

Міністерство освіти і науки України
 Національний технічний університет
 «Дніпровська політехніка»

Навчально-науковий інститут електроенергетики

(інститут)

Електротехнічний факультет

(факультет)

Кафедра кіберфізичних та інформаційно-вимірювальних систем

(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА кваліфікаційної роботи ступеню магістра

студента Снитко Анастасія Олександрівна

(П.І.Б.)

академічної групи 151М-19-1

(шифр)

спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології

(код і назва спеціальності)

за освітньо-професійною програмою 151 Автоматизація та комп'ютерно-інтегровані технології

(офіційна назва)

на тему Синтез та дослідження системи керування промисловим роботом компанії Fanuc в умовах безперервного виробництва

(назва за наказом ректора)

Консультанти	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинг.	інституційною	
Керівник кваліфікаційної роботи	проф. Ткачов В.В.			
Провідний консультант	ст.викл. Бойко О.О.			
Синтез системи керування	доц. Бубліков А.В.			
Експериментальний розділ	ст.викл. Бойко О.О.			
Економічна частина	ст. викл. Яремчук І.О.			
Охорона праці та безпека в надзвичайних ситуаціях	проф. Чеберячко Ю.І.			
Рецензент				
Нормоконтролер	ас. Славінський Д.В.			

Дніпро
2020

ЗАТВЕРДЖЕНО:
завідувач кафедри
кіберфізичних та інформаційно-
вимірювальних систем
(повна назва)

_____ Ткачов В.В.
(підпис) (прізвище, ініціали)

« _____ » _____ 2020 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня магістра

студенту Снитко А.О. академічної групи 151М-19-1
(прізвище та ініціали) (шифр)

спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології

за освітньо-професійною програмою 151 Автоматизація та комп'ютерно-інтегровані технології
(офіційна назва)

на тему Синтез та дослідження системи керування промисловим роботом компанії Fanuc в умовах безперервного виробництва

затверджену наказом ректора НТУ «Дніпровська політехніка» від 20.11.2020 № 965-с

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання.	Вступ. Опис технологічного процесу для об'єкта автоматизації. Огляд існуючих систем автоматизації. Стан питання. Вибір напрямку створення автоматизованої системи.	12.10.2020
Теоретичний розділ	Розробка й обґрунтування методів та інструменту дослідження системи керування	02.10.2020
Синтез системи керування	Обрання структури системи керування та її синтез. Дослідження функціонування системи керування на базі обраного алгоритму керування об'єктом автоматизації	02.11.2020
Експериментальний розділ	Розробка програмного забезпечення системи керування на основі запропонованого алгоритму керування об'єктом автоматизації	23.11.2020
Економічна частина	Економічне обґрунтування доцільності витрат на створення системи керування.	30.11.2020
Охорона праці та безпека в надзвичайних ситуаціях	Розробка організаційно-технічних заходів, щодо реалізації правил безпеки при експлуатації системи.	07.12.2020

Завдання видано

_____ (підпис кер.)

проф. Ткачов В.В.
(прізвище, ініціали)

Дата видачі 01.09.2020

Дата подання до атестаційної комісії 15.12.2020

Прийнято до виконання

_____ (підпис студента)

Снитко А.О.
(прізвище, ініціали)

ABSTRACT

Explanatory note: __ p., __figs., __tab.,__ app., __sources.

Object of research: to design a robotic automatic system applied to a production process in an industrial environment by using a simulation software tool.

The purpose of the project: the study and analysis of the capabilities and advantages offered by current tools for offline simulation in production environments compared to programming carried out in the real environment.

The entire robotic system is a continuous system that constantly analyzes the state of the system, its position and its entry and exit signals. The positioning accuracy of the robot in the simulation program is 0.001 mm, the positioning accuracy of the real robot is 0.05 mm.

To implement the process, one must assimilate and understand the programming methods that the simulation tool supports, as well as the different functionalities and configuration options of this software to carry out the entire process virtually.

In this way, the real process will be carried out in a simulated way, having previously carried out the total configuration of the system: definition of robot tools, specification of reference systems, definition of parts and objects, definition of fixed parts, definition of points, the tracing of the different trajectories. The robotic system implemented by simulating the process will be analyzed.

It will also be intended to carry out an expansion in the implemented system to evaluate the ability to insert changes in the work cell through the use of the simulation tool. The concept of simultaneous engineering applied to industrial environments capable of implementing robotic simulation in their production lines will be analyzed and will be represented general software development methodology.

After carrying out this project, we must be able to face the "offline" programming of a workcell applied to the production of an industrial plant.

ROBOTIC SYSTEM, CONTROL, ROBOT, RESEARCH, FANUC ROBOTS,
ROBOGUIDE, SIMULATION, INDUSTRIAL PROCESS

CONTENT

List of acronyms	6
Introduction	7
1 Status of the question and statement of the task	9
1.1 Industry	9
1.2 Simulation in industrial robotics	10
1.3 Analysis of robotic systems	17
1.3.1 Characteristic of the manipulator's universe	17
1.3.2 Physical characteristics of the process	17
1.3.3 Mathematical descriptions of robot's motions	19
1.4 Choice justification of Roboguide FANUC	20
1.5 Research task statement	29
1.6 Conclusions	30
2 FANUC robot programming	31
2.1 TPE programming	31
2.2 Programming in the simulator software editor	34
2.3 Positioning error of industrial robots	34
2.4 Conclusions	38
3 Robotic system simulation	40
3.1 Workcell design	40
3.2 Automatic process diagram. Petri net	51
3.3 Presentation of the manipulator	58
3.4 Automatic system implementation	40
3.5 Definition and configuration of the robot and the controller	51
3.6 Definition of manipulated objects (PARTS)	58
3.7 Tool definition and configuration (UTOOL)	40
3.8 Creation and configuration of fixed parts (FIXTURES)	51
3.9 Definition of user reference system (UFRAME)	58
3.10 Creation of the program	40
3.11 Simulation of the process	51

	5
3.12 Conclusions	58
4 Extension of the work cell	59
4.1 Robotic cell redesign	59
4.2 Cell configuration with additional robot	63
4.3 Creation of the extended program	66
4.4 System simulation. Testing	69
4.5 General software development methodology	70
4.6 Conclusions	70
5 Solution for the transmission of information between PLC and robot	70
6 Economics	71
7 Labor protection, industrial safety and civil protection	84
Evaluation of results and conclusions	90
References	91
Appendix A – Robotic system software for the first simulation	92
A.1 Main program	92
A.2 Definitions of positions for the main program	94
Appendix B – Robotic system software for the extended simulation	98
B.1 List of programs	92
B.2 First robot’s program	98
B.2.1 Listing of the program	98
B.2.2 Definitions of positions for the program	98
B.3 Second robot’s program	98
B.3.1 Listing of the program	98
B.3.2 Definitions of positions for the program	98

LIST OF ACRONYMS

TP – Teach Pendant;

PC – personal computer;

CAD – computer-aided design;

TPP – Teach Pendant programming;

OLP – offline programming;

CAM – computer aided manufacturing;

EOAT – end of arm tool;

TPE – Teach Pendant editor;

TCP - tool center point or tool central point.

INTRODUCTION

The industrial robot is a programmable manipulator in three or more axes with various purposes, automatically controlled and reprogrammable. The field of industrial robotics can be defined as the research, design and use of robots for the execution of industrial processes.

Industry 4.0 will play an increasingly important role in global manufacturing. As obstacles such as systems complexity and data incompatibility are overcome, manufacturers will integrate robots into networks of machines and systems throughout the factory. Robot manufacturers are already developing and commercializing new service models: these are based on real-time data collected by sensors that connect to the robots.

Analysts predict a rapidly growing market for cloud robotics in which data from one robot is compared to data from other robots in the same or different locations. The network allows these connected robots to perform the same activities. This will be used to optimize the robot's motion parameters such as speed, angle, or force. Ultimately, the advent of big data in manufacturing could redefine the industry boundaries between equipment manufacturers and manufacturers.

At present, industrial robots have improved the working time and quality of some products in large factories, even more they have also helped man in some tasks that are highly dangerous, leaving the work to a mechanical arm.

ROBOGUIDE is the leading of offline programming product on the market for FANUC robots. The ROBOGUIDE family of process focused software packages allows users to create, program and simulate a robotic workcell in 3-D without the physical need and expense of a prototype workcell setup. With virtual robots and workcell models, of offline programming with ROBOGUIDE reduces risk by enabling visualization of single and multi-robot workcell layouts before actual installation. To ensure minimal impact on production, cells can be designed with imported CAD tested and modified entirely offline. Built on Virtual Robot Controllers to give you accurate motion and cycle times.

FANUC, one of the leading four robot manufacturers, provides not one but two different programming languages: Teach Pendant (or TP) and KAREL. TP programs are

binary files that can be edited through the robot's teach pendant buttons (or touch screen for newer robots). TP files can also be compiled/decompiled from an LS file (human-readable ASCII file). TP programs offer a limited assembler-like functionality. Alternatively, with FANUC, you can program your own algorithms using a PC and KAREL (programming language based on Pascal), but KAREL does not allow you to do robot movements nor edit the program from the controller's teach pendant.

ROBOGUIDE incorporates many application-specific tools into its software options. With HandlingPRO software, there is no need for a prototype work cell setup. This software allows users to simulate a robotic process or study feasibility options for applications in a 3D space.

1 STATUS OF THE QUESTION AND STATEMENT OF THE TASK

1.1 Industry

Industrial robotics and automation are the pillars that have made possible the consolidation of Industry 4.0, in addition to bringing with it numerous benefits for the productivity and efficiency of production resources.

The different models of industrial automation that are established today eliminate the subjective factor of human decisions, achieving lower margins of error and more precise processes, while freeing human labor from repetitive or dangerous tasks.

Countless manufacturers use industrial robots to automate tasks, improve worker safety, and increase overall production, while reducing waste and operating costs. With the increasing prevalence of industrial robots in manufacturing environments, there has been an increase in demand for many different types of industrial robots to suit specific applications and industries.

Industrial robots are mechanical devices that, to some extent, replicate human movements. They are used whenever it is necessary to reduce danger to a human, provide more force or precision than a human, or when continuous operation is required. Most robots are stationary, but some move around the job site delivering materials and supplies.

Industrial robotics are used to perform highly accurate and repeatable tasks that result in higher quality products. The ability of industrial robots to work continuously without rest is helping manufacturers to increase production. Furthermore, robots can work in dangerous and noxious environments, thus improving working conditions and safety in the production plant. Thus, the various advantages of industrial robots are encouraging manufacturers to integrate different types of industrial robots into their production line to increase plant efficiency and profitability, that is, technological maturity.

1.2 Simulation in industrial robotics

When a complete study is carried out for the automation of an industrial process, it is necessary to use the simulation of the process using computers to be able to evaluate the system once it has been designed and thus avoid possible failures and mismatches with the

actual robotic system mounted. The main reason for carrying out the simulation is mainly the prevention of failures and possible damage to the robotic arm mechanisms and the production environment, thus avoiding costs derived from online programming.

Today, the software tools, designed mostly by the manufacturers of the industrial robots themselves, allow us to analyze the process through the complete simulation of the robotic cells using computers.

The simulation concept is possible thanks to the "offline" programming of the system using specific software. The characteristics of these applications are so advanced that they allow us to adjust to the maximum all the parameters of the movements and actions of our robot, including transferring this programming directly to the robot and avoiding reprogramming of the entire process once the system is physically mounted on the production environment.

It is worth mentioning that the interface for the "offline" programming of robotic systems is much more efficient than the programming that we can carry out using the "teach pendant" controllers or through a "lead-by-the-nose" to record the movements that the robot has to perform in the determined process "online".

For "offline" programming, a graphic environment is currently required in the programming and simulation software, in order to visualize the process, having previously defined the "world" of the arm's working environment. Thanks to the power of computers and the high performance of computer-aided design (CAD) software systems, the systems used for 3D modeling are ideal for today's design and manufacturing environments, especially for robotic industrial processes, such as for the manipulation of objects or parts, welding processes, painting, deburring, assembly of parts, palletizing. In general, we can affirm that any industrial process that has a level of repetition within a production chain can be robotized, benefiting from the advantages that current systems provide.

These programming techniques known as offline programming OLP used appropriately would allow us to perform the following tasks before working in a productive environment:

1. Develop models, test them and optimize them, before being used in the manufacture of parts and tools. With this, it is possible to analyze the behavior of very complex systems that are difficult to evaluate by other procedures.

2. Know what the behavior of the systems will be before building them, without losing sight of the fact that the final values of the simulators will be approximations of the real values. This implies that the technicians who carry out the design using simulators must know how to correctly interpret the simulation results at all times.

3. Reprogram the process outside of a manufacturing line that is already in production, if for any reason needs change. With this method the downtime of machines and robots is reduced.

4. Anticipate the operation and commissioning of production lines, since it is an independent system and can be carried out in parallel with the assembly of robotic cells.

5. Improve the quality and accuracy of welding points that could be performed by operators, even the most expert ones.

6. Correctly design the trajectories of the terminal element (gripper or claw) as well as its speeds and accelerations. During modeling, simulator programs report any collision or loss of proximity between the elements of the model and the environment.

Clamp – tool used to capture and manipulate objects with a prehensile action in the automated process with robots.

Claw – fixed tool used to capture objects that will be manipulated in an industrial production process.

Concurrent Engineering, also called Simultaneous Engineering, Parallel Engineering, Total Engineering or Integrated Product Design, among other names, is a philosophy that directly affects the culture of organizations and rethinks the conventional way of working on projects. We will consider some definitions that detail and summarize the concept as such.

The term Concurrent Engineering initially emerged in the summer of 1986 when it was used in report R-338 of the Institute for Defense Analysis (IDA) from which one of the most universally accepted definitions arises: “A systematic effort for an integrated, concurrent design of the product and its corresponding manufacturing and service process”.

It intends that those responsible for development from the beginning take into account all the elements of the Product Life Cycle (PLC), from the conceptual design to its availability, including quality, cost and user needs: “Simultaneous engineering is the simultaneous project of a product and its manufacturing process. It is an integrated approach to product development that emphasizes customer expectations by manufacturing high-quality products, faster and at lower cost. Supports the values of multidisciplinary teamwork, such as cooperation, trust, and sharing and exchanging knowledge and information, so that decision-making during the design stage proceeds with emphasis on the simultaneous consideration of all aspects of the product life cycle”.

A group working on product design or a committee drawn from different departments that meets regularly are not Concurrent Engineering. For Simultaneous Engineering to really apply, the team must be comprised of product design engineers, manufacturing engineers, marketing, purchasing, finance, and major suppliers of manufacturing equipment and components.

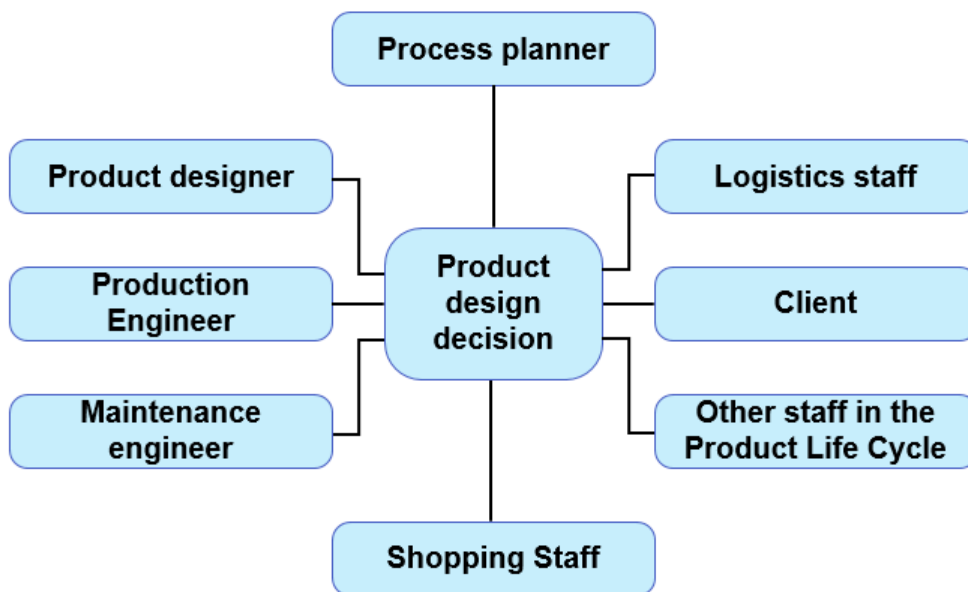


Figure 1.1 Concurrent Engineering Multidisciplinary Task Team

It must also be permanent throughout the duration of the project so that your work receives the priority it requires. The key in multidisciplinary work teams is that from the start, when the design is still only a sketch, the manufacturing engineers who are part of the

team have as much information about the product as any other member of the group. This way they can begin to plan the manufacturing facilities with the same concept with which the design engineers are planning the object that they are going to produce, that is, they carry out simultaneous work.

There is a direct and permanent relationship that allows recommendations to reduce costs and number of pieces, considerably increasing quality. Having marketing staff on the team ensures that sales goals are achievable, and most importantly, this staff contributes directly to emphasizing customer expectations, allowing greater weight to be given to this aspect in the simultaneous engineering than in traditional engineering. This approach makes it possible to identify, at an early enough stage, points where rectifications should be made, at a much lower cost than if they were done later. As we can see, the concept of simultaneous engineering adapts perfectly to the production environments programmed by means of simulation software, making it easier to reduce process corrections and ultimately the cost of the product life cycle.

It has been shown in projects carried out that 80% of the manufacturing cost is determined at the design stage, which is one of the first within the product life cycle.

This determines the ultimate importance of the design stage and how the activities carried out here can impact the entire organization. To illustrate the concept, the figure below shows the life cycle of a product development project under both approaches: that of traditional engineering and that of simultaneous engineering, which is notable for its product life cycle time (PLC) reduction.

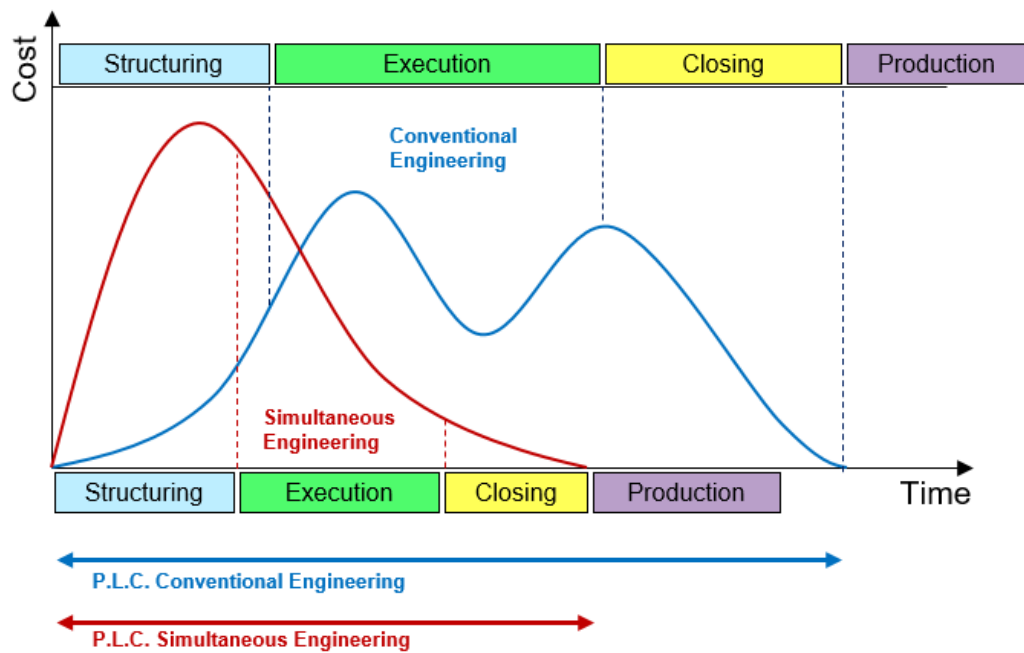


Figure 1.2 Product life cycle under the traditional and concurrent approach

Among the specialized bibliography we highlight the book “Concurrent Engineering” by John R. Hartley, where it clearly shows what traditionally happens in companies depending on their “culture”. Conventional engineering drives a sequential approach to the product development process. According to this approach, each area of the company, after executing its part, transfers its result to the next sector and is waiting. Each organizational unit that receives the information will inevitably find fault according to the perspective of its own specialty, and will return it to the sector of origin for the corresponding adjustments.

The traditional or compartmentalized system generates conflicts and consequently brings many changes and feedbacks in the different stages, originated because some necessary characteristics in the later stages were not considered from the beginning of the process, a fact that directly affects the increase in costs and product development time. Using traditional engineering, the quality of the product is put at risk, either because the corrective measures were not taken or because the changes are seen as rectifications that would not exist if an integrated product design had been worked on from the beginning. In the graph we can see that in the early stages the time increases, but the total cycle time is substantially reduced. A key principle of concurrent engineering is the introduction of quality from the

start of design, eliminating any element that may be affected by variations in production. A culture is required in which each of the people involved in the process is responsible for quality. This is where the concurrent engineering philosophy and the total quality philosophy, whose common goal is customer satisfaction, totally fit.

Taking on the challenge of working under the simultaneous engineering approach requires time and a special effort to train human talent, initially to change the way of working and later trained in the techniques and skills required for its implementation. Initially, there may be resistance and conflicts on the part of people, especially by the area managers who feel the loss of power in favor of the project teams, or also by the pressure of the deadlines for the delivery of tasks that were previously carried out after months, and with the simultaneous approach many times they should be taken in a few days.

It is definitely not easy when sequential engineering with well-defined stages has been used, where each department has responsibility for a particular function and it is developed after the previous one has been completed.

The hierarchical systems that most organizations have make teamwork difficult, so it is necessary to be careful and prepare for a transition process towards more flexible and dynamic organizational structures that facilitate and support the development of new attitudes, valuing and promoting teamwork, as well as towards performance evaluation systems where not only technical competence is taken into account, but also creativity and success as members of a team.

Sometimes the trend is a reorganization with an emphasis on the market and competitiveness, as in the case of circular structures, matrix structures, and strategic business units.

The current paradigms that govern both the training of professionals and the performance of human talent in companies tend to value individual work. Concurrent engineering requires a cultural change in which creativity, versatility, multidisciplinary work, trust and enthusiasm are the *raison d'être* of the conception of work. In many world-class companies, especially Japanese ones, the simultaneous approach is so ingrained in their culture that teamwork is inherent in political decisions and day-to-day operations and, of course, they do not conceive of a different way of working. It is simply common sense.

Although it is not strictly necessary for the company to make strong investments in computers, sophisticated software and state-of-the-art equipment to apply the fundamentals of concurrent engineering, there is no doubt that it would be ideal and of great importance to have tools and techniques that facilitate and expedite the decision-making process, as well as that of cultural change. The challenge to manufacture today is to rely on techniques that remain a competitive advantage through the use of information technology and tools that allow efficient linking throughout the entire value chain.

However, as long as traditional management techniques continue to be used, it cannot be said that concurrent engineering is being applied. Leadership has to come from above, with senior managers resolutely supporting and facilitating the means to develop the capacities of work teams. In addition, they must practice a delegation policy, allowing the team to carry out the project with great autonomy, to the point that if a senior manager really believes that the project is going in the wrong direction, he must gather the entire work team to reach a true consensus decision.

Essential tools for Simultaneous Engineering include Computer Aided Design (CAD) Systems, which are very important because they allow simulation in parallel, which reduces the risk when deciding on the most practical option. To maximize the benefits of Concurrent Engineering, one must tend towards the use not only of Design but also of its integration with Computer Aided Manufacturing (CAD/CAM).

With a correct combination of "hardware" and "software" that allows engineers from different disciplines to work in parallel, prototype manufacturing can be reduced and its lead times can be reduced considerably. Likewise, Computer Aided Engineering (CAE) is another tool that allows projects to be developed more efficiently through software. It is important to mention that the use of CAO / CAM / CAE has as a priority to automate the work of the elaboration of projects, while Simultaneous Engineering is more concerned with ensuring interaction between team members working on the project. The simultaneous approach is supported by techniques such as:

- Deployment of quality function (QFD). The product is specified in a matrix, relating the consumer's wishes (customer attributes) with the quantified engineering characteristics.

- Statistical process control (SPC). Set of techniques and procedures applied to the various phases of the manufacturing process to reduce or eliminate failures in the quality of the final product.
- Failure Analysis (FMEA): Set of activities that identify possible failures of a product or process and their causes, measures that can prevent or reduce the possibility of occurrence and documentation of the process, the result of which will be the recommendation of improvements.
- Design for manufacturing and assembly (DFMA). Through software, the product designer is alerted to the implications of their work in the manufacturing phase.
- Taguchi's methods. They lead to a robust design unaffected by the variables of the production process.
- Just in time (JIT). A production method that attempts to make materials available only when required, greatly reducing inventory costs.
- Benchmarking. Set of procedures through which parameters and specifications of a product are compared with those of the competition, which has the highest performance.
- Computer integrated manufacturing. Use of software that makes it possible to take advantage of computing resources to connect the manufacturing equipment with the database of the project area.

The current trend is to seek greater functional integration. Ideally, simultaneous engineering seeks to apply the tools that allow all members of the project team to have shared access to its updated information, in such a way that they can be stored and processed transparently. The factory of the future must be, in its ideal conception, free of organizational and geographical barriers, with some limitations imposed by the dynamics of business. Fortunately, Simultaneous Engineering lends itself to a gradual introduction and is as useful on small projects as it is on large ones. Therefore, small and large companies can select the items they need. On the other hand, smaller companies are likely to use some typical concurrent engineering elements in isolation, simply because employees often have to take on more than one responsibility. Some sectors have been interested in this philosophy, such is the case of the metal-mechanical sector, which has directed efforts to

the knowledge of the subject. Every day more companies will take into consideration the simultaneous way of working, especially in the field of automated production.

In short, simultaneous engineering allows companies to increase productivity in the face of a significant reduction in delivery times and costs, avoiding product rectifications, as well as integration between different departments or areas and improvements in their communication. In addition, it allows to improve the quality of the product, since from its very conception and design the characteristics and conditions in which it will be developed are foreseen, in such a way that the product is conditioned to the client's specifications from the beginning of the life cycle of the product. For this reason, Concurrent Engineering brings excellent results that allow companies that apply it to achieve true competitive advantages.

It must be borne in mind that simultaneous engineering must be adapted to the characteristics of each organization, in order to create an environment and a structure for its efficient application.

1.3 Analysis of robotic systems

1.3.1 Characteristic of the manipulator's universe

In Robotics Industries Association (RIA) an industrial robot – is a reprogrammable multifunctional manipulator designed to move materials, pieces, tools or special devices, by means of variable movements programmed for the execution of a variety of tasks.

The planning process of an industrial process for chain production, or any type of activity that has a certain function in an automatic system, can be considered as an automatic action carried out by an industrial robotic manipulator. Likewise, we must consider that each action within an automated production chain has a series of conditions that the robot manipulator must meet.

Therefore, the robots that are used in the industrial processes are analyzed for the task that they must perform according to the following classification criteria:

- Degree of freedom: It is the number of axes that the robot has to carry out its movements. Two axes are required to reach any point on a plane, three axes are required to reach any point in the space of a bounded system. To fully control the orientation of the arm end (i.e. the wrist), three more axes (yaw, pitch, and roll) are

required. Some designs (for example, the SCARA robot) exchange limitations on the possibilities of movement in cost, speed and precision.

- Degrees of freedom: it is the same classification as the number of axes.
- Work envelope: the region of space a robot can reach.
- Kinematics: the actual arrangement of the robot's rigid limbs and joints, which determines the possible movements of the robot. Robot kinematics classes are: Articulated, Cartesian, Parallel, and SCARA.
- The load capacity or useful load: it is weight that a robot can lift.
- Speed: the speed with which the robot can position the end of its arm. This can be defined in terms of the angular or linear speed of each axis, or a compound speed, that is, the speed of the end of the arm when all the axes move.
- Acceleration: is the capacity with which an axis can be accelerated. This is a limiting factor, since a robot cannot reach its maximum specified speed for short distance movements or a complex path that requires frequent changes of direction.
- Accuracy: is the ability of a robot to reach a certain exact position. When the absolute position of the robot is measured and compared to the defined position, the error is a precision measurement. Accuracy can be improved with external detection, for example, an IR or vision system. Accuracy may vary with speed and position within the work envelope and with the payload.
- Repeatability: ability of the robot to return to a programmed position. It is not considered the same as precision. It may be that when the robot adopts a certain position x, y, z it only achieves a precision of 1 mm from that position. This would be the accuracy that can be improved by calibration. If that position is programmed into the controller memory and each time it is sent there it returns 0.1mm from the programmed position then the repeatability will be within 0.1mm.
- Motion Control: for some applications, such as “pick and place” mounting, the robot only needs to repeatedly return a limited number of times to predefined positions. For more sophisticated applications, such as welding and finishing (spray painting), movement must be continuously controlled to follow a path in space, with controlled speed and orientation.

- Power supply: some robots use electric motors, others use hydraulic actuators. The former are faster, and the latter are stronger and more suitable in applications such as spray painting, where a spark could cause an explosion, however, under the internal pressurizing air of the arm it can prevent the entry of flammable vapors, thus like other pollutants.
- Transmission: some robots use electric motors on the joints through gears and others use the motor for direct articulation (direct transmission). The small robot arms are capable of working at high speed, under direct current motor torques, which generally require a complex gear system. In these cases, harmonic transmission is often used to avoid gear mismatches.
- Conformity: This is the measure of the angle or distance that a robot axis moves when a force is applied. Compliance when a robot goes into a position carrying its maximum payload will be in a slightly lower position than it is carrying no load. Compliance may also be responsible for mismatch carrying heavy loads in which case the acceleration would have to be reduced.

1.3.2 Physical characteristics of the process

The industrial process to be performed is a “pick and place” handling system located in an environment of an industrial production plant.

The manipulator will aim to move parts and materials from one location to another within its range. By means of automatic accessory systems (such as conveyor belts and other manipulator robots) the robot will be provided with parts so that it can manipulate, capturing them from their initial position and depositing them in the desired final position. The pieces that will take part in the process will be defined on supports, always maintaining the same origin position, and thanks to the programming of the manipulator robot, they will always maintain the same destination positions.

As for the manipulated objects, they will be pieces and tools whose mass n will exceed 20Kg, and the distance between supports will not exceed 2.5 meters in length. In order to manipulate the mentioned pieces it will be necessary to equip the robotic arm with an EOAT tool (End of Arm Tool) capable of picking up these pieces in order to transport

them from their original position to their destination position. For this, a clamp with a prehensile capacity will be used with the capacity to hold a mass of 20 kg, which is the maximum mass to be handled.

Taking these characteristics into account by calculating the "payload" (payload including the weight of the robotic doll, the EOAT and the part) we have to select a robot with a sufficient "payload" and range of range to be able to tackle the stated needs . The chosen robot must have degrees of freedom enough to allow translational movements in the work cell.

We must bear in mind that the pieces must be turned while they are moved from position, which will imply that the freedom of movement of the robot must be high.

1.3.3 Mathematical descriptions of robot's motions

We know that the position of a point in three-dimensional Euclidean space is determined by three quantities, which we call its coordinates, and we say that they are expressed in some reference system, formed by three axes, usually rectilinear.

Hereinafter we will exclusively use straight, orthogonal (that is, with its three perpendicular axes two by two), normalized (that is, the lengths of the basic vectors of each axis are equal) and right-handed (the third axis is product vector of the other two). So we will simply use the term "system" to refer to orthonormal systems.

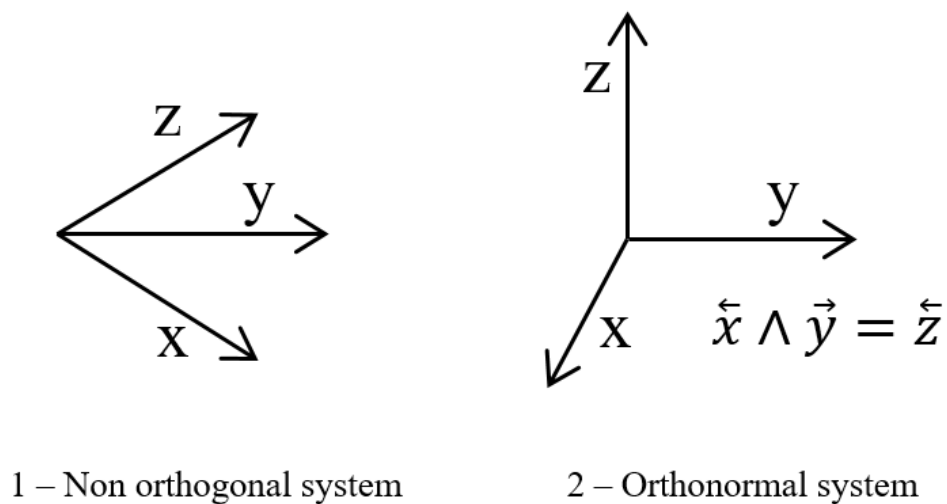


Figure 1.3 Types of referral systems

The coordinates of a point, denoted by $(x; y; z)$, are the projections of that point perpendicular to each axis, or, equivalently, the components of the vector that joins it to the coordinate origin. Instead of using these, it will be more convenient to use the so-called homogeneous coordinates, in the form:

$$\begin{pmatrix} x' \\ y' \\ z' \\ \omega \end{pmatrix} \text{ where } \begin{aligned} x' &= x\omega \\ y' &= y\omega, \\ z' &= z\omega \end{aligned} \quad (1.1)$$

Where ω is an arbitrary quantity, which is usually taken as 1. If, as a result of some calculation, ω were other than 1, the usual coordinates are reconstructed simply by dividing the first three homogeneous coordinates by this fourth. The translation of a point x by a vector v is obviously point x' such that:

$$\vec{x}' = \vec{x} + \vec{v}, \quad (1.2)$$

This expression can be written as:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}, \quad (1.3)$$

But also as the product of a matrix by a homogeneous vector, in the form:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \quad (1.4)$$

This has the advantage that, if:

$$\vec{x}' = H \cdot \vec{x}, \quad (1.5)$$

then:

$$\vec{x} = (H)^{-1} \cdot \vec{x}', \quad (1.6)$$

where the inverse can be calculated, it turns out to be:

$$(H)^{-1} = \begin{pmatrix} 1 & 0 & 0 & -v_x \\ 0 & 1 & 0 & -v_y \\ 0 & 0 & 1 & -v_z \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (1.6)$$

which is consistent with the fact that x is translated by a vector $-v$ with respect to x' .

Regarding rotation about an axis, in the two-dimensional case, it is rotating with respect to a z axis perpendicular to the plane of the figure.

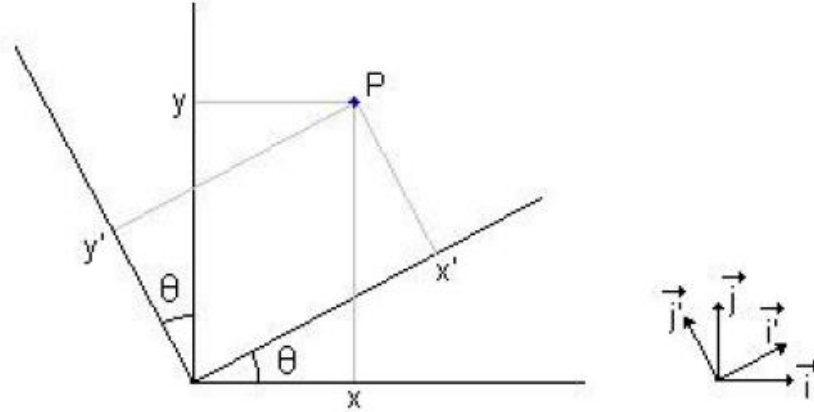


Figure 1.4 Rotation of the reference system with respect to the Z-axis

Calling \vec{i} , \vec{j} the basic vectors of the original system, and \vec{i}' and \vec{j}' those of the rotated system, we have to:

$$\vec{x} = x\vec{i} + y\vec{j} = x'\vec{i}' + y'\vec{j}', \quad (1.7)$$

and

$$\begin{aligned} \vec{i}' &= \cos \theta \vec{i} + \sin \theta \vec{j} \\ \vec{j}' &= -\sin \theta \vec{i} + \cos \theta \vec{j} \end{aligned} \quad (1.8)$$

To consider that:

$$x\vec{i} + y\vec{j} = x'(\cos \theta \vec{i} + \sin \theta \vec{j}) + y'(-\sin \theta \vec{i} + \cos \theta \vec{j}), \quad (1.9)$$

Matching component to component, we write the matrix as:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}, \quad (1.10)$$

If we generalize to three dimensions, since the z coordinate does not vary and the fourth homogeneous coordinate is still 1, we have:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}, \quad (1.11)$$

To find the inverse transformation, it is enough to see that from the point of view of R' , R is rotated one angle $-\theta$. Then we can affirm that:

$$\begin{aligned}
\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} &= \begin{pmatrix} \cos(-\theta) & -\sin(-\theta) & 0 & 0 \\ \sin(-\theta) & \cos(-\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \\
&= \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}.
\end{aligned} \tag{1.12}$$

This operation is simpler than inverting the matrix, although, of course, equivalent. In general, if we had rotated around another of the basic axes, you can see that:

$$Rot(x, \theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{1.13}$$

$$Rot(y, \theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{1.14}$$

$$Rot(z, \theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{1.15}$$

It is worth noting the change of sign in the rotation around the y axis, because, if the axis around which we rotate points at us, the other two form an angle of 90° in the case of xyz, but -90° in the case of y. As many successive transformations (rotations and translations) as desired can be applied to a point. The resulting operation would be given by a matrix that would be the product of the matrices of each operation, applied in the correct order, since the product of matrices is not commutative. The first transformation applied is placed to the right, being expressed as:

$$Y = T_2 R_3 T_1 R_2 R_1 X, \tag{1.16}$$

It means that rotation 1 is applied to point X, followed by rotation 2, followed by translation 1, then rotation 3 and finally translation 2.

Let us now see what the rotation matrix would be with respect to any axis. Let be an axis that passes through the origin defined by a unit vector around which we will rotate an angle θ .

$$\vec{r} = (r_x, r_y, r_z), \quad (1.17)$$

This rotation can be broken down into three rotations on the basic axes, which will be equivalent to:

- Rotate an angle α around x, whereby P will move to position P';
- Rotate an angle $-\beta$ around y, whereby P' will move to position P'';
- Rotate an angle θ around z, which is the requested rotation;
- Rotate an angle β around y, undoing the second rotation;
- Rotate an angle $-\alpha$ around x, undoing the first rotation.

So we have:

$$R_{\vec{r},\theta} = R_{x,-\alpha} R_{y,\beta} R_{z,\theta} R_{y,-\beta} R_{x,\alpha}, \quad (1.18)$$

$$R_{\vec{r},\theta} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha & s\alpha & 0 \\ 0 & -s\alpha & c\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c\beta & 0 & s\beta & 0 \\ 0 & 1 & 0 & 0 \\ -s\beta & 0 & c\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c\theta & -s\theta & 0 & 0 \\ s\theta & c\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c\beta & 0 & -s\beta & 0 \\ 0 & 1 & 0 & 0 \\ s\beta & 0 & c\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha & -s\alpha & 0 \\ 0 & s\alpha & c\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (1.19)$$

The treated example is summarized in the following figure:

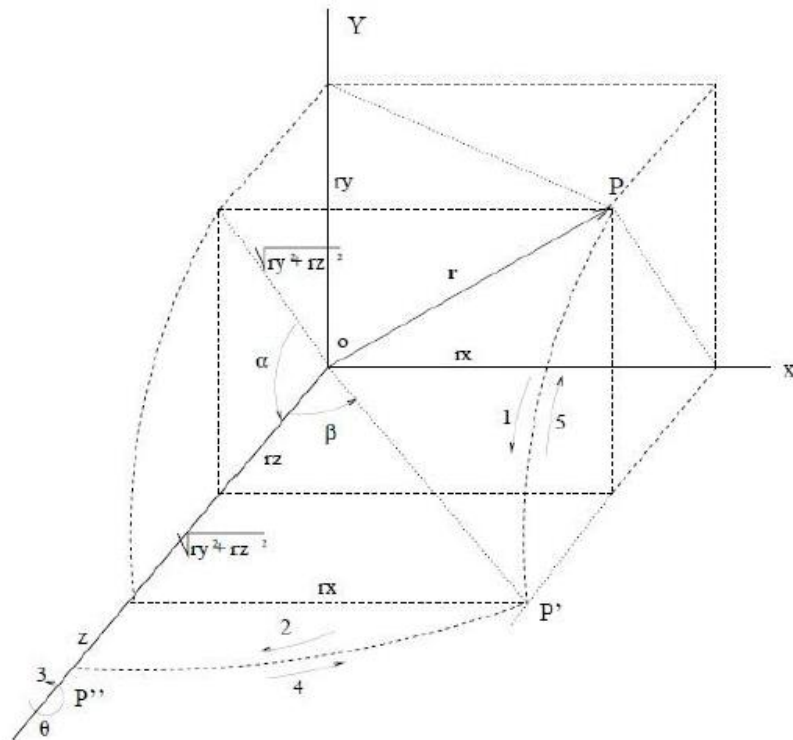


Figure 1.5 Rotation about an arbitrary axis from basic axes

We can check that it is met:

$$s\alpha = \frac{r_y}{\sqrt{r_y^2 + r_z^2}}, \quad (1.20)$$

$$c\alpha = \frac{r_z}{\sqrt{r_y^2 + r_z^2}}, \quad (1.21)$$

$$s\beta = \frac{r_x}{|r|} = r_x, \quad (1.22)$$

$$c\beta = \frac{\sqrt{r_y^2 + r_z^2}}{|r|} = \sqrt{r_y^2 + r_z^2}. \quad (1.23)$$

By multiplying it all remains:

$$R_{\vec{r},\theta} = \begin{pmatrix} r_x^2 v\theta + c\theta & r_x r_y v\theta - r_z s\theta & r_x r_z v\theta + r_y s\theta & 0 \\ r_x r_y v\theta + r_z s\theta & r_y^2 v\theta + c\theta & r_z r_y v\theta - r_x s\theta & 0 \\ r_x r_z v\theta - r_y s\theta & r_z r_y v\theta + r_x s\theta & r_x^2 v\theta + c\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (1.24)$$

$$s\theta = \sin\theta, c\theta = \cos\theta, v\theta = 1 - \cos\theta, \quad (1.25)$$

Obviously, if $r_x = r_y = 0$ is done, the rotation around the z axis is obtained. In the same way we can obtain the rotation of the other two axes applying this same method. We must emphasize that any consecutive sequence of transformations can be specified in two ways:

1. Performing the rotation that takes one system to the other, around the initial axes;
2. Performing the rotation that takes one system to the other around one of the rotated axes, that is, those that resulted from the last transformation.

In the first case, the matrix that describes this transformation must be pre-multiplied by the one that described the transformations carried out so far, obtaining the total transformation.

In the second case, the matrix that describes this transformation must be post-multiplied by which she described the transformations carried out so far, obtaining the total transformation.

1.4 Choice justification of Roboguide FANUC

The current market in the field of industrial robotics offers various possibilities in terms of the choice of a simulation tool, generally associated with a specific brand of robots.

The vast majority of commercial simulation tools are developed by the manipulator robot brand with which it intends to work, taking into account that robot manufacturers provide simulation software adapted to their controller models. This fact has some logic since the manufacturer is the provider of the controller and is the one that best knows all the functionalities that your device offers when transferring them to software of this nature.

Given the great advantages offered by the simulation system of a process in an "offline" way, manufacturers have expanded their commercial offer by offering these tools to their customers, in order to improve the programming system of their manipulators and also to expand their product offer. with this powerful software.

After reviewing the different manufacturers of current industrial robots (ABB, FANUC, KIKA) we have been able to observe that the Roboguide tool from FANUC is one of the most complete. Given the relevance of the FANUC brand in the industrial environment, we have opted for its tool, taking into account the real application of the industrial process to be simulated in the current industrial environment. Roboguide is a powerful "offline" programming software, which allows us to supervise the programmed process at all times, avoiding programming and testing costs in a real environment.

The interface that the application offers allows us to design the environment of the cell in a very complete way and with some ease, as well as allowing us to configure each of the objects that will intervene in the production process. The final objective that allows us to achieve is the verification of the complete operation of the robot. We can easily detect collisions or interferences produced between the robot and the other objects in the cell. It allows us to import CAD files to include existing parts in the virtual environment. It also includes an integrated virtual "Teach Pendant" that simulates real consoles.

One of the outstanding features of the application is that it is capable of automatically creating reference programs with the real robot in order to calibrate the robotic system of control, whenever the connection with the real robot is available. General control system is shown in the figure below.

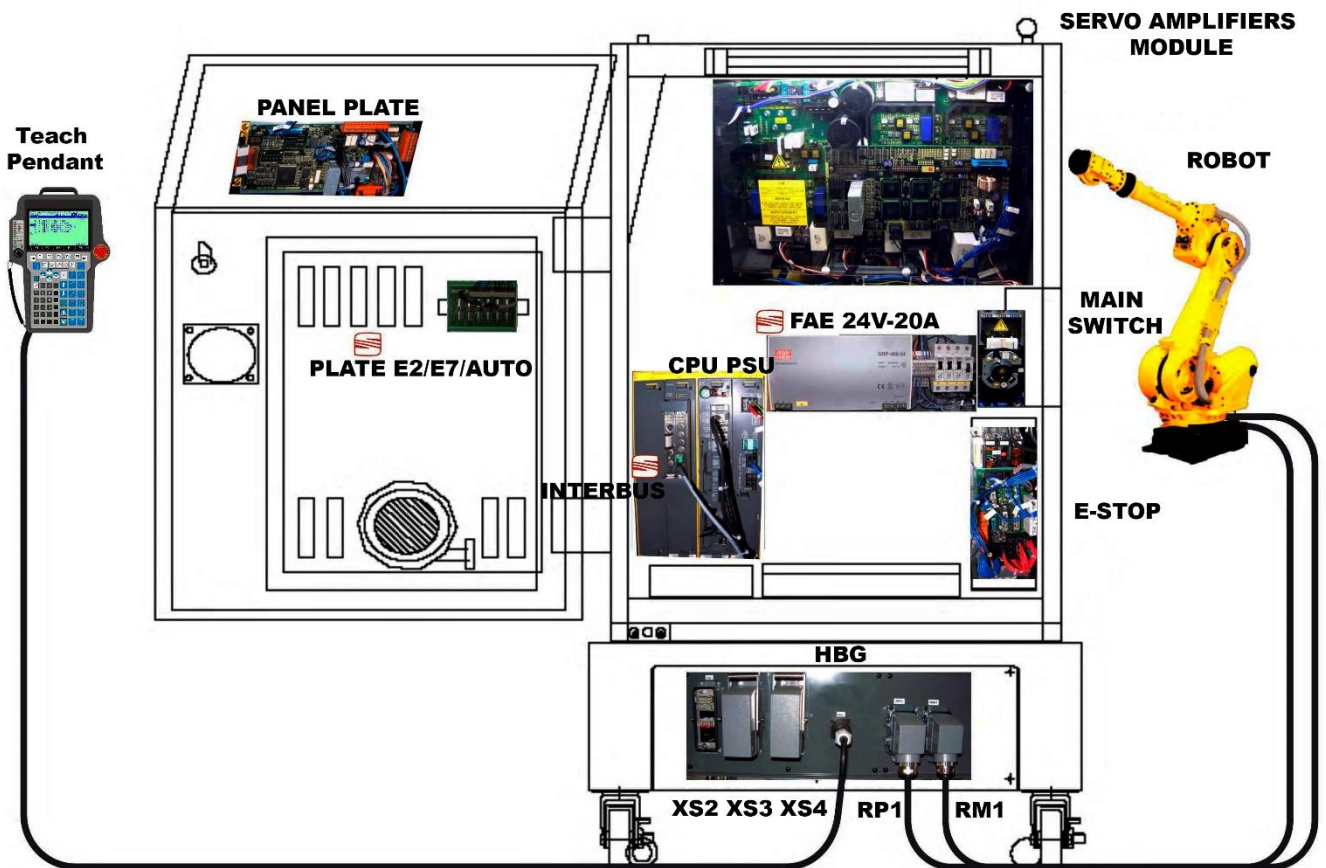


Figure 1.5 System of control for the robotic system Fanuc

Roboguide's HandlingPRO module allows us to carry out a “Pick and place” handling system with the elements most frequently used for this type of process, which is perfectly adapted to the process that needs to be programmed and simulated.

1.5 Research task statement

Based on the fact that the main objective of the research work is the development of the Fanuc industrial robot control program by constructing a simulation model of a robotic production workcell in the Roboguide software environment, we can define our main research tasks:

- Calculate the payload, which includes the weight of the robot arm, EOAT, and part.
- Choose a robot in accordance with the given criteria, with sufficient “payload” and range to be able to fulfill the stated needs. The selected robot must have a sufficient degree of freedom to allow translational movements in the working area.

- Estimate the cycle time and volume of machined parts.
- Make the necessary changes to the simulation model to ensure the specified criteria.
- Add a second robot to the simulation model.
- Adjust the trajectories of the robots so as to avoid a collision.
- Define a general methodology for building control programs for Fanuc robots.

Given the following system requirements:

- Ensuring minimum runtime.
- Ensuring the maximum number of machined parts.
- The ability to implement control programs in the controller of a real Fanuc robot.
- Creating a simulation model in accordance with a given layout of elements.
- Ensuring synchronization between two robots.

1.6 Conclusions

The state of Industrial robotics today, the importance of Simulation in industrial robotics was analyzed and the concept of "offline" programming was identified. In Analysis of robotic systems, we identified the Characteristic of the manipulator's universe and Mathematical descriptions of robot's motions, which gave us the opportunity to more comprehensively understand the concept of a robotic system as a whole.

After reviewing the different manufacturers of current industrial robots we have been able to observe that the Roboguide tool from FANUC is one of the most complete. Given the relevance of the FANUC brand in the industrial environment, we have opted for its tool, taking into account the real application of the industrial process to be simulated in the current industrial environment. Roboguide is a powerful "offline" programming software, which allows us to supervise the programmed process at all times, avoiding programming and testing costs in a real environment.

Roboguide's HandlingPRO module allows to carry out a "Pick and place" handling system with the elements most frequently used for this type of process, which is perfectly adapted to the process that needs to be programmed and simulated. In this way, we were able to identify the formulation of the research problem in this work.

2 FANUC ROBOT PROGRAMMING

2.1 TPE programming

The programming of a robot could be defined as the execution of a set of commands and orders in a sequential way so that, step by step, they will execute each one of the orders that will complete the different processes of our system. We have two possible methods to design the manipulator program using the simulation tool:

“Roboguide Virtual Teach Pendant Editor” - the application offers us the possibility of programming the robot with a virtual "teach pendant", capable of working with all the manipulator functions administered by the controller. The virtual teach pendant has the same software as a real teach pendant physically connected to the manipulator. It is used to create complex instructions with the functionalities of TP programs.

“Roboguide Simulation Programs and editor” - the application also offers us an intuitive and very complete editor, which allows programming of programs through a graphical environment. The great advantages of this editor are that it allows creating robotic programs in a more comfortable way than in a TPE terminal and offers the great advantage of evaluating the cycles of work cell processes “offline” using the animations generated by the simulation program, controlling the various factors that can interfere with our simulation without having to suffer them in a real environment (incorrect or impossible trajectories, collisions, points out of reach).

For the robotic arm to execute the given set of commands in accordance with its working environment, will be written the program using the Virtual Teach Pendant robot's virtual control panel, as this happens on real production lines. We will also carry out the implementation process using the simulation editor offered by the software.

We will start with a brief description of the use of this interface, main functions, menus and keys. There are different models depends on the model of the robot and the controller you need. The real Teach Pendant FANUC robot control panel looks like:



Figure 2.1 Teach Pendant FANUC robot control panel

The functionality of each virtual Teach Pendant’s key is the same as in real control panel shown in the diagram. There are differences between different controller models, but the difference is not significant. Teach Pendant makes possible to control HandlingTool. It has keys for direct control over the motions of the HandlingTool, displaying the software menu, selecting various options from different menus, prompts in creating TP programs, and perform certain functions of palletizing.

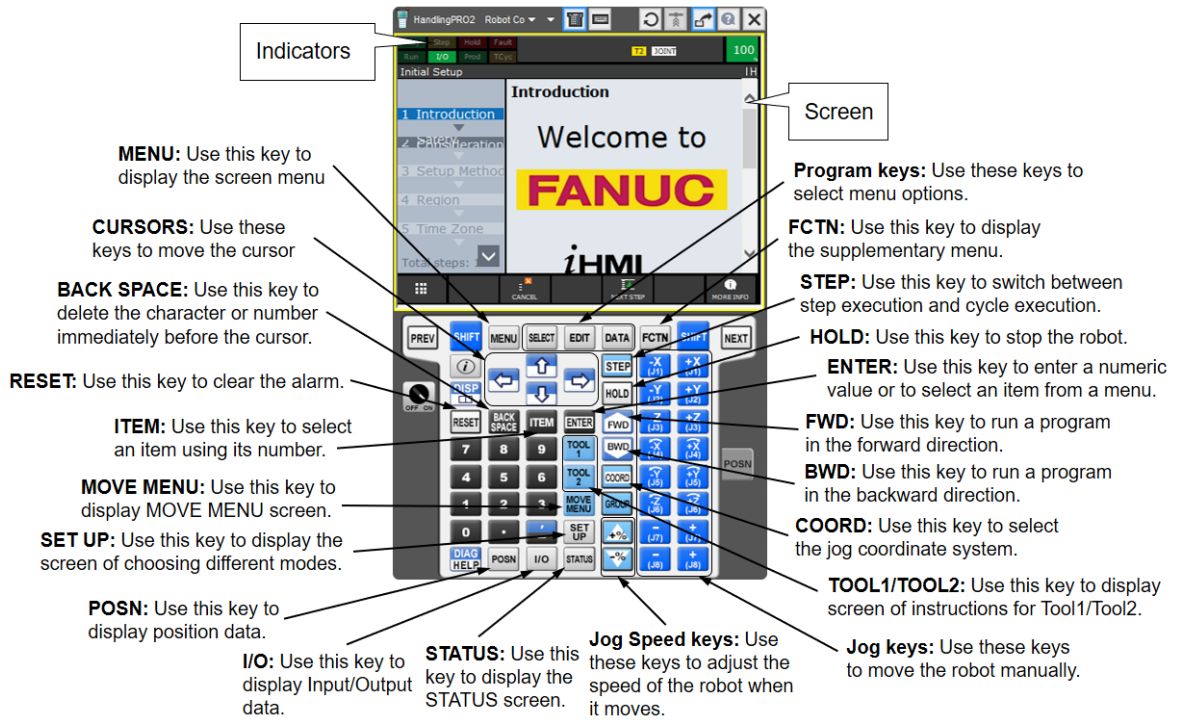


Figure 2.2 Functionality of each virtual Teach Pendant’s key

The following table specifies each of the states according to the upper left LED panel:

Table 2.1 – States of the “led” indicators of the “Teach Pendant” console



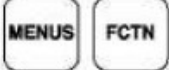







FAULT	indicates that an alarm exists
HOLD	indicates that the HOLD button is pressed or that a HOLD signal is received
STEP	indicates that you are under step-by-step operation mode
BUSY	lights up while the robot is working. It also lights up when the CPU performs jobs related to the movement of the robot (copy, paste, print ...)
RUNNING	indicates that the program is running







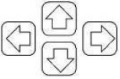

GUN ENBL	indicates that the gripper can be opened or closed
WELD ENBL	indicates that spot welding can be performed
I/O ENBL	indicates that the I/O signals are activated
WELD ENBL	when illuminated, indicates arc welding is activated
ARC ESTAB	illuminates when arc welding is being used
DRY RUN	when DRY RUN is lit, it indicates that the operating mode is selected using dry start
JOINT	lights when JOINT movement is selected as the manual movement coordinate system
XYZ	lights when Cartesian motion is selected as the manual motion coordinate system
TOOL	lights when tool movement is selected as the manual movement coordinate system

The LEDs that we will take into account for our simulation process will be those of the “handling tool” since the LEDs that indicate the states of some welding process will not be used (those marked in gray).

On the Teach Pendant keyboard we can find the buttons that we will use to operate the manipulator programming software. The main functionalities are specified in the following Table 2.2:

Table 2.2 – Overview of the Teach Pendant Console Buttons

KEY	FUNCTION
	The function keys (F) are used to select a function menu on the last line of the display.
	The NEXT key is used to enable more function keys on the next page.
	The MENU key to display the screen menu. The FCTN key to display the function menu.
	The SELECT key to display the program selection screen. The EDIT key to display the program edit screen. The DATA key to display the program data screen.
	The MAN FCTNS key displays the manual operation screen.
	The STATUS key displays the current position screen. The I/O key displays the I/O screen. The POSN key displays the current position screen.
	The SHIFT keys are used to enable robot motion, program position data, and start a program.
	The movement keys are effective as long as a Shift key is held down. They are used for motion enablement.
	The COORD key selects a manual move coordinate system. Each time the COORD key is pressed, it selects the next type of movement in the order: JOINT, JGFRM, World frame, TOOL, USER. When this key is pressed while holding down a Shift key, a move menu appears for changing the coordinate system.
	The speed variation key. Each time it is pressed it varies in the order: VFINE, FINE, 1%, 5%, 50%, 100%. (5% change of the amount for 5% or less and 5% change of the amount for 5% or more.

	<p>The FWD key or the BWD key (+ SHIFT key) starts a program. When the shift key is released during regeneration, the program is interrupted.</p>
	<p>The HOLD key causes a program to be interrupted.</p>
	<p>The STEP key selects step by step or performs continuous cycle operation.</p>
	<p>The PREV key recalls the most recent state. In some cases, the key cannot immediately return to the previous state.</p>
	<p>The ENTER key enters, validates and selects a number or a menu.</p>
	<p>The BACK SPACE key deletes the character or number immediately before the cursor.</p>
	<p>The arrow keys move the cursor. The cursor is the highlighted area that can be moved on the teach pendant screen. This zone becomes the operation object (input or change of value or contents) of the key on the programming console.</p>
	<p>The ITEM key moves the cursor to a line whose number is specified.</p>

The last part that we can distinguish in the "Teach Pendant" is the "display", which shows us the menus and the different screens available in the manipulator programming software.

The following figure shows an example "SAMPLE1" of the "display" indicating the main characteristics in an arm positioning program:

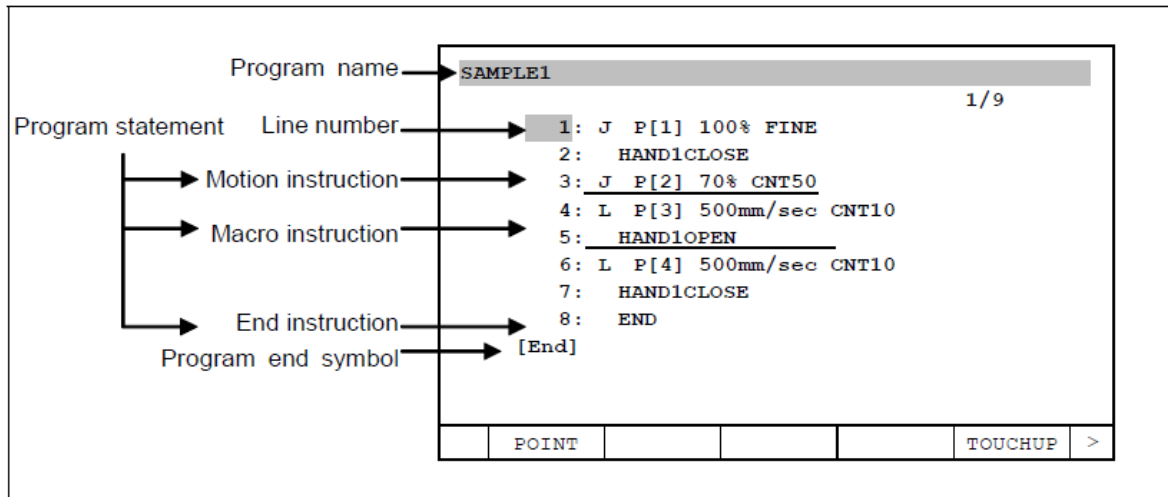


Figure 2.3 Example of program in a display TPE

Knowing the main characteristics of the TPE programming device, we can begin to use it to be able to carry out our program, thus composing a routine for the robot and the process in general.

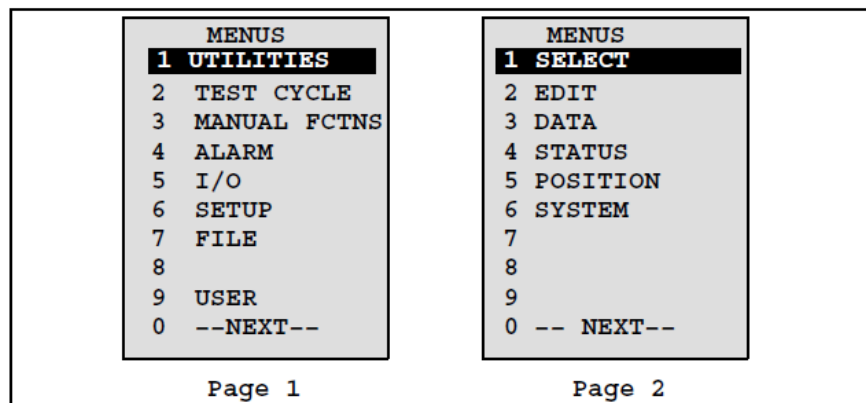


Figure 2.4 TPE device main menus

In the Figure 1.9 we can see the main menu of our TPE device. We will explain the main sections in a general way and then specify those used in our programming process using the Roboguide TPE device:

- UTILITIES The utility screen is used to display the tracks.
- TEST CYCLE The test cycle screen is used to specify the data for the test operation.
- MANUAL FCTNS The manual operation screen is used to execute macro instructions.

- ALARM The alarm history screen shows the history and details of the alarms.
- I / O The I / O screen for viewing, forcing, simulating, and configuring input and output signals.
- SETUP The setup screen is used to set the system.
- FILE The file screen is used to read or store files.
- USER The user screen shows user messages.
- SELECT The program selection screen is used to list or create programs.
- EDIT The program edit screen is used to correct and run the program.
- DATA The data screen shows the values in registers, position registers and other variables.
- STATUS The status screen shows the status of the system.
- POSITION The current position screen shows the current position of the robot.
- SYSTEM The system screen is used to set system variables and mastering.

Program detail information names a program and defines the attributes of the program. Program detail information consists from attribute-related information items (such as a creation date, modification date, a copy source file name, presence/absence of position data, and program data size) and information items (related to an execution environment such as a program name, subtype, comment, group mask, write protection, interruption disable and stack size).

Program detail		1/7
Creation Date:	16-Jan-1994	
Modification Date:	08-Mar-1994	
Copy Source:		
Positions:	FALSE	Size: 312 Bytes
Program name:		
1	SAMPLE3	
2	Sub Type: [None]	
3	Comment: [SAMPLE PROGRAM 3]	
4	Group Mask: [1,*,*,*,*,*,*,*]	
5	Write protect: [OFF]	
6	Ignore pause: [OFF]	
7	Stack size: [500]	
END	PREV	NEXT

Figure 2.5 Program detail information screen

The program information screen is used to set program detail information. The program information screen is displayed by selecting F2, DETAIL on the program selection screen. After displaying the program selection screen. We can move the cursor to the program that should be edited and press ENTER key. The program edit screen appears:

```

SAMPLE1
1/6
1: J P[1] 100% FINE
2: J P[2] 70% CNT50
3: L P[3] 1000cm/min CNT30
4: L P[4] 500mm/sec FINE
5: J P[5] 100% FINE
[End]
POINT TOUCHUP >

```

Figure 2.6 Program edit screen

To move the cursor, use the arrow keys such as up, down, right, and left. To move quickly through the information, press and hold the SHIFT key and press the down or up arrow keys. To select the line number, press the ITEM key and enter the line number you want to move the cursor.

A motion instruction moves a robot tool to a specified point within the operating area at a specified feed rate and in a specified traveling mode. The items listed below must be specified in a motion instruction. The format of a motion instruction is shown in Figure 1.12.

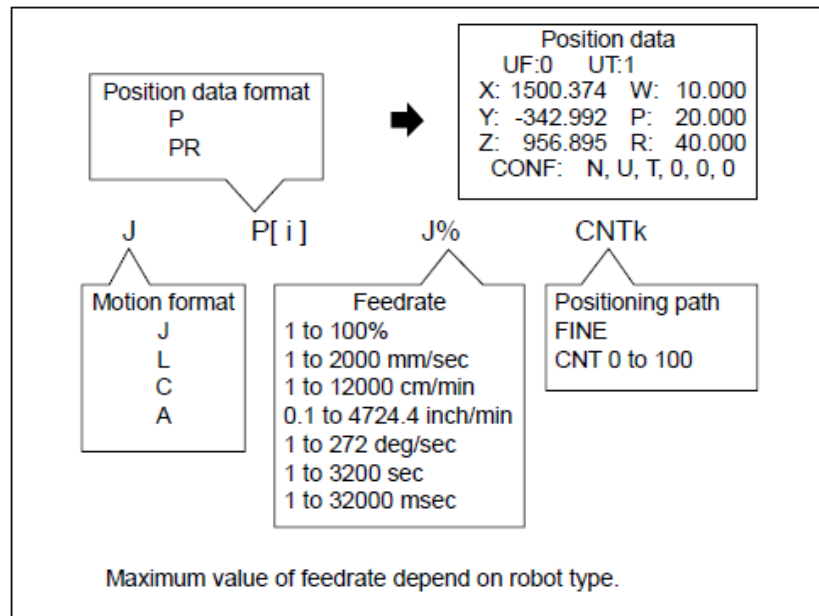


Figure 2.7 Motion instructions

In teaching a motion instruction, a standard motion instruction is selected using function keys. Motion instruction determined by:

- Motion format: Specifies how to control the path of motion to a specified position.
- Position data: Teaches a position to which the robot is to move.
- Feed rate: Specifies the feed rate of the robot.
- Positioning path: Specifies whether to position the robot at a specified point.
- Additional motion instruction: Specifies the execution of an additional instruction while the robot is in motion.

Every time we record a point, it can be represented in degrees and in Cartesian coordinates. In Cartesian coordinates, the recorded coordinates are those of the TCP (Tool Center Point or Tool Central Point), with respect to the origin of the Cartesian coordinate system currently active and previously chosen by the user. (WORLD by default). By default the TCP is located in the center of the outermost axis plate of the robot.

TCP is the source of the tool reference. When a tool reference is created, the TCP moves to the end of the used tool. The tool reference can be oriented according to the attack axis of that tool.

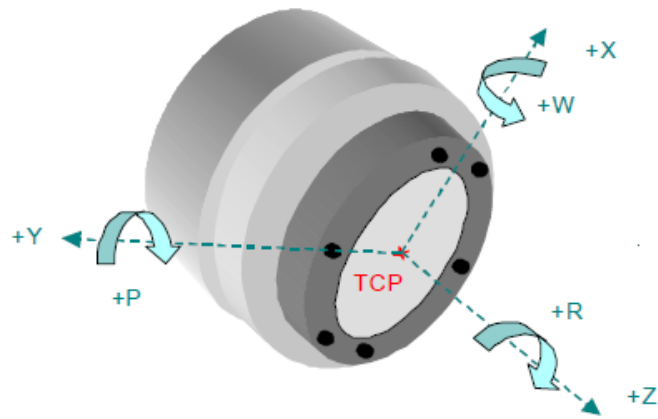


Figure 2.8 Tool Center Point

We distinguish 2 types of tools.

Simple tool. A simple tool is a tool in which the cutting axis is parallel to the Z axis of the tool by default. In this case the orientation of the tool does not change with respect to the default tool; only TCP moves. The 3-point learning method is the one chosen to memorize the tool.

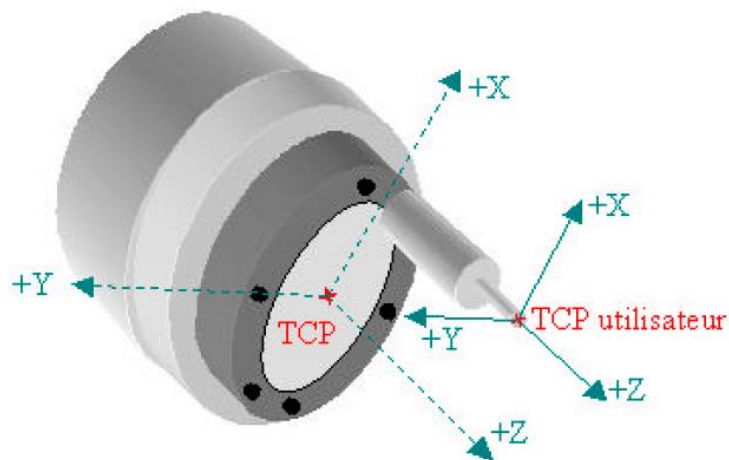


Figure 2.9 Simple tool

Complex tool. A complex tool is a tool in which the cutting axis is not parallel to the Z axis of the tool by default. In this case the TCP is displaced and its orientation is redefined.

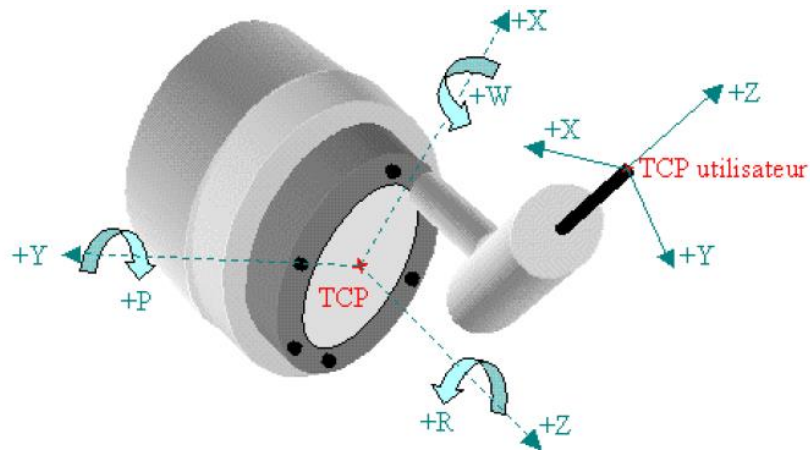


Figure 2.10 Complex tool

To define a tool select MENU - SETUP - F1: [TYPE] - FRAMES - F3: [OTHER] - TOOL - ENTER. The TOOL FRAME SETUP page appears:

SETUP Frames				JOINT 10 %
Tool Frame Setup/ Direct Entry				1/9
	X	Y	Z	Comment
1:	0.0	0.0	0.0	*****
2:	0.0	0.0	0.0	*****
3:	0.0	0.0	0.0	*****
4:	0.0	0.0	0.0	*****
5:	0.0	0.0	0.0	*****
6:	0.0	0.0	0.0	*****
7:	0.0	0.0	0.0	*****
8:	0.0	0.0	0.0	*****
9:	0.0	0.0	0.0	*****
Active TOOL \$MNUTOLNUM[1] = 1				
[TYPE]	DETAIL	[OTHER]	CLEAR	SETIND

Figure 2.11 TOOL FRAME SETUP page

Direct input method of values. In this method, the coordinates and orientation of the tool to be defined must be perfectly known. These coordinates will be entered directly by hand in the following window: F2: [METHOD] - DIRECT ENTRY.

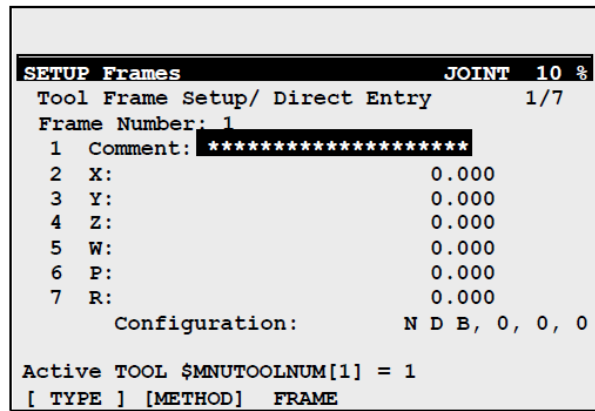
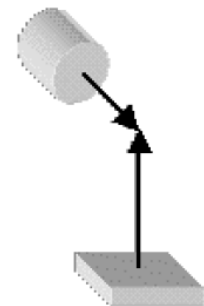
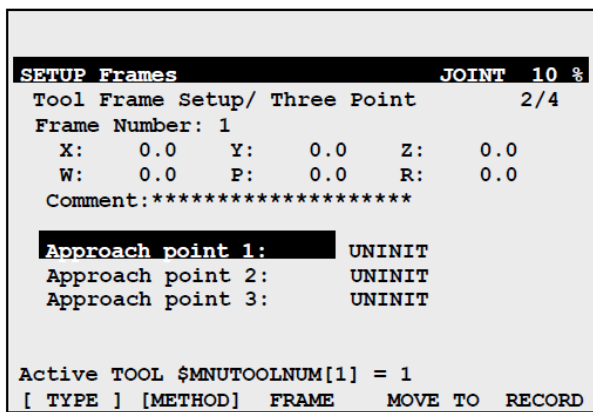


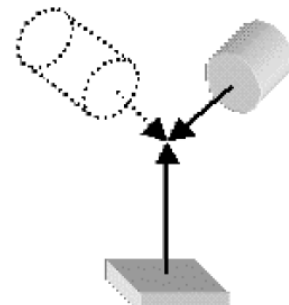
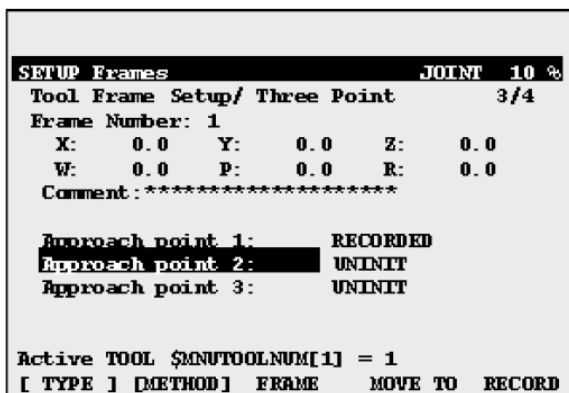
Figure 2.12 Window of direct input method of values

3-point method. The purpose of this method is to move the TCP to the end of the tool used. For this we have to mark the same point with 3 different orientations and memorize those positions.



SHIFT + F5: RECORD

Figure 2.13 Step 1



SHIFT + F5: RECORD

Figure 2.14 Step 2

```

SETUP Frames                                JOINT 10 %
Tool Frame Setup/ Three Point                 4/4
Frame Number: 1
X:   0.0   Y:   0.0   Z:   0.0
W:   0.0   P:   0.0   R:   0.0
Comment:*****

Approach point 1:      RECORDED
Approach point 2:      RECORDED
Approach point 3:      UNINIT

Active TOOL $MNUTOLNUM[1] = 1
[ TYPE ] [METHOD] FRAME MOVE TO RECORD

```

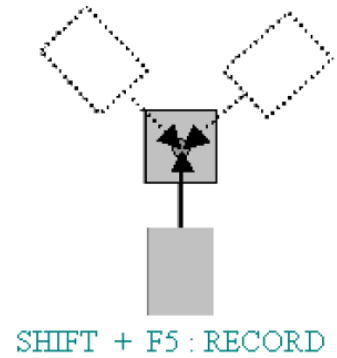


Figure 2.15 Step 3

When the three points have been memorized, the x, y, z coordinates of the new TCP are displayed in the upper part of the window. These coordinates are given with respect to the original factory TCP. The sense of the Z coordinate of the TCP created by the 3P method is the same as the original TCP of the robot. Final state of the window shown in the Figure 1.21:

```

SETUP Frames                                JOINT 10 %
Tool Frame Setup/ Three Point                 1/4
Frame Number: 4
X:  28.1   Y:  53.3   Z: 140.6
W:   0.0   P:   0.0   R:   0.0
Comment:*****

Approach point 1:      USED
Approach point 2:      USED
Approach point 3:      USED

Active TOOL $MNUTOLNUM[1] = 1
[ TYPE ] [METHOD] FRAME

```

Figure 2.16 Final state screen using the 3-point method

The purpose of this method is to move the original TCP of the robot to a specific point on the tool used and to reorient the tool based on that point. The sense of the Z coordinate of the TCP created by the 6P method is different from that of the original TCP of the robot. In this case it is imposed by the user.

Steps 1, 2 and 3 are – the first three steps are identical to the first three steps as the three-point method. The TCP is defined and now we must reorient the tool and memorize three additional points.

Step 4 – Orient Origine Point. To memorize the orientation origin point, the OZ axis of the tool must be positioned vertically, as in the following figure:

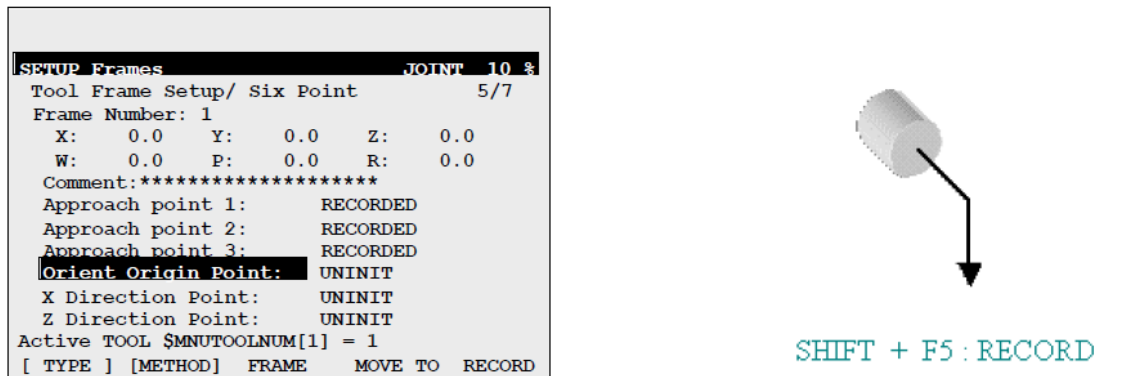


Figure 2.17 Orient Origine Point definition of the tool

Step 5 – X Direction Point. We will now define the orientation and direction of the X axis. For this step and the next, it is more practical to move in the WORLD system, in order to ensure that we move the tool's OZ axis horizontally.

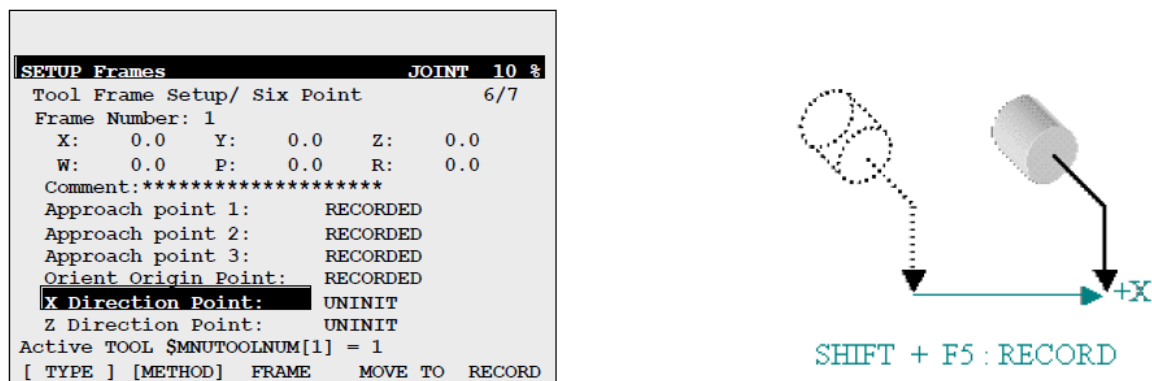


Figure 2.18 Definition of X Direction Point of the tool

Step 6 – Z Direction Point. To give the direction in Z, it is necessary to re-position on the point of origin of the orientation. To do this, place the cursor on the «Orient Origine Point» line and then press SHIFT + F4: MOVE_TO. The robot will reposition itself on the point memorized in step 4. To define the direction and the sense of the Z axis.

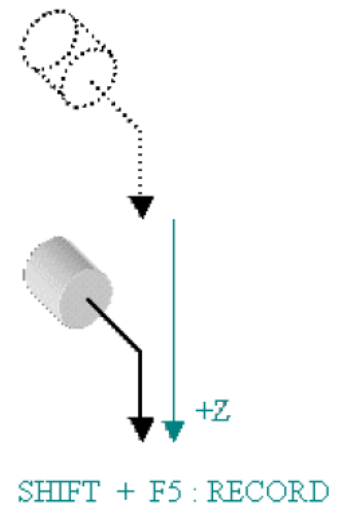
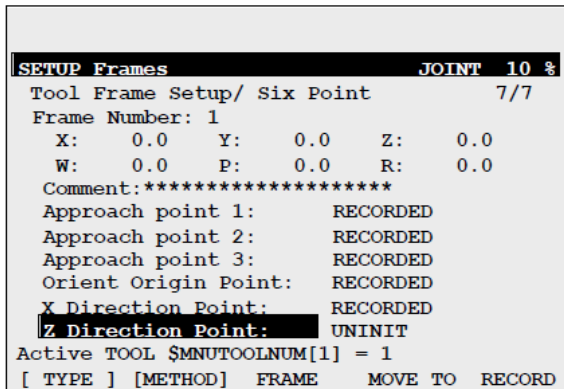


Figure 2.19 Definition of Z Direction Point of the tool

When the 6 points are memorized, the x, y, z coordinates of the new TCP and the w, p and r orientations of the new tool are displayed in the upper part of the window.

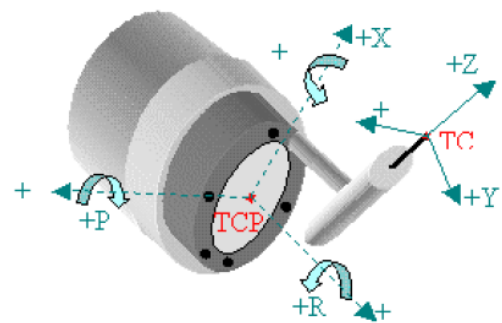
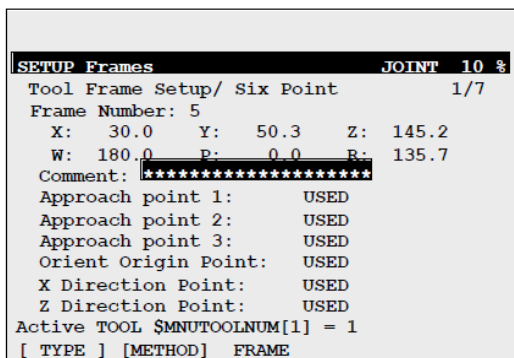


Figure 2.20 Tool origin definition screen by 6-point method

Direct input method of values. In this method, the coordinates and orientation of the USER reference with respect to the WORLD are perfectly known. The coordinates will be entered by hand in the next window:

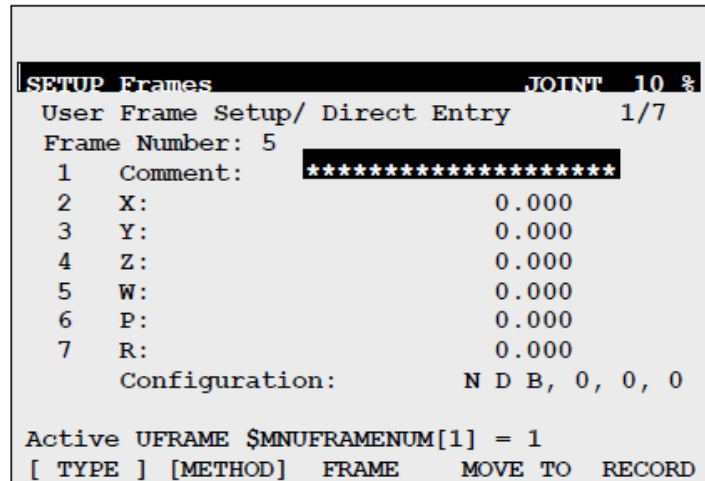


Figure 2.21 Tool origin definition screen by direct input method

User-defined reference system. A USER FRAME user reference system is a three-dimensional and Cartesian reference system on which all the positions of a finished TP program are memorized. The TCP moves and reorients based on that system whenever we move the robot in USER mode. If no user reference system is defined, by default, the positions will refer to the WORLD coordinate system.

The figure below represents a custom reference system achieving a user-modified work environment:

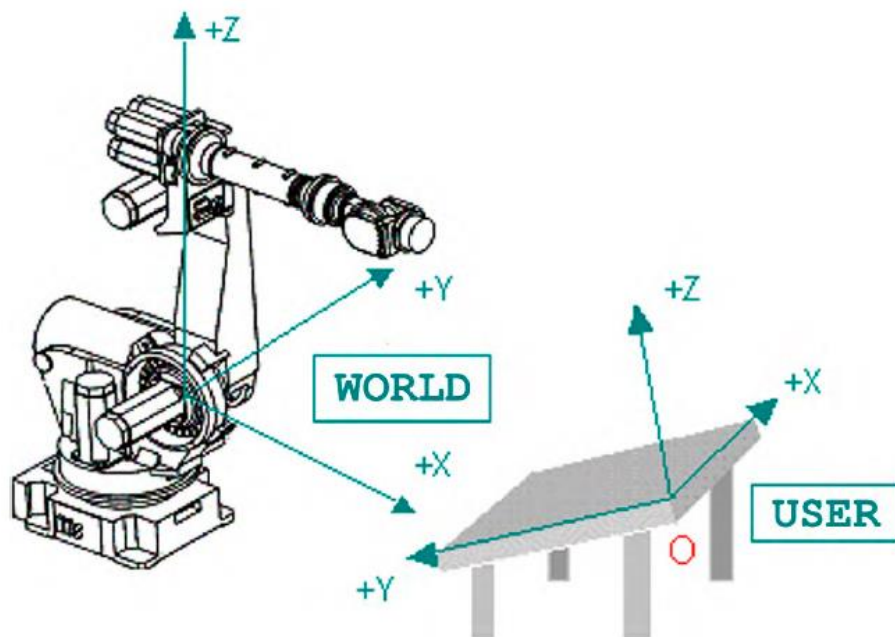


Figure 2.22 User-defined reference system (UFRAME)

Configuration methods. In order to define a user reference system, we can do it using the following instructions: MENU - SETUP - F1: [TYPE] - FRAMES - F3: [OTHER] - USER – ENTER.

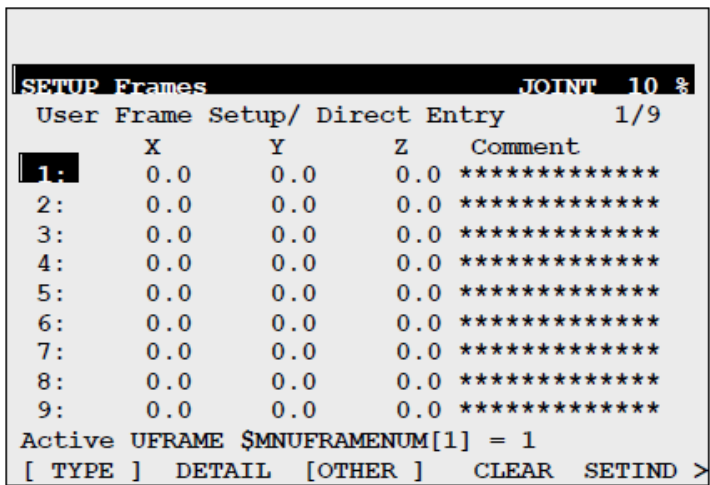


Figure 2.23 UFRAME configuration methods screen 1

We will access the previous screen and we can configure the specific tool with the cursor: F2:DETAIL - F2:METHOD.

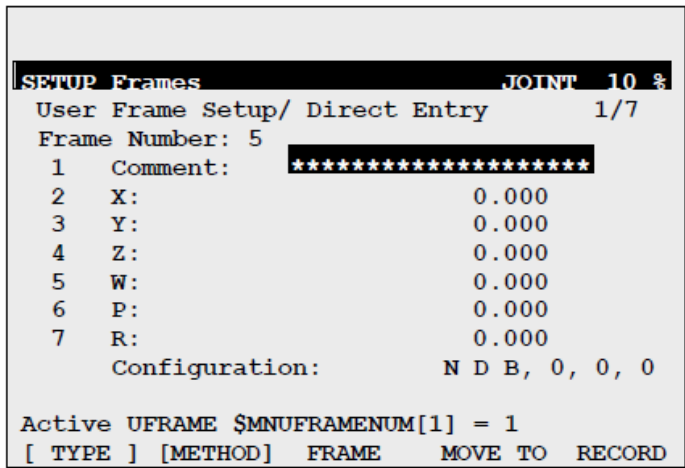


Figure 2.24 UFRAME configuration methods screen 2

We will now choose one of the three learning methods to define the user reference. The origin of the reference system will move to the desired location and the position and orientation following the three defined directions. 3 Points we will select from the menu: F2: METHOD - THREE POINT. With this method we will define two intersecting lines that determine a plane, with a fixed origin at the crossing point and Z perpendicular to the

defined plane. Orient Origin Point - we will memorize the origin point to determine the custom reference per user.

With the cursor we will move and determine each of the values to define.

```

SETUP Frames                                JOINT 10 #
User Frame Setup/ Three Point                2/4
Frame Number: 1

X:  0.0  Y:  0.0  Z:  0.0
W:  0.0  P:  0.0  R:  0.0

Comment:*****
Orient Origin Point:  UNINIT
X Direction Point:      UNINIT
Y Direction Point:      UNINIT

Active UFRAME $MNUFRAMENUM[1] = 1
[ TYPE ] [METHOD] FRAME  MOVE TO RECORD
    
```

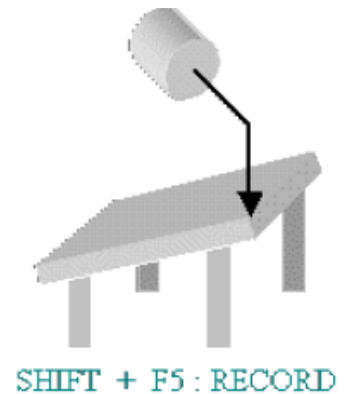


Figure 2.25 Orient Origin Point configuration

We then indicate the direction and sense of the X axis by memorizing a point that belongs to + X.

```

SETUP Frames                                JOINT 10 #
User Frame Setup/ Three Point                3/4
Frame Number: 1

X:  0.0  Y:  0.0  Z:  0.0
W:  0.0  P:  0.0  R:  0.0

Comment:*****
Orient Origin Point:      RECORDED
X Direction Point:    UNINIT
Y Direction Point:      UNINIT

Active UFRAME $MNUFRAMENUM[1] = 1
[ TYPE ] [METHOD] FRAME  MOVE TO RECORD
    
```

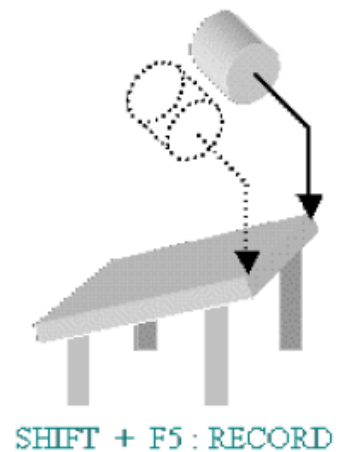


Figure 2.26 X Direction Point configuration

This last step determines the orientation and direction of the Y axis and finally, by calculation, that of the Z axis. The memorized point must be a point that belongs to the Y axis. Final state of the window:


```

SETUP Frames JOINT 10 3
User Frame Setup/ Three Point 1/4
Frame Number: 1

X: 1474.6 Y: 425.0 Z: -8.6
W: -0.9 P: 0.5 R: 89.9

Comment:*****
Orient Origin Point: USED
X Direction Point: USED
Y Direction Point: USED

Active UFRAME $MNUFRAMENUM[1] = 1
[ TYPE ] [METHOD] FRAME

```

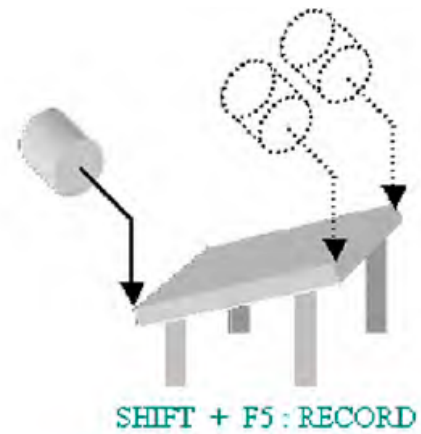


Figure 2.27 Y and Z Direction Points configuration

When the 3 points are memorized, the x, y, z coordinates of the origin and the w, p, r orientations of the axes of the new reference system are displayed in the upper part of the window. These coordinates are given with respect to WORLD (Figure 1.27).

Routine programming instructions with registers and position registers. The variables available to use are: registers (real (32 bits) or integer) and position records (points in joint coordinates, points in Cartesian coordinates or matrices). These are global variables (all programs have access to all registers and position registers).

The registers has a maximum of 256 (configurable) and they allow to be commented with a name. Records can be defined in the following ways:

- Direct: $R [1] = 2$ -> the value is stored directly in $R [1]$;
- Indirect: $R [R [1]] = 5$ -> the affected register depends on the value contained in $R [1]$. If $R [1] = n$, therefore the value 5 is stored in $R [n]$.

In a register it is possible to store the result of an arithmetic operation: $R [n] = [value] [operator] [value]$. The [operator] can be:

- sum (+);
- subtraction (-);
- multiplication (*);
- division (/);
- integer division (DIV);
- rest (MOD).

The [value] can be:

- constant;
- analog input-output AI [n] / AO [n];
- digital input-output DI [n] / DO [n];
- group input-output GI [n] / GO [n];
- robot input-output RI [n] / RO [n];
- register R [n];
- element of a position register PR [i, j].

To insert into a program -> F1: [INST] -> Registers. To display the list of registers and their content -> DATA -> F1: [TYPE] -> Registers.

The position records store a point. Position records can be defined in the following ways:

- Direct: $PR [1] = P [1]$ -> the point is stored directly in PR [1];
- Indirect: $PR [R [1]] = P [3]$ -> the affected position register depends on the value contained in R [1]. If $R [1] = n$, then the point P [3] is stored in PR [n].

You can store a point in a position register: $PR [n] = [point] [operator] [point]$. The [operator] can be:

- sum (+);
- subtraction (-).

The [period] can be:

- position P [n];
- position register PR [n];
- current position of the robot in degrees axis by axis JPOS;
- current position of the robot in Cartesian LPOS.

The position records are also accessible element by element. For example, the j coordinate of PR [i] is defined by PR [i, j]. And $PR [1,2] = 250$ -> the Y coordinate of PR [1] is initialized to 250mm.

Or indirectly, we have $R [1] =$ and $R [2] = 2$. If we save coordinates like $PR [R [1], R [2]] = 250$ -> the Y coordinate of PR [1] is initialized to 250mm. Each position and orientation is therefore independently accessible. To insert into a program 1: [INST] ->

registers. To display the list of registers and their content DATA -> F1: [TYPE] -> Position Registers.

Conditional jump instructions allow you to jump (or loop) to a label in the same program if (and only if) certain conditions are true. F1: [INST] -> F / SELECT.

IF statement performs a jump based on a true condition: IF [value1] [operator] [value2] [jump]. The [value1] can be:

- register R [n];
- analog inputs-outputs AI [n] / AO [n];
- digital inputs-outputs DI [n] / DO [n];
- group inputs-outputs GI [n] / GO [n];
- robot inputs-outputs RI [n] / RO [n].

The [operator] can be:

- comparison (=);
- differentiation (<>);
- minor (<);
- major (>);
- less than or equal (<=);
- greater than or equal (=>).

The [value2] can be:

- constant;
- ON;
- OFF;
- register R [n];
- analog inputs-outputs AI [n] / AO [n];
- digital inputs-outputs DI [n] / DO [n];
- group inputs-outputs GI [n] / GO [n];
- robot inputs-outputs RI [n] / RO [n].

The [jump] can be:

- JMP LBL [n];

- CALL program.

SELECT instruction performs one or more jumps depending on the value of a register.

Structure: SELECT R [n] = [value1], [jump], [value n], [jump], ELSE, [jump].

The [values] can be:

- constant;
- register R [n].

The [jumps] can be:

- JMP LBL [n];
- CALL program.

Do not forget the ELSE instruction, since it takes into account the rest of the possible values of the R [n] register.

Waiting instructions delay the execution of a program by a specified time or until a condition is true: F1: [INST] -> WAIT.

Timing – delays the execution of a program for a specified time. Duration is expressed in seconds; there is a minimum of 0.01 seconds WAIT [time].

The [time] can be:

- constant;
- register R [n].

Wait for a condition delays the execution of a program until the condition is met.

With structure: WAIT [value1] [operator] [value2] [time]. The [value1] can be:

- register R [n];
- digital inputs-outputs DI [n] / DO [n];
- robot inputs-outputs RI [n] / RO [n].

The [operator] can be:

- comparison (=);
- differentiation (<>).

The [value2] can be:

- constant;
- ON;

- OFF;
- register R [n];
- digital inputs-outputs DI [n] / DO [n];
- robot inputs-outputs RI [n] / RO [n].

The [time] can be:

- FOREVER -> wait while the condition is not met
- TIMEOUT LBL [n] -> wait the time specified in the timeout variable (\$ WAITTMOUT), then jump to "label n" if the condition has not been met.

As a peripheral I/O we have registers (OUP). Peripheral input registers summary is in Table 1.3.

Table 1.3 Input registers (UI)

Name	Instructions
*IMSTP input UI[1] (Always enabled.)	The immediate stop signal turns servo power off by the software. The *IMSTP input is on in the normal status. When this signal is turned off, the following processing is performed: <ul style="list-style-type: none"> – An alarm is generated and the servo power is turned off. – The robot operation is stopped immediately. Execution of the program is also stopped.
*HOLD input UI[2] (Always enabled.)	The temporary stop signal specifies a temporary stop from an external device. The *HOLD input is on in the normal status. When this signal is turned off, the following processing is performed: <ul style="list-style-type: none"> – The robot is decelerated until its stops, then the program execution is halted. – If ENABLED is specified at "Break on hold" on the general item setting screen, the robot is stopped, an alarm is generated, and the servo power is turned off.

<p>*SFSPD input UI[3] (Always enabled.)</p>	<p>The safety speed signal temporarily stops the robot when the safety fence door is opened. This signal is normally connected to the safety plug of the safety fence door. The *SFSPD input is on in the normal status. When this signal is turned off, the following processing is performed:</p> <ul style="list-style-type: none"> – The operation being executed is decelerated and stopped, and execution of the program is also stopped. At this time, the feed rate override is reduced to the value specified for \$SCR.\$FENCEOVRD. – When the *SFSPD input is off and a program is started from the teach pendant, the feed rate override is reduced to the value specified for \$SCR.\$SFRUNOVLIM. When jog feed is executed, the feed rate override is reduced to the value specified for \$SCR.\$SFJOGOVLIM. When *SFSPD is off, the feed rate override cannot exceed these values.
<p>Fault RESET UI[5]</p>	<p>The RESET signal cancels an alarm. If the servo power is off, the RESET signal turns on the servo power. The alarm output is not canceled until the servo power is turned on. The alarm is canceled at the instant this signal falls in default setting.</p>
<p>START input UI[6] (Enabled in the remote state.)</p>	<p>This is an external start signal. This signal functions at its falling edge when turned off after being turned on. When this signal is received, the following processing is performed:</p> <ul style="list-style-type: none"> – When FALSE is selected for START for CONTINUE only on the Config system setting screen, the program selected using the teach pendant is executed from the line to which the cursor is positioned. A temporarily stopped program is also continued (Default). – When TRUE is selected for START for CONTINUE only on the Config system setting screen, a temporarily stopped program

	is continued. When the program is not temporarily stopped, it cannot be started.
--	--

These signals allow the robot to be controlled remotely by means of an operator panel (UOP) or PLC. The functions of the UOP outputs (UI[n] UO[n]) are predefined and can be wired on digital modular boards or configured through fieldbus boards (Interbus, Profibus, Devicenet) 18 inputs and 20/24 outputs (4 optional) can be connected (minimum 8 inputs or outputs). Peripheral output registers summary is in Table 1.4.

Table 1.3 Output registers (UO)

Name	Instructions
SYSRDY output UO[2]	SYSRDY is output while the servo power is on. This signal places the robot in the operation enable state. In the operation enable state, jog feed can be executed and a program involving an operation (group) can be started. The robot enters the operation enable state when the following operation enable conditions are satisfied: <ul style="list-style-type: none"> – The ENBL input of the peripheral device I/O is on. – The servo power is on (not in the alarm state).
CMDENBL input UO[1]	The input accept enable (command enable) signal is output when the following conditions are satisfied. This signal indicates that a program including an operation (group) can be started from the remote controllers. <ul style="list-style-type: none"> – The remote conditions are satisfied. – The operation enable conditions are satisfied. – The mode is continuous operation (single step disable).
PROGRUN output UO[3]	PROGRUN is output while a program is being executed. It is not output while a program is temporarily stopped.
PAUSED output UO[4]	PAUSED is output when a program is temporarily stopped and waits for restart.

HELD output UO[5]	HELD is output when the hold button is pressed or the HOLD signal is input. It is not output when the hold button is released.
FAULT output UO[6]	FAULT is output when an alarm occurs in the system. The alarm state is released by the FAULT_RESET input. FAULT is not output when a warning (WARN alarm) occurs.

To get to the robot's input and output configuration screen, we must press: MENU - I / O and pressing F1: [TYPE] we see the different types of inputs and outputs (Figure 1.33).

I/O Digital Out					JOINT 10 %
#	RANGE	RACK	SLOT	START PT	1/32
1	DO[1- 8]	1	2	1	
2	DO[9- 16]	1	2	9	
3	DO[17- 24]	2	1	1	
4	DO[25- 32]	2	1	9	
5	DO[33- 40]	0	0	0	
6	DO[41- 48]	0	0	0	
7	DO[49- 56]	0	0	0	
8	DO[57- 64]	0	0	0	
9	DO[65- 72]	0	0	0	

I/O Digital Out					JOINT 10 %
#	SIM	STATUS			1/256
DO[1]	U	OFF	[]	
DO[2]	U	OFF	[]	
DO[3]	U	OFF	[]	
DO[4]	U	OFF	[]	
DO[5]	U	OFF	[]	
DO[6]	U	OFF	[]	
DO[7]	U	OFF	[]	
DO[8]	U	OFF	[]	
DO[9]	U	OFF	[]	
DO[10]	U	OFF	[]	

Figure 2.28 I/O peripheral registers configuration screens

In any input and output screen, if we press F2 [CONFIG] we enter their configuration screen.

The parameters of these I/O are the same as those of the digital I/O. Note that the ranges of these groups cannot be overlapped on the I / O map. For configuring cell interface I/O appears we need press the MENU key and select I/O, then press F1, [TYPE] and select Cell Interface. The cell input screen or cell output screen appears. The cell input screen is shown below as an example in the Figure 1.34. The display contents differ depending on the program start method.

I/O Cell Inputs					
					1/1
INPUT SIGNAL	TYPE	#	SIM		STATUS
1 Tryout Mode	DI[0]	U		***

[TYPE]	CONFIG	IN/OUT	SIM	UNSIM>	>
----------	--------	--------	-----	--------	---

Figure 2.29 Configuring cell interface I/O

To switch between the input screen and the output screen, press F3, IN / OUT. The cell output screen is shown below in the Figure 1.35. The display contents differ depending on the program start method.

I/O Cell Outputs					
					1/10
Output signal	Type	#	SIM		STATUS
1 Set if INPUT SIMULATED	DO[0]	U		***
2 Set if OUTPUT SIMULATED	DO[0]	U		***
3 OVERRIDE = 100	DO[0]	U		***
4 In cycle	DO[0]	U		***
5 Abort Program	DO[0]	U		***
6 Tryout Status	DO[0]	U		***
7 Heartbeat signal	DO[0]	U		***
8 MH Fault	DO[0]	U		***
9 MH Alert	DO[0]	U		***
10 Robot motion G1	DO[0]	U		***

[TYPE]	CONFIG	IN/OUT	SIM	UNSIM	>
----------	--------	--------	-----	-------	---

Figure 2.30 Output screen

We can see robot configuration summary from the system configuration screen. To obtain a summary of the configuration of the robot, we will access as follows through the TPE controller.

```

System/Config                                JOINT 30#
                                           1/27
1 Use HOT START:                            FALSE
2 I/O power fail recovery: RECOVER ALL
3 Autoexec program                          [*****]
  for Cold start:
4 Autoexec program                          [*****]
  for Hot start:
5 HOT START done signal:                    DO[0]
6 Restore selected program:                 TRUE
7 Enable UI signals :                      TRUE
8 START for CONTINUE only :                FALSE
9 CSTOP1 for ABORT :                      FALSE
10 Abort all programs by CSTOP1 : FALSE
11 PROD_START depend on PNSTROBE :FALSE
12 Detect FAULT_RESET signal :             FALL
13 Use PPABN signal :                      < *GROUPS * >
14 WAIT timeout :                          30.00 sec
15 RECEIVE timeout :                       30.000 sec
16 Return to top of program :              TRUE
17 Original program name (F1) : [PRG      ]
18 Original program name (F2) : [MAIN    ]
19 Original program name (F3) : [SUB     ]
20 Original program name (F4) : [TEST    ]
21 Original program name (F5) : [*****]
22 Default logical command : < *DETAIL * >
23 Muximum of ACC instruction :            150
24 Minimum of ACC instruction :            0
25 WJNT for default motion :              *****
26 Auto display of alarm menu :           FALSE
27 Force Message :                        ENABLE
28 Reset CHAIN FAILURE detection : FALSE
29 Allow Force I/O in AUTO mode : TRUE
30 Allow chg. ovr. in AUTO mode : TRUE
31 Signal to set in AUTO mode DOUT [ 0]
32 Signal to set in T1 mode DOUT [ 0]
33 Signal to set in T2 mode DOUT [ 0]
34 Signal to set if E-STOP DOUT [ 0]
35 Hand broken :                          < *GROUPS * >
36 Remote / Local setup :                  Remote
37 External I/O (ON : Remote) : DI [ 0]

[TYPE]                                     [CHOICE]

```

Figure 2.31 Summary of robot configuration

Profibus network configuration screen access by using keys:

- <Menu> 6 Setup;
- F1 [TYPE] 0 NEXT 7 PROFIBUS;
- F3 [OTHER].

Where:

- 1 Slave- Slave profibus configuration
- 2 Master- Configuration of the profibus of the master.
- 3 Bus Param.- Configuration of the profibus parameters.
- 4 Slave Param. - Configuration screen of the parameters of the different slaves of the robot.

Inside the "Slave Param" screen. Pressing F2 [DETAIL] we enable/disable the slave, configure its address, set the name, input and output bytes, flags. Configuration of inputs and outputs:

- Digital I / O:
 - Rack- Address of the robot's I / O card (66 Master; 67 Slave);
 - Slot- Physical place where the I / O card is located;
 - Range- Number of “associated” I / O;
 - Start- Bit number at which the I / O range begins.
- Groups:
 - Rack- Address of the robot's I / O card (66 Master; 67 Slave);
 - Slot- Physical place where the I / O card is located;
 - Range- Number of “associated” I / O;
 - Start- Bit number at which the I / O range begins.

To configure a Payload we will have to access the system screen <Menu> 0 NEXT - 6 SYSTEM - F1 [TYPE] - 6 MOTION. We set the weight, the position of the center of gravity and assign a name to it. We have up to a maximum of 10 different payloads.

In case of Profibús the information transmission system that greatly simplifies the installation and operation of industrial machines and equipment used in production processes.

In Payload the weight of the “robotic doll” including the weight of the EOAT (End of Arm Tool) and the part. The "payload" is characteristic of each robot model and allows us to choose the robots suitable for the previously specified processes.

Macros Settings - a MACRO is a program that performs a specific operation whose execution can be commanded by:

- Activation of a user key of the Teach Pendant (UK [n]). Group Mask (*; *; *; *; *)
- Activation of a user key of the teach pendant SHIFT + (SU [n])
- Activation of a controller user key (option) (SP [n])
- Selecting an item from the MANUAL FCTNS menu (MF [n])
- CALL instruction TPE Macro Keys

- RUN instruction
- The activation of an input (DI [n] / RI [n]). To enlarge \$ MACROMAXDRI.
- The activation of a UI [n] input.

To create a Macro, the following steps must be performed:

- Press the SELECT key.
- Press F2: CREATE.
- We give a name to the macro.
- We press the F2 key: DETAIL.
- In section 2 (Subtype), press F \$: CHOICE and select the MACRO option and press END.
- We edit the macro that we have created and write the code that we want to be executed when we call this macro, for example from a program.
- Later we configure the macro, for that we will press:
 - 1) MENU
 - 2) 6 SETUP
 - 3) F1: TYPE
 - 4) 3 MACRO
- In the “Macro Command” screen, we place the cursor on Program.
- We press F4: CHOICE and the list of programs will appear (we choose the one defined)
- We place the cursor on the Assign column, and press F4: CHOICE.
- We select MF, so that the macro can be called from another program.

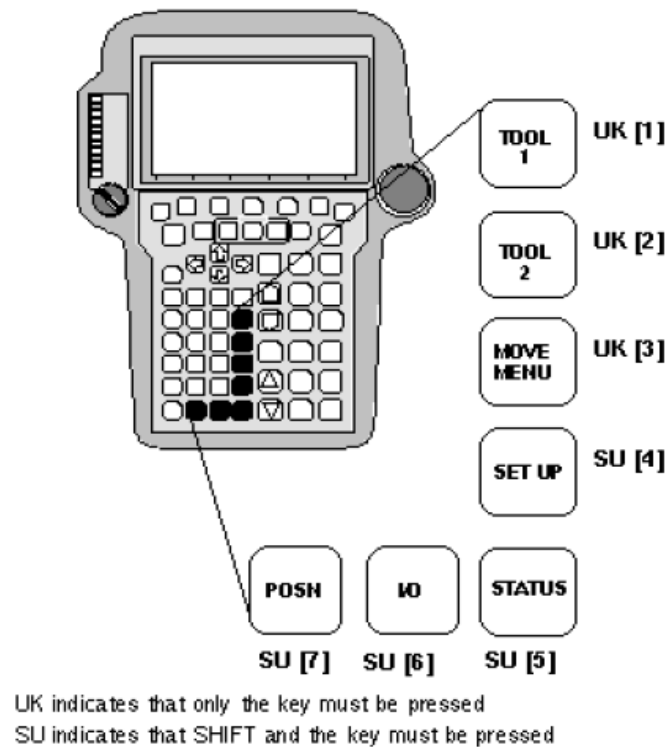


Figure 2.32 Direct macro execution buttons

Axis limit adjustment. There are 3 types of axis travel limitation: software limits, electrical limits and mechanical limits.

Software limits – these are the first limits the robot encounters (if they are correctly defined). When a software limit is reached, the robot does not fail, it simply stops and does not allow movement in that direction. To be able to move the robot again, it is sufficient to move the robot in reverse. If we want to access the image screen, we must proceed as indicated below:

- 1) MENU
- 2) 0-NEXT
- 3) 6-SYSTEM
- 4) F1- [TYPE],
- 5) Axis limits.

SYSTEM\Axis\Limits			JOINT	10 %
AXIS	GROUP	LOWER	UPPER	1/16
1	1	-90.00	90.00	dg
2	1	-50.00	90.00	dg
3	1	-130.00	205.00	dg
4	1	-360.00	360.00	dg
5	1	-125.00	125.00	dg
6	1	-360.00	360.00	dg
7	0	0.00	0.00	mm
8	0	0.00	0.00	mm
9	0	0.00	0.00	mm

[TYPE]

Figure 2.33 Software limits

For software limit modifications to be taken into account, the controller must be shutdown and rebooted.

Mechanical limits - it is possible to set certain mechanical limits depending on the axes and the robot model. If a mechanical limit is reached, the electrical limits and software limits must be verified. An engine overconsumption collision alarm will occur.

Remote program start via UI: to use the UOPs, the following protocol must be respected:

- Configure the UOP system signals.
- Wire the mandatory system signals and those that are desired to control the installation.

- For the input signal UI [6: START] to take effect, two conditions must be met:

Enable UI signals:

- 1) MENU
- 2) O-NEXT
- 3) 6-SYSTEM
- 4 F1: TYPE
- 5) 5-CONFIG
- 6) ENABLE UI SIGNALS to TRUE.

The robot has to give us the output signal UO [1: CMD ENABLE] = ON:

When does this situation occur?

- 1- UI [1: * IMSTP] = ON, no external software emergency is received.
- 2- UI [2: * HOLD] = ON, no external program stop is received.
- 3- UI [3: * SFSPD] = ON, no program stop associated with a start with a predefined speed in a variable is received.
- 4- UI [8: * ENABLE] = ON, enabling robot movements is allowed.
- 5- Key T1, T2, AUTO is in AUTO mode, which means that the external hardware safeguards are enabled.
- 6- Controller in REMOTE mode, allowing the robot to start from a remote system, for example a start button associated with input UI [6: START] that will generate a pulse that will have its effect on the falling edge. For this R-J2 and R-J3 the LOCAL / REMOTE key must be positioned in REMOTE. In the R-J3i controller we must configure in Menu, 0-Next, 6-System, F1: TYPE, Config, line 36, Option, Local / Remote = Remote
- 7- System variable \$ RMT_MASTER = 0 if it is not. MENU, 0- NEXT, 6- SYSTEM, F1: TYPE, 2- VARIABLES.
- 8- Make sure that the option “Start For Continue Only” is set to false: MENU, 0- NEXT, 6- SYSTEM, F1: TYPE, 5- CONFIG, START FOR CONTINUE ONLY to “FALSE”. (if it was “True”, set it to False and then OFF / ON for it to take effect, or modify the variable \$ SHELL_CFG. \$ CONT_ONLY = FALSE)
- 9- Teach Pendant in OFF and in non-STEP conditions (step by step).
- 10- UO [2: SYS READY] = ON, the robot has no fault. Fault reset: Reset of external faults via software through the UI’s. Reset of external faults via hardware (External Emerg, Fence Open, ...)
- 11- FCTN, 1-Abort All, Select, select the program to start.
- 12- UI [6: Start] has its effect on the robot with a falling edge.

2.2 Programming in the simulator software editor

Thanks to the Roboguide HandlingPRO simulation tool, and the graphical software interface, we can carry out the programming of the robot following the application menus with a much more comfortable and visual environment than programming using TPE. In the following section, everything related to the use of the simulation interface, functionalities

and ultimately the offline programming of a robotic process using the HandlingPRO simulation tool of the Roboguide software will be discussed.

Workcell Wizard – represent based steps through the material handling application: choosing robot model, controller, gripper, machines.

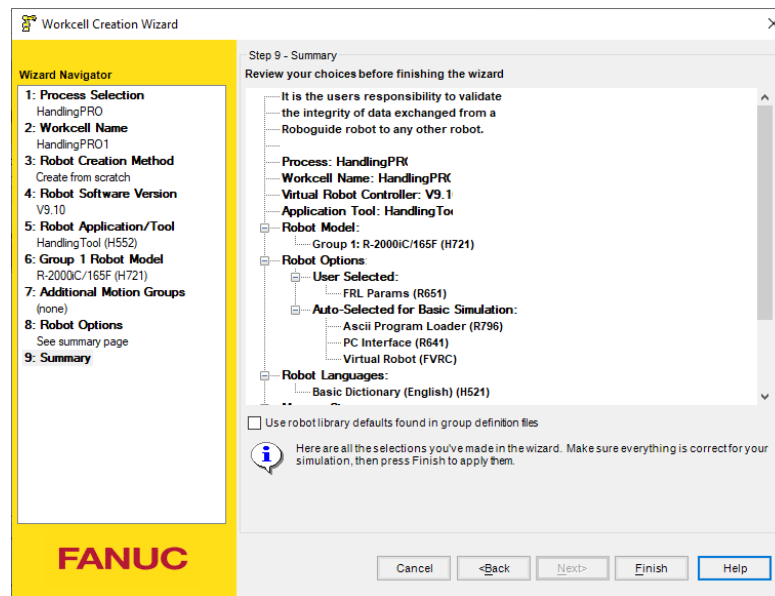


Figure 2.34 Workcell Wizard

Navigation in software is provided by:

- Menu – represents based operations.
- Toolbar – sets of shortcut icons for the basic menus.
- Status bar - representing a position of the current robot, tools and icons which perform operations (such as modifying a robot position, undoing the robot's jogging).
- Workcell Browser – provides review and changing of each detail of the workcell that can be expanded, checked and modified.

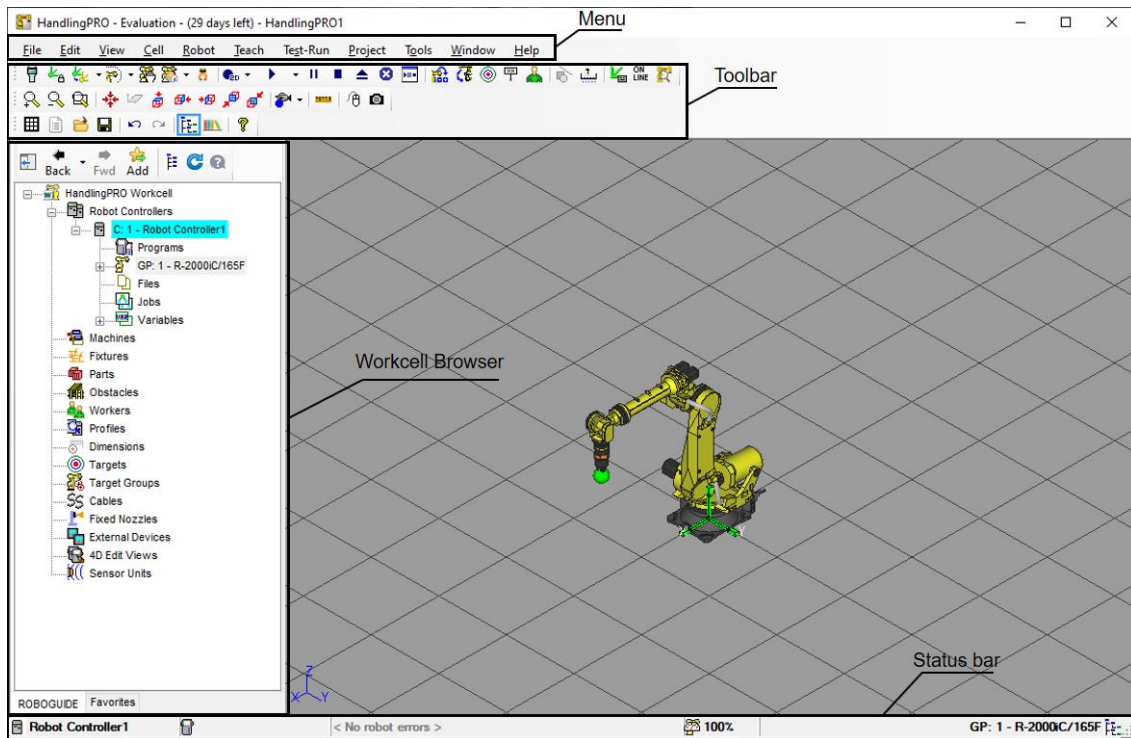


Figure 2.34 Roboguide Software

The main configurations of the robot can be carried out from the application using mainly the main menu bar, the toolbars and the “Cell Browser” menu.

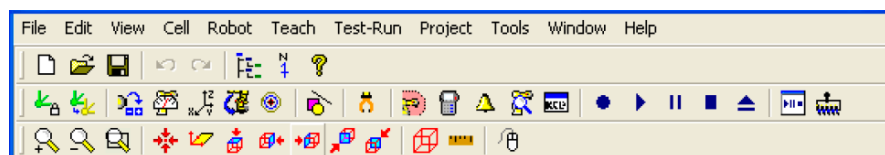


Figure 2.35 HandlingPRO Roboguide Software Quick Access Bars

In the first bar (main menus) we can distinguish the main menus that the application consists of (File, Edit, View, Cell, Robot, Teach, Test-Run, Project, Tools, Window and Help), whose names indicate the menu which we can access. In each of them we can see the different options that they consist of.

The second bar shows us the file options, the "Undo" and "Redo" buttons and finally the visibility of the "Cell Browser" and "Navigator", as well as access to the help of the current window.

The third bar shows us the quick access buttons to the most used functionalities and the most useful quick bars (Teach tool selection, Moveto Retry, Jog Coordinates Quick Bar,

Gen Override Quick Bar, Teach Quick Bar, Moveto Quick Bar, Target tools) . We also have buttons for interaction with the graphic environment (Draw features on parts, Hand, Work Envelope, Teach Pendant, Robot Alarms, Robot Browser and KCL). The following set of buttons allows us to control the execution of our simulation and even generate an AVI video file (Record, Play, Pause, Stop, Fault Reset, Fault Reset and Position Edit).

Each of the quick buttons and menus that they access in the application is detailed below:

- File options - they allow to create, open or load a simulation file.
- Undo/Redo - allow undo and redo the actions carried out in the simulation.
- Cell Browser - main menu where to configure each of the simulation objects.

File Options



Undo / Redo



Cell Browser



Figure 2.36 Quick buttons 1

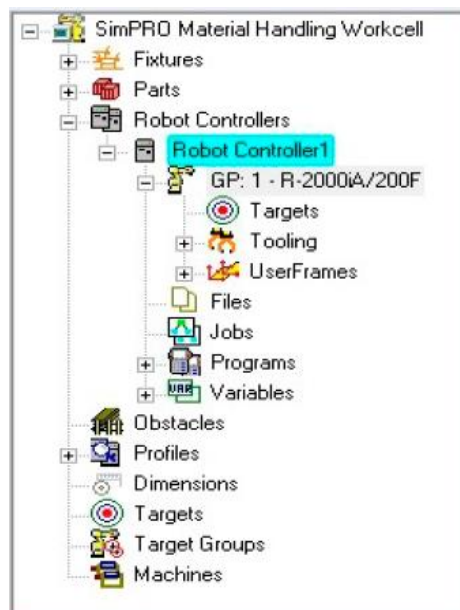


Figure 2.37 Image of the “Cell Browser” menu

Navigator - this menu tells us step by step each of the actions to be able to program a simulation.

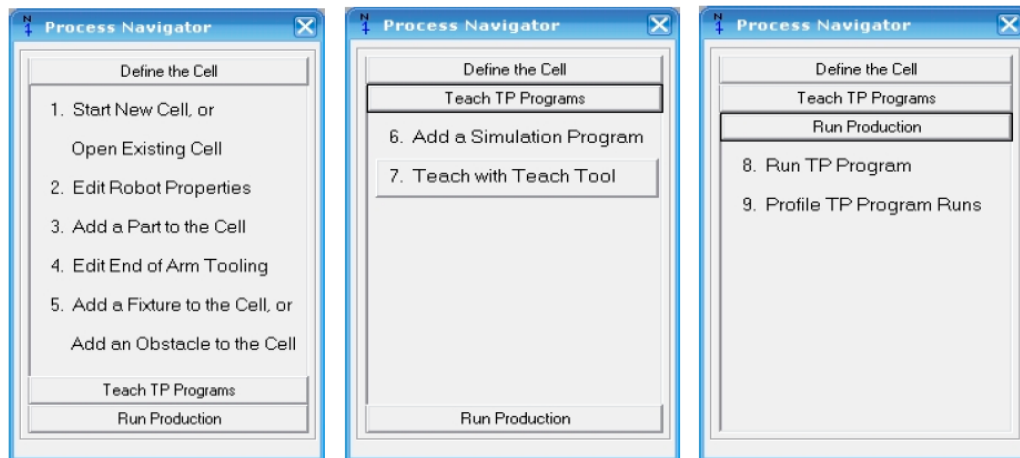


Figure 2.37 Image of the “Navigator” menu

Help current window - shows the help menu of the window in which we are working at any given moment. Teach Tool Selection - allows blocking or unblocking the use of the learning tool by selecting the graphical environment. It facilitates the use of the tool since the rest of the objects in the environment are disabled. This method is equivalent to defining points using “teach by nose”. Move to Retry - enables or disables the retry of the “MoveTo” function if it fails. Jog Coordinates Quick Bar - shows or hides the quick bar of the robot movement coordinate system.



Figure 2.38 Quick buttons 2



Figure 2.39 Image of the “Jog Coordinates Quick Bar” menu

Override Quick Bar - shows or hides the override speed setting bar. Teach Quick Bar - show or hide the quick learning bar.

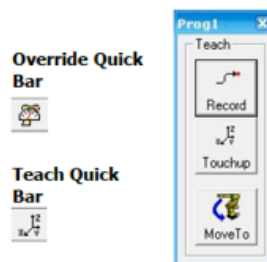


Figure 2.40 Image of the “Teach Quick Bar” menu

MoveTo Quick Bar - shows or hides the quick robot movement bar to certain surfaces, edges, vertices.

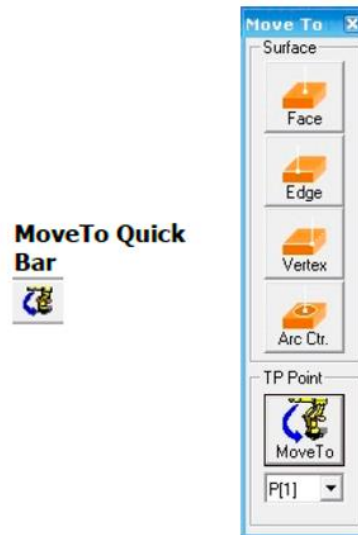


Figure 2.40 Image of the “MoveTo Quick Bar” menu

Target Tools - shows the toolbar for the configuration of the different positions (objectives) to be configured.



Figure 2.41 Image of the “Target Tools” menu

Draw features on parts - allows to draw different lines, shapes and surfaces on some objects in the simulation.

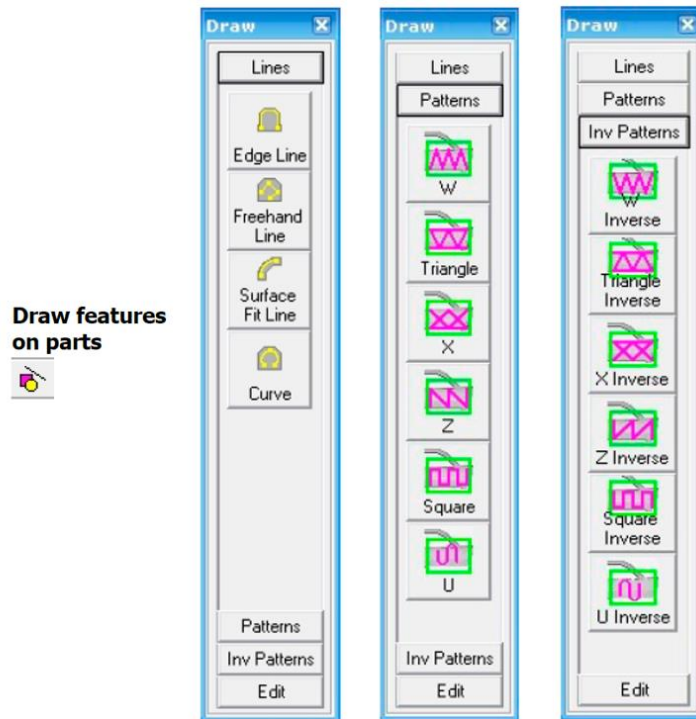


Figure 2.42 Image of the “Draw features on parts” menu

Open/Close Hand - it offers us the possibility of opening and closing the tool configured in the robot's TCP. Work Envelope - shows the scope or range of movement allowed for the robot with which we are performing the simulation.

Open/Close Hand



Work Envelope



Teach Pendant



Figure 2.42 Image of the menus related to the tool

Teach Pendant - shows or hides the Virtual Teach Pendant provided by the HandlingPRO simulation tool.

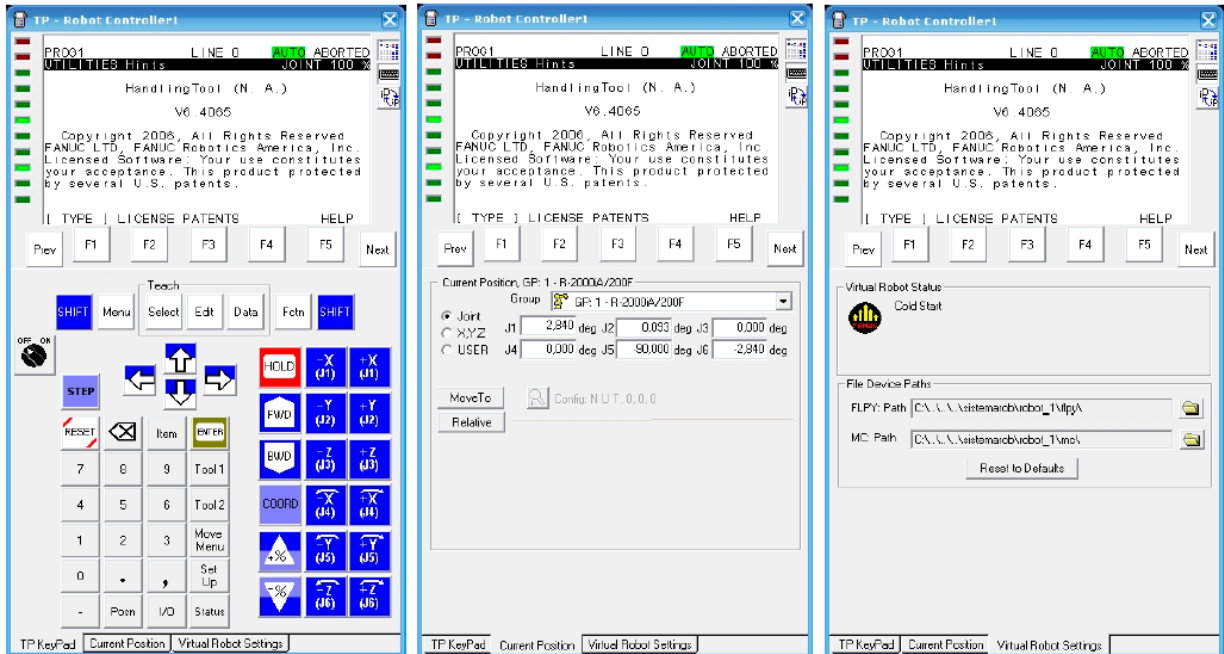


Figure 2.42 Image of the “Teach Pendant” menu

Robot Alarms - shows or hides the robot's alarm console.

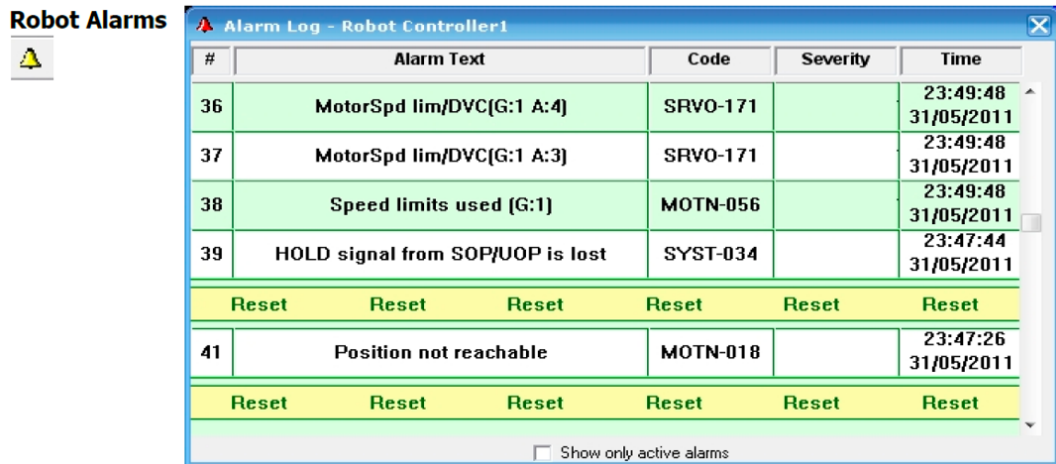


Figure 2.43 Image of the “Robot Alarms” menu

Robot Browser - shows or hides the robot browser to show general information, input and output and monitoring of some indicators of the controller.

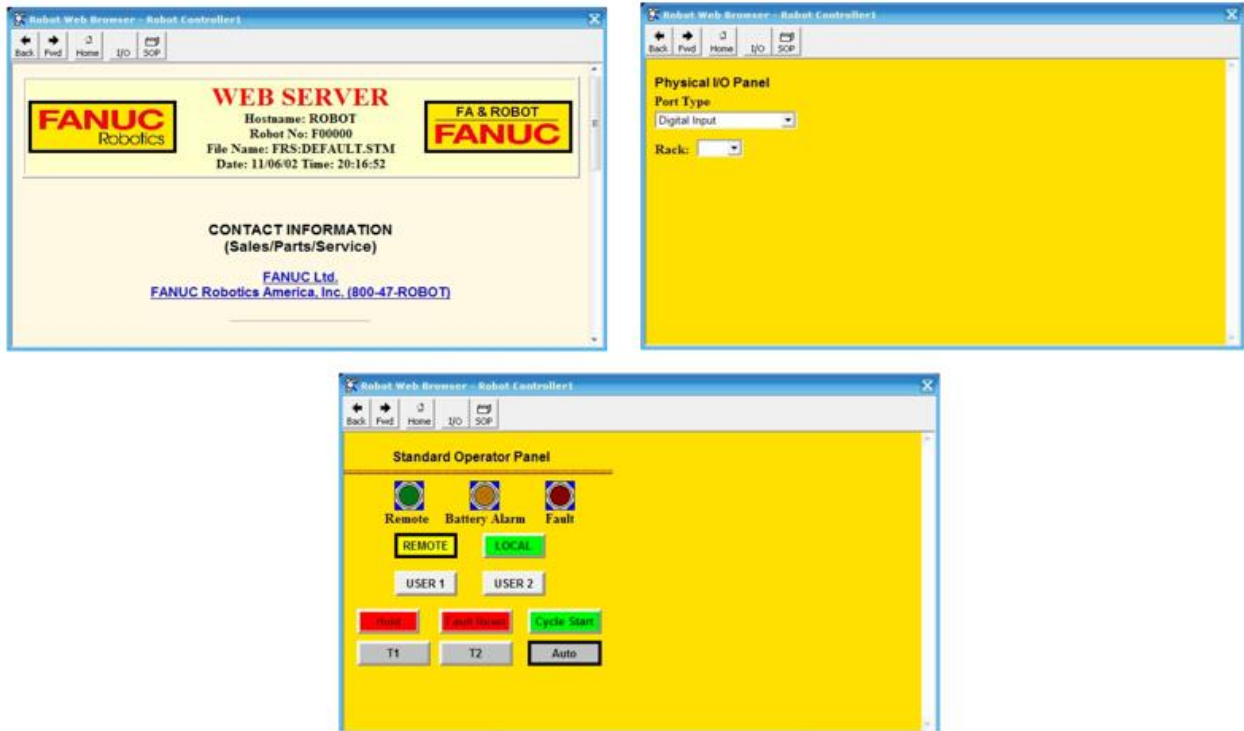


Figure 2.44 Image of the “Robot Browser” menu

Record AVI - makes possible to obtain a video in AVI format of the simulation. Cycle Start - starts the simulation cycle with the sequential execution of the instructions. Hold - pause the simulation. Abort - abort the process simulation. Fault Reset - resets the set of errors obtained in the simulation execution.

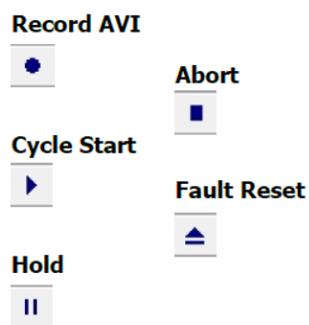


Figure 2.44 Image of menus for recording video

Run Panel - shows or hides the simulation run panel.

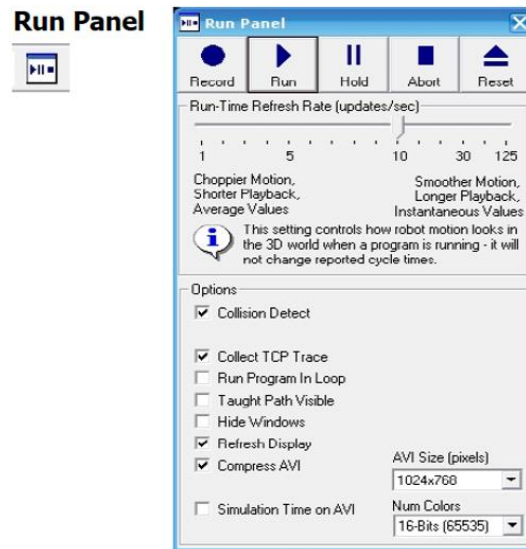


Figure 2.45 Image of the “Run Panel” menu

Position Edit - shows the window for editing the position of the different points defined in the instructions. It also allows you to edit and redefine them in a comfortable way.

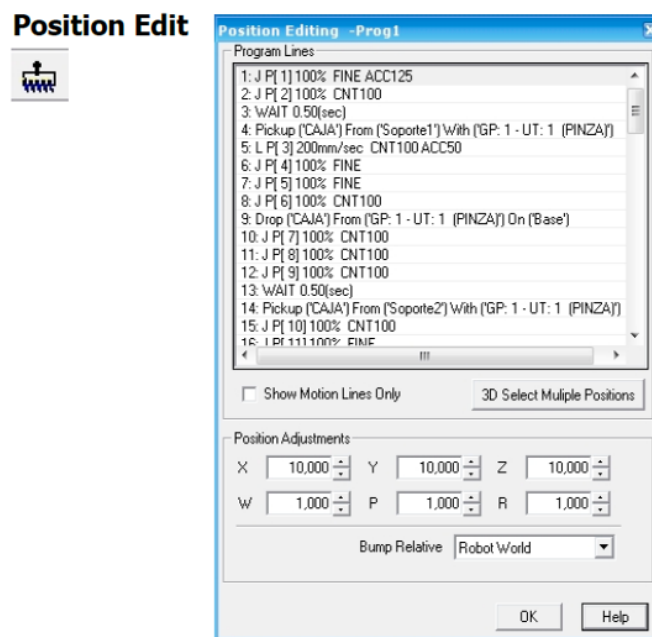


Figure 2.46 Image of the “Position Edit” menu

Zoom Options - it offers the options to enlarge view, zoom out and window zoom. Center View Selected Object - centers the selected object on screen. Change Camera Direction - it makes it possible to change the camera in the direction of the three axes of the environment coordinate system. View Options - provides the option to change the front, side, rear, and overhead views of the simulation environment.

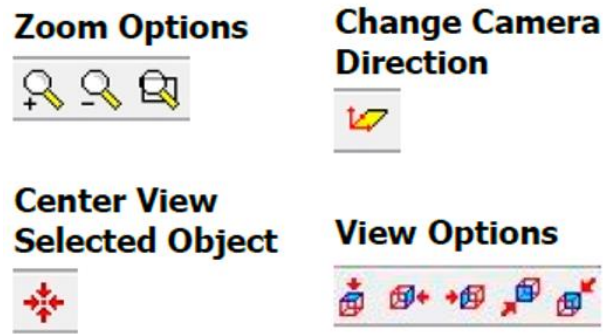


Figure 2.47 Image of menus for visual representing

View WireFrame - changes the presentation of the simulation showing the edges of the objects.

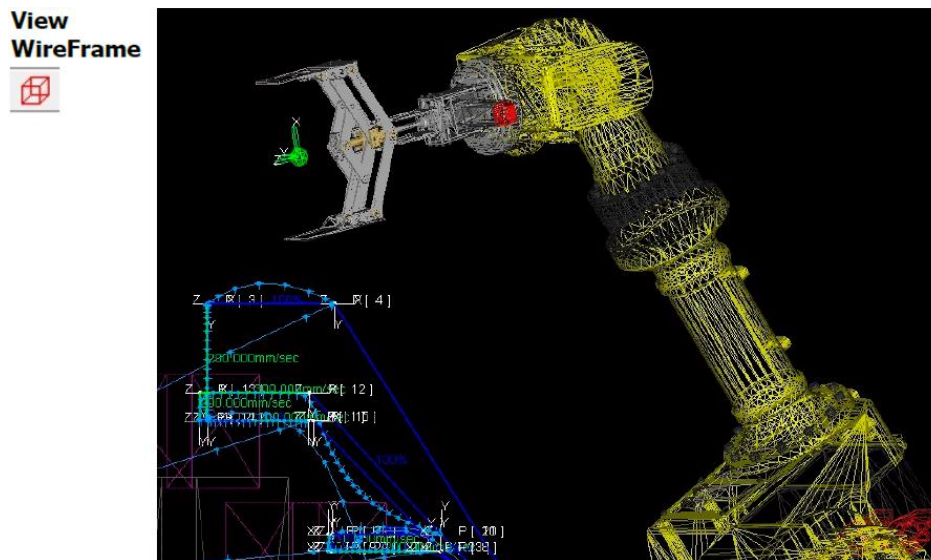


Figure 2.48 Image of the “View WireFrame”menu

MeasureTool - displays the measurement tool-menu.

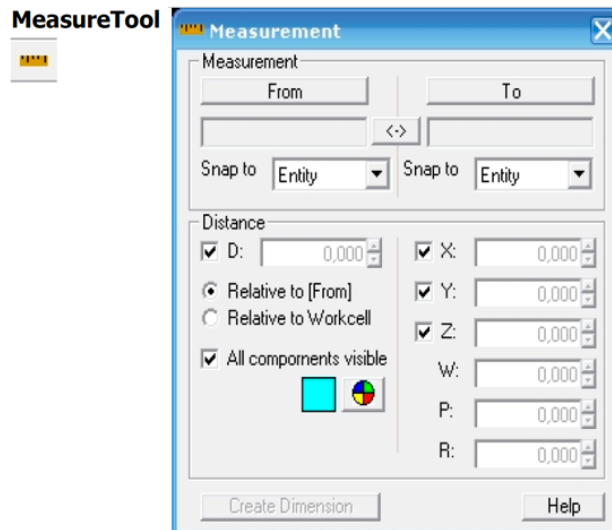


Figure 2.49 Image of the “MeasureTool”menu

Mouse Commands - show or hide special combinations and mouse commands.

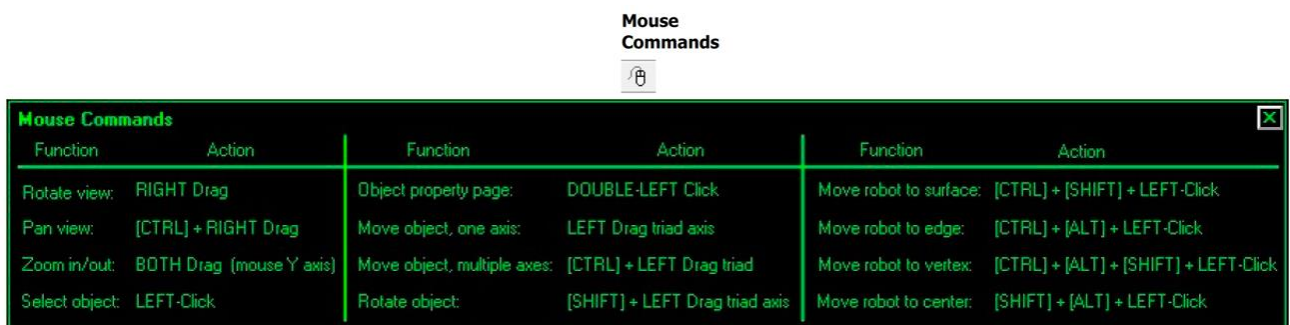


Figure 2.50 Image of the “Mouse Commands”menu

Once the quick access buttons have been specified, we will describe the functionalities of the Roboguide application accessible through the main menus of the application. These allow us to perform all the functions of the simulation application, in addition to the functions of the buttons on the quick access bar.

The main menus of the application will be described below and the actions they allow to perform in the simulation will also be described. Description of main menus. File - it is the menu that allows you to manage the files of the simulations.

From it we can handle the options of creating a new simulation, opening a saved simulation job, saving the work done for the current simulation and other file management tasks. Among them, we highlight the option of exporting the work to the IGES data standard of CAD design programs using the “Export” option, thus generating a file with the extension

type “.igs” to be imported into a CAD application. It should be noted among the options that of "Package cell", which is the utility that we must use to be able to take a simulation job from one computer to another with Roboguide installed, since the options of "restore save point" "save cell" and " open cell ”are for simulation jobs of the same equipment. To load this simulation package onto a new computer, we can do it by double clicking on the “.frw” file (in our case SistemaROB.frw).

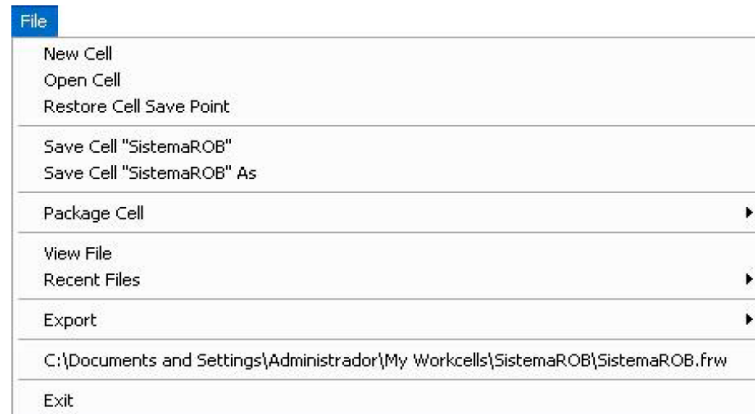


Figure 2.51 Image of the “File” menu

Edit - is the standard object edit dropdown. The options offered are those of "undo", "redo", "cut", "copy", "paste" and "delete" to apply them to any object in the simulation environment.



Figure 2.51 Image of the “Edit” menu

View - this menu shows all the display options of the simulation environment. We can configure the view modes described above in the quick access buttons, in addition to the movement configuration quick access bars. We highlight the option "Program Details"

that offers us different options for viewing details related to the robot, configured EOATs and additional movement options.

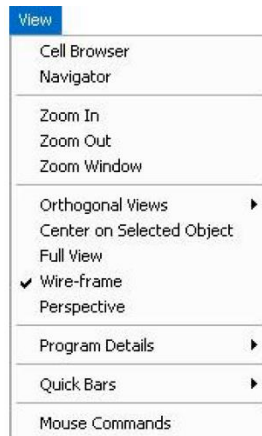


Figure 2.52 Image of the “View” menu

Cell - the menu contains all the options related to the simulation (work cell) and the objects that it may have. From this menu we can add the aforementioned objects, such as robots, fixed elements, moving parts, obstacles and objectives. In this set of options, when defining a new object, we can choose to load a CAD design file from the predefined library, custom CAD designs made by the user, and typical geometric bodies, such as cubes, cylinders and spheres. Among the options, we highlight the “I/O interconnectons” that allow interconnecting the input and output interfaces between robots, and the “Workcell Properties” that show general configuration properties of the cell.

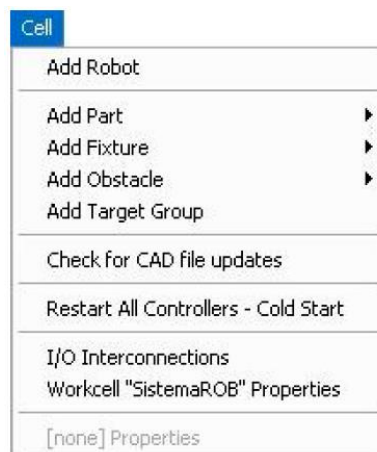


Figure 2.52 Image of the “Cell” menu

Robot - the menu indicates the options we can choose to handle the robot in question. We have access to the “Teach pendant” console to make this configuration, in addition to using the rest of the menus and the “teach tool”. The “teach tool” is the main tool for TCP movement, which will allow us to define points in space and reach objectives in a simple way, in order to later obtain the coordinates and adjust the movement with greater precision. For this we can use the TP interface or the program definition menu "Teach prog1 program", which we will explain in the next point. We will be able to find several of the quick access buttons, and as a noteworthy option that of “Robot GP1: Properties model”, which will allow us to access the configuration options of the selected robot.



Figure 2.53 Image of the “Robot” menu

Teach - the “teach” menu shows us the options that we have to carry out the programming of the movements and actions that the robot can carry out in the different simulation programs. We can add a simulation program, a program through the “teach pendant” console, or load an already defined program.

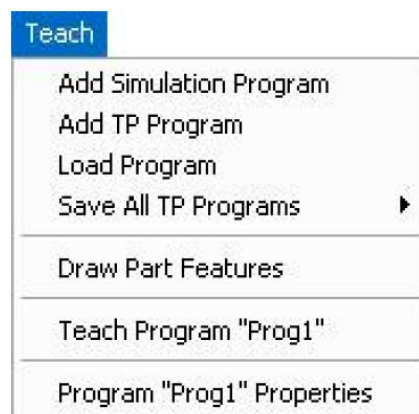


Figure 2.54 Image of the “Teach” menu

We also have access to the "draw part features" menu described above in the quick access bars, as well as two new menus, "Teach Program ProgX" which will allow us to access the main programming menu.

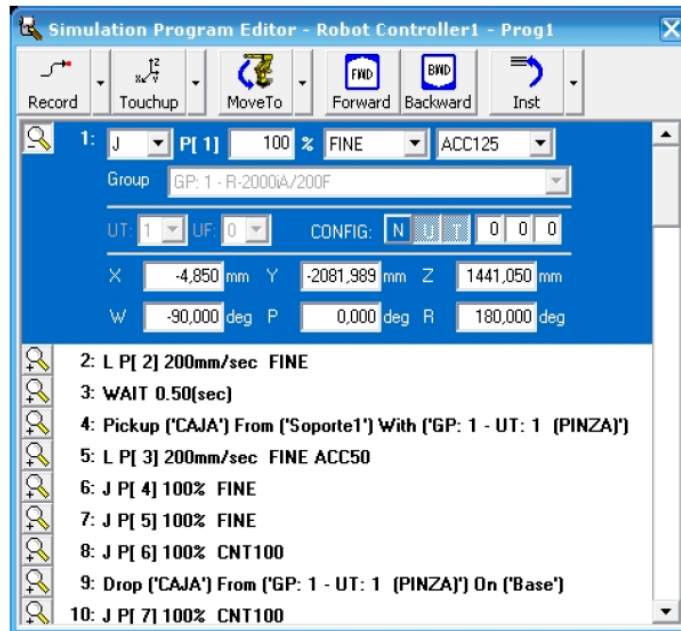


Figure 2.55 Image of the “Teach Program”menu

And "Program ProgX Properties" which shows us the configuration options for the selected Program.

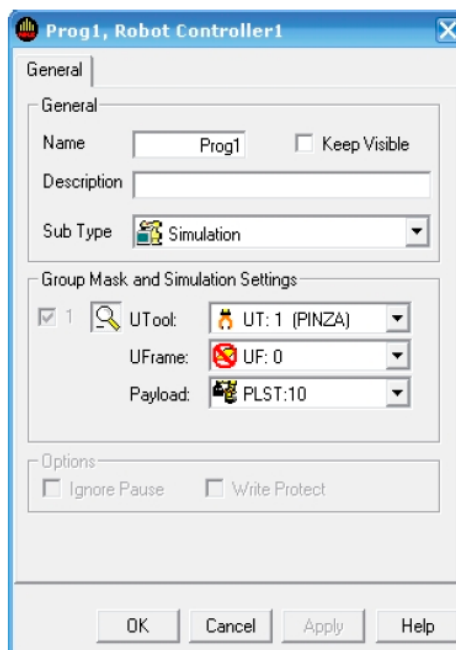


Figure 2.56 Image of the “Program Properties”menu

Test-run - the drop-down offers us the different simulation execution options. We can access the execution panel, execution configuration and execution options.



Figure 2.57 Image of the “Test-run” menu

Among the options we can find some functionalities, among which we can highlight the "Collision Detect", which analyzes the possible collisions that may occur during the execution of the simulation.

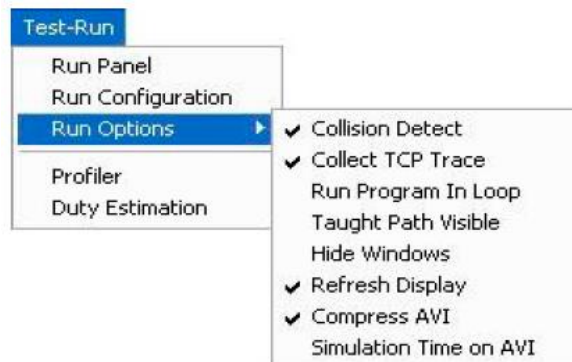


Figure 2.58 Image of the “Run Options” of the menu “Test-run”

Project - the menu offers us an alternative way to access the development options for the controller used. The options that appear are equivalent to the right-click menus of the “Cell Browser” development buttons. We highlight the “Export” and “Import” options, which allow us to massively export and import simulation files, programs and system configurations to the simulation application or to the production robot.

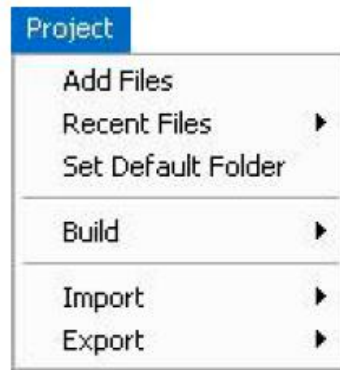


Figure 2.58 Image of the main menu "Project"

Tools - allows us to access the default directory of the selected robot to treat the associated files, and also gives us access to a configuration menu of general options, such as the creation of templates in point definition sentences, image refresh time in "teach time", configuration of object colors, definition of application work paths.

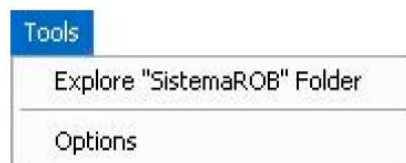


Figure 2.59 Image of the main menu "Tools"

Window - offers us the possibility of changing the application interface types, being able to define a simpler or more complex graphical interface, through the option "3D Panes".



Figure 2.60 Image of the main menu "Window"

Help - shows the different help options that the program offers.



Figure 2.61 Image of the main menu "Help"

2.3 Positioning error of industrial robots

The positioning accuracy of industrial robots is determined by the design features of the mechanical unit, geometric errors, errors of control systems and feedback sensors, as well as thermal errors and errors caused by the action of gravity forces. Also, some error is caused by the impossibility of the robot control system to recognize the difference in two positions, the conditional distance between which is less than the robot's digit capacity. When performing a technological operation, the actual position of the movement of the center point of the working tool (TCP) differs from the calculated one. Its position, speed and acceleration at any point of the trajectory realized by it in the general case may not coincide with the calculated ones.

Geometric errors include deviations of the linear dimensions of the links and the shape of their longitudinal axes from the specified values. As a result of the presence of technological errors arising in the manufacture of parts of the links, as well as errors arising during the operation of the mechanical unit (temperature, power and wear), the actual dimensions of the links that determine the position of the TCP in space differ from the ideal ones, on the basis of which the calculation algorithms were compiled and the actual position of the working body differs from the calculated one. Such errors are called primary errors, and the effect of each such error on the value of the positioning error of the industrial robot TCP point can be determined independently of the effect of other primary errors. The resulting positioning error is calculated in accordance with the superposition principle. To implement the movement of the working body, it is necessary from the control system with the help of autonomous motors to set movements for several or all links of the mechanical block of an industrial robot. Since the control system and motors operate with errors, as a result, the control is carried out inaccurately and the actual movements of the

links differ from the calculated ones. Robots have a number of error sources that affect the accuracy, these include but are not limited to:

- Commissioning errors, such as the definition of the: tool center point (TCP) offset, workpiece and robot spatial relationship.
- Manufacturing/mechanical errors, such as: linkage length, axis perpendicularity, eccentricities, linkage elasticity, backlash.
- Dynamic & inertia-based errors.
- Temperature based errors: robot warm-up, ambient temperature influences.

One of the most cost effective, and high impact methods for improving a robotic application is to accurately measure the relationship between the robot base position, the tool center point of the end-effector and the workpiece. Taking each in turn:

The robot base is the origin of the robot. In practice, it is the starting point for any forward kinematics calculations; however, it is essentially impossible to measure directly. The physical robot base does not coincide with the same robot base used in motion control.

The tool centre point is the frame on the end-effector that is used to interact with the work-piece; this could be a gripper, spindle or weld nozzle. The TCP needs to be defined in six degrees of freedom (6DOF) from the manufactured end of the robot (the tool flange). A CAD/drawing nominal is often used to define the TCP. Otherwise, robots have a built-in routine that can derive the TCP. Both methods are likely to introduce inaccuracy.

The workpiece is the object (or objects) that the robot process is interacting with. Robots use routines for locating the part relative to the robot base, but again, this is a relatively subjective process that can vary depending on how the operator programs it.

The geometric relationship between these parameters can be determined using large volume metrology tools, with minimal time and cost impact to significantly enhance the cell accuracy.

The most common manipulators today are anthropomorphic and resemble a human hand. Low load capacity and low accuracy are due to the architecture of the existing manipulators and, in particular, the sequential arrangement of their links. Each of them carries the weight of the subsequent segment in addition to the payload, so they are subjected to large bending moments, which increases the stiffness requirements and therefore leads to

an increase in mass. A significant source of positioning errors is the violation of the specified geometric relationships between the link axes. The sequential arrangement of the links together with the requirement for their rigidity implies that the moving parts of the robot have a significant mass. As a result, during high-speed movements, the manipulator is influenced by inertial forces, centrifugal forces and Coriolis forces, which complicates the control of the robot. The robots' accuracy is determined by the positioning errors of the TCP characteristic point and the errors of its angular orientation.

Positioning errors are determined by technological deviations in the dimensions of the manipulator links, gaps in the kinematic pairs of the manipulator and drive mechanisms, deformations (elastic and temperature) of the links, as well as errors of the control system and feedback sensors.

The transition matrix T_n connecting the TCP {T} coordinate system and the world coordinate system {B} can be represented as follows:

$$T_n = A_1 A_2 A_3 \dots A_n, \quad (2.1)$$

$$A_i = Rot(Z, \theta_i) \cdot Trans(0, 0, d_i) \cdot Trans(a_i, 0, 0) \cdot Rot(X, \alpha_i) \quad (2.2)$$

where A_i - matrix of transformation of coordinates between links $i-1$ and i of the kinematic chain, a_i - is the length of the i -link of the kinematic chain, α_i - is the angle of the i -link of the kinematic chain, d_i - is the deviation of the i -link of the kinematic chain, and θ_i is the angle of rotation of the i -joint.

Equation (2.2) implies that A_i depends on 4 parameters. In the case of free rotation, the angle θ_i - is a variable of parameters, while the other three are fixed. And for a swivel mechanical block, the angle θ_i is the joint coordinate. Differentiation of equation (2.2) gives:

$$dA_i = \frac{\partial A_i}{\partial a_i} \Delta a_i + \frac{\partial A_i}{\partial \alpha_i} \Delta \alpha_i + \frac{\partial A_i}{\partial d_i} \Delta d_i + \frac{\partial A_i}{\partial \theta_i} \Delta \theta_i = A_i \delta A_i, \quad (2.3)$$

where Δa_i - offset a_i , $\Delta \alpha_i$ - offset α_i , Δd_i - offset d_i and $\Delta \theta_i$ - offset θ_i . In this way we received δA_i - matrix of the errors A_i :

$$\delta A_i = \begin{bmatrix} 0 & -\delta z_i^A & -\delta y_i^A & -\delta x_i^A \\ \delta z_i^A & 0 & -\delta x_i^A & -\delta y_i^A \\ \delta z_i^A & \delta x_i^A & 0 & \delta z_i^A \\ 0 & 0 & 0 & 0 \end{bmatrix}. \quad (2.4)$$

In this matrix dx_i^A, dy_i^A, dz_i^A - positioning errors during transition from $i-1$ to i , and $\delta x_i^A, \delta y_i^A, \delta z_i^A$ - angle errors. With these errors in mind, the transformation model between the working and world coordinate system is:

$$T_n + dT_n = (A_1 + dA_1)(A_2 + dA_2)(A_3 + dA_3) \dots (A_n + dA_n) = \prod_{i=1}^n (A_i + dA_i) \quad (2.5)$$

If we neglect the high-order differential term, we get:

$$dT_n = T_n \sum_{i=1}^n (U_{i+1}^1 \delta A_i U_{i+1}^1) = T_n \delta T_n \quad (2.6)$$

where δT_n - matrix of the errors, T_n and $U_i^1 = A_1 A_2 A_3 \dots A_n$. According to differential kinematics:

$$\delta T_n = \sum_{i=1}^n (U_{i+1}^1 \delta A_i U_{i+1}^1) = \begin{bmatrix} 0 & -\delta z_n & \delta y_n & dx_n \\ \delta z_n & 0 & -\delta x_n & dy_n \\ -\delta y_n & \delta x_n & 0 & dz_n \\ 0 & 0 & 0 & 0 \end{bmatrix}. \quad (2.7)$$

where dx_n, dy_n, dz_n - errors in the position of the coordinate system $\{n\}$ with respect to the coordinate system $\{0\}$, and $\delta x_n, \delta y_n, \delta z_n$ - angle errors. In this way, the final expressions for positioning errors are:

$$d_n = [dx_n, dy_n, dz_n]^T \quad (2.8)$$

$$\delta_n = [\delta x_n, \delta y_n, \delta z_n]^T \quad (2.9)$$

The working coordinate system of the robot is tied to the base of the robot, relative to which the reference position of the entire mechanical unit is set. The measurement of the positioning accuracy of the robots was carried out using a laser tracker, with the measuring marks installed in several places on the measuring unit attached to the flange of the robot's wrist.

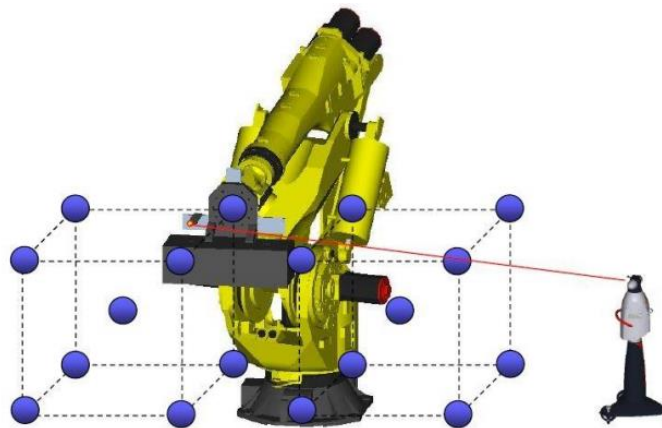


Figure 2.30 Measurement scheme of the absolute positioning accuracy of the mechanical unit of an industrial robot Fanuc

We used the Fanuc R-2000iC / 165F robot as a test robot. Robots of this type are among the most demanded in the world. With high productivity and lifting capacity ranging from 100 to 300 kilograms, these robots are a versatile solution for a huge range of tasks. Each of these robots contains a six-axis mechanical unit and a control cabinet. Geometric errors of the Fanuc R-2000iC / 165F robot are presented in the table below:

Table 2.1 Geometric errors of the Fanuc R-2000iC / 165F robot

Axis	Δa (mm)	Δd (mm)	$\Delta \alpha$ (radians)	$\Delta \theta$ (radians)
1	0.25	-0.78	$1.61 \cdot 10^{-5}$	$3.43 \cdot 10^{-4}$
2	-0.22	$0.05 \cdot 10^{-4}$	$1.94 \cdot 10^{-5}$	$-1.07 \cdot 10^{-3}$
3	-0.7	$-0.03 \cdot 10^{-4}$	$-0.5 \cdot 10^{-4}$	$4.45 \cdot 10^{-4}$
4	0.18	-0.13	$-6.89 \cdot 10^{-5}$	$1.01 \cdot 10^{-5}$
5	$1.21 \cdot 10^{-2}$	$-0.23 \cdot 10^{-5}$	$3.2 \cdot 10^{-4}$	$-3.34 \cdot 10^{-4}$
6	$-2 \cdot 10^{-5}$	0.05	$-7.34 \cdot 10^{-4}$	$4.65 \cdot 10^{-6}$

2.4 Conclusions

After analyzing the methods proposed by the Roboguide software, it was decided to use the Roboguide Simulation Programs and editor. This application offers us an intuitive and very complete editor, which allows programming of programs through a graphical environment, in active experiment mode. Also, the analysis of the positioning accuracy of the industrial robot Fanuc has shown that the selected robot will have sufficient accuracy.

The great advantages of this editor are that it allows creating robotic programs in a more comfortable way than in a TPE terminal and offers the great advantage of evaluating the cycles of work cell processes “offline”. By using the animations generated by the simulation program, controlling the various factors that can interfere with our simulation without having to suffer them in a real environment (incorrect or impossible trajectories, collisions, points out of reach) we can track and correct in real time all movements of the robot, this is fully consistent with the purpose of this work.

3 ROBOTIC SYSTEM SIMULATION

3.1 Workcell design

To carry out the process design we will take into account the specifications defined in section 1.5, composing a typical work cell, with characteristics of an industrial environment.

Several zones are defined within the system, through which the manipulated objects are going to be transferred according to the state of the robotic process. We therefore denote four differentiated areas that are:

- Position of the robot. The coordinate origin for the robotic system will be defined in this position.
- The box storage area. Boxes will be supplied through an automatic system in a storage, and by means of the manipulator action, their positions will be interleaved in the three levels as a FIFO queue.
- The storage area for part 1. Parts will be provided through an automatic system in tables and will be removed once used, through the same automatic system.
- The work area. A certain job will be carried out with the objects transferred to the area by a manipulator robot.

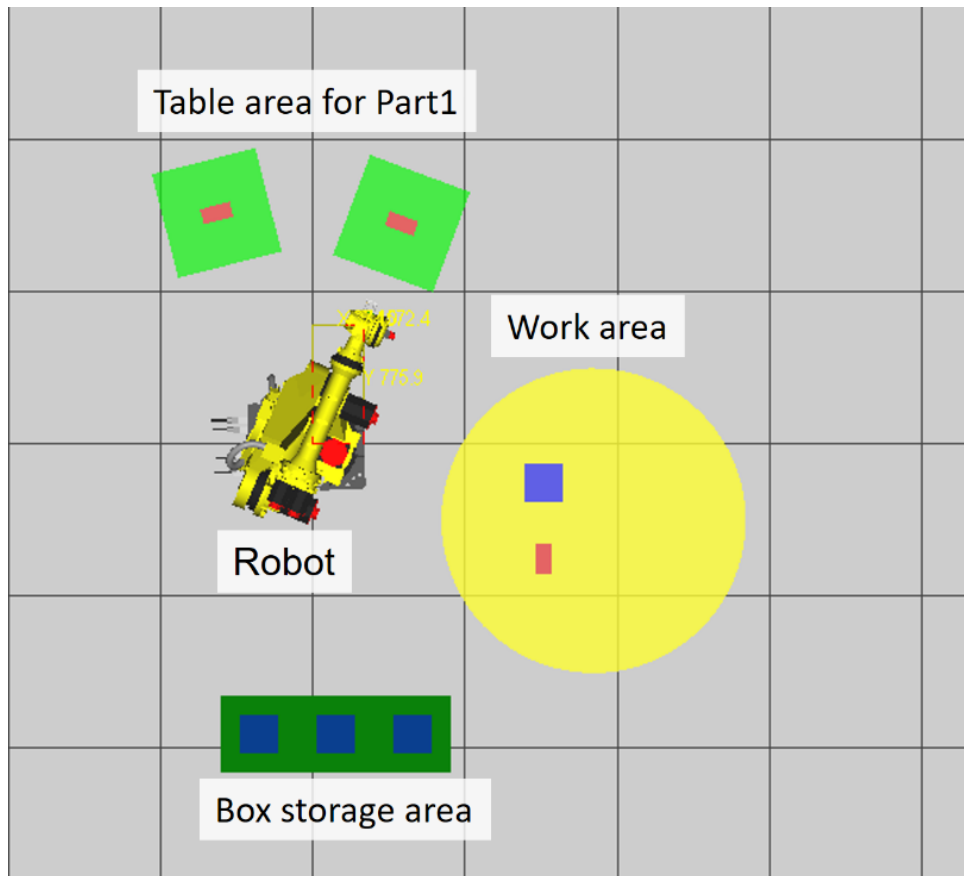


Figure 2.61 Design of the robotic system work cell

3.2 Automatic process diagram. Petri net

The robotic process state diagram is made up of four available actions and seven supported positions in the “world” of the robotic manipulator.

The actions of the process are:

- Capture of "Box";
- Capture of "Part1";
- Deposit of "Box";
- Deposit of "Part1".

The positions allowed in the manipulator environment are:

- Storage1;
- Storage 2;
- Storage3;
- Base (Work area);
- Table 1;

- Table 2.

The process itself will consist of the following steps in order of execution:

- Initial position.*
- Positioning in "Storage 1".
- Capture of "Box".
- Positioning in "Base".
- Deposit of "Box".
- Positioning in "Storage 2".
- Capture of "Box".
- Positioning in "Storage 1".
- Deposit of "Box".
- Positioning in "Storage 3".
- Capture of "Box".
- Positioning in "Storage 2".
- Deposit of "Box".
- Positioning in "Table2"
- Capture of "Part1".
- Positioning in "Base".
- Deposit of "Part1".
- Positioning in "Table1".
- Capture of "Part1".
- Positioning in "Table2"
- Deposit of "Part1".
- Waiting position. *
- Positioning in "Base".
- Capture of "Box".
- Positioning in "Storage 3".
- Deposit of "Box".
- Positioning in "Base".
- Capture of "Part1".

- Positioning in "Table1".
- Deposit of "Part1".
- Final position. *

The positions marked with an asterisk indicate that the point of that position exactly coincides in the defined space (start, wait and end).

Following the execution flow, we can compose the graph of the Petri net, which is divided into the four possible states (actions for capturing and depositing the objects) and the 7 transactions (positioning in the different manipulation spaces). Below is the mentioned industrial process through the design of the Petri net:

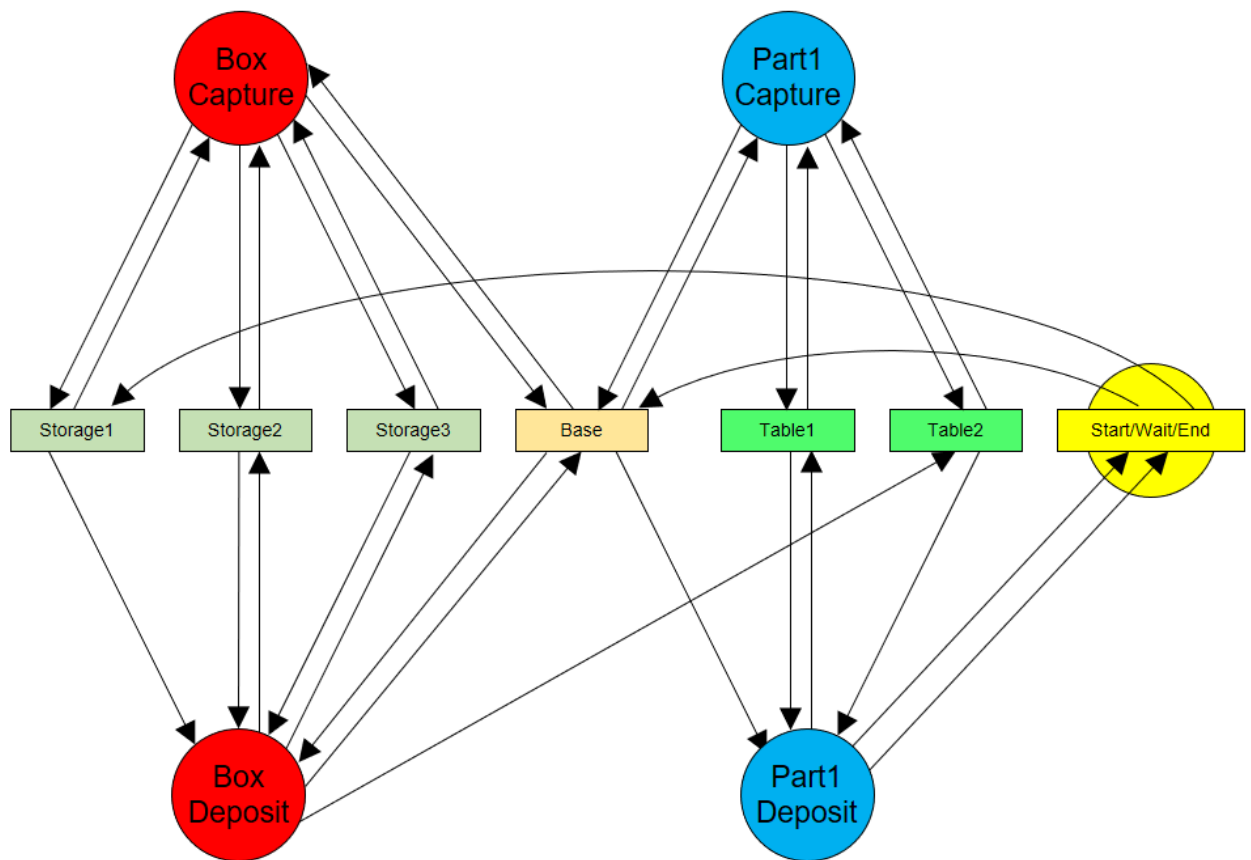


Figure 2.61 Petri net diagram of the robotic system

3.3 Presentation of the manipulator

The Fanuc R-2000iC/165F manipulator has been chosen to carry out the process.

Fanuc's R-2000iC robot series is the latest generation of heavy duty robots with high industrial performance. We can highlight the performance, safety and handling of this

model, ensuring the highest reliability. The 165F model is the standard model, of a wide range with minor modifications according to the needs of the processes to be carried out.

Equipped with 6 axes, with a maximum load of 165 kg, it offers a maximum reach 2655 mm, with a repeatability of ± 0.05 (based on ISO9283). The 165F model is the standard model of a wide range with minor modifications according to the needs of the processes to be carried out.

Thanks to the payload capacity of this robot model and the degrees of freedom provided by the 6 axes it has, we have a manipulator that is perfectly adapted to the process we want to implement. These characteristics make this model a very versatile robot to reach a large number of trajectories and provides the system with a great capacity to be able to be expanded without significant repercussions.



Figure 2.61 Robot FANUC R2000iC Series

The following images show the robot drawings with the general measurements specified by different views and details of the R2000iC / 165F robot.

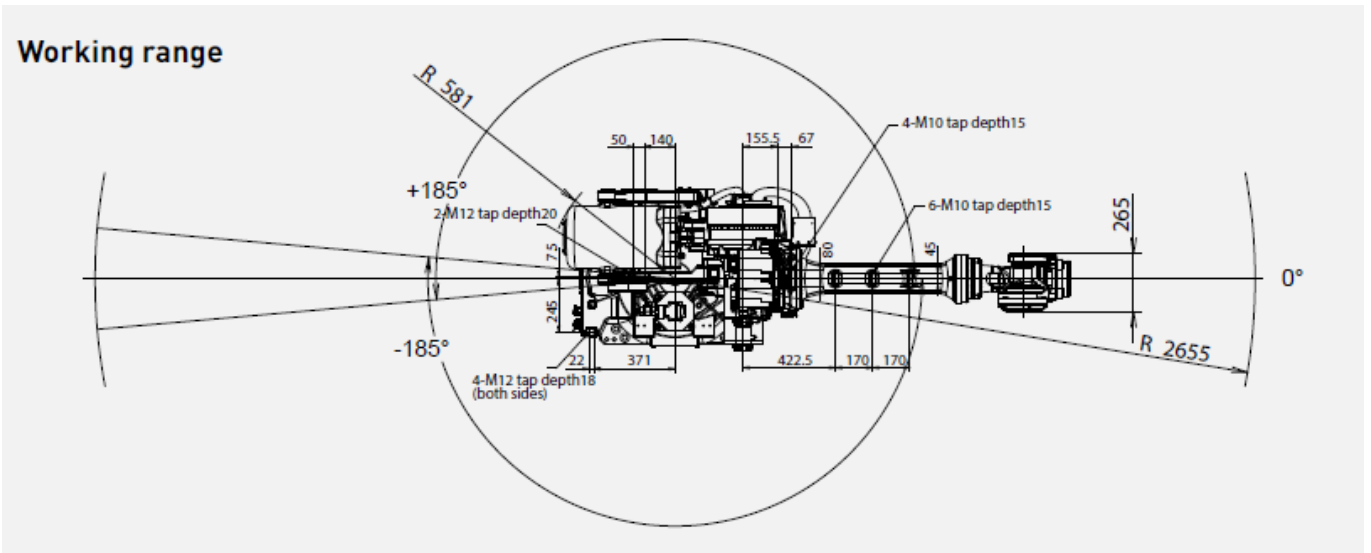


Figure 2.61 Working range of the robot FANUC R2000iC Series

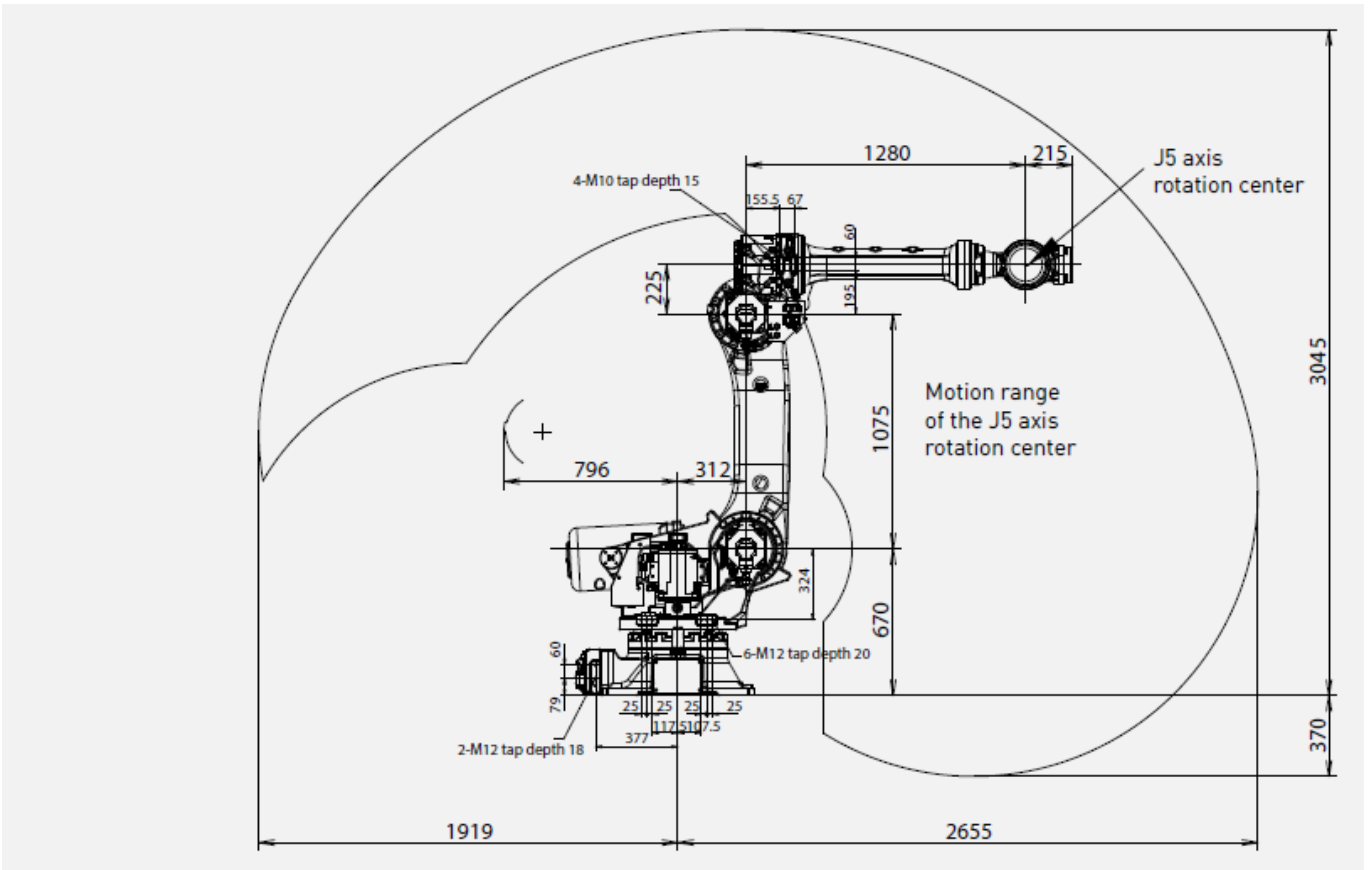


Figure 2.61 Measurements of the robot FANUC R2000iC Series

3.4 Automatic system implementation

Once the industrial process to be programmed has been specified, we will develop the FANUC Robotics Roboguide HandlingPRO virtual software simulation environment.

For the development of the simulation project we will use the evaluation version 9 (Rev. H) for Windows 10.



Figure 2.61 Information window of Roboguide HandlingPRO virtual software

FANUC Robotics Roboguide HandlingPRO virtual software simulation environment allows you to develop software for each specific situation with the possibility of its future implementation in the physical control system presented in the figure below. An important advantage is the ability to set up and test in a simulation environment, which eliminates the risks of working with physical equipment during development and testing.

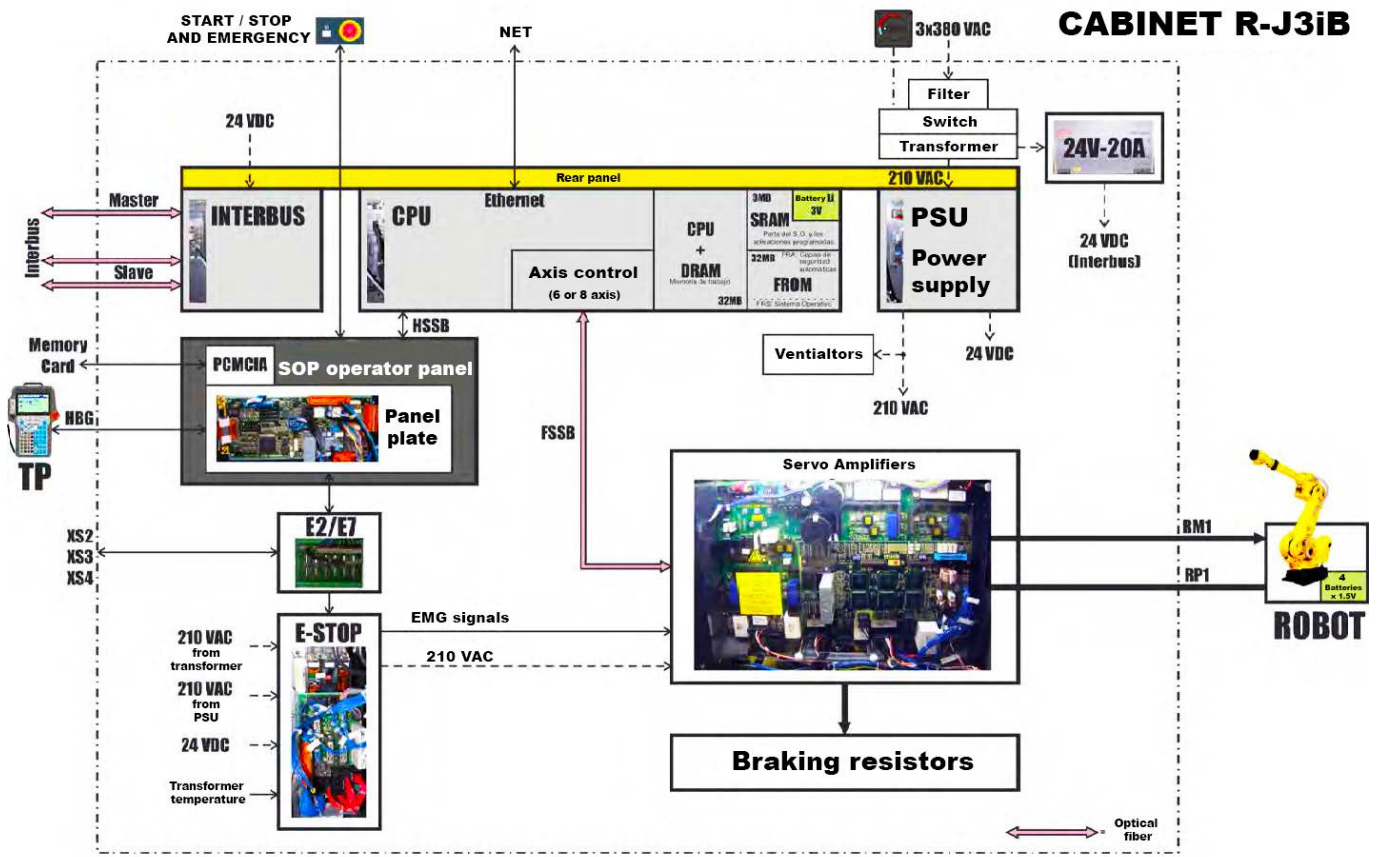


Figure 2.61 Fanuc robot’s control system

3.5 Definition and configuration of the robot and the controller

To start using the application, once installed in the operating system we must create a new work cell. We can use the File> New Cell menu, or use the “navigator” menu, which will guide us step by step to start our 3D simulation.

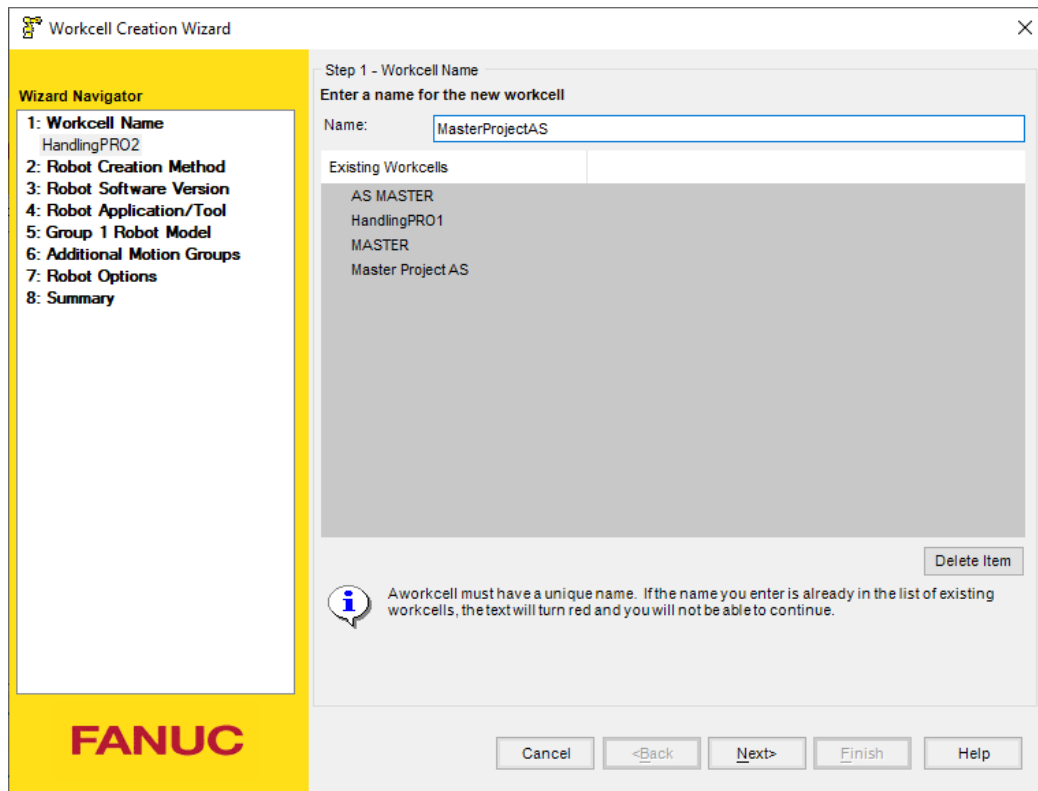


Figure 3.1 “Step 1” work cell creation wizard

From this screen we have the option of being able to eliminate the work cells that are not necessary, once their programming and simulation have been carried out. We will give a name to the "WorkCell" and we will go to the next menu window. On this screen we will be able to choose between: creating a new robot using the HandlingPRO default configuration; create a robot with the last used configuration; create a robot from a backup file or make a copy of an existing job.

We will use the first option, since it allows us to modify the configuration options a posteriori if necessary for the simulation. The following figure shows the wizard screen referring to this point:

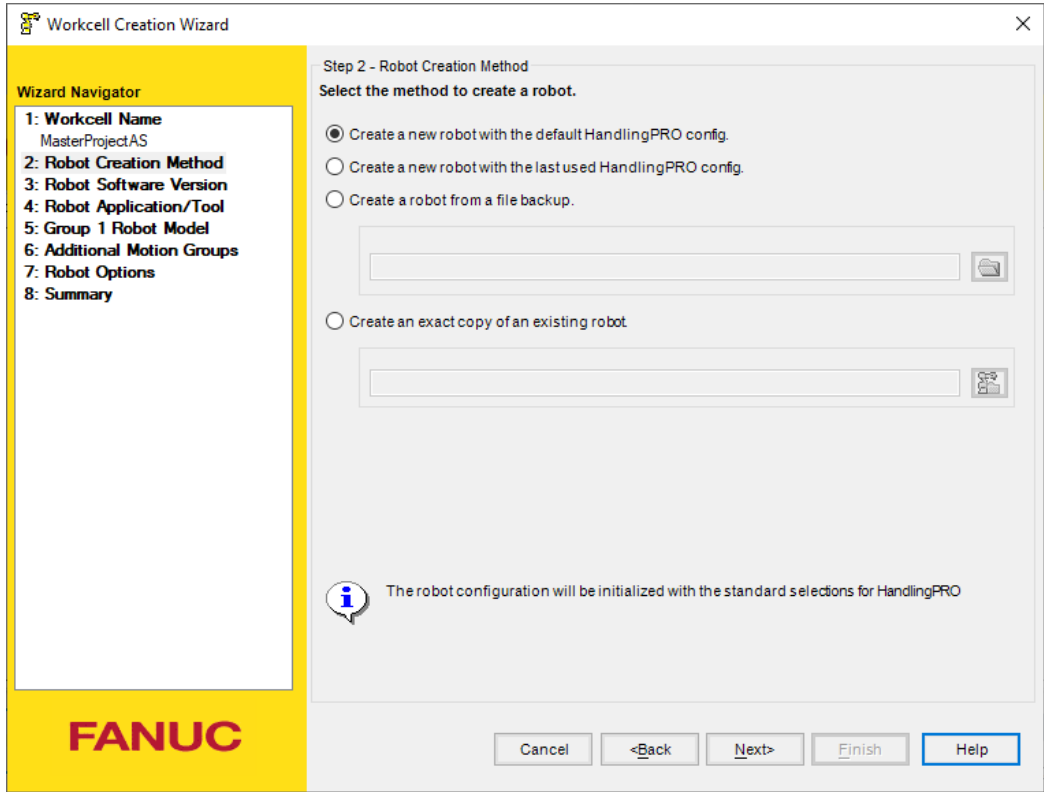


Figure 3.2 “Step 2” work cell creation wizard

In the third screen of the wizard it is necessary to choose the version of the robot controller software offered by the application. We have different versions developed by the manufacturer FANUC. When the application is installed, the software versions of the controller to be installed are requested, of which we have configured versions 9.10, 9.0, 8.30 and 8.13. To carry out our simulation we have chosen version 8.30.

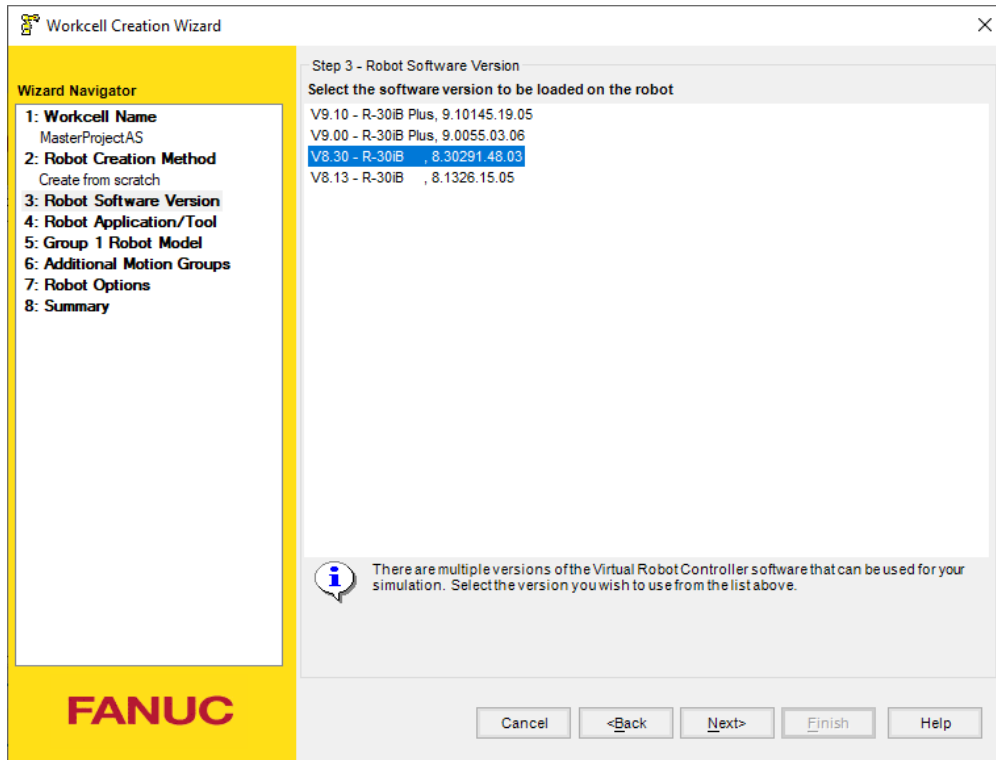


Figure 3.3 “Step 3” work cell creation wizard

In the screen shown below we can choose the application package that we are going to use. In this case it is the “HandlingTool” package (H552), which will allow us to simulate a “pick and place” process. “Set Eoat later” will allow us to choose robot’s tool later, in the process of robot’s configuration.

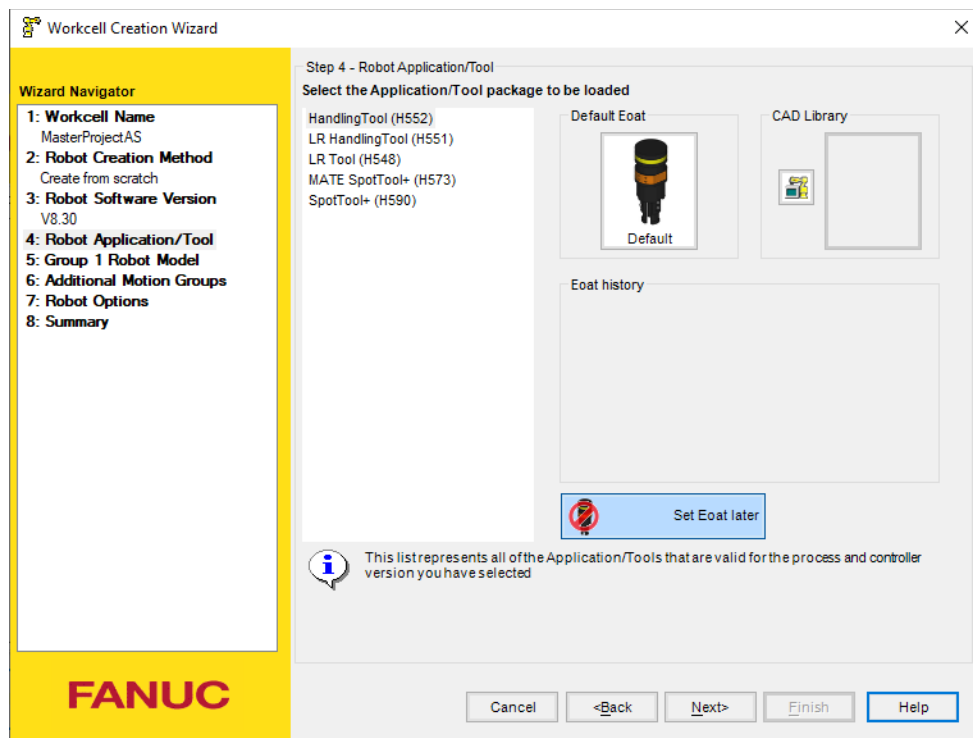


Figure 3.4 “Step 4” work cell creation wizard

In step five of the wizard we can choose the robot that we are going to use to carry out the process applied to the industrial plant. After a preliminary study of the physical needs, we will choose the model that best suits our requirements.

We have chosen the R2000iC/165F robot model for having a good load capacity and being one of the most versatile in terms of mobility.

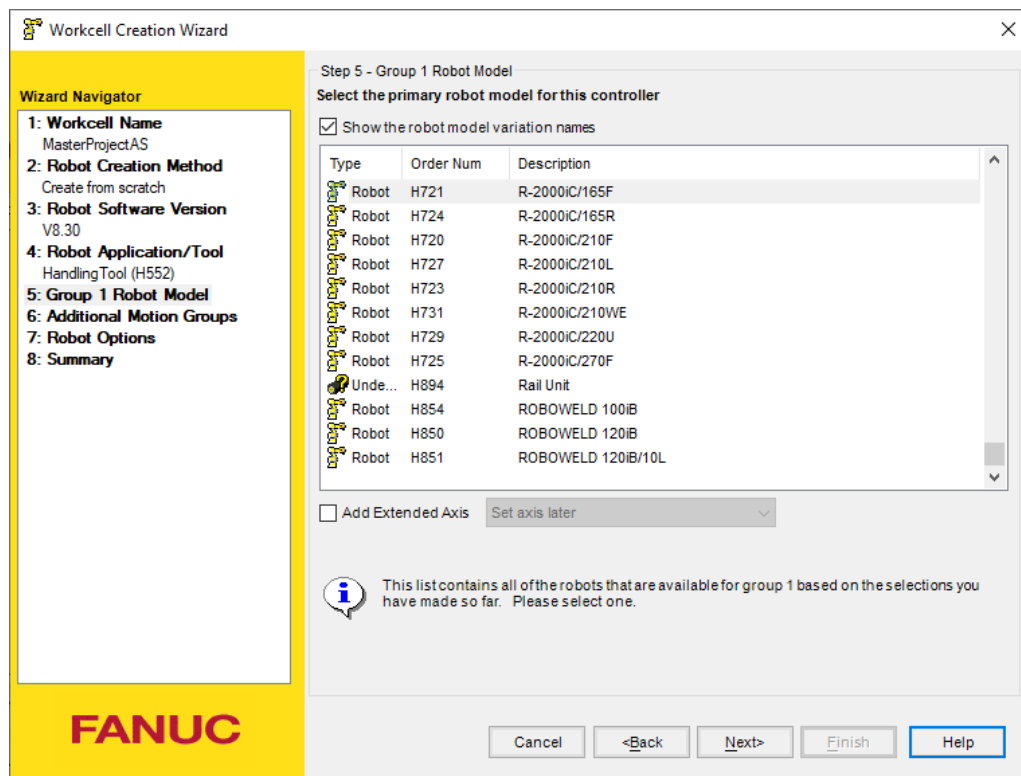


Figure 3.5 “Step 5” work cell creation wizard

In the next step, the application allows us to define "movement groups", adding more robots to the cell in addition to the one already selected. It even allows us to create several movement groups with different robots in each of them.

This option can be reconfigured once in the application and according to the needs, so we have selected a single robot to be able to add it later from the application.

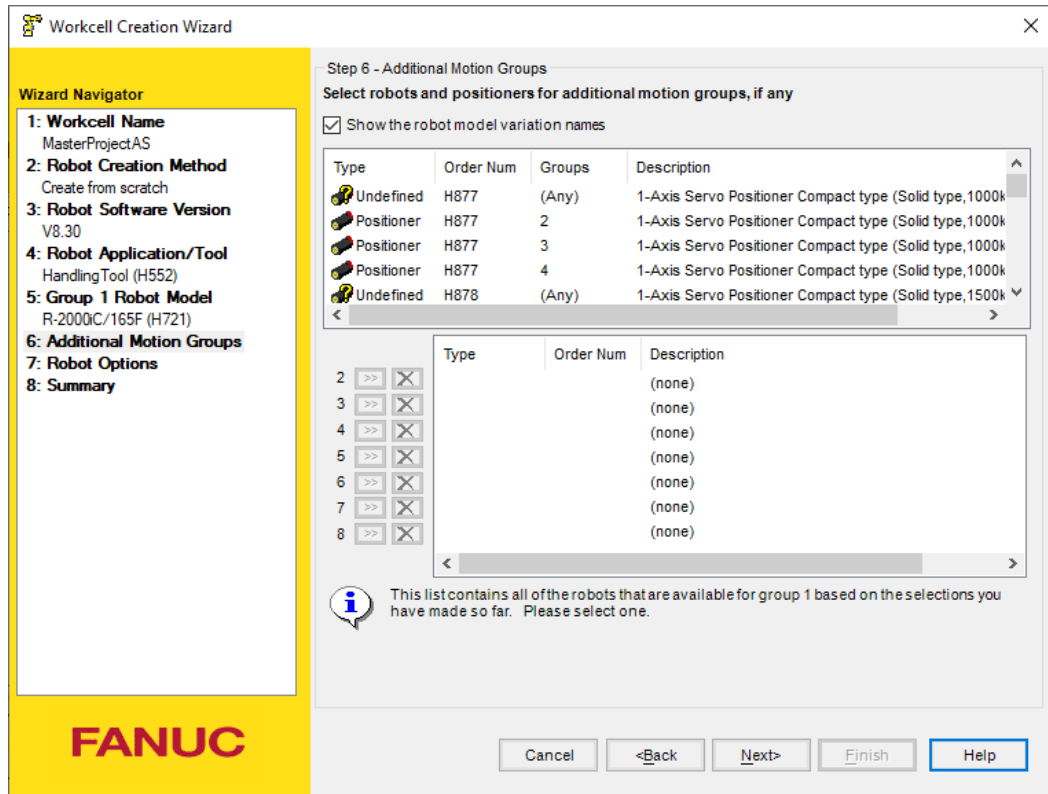


Figure 3.6 “Step 6” work cell creation wizard

Next we will define the specific FANUC software modules that we will load directly onto the robot controller. These are functionalities that this module offers, such as: vision systems integrated in the 2D or 3D manipulator, various sensors, tools for collision treatment, measurement systems, special machines.

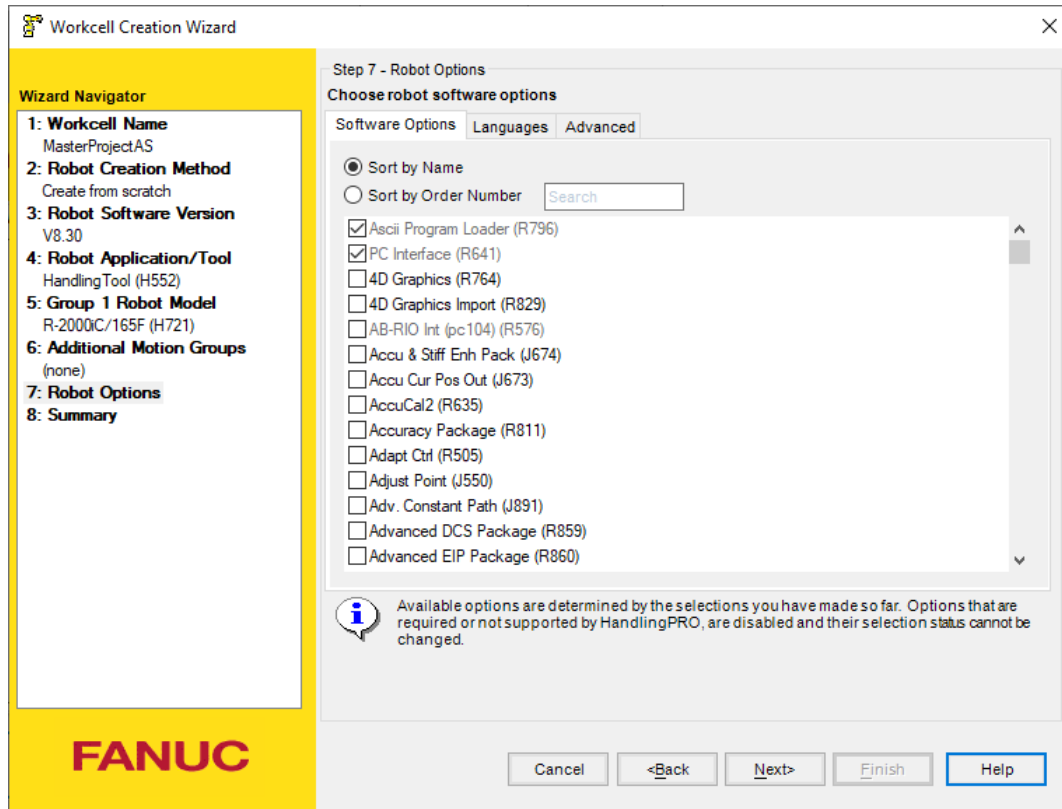


Figure 3.7 “Step 7” work cell creation wizard

In the last screen of the wizard we will obtain a summary of the options determined in the previous stages. Once the different parameters have been reviewed, we finish creating the robot in the simulation software.

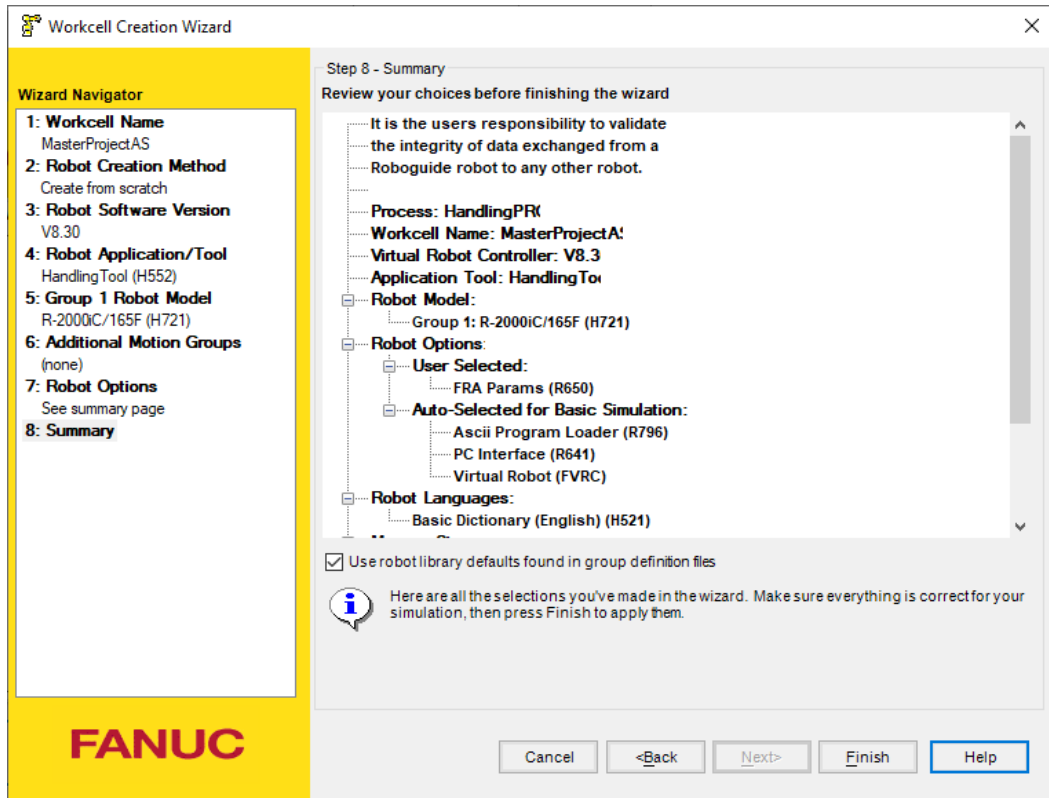


Figure 3.8 “Step 8” work cell creation wizard

Finally, we observe that the simulation software initializes the virtual controller of the robot as it has been configured. It will define the initial values of the internal variables of the system, the previously loaded functionalities, the virtual "teach pendant". Once the initialization process is finished, the configured options are available in the virtual 3D work environment.

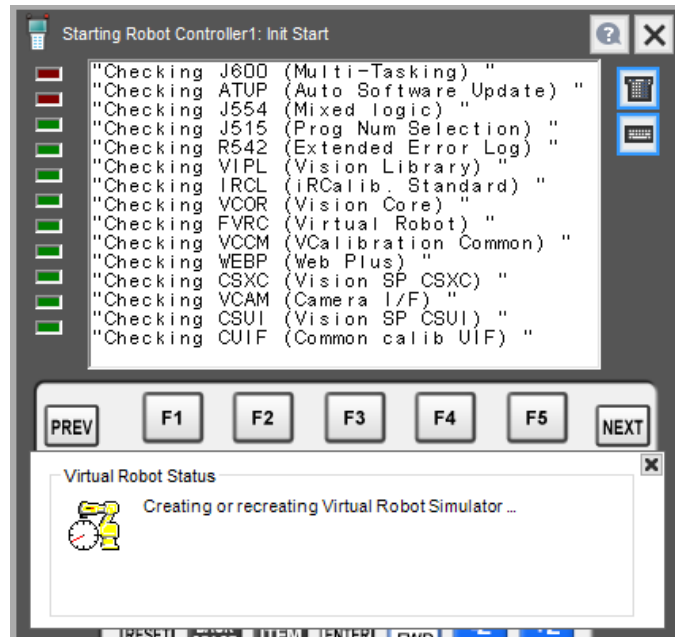


Figure 3.9 Initialization of the robot controller

After having created the robot object, we will start working with it by checking its properties settings, using the menu accessible with the right mouse button in the “Cell Browser” window.

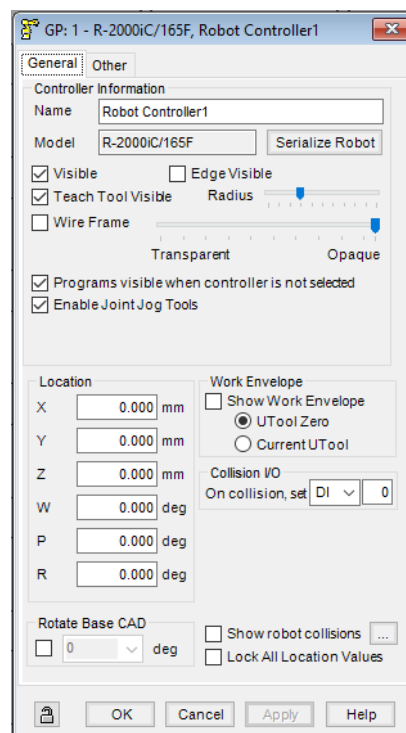


Figure 3.10 Robot configuration properties menu

As we can see in the previous figure, the most important properties of the robot that we can edit are: the name of the robot, the visibility options during the “teach run”, the transparency or opacity, the visualization of the collisions and the position in the environment of the cell.

The steps defined below will allow configuring the different objects that will make up the work environment for the simulation, following the robotic cell design scheme.

3.6 Definition of manipulated objects (PARTS)

Once the robot is configured, we will define the properties of the parts that are going to be manipulated during the simulation. To carry out this action we will use the main menu “Cell Browser” and on the drop-down “parts” we will select, with the right button of the mouse, the option “add a part”. The menu shows us different objects to load as manipulated. It also allows us to load a predefined part of the application, a custom part and different geometric bodies (box, cylinder and sphere).

In this case we have defined two parts by choosing the “box” form, which will serve as an example of a manipulated object.

Once the aforementioned parts have been created, we must configure them with the "Properties" option, accessing again with the right button from the "Cell Browser" menu. In the following figure we visualize the configuration screen of the “BOX” part.

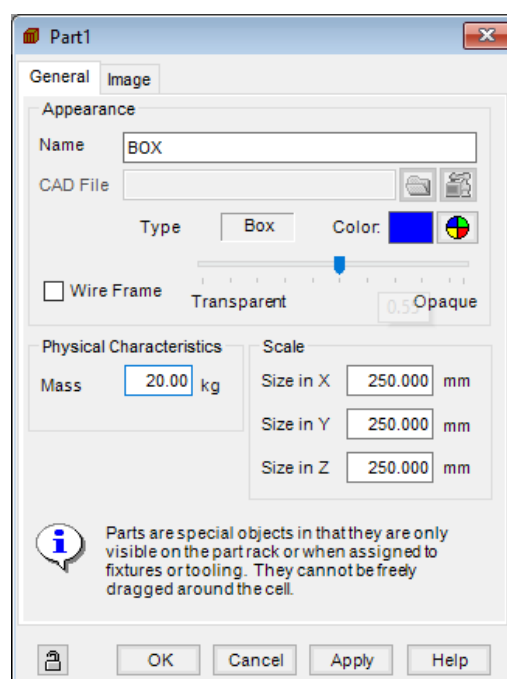


Figure 3.11 Properties menu of the “BOX” object

We check that this menu allows us to configure the name of the object, the CAD file in the event that we load a CAD image file, the type of object (box in this case), color, opacity, mass and size defined in the x, y, z axes.

Similarly we define the second manipulated object called “PART1”, and we access to its properties window:

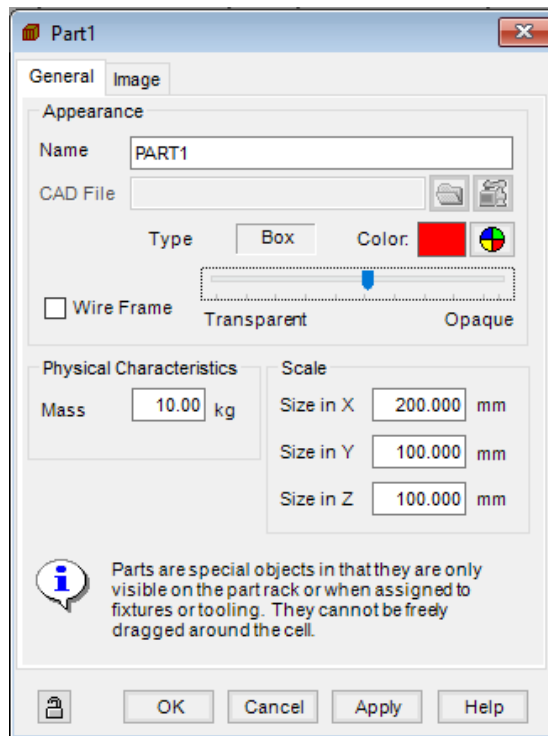


Figure 3.12 Properties menu of the “PART” object

Once the parts that will be manipulated in the simulation have been defined, the configuration of these objects in the “Cell Browser” will be as follows:

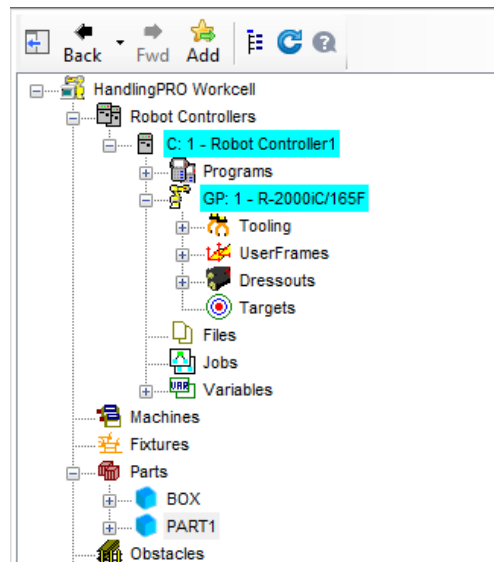


Figure 3.13 Definition of manipulated objects in the simulation (parts)

3.7 Tool definition and configuration (UTOOL)

In this section, a tool will be added to the robot's TCP to be able to manipulate the objects defined in the “parts” menu. To define an EOAT (End of Arm Tool) tool, access the “Cell Browser” menu, in the “tooling” option, we will define a new “UTOOL” for this.

When we open the “tooling” drop-down we check that the robot allows the definition of different “UTOOLS” (In the simulation software we have found eleven possible tools).

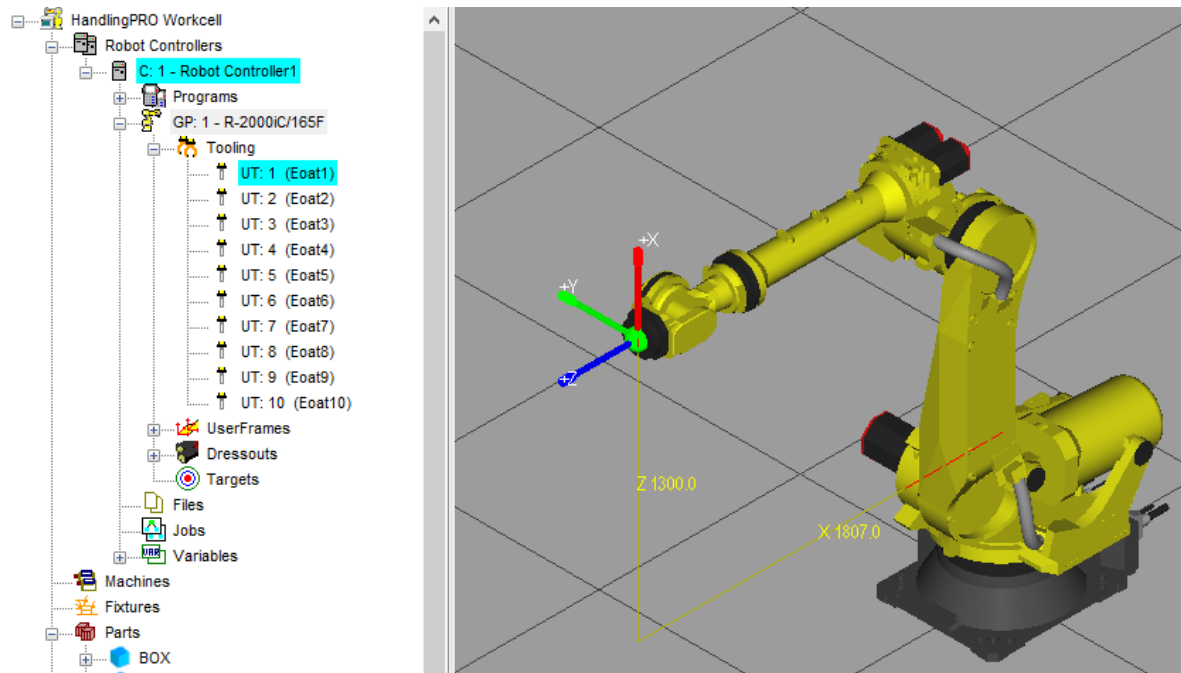


Figure 3.14 “Tooling” menu of HandlingPRO simulation software

The procedure to define a tool is carried out by choosing one of the "UTOOL" defined in the "Cell Browser". The right button of the mouse allows us to access the option "Eoat1 Properties".

Within the menu, in the "General" tab, we will name the tool to define, we will load the primary CAD file of the tool (since if it is a prehensile tool, we will have a secondary CAD file in the "closed" state), we will position the tool in TCP with the desired offset ("offset"), we will define the mass of the tool and the scale used in the graphic environment.

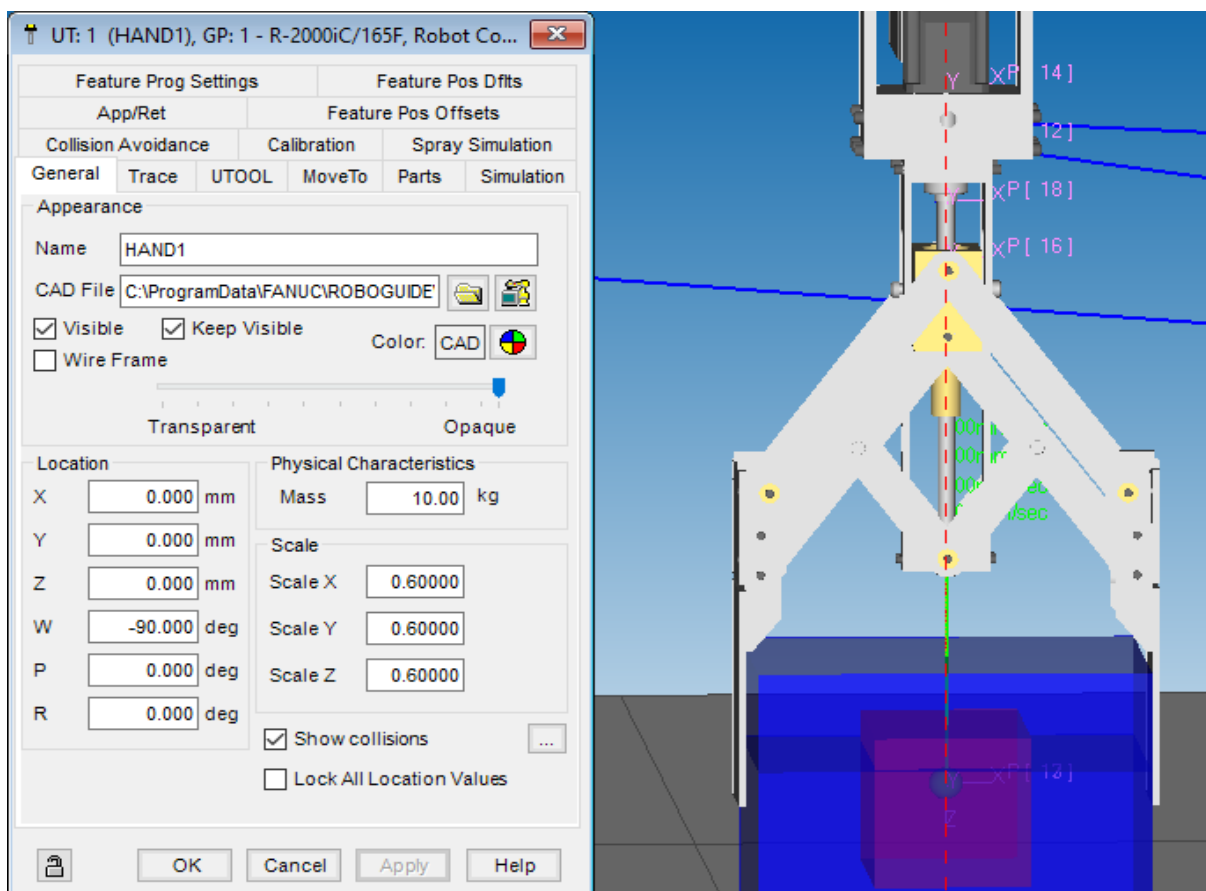


Figure 3.15 Definition of characteristics in the "General" tab

In the UTOOL tab we will define the actuation point of the gripper. Due to the displacement in the tool configuration, we will define the reference system of the actuation point to specify the movements of the robot so that the gripper can be properly positioned on the objects to be manipulated.

To carry out this operation, we will insert the coordinates directly into the reference system relative to the gripper or use the "Use Current Triad Location" functionality to establish the current point defined with the robot movement tool as the reference system.

In this configuration tab, optionally, we can also define the normal that the system will provide us by default to use the movement option "Move Robot Normal-to-surface" using the Ctrl-Shift-Click key combination.

In this case we will leave the value defined for the default normal on the -Z coordinate axis. The "current triad location" method is based on taking the TCP coordinates after positioning the robot with the "teach tool selection", equivalent to the "teach by nose" system of a real robotic system.

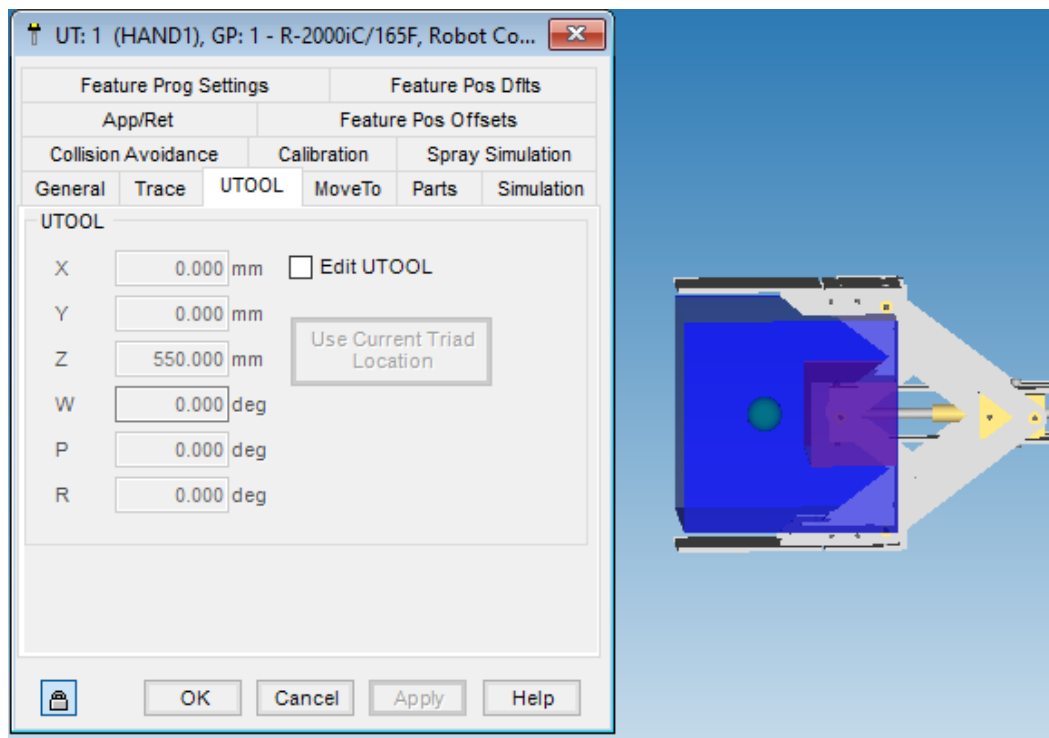


Figure 3.16 Definition of characteristics in the "UTOOL" tab

In the next tab we define the parts that this tool can manipulate. We have previously defined the parts in the "parts" menu to be able to assign them to the tool that we are defining and that they can be manipulated in the simulation.

For each of the defined parts we must define the "offset" when the tool is acting on it. In our case, as it is a part handling process, we have to define the "offset" of the part when it is being "grabbed" by the gripper that is going to rotate or move it, according to the movement defined in the program. .

The "offset" can be defined in the relative system x, y, z, in addition to the orientation with the angles w, p, r.

To edit the "offset", we must check the box "Edit part offset", choose one of the parts defined in the system through the drop-down menu, and then we can enter the values of the variables x , y , z , w , p , r .

We can also define if we want this part to be visible in the learning time "teach time" and / or in the simulation time "run time".

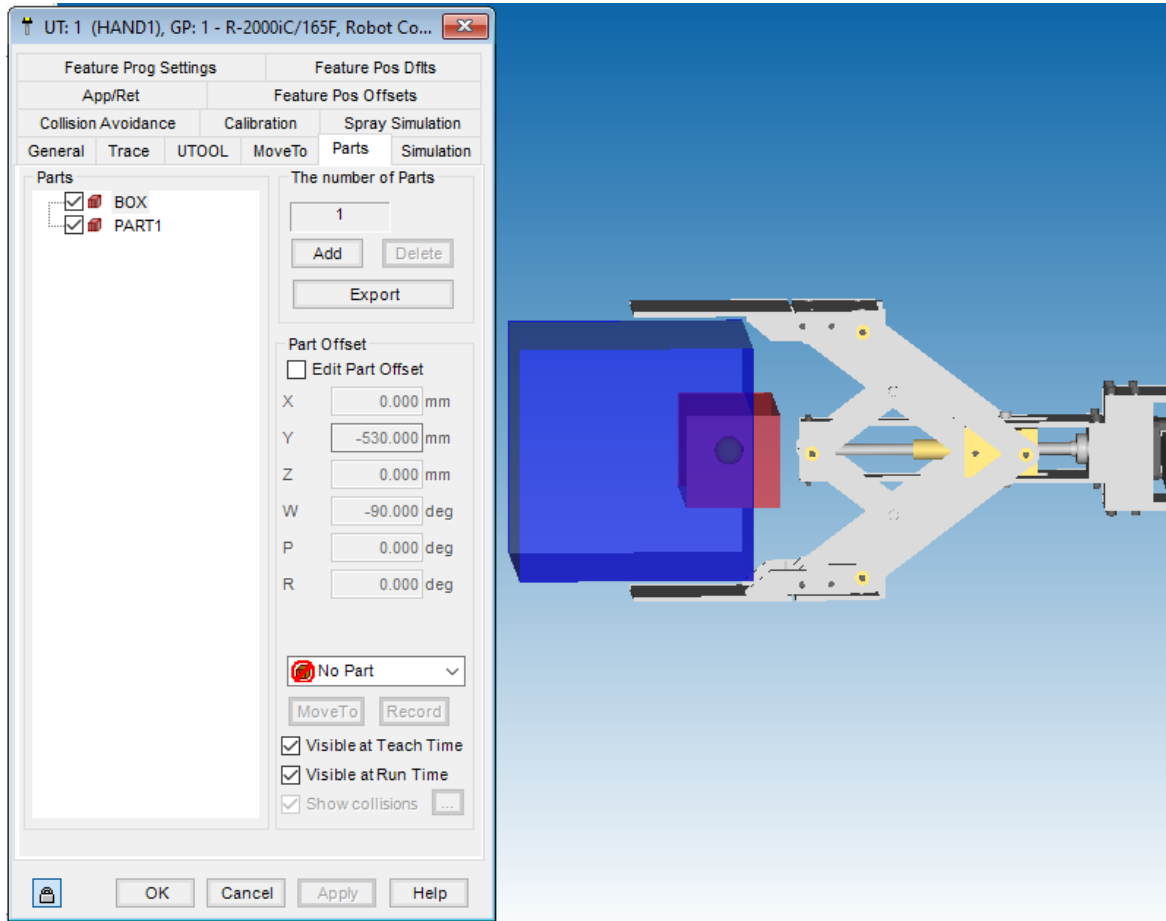


Figure 3.17 Definition of characteristics in the "parts" tab

Finally, we will define the characteristics of the "Simulation" tab, which will allow us to control some aspects of the tool during simulation time.

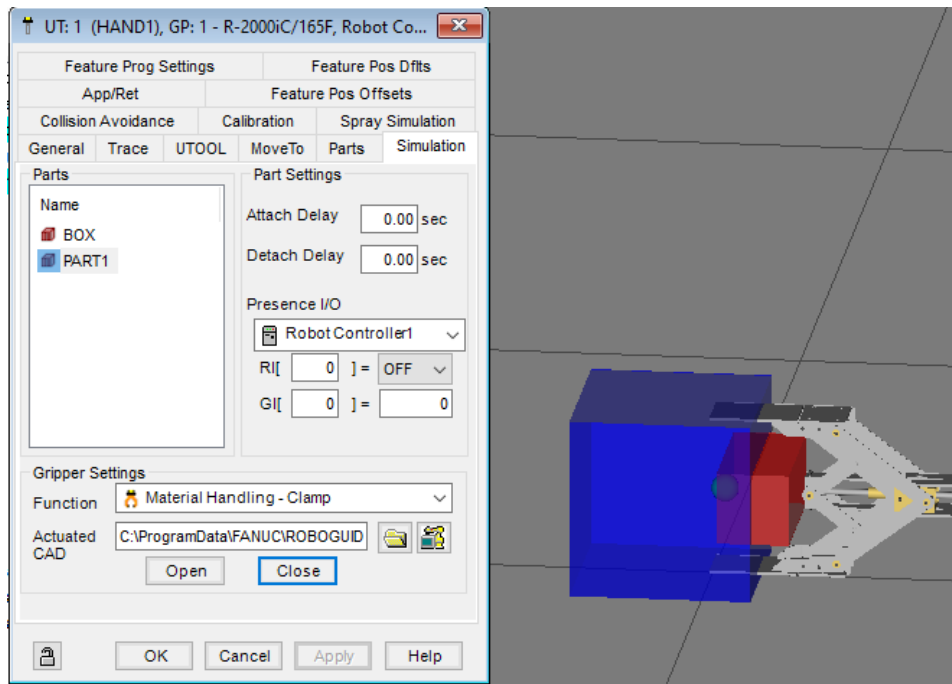


Figure 3.18 Definition of characteristics in the "Simulation" tab

In this tab we will define the default capture and deposit times of the different defined pieces. We will also configure the function of the tool used (in our case “Material Handling - Clamp”) and we will associate an “acted” CAD image, which will allow us to animate the tool when it changes to the secondary state in a manipulation of any of the parts or parts. In the simulation, as it is a gripper, the primary state would be the gripper open and the "actuated" state would correspond to the gripper in the closed state with the captured part.

3.8 Creation and configuration of fixed parts (FIXTURES)

The definition of the fixed parts of the system is essential for the manipulation of the different parts in the cell. In these fixed parts we will position the objects that will be used in the process.

Each of the pieces defined in the previous section must be linked to the fixed parts so that the software can recognize the situation of these objects. Ultimately, these fixed parts will be the objectives of the manipulation, that is, they will support the pieces that must be captured and they will be the supports whose upper surface will receive the manipulated pieces.

We will therefore define, with the following procedure, the fixed parts of the simulation following the design of the robotic system of the cell: we will access the configuration menu using the main window of the “Cell Browser” and we will choose the option “Add a Fixture”, using the right click on the “Fixtures” menu.

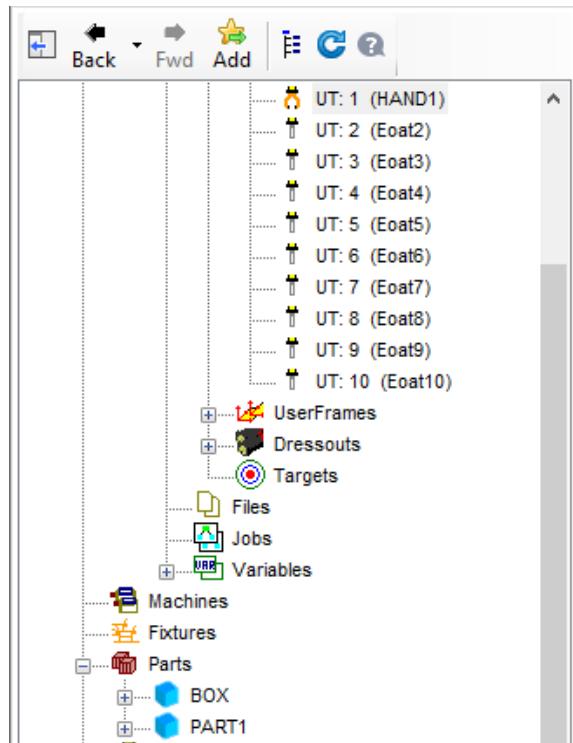


Figure 3.19 Creation of “fixtures” from the “Cell Browser” menu

We will select the geometric body "box" and configure it according to the images shown below:

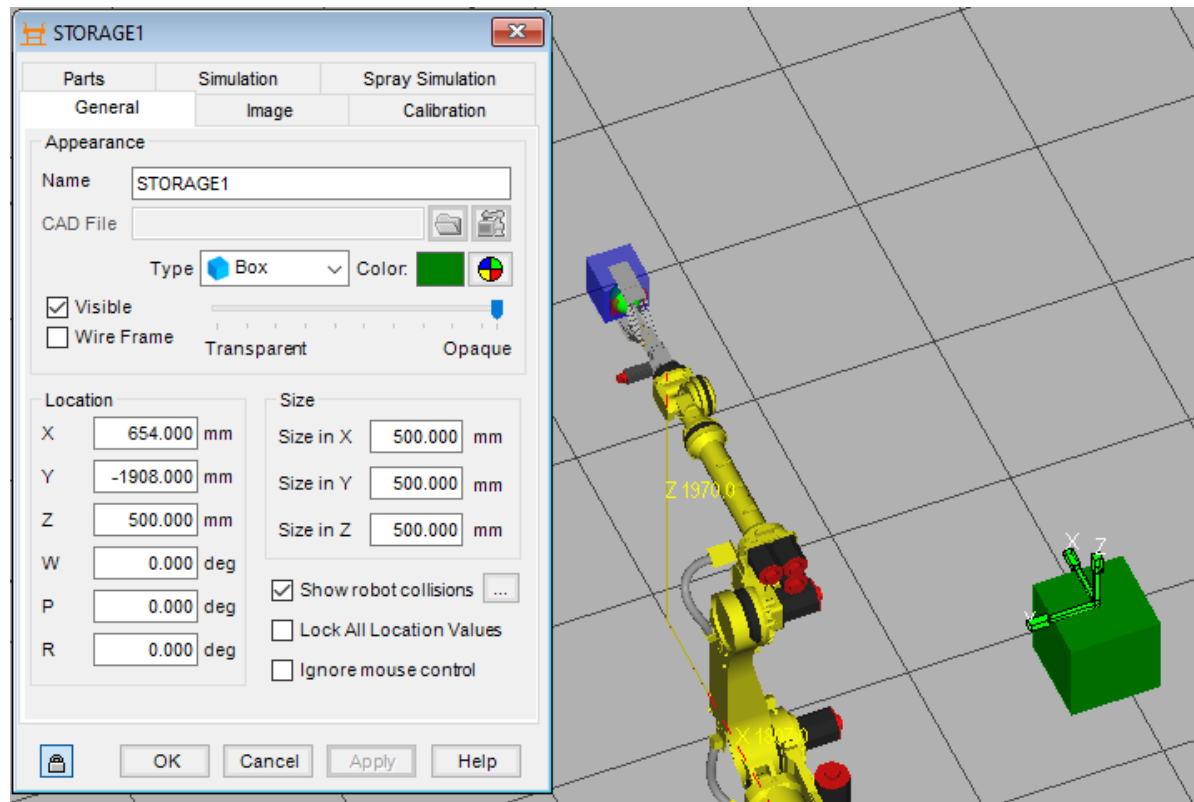


Figure 3.20 Configuration of the "General" tab of the fixed part "storage1"

We will access the "General" tab and the following actions will be carried out: we will name this "fixture" as "Support1", we will select it as "visible", we will define the visual aspects such as color, opacity or transparency, the scale used in the environment of the cell and the position in relation to the position of the robotic arm.

We will also configure that, in the event of a collision of the robot with this fixed part, the alarm is indicated in the "run time" period. For this we will use the tab "Show robot collisions".

In the next tab "Calibration" we can see all the calibration options offered by the simulation software. Roboguide is able to generate calibration programs automatically if connection with a real robot is available from the application.

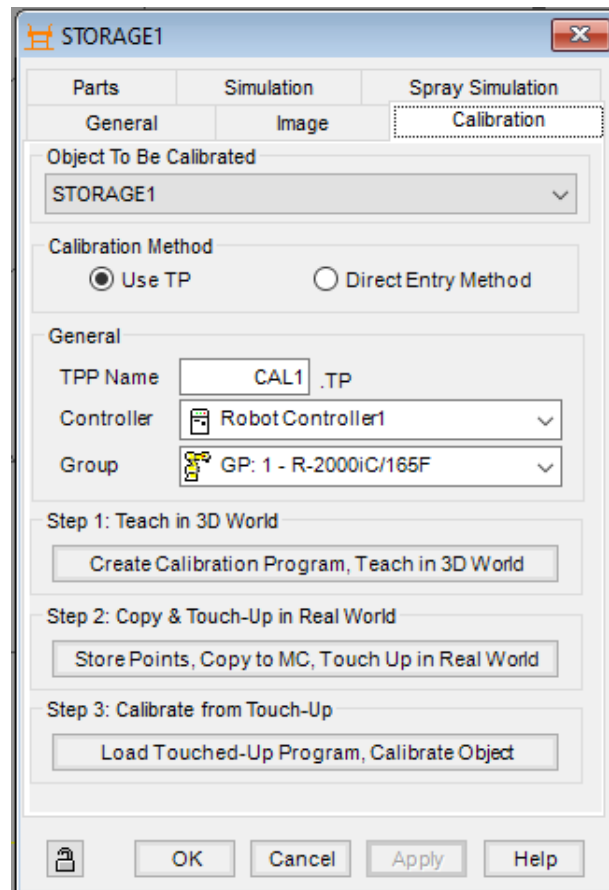


Figure 3.21 Configuration of the tab “Calibration” of the fixed part “Support1”

The "Parts" tab will allow us to define each of the parts associated with this fixed part. We will be able to select the object from the set of parts defined in the system and position it in the fixed part as specified in the cell design, thus adapting the future real environment with the simulation software.

To position the selected parts we will define the values x , y , z , w , p , r . We can also configure the visibility of these parts during the “teach time” and the “run time”.

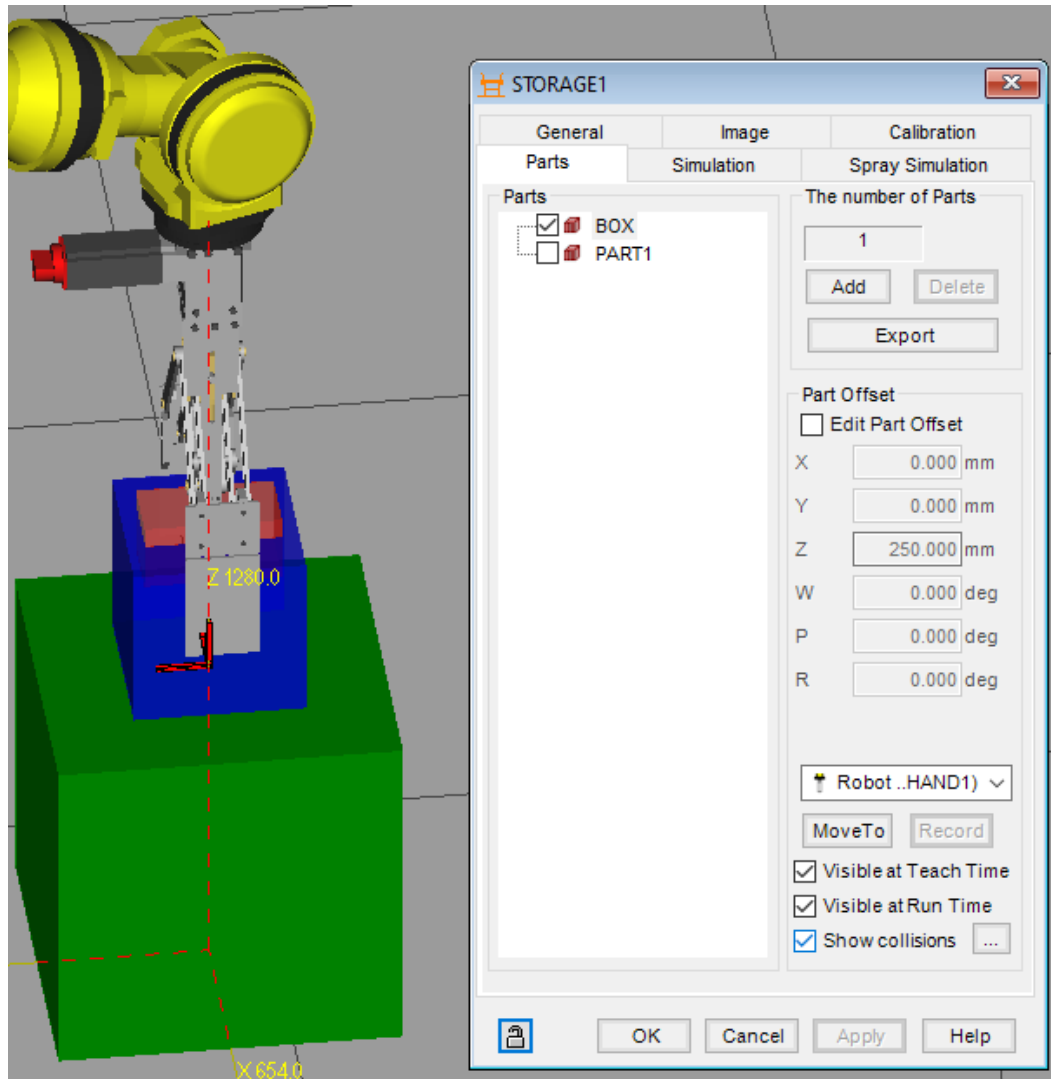


Figure 3.22 Configuration of the “Parts” tab of the fixed part “Storage1”

In the last tab "Simulation" we will define the options of the fixed part in the period of "run time". In this menu, only the parts selected in the “Parts” tab will appear and we can define for each of them whether they can be captured and/or positioned, as well as the delay times for these two actions during the simulation time.

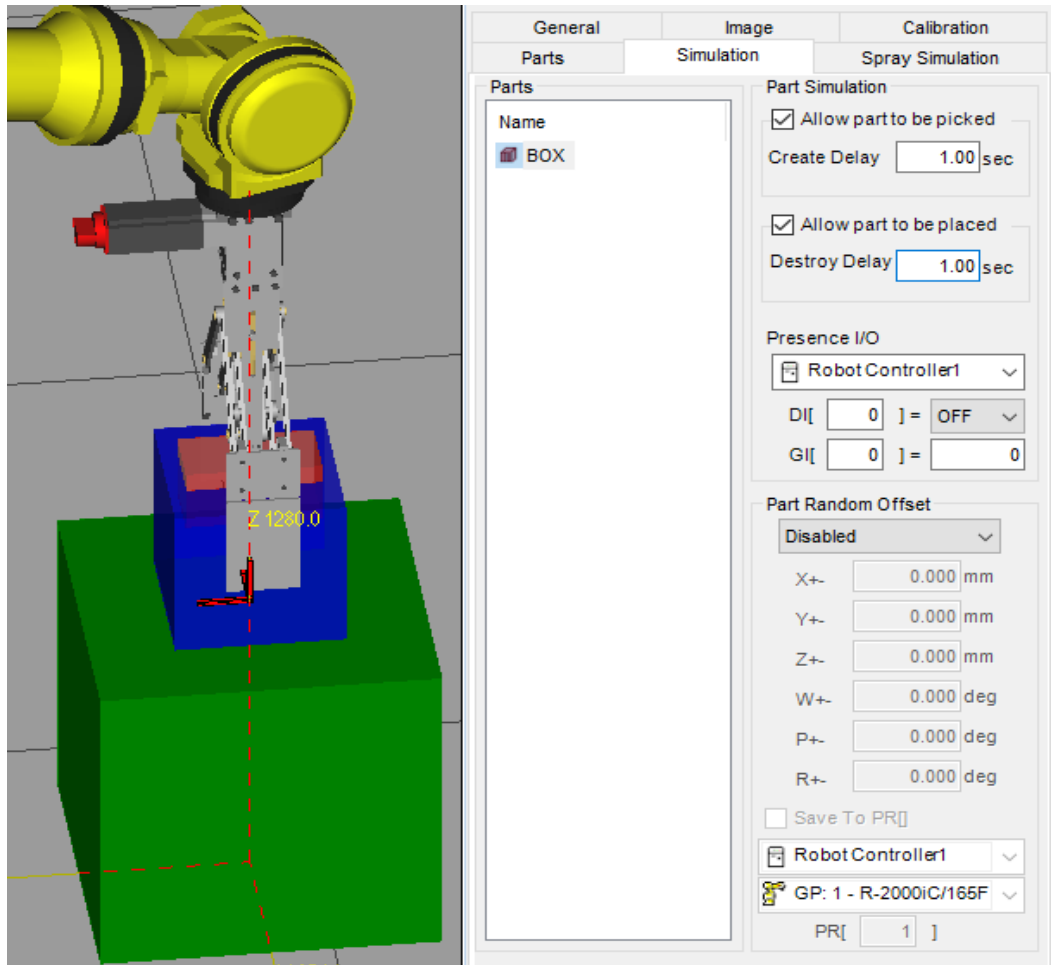


Figure 3.23 Configuration of the "Simulation" tab of the fixed part

Following the procedure described, we define two more fixed parts, called "Storage2" and "Storage3", which will be the same as the already defined "Storage1", except in height. We will define different values of the Z coordinate variable for each of the supports, following the scheme of the designed system.

In addition to these fixed parts labeled "StorageX", we will create a fixed part of the "Work Zone" called "Base". And two more fixed parts "Table1" and "Table2", equal to each other, that will serve as support for the "Part1" objects.

The following shows the configuration of this fixed part in the simulation software, having in each one of the images each tab of the properties menu:

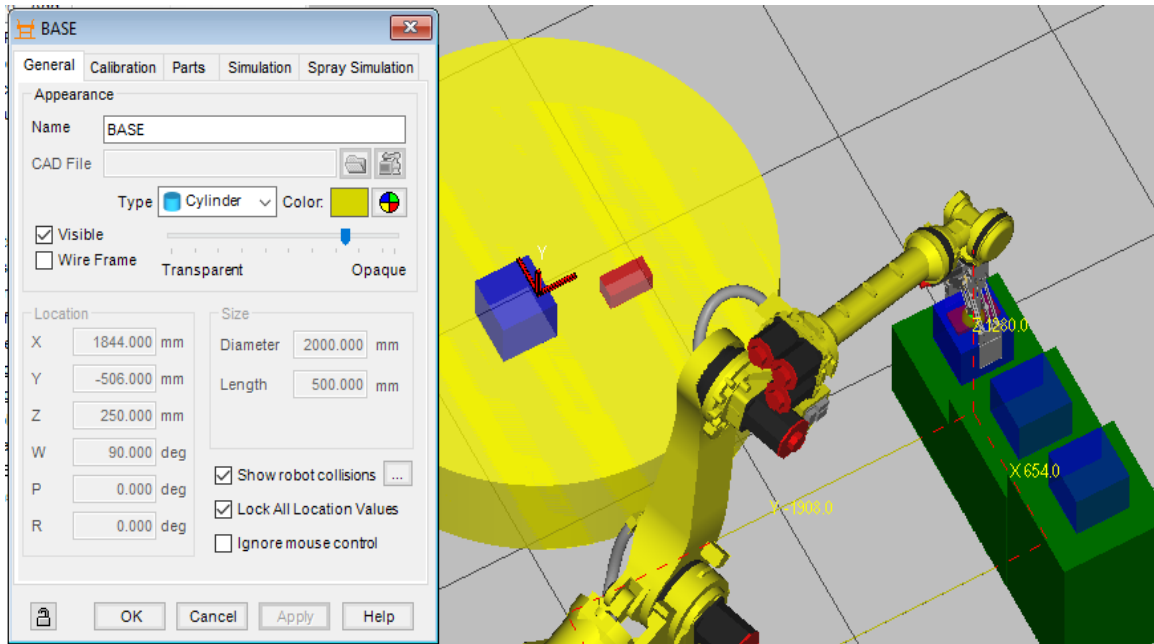


Figure 3.24 Configuration of the “Base” fixed part

Next step is position configuration of the parts “Part1” and “Box”.

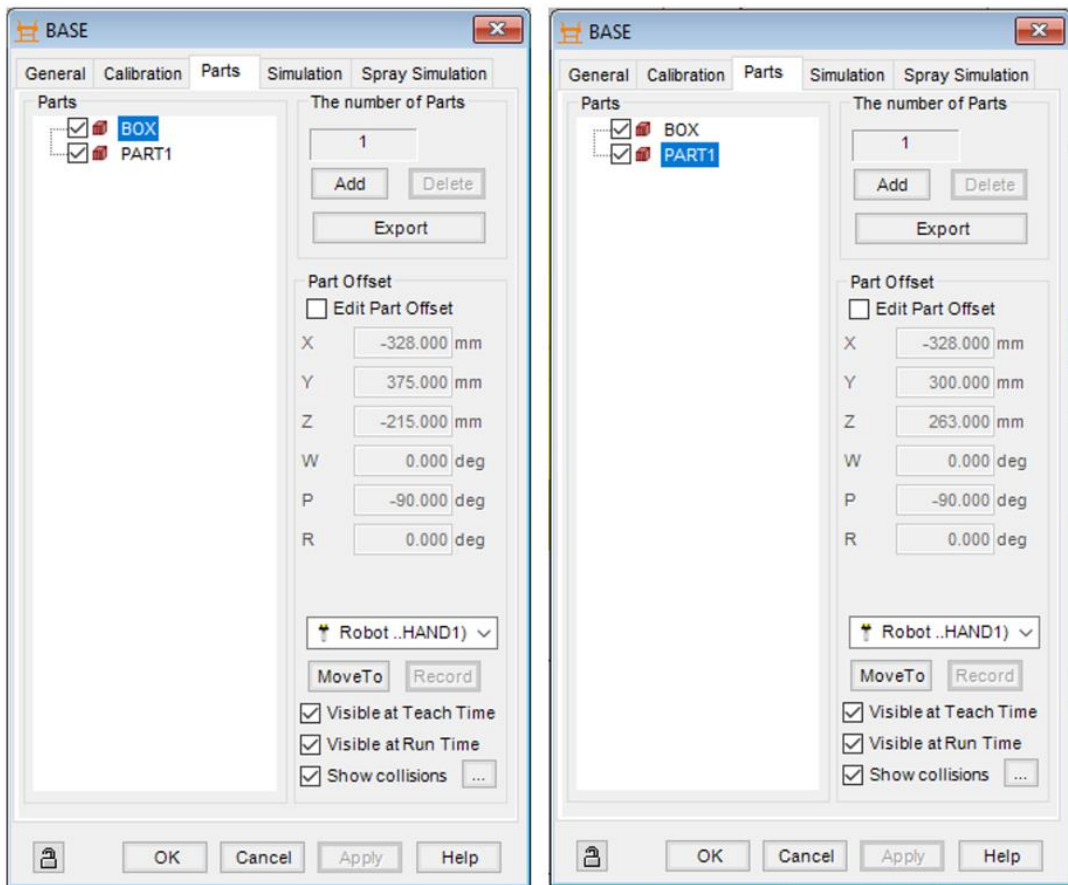


Figure 3.25 Configuration of the “Box” and “Part1” fixed parts on “Base”

Finally, we define two other fixed parts “Table1” and “Table2”, which will correspond to the supports of the “Part1” objects, which are equal to each other, saving the defined position of each support. The configuration of these fixed parts will be as follows:

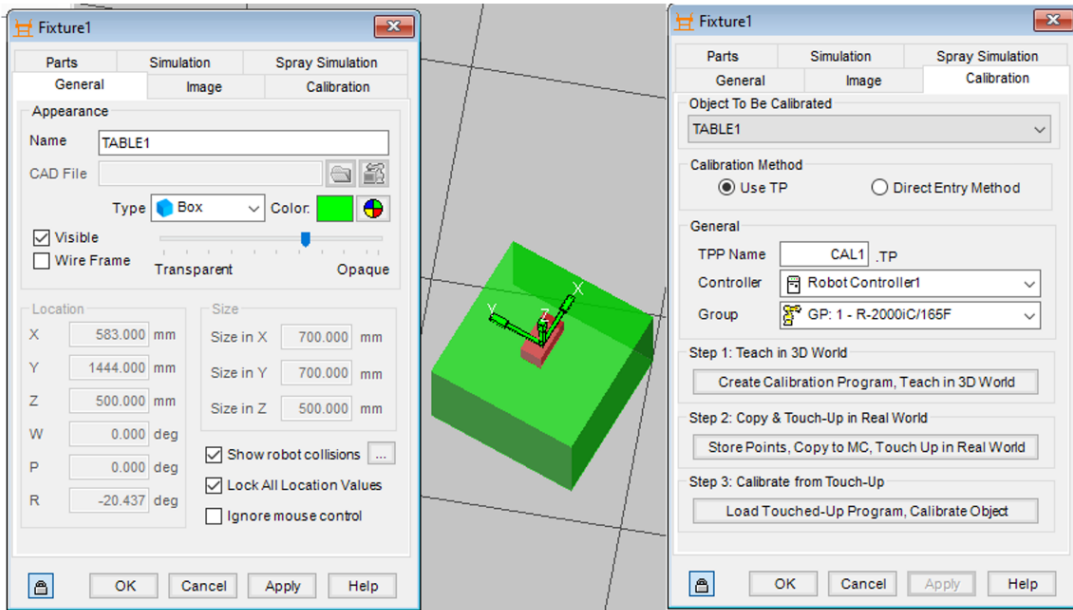


Figure 3.26 Configuration of the “TableX” fixed part

For the part, which will be related to “TableX” fixed part, we use only “Part1”. Configuration is shown below:

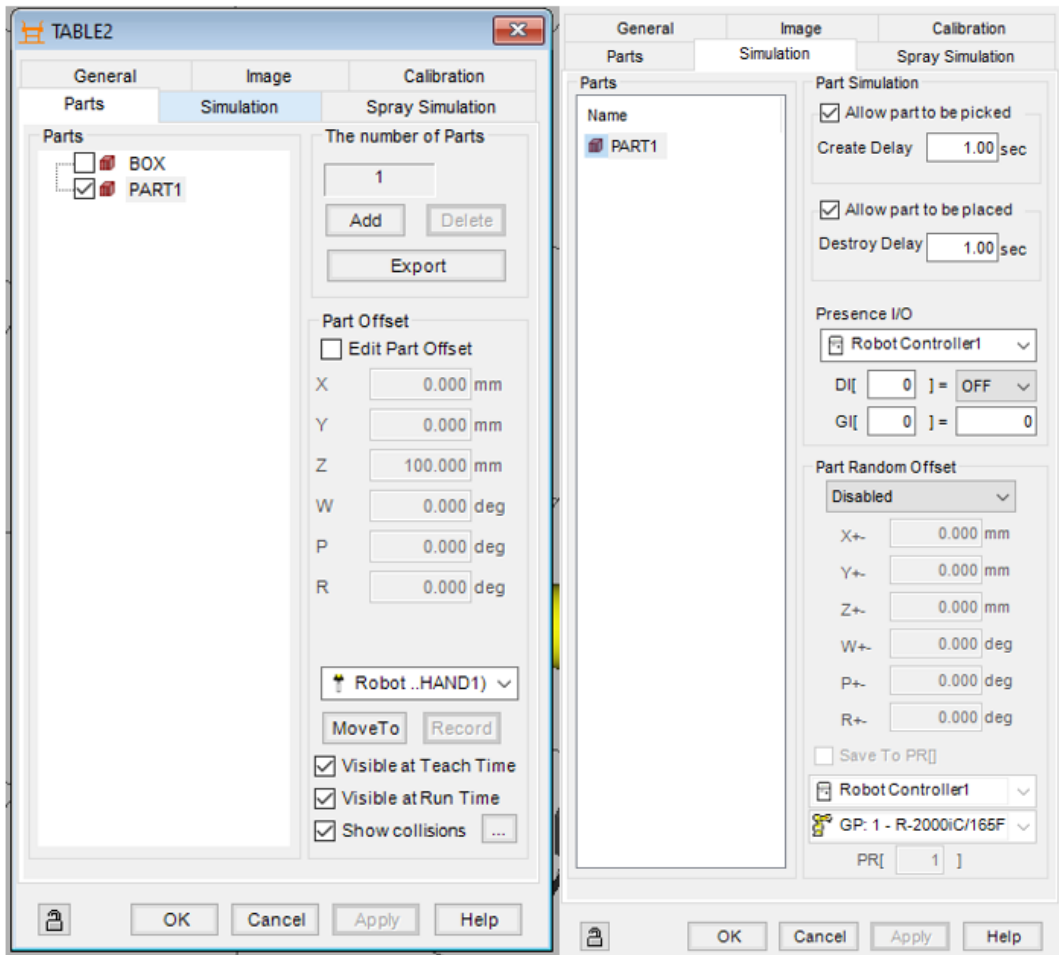


Figure 3.27 Configuration of the “Part1” on “TableX”

Once all the fixed parts that will be included in the system have been defined, if we review the “Cell Browser” menu, we can see that the configuration of the “Fixtures” option is as follows:

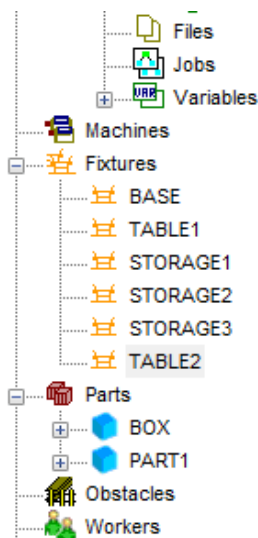


Figure 3.28 Setting the fixed parts in the “Cell Browser” menu

3.9 Definition of user reference system (UFRAME)

In the same way as in TP programming, in the HandlingPRO simulation software we can also create reference systems defined by the user "user frames", as long as the needs of the program require it for the convenience of defining points and certain movements (such as an inclined surface defined in the robot's working environment).

In the simulation, we consider that it is not necessary to define any inclined plane, since the predefined reference system allows us to identify the points easily. The possible "UFRAME" defined in our program would be a translation of the current reference system so that it would be relative to a position closer to a target, part or object of the robot's working environment. For this reason, we have defined a UFRAME, the result of a translation of the initial reference system, without performing any rotation in any of the coordinate axes.

This new "user-defined" reference system has been positioned in the fixed part (fixture) of the "work zone" of the program.

To define a new UFRAME we must access the "UserFrames" drop-down menu through the "Cell Browser" menu. Using the right button menu, we access "Uframe1 Properties" where we will configure the new user reference system.

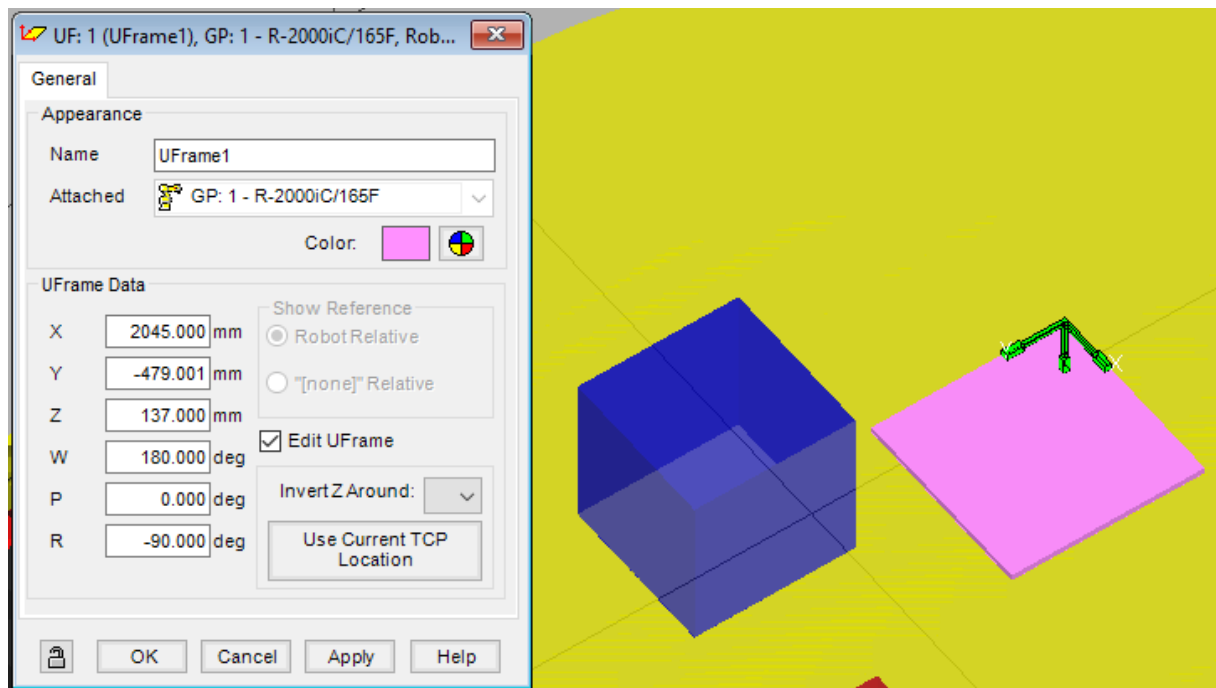


Figure 3.28 Definition of UFRAME as a new reference

In the UFRAME configuration window, we will define the name of the new reference system, the robot to which it is associated, the color and the position of the new reference system. This position can be entered with the values in the positions x, y, z and with the angles w, p, r, or by using the button "Use current TCP Location", defining the UFRAME in the position where the TCP of the robot at a certain time in the 3D tool environment.

The UFRAME definition menu also allows us to use the coordinates relative to the robot to define the new coordinate system, or not to use any reference, all in the "Show Reference" option.

3.10 Creation of the program

After creating each of the objects in the robot's work environment, we begin with the implementation of the robotic process program.

To program the movements of the system, we will use the tools offered by the application, through the "Teach Program" menu. The definition of points and actions is the basis for developing the manipulator's work algorithm. As an additional utility, we will use the virtual console "Teach Pendant" to be able to do tests by directly entering some coordinates in the "Current Position" tab and later being able to store said position as a new point within the program.

We will therefore explain the point definition method for programming the manipulator:

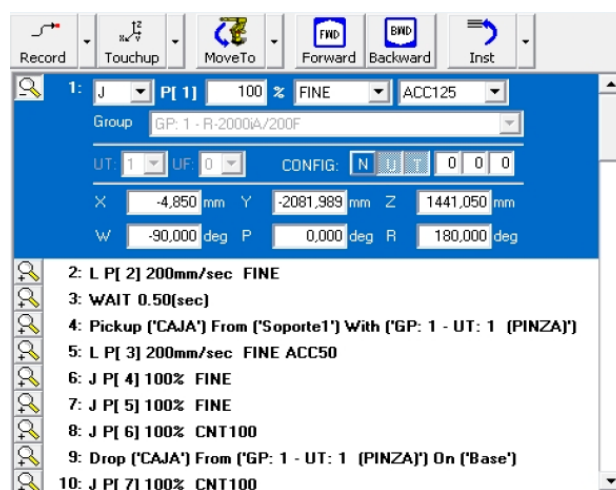


Figure 3.29 Appearance of the "Teach Program" menu

When accessing the “Teach Program” menu (through the “Teach” main menu), you will find the interface for defining the points and actions to compose the desired trajectories.

At the top of the window is the header with the different configuration options for this definition of points and actions. In the central and lower part we see the space reserved for the defined instructions, which make up the program. The definition buttons at the top are as follows:

Record - button used to define a point in the program with the current position of the robot. We can modify the position of the robot with the “Teach Tool” by moving the green sphere of this tool (which marks the reference origin of the tool) to be able to use this position in the definition of the point. To define the positions with this button, there is a very useful tool that is the quick access bar "Move To Quick Bar", since it allows us to reach faces, vertices, centers and edges of the various objects defined in the cell.

It also allows us to define the x, y, z coordinates, and the rotation angles w, p, r, with respect to the chosen reference system (in our case the one defined for the “Hand” tool). We can modify the values of these variables using the “Teach Tool” or generate a new point with the values started at zero, using the drop-down menu on the right side of the button.

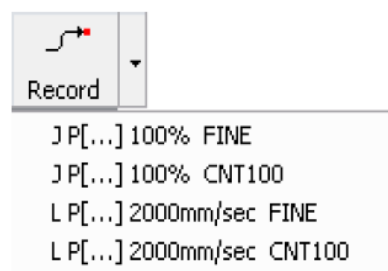


Figure 3.30 Appearance of the button “Record

Touchup - this button allows us to define a point in space by pressing later in the 3D graphic environment of the “teach mode”. It behaves similarly to the "Record" button, but it does not capture the position variables of the current situation of the robot, but of the subsequent placement of the mouse.



Figure 3.31 Appearance of the “Touchup” button

MoveTo - the button offers the possibility of moving the robot to any point defined in the system program, or to any of the positions established in the “Fixtures” menu with the definition of some object (“part”) associated with the drop-down.



Figure 3.32 Appearance of the “MoveTo” button

Forward - allows the robot to advance step by step through the points defined in the movements algorithm. With this tool we can visualize the sequence of defined movements with greater comfort.



Figure 3.33 Appearance of the “Forward” button

Backward - allows the robot to go back step by step through the points defined in the movement algorithm. With this tool we can view the sequence of defined movements more comfortably in the same way as with the "Forward" button.



Figure 3.33 Appearance of the “Backward” button

Inst - makes it possible to insert an action into the program without having to define a point directly. It offers us the possibility of capturing an object previously defined in the “Parts” menu (“Pickup”); deposit a defined object on a previously created support

(“Fixture”) (“Drop”), insert a wait instruction with a defined time (“Wait”) and make a call to another program defined with the “Call” instruction.

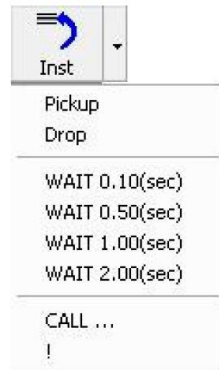


Figure 3.34 Appearance of the “Backward” button

After having reviewed the buttons on the header, the format for defining the points of the process program will be explained. The positions belonging to the “world” of the robot are defined following the format specified in the previous section “TPE programming”:

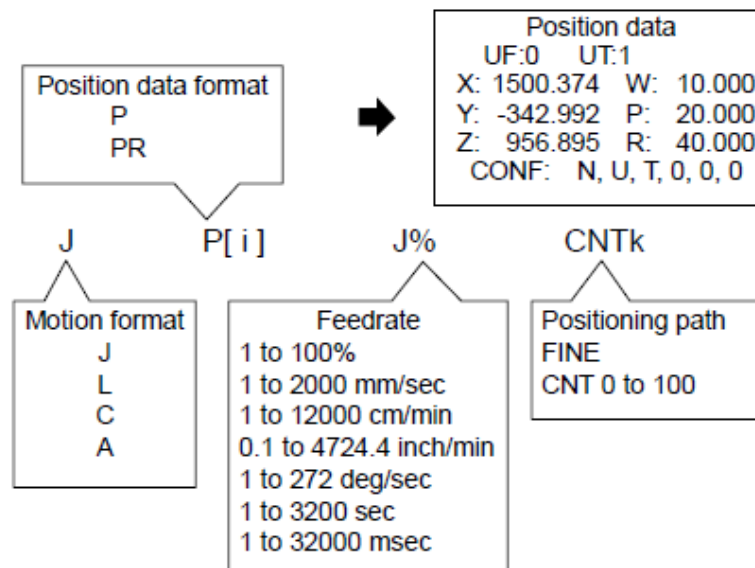


Figure 3.35 Statement format for defining positions

The first field of the sentence indicates the type of movement, and allows us to choose the movement options: linear, which is precise from the beginning to the end following a defined path; and the link (join), which moves the TCP to each of the defined

points linking the movements with each other. The following fields indicate "the type of position" and "the position" of the program we are defining.

The fourth field determines the speed with which the movement is to be performed. Depending on whether the movement is linear (L) or link (J) we can determine the speed more precisely (in mm / sec) or more generally (in%) respectively.

The fifth field indicates the type of completion of the process. This can be continuous (CNT) or precise (FINE).

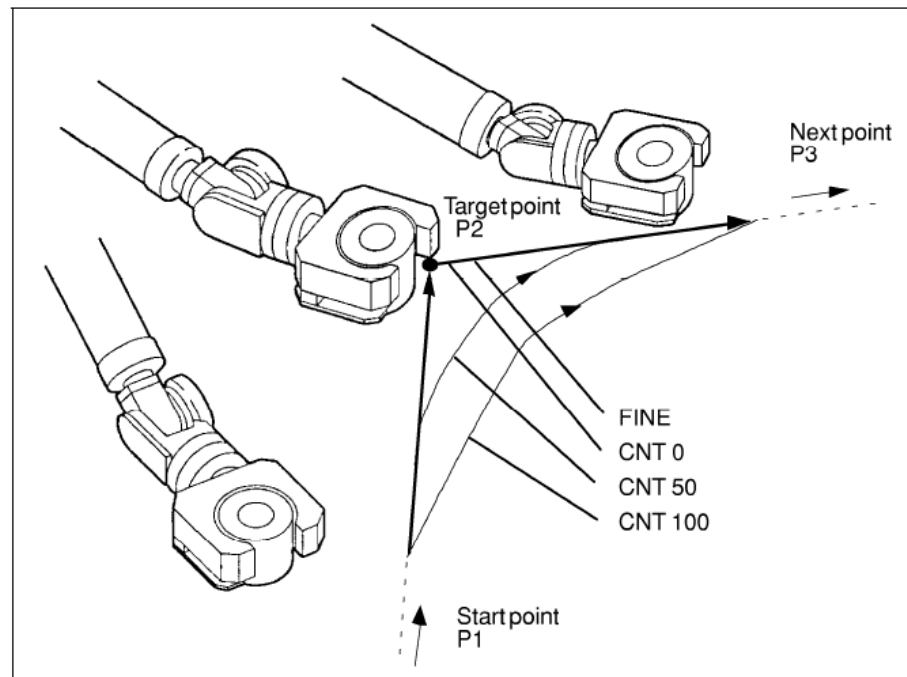


Figure 3.36 Types of motion completion

The last field is optional and determines the dynamics of the movement. In the case of the “Teach Program” menu, we can choose the ACC option, which determines the acceleration / deceleration rate when moving from one point to another. To define an action in the menu, such as capturing an object or depositing it, we must current as follows:

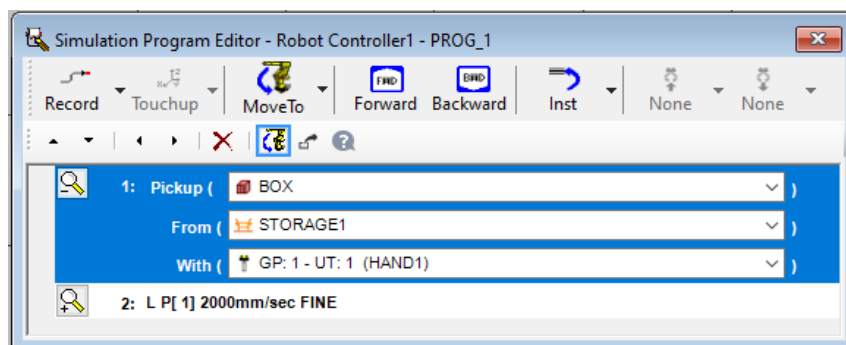


Figure 3.36 Definition of action "Pickup"

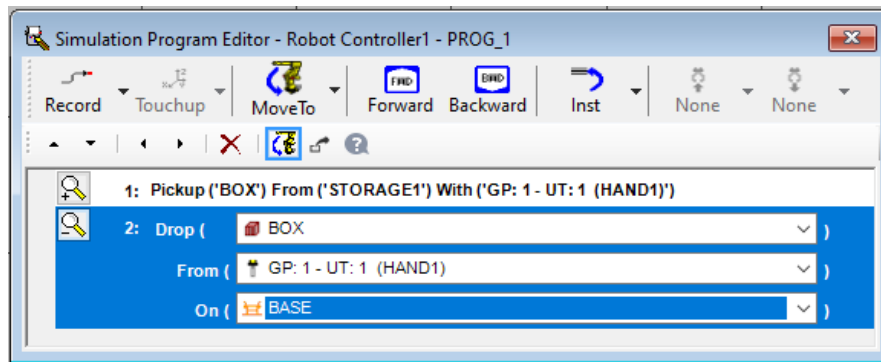


Figure 3.36 Definition of action "Drop"

To determine a manipulation action we must first define the object created in the “Parts” menu that we want to manipulate, the specific tool from the “Tooling” menu with which we want to work and the fixed part determined in “Fixtures” where we want to perform the action.

To create a point, we will use the menu of the virtual “Teach Pendant” “Current Position” as a tool, since it allows us to move the robotic arm to a point in the robot's space, as long as it is allowed in its range of motion.

This option allows us to do tests before recording a point and visualize that the desired position is possible, and consequently that the robot is positioned to reach the point that we are defining. The menu of the virtual "Teach Pendant" is as shown below:

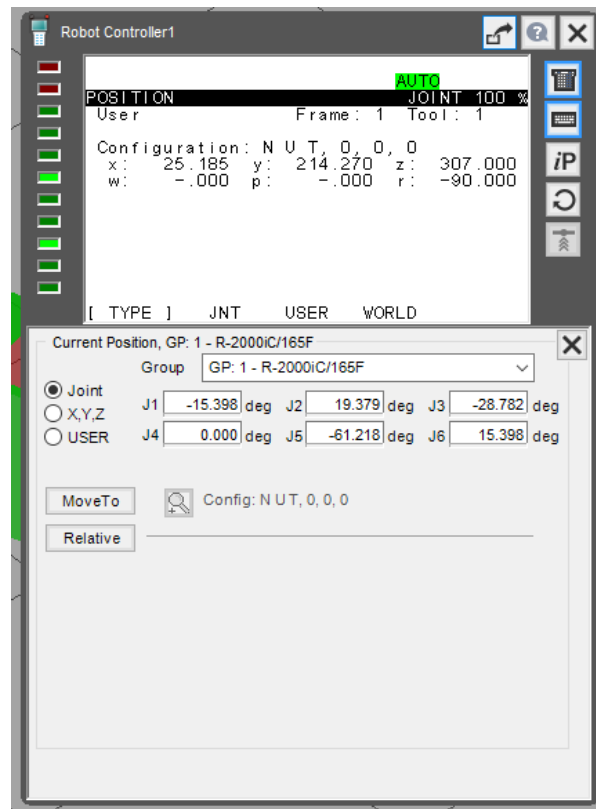


Figure 3.37 TPE “Current Position” menu

This menu offers three ways of entering data to reach a specific position with TCP. First, we can specify the degrees of each of the axes (Joint); second, the XYZ coordinates and WPR rotations of the point to be reached with respect to the base reference system; or third, the XYZ coordinates and WPR rotations in a user reference system (UFRAME).

Having analyzed the tools for the definition of positions and creation of the algorithm of the process, the definition of some example points in reference to approximations to some objects of the work cell will be described below. Approach to deposit “BOX” in “Storage1”:

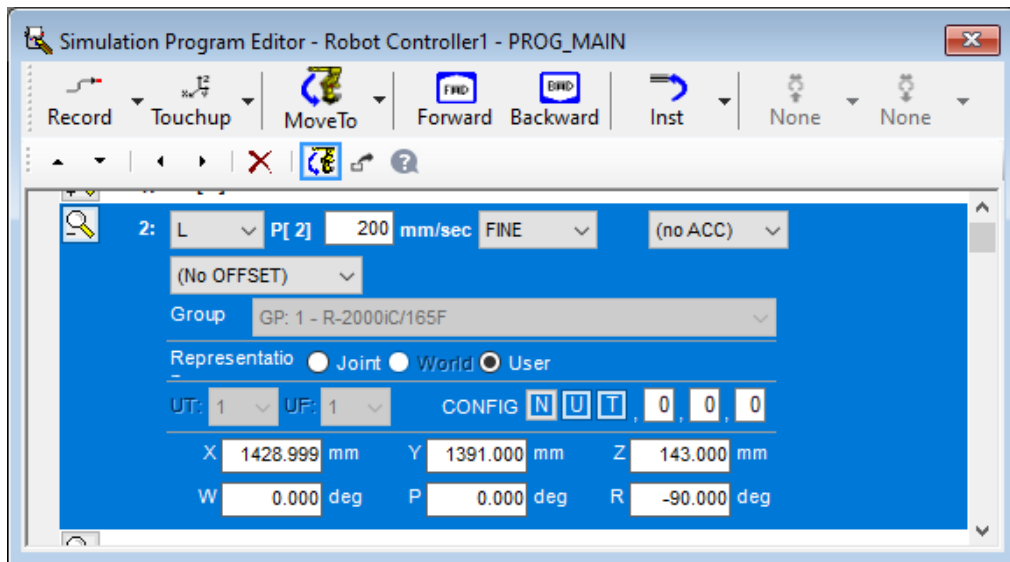
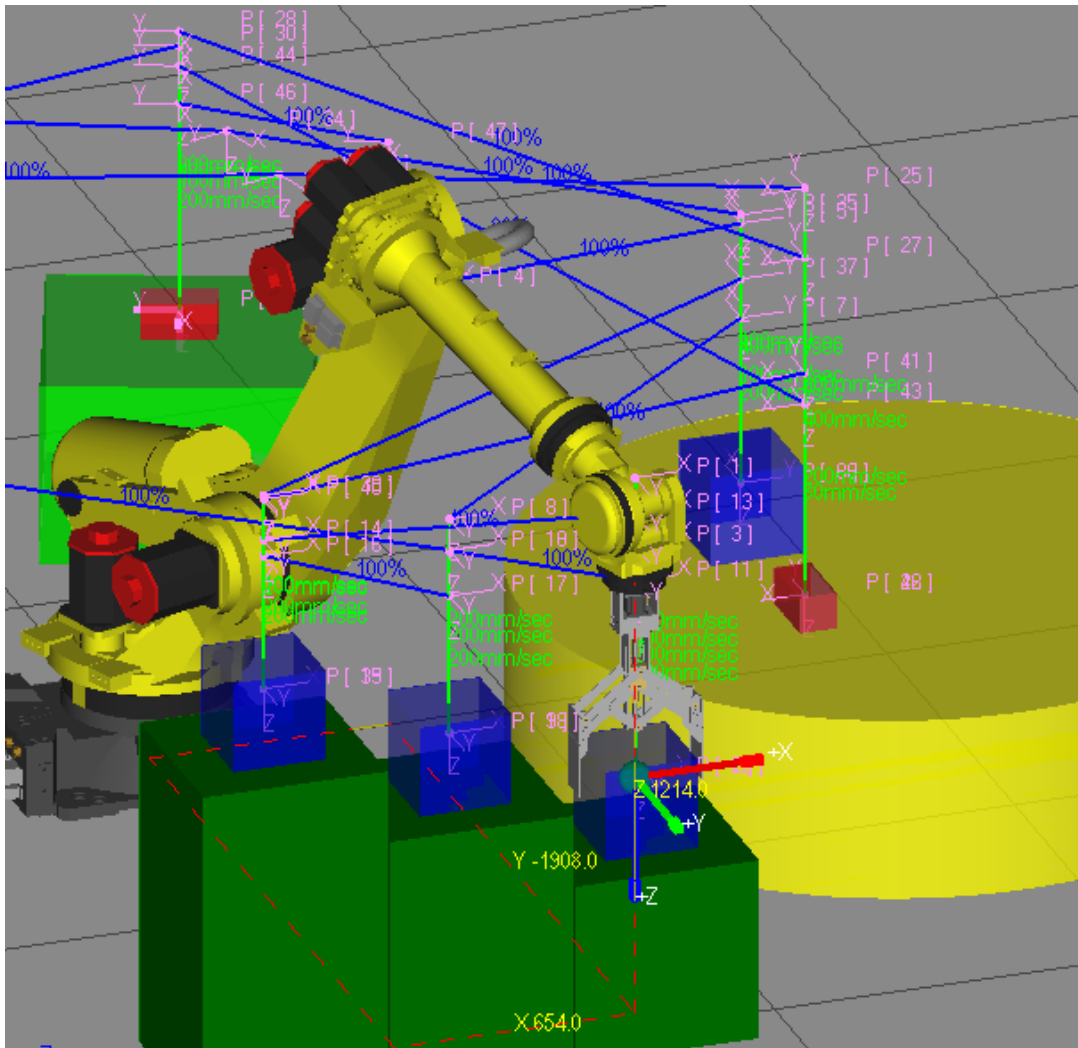


Figure 3.37 Definition of position for the approach to "Storage1"

Approach to capture "BOX" from "Storage3":

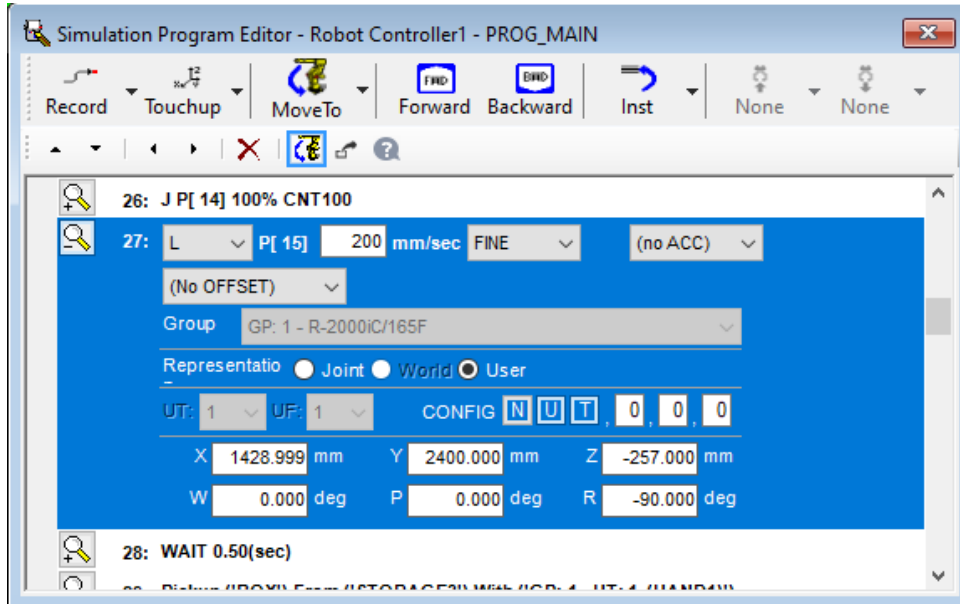
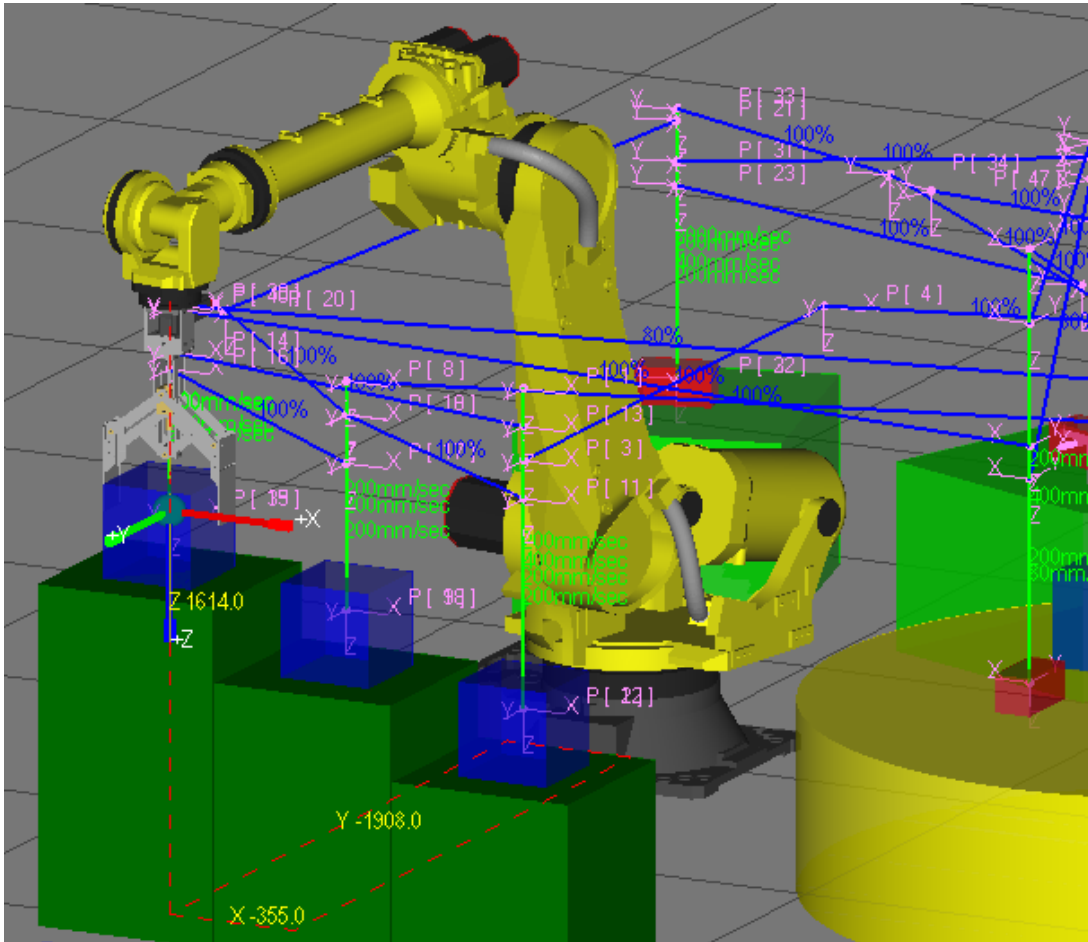


Figure 3.37 Position definition for the approach to "Support3"

Positioned to deposit "BOX" in "Storage2":

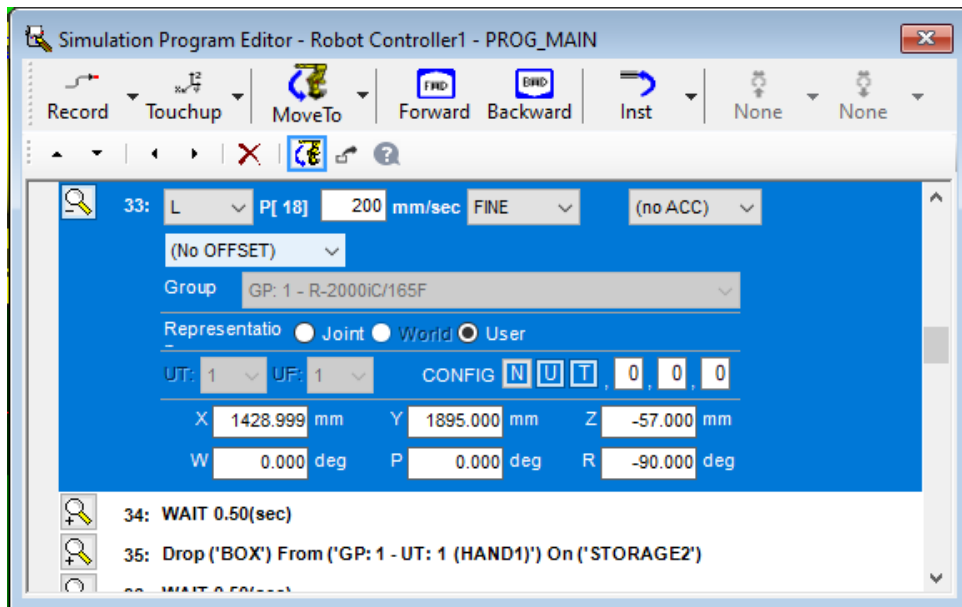
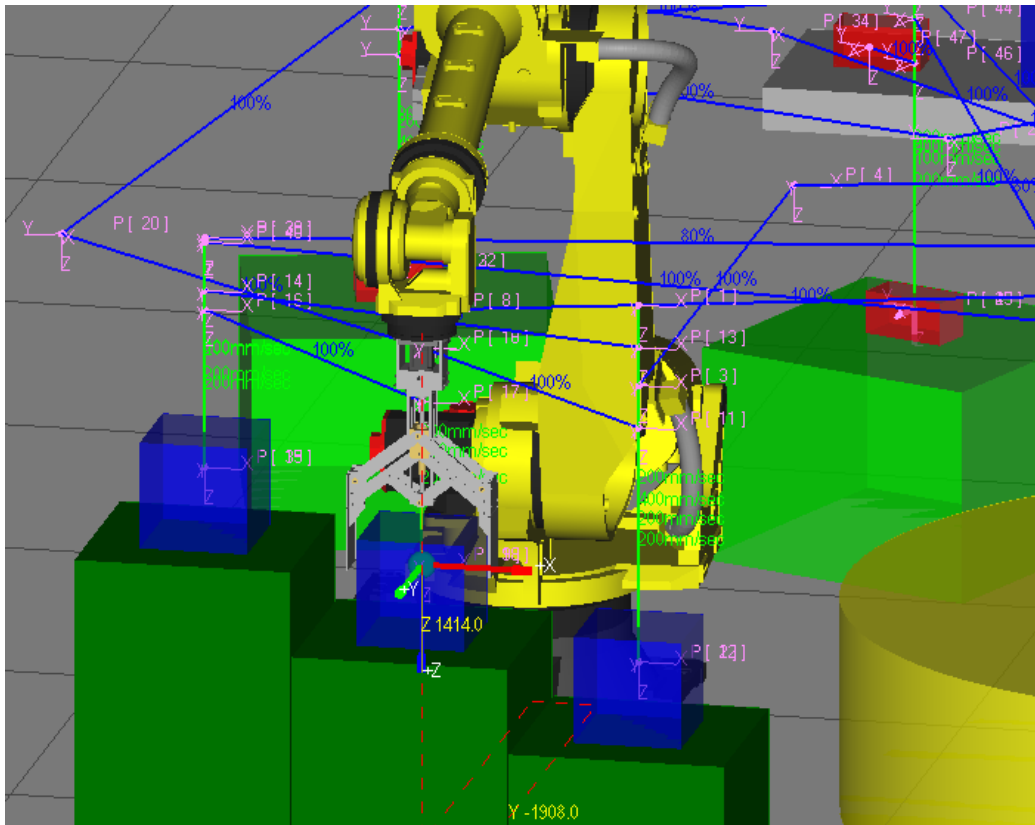


Figure 3.37 Definition of position to deposit "BOX" in "Storage2"

Robot positioning errors in the simulation environment for the position to deposit "BOX" in "Storage2" are defined in the table below:

Table 3.1 Robot positioning errors for deposit "BOX"

	X (mm)	Y (mm)	Z (mm)	W (deg)	P (deg)	R (deg)
--	--------	--------	--------	---------	---------	---------

Planned	1429	1895	-57	0	0	-90
Real	1428.999	1895	-57	0	0	-90
Δ	0.001	0	0	0	0	0

Approach to capture "Part1" in "Table2":

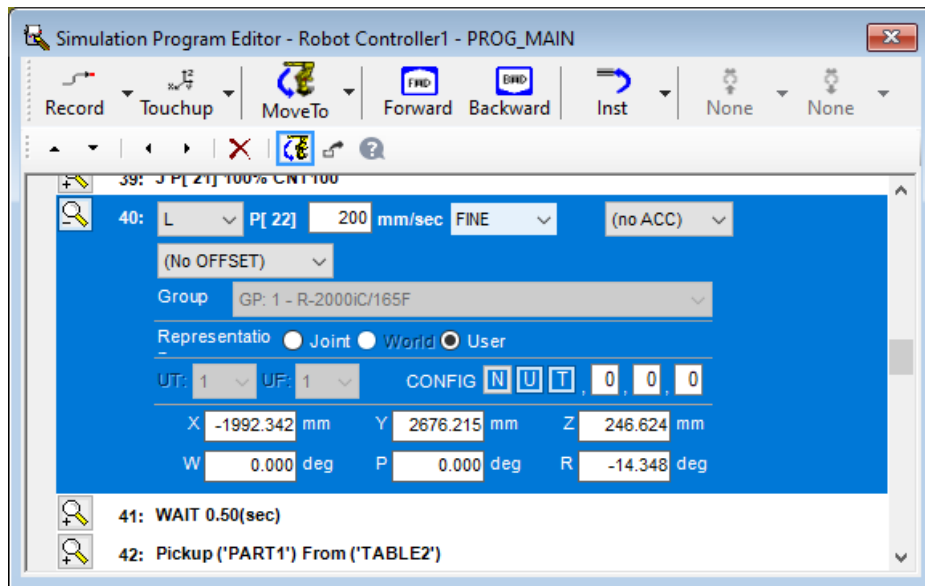
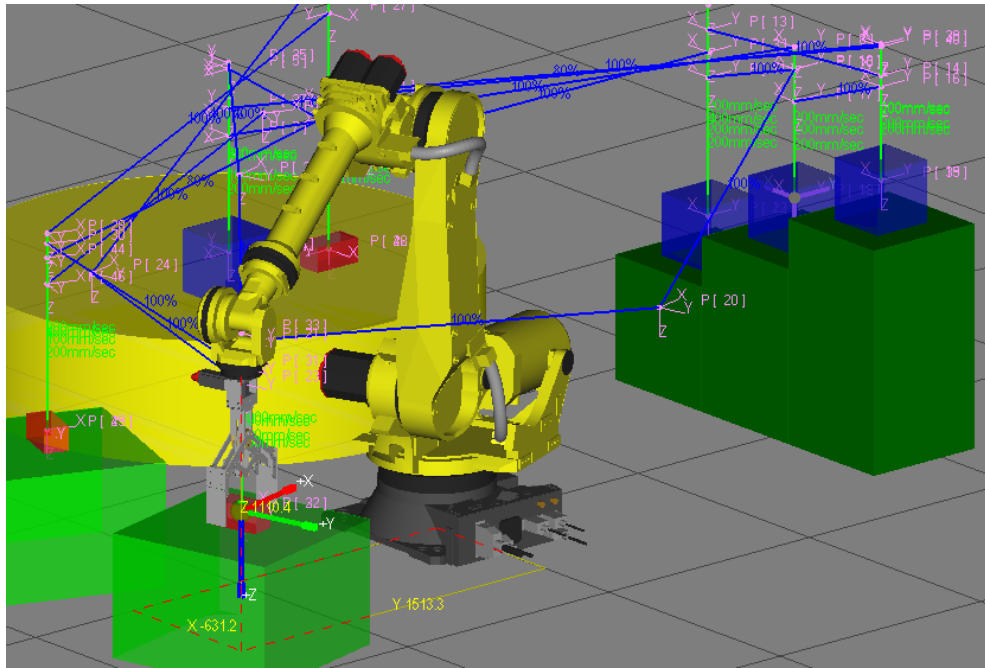


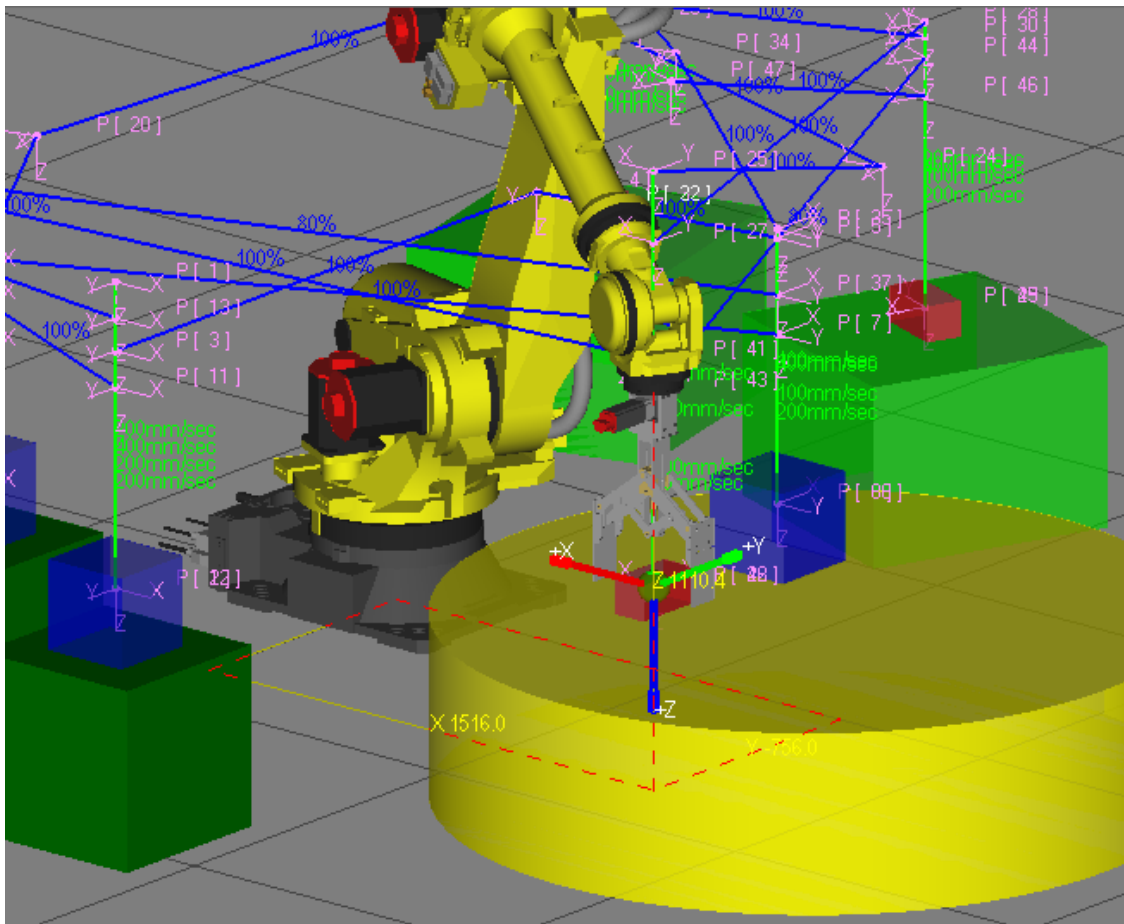
Figure 3.37 Definition of position to capture "Part1" in "Table2"

Robot positioning errors in the simulation environment for the position to deposit "Part1" in "Table2" are defined in the table below:

Table 3.1 Robot positioning errors for deposit "Part1"

	X (mm)	Y (mm)	Z (mm)	W (deg)	P (deg)	R (deg)
Planned	-1992.340	2676.215	246.625	0	0	-14.350
Real	-1992.342	2676.215	246.624	0	0	-14.348
Δ	0.002	0	0.001	0	0	0.002

Position on "Base" after depositing "Part1":



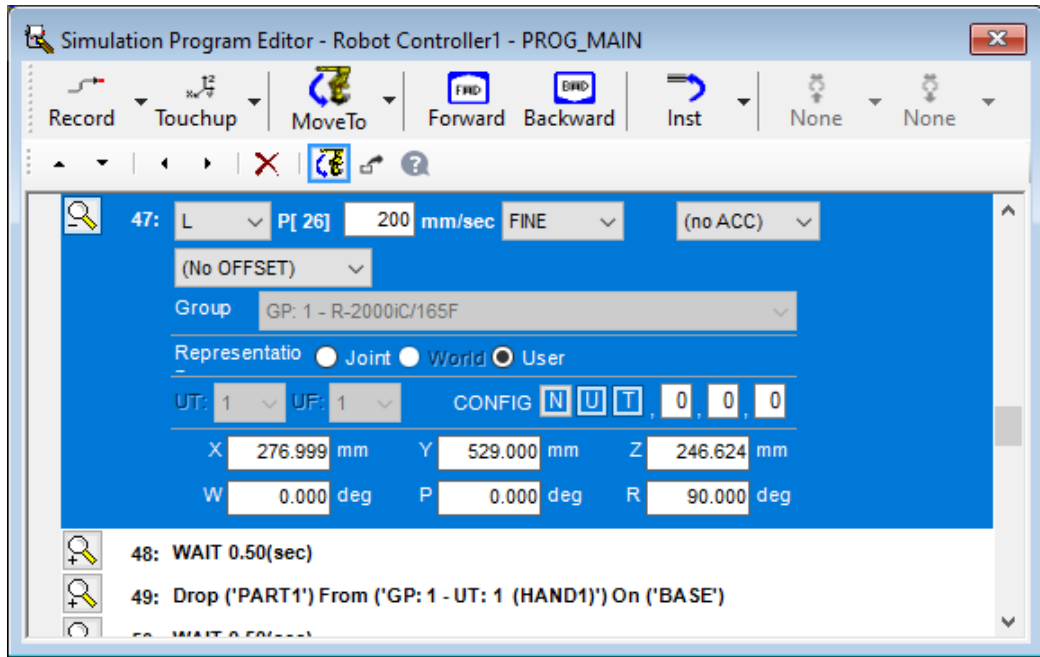


Figure 3.37 Definition of position to deposit "Part1" on "Base"

Robot positioning errors in the simulation environment for the position to deposit "Part1" on "Base" are defined in the table below:

Table 3.1 Robot positioning errors for the positioning on "Base"

	X (mm)	Y (mm)	Z (mm)	W (deg)	P (deg)	R (deg)
Planned	277	529	246.625	0	0	90
Real	276.999	529	246.624	0	0	90
Δ	0.001	0	0.001	0	0	0

Approach "Table1" to capture "Part1":

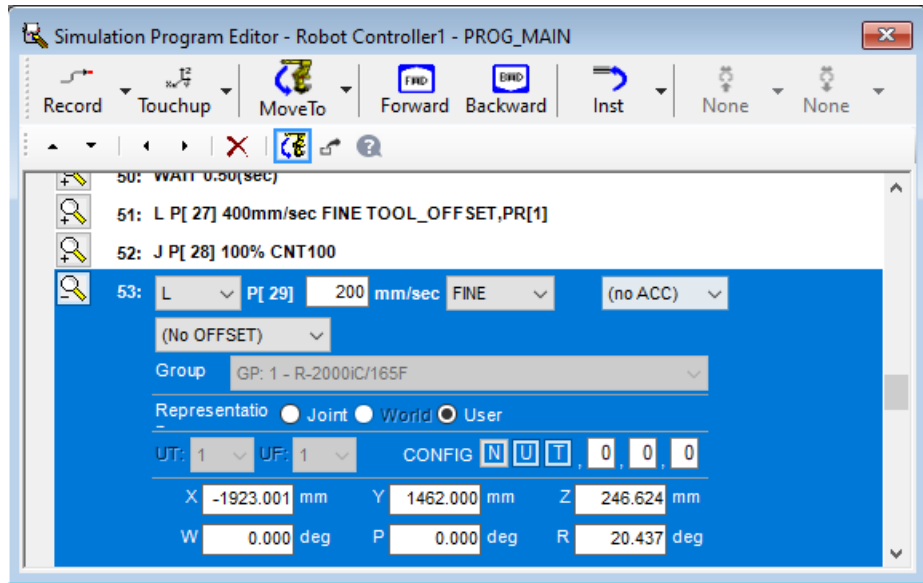
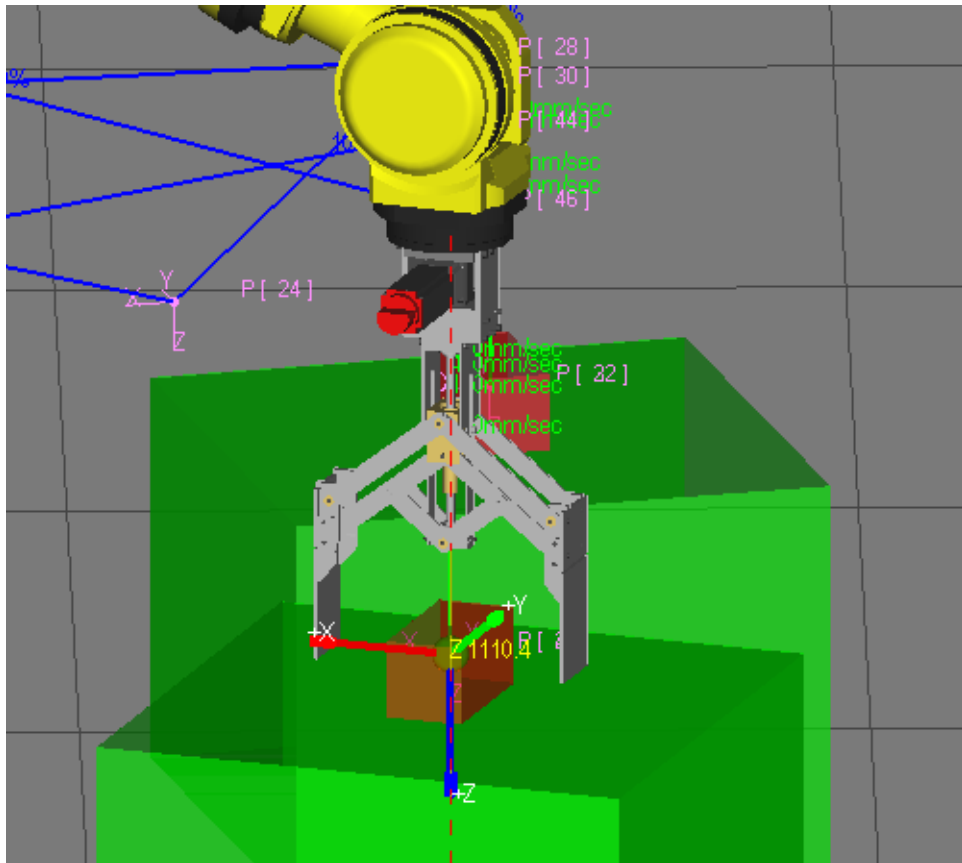


Figure 3.37 Definition of position to capture "Part1" from "Table1"

Robot positioning errors in the simulation environment for the position to capture "Part1" from "Table1" are defined in the table below:

Table 3.1 Robot positioning errors to capture "Part1" from "Table1"

	X (mm)	Y (mm)	Z (mm)	W (deg)	P (deg)	R (deg)
Planned	-1923	1462	246.625	0	0	20.437
Real	-1923.001	1462	246.624	0	0	20.437
Δ	0.001	0	0.001	0	0	0

An important condition in software development is to avoid collisions between the robot and the environment, other parts and other robots. In the study in this work, a linear vertical approximation to each object was used, which is part of the general algorithm for developing software for a robotic system. P[8], P[17], P[18] on the figure below are the positions from which we linearly approached the object to avoid collisions and ensure the correct movement of the detail.

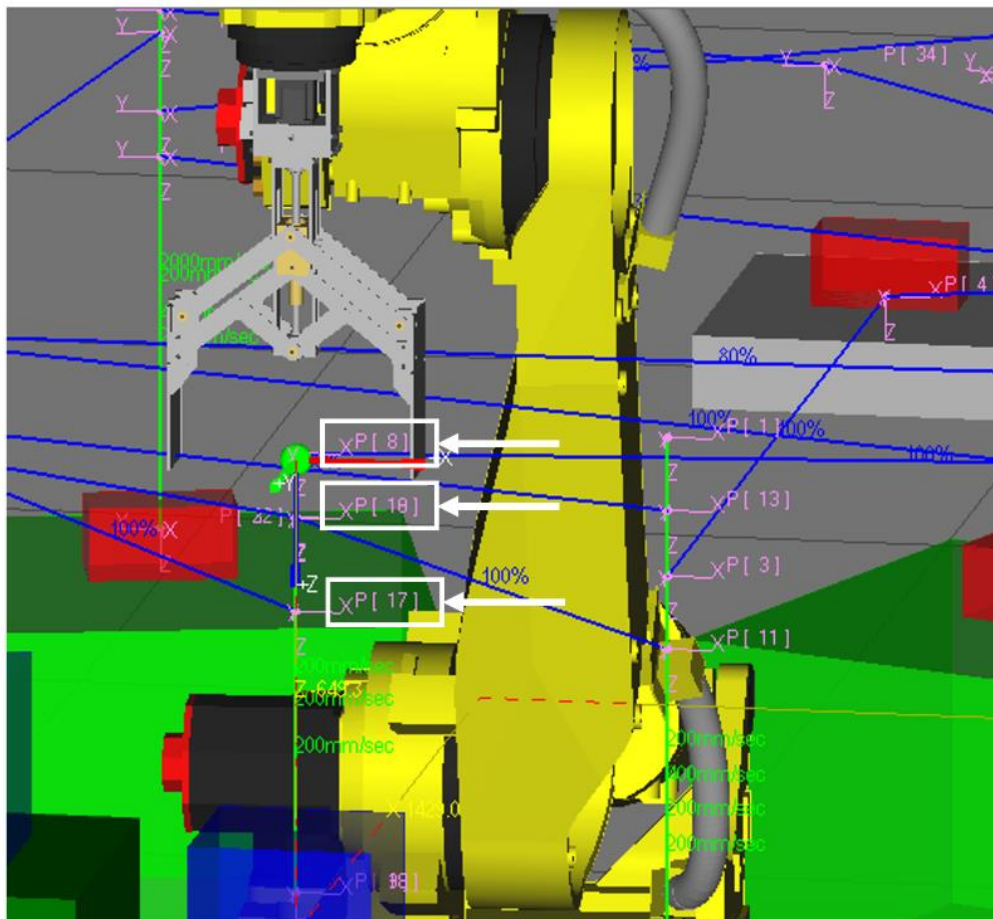


Figure 3.37 Definition of the positions for linear vertical approximation

Blue line shows us paths along the positions indicated in the code.

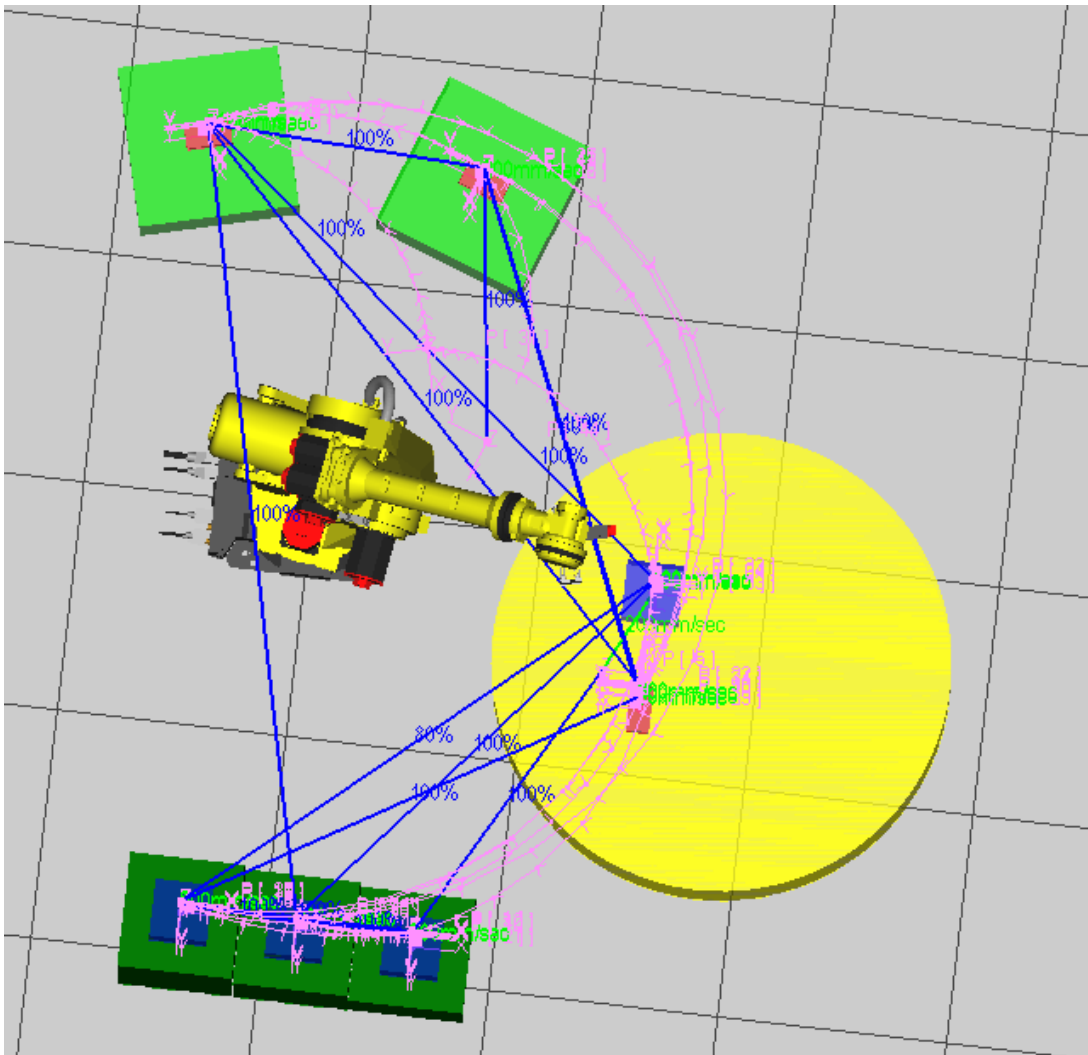


Figure 3.37 Trajectories according to positions

3.11 Simulation of the process

With the robotic system fully realized and the position definition algorithm completed, we run the simulation of the process to monitor the programmed system and each of the planned actions to be performed in the robotic cell.

Using the buttons on the quick access bar, or through the “Run Panel” menu, located in the “Teach” main menu, we can run the simulation to view the complete process, obtaining information on possible collisions, erroneous movements, or inaccuracies in approximations of arm movements. In this way, we will visually identify errors and easily correct them before they appear in the real production system, since the proactive effect of working virtually is one of the main advantages of simulation systems.

The “Run Panel” menu allows you to control various aspects of the simulation, to be able to adjust it according to the needs that affect the supervision of the system.

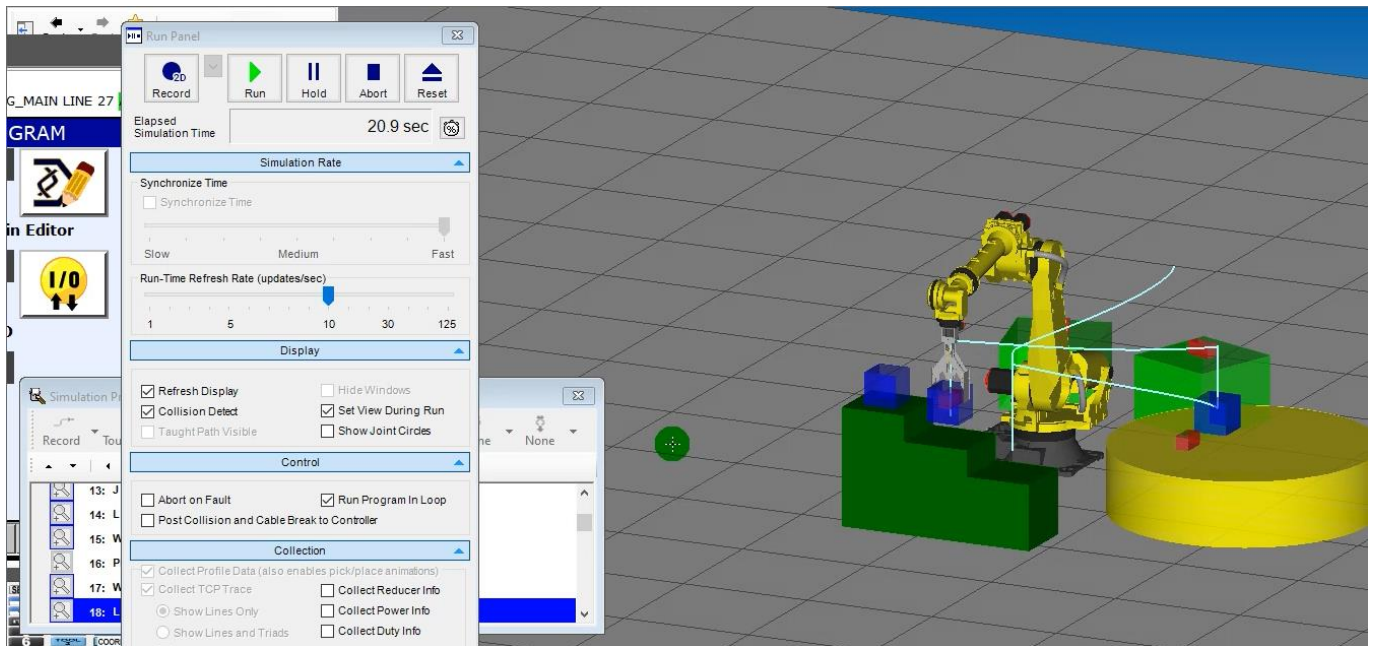


Figure 3.37 Simulation using Run Panel

The "Run Panel" menu consists of the run buttons at the top, and various configuration options at the center and bottom of the window.

In the “Run time refresh rate” part we can configure the number of “frames” or images per second that the process simulation will show, thus configuring the fluidity of the simulation animation.

In "Options" we can activate or deactivate some specific options, which are detailed below:

- Collision detection.
- Definition of the TCP trace (arm trajectories).
- Play in a loop ("loop" mode).
- Tracing of points defined in the program.
- Hide windows in simulation.
- Update screen.
- Compress AVI video file.

Once the simulation is visualized, we will correct the collisions that occur, the errors and the mismatches detected in the process. We will redefine the “problem” points to achieve the process with the positions and the desired precision.

In this way we will adjust all the trajectories of the robotic process of the industrial system to obtain the final program.

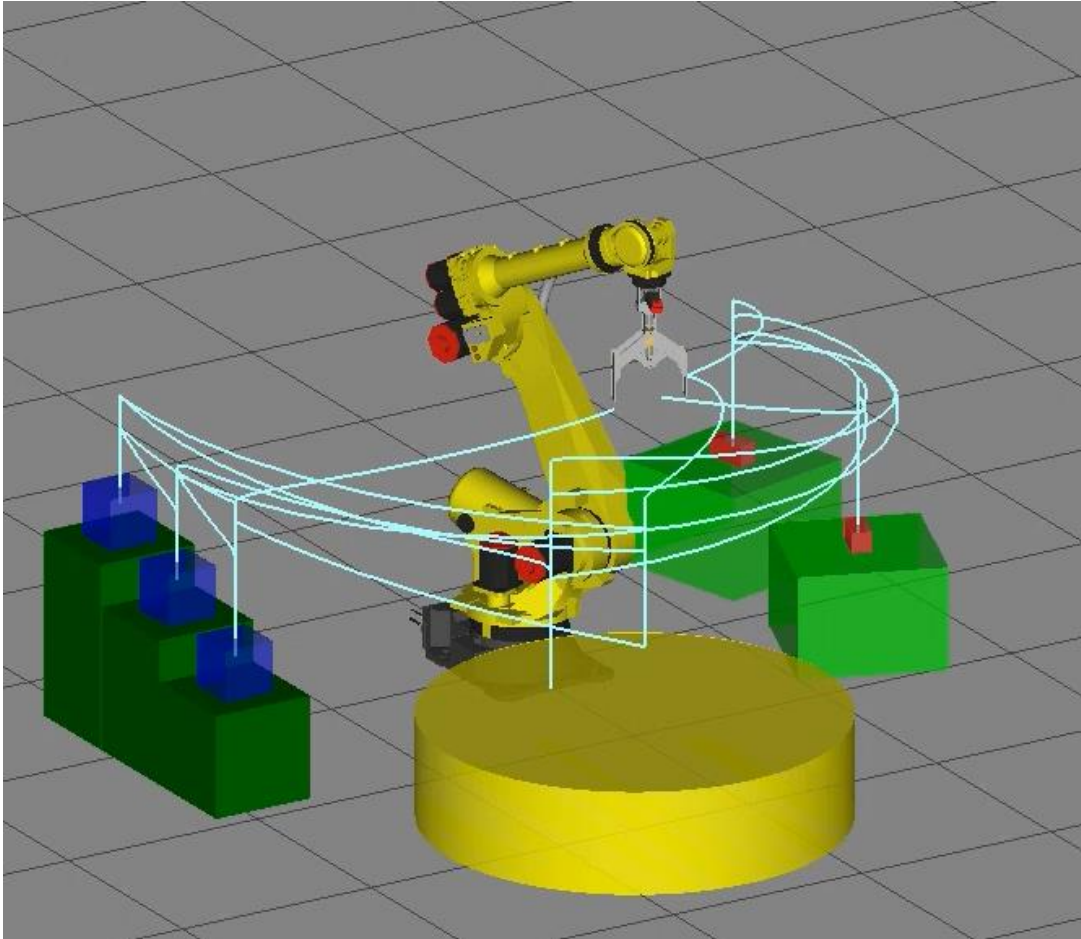


Figure 3.37 Final trajectories

3.12 Conclusions

The task and goals of our robotic system were outlined. Accordingly, the robot controller was configured and the simulation environment was created. Thus, we have a ready-made, debugged simulation model ready for research and testing before direct implementation into the real system.

The Roboguide application allows to perform the Calibration of the system automatically if we have the robot connected to the simulation application. It would be time

to perform the calibration and upload the program in the real environment (“upload”) by exporting the program. Finally, we would have the robotic cell ready to go into production.

4 EXTENSION OF THE WORK CELL

4.1 Robotic cell redesign

With the robotic cell programmed and assembled in production, it may be the case that it is necessary to expand the system, due to needs that may arise in the industrial process. We will then have to redesign the cell, even if it is in full production and with all systems at full capacity.

That is why, to implement the robotic system extension, we will reuse the design and programming made in the Roboguide HandlingPRO simulation tool previously.

The reprogramming of the work cell using software will allow us to redesign the new system without having to intervene in the real environment, which is operating in production. In this way, we will minimize the impact on the industrial plant's work chain.

After we have completed the design and implementation of the process, we will monitor the new system as was done with the first cell; We will correct any faults, misalignments and collisions that may appear, thus ensuring the full functionality of the new system in the virtual environment.

To expand the cell, a second manipulator will be configured, in addition to adding new fixed parts that intervene in the system and a new part to be manipulated, in addition to the objects already defined in the previous process.

It is intended to control the movements of the two manipulators so that they can both work simultaneously, and without collisions with each other, completing the process in an efficient way and controlling the workspaces of both robots (adjusting with special care the area common work, which will be the area where the greatest problems may arise).

Next, the procedure carried out to carry out the expansion of the work cell already defined in the previous sections will be presented.

We face the redesign starting from the initial work cell, adjusting the new work needs in the system and composing a new cell that meets the aforementioned requirements.

An additional “Robot2” manipulator is added to the existing system, two new work areas, and two of the three box supports are eliminated, leaving the cell design as shown in the following figure:

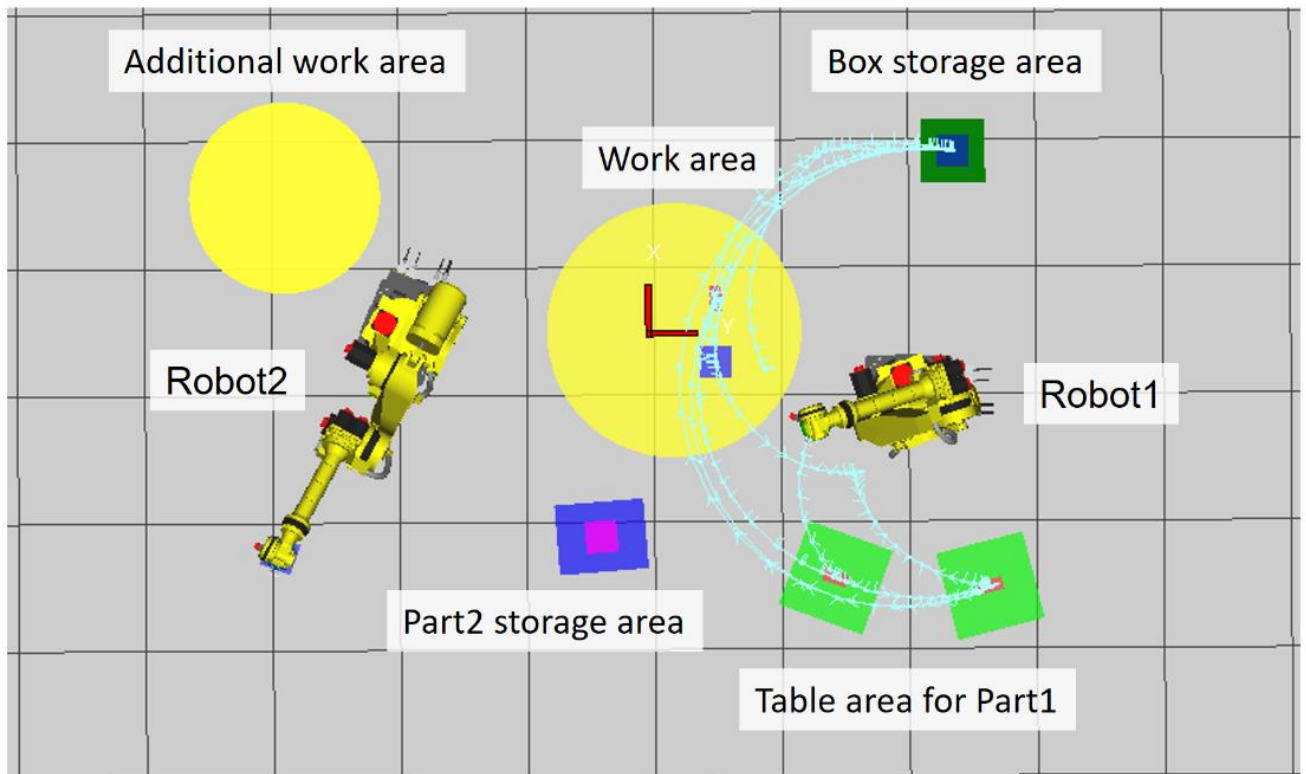


Figure 4.1 Expansion and redesign of the work cell

The new zones are:

- The zone of the additional manipulator. The new robot “Robot2” is located in this space, which is integrated into the work cell.
- The storage area of Part 2. This area will provide a new part to the system, which will intervene in the new defined areas.
- The additional work area. It is destined for a new mechanized action. Pieces will be moved to be handled and treated, to later be returned to the main work area.

The new state diagram will consist of six possible actions and six allowed positions. The execution of these actions and the positioning at these points will be carried out simultaneously between the two manipulators, synchronizing the movements to avoid any collision that may occur in the environment of the work cell. The actions of the new process are:

- Capture of "Box";
- Capture of "Part1";
- Capture of "Part2";
- Deposit of "Box";
- Deposit of "Part1";
- Deposit of "Part2".

The positions allowed in the manipulator environment are:

- Storage1;
- Base (Work area);
- Table 1;
- Table 2;
- Storage A;
- Base A.

The new process will consist of the following steps, specified for each manipulator:

Table 4.1 - Algorithm of movement for each robot

Robot 1:	Robot 2:
<ul style="list-style-type: none"> - Initial position.* - Positioning in "Storage1". - Capture of "Box". - Positioning in "Base". - Deposit of "Box" - Positioning in "Table2" - Capture of "Part1" - Positioning in "Base". - Deposit of "Part1" - Positioning in "Table1". - Capture of "Part1" - Positioning in "Table2" 	<ul style="list-style-type: none"> - Initial position.* - Positioning in "StorageA". - Capture of "Part2". - Positioning in "Base". - Deposit of "Part2" - Waiting position. - Positioning in "Base" - Capture of "BOX" - Positioning in "BaseA". - Deposit of "BOX" - Waiting position. - Positioning in "BaseA"

<ul style="list-style-type: none"> - Deposit of "Part1". - Waiting position. * - Positioning in "Base". - Capture of "Box". - Positioning in "Storag1". - Deposit of "Box". - Positioning in "Base". - Capture of "Part1". - Positioning in "Table1". - Deposit of "Part1". - Final position. * 	<ul style="list-style-type: none"> - Capture of "BOX" - Positioning in "Base". - Deposit of "BOX" - Capture of "Part1" - Positioning in "BaseA". - Deposit of "Part1" - Standby position. - Positioning in "BaseA" - Capture of "Part1" - Positioning in "Base". - Deposit of "Part1" - Capture of "Part2" - Positioning in "SupportA" - Deposit of "Part2". - Final position. *
--	---

Thanks to the execution sequences of the two manipulators, through the analysis of actions and positions of the system, we can generate the new Petri net graphs (separated for each manipulator to facilitate reading).

It is worth mentioning the introduction of a new waiting state, used by "Robot2" at some points in the process to avoid collisions with "Robot1".

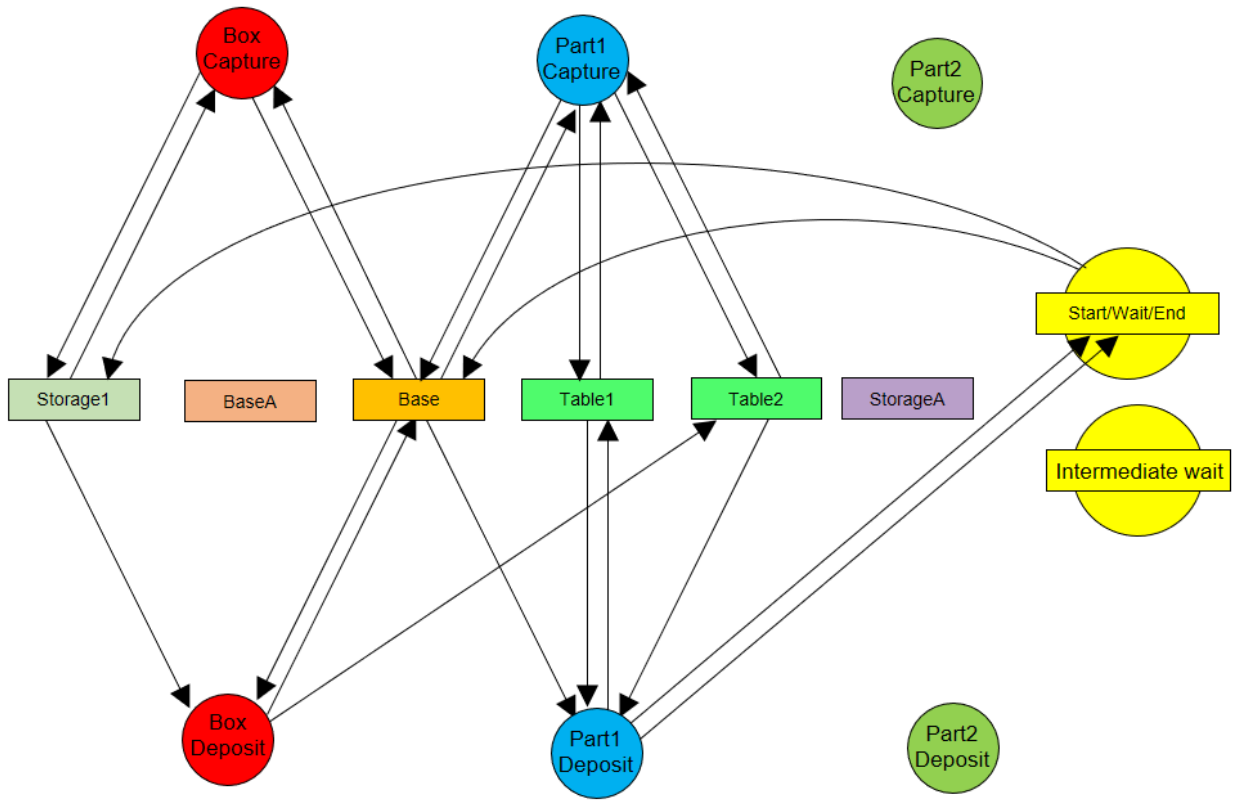


Figure 4.1 Petri net for the "Robot1" movements

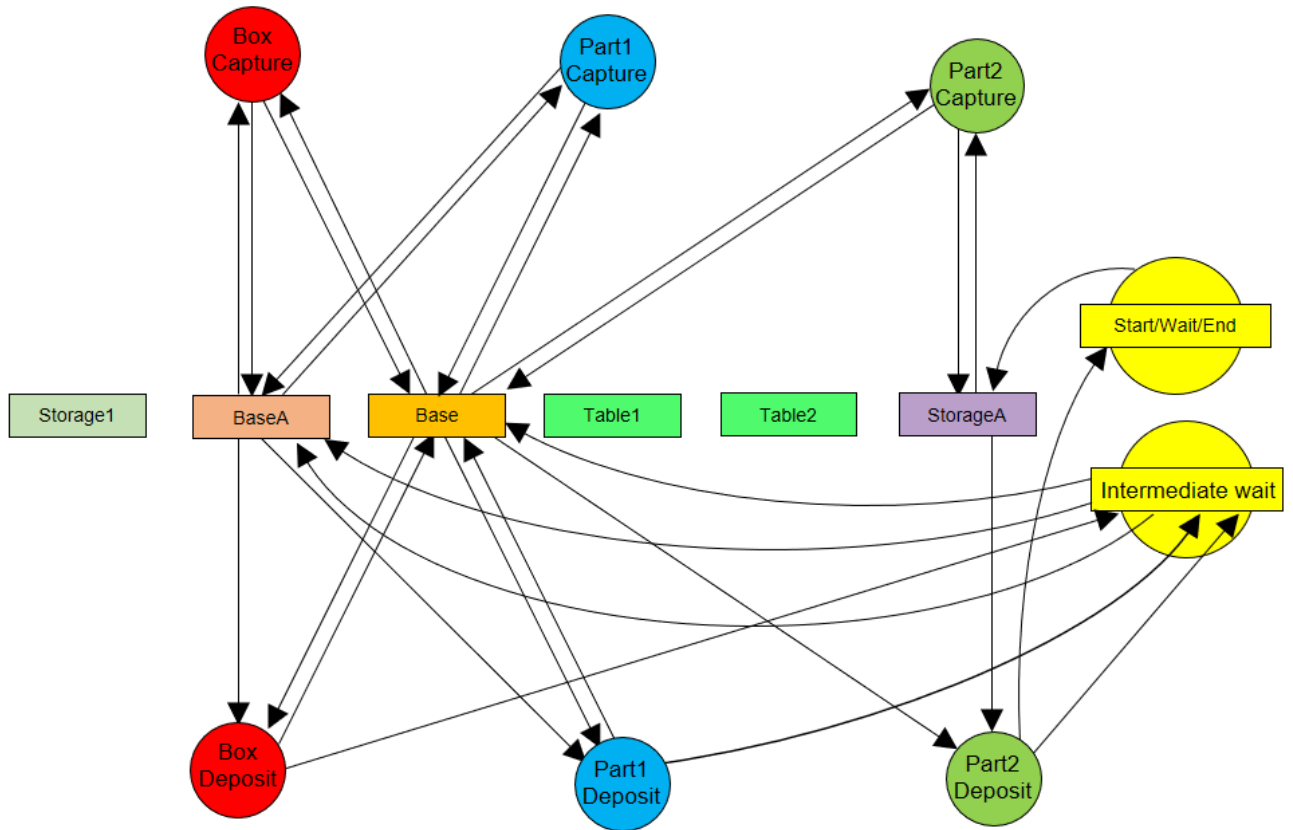


Figure 4.1 Petri net for the "Robot2" movements

4.2 Cell configuration with additional robot

To configure the new cell, first we must add a new manipulator to the work environment, using the "Cell Browser" menu and on the "Robot Controllers" menu, using the right mouse button, we will choose the option "Add Robot". Next, the robot creation wizard will appear, already used in the first creation of "Robot1", where we must choose the option "Create a copy of an existing robot". In this way we will obtain a robot just like the previous one to be able to start configuring the new cell. First of all we must configure the position of this new robot, since it will appear in the same position as the first (overlapped).

Then we must redefine the UTOOL tool used for this second robot. We use the "Gripper" tool already defined above. Finally, we will create the rest of the objects in the environment of the new cell. In the "Cell Browser" menu on the "Parts" button, we use the right mouse button to choose the "Add Part" option, configuring it as indicated below:

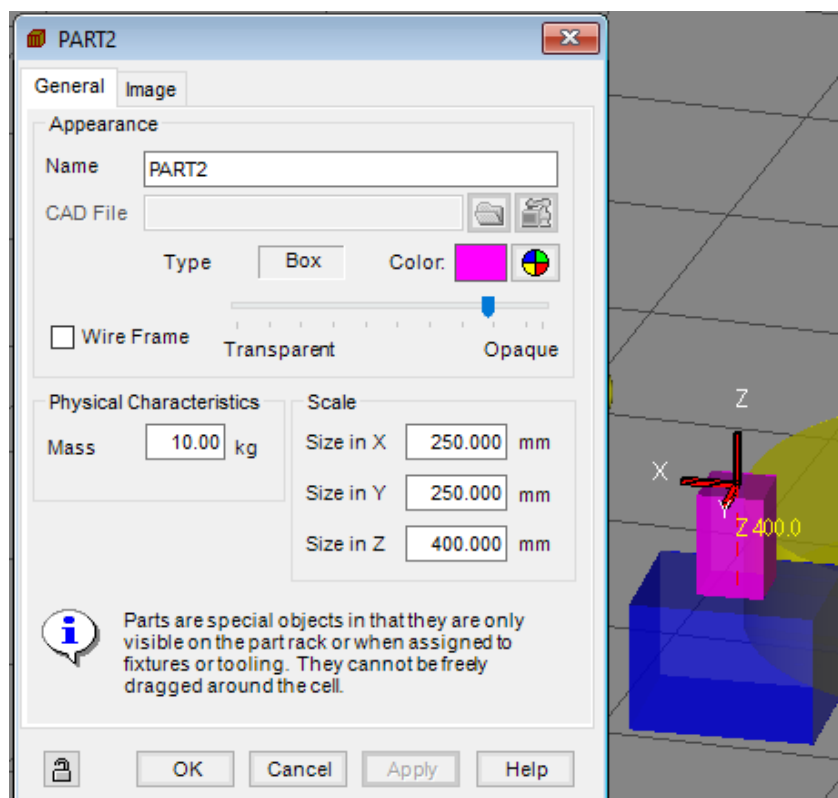


Figure 4.1 Properties menu of the object "Part2"

Storage creation for "Part2" is in the "Cell Browser" menu on the "Fixtures" button, we use the right mouse button to choose the "Add Fixture" option, configuring it as indicated below:

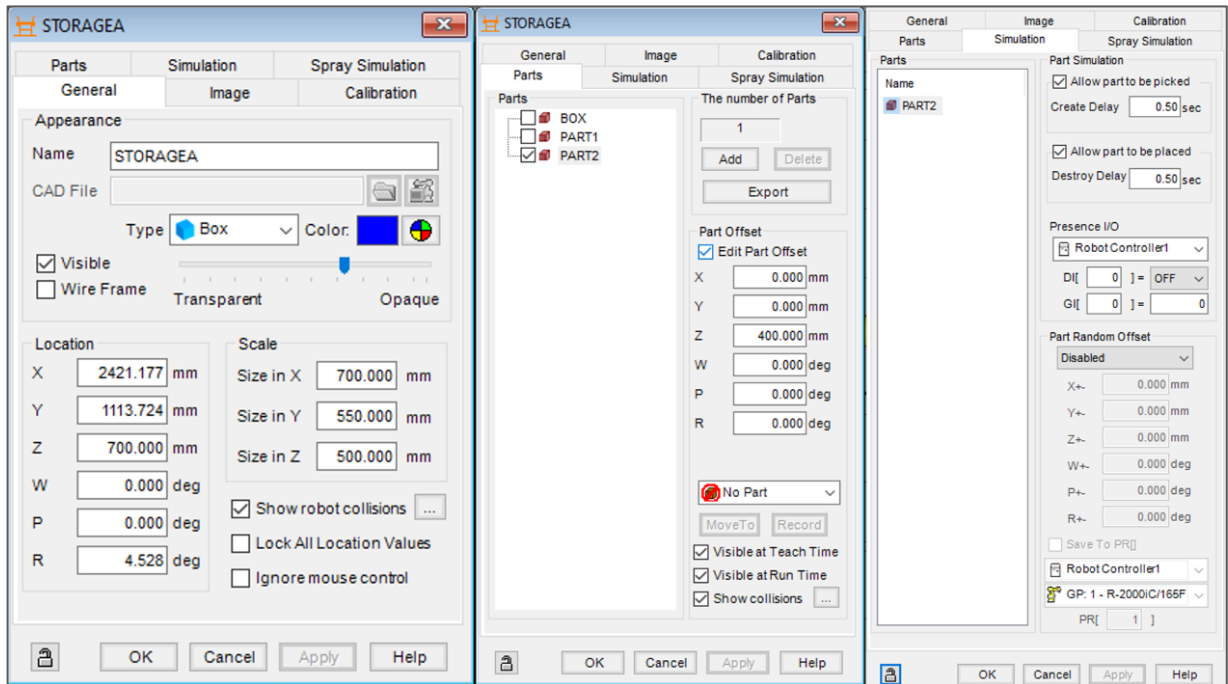


Figure 4.1 Properties menu of the object "StorageA"

Creation of the additional work zone "BaseA" is going by selecting the fixed part "Base" in the "Cell Browser" menu, or selecting the "Base" object in the 3D graphic environment, we can choose the "Copy Base" option to later use "Paste Base" and reconfigure it according to the required needs. We must configure the position of the new fixed part, in addition to the label, to start configuring the rest of the options. We will configure this new base as indicated in the following images:

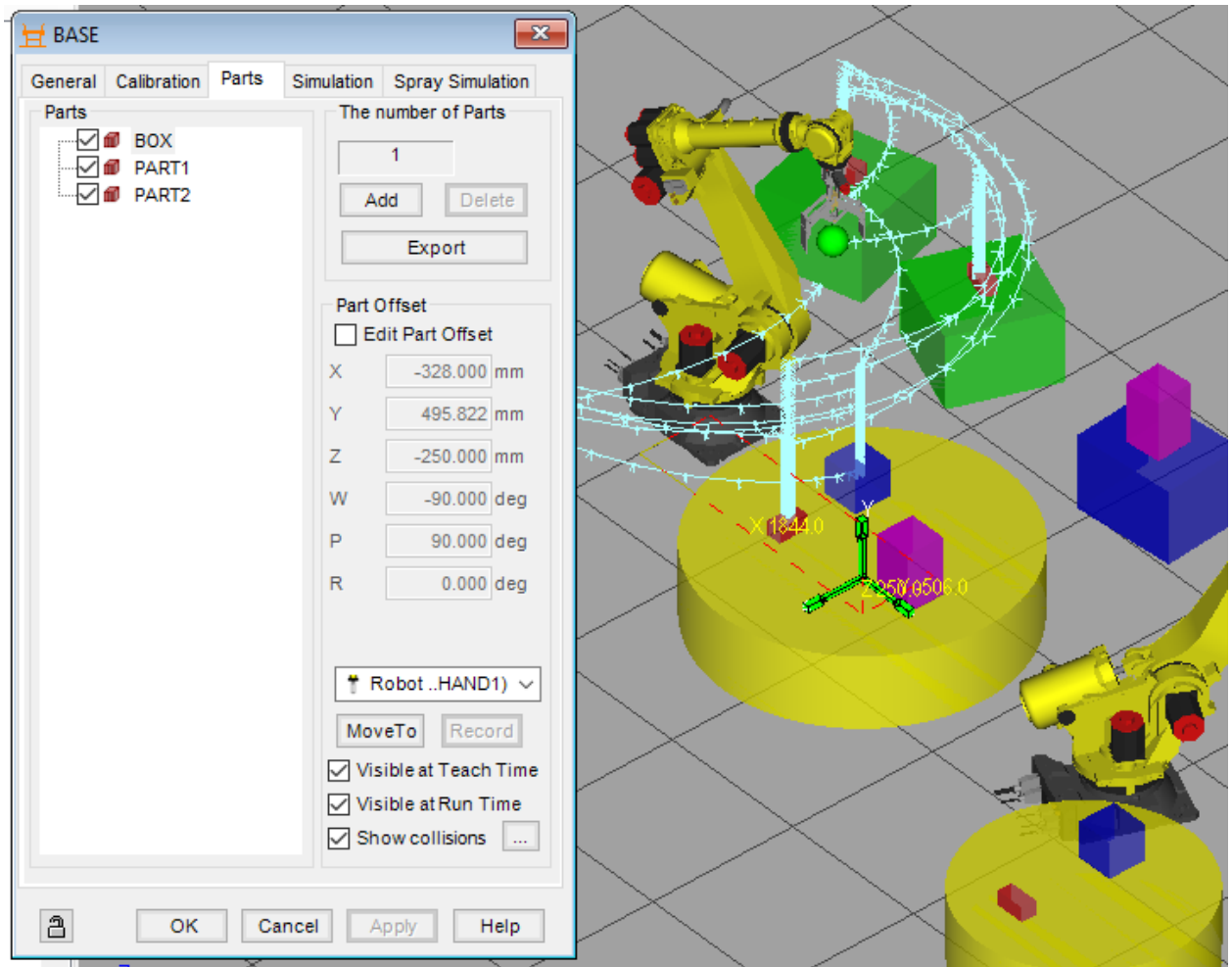


Figure 4.1 Configuration of objects located on the fixed part "Base"

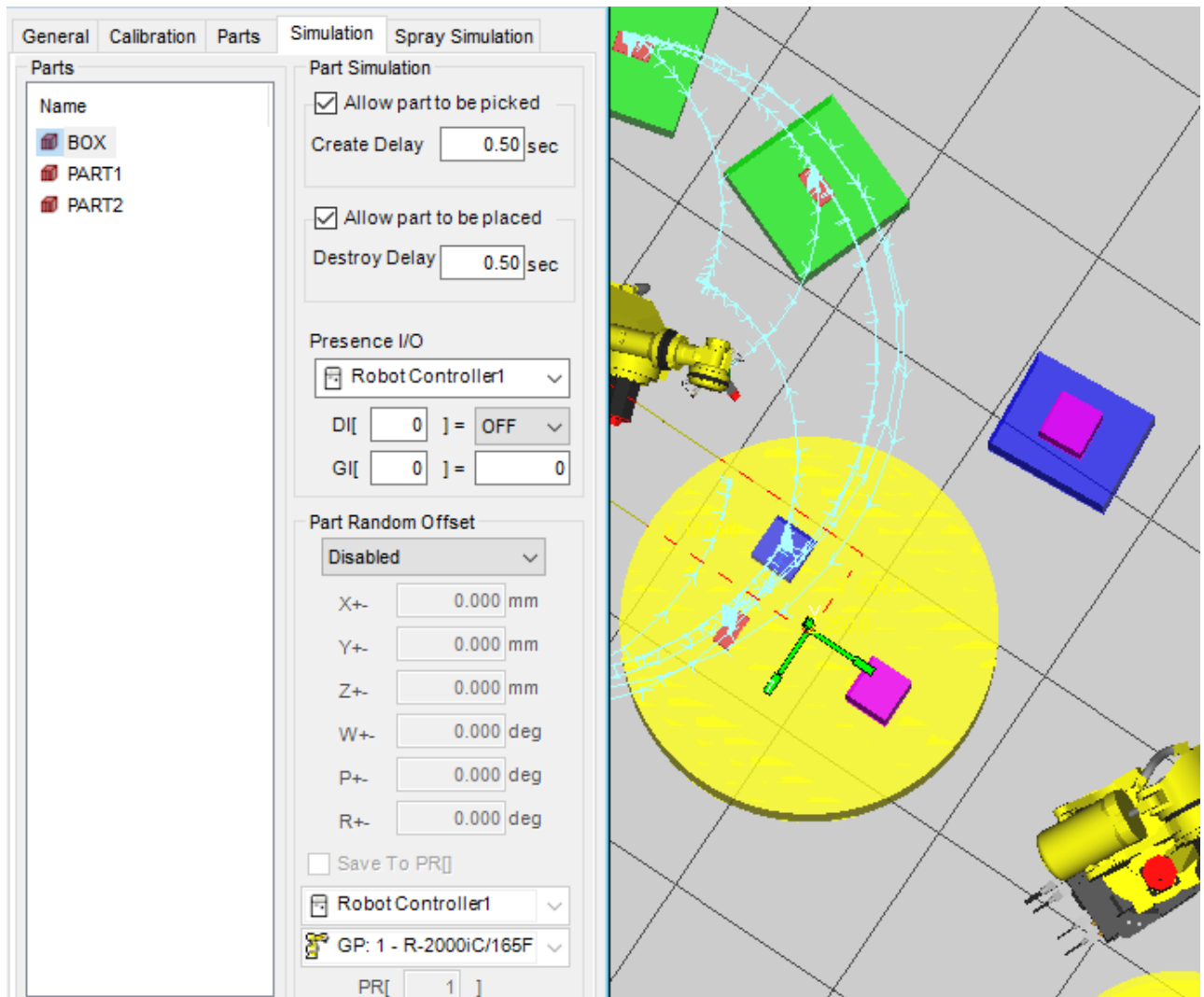


Figure 4.1 Configuration of the simulation options of the fixed part "Base"

4.3 Creation of the extended program

Following the definition of the sequence of actions and movements set forth in section 4.1, we proceed to implement each of the positions and each of the interactions with the objects manipulated by the two robots.

Before starting to implement the manipulator algorithms, we must place special emphasis on the possibility of collisions between the two and control this situation at all times. By the simulation tool and its collision detection, we will be able to redefine the programs at any time and have this collision situation in the cell under control.

The following image shows the collision zone of the arms of the system, outlined with a black line:

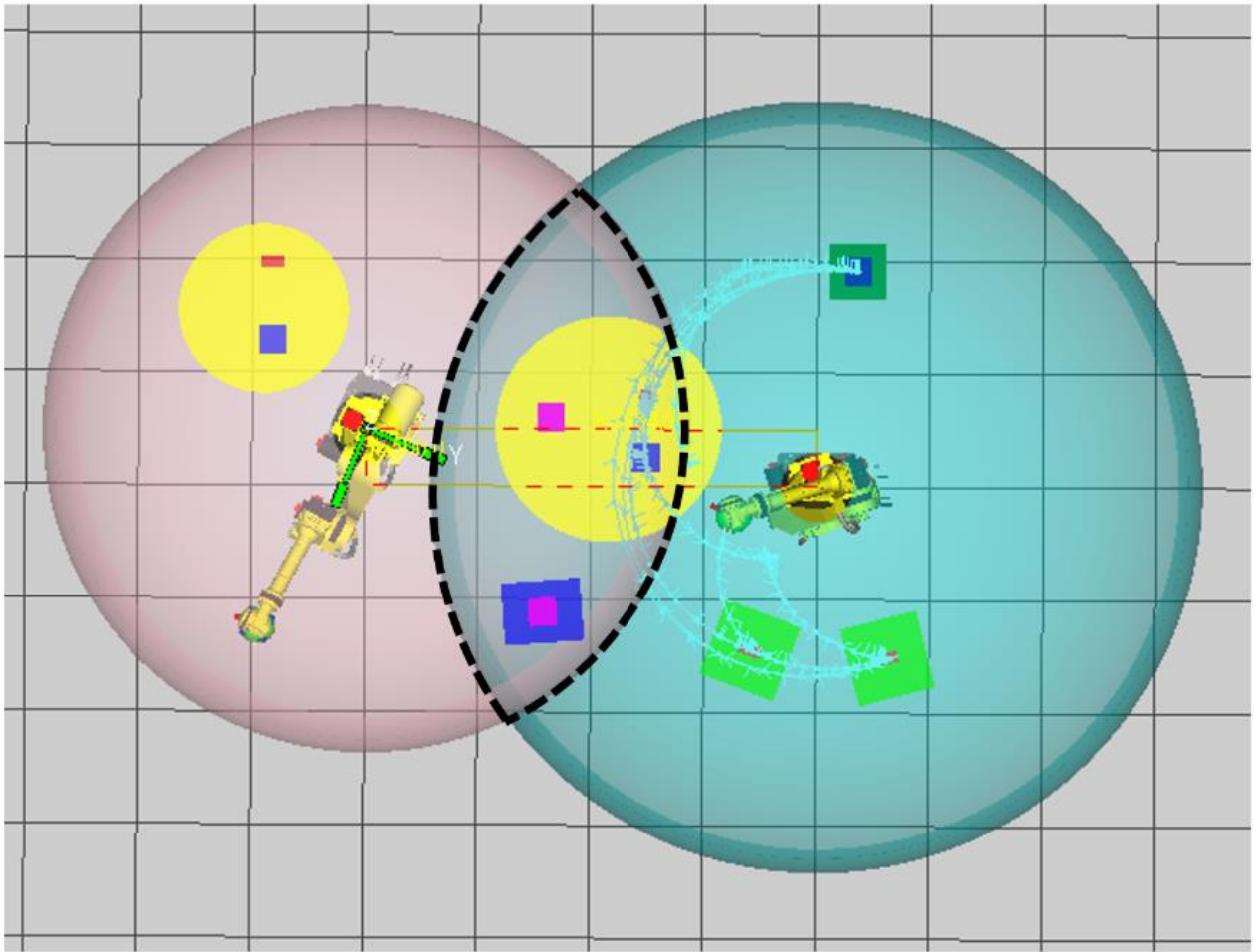


Figure 4.1 Collision zone between manipulators in the system

Always keeping in mind the possibility of collision, we will then define the positions and actions by implementing the algorithms of both robots. Below are some points defined in the simulation tool, illustrated with the 3D environment screenshots. Approach to "StorageA" to capture "Part2":

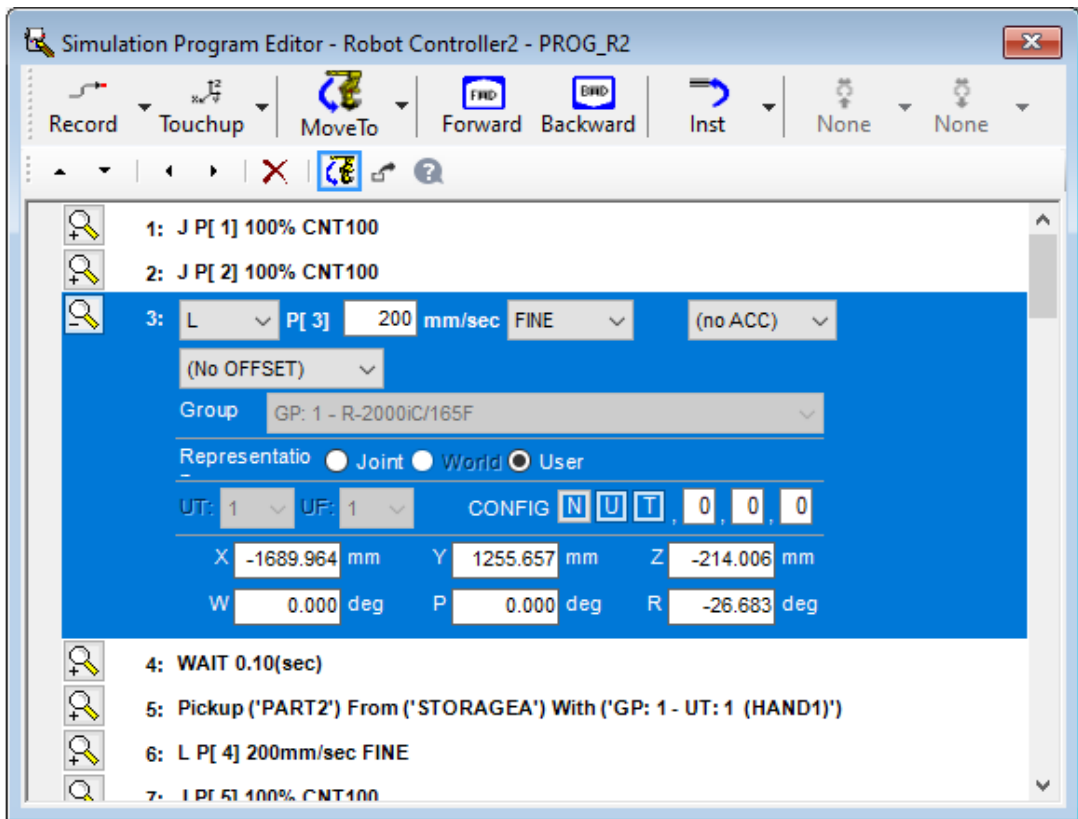
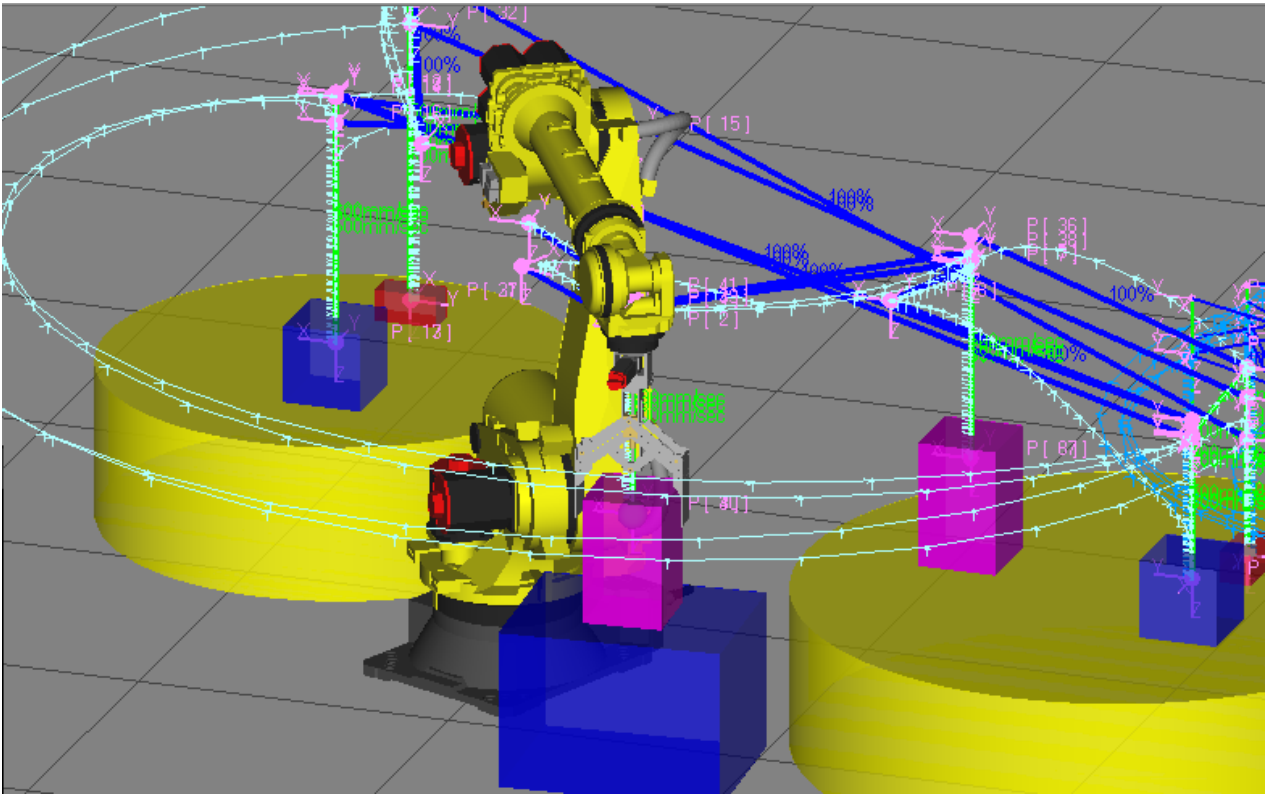


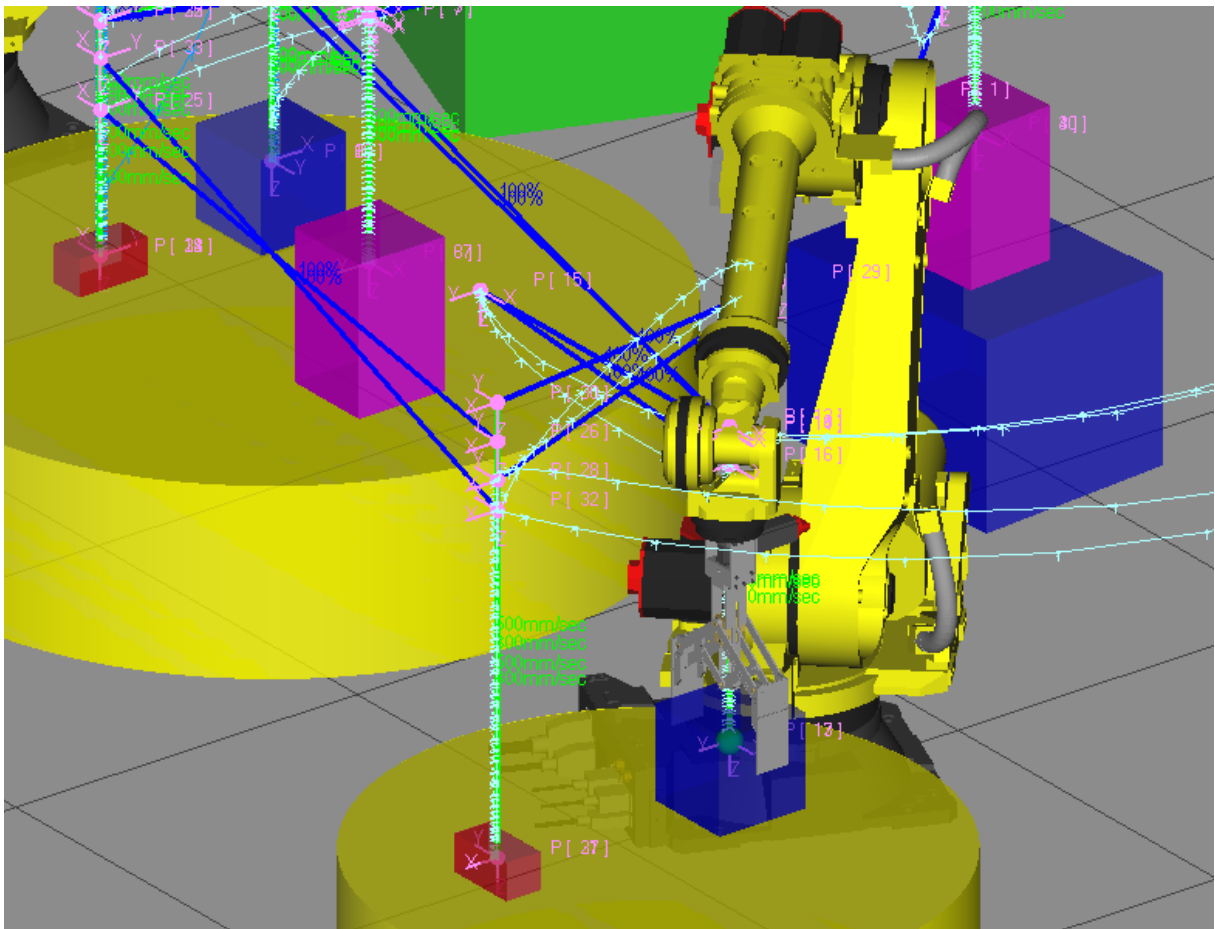
Figure 4.1 Definition of position for the approach to "StorageA"

Robot positioning errors in the simulation environment for the approach to "StorageA" are defined in the table below:

Table 3.1 Robot positioning errors for the approach to "StorageA"

	X (mm)	Y (mm)	Z (mm)	W (deg)	P (deg)	R (deg)
Planned	-1689.965	1255.655	-214.005	0	0	-26.683
Real	-1689.964	1255.657	-214.006	0	0	-26.683
Δ	0.001	0.002	0.001	0	0	0

Approach to "BaseA" to capture "BOX":



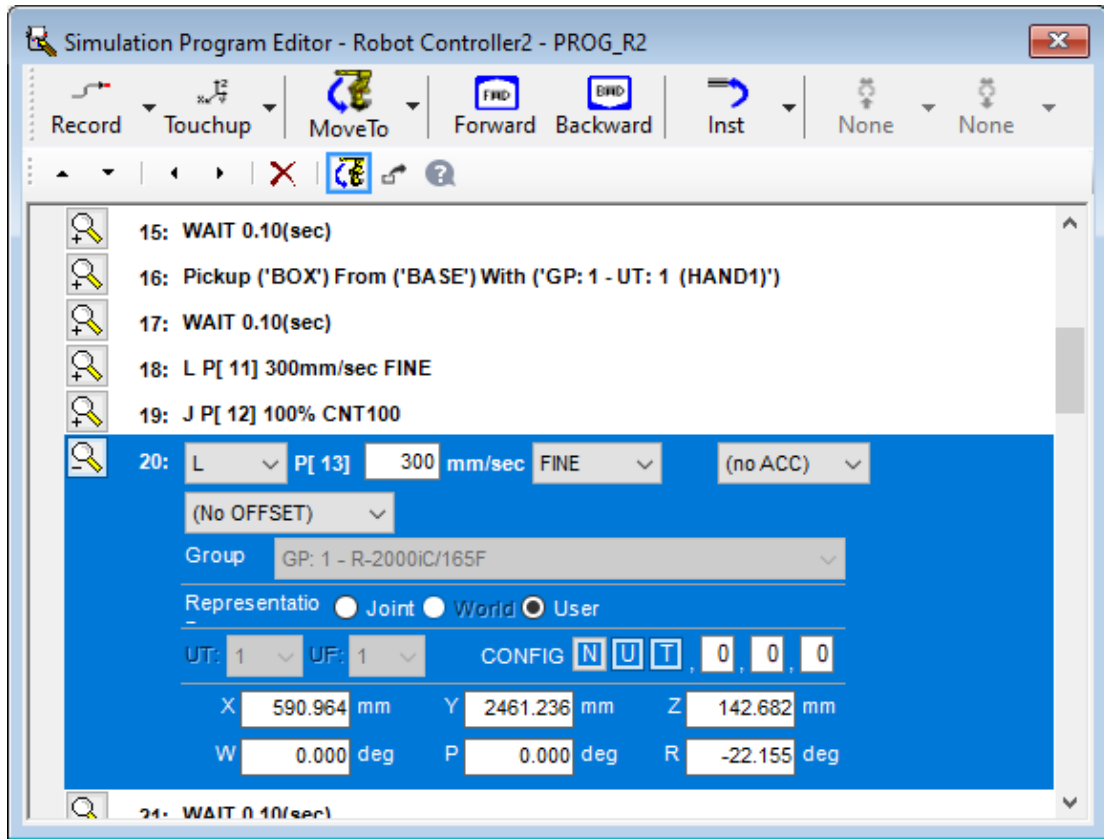


Figure 4.1 Definition of position for approach to "BaseA"

Robot positioning errors in the simulation environment for the approach to "BaseA" are defined in the table below:

Table 3.1 Robot positioning errors for the approach to "BaseA"

	X (mm)	Y (mm)	Z (mm)	W (deg)	P (deg)	R (deg)
Planned	590.965	2461.235	146.682	0	0	-22.155
Real	590.964	2461.236	146.682	0	0	-22.155
Δ	0.001	0.001	0	0	0	0

Deposit of "Part1" on "Base":

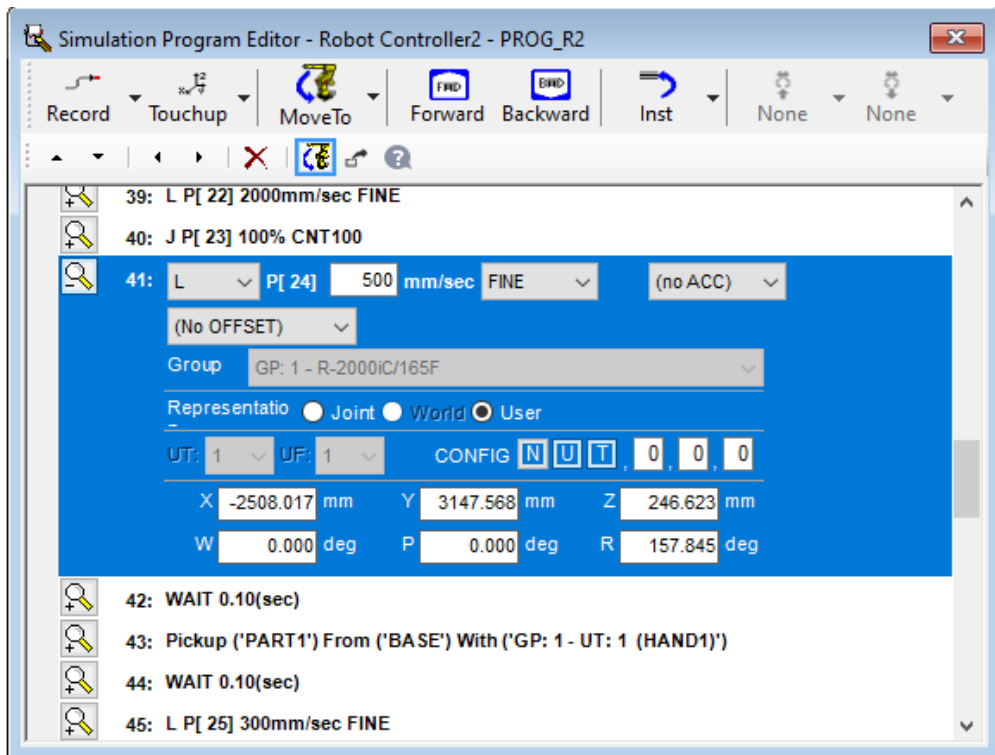
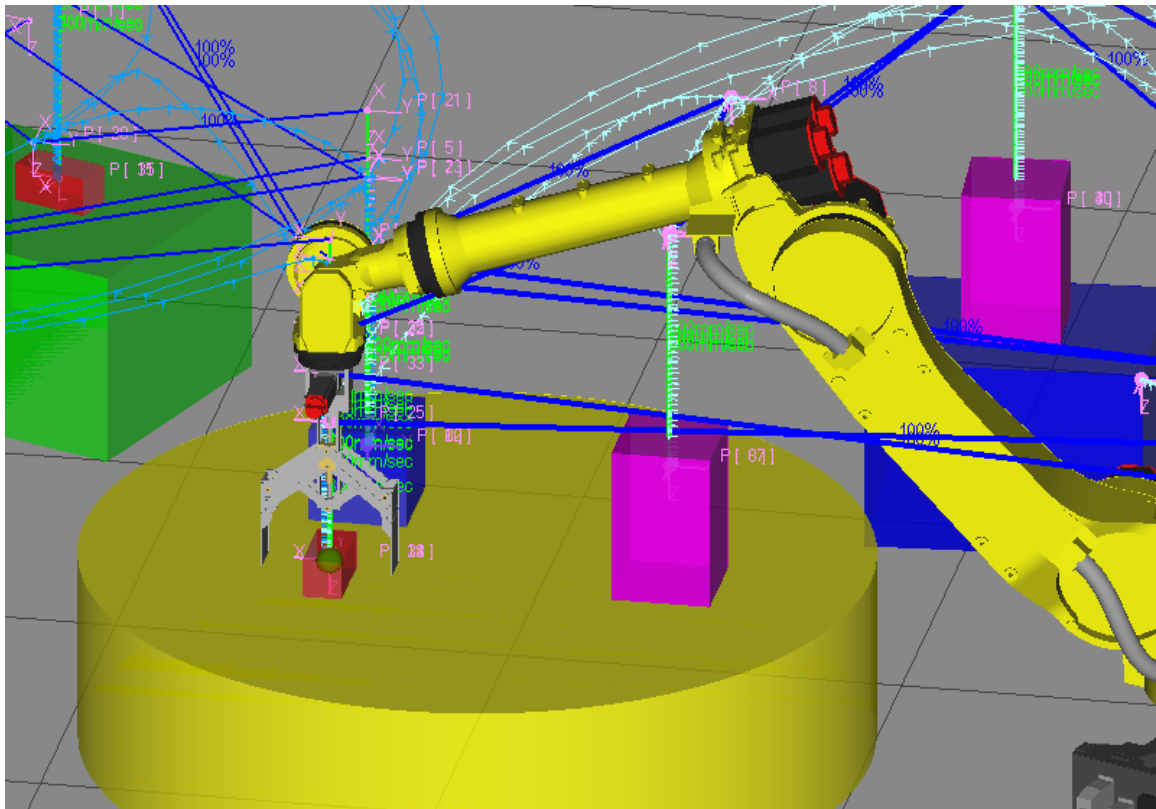


Figure 4.1 Definition of the position for "Part1" on "Base"

Robot positioning errors in the simulation environment for the position for "Part1" on "Base" are defined in the table below:

Table 3.1 Robot positioning errors for the position for "Part1" on "Base"

	X (mm)	Y (mm)	Z (mm)	W (deg)	P (deg)	R (deg)
Planned	-2508.015	3147.570	246.625	0	0	157.845
Real	-2508.017	3147.568	246.623	0	0	157.845
Δ	0.002	0.002	0.002	0	0	0

When we have the programs implemented for the two manipulators, we will proceed to simulate the trajectories and check if collisions occur, to later correct the erroneous defined points, approaches to fixed parts, movement speeds and waiting times in the execution of the process.

The following image shows the implementation of the system with the definitions of the programs of the two robots ("Prog_R1" and "Prog_R2").

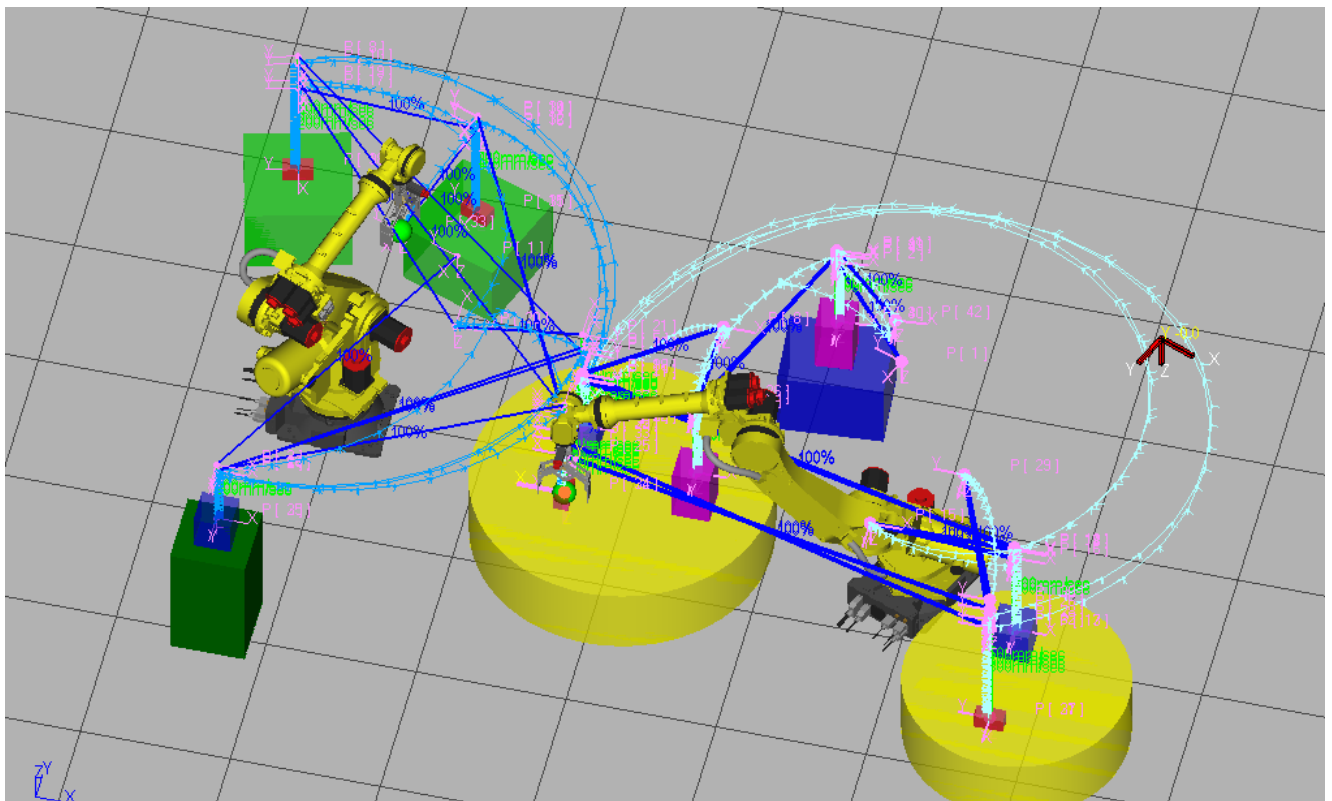


Figure 4.1 Image of the expansion of the system, implemented in "Prog_R1" and "Prog_R2"

4.4 System simulation. Testing

With the new robotic system fully realized, we run the simulation of the process to supervise the programmed system and check the possible collisions, supervising the critical area where these “interferences” between the manipulators can occur.

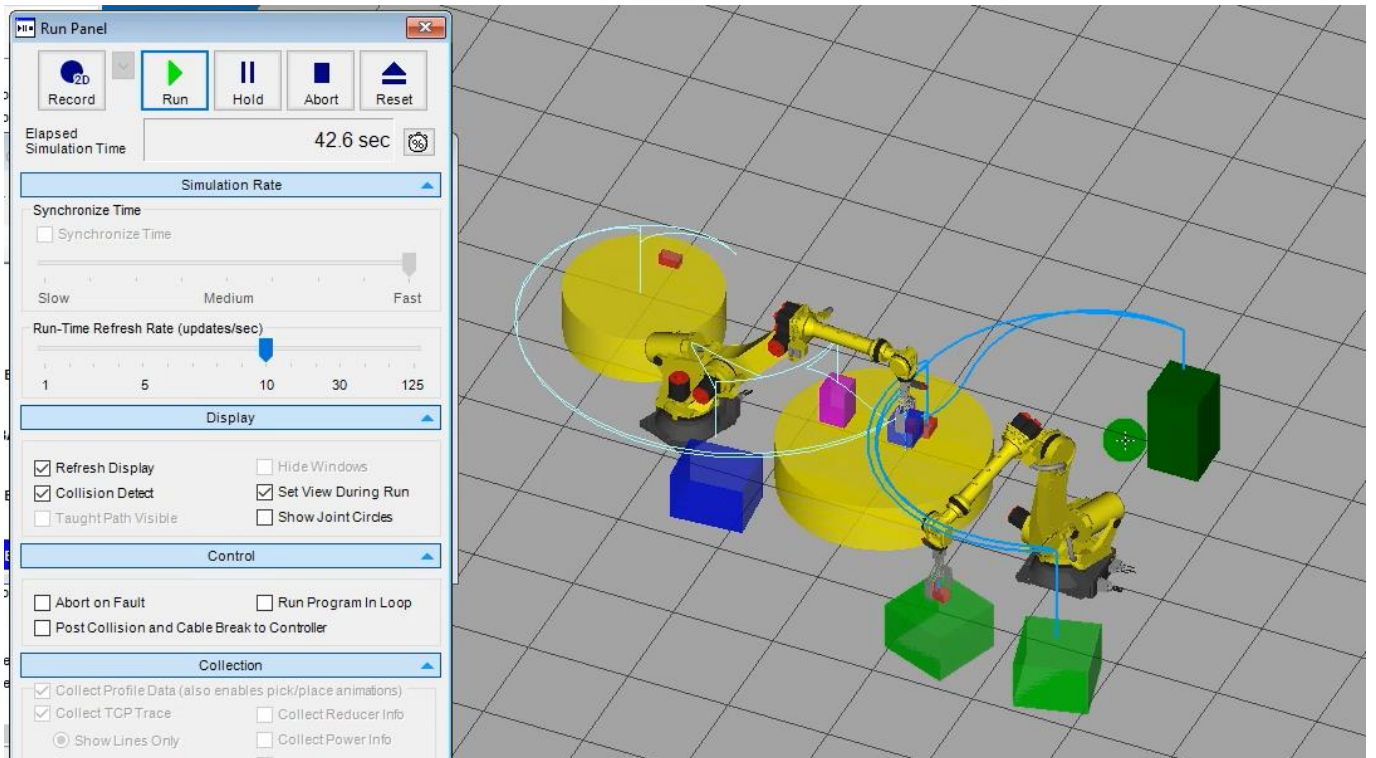


Figure 4.1 Image of the middle of the work cell’s simulation sequence

After numerous tests and various adjustments in the process, the script is completed without getting any collision between the robotic arms.

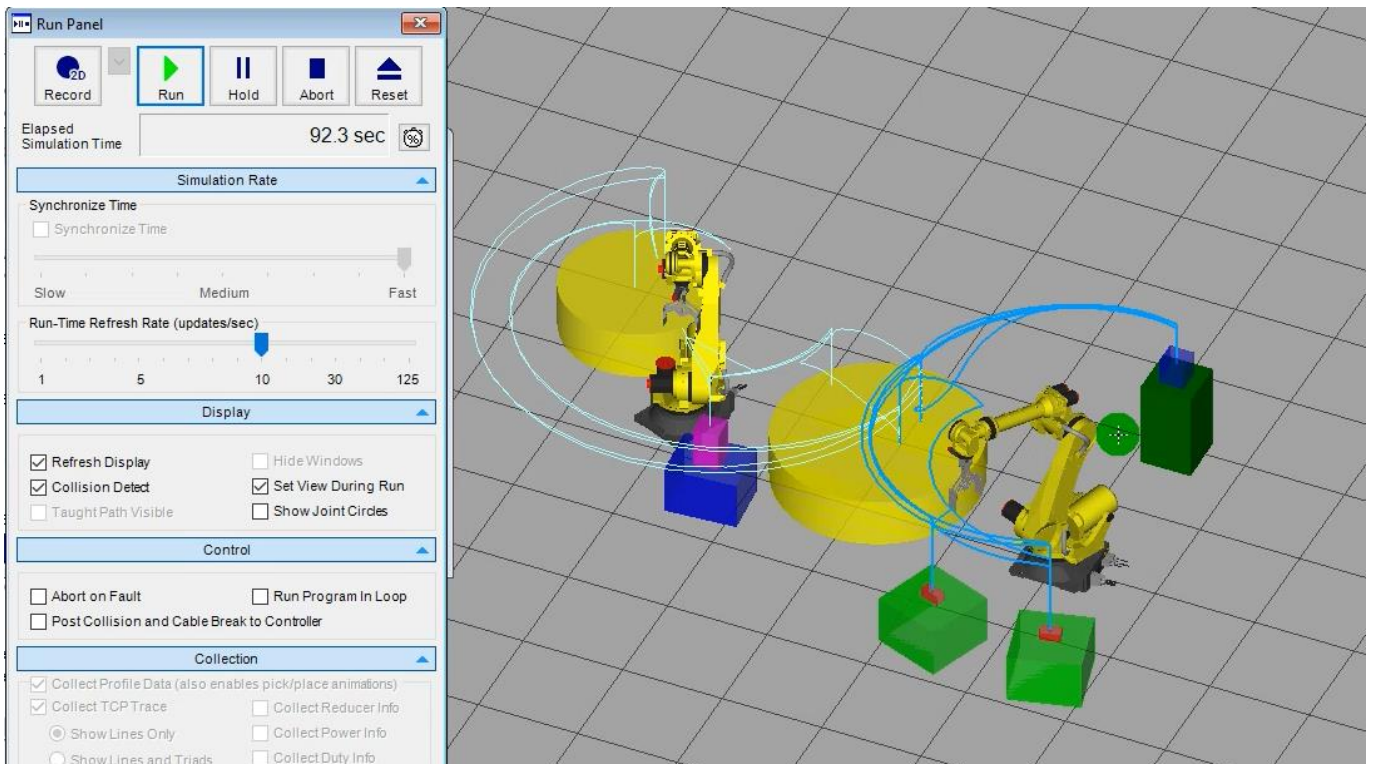


Figure 4.1 Image of the end of the work cell's simulation sequence

4.5 General software development methodology

After the conducted research, we can outline a general methodology for developing software for robotic systems using industrial robots. Each item is a stage, the development of which ensures the successful implementation and correct use of the software. The methodology is presented below:

- A detailed study of software that will be used in the development or testing of a simulation model of an industrial system. Its main features, capabilities and tools.
- Development of a graphical layout of elements that reflects the real location of each element in the robotic system. This helps to start software development correctly. This scheme will subsequently be introduced into the simulation model of the robotic system.
- Building a process diagram of a robotic system. It is recommended to use the “Petri net” diagram. The entire industrial process is represented as possible states (actions for capturing and depositing the objects) and transactions (positioning in the different manipulation spaces). Thus, we can represent each position and movement of the robot in a convenient graphical diagram for a better understanding of the work and, directly, the task of this robotic system. It should be noted that a separate process diagram must be developed

for each robot in the system, otherwise the diagram will be too cumbersome, which makes it difficult to understand the structure of the process.

- The choice of a robot in accordance with the task set by the system. Like Fanuc, almost all companies offer a wide variety of robots. It is very important to choose a robot in accordance with the tasks set by the system. Such as payload capacity, maximum reach, degrees of freedom and other characteristics that are important for the correct implementation of a robot in a given industrial process.

- Definition and configuration of the robot and the controller. It is necessary to choose the version of the robot controller software offered by the application. We have different versions developed by the manufacturer FANUC. Each version has its own characteristics and sets of tools, it is important to configure the controller in accordance with the set of functions that will need to be used to successfully build a simulation system.

- Definition of manipulated objects. Once the robot is configured, we will define the properties of the parts that are going to be manipulated during the simulation.

- Tool definition and configuration. In this section, a tool will be added to the robot's TCP to be able to manipulate the objects defined in the "parts" menu. In the UTOOL tab we will define the actuation point of the gripper. Due to the displacement in the tool configuration, we will define the reference system of the actuation point to specify the movements of the robot so that the gripper can be properly positioned on the objects to be manipulated.

- Creation and configuration of fixed parts. The definition of the fixed parts of the system is essential for the manipulation of the different parts in the cell. In these fixed parts we will position the objects that will be used in the process.

- Definition of user reference system. In the same way as in TP programming, in the HandlingPRO simulation software we can also create reference systems defined by the user "user frames", as long as the needs of the program require it for the convenience of defining points and certain movements (such as an inclined surface defined in the robot's working environment).

- Creation of the program. After creating each of the objects in the robot's work environment, we begin with the implementation of the robotic process program. We can

program the movements of the system by using the tools offered by the application, for example, through the “Teach Program” menu. The definition of points and actions is the basis for developing the manipulator's work algorithm. As an additional utility, we will use the virtual console "Teach Pendant" to be able to do tests by directly entering some coordinates in the "Current Position" tab and later being able to store said position as a new point within the program.

- Simulation of the process. With the robotic system fully realized and the position definition algorithm completed, we run the simulation of the process to monitor the programmed system and each of the planned actions to be performed in the robotic cell.

- Testing and fixing bugs, collisions. With the new robotic system fully realized, we run the simulation of the process to supervise the programmed system and check the possible collisions, supervising the critical area where these “interferences” between the manipulators can occur.

- Implementation of the developed software in a real robot. Simulation software Roboguide from Fanuc allows you to integrate successfully developed software directly into a real robot controller that is part of a developed robotic system.

Upon successful completion of all stages of the developed methodology, we receive the correct software, which is completely ready for implementation in a real robotic system.

4.6 Conclusions

By using the simulation tools, we have been able to analyze any robotic process, virtually, before having the entire real system deployed. We have simulated a "pick and place" type process, characteristic of the industrial sector, checking its operation, monitoring the movement trajectories, evaluating the entire process globally and in particular we have analyzed and verified each one of the specific actions of the system.

Therefore, we can say that we have met the objectives proposed at the beginning, since we have managed to efficiently determine and correct the incidents that could have arisen in an implementation carried out directly on a real environment.

The concept of simultaneous engineering has also been analyzed, and its application to industrial simulation environments has been evaluated. We have been able to verify that it adapts perfectly to the way of working of this engineering concept.

The mathematical aspects of the movements of translation and rotation of a reference system have been introduced, in order to understand the fundamentals of the movements of the manipulator in the work cell.

5 SOLUTION FOR THE TRANSMISSION OF INFORMATION BETWEEN PLC AND ROBOT

A programmable logic controller PLC is used in industrial automation engineering to automate electromechanical processes, such as the control of machinery or assembly lines. Unlike general-purpose computers, the PLC is designed to handle input and output signals and manage processes sequentially.

The PLC has a real-time data management system, where the output results must be produced in response to the input conditions within a limited time, otherwise it will not produce the desired result, among its basic and primary functions is include the sequential control of the processes by manipulating the different actions.

Regarding the operation of these controllers, it is highly accepted due to the high robustness that these components present, either at the high temperatures or the noise that exists in the installation.

Among the advantages of these teams is the possibility of saving time in the preparation of projects, being able to carry out modifications without additional costs. Another advantage of these teams is their size, since they are quite small if we compare them with a PC with similar characteristics, they also have a fairly cheap maintenance, due to the possibility of changing the parts or modules that are affected by a defect. What supposes a very big economic saving in labor and materials.

PLCs are usually classified by their size, there are 3 different modes, the small ones with a number of inputs / outputs less than 500, the medium ones with a number of inputs / outputs greater than 500 and less than 5000, and the large ones with a number of inputs / outputs greater than 5000. For this reason, having a large number of inputs and outputs, the PLC needs a control capable of managing all the actions and communications of the system and therefore a modular design has been chosen, in this way the necessary modules that we need in the application can be added.

The following table shows a representation of the architecture made by the different modules that exist in the PLC:

Table 5.1 Architecture made by the different modules in the PLC

OPERATION PER BIT	0,08 μ s
WORKING MEMORY	150Kbyte
DIGITAL CHANNELS	>1000
ANALOG CHANNELS	>100
TEMPERATURE	~30°C
POWER SUPPLY	24VDC
MODULABLE	>10 stations
PROGRAMMING LANGUAGES	LAD and SCL
COMMUNICATIONS	PROFINET, PROFIBUS y RS232

After the study of the PLCs has been chosen Siemens among the most used PLCs in the market (Siemens, Omron and Telemecanique brands) and because it has more favorable quality / price benefits. This PLC has features that sufficiently meet the specifications detailed above. It should also be noted that the TIA Portal programming software is one of the most powerful and easy to work with, therefore it is the best possible choice even if the price is the highest. Next, we will detail the specifications of it.

The PLCs SIMATICS7-1500 series is a new generation of PLCs and these controllers are programmed using the TIA Portal software, being one of the best programming platforms for PLCs, in terms of the range they occupy we can assure that they are between a medium-high level . One of the best advantages of this PLC is the possibility of expanding the work memory with the insertion of a Flash Epron card of up to 8MB, which gives the controller the possibility of managing a large number of data or variables.

The controller design integrates very interesting devices, such as; a visualization display capable of indicating and diagnosing the operation of the CPU and its modules. The display can be attached and detached from the CPU during its operation and has a protection system by means of a password.

Regarding communication, the PLC integrates a PROFINET interface in each CPU, which ensures very short response times and high precision in the behavior of the PLC. The equipment also has a Web Server that gives the ability to view information in real time. Within the 1500 range there are four types of PLCs capable of working or configuring to the needs of the system we need, these models are:

- Standard CPUs: These CPUs are characterized by their modularity, which have a process module, input-output modules and communication modules.
- Compact CPUs: These CPUs integrate a number of inputs and outputs together with the process module. There is also the possibility of expanding the configuration with input-output and communication modules.
- Safety CPUs: These CPUs are ideal for those safety applications since they are based on a cat.4 safety structure, both in inputs and outputs and information management.
- Extreme conditions CPUs: This type of CPUs are capable of working in temperature conditions between -40°C to 70°C and in very high humidity conditions.

Regarding the choice of our application, we will opt for the standard CPU, since it meets the needs of the system and the price is lower than the other options, as for the benefits of the chosen CPU, they are detailed in the following table:

Table 5.2 The benefits of the chosen CPU

OPERATION PER BIT	0,04 μs
WORKING MEMORY	300Kbyte
DIGITAL CHANNELS	2048
ANALOG CHANNELS	2048
TEMPERATURE	40 $^{\circ}\text{C}$
POWER SUPPLY	24VDC
MODULABLE	32 stations
PROGRAMMING LANGUAGES	LAD and SCL
COMMUNICATIONS	PROFINET, PROFIBUS y RS232

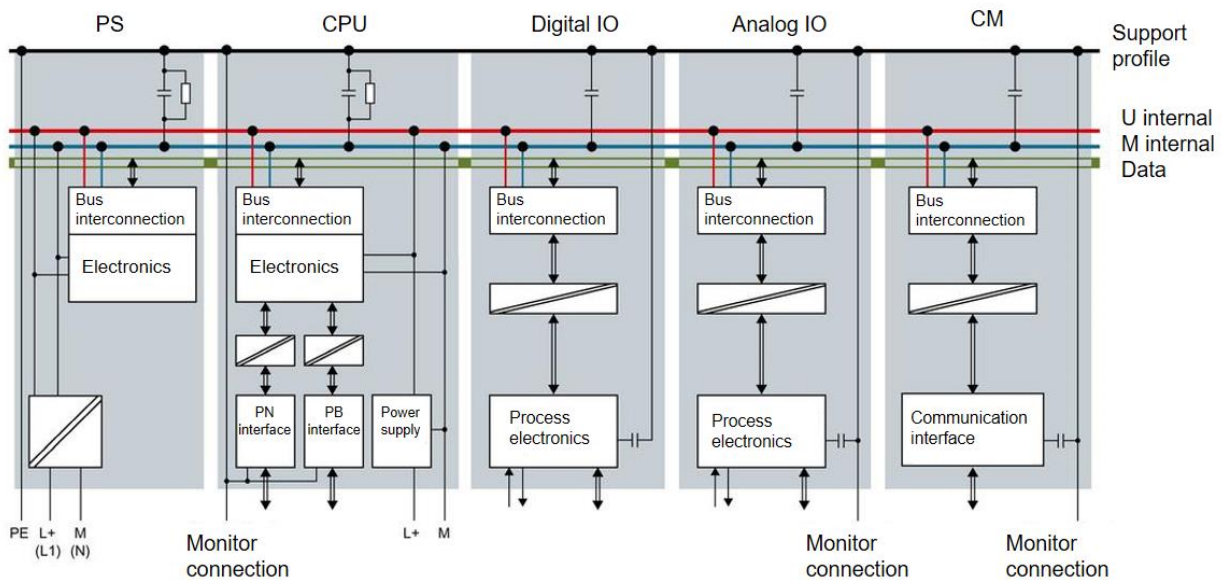


Figure 5.1 Electrical configuration PLC 1500

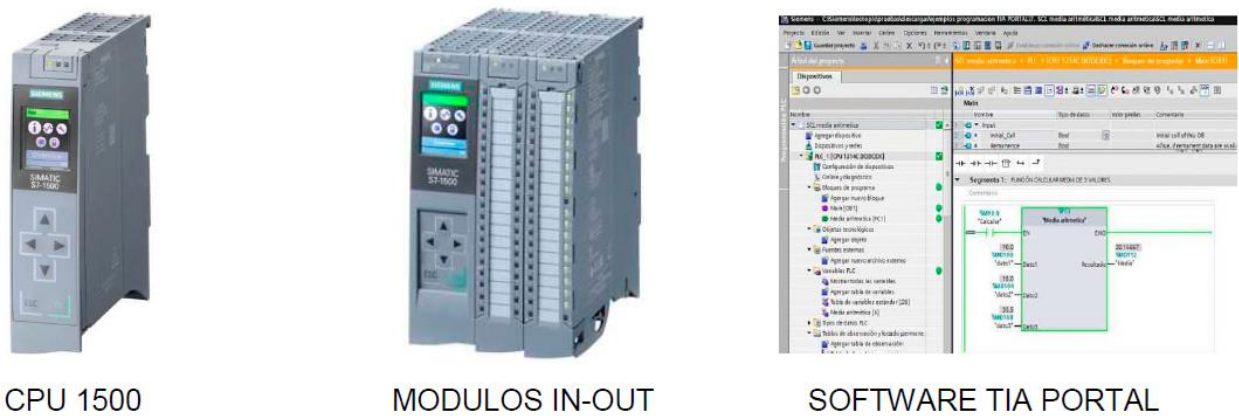


Figure 5.2 Siemens PLC working environment

Safety controllers offer a simple and flexible solution to make a design in the safety of machines or facilities. These safety controllers constitute a modular hardware platform with a high range of modules to perform different tasks, one of their main advantages is that they are easy to operate and can be easily integrated with all the components of the safety controller. These relays have an easy and very versatile user interface, capable of inserting different types of safety at any time, such as; emergency mushrooms, photoelectric barriers, scanners, etc.

The software has a system of permissions per user, this system attributes, depending on the user, a hierarchy within the program, for example, the first user can only see the program without making any type of modification, the second user can open the program and simulate it, while the third user can open, simulate and modify the program.

With this type of security, infringements produced by unqualified operators in the creation of security systems are avoided. Commissioning is simple and fast, saving additional time and therefore money.

In addition, it complies with a maximum safety level and avoids the possibility of error masking and unnecessary emergency stops in the machines, which always entails a reduction in productivity and possible risks produced with these types of stops. These systems also have a checksum system so that if an operator changes the programming of the control unit, it is possible to know when and how it occurred. The characteristics that the safety relay must meet are the following:

Table 5.3 Safety characteristics

CATEGORY	4
PERFOMANCE LEVEL	PL _e
USE TIME	>10 years
COMMUNICATION	Profinet o RS232
TEMPERATURE	30°C
POWER SUPPLY	24VDC
MODULABLE	>5 stations

If we compare diferent safety relays, it is observed that the Sick programmable relay is the most widely used and are the ones that best adapt to the needs of the handling cell, therefore it will be the chosen relay. It's characteristics are defined below.

The Sick safety controller is a central programming element based on category 4 safety. This controller is very easy to configure and assemble since it is modular and you only have to insert modules depending on the needs we have in the system. . The modules that we can find in these devices range from control, emergency stops, restoration function and feedback controls.

The programming interface of this programmable safety relay is Flexi Soft Designer, this interface is very simple and powerful since it allows to simulate the safety system before loading it into the CPU, having this possibility we can see if it is necessary to carry out a ticket expansion or check if security is optimal for our application. In the following table we see the characteristics that the Sick relay meets:

Table 5.4 Safety characteristics

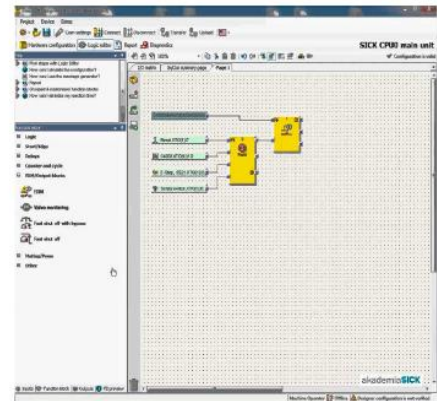
CATEGORY	4
PERFORMANCE LEVEL	PLe
USE TIME	20 years
COMMUNICATION	Profinet
TEMPERATURE	50°C
MODULABLE	12 stations



FX3-CPU3



MODULOS IN-OUT



SOFTWARE FLEXI SOFT DESIGNER

Figure 5.3 Sick programmable safety relay working environment

Solution for the transmission of information between the PLC and the Safety Relay is carried out by the Profinet communication protocol, this type of communication has 4 notable characteristics such as:

- Sturdiness;
- Reliability;
- Simple wiring;
- Fast network diagnostics.

Therefore, the communication system chosen to replace the Profinet protocol must provide the same parameters as the previous one. Regarding the communication between the PLC and the Robot, a digital communication system is used, this system is carried out by means of digital inputs on free potential, that is, by means of relays and without exchanging the voltage sources of each device we exchange the necessary data. In this method we have to bear in mind that sometimes a single bit will indicate a certain action while sometimes it will be necessary to group inputs or outputs 8 by 8, to perform bytes in communication.

This method is not very reliable since if for any reason the signal of a bit fails or is lost, the information we want to send or receive will not be correct. Solution for PLC and Safety Relay is defined below. In these devices it is advisable to use the Modbus system or the Profibus system, these communication protocols are widely used both in the Siemens PLC and in the safety relay, so making a change to the system can be done easily and quickly. These protocols are detailed below:

- Modbus / TCP: it is a 7th level communication protocol, with a master / slave structure, the operation is very similar to Profinet, all the devices have a unique address, when sending the message all the devices are listening but only the equipment where the message is destined acquires the datagram.
- Profibus: This protocol works in a different way since communication is done through a differential voltage system. With this system, each device has an address, but they all share the same bus and all the peripherals also read the entire datagram, although it is only executed by the peripheral where it is addressed.

Advantages of the Modbus protocol:

- Easy to implement.
- Data without restrictions.
- Wiring with RJ45.

Advantages of the Profibus protocol:

- Cyclical and fast bus.
- Ideal for connections of real-time reading systems.
- Connection of networks in parallel.

Regarding the chosen system, it must be taken into account that the corresponding communication cards must be installed in the PLC as well as in the Safety Relay, either Profibus or Modbus. Regarding the wiring, in the Modbus system it would not be necessary to make any changes since the connections are through the RJ45 connector, which is the same as the one used in the Profinet system.

On the other hand, if the Profibus system is used, all connections must be changed to DB9 connectors, and it is also advisable to change the wiring to that corresponding to the Profibus system. Solution for PLC and Robot is defined next.

For this communication a digital communication by bits is used, this type of communication is not too secure, therefore it is advisable to change to a Profinet type communication, Profinet communication is ideal to carry out the transfer of information and have all the communications of the cell welding with the same protocol. Advantages of the Profinet protocol:

- High security of communications.
- Ethernet communication compatibility.
- Remote connection capacity through an industrial network, (VPM).

The only drawback in implementing this type of communication with the current system is installing a Profinet module in the robot and changing the wiring from the robot to the PLC.

EVALUATION OF RESULTS AND CONCLUSIONS

For the development of this final master's degree project, the operation of the FANUC Roboguide simulation application has been learned and analyzed, one of the most used tools in real environments to carry out the “offline” programming of industrial robotic cells. An industrial robot, used in productive environments, with high performance and wide versatility in robotic production lines, has been chosen, thanks to its degrees of freedom, payload capacity and handling range.

A work cell has been programmed, defining each of the positions and each of the actions carried out by the manipulators. For this, the tools available in the visual interface offered by the simulation application (teach mode) have been used, to later simulate the entire process using the same tool (run mode).

With the simulation of the process, we have specified position errors in the system, improved the precision of approaches to objects, detecting collisions produced with objects, fixed parts and other robots in the environment. These detected errors have been corrected in the simulation tool itself, to later carry out more tests and adjust the system until the required performance is achieved.

The work cell has been redesigned representing an extension of the system, adding a second manipulator in addition to new objects. The possibility of collision between the manipulators in the expansion of the cell has been analyzed and controlled and it has been learned to master this situation, implementing the programs of both without collisions occurring.

Possible extensions to this system would be: integration with a vision system for automatic positioning, integration with sensors to perform mechanized tasks dependent on the input and output registers and the application of a simulation tool similar to a robot to be able to implement and perform system calibration with actual position.

Appendix A

Robotic system software for the first simulation

A.1 Main program

```

/PROG PROG_MAIN
/ATTR
OWNER          = MNEDITOR;
COMMENT        = "";
PROG_SIZE      = 4637;
CREATE         = DATE 20-09-03 TIME 11:20:46;
MODIFIED      = DATE 20-09-03 TIME 11:20:46;
FILE_NAME     = ;
VERSION       = 0;
LINE_COUNT    = 109;
MEMORY_SIZE   = 4957;
PROTECT       = READ_WRITE;
TCD: STACK_SIZE = 0,
    TASK_PRIORITY = 50,
    TIME_SLICE = 0,
    BUSY_LAMP_OFF = 0,
    ABORT_REQUEST = 0,
    PAUSE_REQUEST = 0;
DEFAULT_GROUP = 1,*,*,*,*;
CONTROL_CODE  = 00000000 00000000;
/MN
1: !FANUC America Corp. ;
2: !ROBOGUIDE Generated This TPP ;
3: !Run SimPRO.cf to setup frame and ;
4: UTOOL_NUM[GP1]=1 ;
5: UFRAME_NUM[GP1]=1 ;

```

6:J P[1] 100% FINE ACC125 ;
7:L P[2] 200mm/sec FINE ;
8: WAIT .50(sec) ;
9: ! Pickup ('BOX') From ('STORAGE1' ;
10: !WAIT 0.00 (sec) ;
11: WAIT .50(sec) ;
12:L P[3] 200mm/sec FINE ACC50 ;
13:J P[4] 100% FINE ;
14:J P[5] 100% FINE ;
15:L P[6] 400mm/sec FINE ;
16: WAIT .50(sec) ;
17: ! Drop ('BOX') From ('GP: 1 - UT: ;
18: !WAIT 0.00 (sec) ;
19: WAIT .50(sec) ;
20:L P[7] 200mm/sec FINE ;
21:J P[8] 100% CNT100 ;
22:L P[9] 200mm/sec FINE Tool_Offset,PR[1] ;
23: WAIT .50(sec) ;
24: ! Pickup ('BOX') From ('STORAGE2' ;
25: !WAIT 0.00 (sec) ;
26: WAIT .50(sec) ;
27:L P[10] 200mm/sec FINE ;
28:J P[11] 100% CNT100 Tool_Offset,PR[1] ;
29:L P[12] 200mm/sec FINE ;
30: WAIT .50(sec) ;
31: ! Drop ('BOX') From ('GP: 1 - UT: ;
32: !WAIT 0.00 (sec) ;
33: WAIT .50(sec) ;
34:L P[13] 400mm/sec FINE ;
35:J P[14] 100% CNT100 ;

36:L P[15] 200mm/sec FINE ;
37: WAIT .50(sec) ;
38: ! Pickup ('BOX') From ('STORAGE3' ;
39: !WAIT 0.00 (sec) ;
40: WAIT .50(sec) ;
41:L P[16] 200mm/sec FINE ;
42:J P[17] 100% CNT100 ;
43:L P[18] 200mm/sec FINE ;
44: WAIT .50(sec) ;
45: ! Drop ('BOX') From ('GP: 1 - UT: ;
46: !WAIT 0.00 (sec) ;
47: WAIT .50(sec) ;
48:L P[19] 200mm/sec CNT100 ;
49:J P[20] 100% CNT100 ;
50:J P[21] 100% CNT100 ;
51:L P[22] 200mm/sec FINE ;
52: WAIT .50(sec) ;
53: ! Pickup ('PART1') From ('TABLE2' ;
54: !WAIT 0.00 (sec) ;
55: WAIT .50(sec) ;
56:L P[23] 200mm/sec FINE Tool_Offset,PR[1] ;
57:J P[24] 100% CNT100 ;
58:J P[25] 100% CNT100 Tool_Offset,PR[1] ;
59:L P[26] 200mm/sec FINE ;
60: WAIT .50(sec) ;
61: ! Drop ('PART1') From ('GP: 1 - U ;
62: !WAIT 0.00 (sec) ;
63: WAIT .50(sec) ;
64:L P[27] 400mm/sec FINE Tool_Offset,PR[1] ;
65:J P[28] 100% CNT100 ;

66:L P[29] 200mm/sec FINE ;
67: WAIT .50(sec) ;
68: ! Pickup ('PART1') From ('TABLE1' ;
69: !WAIT 0.00 (sec) ;
70: WAIT .50(sec) ;
71:L P[30] 400mm/sec FINE ;
72:J P[31] 100% CNT100 ;
73:L P[32] 400mm/sec FINE ;
74: WAIT .50(sec) ;
75: ! Drop ('PART1') From ('GP: 1 - U ;
76: !WAIT 0.00 (sec) ;
77: WAIT .50(sec) ;
78:L P[33] 2000mm/sec FINE ;
79:J P[34] 100% CNT100 ;
80: WAIT 2.00(sec) ;
81:J P[35] 100% CNT100 ;
82:L P[36] 100mm/sec FINE ;
83: WAIT .50(sec) ;
84: ! Pickup ('BOX') From ('BASE') Wi ;
85: !WAIT 0.00 (sec) ;
86: WAIT .50(sec) ;
87:L P[37] 100mm/sec FINE ;
88:J P[38] 80% CNT100 ;
89:L P[39] 100mm/sec FINE ;
90: WAIT .50(sec) ;
91: ! Drop ('BOX') From ('GP: 1 - UT: ;
92: !WAIT 0.00 (sec) ;
93: WAIT .50(sec) ;
94:L P[40] 200mm/sec FINE Tool_Offset,PR[1] ;
95:J P[41] 100% CNT100 ;

```

96:L P[42] 200mm/sec FINE ;
97: WAIT .50(sec) ;
98: ! Pickup ('PART1') From ('BASE') ;
99: !WAIT 0.00 (sec) ;
100: WAIT .50(sec) ;
101:L P[43] 80mm/sec CNT100 ;
102:J P[44] 80% CNT100 ;
103:L P[45] 100mm/sec FINE ;
104: WAIT .50(sec) ;
105: ! Drop ('PART1') From ('GP: 1 - U ;
106: !WAIT 0.00 (sec) ;
107: WAIT .50(sec) ;
108:L P[46] 200mm/sec FINE ;
109:J P[47] 100% CNT100 ACC50 ;

```

A.2 Definitions of positions for the main program

```
/POS
```

```
P[1]{
```

```
GP1:
```

```

UF : 1, UT : 1,          CONFIG : 'N U T, 0, 0, 0',
X = 1429.000 mm,      Y = 1391.000 mm,      Z = -700.000 mm,
W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

```

```
};
```

```
P[2]{
```

```
GP1:
```

```

UF : 1, UT : 1,          CONFIG : 'N U T, 0, 0, 0',
X = 1429.000 mm,      Y = 1391.000 mm,      Z = 143.000 mm,
W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

```

```
};
```

```
P[3]{
```

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 1391.000 mm, Z = -510.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[4]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -271.000 mm, Y = 1391.000 mm, Z = -510.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[5]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -223.000 mm, Y = 529.000 mm, Z = -583.820 mm,
 W = 0.000 deg, P = 0.000 deg, R = -180.000 deg

};

P[6]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -223.000 mm, Y = 529.000 mm, Z = 147.180 mm,
 W = 0.000 deg, P = 0.000 deg, R = -180.000 deg

};

P[7]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -223.000 mm, Y = 529.000 mm, Z = -317.820 mm,
 W = 0.000 deg, P = 0.000 deg, R = -180.000 deg

};

P[8]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 1895.000 mm, Z = -659.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[9]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 1895.000 mm, Z = -57.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[10]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 1895.000 mm, Z = -570.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[11]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 1391.000 mm, Z = -410.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[12]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 1391.000 mm, Z = 143.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[13]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 1391.000 mm, Z = -600.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[14]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 2400.000 mm, Z = -675.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[15]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 2400.000 mm, Z = -257.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[16]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 2400.000 mm, Z = -630.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[17]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 1895.000 mm, Z = -442.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[18]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 1895.000 mm, Z = -57.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[19]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 1895.000 mm, Z = -571.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[20]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -266.100 mm, Y = 3093.450 mm, Z = -295.370 mm,
 W = 0.000 deg, P = 0.000 deg, R = -14.350 deg

};

P[21]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1992.340 mm, Y = 2676.210 mm, Z = -432.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = -14.350 deg

};

P[22]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1992.340 mm, Y = 2676.210 mm, Z = 246.620 mm,
 W = 0.000 deg, P = 0.000 deg, R = -14.350 deg

};

P[23]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1992.340 mm, Y = 2676.210 mm, Z = -262.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = -14.350 deg

};

P[24]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1571.440 mm, Y = 1311.340 mm, Z = -262.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = -14.350 deg

};

P[25]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 277.000 mm, Y = 529.000 mm, Z = -895.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 90.000 deg

};

P[26]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 277.000 mm, Y = 529.000 mm, Z = 246.620 mm,
 W = 0.000 deg, P = 0.000 deg, R = 90.000 deg

};

P[27]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 277.000 mm, Y = 529.000 mm, Z = -699.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 90.000 deg

};

P[28]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1923.000 mm, Y = 1462.000 mm, Z = -536.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 20.440 deg

};

P[29]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1923.000 mm, Y = 1462.000 mm, Z = 246.620 mm,
 W = 0.000 deg, P = 0.000 deg, R = 20.440 deg

};

P[30]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1923.000 mm, Y = 1462.000 mm, Z = -496.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 20.440 deg

};

P[31]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1992.340 mm, Y = 2676.210 mm, Z = -324.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = -14.350 deg

};

P[32]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1992.340 mm, Y = 2676.210 mm, Z = 246.620 mm,
 W = 0.000 deg, P = 0.000 deg, R = -14.350 deg

};

P[33]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1992.340 mm, Y = 2676.210 mm, Z = -463.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = -14.350 deg

};

P[34]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1100.700 mm, Y = 1618.260 mm, Z = -630.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = -14.350 deg

};

P[35]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -223.000 mm, Y = 529.000 mm, Z = -607.820 mm,
 W = 0.000 deg, P = 0.000 deg, R = -180.000 deg

};

P[36]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -223.000 mm, Y = 529.000 mm, Z = 147.180 mm,
 W = 0.000 deg, P = 0.000 deg, R = -180.000 deg

};

P[37]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -223.000 mm, Y = 529.000 mm, Z = -426.820 mm,
 W = 0.000 deg, P = 0.000 deg, R = -180.000 deg

};

P[38]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 2400.000 mm, Z = -800.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[39]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 2400.000 mm, Z = -257.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[40]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 2400.000 mm, Z = -790.000 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[41]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 277.000 mm, Y = 529.000 mm, Z = -376.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 90.000 deg

};

P[42]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 277.000 mm, Y = 529.000 mm, Z = 246.620 mm,
 W = 0.000 deg, P = 0.000 deg, R = 90.000 deg

};

P[43]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 277.000 mm, Y = 529.000 mm, Z = -289.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 90.000 deg

};

P[44]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1923.000 mm, Y = 1462.000 mm, Z = -440.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 20.440 deg

};

P[45]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1923.000 mm, Y = 1462.000 mm, Z = 246.620 mm,
 W = 0.000 deg, P = 0.000 deg, R = 20.440 deg

};

P[46]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1923.000 mm, Y = 1462.000 mm, Z = -334.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 20.440 deg

};

P[47]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -720.710 mm, Y = 1311.330 mm, Z = -715.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 20.440 deg

};

/END

Appendix B

Robotic system software for the extended simulation

B.1 List of programs

```
/PROG PROG_R1
```

B.2 First robot's program

B.2. Listing of the program

```
/PROG PROG_R1
```

```
/ATTR
```

```
OWNER          = MNEDITOR;
```

```
COMMENT        = "";
```

```
PROG_SIZE      = 3449;
```

```
CREATE         = DATE 20-09-05 TIME 13:45:04;
```

```
MODIFIED = DATE 20-09-05 TIME 13:45:04;
```

```
FILE_NAME      = ;
```

```
VERSION        = 0;
```

```
LINE_COUNT     = 79;
```

```
MEMORY_SIZE    = 3761;
```

```
PROTECT        = READ_WRITE;
```

```
TCD: STACK_SIZE = 0,
```

```
    TASK_PRIORITY = 50,
```

```
    TIME_SLICE = 0,
```

```
    BUSY_LAMP_OFF = 0,
```

```
    ABORT_REQUEST = 0,
```

```
    PAUSE_REQUEST = 0;
```

```
DEFAULT_GROUP  = 1,*,*,*,*;
```

```
CONTROL_CODE   = 00000000 00000000;
```

```
/MN
```

```
1: !FANUC America Corp. ;
2: !ROBOGUIDE Generated This TPP ;
3: !Run SimPRO.cf to setup frame and ;
4: UTOOL_NUM[GP1]=1 ;
5: UFRAME_NUM[GP1]=1 ;
6:J P[1] 100% CNT100 ;
7:J P[2] 100% CNT100 ;
8:L P[3] 200mm/sec FINE ;
9: WAIT .50(sec) ;
10: ! Pickup ('BOX') From ('STORAGE1' ;
11: !WAIT 0.00 (sec) ;
12: WAIT .50(sec) ;
13:L P[4] 200mm/sec FINE ;
14:J P[5] 100% CNT100 ;
15:L P[6] 400mm/sec FINE ;
16: WAIT .50(sec) ;
17: ! Drop ('BOX') From ('GP: 1 - UT: ;
18: !WAIT 0.00 (sec) ;
19: WAIT .50(sec) ;
20:L P[7] 400mm/sec FINE ;
21:J P[8] 100% CNT100 ;
22:L P[9] 400mm/sec FINE ;
23: WAIT .50(sec) ;
24: ! Pickup ('PART1') From ('TABLE2' ;
25: !WAIT 0.00 (sec) ;
26: WAIT 1.50(sec) ;
27:L P[10] 400mm/sec FINE ;
28:J P[11] 100% CNT100 ;
29:L P[12] 200mm/sec FINE ;
30: WAIT .50(sec) ;
```

31: ! Drop ('PART1') From ('GP: 1 - U ;
32: !WAIT 0.00 (sec) ;
33: WAIT .50(sec) ;
34:L P[13] 200mm/sec FINE ;
35:J P[14] 100% CNT100 ;
36:L P[15] 200mm/sec FINE ;
37: WAIT .50(sec) ;
38: ! Pickup ('PART1') From ('TABLE1' ;
39: !WAIT 0.00 (sec) ;
40: WAIT .50(sec) ;
41:L P[16] 200mm/sec FINE ;
42:J P[17] 100% CNT100 ;
43:L P[18] 200mm/sec FINE ;
44: WAIT .50(sec) ;
45: ! Drop ('PART1') From ('GP: 1 - U ;
46: !WAIT 0.00 (sec) ;
47: WAIT .50(sec) ;
48:L P[19] 200mm/sec FINE ;
49:J P[20] 100% CNT100 ;
50: WAIT 1.00(sec) ;
51:J P[21] 100% CNT100 ;
52:L P[22] 400mm/sec FINE ;
53: WAIT .50(sec) ;
54: ! Pickup ('BOX') From ('BASE') Wi ;
55: !WAIT 0.00 (sec) ;
56: WAIT .50(sec) ;
57:L P[23] 200mm/sec FINE ;
58:J P[24] 100% CNT100 ;
59:L P[25] 200mm/sec FINE ;
60: WAIT .50(sec) ;

```

61: ! Drop ('BOX') From ('GP: 1 - UT: ;
62: !WAIT 0.00 (sec) ;
63: WAIT .50(sec) ;
64:L P[26] 200mm/sec FINE ;
65:J P[27] 100% CNT100 ;
66:L P[28] 400mm/sec FINE ;
67: WAIT .50(sec) ;
68: ! Pickup ('PART1') From ('BASE') ;
69: !WAIT 0.00 (sec) ;
70: WAIT .50(sec) ;
71:L P[29] 200mm/sec FINE ;
72:J P[30] 100% CNT100 ;
73:L P[31] 400mm/sec FINE ;
74: WAIT .50(sec) ;
75: ! Drop ('PART1') From ('GP: 1 - U ;
76: !WAIT 0.00 (sec) ;
77: WAIT .50(sec) ;
78:L P[32] 400mm/sec FINE ;
79:J P[33] 100% CNT100 ;

```

B.2.2 Definitions of positions for the program

/POS

P[1]{

GP1:

```

UF : 1, UT : 1,          CONFIG : 'N U T, 0, 0, 0',
X = -720.710 mm,      Y = 1311.330 mm,      Z = -715.380 mm,
W = 0.000 deg, P = 0.000 deg, R = 20.440 deg

```

};

P[2]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 1429.000 mm, Y = 2400.000 mm, Z = -792.320 mm,
W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[3]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 1429.000 mm, Y = 2400.000 mm, Z = -257.320 mm,
W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[4]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 1429.000 mm, Y = 2400.000 mm, Z = -728.320 mm,
W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[5]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -223.000 mm, Y = 356.000 mm, Z = -635.140 mm,
W = 0.000 deg, P = 0.000 deg, R = -180.000 deg

};

P[6]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -223.000 mm, Y = 356.000 mm, Z = 146.860 mm,
W = 0.000 deg, P = 0.000 deg, R = 180.000 deg

};

P[7]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -223.000 mm, Y = 356.000 mm, Z = -579.140 mm,
 W = 0.000 deg, P = 0.000 deg, R = 180.000 deg

};

P[8]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1992.340 mm, Y = 2676.220 mm, Z = -878.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = -14.350 deg

};

P[9]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1992.340 mm, Y = 2676.210 mm, Z = 246.620 mm,
 W = 0.000 deg, P = 0.000 deg, R = -14.350 deg

};

P[10]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1992.340 mm, Y = 2676.220 mm, Z = -801.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = -14.350 deg

};

P[11]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 277.000 mm, Y = 356.000 mm, Z = -577.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 90.000 deg

};

P[12]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 277.000 mm, Y = 356.000 mm, Z = 246.620 mm,
W = 0.000 deg, P = 0.000 deg, R = 90.000 deg

};

P[13]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 277.000 mm, Y = 356.000 mm, Z = -479.380 mm,
W = 0.000 deg, P = 0.000 deg, R = 90.000 deg

};

P[14]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -1923.000 mm, Y = 1462.000 mm, Z = -684.380 mm,
W = 0.000 deg, P = 0.000 deg, R = 20.440 deg

};

P[15]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -1923.000 mm, Y = 1462.000 mm, Z = 246.620 mm,
W = 0.000 deg, P = 0.000 deg, R = 20.440 deg

};

P[16]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -1923.000 mm, Y = 1462.000 mm, Z = -560.380 mm,
W = 0.000 deg, P = 0.000 deg, R = 20.440 deg

};

P[17]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1992.340 mm, Y = 2676.210 mm, Z = -561.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = -14.350 deg

};

P[18]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1992.340 mm, Y = 2676.210 mm, Z = 246.620 mm,
 W = 0.000 deg, P = 0.000 deg, R = -14.350 deg

};

P[19]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1992.340 mm, Y = 2676.210 mm, Z = -636.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = -14.350 deg

};

P[20]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -223.000 mm, Y = 1204.000 mm, Z = -607.140 mm,
 W = 0.000 deg, P = 0.000 deg, R = -180.000 deg

};

P[21]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -223.000 mm, Y = 356.000 mm, Z = -764.140 mm,
 W = 0.000 deg, P = 0.000 deg, R = 180.000 deg

};

P[22]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -223.000 mm, Y = 356.000 mm, Z = 146.860 mm,
 W = 0.000 deg, P = 0.000 deg, R = 180.000 deg

};

P[23]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -223.000 mm, Y = 356.000 mm, Z = -585.140 mm,
 W = 0.000 deg, P = 0.000 deg, R = 180.000 deg

};

P[24]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 2400.000 mm, Z = -727.320 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[25]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 2400.000 mm, Z = -257.320 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[26]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 1429.000 mm, Y = 2400.000 mm, Z = -736.320 mm,
 W = 0.000 deg, P = 0.000 deg, R = -90.000 deg

};

P[27]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 277.000 mm, Y = 356.000 mm, Z = -628.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 90.000 deg

};

P[28]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 277.000 mm, Y = 356.000 mm, Z = 246.620 mm,
 W = 0.000 deg, P = 0.000 deg, R = 90.000 deg

};

P[29]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = 277.000 mm, Y = 356.000 mm, Z = -535.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 90.000 deg

};

P[30]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1923.000 mm, Y = 1462.000 mm, Z = -697.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 20.440 deg

};

P[31]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1923.000 mm, Y = 1462.000 mm, Z = 246.620 mm,
 W = 0.000 deg, P = 0.000 deg, R = 20.440 deg

};

P[32]{

GP1:

```

UF : 1, UT : 1,          CONFIG : 'N U T, 0, 0, 0',
X = -1923.000 mm,      Y = 1462.000 mm,      Z = -576.380 mm,
W =  0.000 deg, P =  0.000 deg, R =  20.440 deg
};
P[33]{
  GP1:
    UF : 1, UT : 1,          CONFIG : 'N U T, 0, 0, 0',
    X = -1077.780 mm,      Y = 1776.960 mm,      Z = -428.380 mm,
    W =  0.000 deg, P =  0.000 deg, R =  20.440 deg
};
/END

```

B.3 Second robot's program

B.3.1 Listing of the program

```

/PROG PROG_R2
/ATTR
OWNER          = MNEDITOR;
COMMENT        = "";
PROG_SIZE      = 4130;
CREATE         = DATE 20-09-05 TIME 13:45:20;
MODIFIED = DATE 20-09-05 TIME 13:45:22;
FILE_NAME     = ;
VERSION       = 0;
LINE_COUNT    = 95;
MEMORY_SIZE   = 4378;
PROTECT       = READ_WRITE;
TCD: STACK_SIZE = 0,
    TASK_PRIORITY = 50,
    TIME_SLICE = 0,
    BUSY_LAMP_OFF = 0,

```

```
ABORT_REQUEST = 0,
PAUSE_REQUEST = 0;
DEFAULT_GROUP   = 1,*,*,*,*;
CONTROL_CODE    = 00000000 00000000;
/MN
1: !FANUC America Corp. ;
2: !ROBOGUIDE Generated This TPP ;
3: !Run SimPRO.cf to setup frame and ;
4: UTOOL_NUM[GP1]=1 ;
5: UFRAME_NUM[GP1]=1 ;
6:J P[1] 100% CNT100  ;
7:J P[2] 100% CNT100  ;
8:L P[3] 200mm/sec FINE  ;
9: WAIT  .10(sec) ;
10: ! Pickup ('PART2') From ('STORAGE ;
11: !WAIT 0.00 (sec) ;
12:L P[4] 200mm/sec FINE  ;
13:J P[5] 100% CNT100  ;
14:L P[6] 300mm/sec FINE  ;
15: ! Drop ('PART2') From ('GP: 1 - U ;
16: !WAIT 0.00 (sec) ;
17:L P[7] 300mm/sec FINE  ;
18:J P[8] 100% CNT100  ;
19: WAIT  1.00(sec) ;
20:J P[9] 100% CNT100  ;
21:L P[10] 500mm/sec FINE  ;
22: WAIT  .10(sec) ;
23: ! Pickup ('BOX') From ('BASE') Wi ;
24: !WAIT 0.00 (sec) ;
25: WAIT  .10(sec) ;
```


26:L P[11] 300mm/sec FINE ;
27:J P[12] 100% CNT100 ;
28:L P[13] 300mm/sec FINE ;
29: WAIT .10(sec) ;
30: ! Drop ('BOX') From ('GP: 1 - UT: ;
31: !WAIT 0.00 (sec) ;
32: WAIT .10(sec) ;
33:L P[14] 500mm/sec FINE ;
34:J P[15] 100% CNT100 ;
35: WAIT 1.00(sec) ;
36:J P[16] 100% CNT100 ;
37:L P[17] 500mm/sec FINE ;
38: WAIT .10(sec) ;
39: ! Pickup ('BOX') From ('BASEA') W ;
40: !WAIT 0.00 (sec) ;
41: WAIT 2.00(sec) ;
42:L P[18] 300mm/sec FINE ;
43:J P[19] 100% CNT100 ;
44:L P[20] 300mm/sec FINE ;
45: WAIT .10(sec) ;
46: ! Drop ('BOX') From ('GP: 1 - UT: ;
47: !WAIT 0.00 (sec) ;
48: WAIT .10(sec) ;
49:L P[21] 500mm/sec FINE ;
50:L P[22] 2000mm/sec FINE ;
51:J P[23] 100% CNT100 ;
52:L P[24] 500mm/sec FINE ;
53: WAIT .10(sec) ;
54: ! Pickup ('PART1') From ('BASE') ;
55: !WAIT 0.00 (sec) ;

56: WAIT .10(sec) ;
57:L P[25] 300mm/sec FINE ;
58:J P[26] 100% CNT100 ;
59:L P[27] 300mm/sec FINE ;
60: WAIT .10(sec) ;
61: ! Drop ('PART1') From ('GP: 1 - U ;
62: !WAIT 0.00 (sec) ;
63: WAIT .10(sec) ;
64:L P[28] 500mm/sec FINE ;
65:J P[29] 100% CNT100 ;
66: WAIT 2.00(sec) ;
67:J P[30] 100% CNT100 ;
68:L P[31] 500mm/sec FINE ;
69: WAIT .10(sec) ;
70: ! Pickup ('PART1') From ('BASEA') ;
71: !WAIT 0.00 (sec) ;
72: WAIT .10(sec) ;
73:L P[32] 300mm/sec FINE ;
74:J P[33] 100% CNT100 ;
75:L P[34] 300mm/sec FINE ;
76: WAIT .10(sec) ;
77: ! Drop ('PART1') From ('GP: 1 - U ;
78: !WAIT 0.00 (sec) ;
79: WAIT .10(sec) ;
80:L P[35] 500mm/sec FINE ;
81:J P[36] 100% CNT100 ;
82:L P[37] 500mm/sec FINE ;
83: WAIT .10(sec) ;
84: ! Pickup ('PART2') From ('BASE') ;
85: !WAIT 0.00 (sec) ;

```

86: WAIT .10(sec) ;
87:L P[38] 300mm/sec FINE ;
88:J P[39] 100% CNT100 ;
89:L P[40] 300mm/sec FINE ;
90: WAIT .10(sec) ;
91: ! Drop ('PART2') From ('GP: 1 - U ;
92: !WAIT 0.00 (sec) ;
93: WAIT .10(sec) ;
94:L P[41] 500mm/sec FINE ;
95:J P[42] 100% CNT100 ;

```

B.3.2 Definitions of positions for the program

/POS

P[1]{

GP1:

```

UF : 1, UT : 1,          CONFIG : 'N U T, 0, 0, 0',
X = -903.620 mm,      Y = 1631.980 mm,      Z = -715.380 mm,
W = 0.000 deg, P = 0.000 deg, R = 88.290 deg

```

};

P[2]{

GP1:

```

UF : 1, UT : 1,          CONFIG : 'N U T, 0, 0, 0',
X = -1689.960 mm,      Y = 1255.660 mm,      Z = -814.010 mm,
W = 0.000 deg, P = 0.000 deg, R = -26.680 deg

```

};

P[3]{

GP1:

```

UF : 1, UT : 1,          CONFIG : 'N U T, 0, 0, 0',
X = -1689.960 mm,      Y = 1255.660 mm,      Z = -214.010 mm,
W = 0.000 deg, P = 0.000 deg, R = -26.680 deg

```

};

P[4]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -1689.960 mm, Y = 1255.660 mm, Z = -882.010 mm,
W = 0.000 deg, P = 0.000 deg, R = -26.680 deg

};

P[5]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -1789.800 mm, Y = 2692.100 mm, Z = -653.000 mm,
W = 0.000 deg, P = 0.000 deg, R = -22.200 deg

};

P[6]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -1789.800 mm, Y = 2692.100 mm, Z = -14.000 mm,
W = 0.000 deg, P = 0.000 deg, R = -22.200 deg

};

P[7]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -1789.760 mm, Y = 2692.070 mm, Z = -644.010 mm,
W = 0.000 deg, P = 0.000 deg, R = -22.150 deg

};

P[8]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -1998.110 mm, Y = 2090.210 mm, Z = -749.010 mm,
W = 0.000 deg, P = 0.000 deg, R = -22.150 deg

};

P[9]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -2696.570 mm, Y = 2684.480 mm, Z = -368.890 mm,
W = 0.000 deg, P = 0.000 deg, R = -112.150 deg

};

P[10]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -2696.570 mm, Y = 2684.480 mm, Z = 146.860 mm,
W = 0.000 deg, P = 0.000 deg, R = -112.150 deg

};

P[11]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -2696.570 mm, Y = 2684.480 mm, Z = -310.140 mm,
W = 0.000 deg, P = 0.000 deg, R = -112.150 deg

};

P[12]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 590.960 mm, Y = 2461.240 mm, Z = -671.320 mm,
W = 0.000 deg, P = 0.000 deg, R = -22.150 deg

};

P[13]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 590.960 mm, Y = 2461.240 mm, Z = 142.680 mm,
W = 0.000 deg, P = 0.000 deg, R = -22.150 deg

};

P[14]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 590.960 mm, Y = 2461.240 mm, Z = -654.320 mm,
W = 0.000 deg, P = 0.000 deg, R = -22.150 deg

};

P[15]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -300.010 mm, Y = 2824.020 mm, Z = -654.320 mm,
W = 0.000 deg, P = 0.000 deg, R = -22.150 deg

};

P[16]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 590.960 mm, Y = 2461.240 mm, Z = -572.320 mm,
W = 0.000 deg, P = 0.000 deg, R = -22.150 deg

};

P[17]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 590.960 mm, Y = 2461.240 mm, Z = 142.680 mm,
W = 0.000 deg, P = 0.000 deg, R = -22.150 deg

};

P[18]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 590.960 mm, Y = 2461.240 mm, Z = -657.320 mm,
W = 0.000 deg, P = 0.000 deg, R = -22.150 deg

};

P[19]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -2696.570 mm, Y = 2684.480 mm, Z = -366.140 mm,
 W = 0.000 deg, P = 0.000 deg, R = -112.150 deg

};

P[20]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -2696.570 mm, Y = 2684.480 mm, Z = 146.860 mm,
 W = 0.000 deg, P = 0.000 deg, R = -112.150 deg

};

P[21]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -2696.570 mm, Y = 2684.480 mm, Z = -321.140 mm,
 W = 0.000 deg, P = 0.000 deg, R = -112.150 deg

};

P[22]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -2508.020 mm, Y = 3147.570 mm, Z = -360.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 157.850 deg

};

P[23]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -2508.020 mm, Y = 3147.570 mm, Z = -360.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 157.850 deg

};

P[24]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -2508.020 mm, Y = 3147.570 mm, Z = 246.620 mm,
W = 0.000 deg, P = 0.000 deg, R = 157.850 deg

};

P[25]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -2508.020 mm, Y = 3147.570 mm, Z = -130.380 mm,
W = 0.000 deg, P = 0.000 deg, R = 157.850 deg

};

P[26]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 847.400 mm, Y = 3091.030 mm, Z = -831.380 mm,
W = 0.000 deg, P = 0.000 deg, R = 67.850 deg

};

P[27]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 847.400 mm, Y = 3091.030 mm, Z = 246.620 mm,
W = 0.000 deg, P = 0.000 deg, R = 67.850 deg

};

P[28]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 847.400 mm, Y = 3091.030 mm, Z = -731.380 mm,
W = 0.000 deg, P = 0.000 deg, R = 67.850 deg

};

P[29]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -5.730 mm, Y = 2183.770 mm, Z = -731.380 mm,
W = 0.000 deg, P = 0.000 deg, R = 67.850 deg

};

P[30]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 847.400 mm, Y = 3091.030 mm, Z = -932.380 mm,
W = 0.000 deg, P = 0.000 deg, R = 67.850 deg

};

P[31]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 847.400 mm, Y = 3091.030 mm, Z = 246.620 mm,
W = 0.000 deg, P = 0.000 deg, R = 67.850 deg

};

P[32]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = 847.400 mm, Y = 3091.030 mm, Z = -653.380 mm,
W = 0.000 deg, P = 0.000 deg, R = 67.850 deg

};

P[33]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
X = -2508.020 mm, Y = 3147.570 mm, Z = -264.380 mm,
W = 0.000 deg, P = 0.000 deg, R = 157.850 deg

};

P[34]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -2508.020 mm, Y = 3147.570 mm, Z = 246.620 mm,
 W = 0.000 deg, P = 0.000 deg, R = 157.850 deg

};

P[35]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -2508.020 mm, Y = 3147.570 mm, Z = -363.380 mm,
 W = 0.000 deg, P = 0.000 deg, R = 157.850 deg

};

P[36]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1789.760 mm, Y = 2692.070 mm, Z = -738.010 mm,
 W = 0.000 deg, P = 0.000 deg, R = -22.150 deg

};

P[37]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1789.760 mm, Y = 2692.070 mm, Z = -14.010 mm,
 W = 0.000 deg, P = 0.000 deg, R = -22.150 deg

};

P[38]{

GP1:

UF : 1, UT : 1, CONFIG : 'N U T, 0, 0, 0',
 X = -1789.760 mm, Y = 2692.070 mm, Z = -687.010 mm,
 W = 0.000 deg, P = 0.000 deg, R = -22.150 deg

```
};  
P[39]{  
  GP1:  
    UF : 1, UT : 1,          CONFIG : 'N U T, 0, 0, 0',  
    X = -1689.960 mm,      Y = 1255.660 mm,      Z = -888.010 mm,  
    W = 0.000 deg, P = 0.000 deg, R = -26.680 deg  
};  
P[40]{  
  GP1:  
    UF : 1, UT : 1,          CONFIG : 'N U T, 0, 0, 0',  
    X = -1689.960 mm,      Y = 1255.660 mm,      Z = -214.010 mm,  
    W = 0.000 deg, P = 0.000 deg, R = -26.680 deg  
};  
P[41]{  
  GP1:  
    UF : 1, UT : 1,          CONFIG : 'N U T, 0, 0, 0',  
    X = -1689.960 mm,      Y = 1255.660 mm,      Z = -923.010 mm,  
    W = 0.000 deg, P = 0.000 deg, R = -26.680 deg  
};  
P[42]{  
  GP1:  
    UF : 1, UT : 1,          CONFIG : 'N U T, 0, 0, 0',  
    X = -1037.840 mm,      Y = 1513.250 mm,      Z = -923.010 mm,  
    W = 0.000 deg, P = 0.000 deg, R = -26.680 deg  
};  
/END
```