

- Understanding **pose** data: position and orientation of the active tool with respect to the active reference system (reference frame)
- Know how to apply the translation matrix

Process Steps:

1- Open the program by double-clicking the program icon on the desktop 

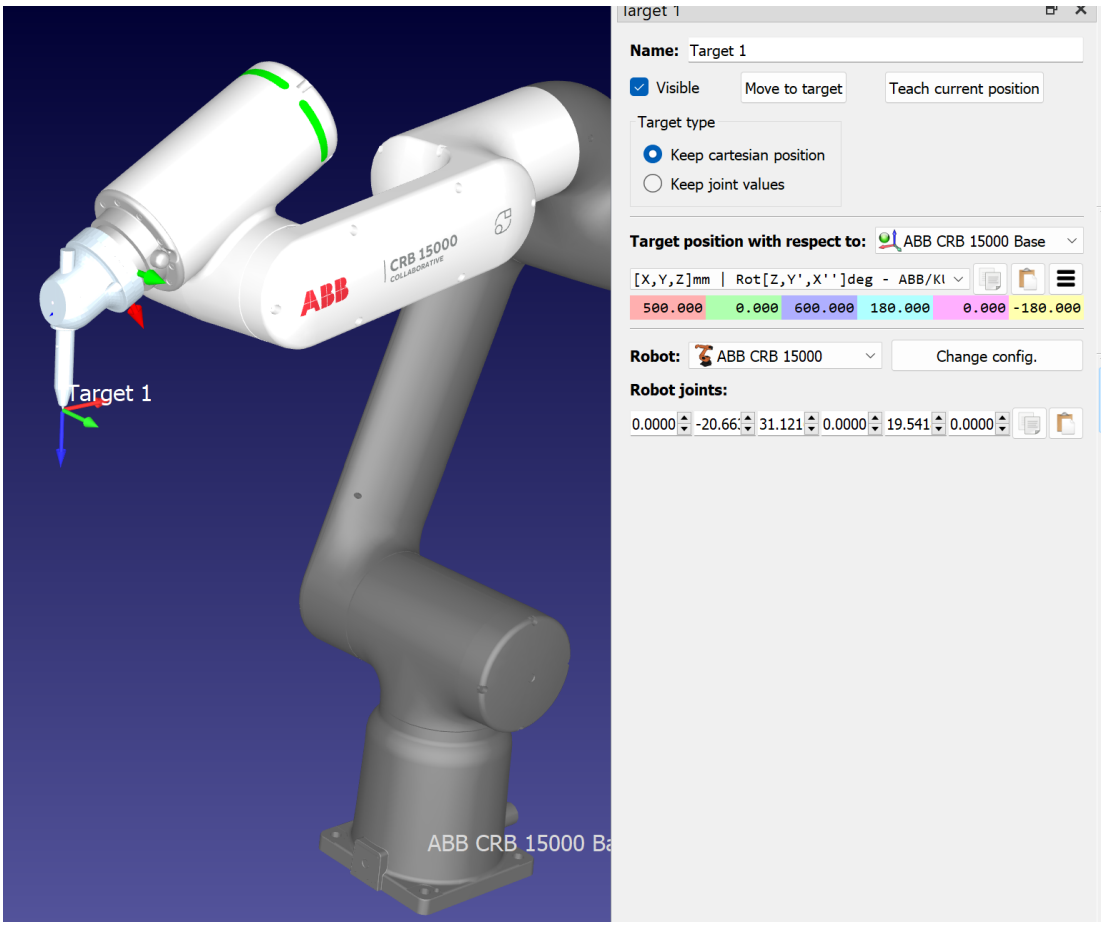
2- Create a new station, select the **ABB CRB 15000** from the Robot library menù

3- Select a **Generic pencil tool** from the library

4- Save the station as **Worksheet_PY_2**

5- Create a target point with the following configuration:

Set **Target 1** to: 500, 0, 600, 180, 0, -180



6- Add the Python program with icon or with the menu Programm

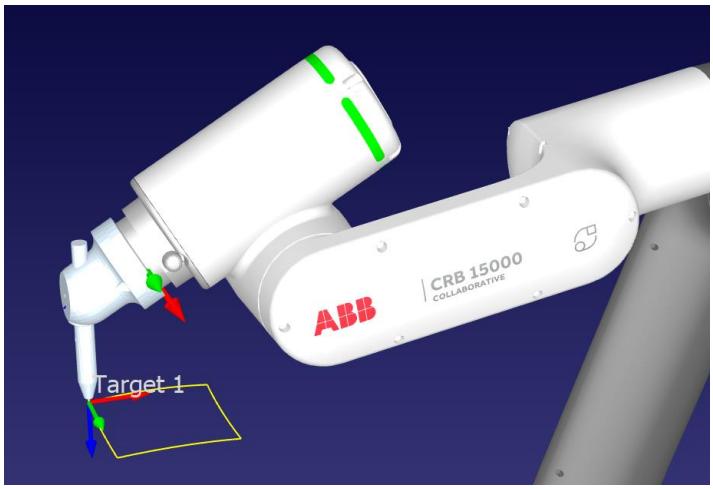
TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

7- Write the following program:

```
# import the robolink library (bridge with RoboDK)
from robodk.robolink import *

# establish a link with the simulator
RDK = Robolink()

# retrieve the robot by name
robot = RDK.Item(' ABB CRB 15000')

#create home position in the Target 1

home=RDK.Item('Target 1')

# import the robotics toolbox
from robodk.robomath import *

# create the vertex1 in a position 100 mm along the X axis of the
tool
#with respect to the home position

vertex_1 = home.Pose()*transl(100,0,0)

# move to the vertex_1
robot.MoveJ(vertex_1)

# create the vertex2 in a position 100 mm along the Y axis of the
tool
#with respect to the vertex1 position

vertex_2 = vertex_1*transl(0,100,0)

# move to the vertex_2
robot.MoveJ(vertex_2)

# create the vertex3 in a position 100 mm along the -X axis of the
tool
#with respect to the vertex2 position

vertex_3 = vertex_2*transl(-100,0,0)

# move to the vertex_3
robot.MoveJ(vertex_3)

# create the vertex4 in a position 100 mm along the -Y axis of the
tool
#with respect to the vertex3 position

vertex_4 = vertex_3*transl(0,-100,0)

# move to the vertex_4
robot.MoveJ(vertex_4)
```

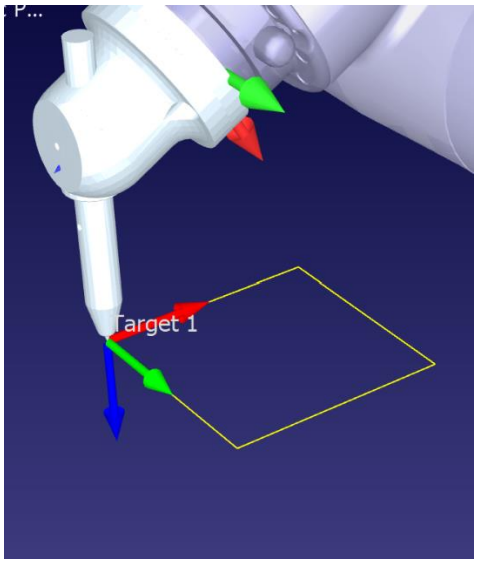
TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

8- The robot describes, with the active tool, a square with side 100:



9- Please, modify the code to have a rectangle with sides 100 and 50

10- Please modify the program using MoveL() instruction instead of MoveJ():



TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

25.DECISION AND LOOP STRUCTURES

Loops and decision structures are used where necessary to determine the direction of the program in a program flow.

25.1. Python Decision – If..elif

If we want the codes to branch according to the desired conditions by controlling the values of two or more variables and running the codes in accordance with the conditions, we use decision structures.

Example 1:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

Example 2:

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

25.2. Python While Loops

We can execute a series of statements as long as a condition is true.

Example 1: Print i as long as i is less than 6

Program Output

```
i = 1
while i < 6:
  print(i)
  i += 1
```

```
1
2
3
4
5
```

25.3. Python For Loops

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

Example Print each fruit in a fruit list:

Program Output

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

```
apple
banana
cherry
```

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

➔ **Python in RoboDK:**

robodk.robomath.rotz(rz)

Returns a rotation matrix around the Z axis (radians)

$$R_x(\theta) = \begin{bmatrix} c_\theta & -s_\theta & 0 & 0 \\ s_\theta & c_\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Parameters

ry (*float*) – rotation around Y axis in radians

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY

Funded by
the European Union

WORKSHEET 12

CONSTRUCTION OF A PENTAGON IN SPACE USING TWO TARGET POINTS

The aims at the end of this worksheet are:

- Knowing how to apply the translation and rotation matrix to move in space
- Knowing how to use the “for” loop to make iterations

Process Steps:

1- Open the program by double-clicking the program icon on the desktop 

2- Create a new station, select the **KUKA KR 6 R900 sixx Base** from the Robot library menù

3- Select the **weld_gun** from the library

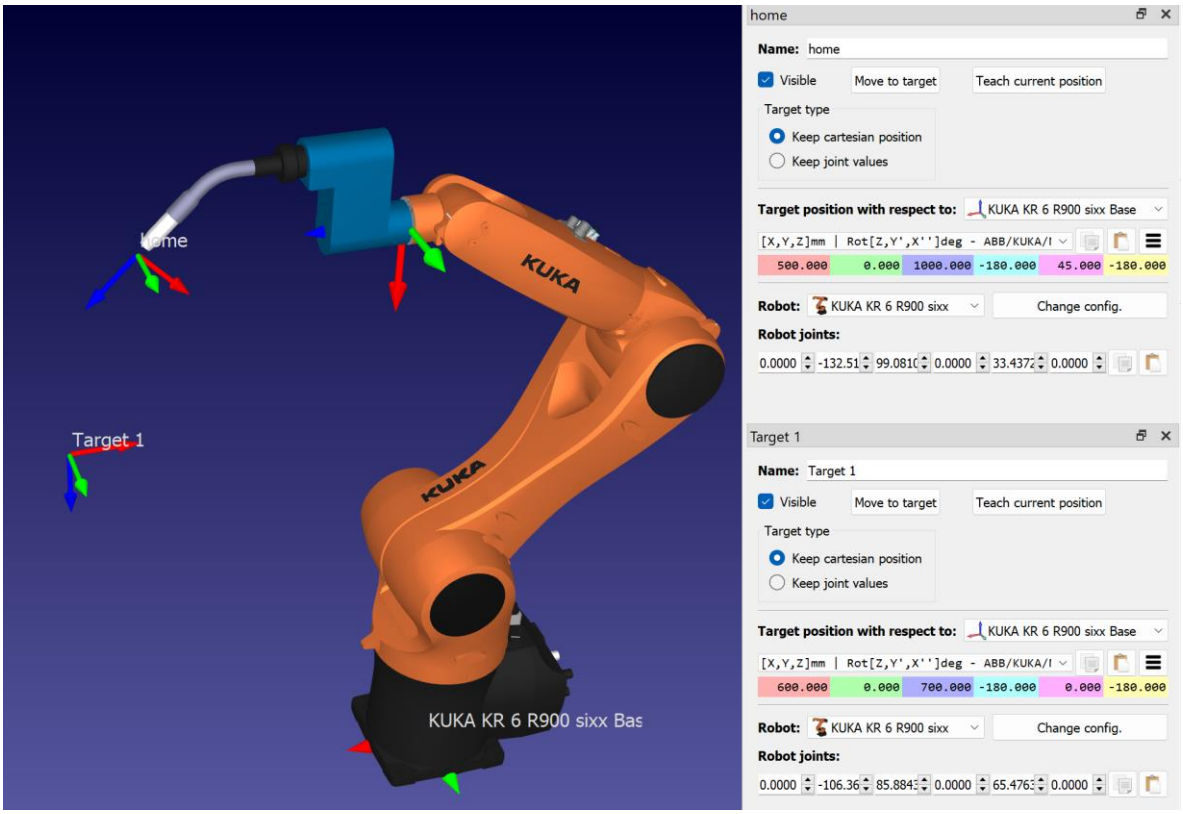
4- Save the station as **Whorksheet_PY_3**

5- Create a target point with the following configuration:

Set **Target 1** to: 600, 0, 700, -180, 0, -180

6- Create an another target point with the following configuration:

Set **home** to: 500, 0, 1000, -180, 45, -180



7- Add the Python program with icon or with the menu Programm

8- Write the following program:

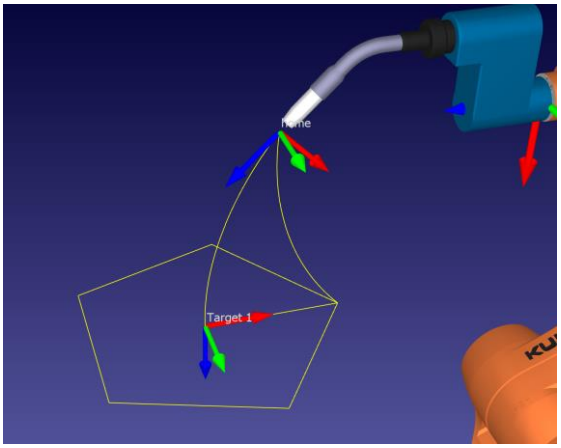
```
#API to communicate with RoboDK
```

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY

Funded by
the European Union

```
#Import the robolink library (bridge with RoboDK)
from robodk.robolink import*

# basic matrix operations
from robodk import *

# Any interaction with RoboDK must be done through Robolink()
# estabilish a link with the simulator
RDK = Robolink()

# get the robot item, (retrieve the robot by name):
robot = RDK.Item('KUKA KR 6 R900 sixx')

# get the home target:
home = RDK.Item('home')

# get the pentagon center:
center = RDK.Item('Target 1')

# get the pose of the ref (4x4 matrix):
poseref= center.Pose()

# move robot to home than to center
robot.MoveJ(home)
robot.MoveJ(center)

# make a pentagon :
for i in range(6):
        ang = i*2*pi/5 #angle: 0,60,120...
        posei = poseref*rotz(ang)*transl(200,0,0)*rotz(-ang)
        robot.MoveL(posei)

# move back to home:
robot.MoveJ(home)
```

9- The robot will describe a pentagon-shaped path and return to the initial position



TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

26.InputDialog USAGE

robodk.robodialogs.InputDialog(msg, value, title=None, default_button=False, default_value=None, embed=False, actions=None, *args, **kwargs)

Show a blocking input dialog, with 'OK' and 'Cancel' buttons.

The input field is automatically created for supported types:

- Base types: bool, int, float, str
- list or tuple of base types
- dropdown formatted as [int, [str, str, …]]. e.g. [1, ['Option #1', 'Option #2']] where 1 means the default selected option is Option #2.
- dictionary of supported types, where the key is the field's label. e.g. {'This is a bool!' : True}.

Parameters

- **msg** (*str*) – Message to the user (describes what to enter)
- **value** – Initial value of the input (see supported types)
- **title** (*embed*) – Window title, optional
- **default_button** – Show a button to reinitialize the input to default, defaults to false
- **default_value** – Default values to restore. If not provided, the original values will be used
- **title** – Embed the window inside RoboDK, defaults to false
- **actions** (*list of tuples of str, callable*) – List of optional action callbacks to add as buttons, formatted as [(str, callable), …]. e.g. [("Button #1", action_1), ("Button #2", action_2)]

Returns

The user input if the user clicked 'OK', None for everything else

Return type

See supported types

Example:

```
print(InputDialog('This is as input dialog.\n\nEnter an integer:', 0))
print(InputDialog('This is as input dialog.\n\nEnter a float:', 0.0))
print(InputDialog('This is as input dialog.\n\nEnter text:', ''))
print(InputDialog('This is as input dialog.\n\nSet a boolean:', False))
print(InputDialog('This is as input dialog.\n\nSelect from a dropdown:', [0,
['RoboDK is the best', 'I love RoboDK!', "Can't hate it, can I?"]]))
print(InputDialog('This is as input dialog.\n\nSet multiple entries:', {
    'Enter an integer:': 0,
    'Enter a float:': 0.0,
    'Set a boolean:': False,
    'Enter text:': '',
    'Select from a dropdown:': [0, ['RoboDK is the best!', 'I love RoboDK!',
"Can't hate it, can I?"]],
    'Edit int list:': [0, 0, 0],
    'Edit float list:': [0., 0.],
}))
```

Funded by the European Union

WORKSHEET 13

CONSTRUCTION OF A REGULAR POLYGON IN SPACE GIVEN TWO TARGET POINTS

The aims at the end of this worksheet are:

- Knowing how to apply the translation and rotation matrix to move in space
- Knowing how to use the “for” loop to make generalized iterations
- Knowing how to create interactive dialog boxes to enter data by the user

Process Steps:

1- Open the program by double-clicking the program icon on the desktop 

2- Open Whorksheet_PY_3

3- Save as **Worksheet_PY_4**

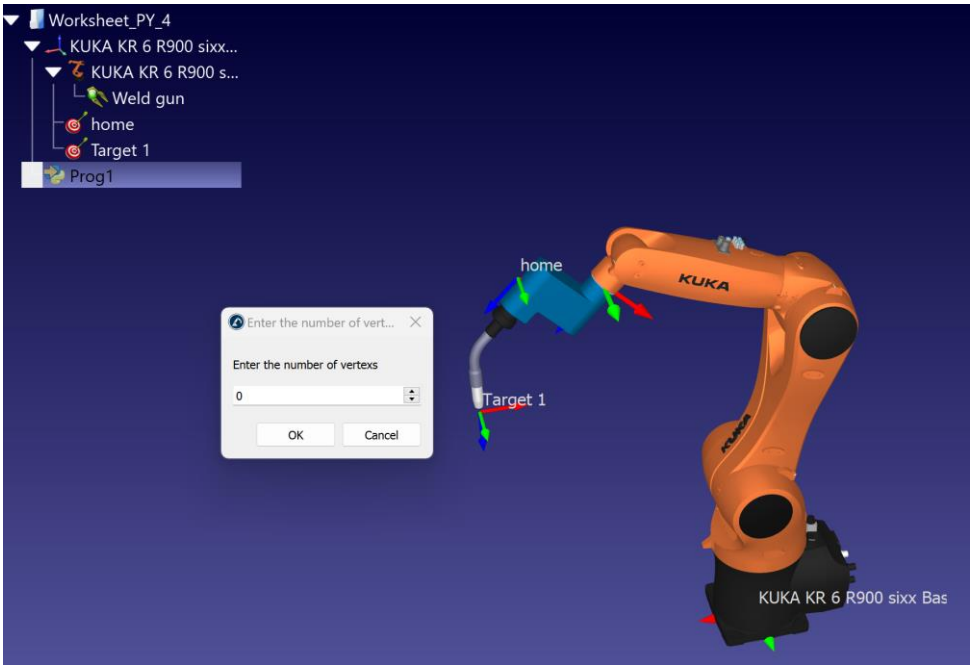
4- Open Python program and modify the code in this way :

```
#API to communicate with RoboDK
#Import the robolink library (bridge with RoboDK)
from robodk.robolink import*

# basic matrix operations
from robodk import *

# Any interaction with RoboDK must be done through Robolink()
# estabilish a link with the simulator
RDK = Robolink()

# get the robot item, (retrieve the robot by name):
robot = RDK.Item('KUKA KR 6 R900 sixx')

# get the home target:
home = RDK.Item('home')

# get the pentagon center:
center = RDK.Item('Target 1')

# get the pose of the ref (4x4 matrix):
poseref= center.Pose()

# move robot to home than to center
robot.MoveJ(home)
robot.MoveJ(center)
```

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

```
#Through a dialog window, define the number of vertexs of the
polygon
nvertexs= InputDialog('Enter the number of vertexs', 0 , )
nvertexs= int(nvertexs)

# make a poligon around center :
for i in range(nvertexs+1):
       ang = i*2*pi/nvertexs #angle: 0,60,120...
       posei = poseref*rotz(ang)*transl(200,0,0)*rotz(-ang)
       robot.MoveL(posei)

# move back to home:
robot.MoveJ(home)
```

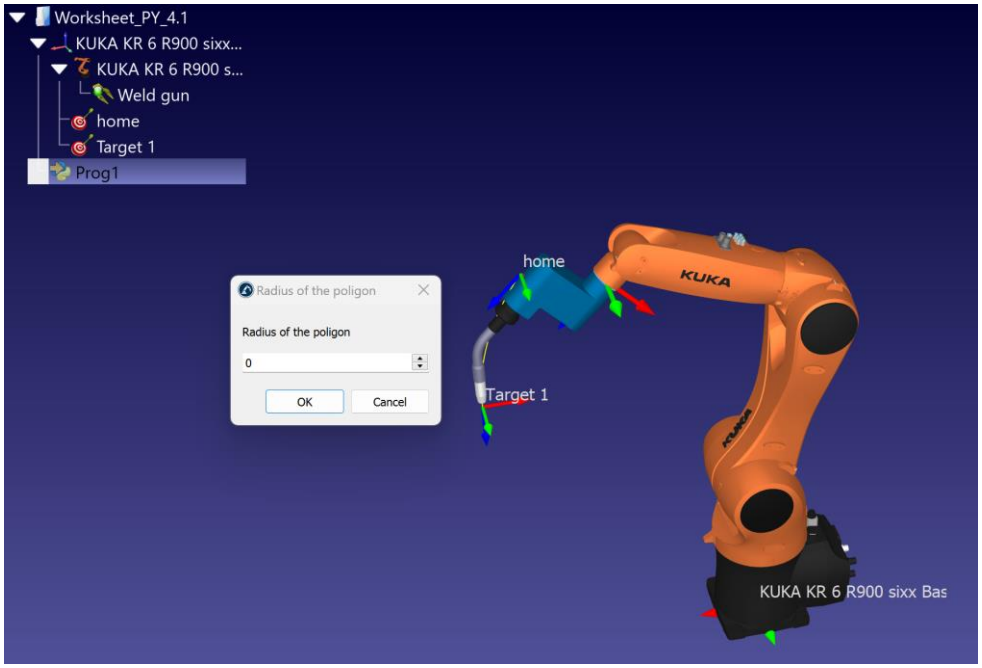
5- The program will allow to choose, through a dialog box, the number of sides of the polygon:



6- Please modify the program by adding the window asking for the radius of the polygon and save as **Worksheet_PY_4.1:**

```
#Through a dialog window, define the radius of the poligon
radius= InputDialog('Radius of the poligon', 0 , )
radius= int(radius)
```

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union



Questions:

What happens as the number of sides of the polygon increases?

Is it possible to enter any value for the radius of the polygon?

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

27.POSITION COMMANDS

Pos()

Returns the position of a pose (assumes that a 4x4 homogeneous matrix is being used)

setPos(*newpos*)

Sets the XYZ position of a pose (assumes that a 4x4 homogeneous matrix is being used)

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

WORKSHEET 14

CONSTRUCTION OF A REGULAR HEXAGON IN THE SPACE FROM ITS CENTER AND ITS RADIUS

The aims at the end of this worksheet are:

- Knowing how to use polar coordinates
- Knowing how to use the “for” loop to make generalized iterations
- Knowing how to create interactive dialog boxes to enter data by the user

Process Steps:

1- Open the program by double-clicking the program icon on the desktop 

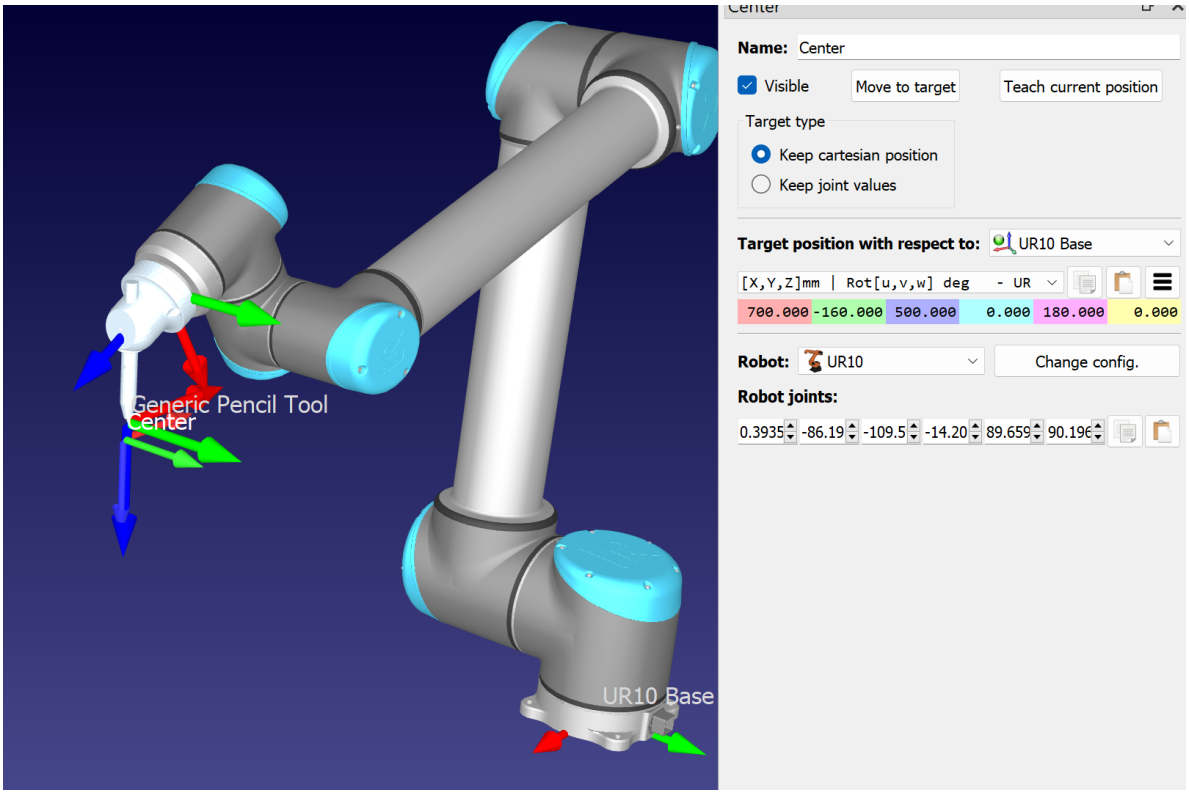
2- Create a new station, select the **UR10** from the Robot library menù

3- Select a **Generic pencil tool** from the library

4- Save the station as **Worksheet_PY_5**

5- Creare un punto di target con la seguente configurazione e denominarlo **Center** Create a target point with the following configuration and change its name in Center:

Set **Center** to: 700, -160, 500, 0, 180, 0



6- Add the Python program with icon or with the menu Programm

7- Write the following program:

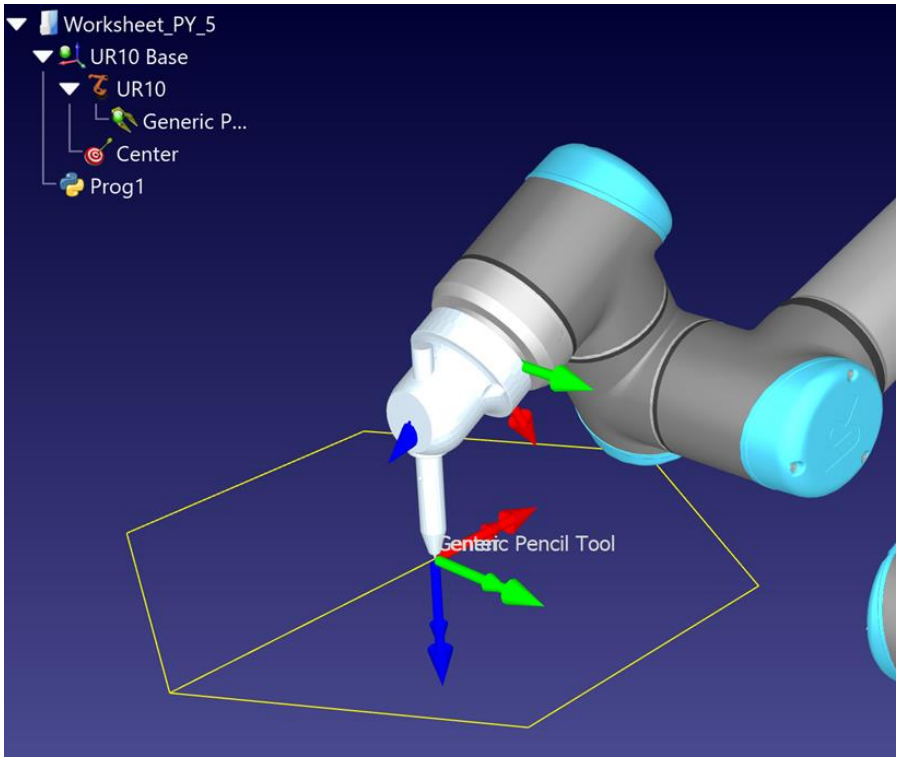
```
# Draw a hexagon around a point called center
```

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

```
#API to communicate with RoboDK
#Import the robolink library (bridge with RoboDK)
from robolink import *

# basic matrix operations
# Math toolbox for robots
from robodk import *

# Start the RoboDK API:
RDK = Robolink()

# Get the robot (first robot found):
robot = RDK.Item('', ITEM_TYPE_ROBOT)

# Get the reference target by name:
center = RDK.Item('Center')
center_pose = center.Pose()
xyz_ref = center_pose.Pos()

# Move the robot to the reference point:
robot.MoveJ(center)

# Draw a hexagon around the reference target:
for i in range(7):
    # Angle = 0,60,120,...,360
    ang = i*2*pi/6
    # Polygon radius
    R = 200
    # Calculate the new position:
    x = xyz_ref[0] + R*cos(ang) # new X coordinate
    y = xyz_ref[1] + R*sin(ang) # new Y coordinate
    z = xyz_ref[2]              # new Z coordinate
    center_pose.setPos([x,y,z])

    # Move to the new target:
    robot.MoveL(center_pose)

# Move back to the reference target:
robot.MoveL(center)
```

Funded by the European Union

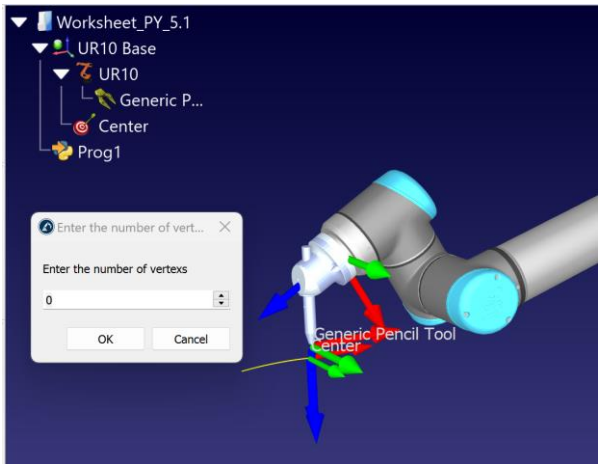
The program allows the tool connected to the flange to describe a regular polygon:

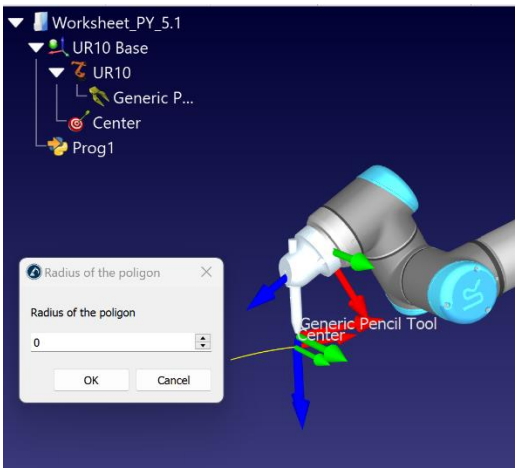


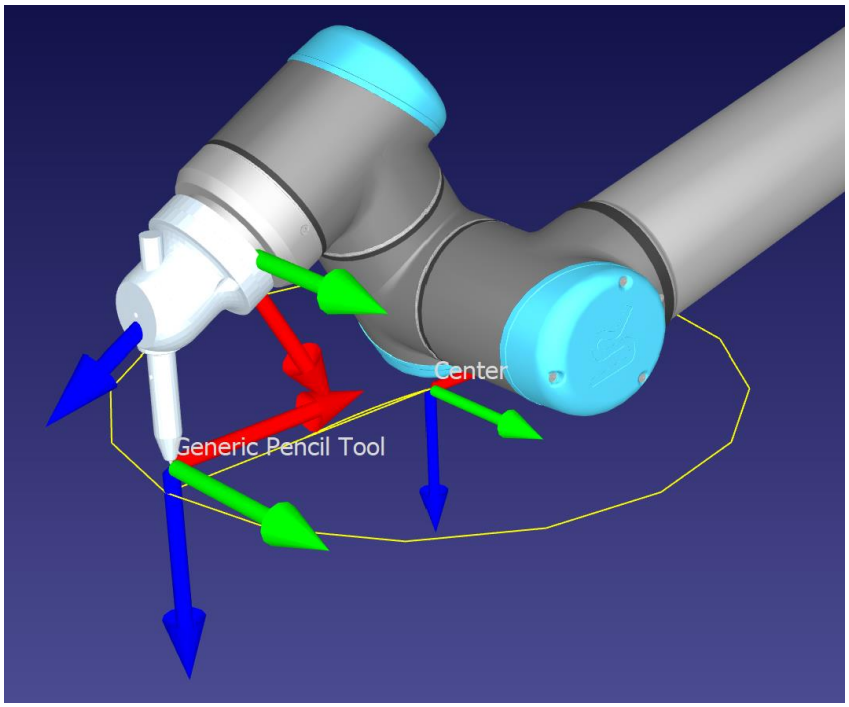
8- Modify the program adding the interactive dialog boxe that asks for the number of sides of the polygon and its radius and save as **Worksheet_PY_5.1:**

```
#Through a dialog window, define the number of vertexs of the
polygon
nvertexs= InputDialog('Enter the number of vertexs', 0 , )
nvertexs= int(nvertexs)

#Through a dialog window, define the radius of the poligon
radius= InputDialog('Radius of the poligon', 0 , )
radius= int(radius)

# Draw a hexagon around the reference target:
for i in range(nvertexs+1):
    # Angle = 0,60,120,...,360
    ang = i*2*pi/nvertexs
    # Calculate the new position:
    x = xyz_ref[0] + radius*cos(ang) # new X coordinate
    y = xyz_ref[1] + radius*sin(ang) # new Y coordinate
    z = xyz_ref[2]                   # new Z coordinate
    center_pose.setPos([x,y,z])

    # Move to the new target:
    robot.MoveL(center_pose)
```

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union







Study Question:

Study the change of reference system in space

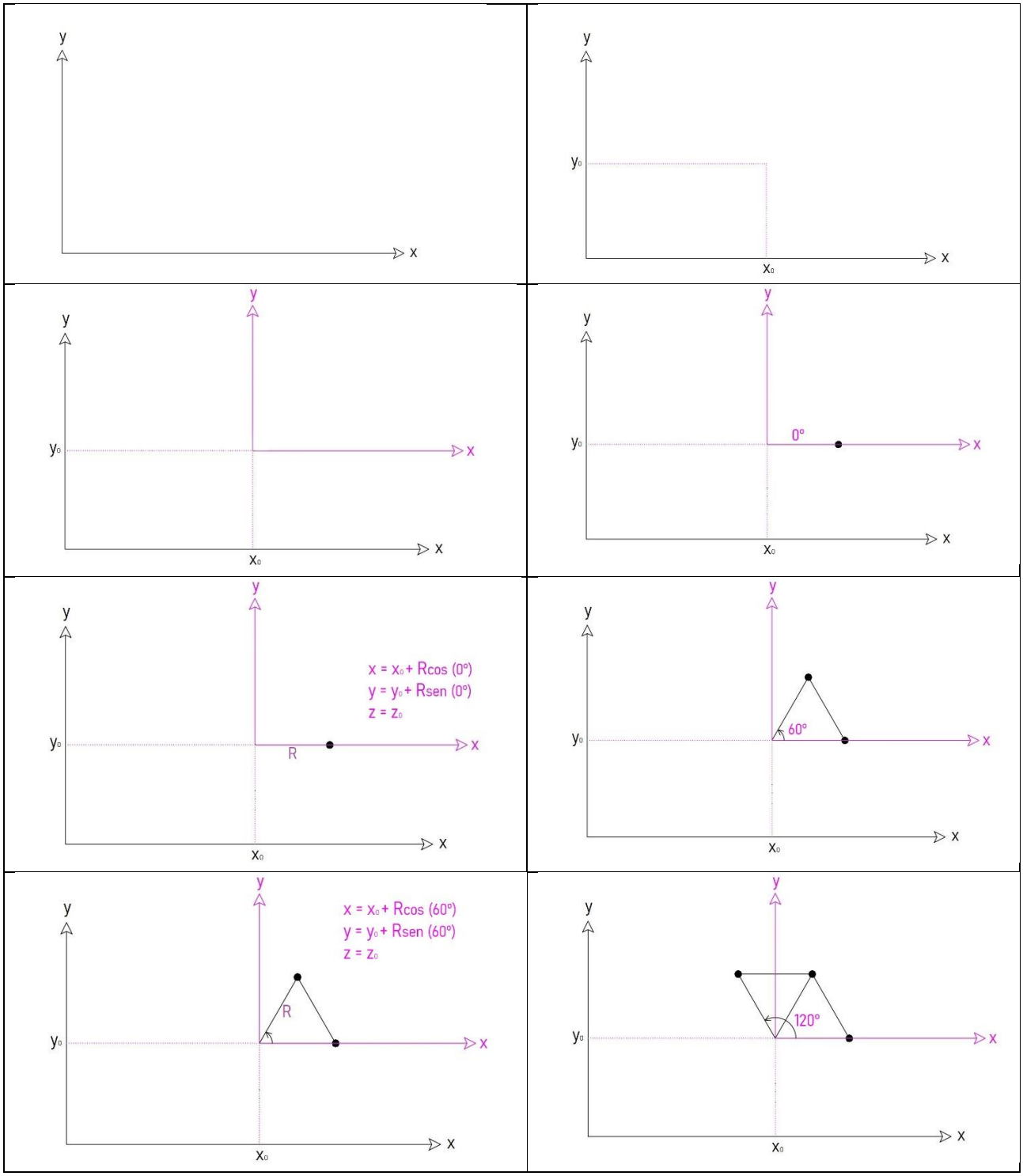
TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

Math Pills:

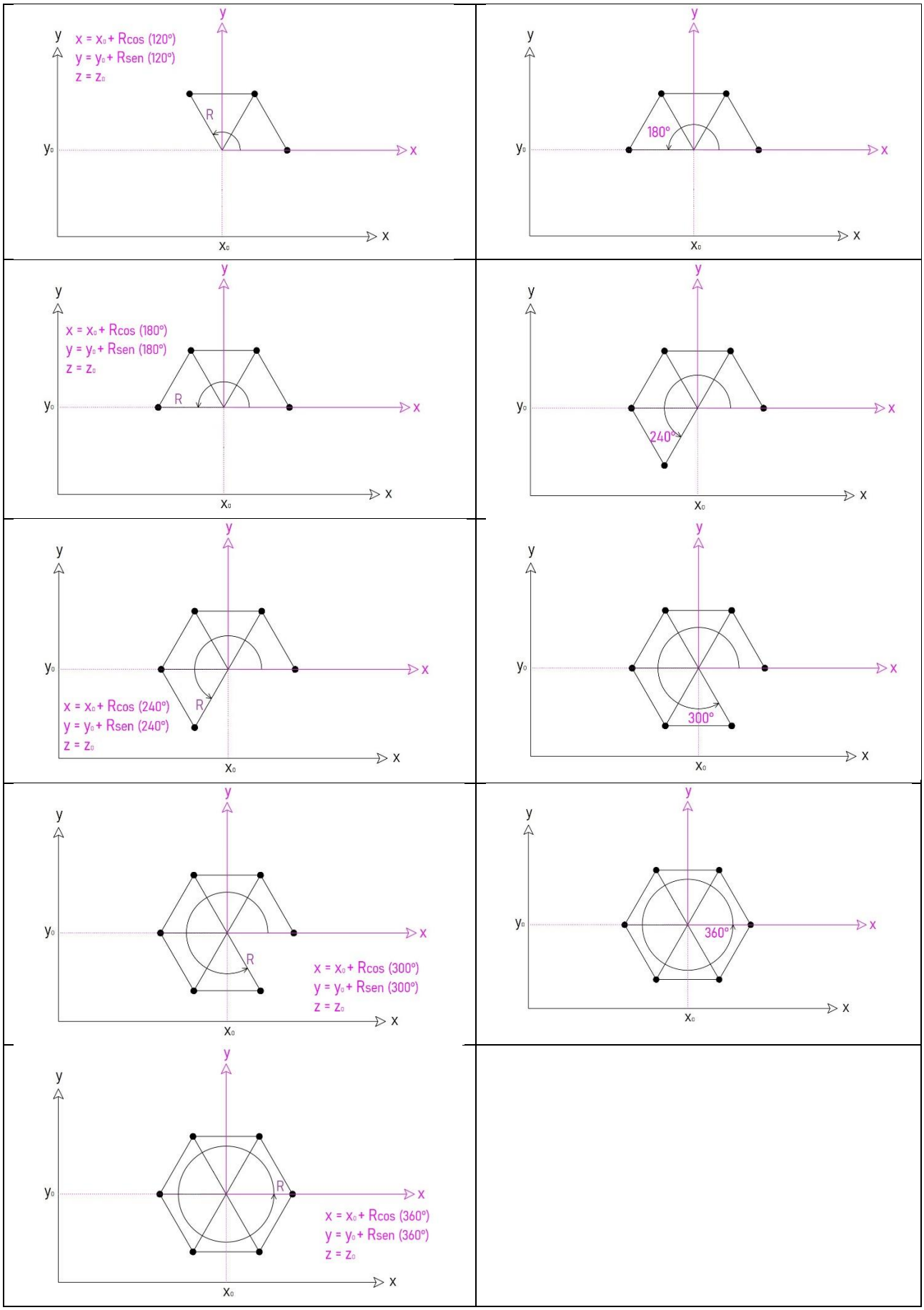
Reference system translation

Polar coordinates

Construction of the hexagon through the for loop:



TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union



TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

MAINTENANCE PRINCIPLES

This chapter explains principles for safety maintenance of collaborative robot. Robot cannot work without the end effector or peripheral equipment. By combined with the end effector and peripheral equipment and assembling the system, robot can demonstrate works. In other words the robot is one part of the system.

28. ROBOT SYSTEM COMPONENTS

The collaborative robot means the robot that work with workers. Therefore the following elements has been verified their safety.

Figure 1. Colloborative robot system example

- Robot
- Robot Controller
- Robot Teach PEndant
- End Effector
- Other peripheral devices
- Workpiece

Technicians conduct risk assessment of robot system, and the following elements must be prepared by the techinicians according to system configuration as the need arises.

- Safeguard
- Interlocked gate
- Interlocking device

Security is already confirmed against following components.

- Robot
- Robot controller and teach pendant

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

29. DEFINITION OF THE USER AND USERS SAFETY

29.1. Definition of Users

The user can be classified as follows.

Collaborative worker:

- Enter collaborative workspace, work with the robot
- Change the robot attitude by forcing robot directly, example push to escape function
- Restart the program with operator button set for collaborative worker.

Operator:

- Turns robot controller power ON/OFF
- Starts robot program from operator's panel

Programmer:

- Operates the robot and performs the teaching using a teach pendant.
- Operates the robot and performs the teaching using the direct teach.

Maintenance technician:

- Operates the robot
- Teaches robot inside the safety fence
- Maintenance (repair, adjustment, replacement)

Table shows the workings to the collaborative robot. In this table, the symbol "O" means the working allowed to be carried out by the personnel.

	Collaborative worker	Operator	Programmer or Teaching operator	Maintenance technician
Power ON/OFF for Robot controller		O	O	O
Select operating mode (AUTO, T1, T2)			O	O
Select Remote/Local mode			O	O
Select robot program with teach pendant			O	O
Select robot program with external device			O	O
Start robot program with operator's panel		O	O	O
Start robot program with teach pendant			O	O
Reset alarm with operator's panel			O	O
Reset alarm with teach pendant			O	O
Set data on the teach pendant			O	O
Teaching with teach pendant			O	O
Teaching with direct teach			O	O
Emergency stop with operator's panel	O	O	O	O
Emergency stop with teach pendant	O	O	O	O
Maintenance for operator's panel				O
Maintenance for teach pendant				O
Enter collaborative workspace, work with the robot	O	O	O	O
Restart the program with operator button which is set for collaborative worker	O	O	O	O

Table 1. List of workings to the collaborative robot

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

29.2. Users Safety

User safety is the primary safety consideration. Because it is very dangerous to enter the operating space of the robot during automatic operation, adequate safety precautions must be observed.
The following lists the general safety precautions. Careful consideration must be made to ensure user safety.

29.2.1. Safety of the Programmer Technician

While teaching the robot, the operator must enter the work area of the robot. Especially the teach pendant operator must secure own safety.

1- Unless it is specifically necessary to enter the robot work area, carry out all tasks outside the area.

2- Before teaching the robot, check that the robot and its peripheral devices are all in the normal condition.

3- If it is inevitable to enter the robot work area to teach the robot, check the locations, settings, and other conditions of the safety devices (such as the EMERGENCY STOP button, the Enabling device (DEADMAN switch) on the teach pendant) before entering the area

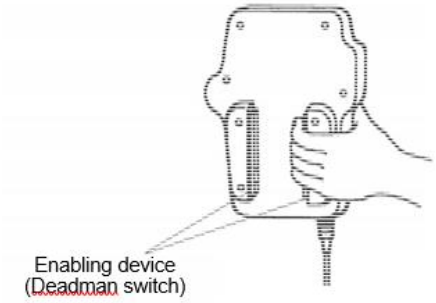


Figure 2. Enabling device (Deadman switch) (iPendant)

4- The programmer must be extremely careful not to let anyone else enter the robot work area.

5- Programming must be done outside of the safety fence as far as possible. If programming needs to be done in the area of the safety fence, the programmer must take the following precautions:

* Before entering the safety fence area, ensure that there is no risk of hazardous situation in the area.

* Be ready to press the emergency stop button whenever it is necessary.

* Operate the Robot at low speed.

* Before starting programming, check the entire system status to ensure that no remote instruction to the peripheral equipment or motion would harm working person.

6- Operator must work under the condition of Contact Stop function activates.

7- Required to deactivate the Contact Stop temporally, take measure to disseminate Contact Stop function deactivates.

8- To start the system using the operator's panel, make certain that nobody is the robot work area and that there are no abnormal conditions in the robot work area.

9- When a program is completed, be sure to carry out the test operation according to the following procedure.

(a) Run the program for at least one operation cycle in the single step mode at low speed.

(b) Run the program for at least one operation cycle in the continuous operation mode at low speed.

(c) Run the program for one operation cycle in the continuous operation mode at the intermediate speed and check that no abnormalities occur due to a delay in timing.

(d) Run the program for one operation cycle in the continuous operation mode at the normal operating speed, and check that the system operates automatically without trouble.

(e) After checking the completeness of the program through the test operation above, execute it in the automatic operation mode.

10- While operating the system in the automatic operation mode, the teach pendant operator must leave the robot work area.

29.2.2. Safety of the Maintenance Technician

For the safety of maintenance technician personel, pay utmost attention to the following.

1- Must never be in the area during its operation.

2- A hazardous situation may occur when the robot or the system, are kept with their power-on during maintenance operations. Therefore, for any maintenance operation, the robot and the system must be put into the power-off state. If necessary, a lock should be in place in order to prevent any other person from turning on the robot and/or the system. In case maintenance needs to be executed in the power-on state, the emergency stop button must be pressed.

3- If it becomes necessary to enter the robot operation area while the power is on, press the emergency stop button on the operator panel, or the teach pendant before entering the area. The maintenance personnel must indicate that maintenance work is in progress and be careful not to allow other people to operate the robot carelessly. (See Section 4.5.)

4- When entering the area enclosed by the safety fence, the maintenance worker must check the entire system in order to make sure that there is no dangerous situation around. In case the worker needs to enter the safety area whilst a dangerous situation exists, extreme care must be taken, and entire system status must be carefully monitored.

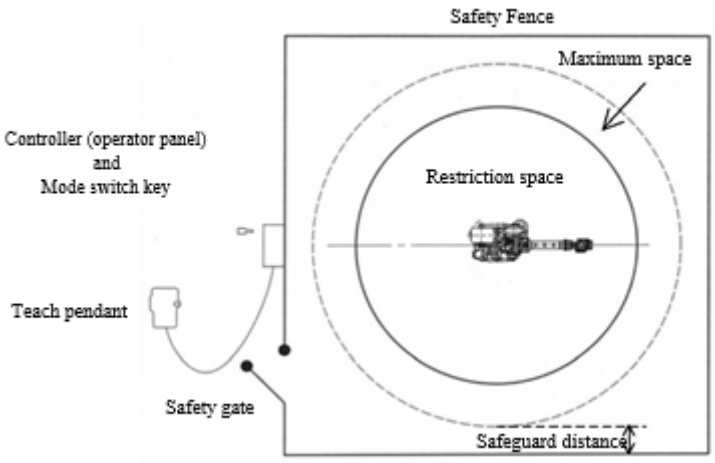


Figure 3. Safety Fence and Safety Gate example

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

5- Before the maintenance of the pneumatic system is started, the supply pressure should be shut off and the pressure in the piping should be reduced to zero.

6- Before teaching, check the robot and its peripheral devices are all in the normal condition.

7- Do not operate the robot in the automatic mode while anybody is in the robot work area.

8- Make certain that their escape path is not obstructed inside the safety fence, or the robot operation area. Provided, however, that the robot secure the operation as a collaborative robot.

9- When a tool is mounted on the robot, or any moving device other than the robot is installed, such as belt conveyor, careful attention required for those motion.

10- Assign an expert near the operator panel who can press the EMERGENCY STOP button whenever he sees the potential danger.

11- In case of replacing a part, please contact your local manufacturer service. Wrong procedure may cause the serious damage to the robot and the worker.

12- Make sure that no impurity into the system in while (in) replacing or reinstalling components.

13- Turn off the circuit breaker to protect again electric shock in handling each unit or printed circuit board in the controller during inspection. If there are two cabinets, turn off the both circuit breaker.

14- A part should be replaced with a part recommended by manufacturer.

15- When restarting the robot system after completing maintenance work, make sure in advance that there is no person in the work area and that the robot and the peripheral devices are not abnormal.

16- In case of remove the motor or brake, suspend the arm by crane or other equipment beforehand to avoid falling.

17- Whenever grease is spilled on the floor, remove them as soon as possible to prevent from falling.

18- The following parts are heated. If a maintenance worker needs to touch such a part in the heated state, the worker should wear heat-resistant gloves or use other protective tools.

Servo motor, Inside of the controller, Reducer, Gearbox and Wrist unit

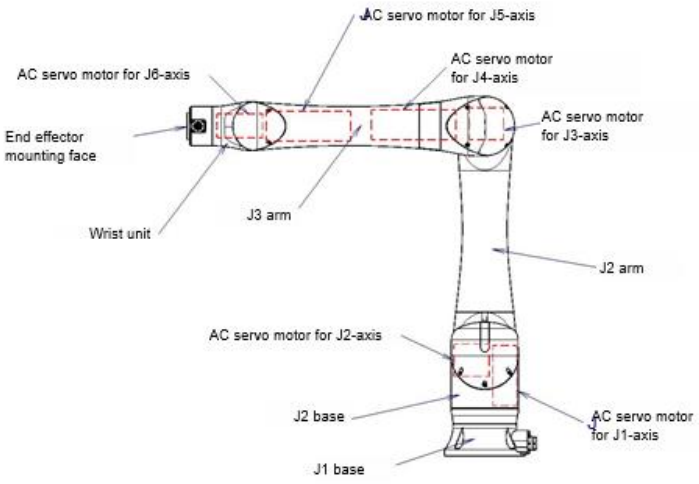


Figure 4. Mechanical unit

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

19- Maintenance must be done with appropriate lightning. Be careful that those lightning will not cause any further danger.

20- When a motor, reducer, or other heavy load is handled, a crane or other equipment should be used to protect maintenance workers from excessive load. Otherwise, the maintenance workers would be severely injured.

21- Must never climb or step on the robot even in the maintenance. If it is attempted, the robot would be adversely affected. In addition, a misstep can cause injury to the worker.

22- Secure footstep and wear the safety belt in performing the maintenance work in high place.

23- Remove all the spilled oil or water and metal chips around the robot in the safety fence after completing the maintenance.

24- All the related bolts and components must return to the original place in replacing the parts. If some parts are missing or left (remained), repeat the replacement work until complete the installation.

25- In case robot motion is required during maintenance, the following precautions should be taken :

Secure an escape route. And during the maintenance motion itself, monitor continuously the whole system so that your escape route will not become blocked by the robot, or by peripheral equipment.

Keep vigilant attention for the potential danger. and to press the emergency stop button whenever it is necessary.

26- Periodic inspection required. (Refer to the robot mechanical manual and controller maintenance manual.) A failure to do the periodical inspection can may adversely affect the performance or service life of the robot and may cause an accident

27- After replacing some parts, a test run required by the predetermined method. (See TESTING section of“Controller operator’s manual”. During the test run, the maintenance staff must work outside the safety fence as the need arises.

28- Make certain that their escape path is not obstructed inside the safety fence, or the robot operation area. Provided, however, that the robot secure the operation as a collaborative robot.

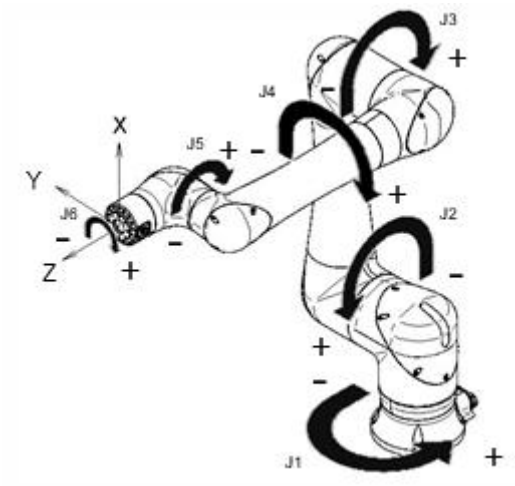


Figure 5. Each axes coordinatesexample

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

30. SAFETY OF THE TOOLS, PERIPHERAL DEVICES AND MECHANISM

30.1. Precautions In Programming Tools

1- Use a limit switch or other sensor to detect a dangerous condition and, if necessary, design the program to stop the robot when the sensor signal is received.

2- Design the program to stop the robot when an abnormal condition occurs in any other robots or peripheral devices, even though the robot itself is normal.

3- For a system in which the robot and its peripheral devices are in synchronous motion, particular care must be taken in programming so that they do not interfere with each other.

4- Provide a suitable interface between the robot and its peripheral devices so that the robot can detect the states of all devices in the system and can be stopped according to the states.

30.2. Precautions For Mechanism Tools

1- Keep the component cells of the robot system clean, operate the robot where insulated from the influence of grease, water, and dust.

2- Don’t use unconfirmed liquid for cutting fluid and cleaning fluid.

3- Adopt limit switches or mechanical stoppers to limit the robot motion, and avoid the robot from collisions against peripheral devices or tools.

4- Observe the following precautions about the mechanical unit cables. Failure to follow precautions may cause mechanical troubles.

* Use mechanical unit cable that have required user interface.

* Do not add user cable or hose to inside of mechanical unit.

* Please do not obstruct the movement of the mechanical unit when cables are added to outside of mechanical unit.

* In the case of the model that a cable is exposed, please do not perform remodeling (Adding a protective cover and fix an outside cable more) obstructing the behavior of the outcrop of the cable.

* When installing user peripheral equipment on the robot mechanical unit, please pay attention that equipment does not interfere with the robot itself.

5- The frequent power-off stop for the robot during operation causes the trouble of the robot. Please avoid the system construction that power-off stop would be operated routinely. (Refer to bad case example.) Please perform power-off stop after reducing the speed of the robot and stopping it by hold stop or cycle stop when it is not urgent. (Please refer to "STOP TYPE OF ROBOT" in SAFETY PRECAUTIONS for detail of stop type.)

* Whenever poor product is generated, a line stops by emergency stop and power-off of the robot is incurred.

* When alteration is necessary, safety switch is operated by opening safety fence and power-off stop is incurred for the robot during operation.

* An operator pushes the emergency stop button frequently, and a line stops.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

* An area sensor or a mat switch connected to safety signal operates routinely and power-off stop is incurred for the robot.

* Power-off stop is regularly incurred due to an inappropriate setting for Dual Check Safety (DCS).

6- Power-off stop of Robot is executed when collision detection alarm (SRVO-050) etc. occurs. Please try to avoid unnecessary power-off stops. It may cause the trouble of the robot, too. So remove the causes of the alarm.

30.3. Precautions In Programming Mechanism

1- When the work areas of robots overlap, make certain that the motions of the robots do not interfere with each other.

2- Be sure to specify the predetermined work origin in a motion program for the robot and program the motion so that it starts from the origin and terminates at the origin.

Make it possible for the operator to easily distinguish at a glance that the robot motion has terminated.

1- To control the pneumatic, hydraulic and electric actuators, carefully consider the necessary time delay after issuing each control command up to actual motion and ensure safe control.

(2) Provide the end effector with a limit switch, and control the robot system by monitoring the state of the end effector.

30.4. Procedure To Move Arm Without Drive Power In Emergency Or Abnormal Situations

For emergency or abnormal situations (e.g. persons trapped in or pinched by the robot), brake release unit can be used to move the robot axes without drive power.

Please refer to this manual and mechanical unit operator's manual for using method of brake release unit and method of supporting robot.

30.5. Precautions in Network System

Ethernet cable which is used to connect networking devices through UTP cable and end is terminated with RJ45 connector. In UTP cable consist of 4pair or 8 wire of different color that is used to terminate on RJ45 or 8P8C connector. Ethernet cable color coding as standardized by EIA(Electronic Industries association) and TIA(Telecommunication Industry Association) there are two standard EIA/TIA-568-A and EIA/TIA-568-B.

Network cables should be made by technicians when necessary.

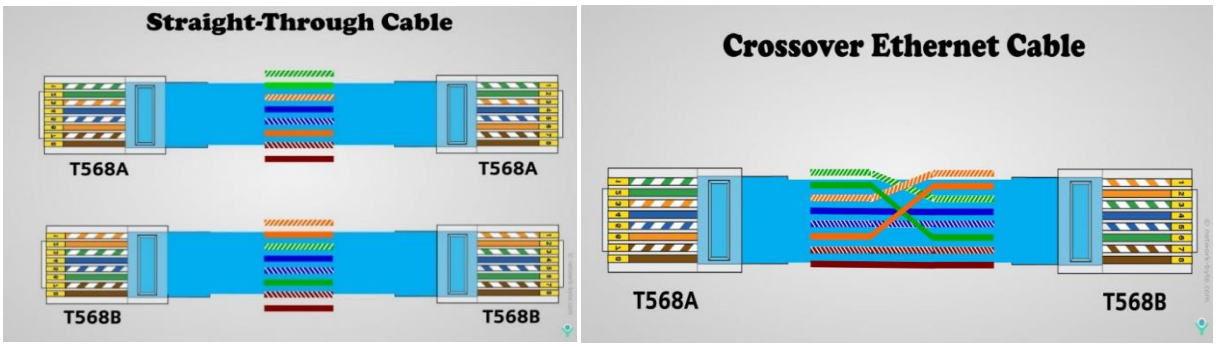


Figure 6. Connection Cables

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

WORKSHEET 15

MAKING AND TESTING NETWORK CABLE

The aims at the end of this worksheet are:

- Knowing how to use network cable
- To be able to make a network cable used to connect between robots and computers.
- To test whether it works or not.

Introduction :

UTP cable contains 4 pairs of copper wires. In order to reduce the electromagnetic effect of the cables on each other, the copper cables are wrapped two by two. Due to its small circumference, it takes up less space in cable ducts and provides a great advantage in large network installations.

RJ (Registered Jack) series connectors are used to terminate twisted pair cables. There are dozens of connector types in the RJ series. The most common of these are RJ-12, which terminates Category 2 (Cat2) cables used in telephone systems, and RJ-45 connectors used to terminate UTP and STP cables. Some tools are required when attaching these connectors to the cable.

These tools are required to strip the cable, separate the twisted pairs, cut the cable and insert the cable into the connector.

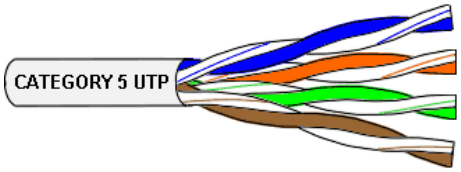


Figure 7. UTP Cat5 Cable

Material List:

- 2 RJ-45 connectors (Cat5)
- 2 pcs RJ-45 insulator cover
- 0.5 metre Cat 5e Cable
- 1 pcs clamping pliers used for RJ-45, RJ-12 connectors
- 1 tool for cleaning and cutting twisted pairs

Process Steps:

1- Firstly, the cable to be prepared is cut and an insulating cover is attached to the end. The caps prevent the cable from being damaged during bending and twisting.

2- The top layer of the cable is cut and removed as a ring with the tool required to remove the outermost layer of insulation.

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

3- The twisted pairs must be unwound in order to insert the cable into the connector. The pairs are unwound up to the edge of the sheath of the cable. The pairs must be placed in a row. For this purpose, the cable is made flat.

4- The pairs must be laid in such a way that a flat layer is formed from the conductors laid in parallel. With cable crimping pliers, a 14 mm piece of the conductors is cut from the edge of the sheath of the cable.

5- The conductors are colour sorted in accordance with the selected standard (T568A or T568B). The common standard in this order is EIA/TIA-T568B (from left to right: orange-white, orange, green-white, blue, blue-white, green, brown-white, brown).

If the cable is to be connected from a PC to a network device, the connector on both ends of the cable must be the same. They must be prepared according to the standard. (Straight Connection)

If the cable is to be connected from one network device to another network device or from one PC to another PC, when the connectors at the ends of the cable are connected to each other according to different standards, it should be prepared.(Cross Connection)

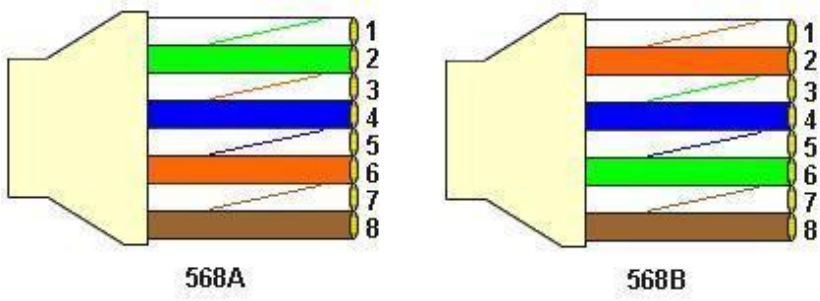


Figure 8. EIA/TIA Cable connection standards

6- The conductors must be arranged in the connector in such a way that they are located in individual channels and that the sheath of the cable enters the connector by at least 6 mm. The fixing key of the connector must be directed downwards.

7- The conductors must be inserted into the connector all the way. The cable must be well seated in the connector so that the blades at the end of the connector can make contact with the conductors. Since the connector is made of transparent plastic, the condition of the conductors can be checked visually. After this stage, the conductors should be checked well as the mistake made will not be reversible.

8- RJ-45 pliers are used to connect the twisted pairs with the blades of the connector. With this process, the blades of the connector enter the connector, cut the sheaths of the conductors

and enter between the wires of the cable and provide electrical contact. Thanks to the RJ-45 pliers, the connector is mounted in such a way that it does not come out of the cable.

9- In the connector of classical construction, the cable is fixed by pressing the cable in the form of a flat lath (length, more or less 7 mm). This pressure, which is an integral part of the connector, ensures that the cable is tightly compressed. In this way, the load of the cable does not fall on the front blades.

10- The insulating cover of the connector is fitted.

11- The process of attaching the cable to the connector is completed. To be sure, the cable and connector are pulled in opposite directions with a small force to check the strength of the assembly.

12- Finally, both ends of the cable are tested with cable testers.

Figure 9. Cable testing aquipment

Report:

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

4. CHECKS AND MAINTENANCE

The robot and robot system must have an inspection and maintenance program to ensure continued safe operation of the robot system.

The inspection and maintenance program must take into account the robot and robot system manufacturer's recommendations.

Personnel who perform maintenance or repair on robots or a robot system must be trained in the procedures necessary to perform safely the required tasks.

Personnel who maintain and repair robot systems must be safeguarded from hazards.

Where possible, maintenance must be performed from outside the safeguarded space or robot operating space or neighborhood by placing the robot arm in a predetermined position.

The results of risk assessment may admit that people they don't maintain or repair but trained about collaborative robot access to the robot operating space and neighborhood easily, during maintenance. In this case, confirm that the contact stop function is enabled.

The following is the safety procedure of entering safeguarded space for maintenance.

Stop the robot system.

Shut off the power of the robot system, and lock the main breaker to prevent powering on during maintenance, by mistake.

If you have to enter the safeguarded space while power is available to the robot system, you must do

the following things prior to entering the safeguarded space:

check the robot system to determine if any conditions exist that are likely to cause malfunctions,

check if the teach pendant works correctly, and

if any damage or malfunction is found, complete the required corrections and perform retest before personnel enter the safeguarded space.

Enter the safeguarded space

After the maintenance working, check if the safeguard system is effective. If it has been suspended

to perform the maintenance working, return their original effectiveness.

4.1. PERIODIC MAINTENANCE

Daily maintenance and periodic maintenance/inspection ensure reliable robot performance for extended periods of time. Before operating the system each day, clean each part of the system and check the system parts for any damage or cracks.

The periodic maintenance procedures described in this chapter assume that the FANUC robot is used for up to 3840 hours a year. In cases where robot use exceeds 3840 hours/year, adjust the given maintenance frequencies accordingly. The ratio of actual operation time/year vs. the 3840 hours/year should be used to calculate the new (higher) frequencies. For example, when using the robot 7680 hours a year with a recommended maintenance interval of 3 years or 11520 hours, use the following

calculation to determine the maintenance frequency: 3 years / 2 = perform maintenance every 1.5 years.

4.1.1. Daily Checks And Maintenance

(1) Daily maintenance

Before operating the system each day, clean each part of the system and check the system parts for any damage or cracks. Also, check the following:

Before operation

Check the cable connected to the teach pendant for excessive twisting.

Check the controller and peripheral devices for abnormalities.

Check the safety function.

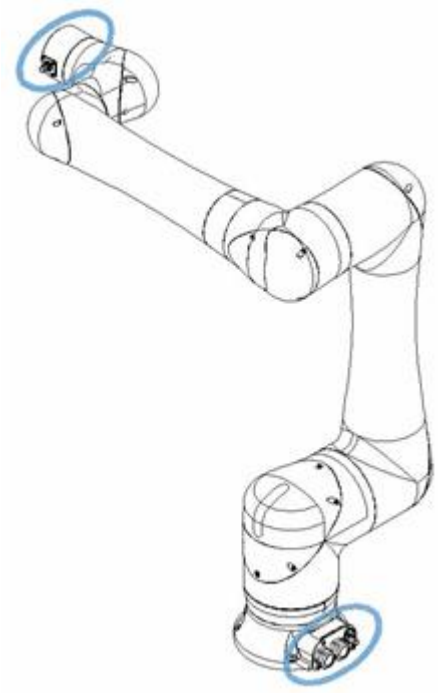


Figure 6. Connector Inspection points

After operation

At the end of service operation, return the robot to the proper position, then turned off the controller.

Clean each part, and check for any damage or cracks.

If the ventilation port and the fan motor of the controller are dusty, wipe off the dust.

(2) Check after one month

Check that the fan is rotating normally. If the fan has dirt and dust built up, clean the fan according to step (d) described below for inspection to be performed every 6 months.

(3) Periodic inspection performed every six months

Please refer to the Section 7.5, and then remove any dirt and dust from the inside of the transformer

compartment. Wipe off dirt and dust from the fan and transformer.

(4) Battery daily check

Replace the battery on the front panel of the main board every 4 years. Please refer to the Section

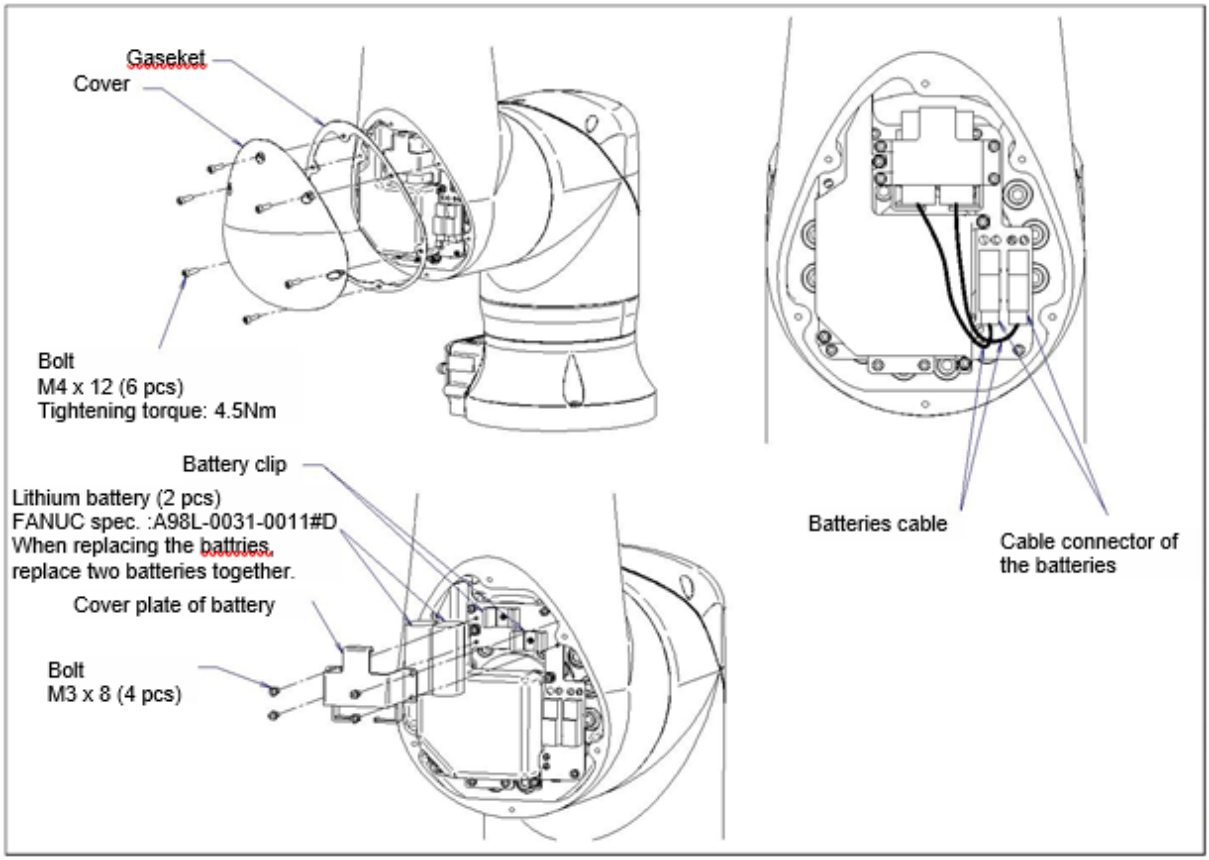


Figure 7. Replacing the battery

(5) Maintenance tools

AC/DC voltmeter (A digital voltmeter is sometimes required.)

Oscilloscope with a frequency range of 5 MHz or higher, two channels

Cross-head screwdrivers: Large, medium, and small

Straight-head screwdrivers: Large, medium, and small

Nut driver set (Metric)

Pliers

Cutting pliers

Diagonal cutting pliers

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union

Check items	Check points and management
Oil seepage	Check to see if there is oil on the sealed part of each joint. If there is an oil seepage, clean it.
Vibration, abnormal noises	Check whether vibration or abnormal noises occur. When vibration or abnormal noises occur, perform measures referring to the following section:
Positioning accuracy	Check that the taught positions of the robot have not deviated from the previously taught positions. If displacement occurs, perform the measures as described in the following section:
Peripheral devices for proper operation	Check whether the peripheral devices operate properly according to commands from the robot and the peripheral devices.
Brakes for each axis	Check that the droppage of the end effector is within 5 mm when the servo power turned off. If the end effector (hand) drops, perform the measures as described in the following section:
Warnings	Check whether unexpected warnings occur in the alarm screen on the teach pendant. If unexpected warnings occur, perform the measures as described in the following manual:

Table 2. List of check items

4.1.2. Periodic Check and Maintenance

Check the following items at the intervals recommended below based on the total operating time or the accumulated operating time, whichever comes first. (○ : Item needs to be performed.)

In addition to the maintenance table below, the following operations can be performed;

To keep the robot system safe, please perform periodic maintenance those are specified in operator's manual or maintenance manual.

In addition, please clean each part of the system and visually check them for any damage or cracks.

Daily check items are as follows (but not limited to).

Input power voltage

Pneumatic pressure

Damage of connection cables

Looseness of connectors

Lubrication

Emergency stop functions

Effectiveness of deadman switch on teach pendant

Safety gate interlocks (in case the robot system has safety gate interlocks)

Vibration, noise by the robot movement

Functions of peripheral devices

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

Fixtures of robot and peripheral devices

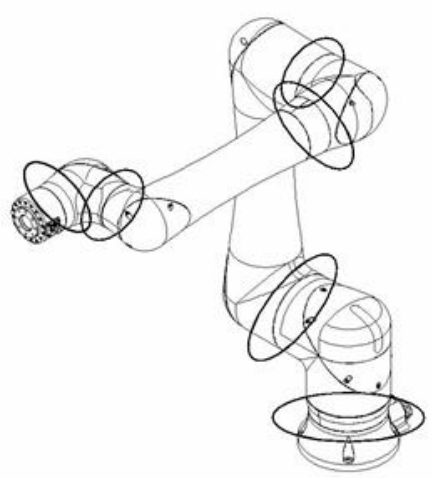


Figure 6. Checks parts of oil seepage

PERIODIC MAINTENANCE TABLE

Items			Accumulated operating time (H) / Check time	Oil Grease amount	First check 320	3 months 960	6 months 1920	9 months 2880	1 year 3840	4800	5760	6720	2 years 7680	8640	9600	10560
Mechanical unit	1	Check for external damage or peeling paint	0.1H	-		○	○	○	○	○	○	○	○	○	○	○
	2	Check for water	0.1H	-		○	○	○	○	○	○	○	○	○	○	○
	3	Check the end effector (hand) cable	0.1H	-		○			○				○			
	4	Check the exposed connector.(Loosening)	0.1H	-		○			○				○			
	5	Tighten the end effector bolt	0.1H	-		○			○				○			
	6	Tighten the cover and main bolt	1.0H	-		○			○				○			
	7	Remove spatter and dust etc.	1.0H			○			○				○			
Controller	8	Check the robot cable, teach pendant cable and robot connecting cable	0.2H	-		○			○				○			
	9	Cleaning the controller ventilation system	0.2H	-	○	○	○	○	○	○	○	○	○	○	○	○

●requires order of parts ⊖does not require order of parts

Table 3. Cobot Periodic Maintenance Table Example (FANUC)

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

Check and maintenance intervals (Operating time, Accumulated operating time)							Check and maintenance item	Check points, management and maintenance method
1 month 320h	3 months 960h	1 year 3840h	2 years 7680h	3 years 11520h	4 years 16380h	5 years 30720h		
○ Only 1st check	○						Cleaning the controller ventilation system	Confirm the controller ventilation system is not dusty. If dust has accumulated, remove it.
	○						Check for external damage	Check whether the robot has external damage due to the interference with the peripheral devices. If an interference occurs, eliminate the cause. Also, if the external damage is serious and causes a problem in which the robot cannot be used, replace the damaged parts. (Perform diary checks for green covers.)
	○						Check for water	Check whether the robot is subjected to water or cutting oils. If water is found, remove the cause and wipe off the liquid.
	○ Only 1st check	○					Check for damages to the teach pendant cable, the operation box connection cable or the robot connection cable	Check whether the cable connected to the teach pendant, operation box and robot are unevenly twisted or damaged. If damage is found, replace the damaged cables.
	○ Only 1st Check	○					Check for damage to the end effector (hand) connection cable	Check whether the end effector connection cables are unevenly twisted or damaged. If damage is found, replace the damaged cables.
	○ Only 1st check	○					Check the exposed connectors	Check the connection of exposed connectors.
	○ Only 1st check	○					Retightening the end effector mounting bolts	Retighten the end effector mounting bolts. Refer to the following section for tightening torque information:
	○ Only 1st check	○					Retightening the external main bolts	Retighten the robot installation bolts (according to procedure in Section 1.2), bolts to be removed for inspection, and bolts exposed to the outside. Refer to the recommended bolt tightening torque guidelines at the end of the manual. An adhesive to prevent bolts from loosening is applied to some bolts. If the bolts are tightened with greater than the recommended torque, the adhesive might be removed. Therefore, follow the recommended bolt tightening torque guidelines when retightening the bolts.
	○ Only 1st check	○					Clean spatters, sawdust and dust	Check that spatters, sawdust, or dust does not exist on the robot main body. If dust has accumulated, remove it. Especially, clean the robot movable parts well (each joint, surroundings of the wrist flange, conduit part, wrist axis hollow part).

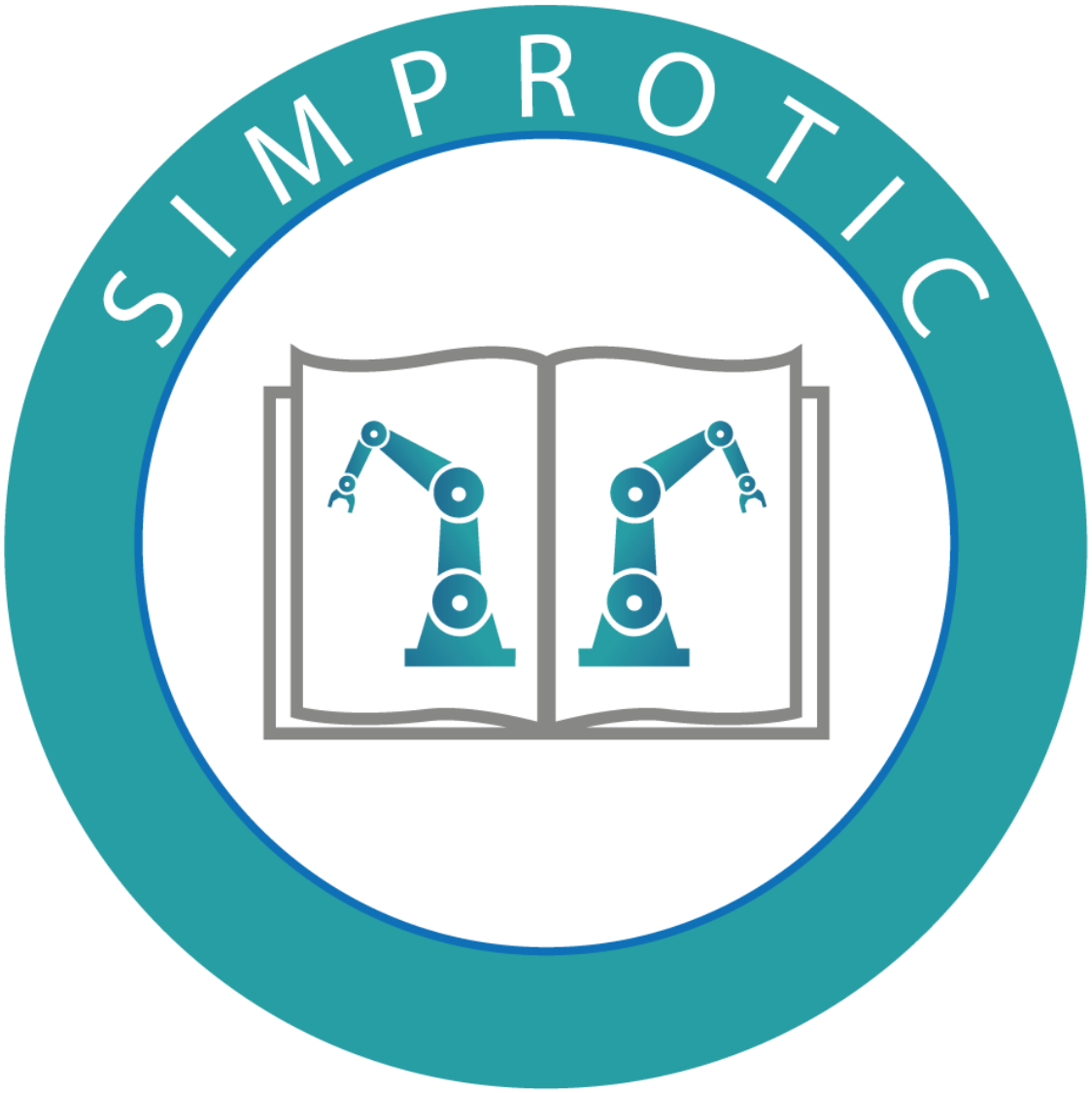
TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

REFERENCES

- ph.d. Serkan DİŞLİTAŞ. Industrial Robot Programming, Çorum 2015
- RoboDK Basic Manuel, https://robodk.com/
- Universal Robot eBooks, https://www.universal-robots.com/tr/akademi/
- International Federation of Robotics Reports, Frankfurt 2018
- Republic of Türkiye Ministry of National Education – Programming Basics 9, Ankara 2023
- FANUC Maintenance Manuel, https://fanuc.eu/

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY

Funded by
the European Union



TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by
the European Union