POLITECNICO
MILANO 1863

Master of Science in Automation and Control Engineering

# Automation of an RFID Measurement Chamber with a Robotic Manipulator

**Supervisors:** Prof. Andrea Maria Zanchettin,
Prof. Dr. Ing. Dieter Uckelmann

Submission by:

Farhan Ahmed Person code: 905740

Academic Year 2019-2020

# Automation of an RFID Measurement Chamber with a Robotic Manipulator

RFID Measuring Chamber Automation, Virtualization and Integration
with Mitsubishi Multi-Axis Robotic Arm.

By

Farhan Ahmed

School of Industrial and Information Engineering

*Automation and Control Engineering*

POLITECNICO DI MILANO

Milano, Italia, 2020

*This work is dedicated to my parents, who always supported and believed in my abilities to do better for mankind. Specially my mother for her endless love, support and encouragement.*

***Farhan Ahmed***

*This page is intentionally left blank*

# Abstract

# Automation of an RFID Measurement Chamber with a Robotic Manipulator

*RFID Measuring Chamber Automation, Virtualization and Integration with Multi-Axis Robotic Arm*

With a fast paced taking over of technology and automation, it is quite clear that robots are overcoming the industry and are a major source of revolution in it. Regardless of the fact how they’re being used, either for industrial machined work, for the ease of mankind or for entertainment purposes, robots have played a vital role in changing humanity for the better. The key difference between them and a human is that robots are not intelligent enough to make their own decisions and they need a proper set of instructions developed by a man. This thesis suggested a scheme of automating a Mitsubishi RV-2SDB multi-axis Melfa series robot with RFID measurement cabinet in an efficient way. The idea is based on development of automation and robotics, the structure of the robot as well as controlling this robot with its dynamic characteristics. Programming of the Robot has been done in MELFA BASIC V language using CIROS Studio tool. A remote access of Lab has also been provided to users so they can perform their experiments without being physically present in the Lab. This feature is implemented by building an application on Visual Studio using Visual Basic Language that communicates with the Robot through TCP/IP.

For this purpose, Mitsubishi RV-2SDB Robot has been programmed so to allow one to send a preferred command to the robot and it recognizes the correct RFID UHF tag, opens the cabinet door, picks up the tag, places the tag in the cabinet, closes the door of cabinet and sends the confirmation command to the user. The user is now able to perform the relative experiment on the tag using the software Tagformance and evaluate the efficiency of transponders so that they could be able to assess their ideal use.

The fact that the equipment used in this project costs a lot, marks a question of its fragility and safe handling. This thesis also suggested a pattern for its efficient usage and safety regulations.

**Key words: Industrial Robotics, CIROS Studio, RFID, MELFA BASIC V, Tagformance, Mitsubishi Robots**

# Acknowledgement

First and foremost, I would like to thank the Almighty God for blessing me with an incredible opportunity to pursue my academic path in a way that I always wished.

I would like to express my sincere gratitude to Prof. Andrea Maria Zanchettin and Prof. Dieter Uckelmann for their guidelines during my thesis and sparing their valuable time for continuous follow up to keep me in the right direction to achieve the desired target.

I would also like to express my heartiest indebtedness to Mr. Hadi Adineh and Mr. Valentin Kammerlohr, for their sincere interest in my thesis and providing me with all their knowledge and time to make this activity possible.

I would like to thank my parents for their persistent belief and support in all the fields of my life, for always being there in my ups and downs, my siblings, my colleagues who were my international family away from home and were always present for my assistance, moral, emotional and mutual support. I want to pay my gratitude specially to my friends Qasim, Shakeel Nazir, Moaaz, Ahmed, Asfandyar, Eleonora, Aatif, Irfan, Munim, Shakeel Haider and Fazul for sharing this beautiful journey with me, for providing all kind of support and for all the amazing moments we spent together during these years.

Finally, I would also like to thank Politecnico Di Milano and its academic and organizational staff for humble dealing in all aspects with international students and for providing me one of the lifetime chances to learn from the best teachers in the world. I would also like to thank Hochschule für Technik Stuttgart staff and Lab members for providing me this opportunity to work on this thesis.

## Table of Contents

# List of Figures

## List of Tables

# 1. Introduction

## 1.1. Background of the thesis

Digitalization in education and research enables new forms of location-independent networking of laboratory infrastructures. This involves mastering technical, organizational and didactic challenges. Digitalization will change the world of work. The Internet of Things (IoT) alone has an economical potential of up to USD 11 trillion by 2020, especially in industry [1]. However, the corresponding digitalization and networking of industrial and logistical systems in academic environments have so far rarely been implemented, although the new requirements involved are best communicated to students in a hands-on environment. Academics are benefitting more than the middle and low-skilled occupational groups in the 4th industrial revolution (Industry 4.0).

The importance of laboratory-based research and teaching is therefore undisputed. Real laboratory structures are personal and cost-intensive and are generally only available to respective research institution. In contrast, purely virtual laboratories offer advantages in terms of security, scalability, remote access and cost efficiency. However, simulations and purely virtual environments cannot replace the success of physical laboratory environments, as these require and promote different knowledge.

In the research project Open Digital Lab for You (DigiLab4U in short), real laboratories are digitized, linked with virtual components and the synergies between the two approaches are explored. Augmented reality can help to close the gap between “virtual” and “real” experience. Methods of engineering education and serious gaming are combined using learning analytics, mixed/augmented reality and open badges to form a unique holistic approach in a hybrid learning and research environment. DigiLab4U provides location-independent access to a digitized and networked learning and research environment.

Multi-user scenarios as well as individual self-directed learning will be supported. For instance, students at HFT Stuttgart can access laboratories at BIBA and University of Parma. The exchange of experiences in research and teaching is promoted beyond the boundaries of individual institutes. As the long title Open Digital Lab for You suggests the inclusion of further laboratories is planned. There is a considerable need for research on this forward-looking approach form a technical, didactic and organizational point of view.

The main goal of Multimodal Learning Analytics (MLA) research is to extend the application of learning analytics tools and services in learning contexts which do not readily provide digital traces and learning data. The learners within this networked lab infrastructure while doing their learning activities will generate a wide range of analogue and digital learning-related data that can be analyzed, and the analytics results used to support and enhance lab-based learning experiences.

## 1.2. Purpose of the Thesis

In DigiLab4U project, we are developing a connected lab infrastructure with learning analytics service as an integral component.

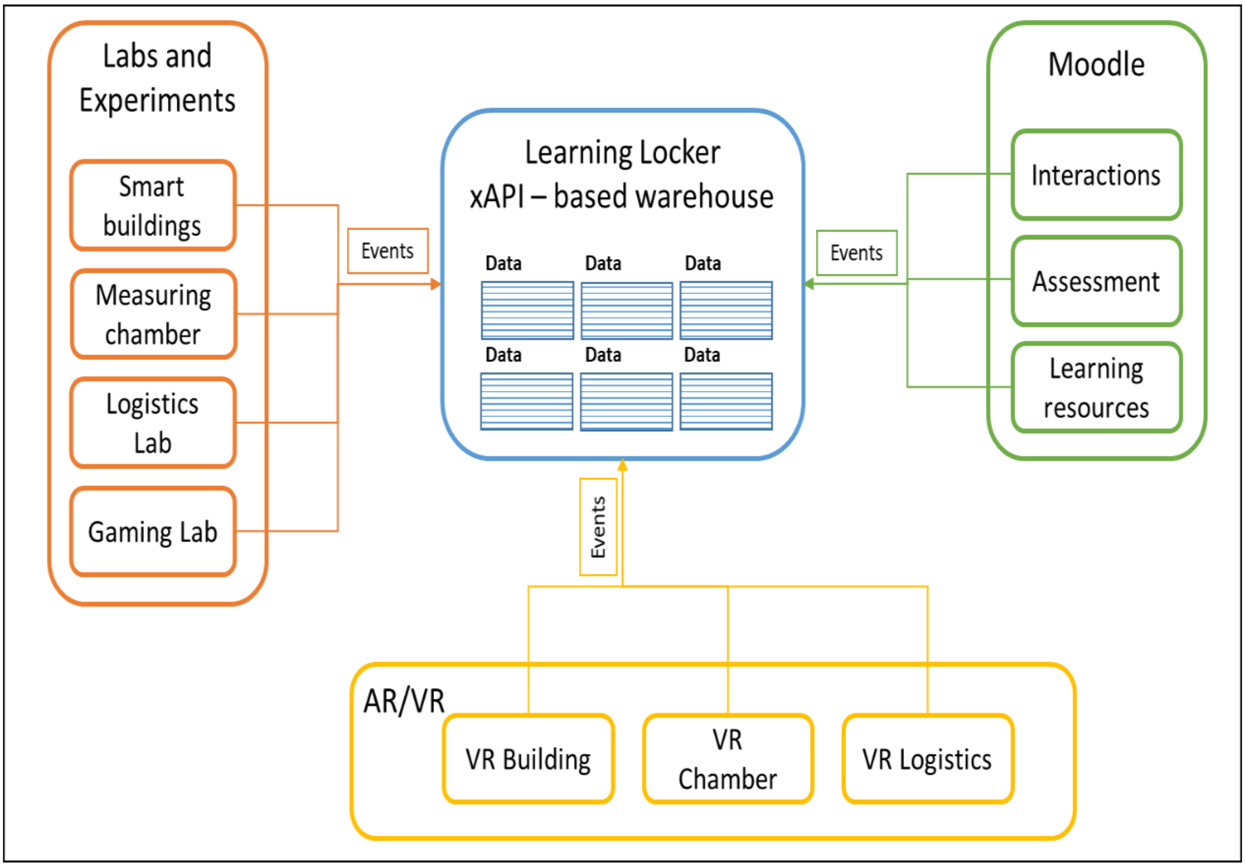


*Figure 1-1: Data aspects of analytics setup in a networked lab infrastructure*

Figure 1-1 sketches out the data aspects of the analytics setup which is part of the learning analytics component of the networked lab infrastructure. The activities in this project include interaction within the learning platform online (including the wide range of learning resources and instructions), physical interactions with the lab equipment, physical presence and movement in the lab including group work, various sensor data from the labs, formative and summative assessment data, and events from the AR/VR equipment installed for the interconnection of the labs.

RFID measurement chamber is a testing environment for RFID UHF-Tags (800-1000 MHz) and it’s one of the industry-related applications that are provided as a learning source for industry 4.0. In this thesis, specific exercises are performed with the RFID measurement cabinet and different transponders on the same substrate or the same transponder on different substrates are checked according to the choice. As an instance, sensitivity and orientation are checked so that the needed power to activate the transponder or theoretical read range can be observed. Finally, the users would be able to compare and evaluate the efficiency of transponders and they are able to assess their ideal use.

Furthermore, a Multi-axis Mitsubishi MELFA series RV-2SDB robot is automated to provide automatic control of identifying and picking up correct RFID tag and feed into the measurement cabinet [2]. In order to provide the correct commands to the robot, an application has been developed on Visual Studio so that efficient communication with the robot through TCP/IP can be made possible.

### 1.3. Outline of the Thesis

This thesis is divided into seven chapters, first being the background and purpose of thesis. This chapter explains the importance of modern era automaton and a need for a unified platform in accordance with virtual or real time structural experimental labs. The main inspiration behind DigiLab4U and projects under development are also discussed here.

There are some requirements and limitations to satisfy/consider before getting hold of the expensive apparatus that contains RFID measurement cabinet and Mitsubishi MELFA RV-2SDB robot. Chapter 2 introduces the partial components of the Lab that are RFID cabinet and RFID tags and brings to lights the true idea of using them. Chapter 3 discusses the major portion of Mitsubishi MELFA RV-2SDB robot and its attributes, usage and benefits. It also elaborates the beginner experience of CIROS Studio by introducing a test exercise and briefly explaining the concepts of MELFA Basic V language program.

Chapter 4 explains the criterion and method of controlling the robot remotely. The basic idea of providing remote access to the users is quite fascinating because it reduces manpower. Before this project, there must always be a person present in the lab who could pick and place the RFID tags himself in the measurement cabinet that is quite a bizarre task. So, the basic purpose of this project is to eliminate the individual effort and take into consideration rapidly growing AI technologies. Automating the robot can provide efficient ways to replace the manpower so that a person no longer must do this job. So, this chapter also describes a way of controlling the robot remotely by introducing PC / Robo Com application.

As we progress through the report, the practical part of the project begins. So, some initialization measures and preferred setting of the robot are discussed in chapter 5. It analyzes the steps from where to start and how to proceed with the robot in detail. To send online commands to the robot and its automatic functionality is also discussed here.

Chapter 6 will discuss the modelling of my whole project on CIROS Studio and programming simulation analysis. It also provides flowchart analysis of complete task.

At the end of the thesis, further techniques for the enhancement of precision/accuracy will be discussed. There are several methods to increase stability of the robot while picking and placement of tags, there are some risks as well. In chapter seven, the future measures and work will be discussed. A few propositions will be presented on how to make the overall lab structure safer and more efficient.

# 2. RFID Measurement Cabinet

This chapter will discuss about the main RFID Measurement Cabinet key features, its structure and the components related to it.

## 2.1. RFID Test Chamber/Cabinet

The RFID (Radio Frequency Identification) is a form of wireless communication that incorporates the use of electromagnetic or electrostatic coupling in the radio frequency portion of electromagnetic spectrum to uniquely identify an object, animal or person. Use cases for RFID technology include healthcare, manufacturing, inventory management, shipping, retail sales and home use. Every RFID system consists of three components: a scanning antenna, a transceiver and a transponder. When the scanning antenna and transceiver are combined, they are referred as an RFID reader or interrogator. The RFID reader is a network-connected device that can be portable or permanently attached. It uses radio frequency signals to activate the tag. Once activated, the tag sends a wave back to the antenna, where it is translated into data [3].

In DigiLab4U, we have one kind of RFID reader and this measurement cabinet is a testing environment for RFID UHF-Tags (800-1000 MHz). It’s one of the industry-related applications that are provided as a learning resource for industry 4.0. The RFID Lab consists of a fully equipped RFID measurement cabinet, paired with a computer-based control system. To perform the experiment, several types of transponder and substrates are available [4]. These substrates have unique characteristics and in order to study the behavior of each substrate, the RFID tag is placed on these materials. So, when the Radio Frequencies are bombarded on the tags in the chamber, the resultant voltage signals are studied and efficiency or required power to activate the tag is studied graphically through Tagformance software.

The lab equipment consists of:

- RFID Test Chamber (Voyantic)
- UHF RFID Reader (Tagformance Lite 2.2)
- Control System: PC with software (Tagformance 10)
- Various Transponder and substrates
- Mitsubishi RV-2SDB MELFA Robot.

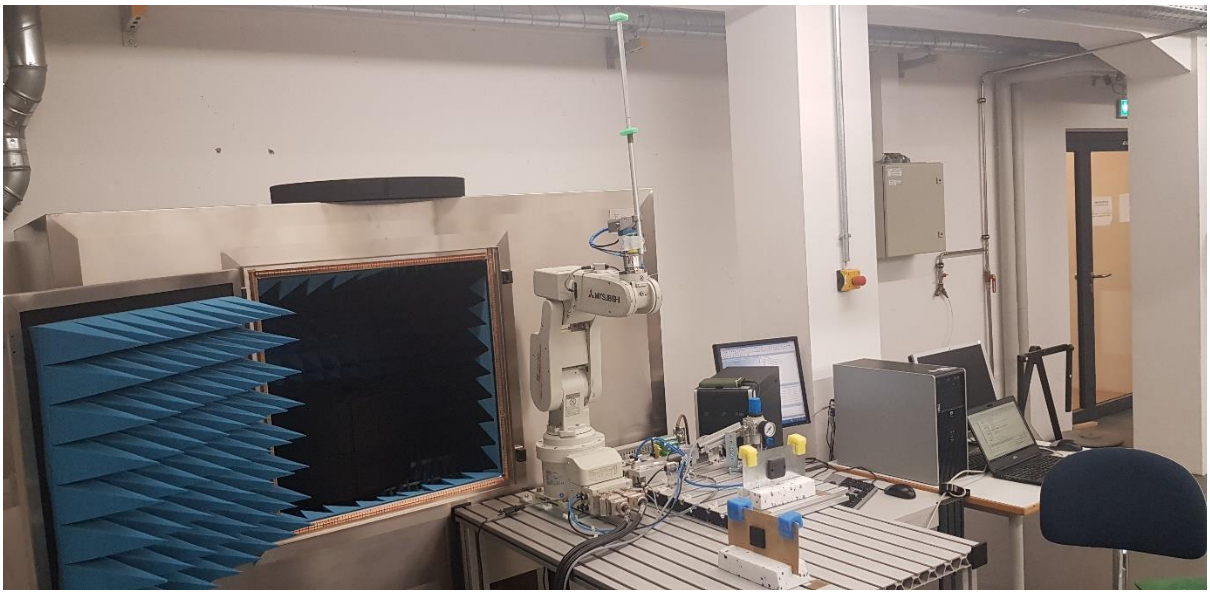


Figure 2-1: DigiLab4U - RFID Cabinet and Robot Experiment

The integration of above-mentioned components of the lab is the main purpose of this project. It is not only feasible for the current staff but in the future, the students or other users would be able to perform experiments remotely. There would be certain limitations of access because scope of the project doesn't allow the user a full hand on the equipment.

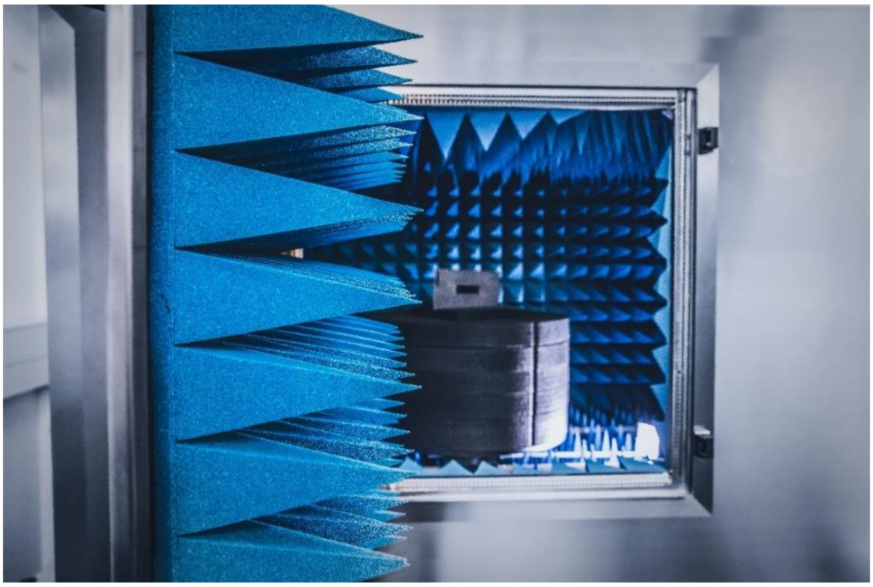


Figure 3-2: RFID Measurement Cabinet

## 2.2. RFID Tags

RFID tags are made up of an integrated circuit (IC), and antenna and a substrate. The part of an RFID tag that encodes identifying information is called RFID inlay. There are two main types of RFID tags: Active RFID and Passive RFID. An active RFID tag has its own power source, often a battery. A passive RFID tag, on the other hand, does not require batteries rather it receives its power from reading antenna whose electromagnetic wave includes a current in the RFID tag's antenna. RFID Tags typically hold less than 2000 KB of data, including a unique identifies/serial number. Tags can be read-only or read-write, where data can be added by the reader or existing data overwritten [5].

In DigiLab4U, we have a testing environment for RFID UHF-Tags (800-1000 MHz). A UHF RFIS system ranges from 300 MHz to 960 MHz, with a typical frequency of 433 MHz and can generally be read from 25-plus feet away. The frequency used will depend on the RFID application.

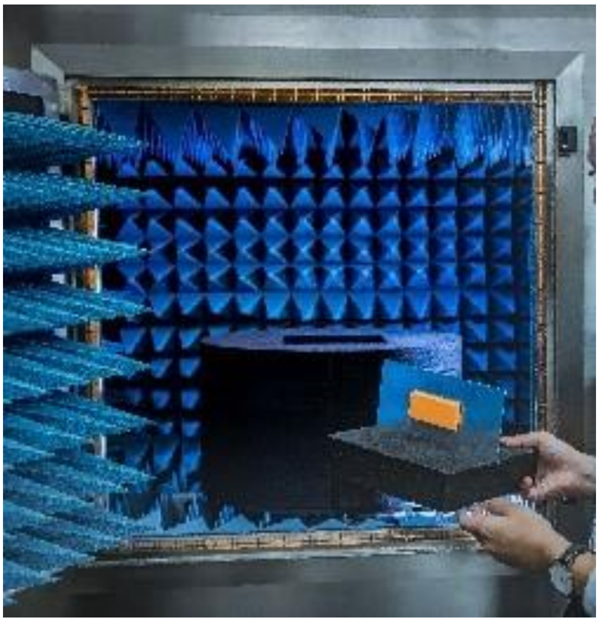


Figure 2-3: RFID UHF Tag testing environment

In this project, many tags were used placed on different substrates/materials in order to calculate the characteristics of the material. The properties associated with the material could be tested using Tagformance Software.

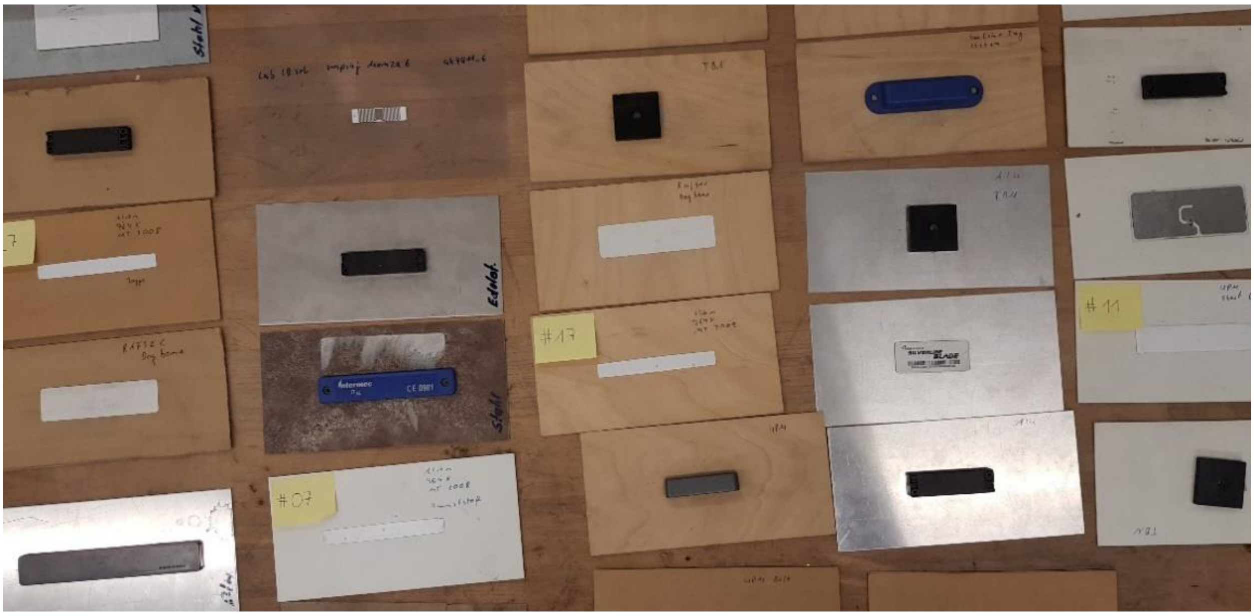


Figure 2-4: RFID UHF Tags

Each tag has a base that was designed manually in order to adjust the tag in the chamber. After adjusting the tags, the application allows for instance sensitivity and orientation testing, so the students would be able to check out the needed power to activate the transponder or the theoretical read range. Finally, student would be able to compare and evaluate the efficiency of transponders and they are able to assess their ideal use.

So, in DigiLab4U RFID Lab, the RFID Transponders will be stored in a tag-magazine in front of the chamber. The robot arm grabs the previously selected transponder/tag from the tag-magazine, moves into the measuring chamber and places the transponder. The user performs the measurement remotely on Tagformance software and the whole process is being transmitted to the user via video stream. Integration with DigiLab4U, remote control for tag testing, measurement data will be stored in Research Data System, integrated learning environment and analytics [6].

*Figure 2-5: Tag measurement Analysis on Tagformance Lite software*

The ultimate lab apparatus then consists on:

- RFID Test Chamber (Voyantic)
- UHF RFID Reader (Tagformance lite 2.2)
- Control System: PC with software (Tagformance 10)
- Various Transponders and Substrates

The user groups could be students or researchers from HFT or other universities (via remote access), Open Lab for private enterprises, companies and public research (possible qualification offer for industry 4.0)

# 3. Mitsubishi RV-2SDB and other Attributes

This chapter deals with the basic understanding of one of the Industrial Robots that has been used for this project i.e. MITSUBISHI MELFA RV-2SDB.

## 3.1. Introduction of the Robot

Mitsubishi MELFA RV-2SDB Series is a compact 6 axis robot that is ideally suited for assembly, material handling, inspection and a variety of other tasks. It is a vertically articulated robot with a new design that is streamlined, high-speed with high functionality and 2 Kg payload and it has a capacity of 256 programs.

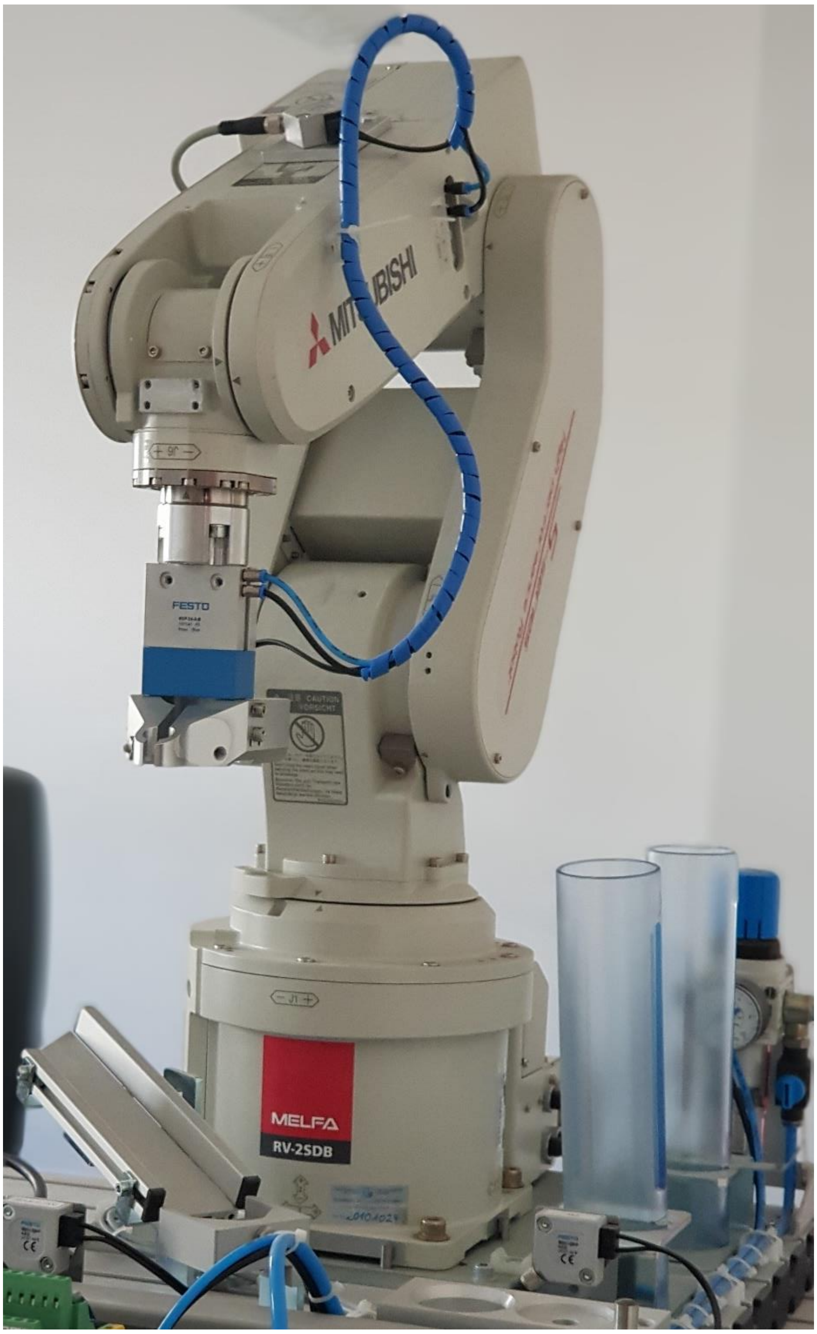


*Figure 3-1: Mitsubishi MELFA RV-2SDB Robot*

**Features:**

The robot has several interesting features such as:

## 1- Reduced profile while maintaining a large operating range

- The length and shape of the arm are designed for optimum performance and maxim reach while providing the ability to reach positions close to the robot base.
- A greater range of motion is insured in applications requiring ceiling or wall mount.
- J1 operation range is expanded to 480º (±240º). This eliminates any rear side dead zone.

## 2- Advanced Servo control provides high-speed and high-accuracy operation

- Maximum composite speed is 4-400 mm/s. Additionally, the speeds of axes J4-J6 have been optimized to satisfy high-speed assembly applications.
- Positioning repeatability of ±0.02mm combined with active-gain control and highly rigid arm design provide for high accuracy positioning at high speed.

## 3- Unique Arm design allows greater range of motion and accessibility

- Offset Arm design greatly reduces the robot's minimum operating radius allowing work close to the robot base.
- Reduced elbow protrusion lessens rear interference points.
- A compact wrist design enables the robot to reach into smaller spaces at many angles.

## 4- Multiple interfaces assure convenience and flexibility in integration

- The robot comes standard with additional axis control, Ethernet and encoder (for line tracking) interfaces. Profibus and DeviceNet are available options.
- Direct communication to GOT is available without the use of a programmable controller.

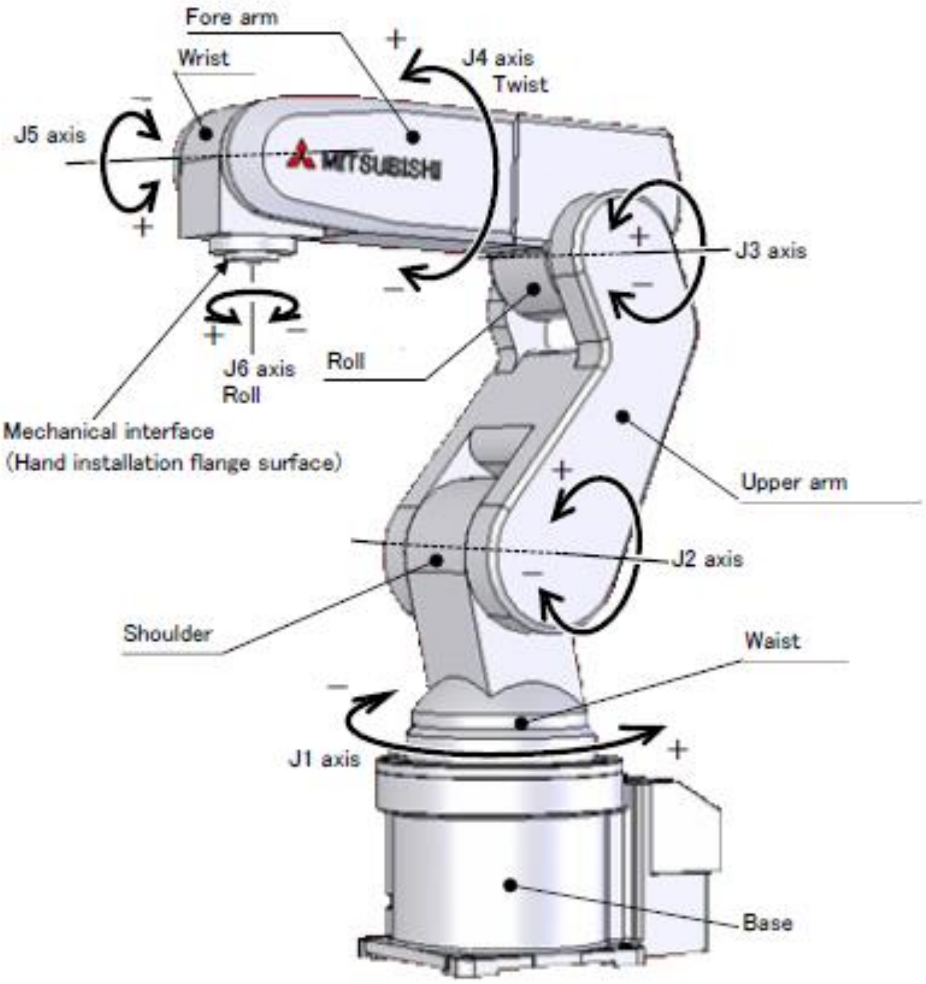


*Figure 3-2: Mitsubishi MELFA RV-2SDB Robot mechanical structure*

**5- Software tools are powerful and user-friendly**

- Software tools provide simplified design, programming, monitoring and diagnostics.
- RT ToolBox2 and CIROS Studio are windows-based software for robot programming, debugging, parameters, simulation and diagnostics.
- MELFA-Vision is a Machine vision software configuration tool for Cognex In-sight vision sensors
- MELFA-Works is an advanced 3-D simulation and design software that works as an add-on to SolidWorks [7].

However, in this project, CIROS Studio is used for the programming, simulation and debugging of the robot.

## 3.2. Robot Controller CR1DA-700

The robot controller that has been used in this project is CR1DA-700 controller that comes equipped with this robotic arm. This is an advanced controller with multiple features and functionality. Supply Voltage for this controller is AC 180 to 230V. installed Power required is 0.5 kW and the range of operating frequencies is 50-60 Hz.

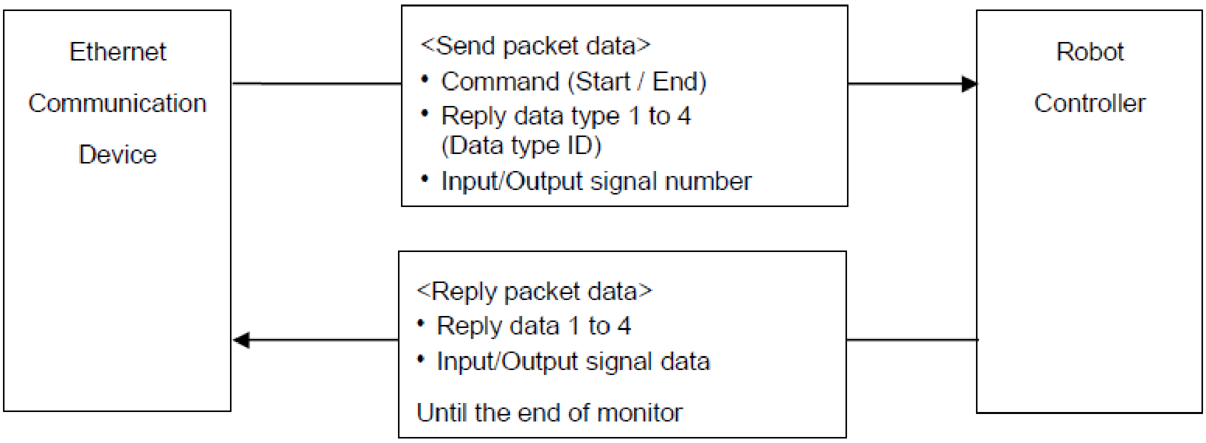

Ethernet Communication Device
<Send packet data>
• Command (Start / End)
• Reply data type 1 to 4 (Data type ID)
• Input/Output signal number
Robot Controller
<Reply packet data>
• Reply data 1 to 4
• Input/Output signal data
Until the end of monitor

Figure 3-3: CR1DA Communication Interface

Some functions of the controller have been explained as followed:

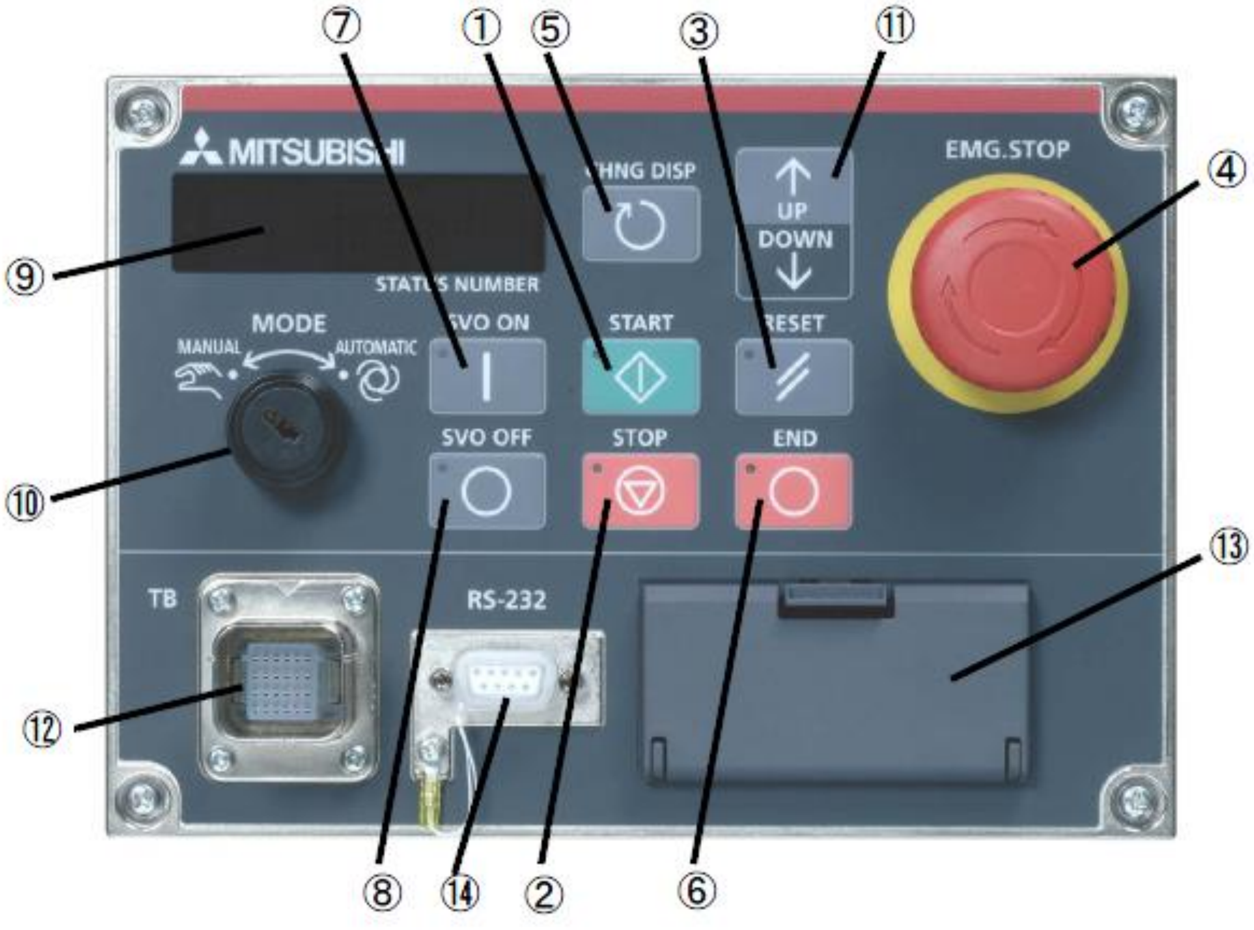


Figure 3-4: CR1DA Operating Panel

1- START Button:
   This executes the program and operates the robot. The program is run continuously.
2- STOP Button:
   This stops the robot immediately. The servo does not turn OFF.
3- RESET Button:
   This resets the error. This also resets the program's halted state and resets the program.
4- Emergency Stop Switch:
   This stops the robot in an emergency state. The servo turns OFF.
5- CHNGDISP Button:
   This changes the details displayed on the display panel in the order of "Override", "Program No.", "Line No."
6- END Button:
   This stops the program being executed at the last line or END statement.
7- SVO. On Button:
   This turns ON the servo power.
8- SVO. OFF Button:
   This shuts down the servo.
9- STATUS NUMBER (Display panel):
   The Alarm No. Program No. Override Value (%) etc. are displayed.
10- MODE Key switch:
   This changes the robot operation mode from automatic to Manual or vice versa.
11- UP/DOWN Button:
   This scrolls up or down the details displayed on the "Status Number" display panel.
12- T/B connection Connector:
   This is a dedicated connector for connecting the T/B. when not using T/B, connect the attached dummy connector.
13- Interface Cover:
   USB Interface and Battery are mounted.
14- RS-232 Connector:
   This is an RS-232C specification connector for connecting the personal computer.

### 3.3. Teaching Pendant (T/B)

The teaching pendant (T/B) is installed with the controller that allows to have maximum control and functionality over the robot. In this project, R32TB has been used that has following attributes:

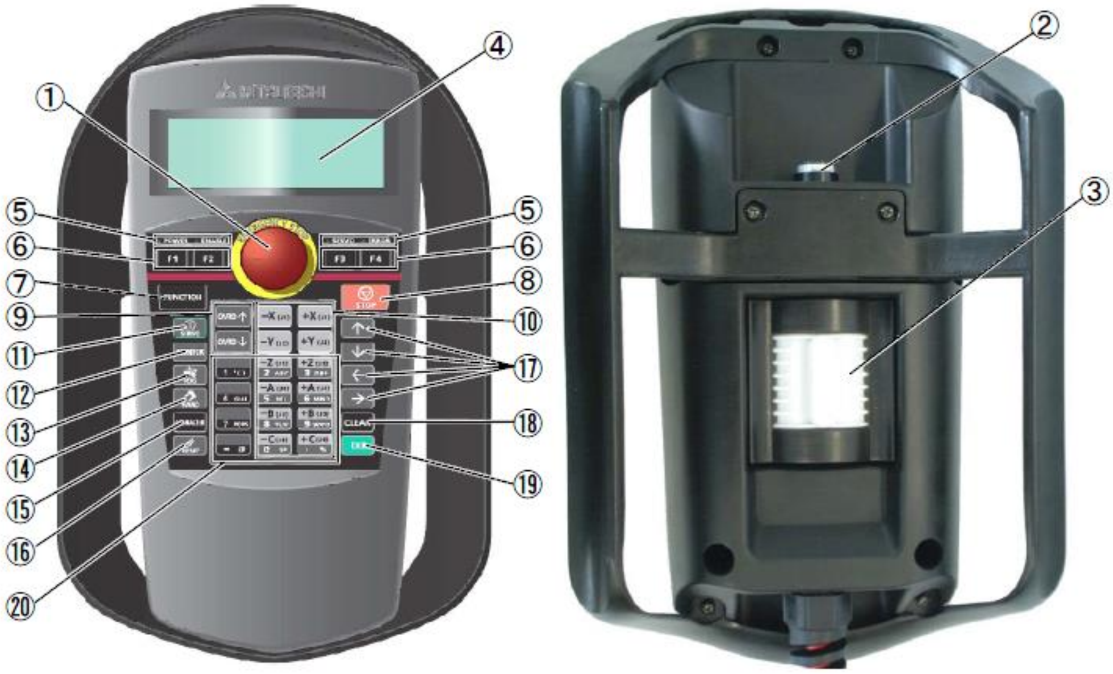


*Figure 3-5: Teaching Pendant- R32TB*

Functions associated with each button are shown below:

1. Emergency Stop Switch:
   The robot servo turns OFF and the operation stops immediately. The release of the emergency stop turns the switch to the right or pulls it.
2. Enable/Disable switch:
   This switch changes the T/B key operation between enable and disable.
3. Enable switch:
   When the Enable/Switch is released, the servo will be turned OFF, if this switch is released or pushed strongly, the robot will stop immediately.
4. LCD Display Panel:
   The robot status and various menu options are displayed.
5. Status Display Lamp:
   Display the state of robot or T/B.
6. [F1], [F2], [F3], [F4]:
   Execute the function corresponding to each function currently displayed on LCD.
7. FUNCTION:
   Change the function display of LCD.
8. STOP Key:
   This stops the program and declares the robot to a stop.
9. OVRD:
   Change moving speed by increasing or decreasing.
10. JOG Operation Key:

Move the robot according to jog mode and input the numerical value.

11. SERVO:
    Press this key with holding the Enable switch then servo will be turned On.
12. MONITOR:
    It becomes monitor mode and displays the monitor menu.
13. JOG:
    It becomes Jog mode and displays the jog operation.
14. HAND:
    It becomes hand mode and displays the hand menu.
15. CHAR:
    This changes the edit screen and changes between numbers and alphabetic characters.
16. RESET:
    This resets the errors. The program reset will execute if this key and EXE keys are pressed.
17. ↑ → ↓ ← :
    Moves the cursor in each direction.
18. CLEAR:
    Erase the one character on cursor position.
19. EXE:
    Input operation is fixed. While pressing this key the robot moves when in direct mode.
20. Number/Character:
    Erase the one character on the cursor position. And, inputs the number or character.

### 3.4. CIROS Studio

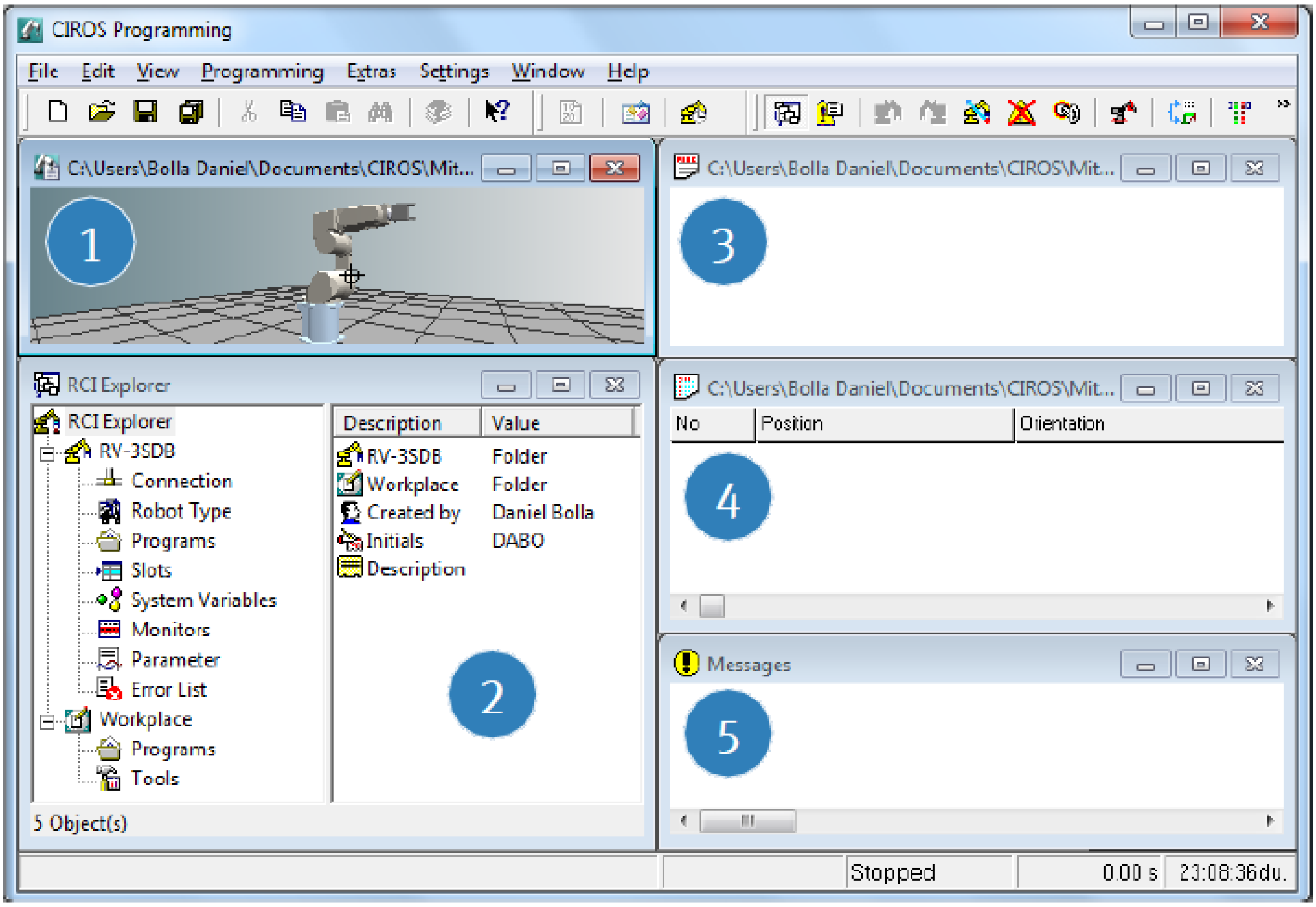
CIROS Studio [8] is an integrated development environment (short IDE), for Mitsubishi Robots that supports fast and easy generation of MELFA-Basic III/IV/V or MoveMaster Command programs. It is a professional tool for creating simulation models. Used by industry, this powerful development platform unites, in one common interface, three essential tools Simulation, Modeling and Programming [9]. 3-D Modeling based on standardized import filters for external CAD systems:

- Import filters for STEP, IGES, VRML and STL
- Basic CAD functions
- Definition of local coordinate systems (Master Frames) for simple relative positioning of objects
- Modeling through parameterization of the geometry, the kinematics and the material and physical characteristics

- Libraries with industrial robot systems and numerous automations components
- Library with powerful automation mechanisms
- Export files for DXF, STEP, IGES, VRML and STL

After various tests and optimization, program is uploaded onto the robot using a direct connection between computer and the robot via network or serial port. CIROS Studio has supervision over the robot while it performs the uploaded code, as well as visualization of movement in 3-D environment.

The software is not open source as people must buy it officially to use it. It comes with a DVD drive and a small key that must always be kept inserted into the PC every time you have to use CIROS Studio. The key is like a license authorization that makes sure the software is licensed or one has purchased it. As soon as the key is unplugged, CIROS Studio stops working.



| Description | Value |
|---|---|
| RV-3SDB | Folder |
| Workplace | Folder |
| Created by | Daniel Bolla |
| Initials | DABO |
| Description | |

*Figure 3-6: CIROS Studio*

The structure of the interface is as followed:

1- ***Virtual Robot Window*** – checks the positions on a virtual robot
2- ***RCI Explorer*** – Project Tree, contains contents of the robot (Programs, Program slots, Variables, Parameters etc.), contents of Workplace (Programs, position lists etc.)
3- ***Program editor*** – used for writing a program/code here
4- ***Position List*** –you can edit the positions here (Add new positions, Modify positions, Remove positions)
5- ***Messages*** – messages, warnings and errors

### 3.5. MELFA-Basic V

For this project, the programming of the robot has been done in MELFA-Basic V [10] language. Some common instructions of the stated language are as follows:

- ***DLY*** – wait for 0.5s before executing next instruction
- ***SERVO ON*** – turns on the servo
- ***SERVO OFF*** – turns off the servo
- ***HOPEN*** – opens the hand/gripper
- ***HCLOSE*** – closes the hand/gripper
- ***MOV*** – moves the robot to a specified position without following a sequenced path

It is necessary to use instruction DLY before opening (HOPEN) or closing the hand (HCLOSE). Also, it is quite recommended to use DLY command before any movement because some precise movements must be done though joint interpolation in high speed and if there is no delay the motor may suffer loss.
There are several motion instructions like straight line motion MVS, Circular motion instructions, Relative motion and Continue motion instructions. The visual representation and attributes of MOV command are discussed as followed:

***Move Commands:***

MOV <Position>

- Movement with joint interpolation.
- All the positions can be reached.
- The path would be unknown through this command.
- This is mostly used for the movements far from the objects.
- E.g. MOV P1 ‘Move to P1 with joint interpolation.

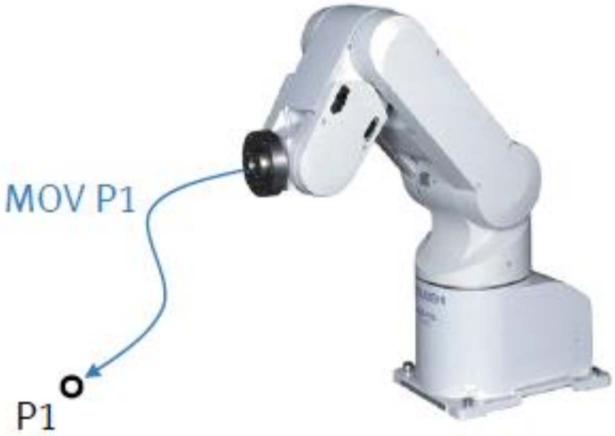


*Figure 3-7: MOV P1 Command*

Some I/O management commands and interrupts instructions are also present in MELFA-Basic V. A conditional branching sequence or syntax is given as followed:

**If <condition> THEN <Instruction> ELSE <Instruction> ENDIF**

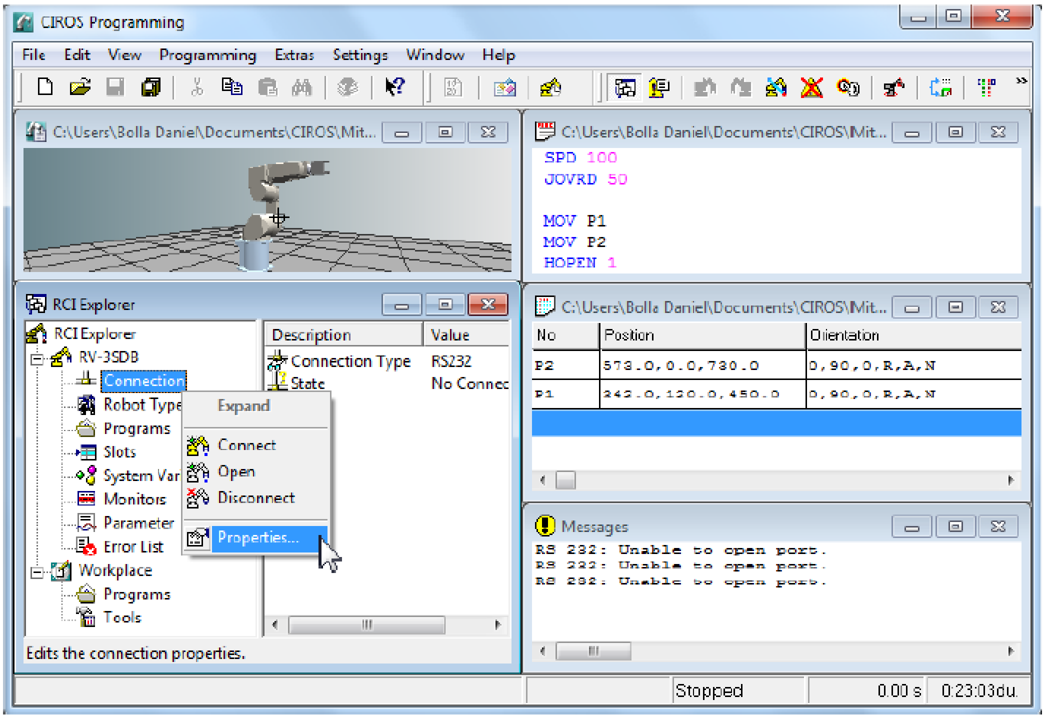
Using JOG mode, it's required to manually place the robot in the desired starting position and save that position data using T/B controller's options and repeat that for all the positions needed in the movement trajectory.

After saving the positions, the code is written in MELFA-Basic V programming language that enables the robot to move over the predefined positions. Robot is programmed so that it moves with different speeds between different positions.

## 3.6. A Test Exercise

An example is shown here in order to get better understanding of starting with RV-2SDB, CIROS Studio and MELFA-Basic V.

- First, the robot has been turned on and after starting CIROS Studio, connection parameters are set accordingly by clicking on Properties of Connection.



```
SPD 100
JOVRD 50

MOV P1
MOV P2
HOPEN 1
```

| Description | Value |
|---|---|
| Connection Type | RS232 |
| State | No Connec |

| No | Position | Orientation |
|---|---|---|
| P2 | 573.0,0.0,730.0 | 0,90,0,R,A,N |
| P1 | 242.0,120.0,450.0 | 0,90,0,R,A,N |

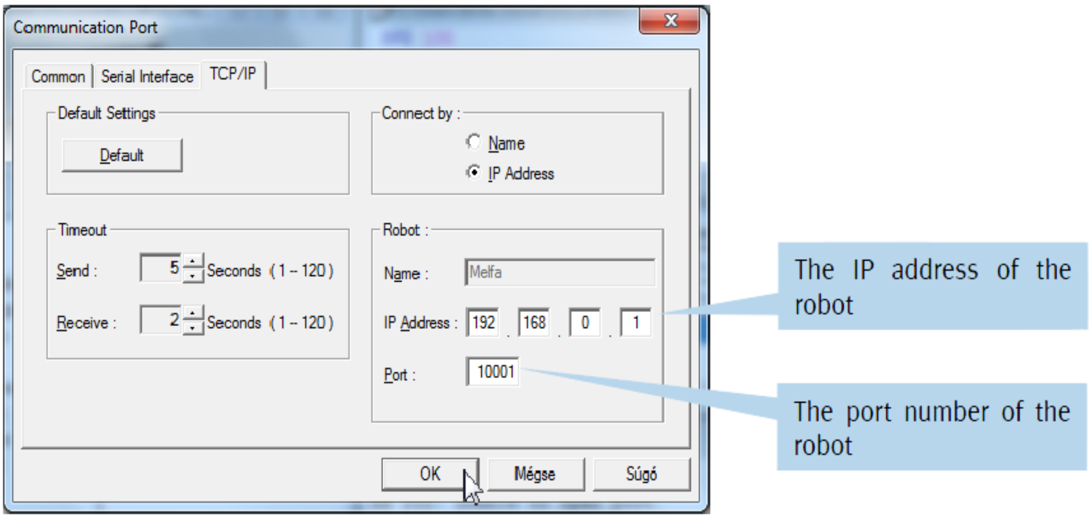
```
RS 232: Unable to open port.
RS 232: Unable to open port.
RS 232: Unable to open port.
```

Edits the connection properties.

Stopped 0.00 s 0:23:03du

Figure 3-8: CIROS Studio Connection Parameter Properties

- After the connection parameters have been opened as shown in following figure, change the TCP/IP properties according to the robot specifications. In my case, the IP address of the robot is specified as ***192.168.0.20*** and the Port No. is specified as ***10002***. These parameters have been set in CIROS Studio in order to have a successful connection between PC and the Robot. These parameters also come with the standard manual of handbook when you first factory ship your robot. In case if someone is not aware of the IP or Port properties of their robot, they need to contact the service provider or manufacturer of the apparatus.



Connect by : ○ Name ● IP Address

Send : 5 Seconds ( 1 - 120 )

Receive : 2 Seconds ( 1 - 120 )

Name : Melfa

IP Address : 192 . 168 . 0 . 1

Port : 10001

The IP address of the robot

The port number of the robot

Figure 3-9: TCP/IP settings in CIROS Studio

- After setting the IP and Port, the connection is made by right clicking connection parameter in RCI explorer and selecting connect. PC can recognize the type of robot through the connection.
- Now, the is written in MELFA-Basic V language for a simple pick and drop purposes. Let's take a following example:

```
HOPEN 1 'To be sure, the gripper is opened
MOV P1HELP '(1)
MVS P1 '(2)
DLY 0.5 '(3)
HCLOSE 1
DLY 0.5 '(4)
MVS P1HELP
MOV P2HELP '(5)
MVS P2 '(6)
DLY 0.5 '(7)
HOPEN 1
DLY 0.5 '(8)
MVS P2HELP
MOV P0 '(9)
```

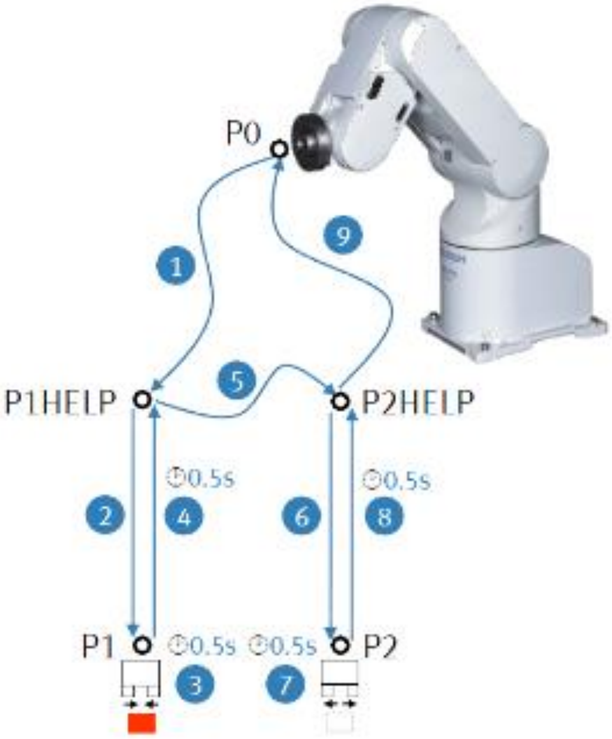
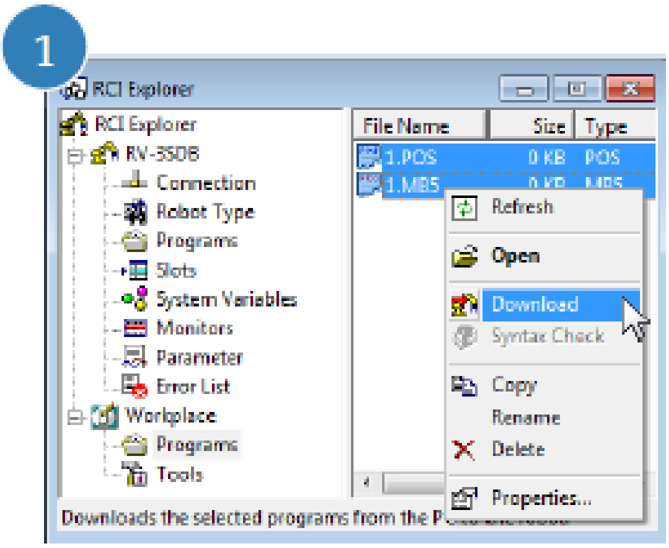


*Figure 3-10: Test Example of Pick and Drop Positions and Movements*

- Now the program is downloaded into the robot. The program and position lists can be downloaded separately. There are two ways to download the program, first one is through RCI Explorer and the second one is through Upload button in the toolbar of CIROS Studio.

1. Select the program and/or position lists in RCI Explorer (in workspace section). Right-click and select download.



Downloads the selected programs from the P

Figure 3-11: Downloading of Program and/or Position Lists

2. Using the download Tool. It lets you download the content of active window.
   E.g. if the program editor window is active, then the program will be downloaded.

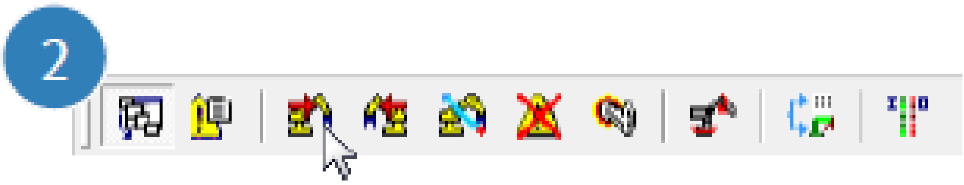


Figure 3-12: Download Tool

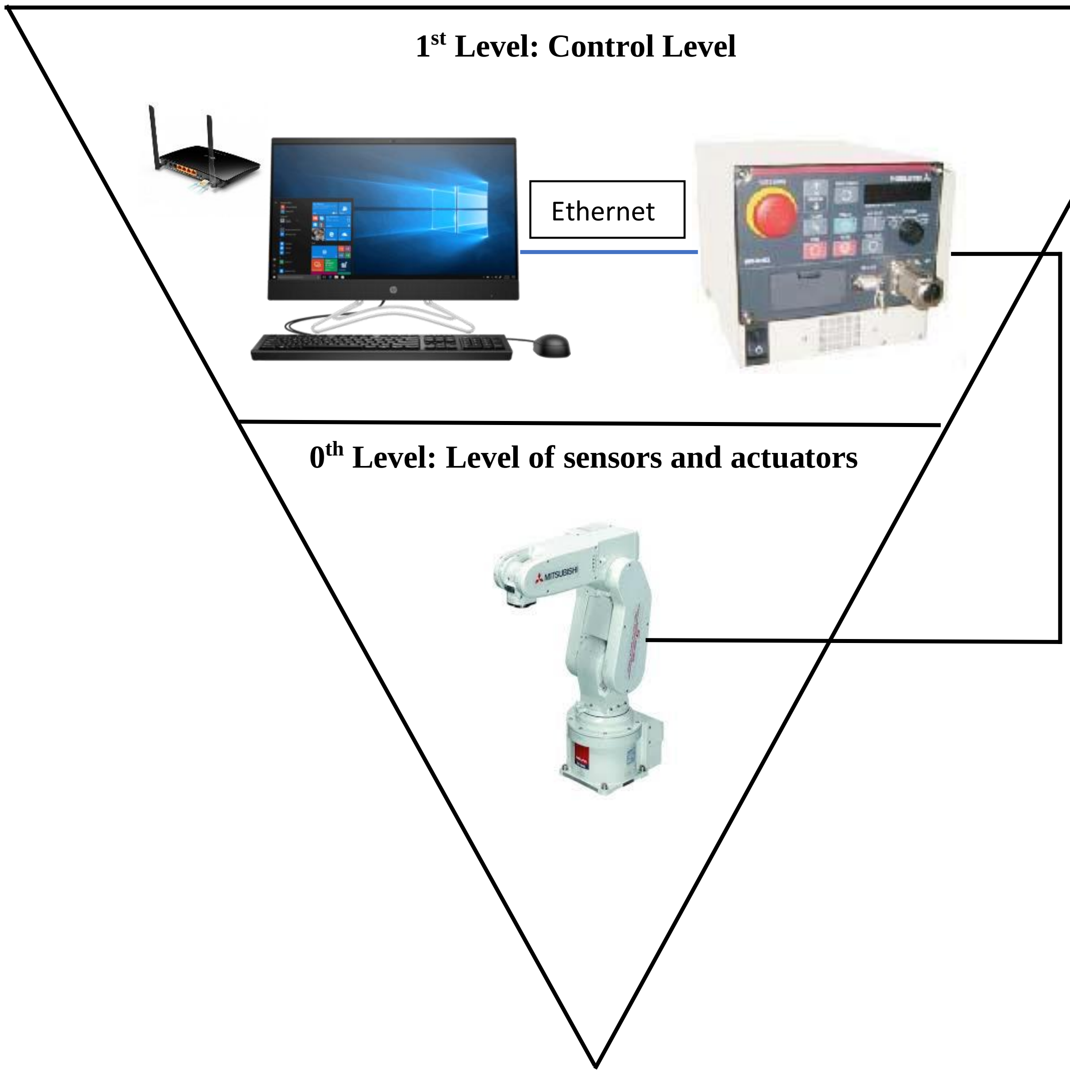
# 4. Remote Control of the Robot

In this chapter, the method of controlling the robot remotely, application interfaces, development and distribution of the system will be described.

## 4.1. Distribution of the System

A distributed control system (DCS) is a process or any dynamic system in which system elements are not placed centrally. They are rather distributed or divided into smaller parts, subsystems that are controlled by one or more control devices. Current industrial and control systems utilize mainly hierarchic (pyramidal) architectures containing logical and physical distribution entities which is more open and scalable [11].

This architecture of our system does not include all levels of a DCS but pertains a program distribution to a certain level that divides control software to more control units. Software is divided into robot controller and personal computer.



Robot control consists of 2 separate applications:

- Program in robot Controller (RC)
- Program in PC

The program in RC is built using simple MELFA-Basic algorithm:

***Open "COM1:" as #1***

***Print #1, "START"***

The program in PC is basically an application that has been designed on Visual Studio using Visual Basic language. Collaboration diagram for the main task/algorithm looks like as followed:

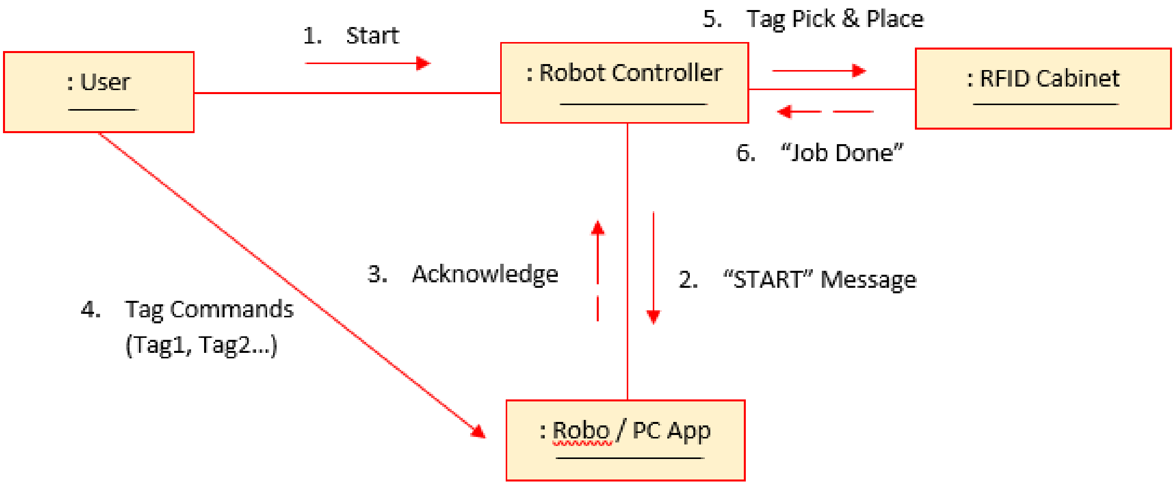
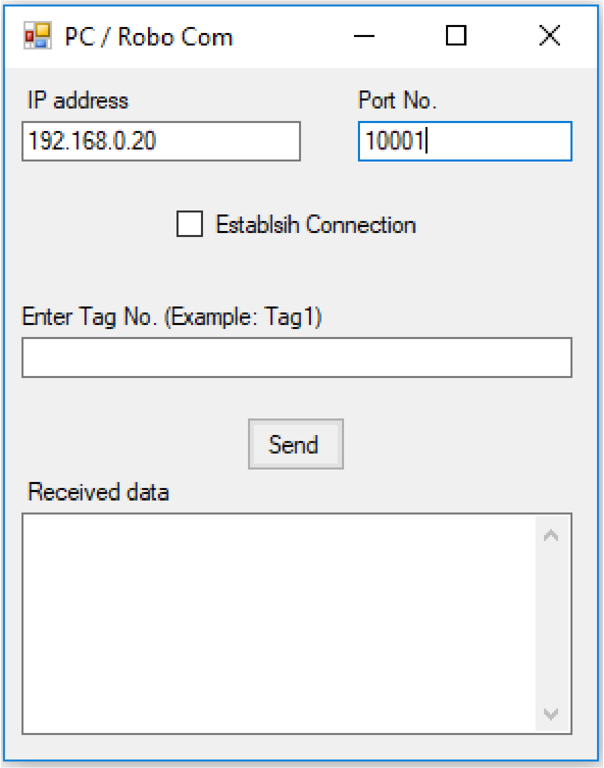


*Figure 4-1: Collaboration Diagram of Robot Algorithm*

First, at user end, after finalizing the code and downloading it in the controller, 'Start' button is pressed on the robot controller. Robot controller then communicates with the application and print a start message on the application display area. This acknowledges that the communication is successful and now user can send or receive commands to and from the robot respectively. Then user can send commands like picking the tag or experiment finishing commands after the status of robot has been verified. The robot then executes the given task and after it has successfully completed the job, it sends a "Job done" message and confirms that it is ready for the next task. After the users are done with their experiments, they can send finish command so that the robot could pick up the tag from the cabinet and place it in the exact position it was picked up from.

## 4.2. PC / Robo Com Application

An application has been designed so that we can control the robot remotely, to be independent of dedicated PC or send commands to the Robot. This application was developed on Microsoft Visual Studio using Visual Basic as a language.



IP address
192.168.0.20

Port No.
10001

Figure 4-2: PC/ Robo Com App

Here, the IP address and Port No. of robot controller is specified. The robot is connected to the PC via Ethernet cable. The LAN connection on robot controller may integrate 8 different ports from 10001, 10002…to 10008. In this scenario, 10002 is used for setting as communication destination of Robot with CIROS Studio software (in case of controlling the robot through Command Tool of CIROS Studio or to download the program into robot). 10001 is specified for the connection of PC / Robo Com application with robot controller. The connection works through socket programming. The application tries to open socket for data exchange over TCP/IP and if successful, the communication can be performed. If after several attempts, the socket is not opened for some arbitrary reasons, the application shows Timeout Error.

The code which handles the connection part is as followed:

```
Private Sub Check1_CheckStateChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Check1.CheckStateChanged
        ' Process for Connect or Disconnect
        Try
            If Check1.CheckState = CheckState.Checked Then
                Client = New TcpClient()
                Client.Connect(Text1.Text,
Convert.ToInt32(Text2.Text)) 'Connect
                Button1.Enabled = Client.Connected
                Timer1.Enabled = Client.Connected
            Else
                Timer1.Enabled = False
                Button1.Enabled = False
                Client.GetStream().Close() 'Disconnect
                Client.Close()
            End If
        Catch ex As Exception
            Check1.Checked = False
            MessageBox.Show(ex.Message, Me.Text,
MessageBoxButtons.OK, MessageBoxIcon.Error,
            MessageBoxDefaultButton.Button1)
        End Try
    End Sub
```

Then there is data exchange part between robot and application. In case of sending data, following code was used:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        'Send process
        Try
            Dim SendBuf As Byte() =
System.Text.Encoding.Default.GetBytes(Text3.Text)
            Dim Stream As NetworkStream = Client.GetStream()
            Stream.Write(SendBuf, 0, SendBuf.Length)
        Catch ex As Exception
            Client = Nothing
            Timer1.Enabled = False
            Button1.Enabled = False
            Check1.Checked = False
            MessageBox.Show(ex.Message, Me.Text,
MessageBoxButtons.OK, MessageBoxIcon.Error,
            MessageBoxDefaultButton.Button1)
        End Try
    End Sub
```

In case of receiving data, the code is as followed:

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
        'Receive process
        Try
            Dim Stream As NetworkStream = Client.GetStream()
            If Stream.DataAvailable Then
                Dim bytes(1000) As Byte
                Dim strReceivedData As String = ""
                Dim datalength = Stream.Read(bytes, 0, bytes.Length)
                strReceivedData =
System.Text.Encoding.Default.GetString(bytes).Substring(0,
datalength)
                Text4.AppendText(strReceivedData)
                Text4.AppendText(System.Environment.NewLine)
            End If
        Catch ex As Exception
            Client = Nothing
            Timer1.Enabled = False
            Button1.Enabled = False
            Check1.Checked = False
            '
            MessageBox.Show(ex.Message, Me.Text,
MessageBoxButtons.OK, MessageBoxIcon.Error,
MessageBoxDefaultButton.Button1)
        End Try
    End Sub
```

### 4.3. MELFA Basic Code for Communication

On the other end of the application, a piece of code in MELFA Basic V on CIROS Studio has been written in order to recognize the commands coming from PC / Robo Com application. That code is downloaded then into robot controller and executed later so that the robot could be controller remotely via PC / Robo Com application.

```
Def Char COMMAND
OPEN "COM1:" AS #1
*LOOP:INPUT #1, COMMAND
```

Open “Com1:” is a specified command in MELFA Basic V and it is used for setting port number 10001 as communication destination and open as file #1. Then there is a loop that takes strings as inputs and stores it into COMMAND variable. After the commands have been sent from PC / Robo Com application, a comparison is done as there could be many tags. So in order to pick a specific tag, the comparison is done as followed:

```
IF COMMAND = "Status?" THEN GOTO *STATUS
IF COMMAND = "tag1" THEN GOTO *TAG1
IF COMMAND = "finished tag1" THEN GOTO *PICK1
IF COMMAND = "tag2" THEN GOTO *TAG2
IF COMMAND = "finished tag2" THEN GOTO *PICK2
```

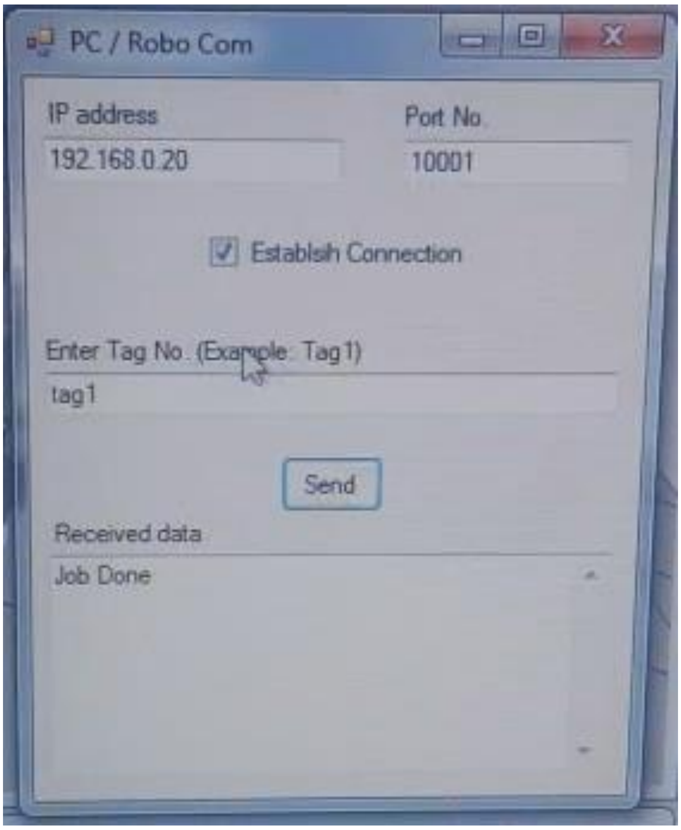
At first, the status of the robot is checked if it is available for a task or not. Then the robot decides the relevant tasks according to the Input COMMAND. Either it picks the tag from table and places it in the cabinet if the command is “Tag1”, “Tag2” and so on or picks the tag from RFID measurement cabinet and places it on the table after the experiment has been performed if the command is “finished tag1”, “finished tag2” and so on. Tag1, Tag2, Tag3 etc. are the tag labels that have been defined by me in order to identify the correct tag among all the other tags. In labels TAG1, TAG2, PICK1 and PICK2, basic movements commands and speed control of the robot is performed. The movements of robot to pick a certain tag are taught and stored manually. Speed/Override change is compulsory because not all the movements should be of same velocity.

For instance, in order to pick Tag1, a command “tag1” is sent via application and the IF statement checks which command it is and goes to the label *TAG1:

```
*TAG1:
SERVO ON
DLY 1
JOVRD 60
MOV P1
DLY 0.5
.
.
` your code
.
.
.
.
.
.
.
SERVO OFF
PRINT #1, "Job Done"
GOTO *LOOP
```

`JOVRD` is the code for setting different Joint movement overrides for the robot speed by the percentages of specified speed of the robot. Because among all the movements of the robot, some transitions must be performed with low speed. For instance, the picking of the tag should be done slowly otherwise there are chances that the robot could damage the apparatus or surroundings.

This is how the robot is controlled remotely through PC / Robo Com application. In order to establish connection between the application and the robot, it is necessary to verify that the robot controller and the PC in which the application is, are in the range of same IP addresses. It does not matter if the robot and PC are connected to a router or directly, the IP addresses range must match so that the communication could be established successfully.



IP address: 192.168.0.20

Port No: 10001

☑ Establsh Connection

Enter Tag No. (Example: Tag1): tag1

Received data: Job Done

*Figure 4-3: PC / Robo Com App after Successful Completion of given Task*

## 4.4. Setting the Parameters

Before TCP/IP communication of the robot with PC application, some parameters of robot controller must be changed. In order to change the parameters, the robot must be used in Manual mode.

After enabling the teaching pendant (T/B), the values of following parameters are changed by using T/B:

**CPRCE11:** change the value of this parameter from '0' to '2'

**COMDEV:** change the value of first entry of this parameter to 'OPT11'

After changing the value, it will look like this: OPT11, , , , , , ,

**RESTARTING THE CONTROLLER** is mandatory after changing the parameters.

More detailed discussion about changing the parameters and operating T/B will be discussed in further chapters.

# 5. Initialization

In this chapter, the main project/task will be discussed from the beginning and settings of the robot will be shown accordingly.

## 5.1. Starting the Robot

Before working with the robot, if a user wants to use the teaching pendant or move the robot themselves, the controller should be on Manual Mode. If they want to run a program that has already been written and downloaded, it should be on Automatic mode.

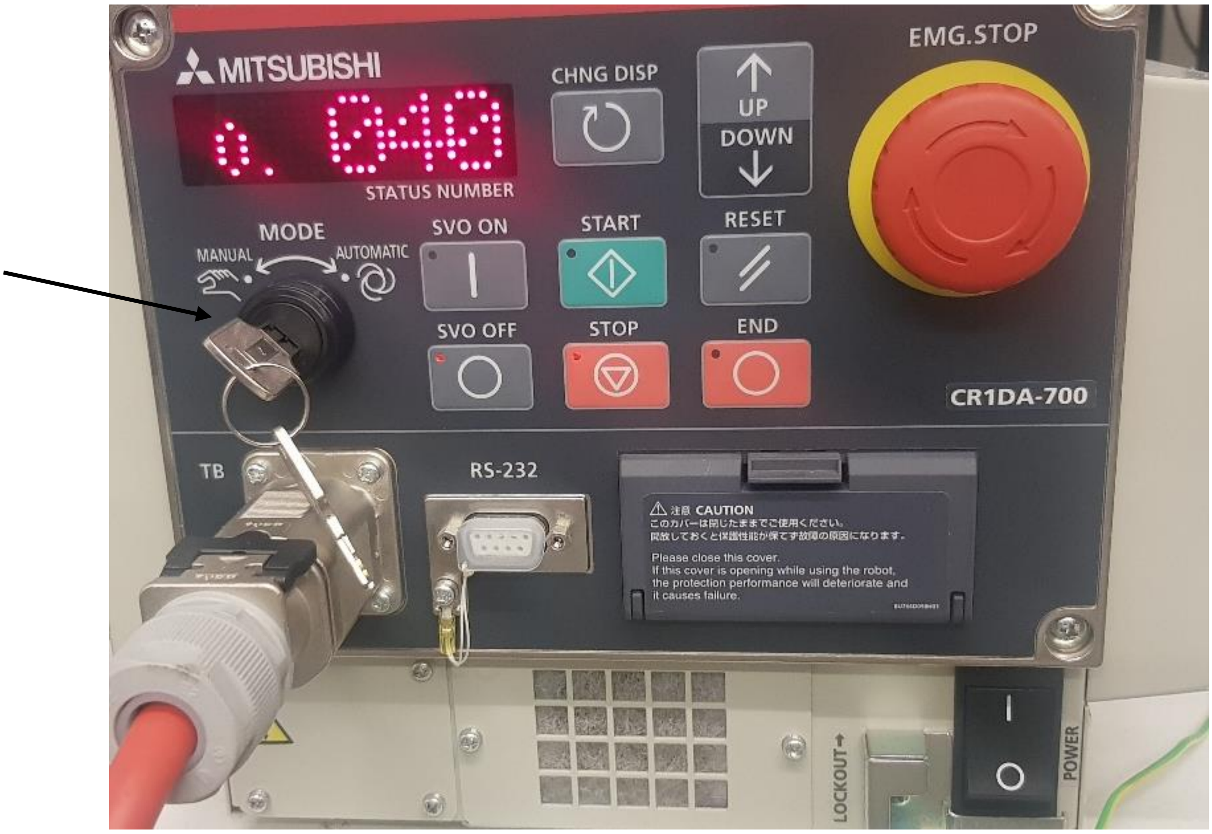

MITSUBISHI
STATUS NUMBER
CHNG DISP
UP
DOWN
EMG.STOP
MODE
MANUAL
AUTOMATIC
SVO ON
START
RESET
SVO OFF
STOP
END
CR1DA-700
TB
RS-232
注意 CAUTION
このカバーは閉じたままでご使用ください。
開放しておくと保護性能が保てず故障の原因になります。
Please close this cover.
If this cover is opening while using the robot, the protection performance will deteriorate and it causes failure.
LOCKOUT
POWER

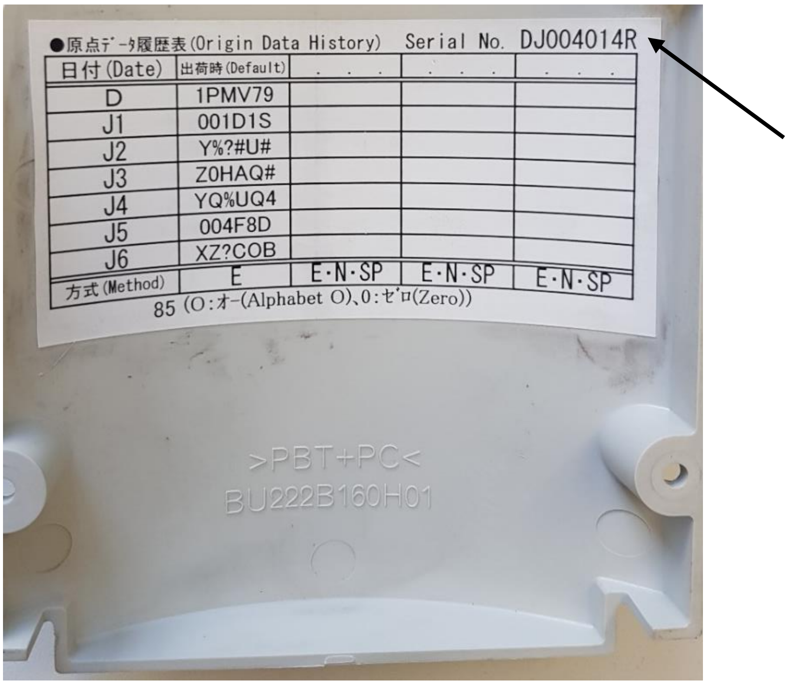
Figure 5-1: Robot Controller Modes

On CR1DA-700 controller, there is a key to shift the modes of robot. If the key is being turned to Automatic mode, it is made sure that the teaching pendant is disabled otherwise an error is generated and the robot must be reset after disabling the Enable key on T/B. After all the correct connections of cables, when the robot is turned on, if there are some errors on the controller screen, the cause and solution of those error are checked and rectified from the document [12]:
**"CRnQ/CRnD Controller Instruction Manual Troubleshooting"**

Every error has a code associated to it. The code can be looked for in the official above-mentioned document provided by **Mitsubishi Electric** so that the cause can be confirmed, and remedy can be implemented properly. The most common errors when the robot is turned on after a long period of time could be:

**C0150**: *Undefined robot serial number*

***Solution***: When the robot is turned on, if there is this error C0150, it means that serial number of the robot should be defined. The serial number of the robot is mentioned at the back side of motor cover. Motor cover must be opened, and the data provided at the back side should be noted. The data contains Position information of the robotic arm when it got shipped for the first time, like factory reset position of robotic arm, and the serial number of robot.



●原点ﾃﾞｰﾀ履歴表(Origin Data History)　Serial No. DJ004014R

| 日付(Date) | 出荷時(Default) | . . . | . . . | . . . |
|---|---|---|---|---|
| D | 1PMV79 | | | |
| J1 | 001D1S | | | |
| J2 | Y%?#U# | | | |
| J3 | Z0HAQ# | | | |
| J4 | YQ%UQ4 | | | |
| J5 | 004F8D | | | |
| J6 | XZ?COB | | | |
| 方式(Method) | E | E·N·SP | E·N·SP | E·N·SP |

85 (O:ｵｰ(Alphabet O)、0:ｾﾞﾛ(Zero))

>PBT+PC<
BU222B160H01

Figure 5-2: RV-2SDB Robot Serial Number and Position Data

## 5.2. Installing the T/B

In order to enter the serial number of the robot, it should be operated on Manual mode. The key is shifted to Manual and T/B is installed and then enabled by pressing the enable button on the back of it.

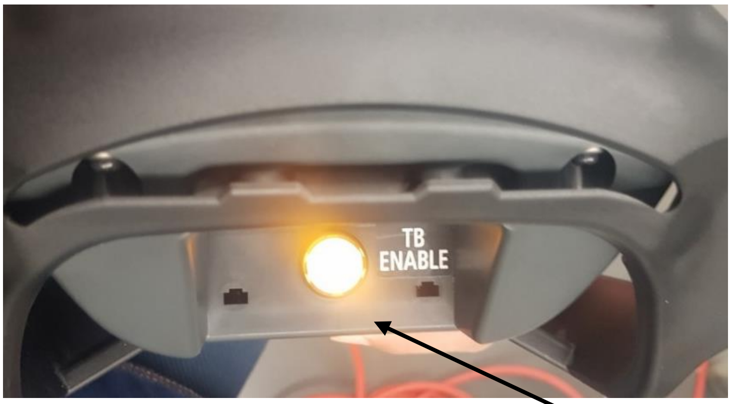
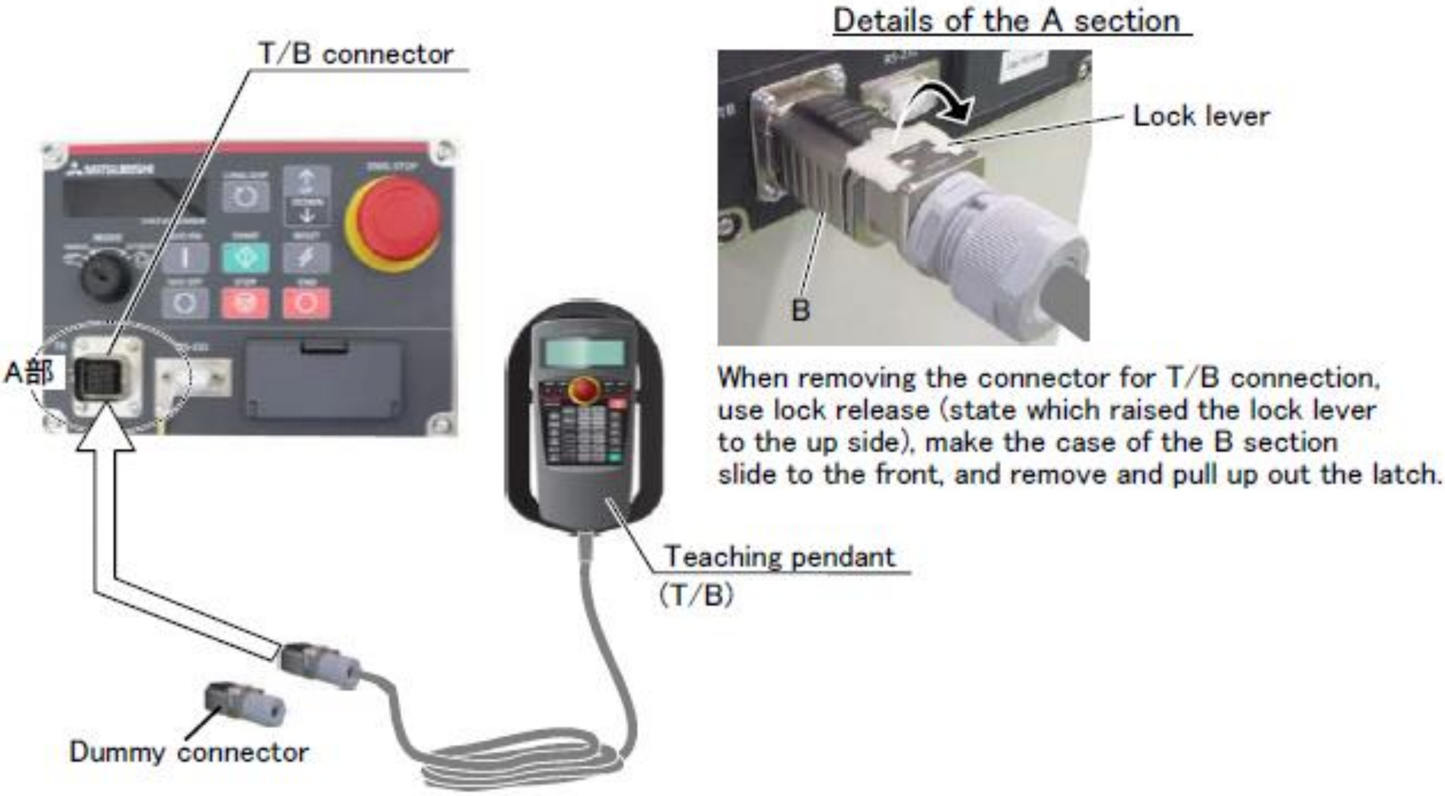


Figure 5-3: Teaching Pendant / T/B Enable Key

In order to install the T/B first, following steps are performed.

- Power Supply (POWER) of the robot controller must be turned off.
- T/B connector is connected to robot controller. Upper surface is used as lock lever and pushed in until there is a sound.



When removing the connector for T/B connection, use lock release (state which raised the lock lever to the up side), make the case of the B section slide to the front, and remove and pull up out the latch.

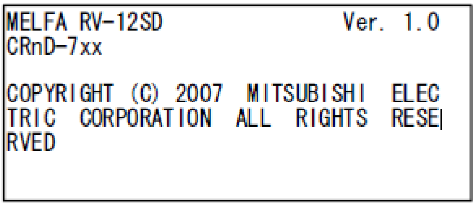
*Figure 5-4: Teaching Pendant / T/B Installment*

After installing the T/B, controller power is turned On. If pressing Enable button turns on the light of the button it means the T/B is connected successfully. Manual program creation, changing parameters, setting the origin, troubleshooting, alarms handling, and many more functions can be performed by T/B. It is made sure that all the safety standards are being followed during the operations. Connection settings are checked first before operating the robot. Manual overrides are quite different from Automatic operation overrides in terms of speed. Manual movements are performed at lesser speeds comparatively and the speed is set by default because even at 100% override, in manual mode, it is still not 100% because of user safety. But a user can change the speed in real time as well by adjusting the override while the robot is moving. These variations in speed can be visualized easily.

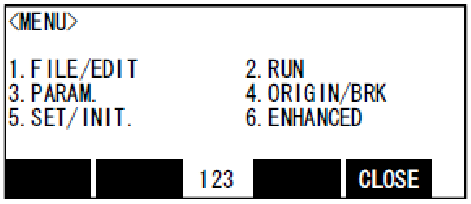
## 5.3. Entering the Serial Number

Serial number is entered into parameter ‘RBSERIAL’.

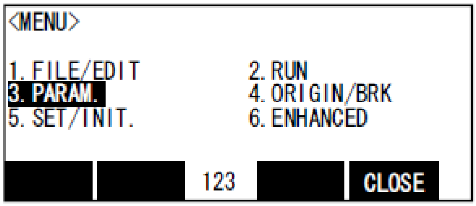
- First, the robot is reset by pressing [RESET] key of T/B and cancel the error of T/B.
- [EXE] key of T/B is pressed then, and Menu Panel is displayed.

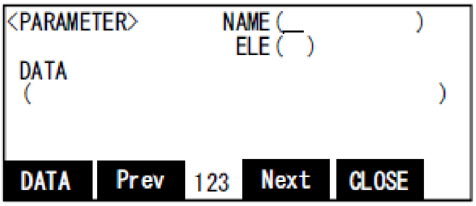


```
MELFA RV-12SD             Ver. 1.0
CRnD-7xx

COPYRIGHT (C) 2007 MITSUBISHI ELEC
TRIC CORPORATION ALL RIGHTS RESE|
RVED
```



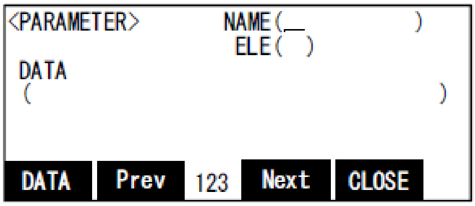
- ‘3’ is pressed on T/B and parameters are displayed.



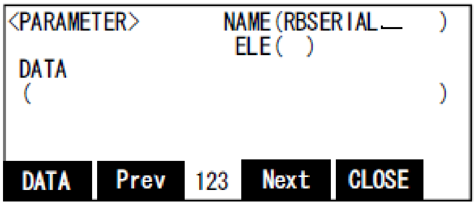


```
<PARAMETER>          NAME (_           )
                      ELE (  )
 DATA
 (                                      )
```

- RBSERIAL is entered as an input into the name panel.

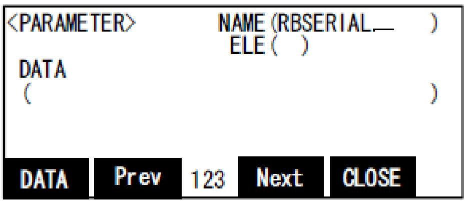


```
<PARAMETER>          NAME (_           )
                      ELE (  )
 DATA
 (                                      )
```

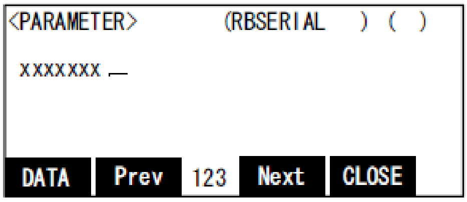


```
<PARAMETER>          NAME (RBSERIAL_   )
                      ELE (  )
 DATA
 (                                      )
```

- Function key [F1] corresponding to “DATA” is then pressed and serial number is entered through the pad of T/B.

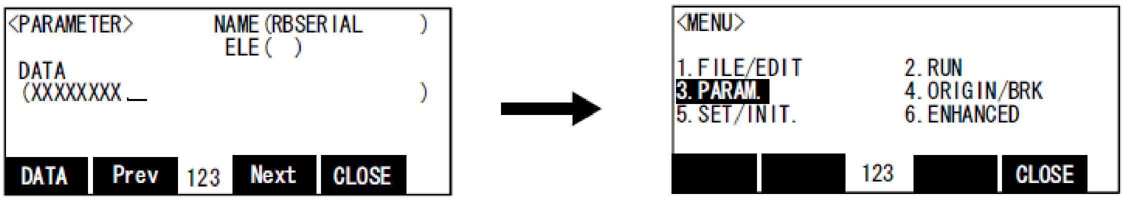


```
<PARAMETER>          NAME (RBSERIAL_   )
                      ELE (  )
 DATA
 (                                      )
```



```
<PARAMETER>          (RBSERIAL   ) (  )
 xxxxxxx _
```

- [EXE] is then pressed again to confirm the operation and fix the value with a confirming sound. Then parameter window is displayed.
- [F1] corresponding to “CLOSE” is again pressed to return to Menu screen.



All other parameters are changed in a similar way by entering parameter window and altering the values associated with those parameters.

## 5.4. Batteries and Associated Errors

When the robot is turned on after so long, the batteries of robotic arm get dried. So, the position data of robotic arm is lost eventually, and following error occurs: **H112n***: *Encoder ABS Position Data Lost*

Basically, Mitsubishi MELFA RV-2SDB has four batteries in the back side. These batteries store the information about real time position of robotic arm.

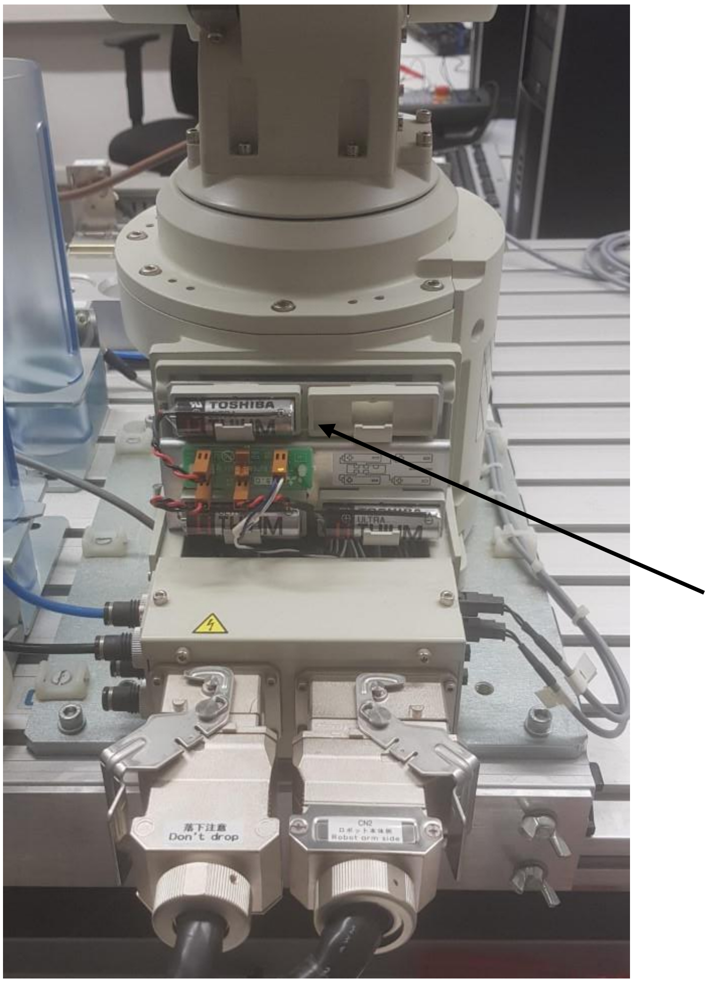


*Figure 5-5: RV-2SDB Batteries*

These batteries must be replaced if the position data has been lost. Once the new batteries have been installed, following task must be performed:

- Robot Controller should be turned ON by STOP and RESET keys being pressed simultaneously.
- When the screen displays: ←----------→, START key is pressed.
- Language on the screen is changed by pressing STOP.
- START is pressed after the language has been adjusted to ENG.
- 000 is then displayed on the screen and it should be replaced with 111 by pressing Up and Down arrows of the controller (⬆ ⬇)
- After 111 is being displayed on the screen, START is pressed and after a while, the robot is verified. Now RV-2SD is displayed on screen.
- START is pressed again then controller is being Reset.
- After reset process, o. 100 is appeared on the screen which means there is no further error and override is 100%.

After reset process, Position Data and Serial No. of the robot must be entered again as explained earlier.

There is one more battery in robot controller CR1DA-700 that must be replaced over time. An absolute encoder is used for position detector, so the position must be saved with backup battery when the power is turned OFF. The controller also uses the backup battery to save the program. These batteries are installed when the robot is shipped from the factory but since these are consumable parts, they must be replaced periodically. The guidelines for replacing the battery is one year, but this will differ according to robot's usage state. It is installed at the front side of controller. In order to replace this battery, the front cover of controller is opened, and battery is replaced.

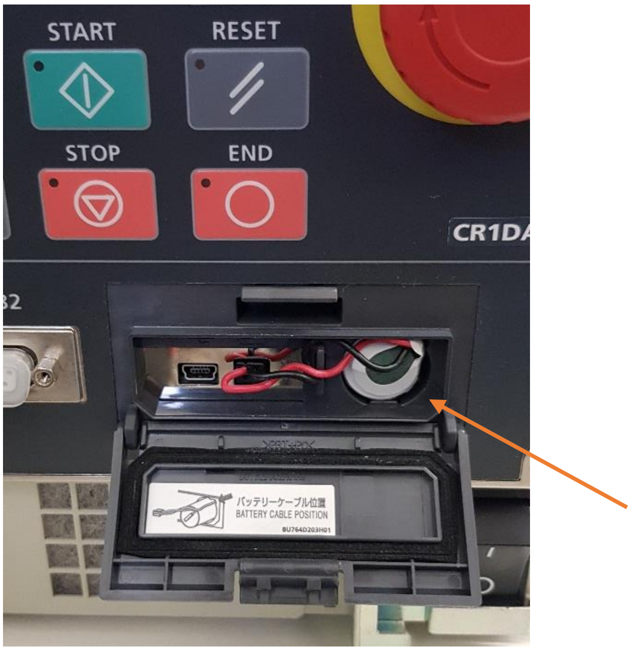


*Figure 5-6: CR1DA-700 Battery*

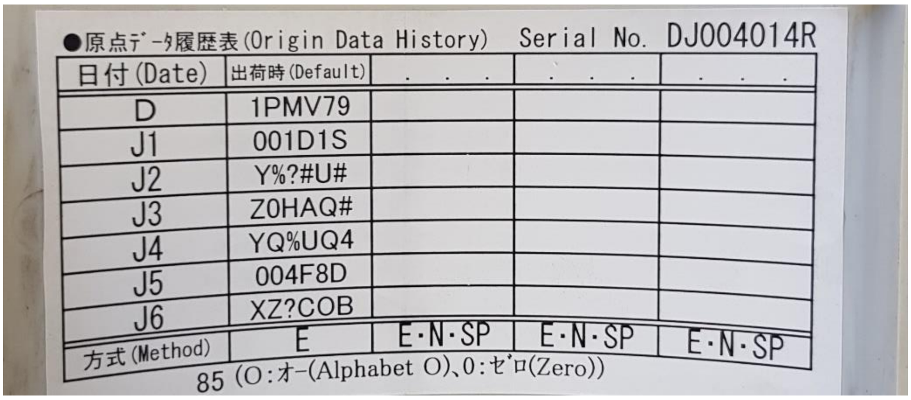
## 5.5. Setting the Origin

The origin is set so that the robot can be used with high accuracy. After purchasing the robot, change of combination of robot and controller or after resetting it, this operation is necessary to carry out before starting work. If the old battery is replaced because it has been used up, it is necessary to set the origin. There are several methods to set the origin.

- Origin data input method
- Mechanical stopper method
- Jig method
- ABS origin method
- User origin method

I have used two methods to set the origin in my project. These methods are explained as followed:

***Origin Data Input method:*** The origin data to be input is noted in the origin data sheet enclosed with the arm, or on the origin data history table attached to the back side of J1 motor cover [13]. The J1 motor cover is removed and data is noted.



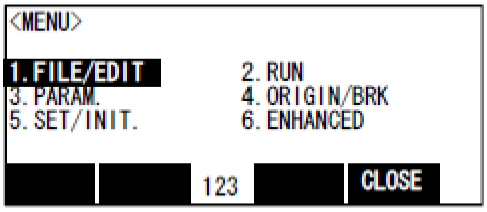
●原点ﾃﾞｰﾀ履歴表(Origin Data History)　Serial No. DJ004014R

| 日付(Date) | 出荷時(Default) | . . . | . . . | . . . |
|---|---|---|---|---|
| D | 1PMV79 | | | |
| J1 | 001D1S | | | |
| J2 | Y%?#U# | | | |
| J3 | Z0HAQ# | | | |
| J4 | YQ%UQ4 | | | |
| J5 | 004F8D | | | |
| J6 | XZ?COB | | | |
| 方式(Method) | E | E・N・SP | E・N・SP | E・N・SP |

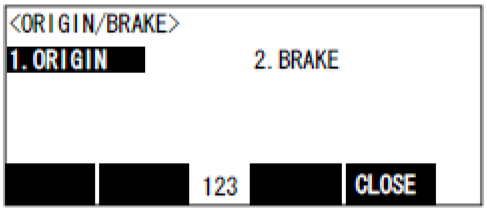
85 (O:ｵｰ(Alphabet O)、0:ｾﾞﾛ(Zero))

Figure 5-7: Origin Data Table at the back of J1 Motor Cover

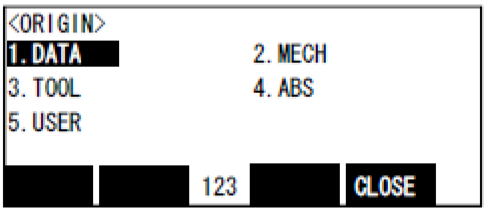
The values given in the default setting column is the origin settings set with the calibration jig before shipment. These values are then set by using T/B. On T/B, origin setting method is selected as followed:



[4] key is pressed and ORIGIN/BRK is selected.

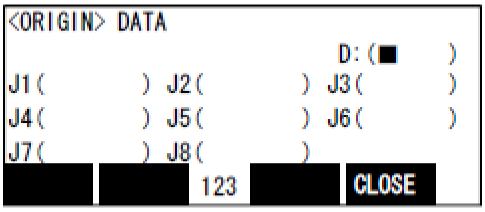


[1] key is pressed to select ORIGIN.



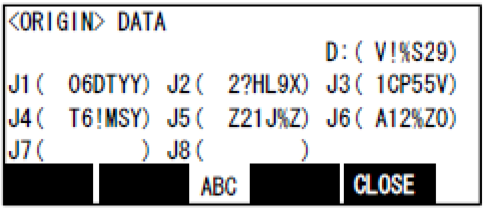
```
<ORIGIN>
1.DATA              2.MECH
3.TOOL              4.ABS
5.USER
              123         CLOSE
```

[1] key is pressed again to select DATA.



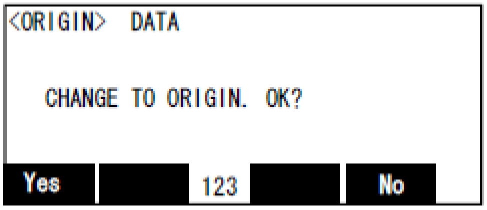
```
<ORIGIN> DATA
                               D:(■      )
J1(          ) J2(          ) J3(        )
J4(          ) J5(          ) J6(        )
J7(          ) J8(          )
              123         CLOSE
```

With the help of cursor and T/B numerical/alphabetical keypad, the data values are entered that was given on the back side of J1 motor cover initially noted.



```
<ORIGIN> DATA
                               D:( V!%S29)
J1(  O6DTYY) J2(  2?HL9X) J3( 1CP55V)
J4(  T6!MSY) J5(  Z21J%Z) J6( A12%ZO)
J7(         ) J8(         )
              ABC         CLOSE
```

After entering all the values, [EXE] on T/B is pressed. The origin setting confirmation screen would appear after pressing EXE.



```
<ORIGIN>  DATA

  CHANGE TO ORIGIN. OK?

Yes           123          No
```

[F1] (Yes) is pressed to end the origin setting.

After the origin settings, the operation is confirmed by moving the robot manually using JOING jog operation.

***ABS Origin method***: When the origin setting of the robot is performed for the first time, this product records the angular position of the origin within one rotation of the encoder as the offset value. If the origin settings are performed using ABS origin method, this value is used to suppress variations in the origin setting operation and to reproduce the initial origin position accurately. First, ABS arrow mark of the axis for which the origin is being set, is set from JOG Operation. This can be set for all axis simultaneously or each axis independently. Each robot axis is moved by manual mode such that ABS mark for each axis gets aligned.
When setting the ABS mark, operation is always viewed from the mark and set at the end of triangular mark. The position of ABS mark on the robot are shown in the following figure.

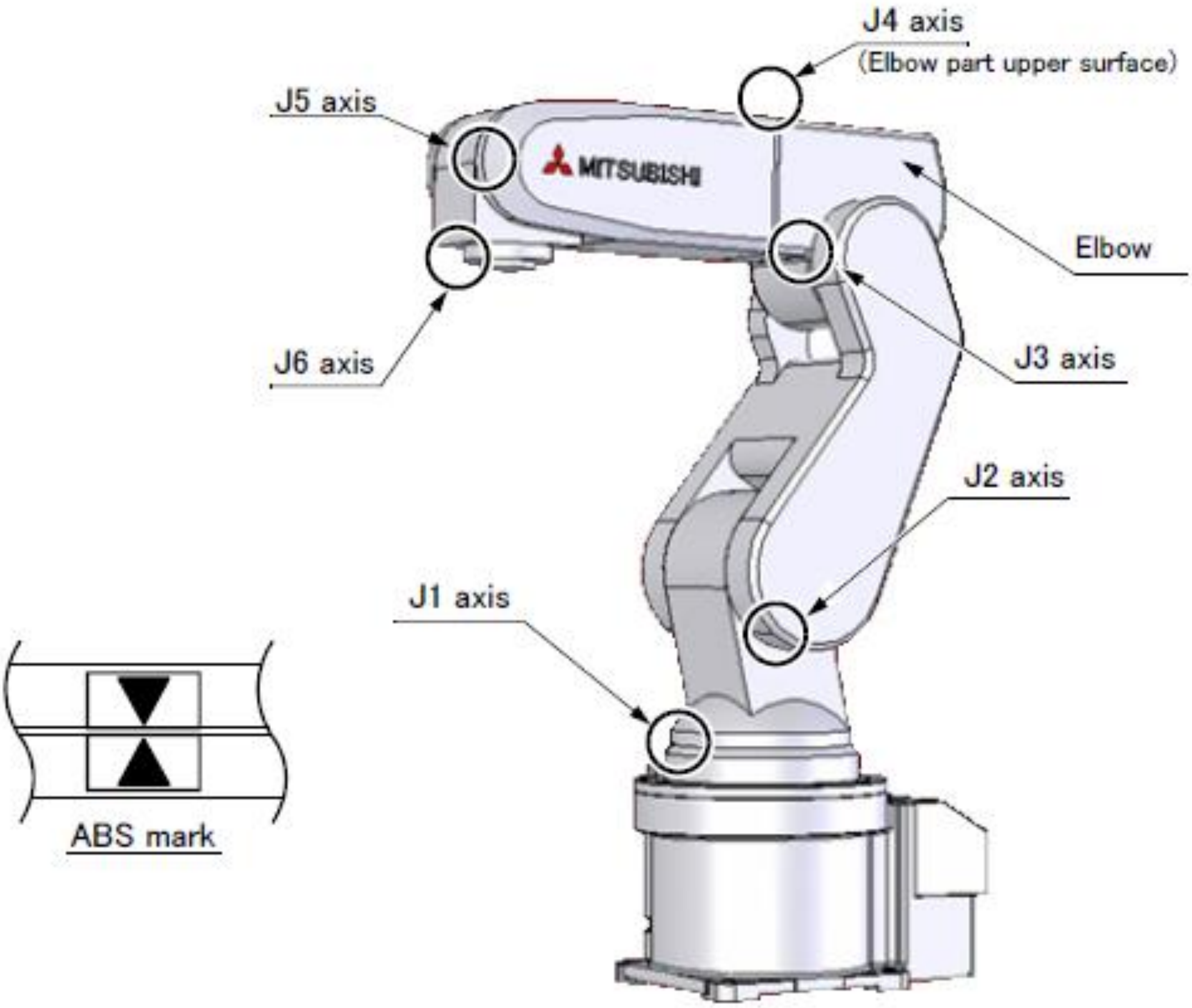
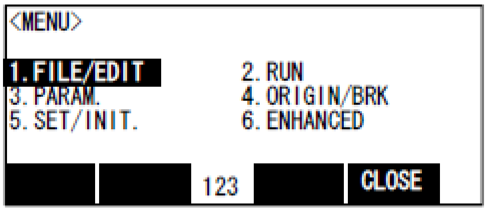
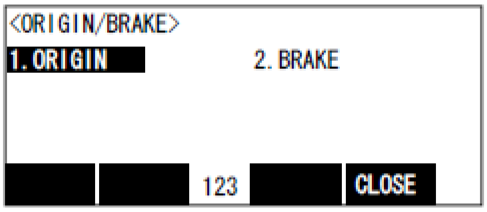


*Figure 5-8: ABS Mark Attachment Position*

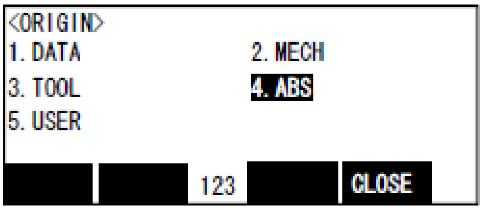
ABS origin setting method is performed as followed:



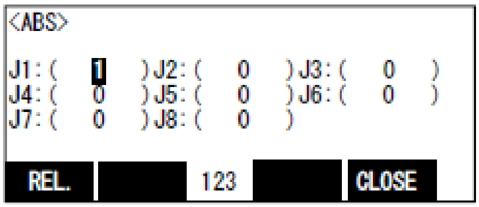
[4] key is pressed and ORIGIN/BRK is selected.



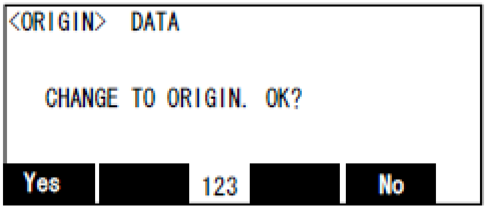
[1] key is pressed to select ORIGIN.



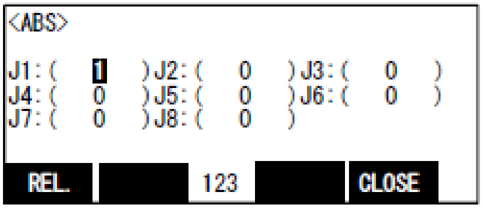
[4] is pressed to select ABS method of origin.


```
<ABS>
J1:(  1  )J2:(  0  )J3:(  0  )
J4:(  0  )J5:(  0  )J6:(  0  )
J7:(  0  )J8:(  0  )
```

‘1’ is input into the axis of origin setting. Then [EXE] is pressed for confirmation screen.


```
<ORIGIN>  DATA

  CHANGE TO ORIGIN. OK?
```

[F1] is pressed to set the origin.


```
<ABS>
J1:(  1  )J2:(  0  )J3:(  0  )
J4:(  0  )J5:(  0  )J6:(  0  )
J7:(  0  )J8:(  0  )
```

This completes the settings of origin by ABS method.

Other origin setting methods are as followed:

| No. | Method | Explanation |
|---|---|---|
| 1 | Origin Data Input Method | The origin data set as default is input from the T/B. |
| 2 | Mechanical Stopper Method | The origin posture is set by contacting each axis against the mechanical stopper. |
| 3 | Jig Method | The origin posture is set with the calibration jig installed. |

| | | |
|---|---|---|
| 4 | ABS Origin Method | This method is used when the encoder backup data lost in the cause such as battery cutting. |
| 5 | User Origin Method | A randomly designated position is set as the origin posture. |

Table 5-1: Origin Setting Method

## 5.6. Storing Current Position of the Robot

To teach the robot in order to go to a certain position of the Tag, it should be operated manually. After enabling the Manual mode and T/B, CIROS Studio is opened and a new Position List is generated for a new project. Then Servo Motor of the robot is turned on by keep pressing the Dead Man Switch or Enable key at the back side of T/B and pressing SERVO key on T/B.

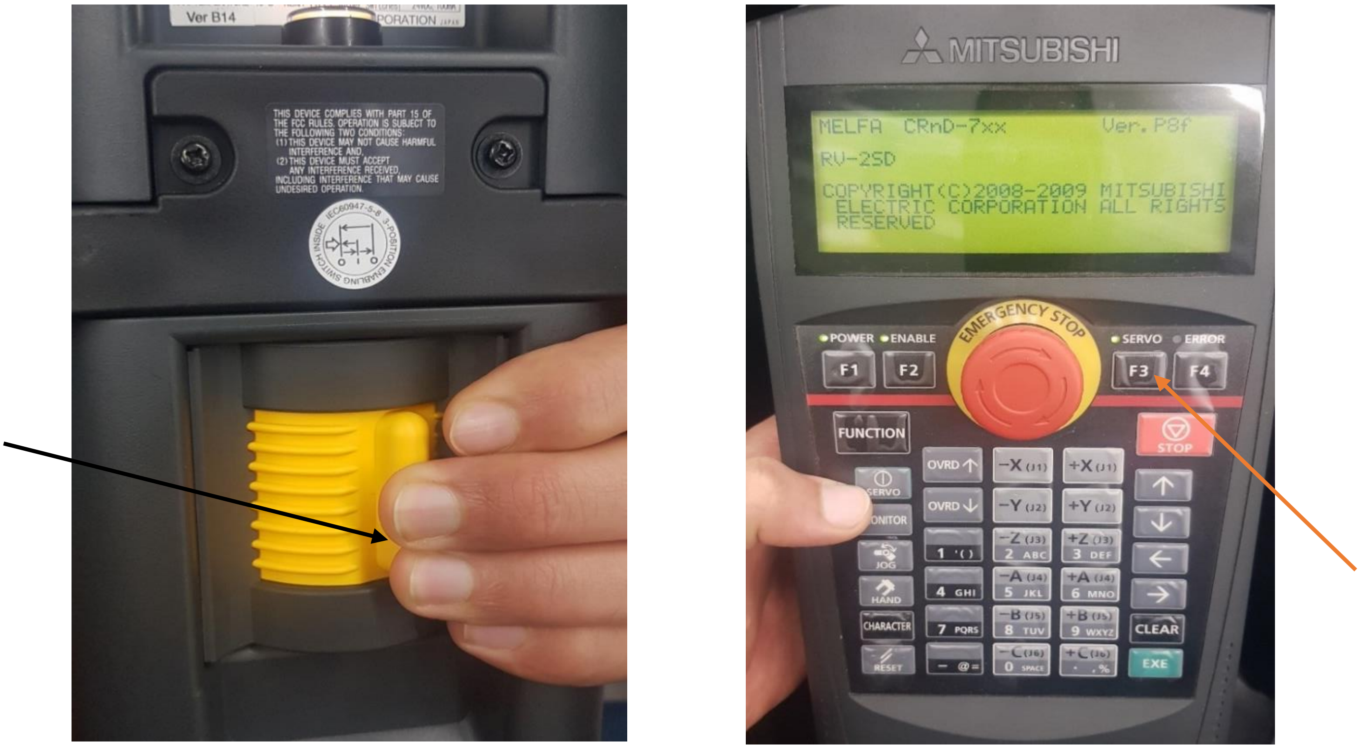


Figure 5-9: T/B Settings for SERVO

The robot is moved through JOG operations. After displaying JOG window, it must be made sure that JOINT operation is enabled at the top of window. If there is some other option being displayed on the screen, corresponding function keys must be pressed to make JOINT appear on the screen. This is done because each axis can be moved independently through this operation and further details about this operation are shown later.

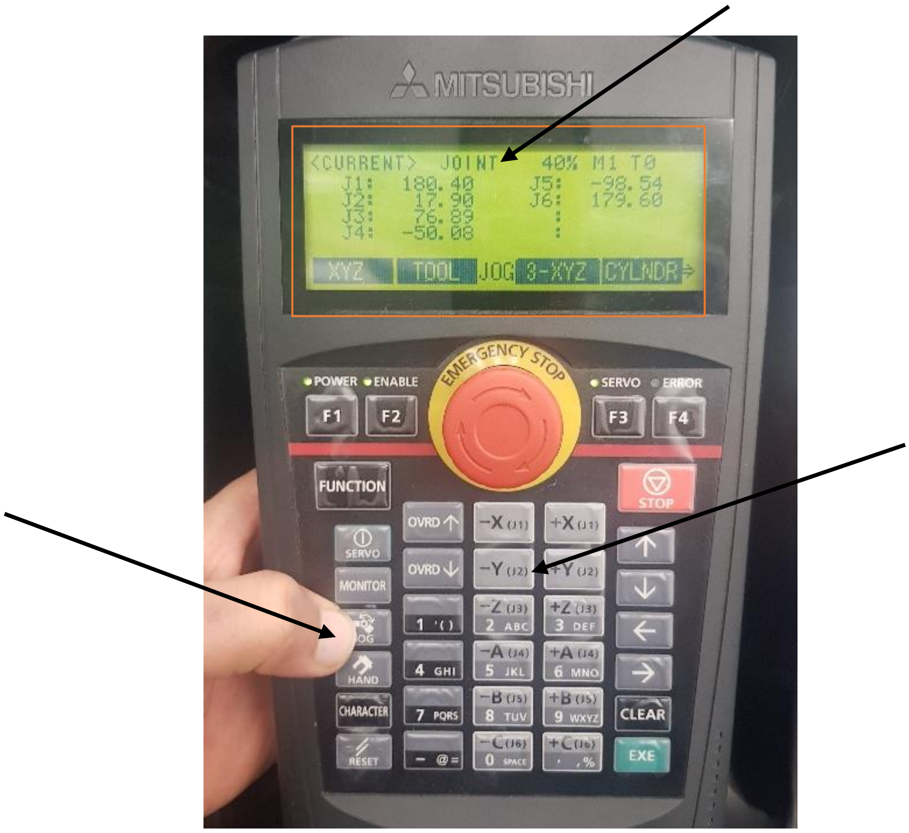


Figure 5-10: T/B Settings for Movement of Robot

***JOINT Jog:*** In this mode, each of the axes can be adjusted independently. It is possible to adjust the coordinates of the axes J1 to J6 as well as the additional axes J7 and J8 independently. Note that the exact number of axes may be different depending on the type of robot, however.

The additional axis keys [J1] and [J2] correspond to axes J7 and J8, respectively.

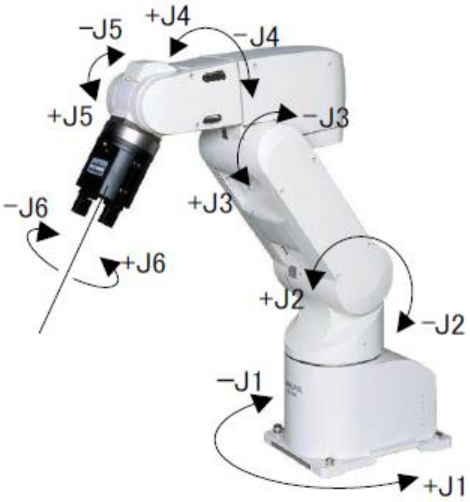


Figure 5-11: Joint Jog Movements of Robot

Some other types of JOG feed are also available, and the robot could be moved accordingly:

- TOOL Jog
- XYZ Jog
- 3-axis XYZ Jog
- CYLINDER Jog
- WORK Jog
- WORK Jog (Ex-T Jog mode)

| Jog Modes | Main Application | Explanation |
|---|---|---|
| JOINT Jog | • Moves each joint<br>• Moves the robot arm largely<br>• Changes the robot posture | Explanation is provided about some methods in the later stage. |
| XYZ Jog | • Accurately sets the teaching position<br>• Moves the axis straight along XYZ coordinate system<br>• Moves the axis straight while maintaining the robot posture<br>• Changes the posture while maintaining the hand position | |
| TOOL Jog | • Accurately sets the teaching position<br>• Moves the axis straight along the hand direction<br>• Changes the posture while maintaining the hand position<br>• Rotates the hand while maintaining the hand position | |
| 3-Axis XYZ Jog | • When the axis cannot be moved with XYZ Jog that maintains the posture<br>• When the tip is to be moved linearly but the posture is to be changed | |
| CYLINDER Jog | • Moves in the cylindrical shape centering on the Z axis while maintaining the posture<br>• Moves linearly in a radial shape centering on the Z axis while maintaining the posture | |

*Table 5-2: Jog Modes*

Taking the safety of surroundings and the robot itself into account, it is then moved to a desired position. Then through CIROS Studio, real time position data is stored into PC. The robot position data defines the coordinates according to the set origin. All the movements are relevant to the origin and the settings of origin have already been explained.

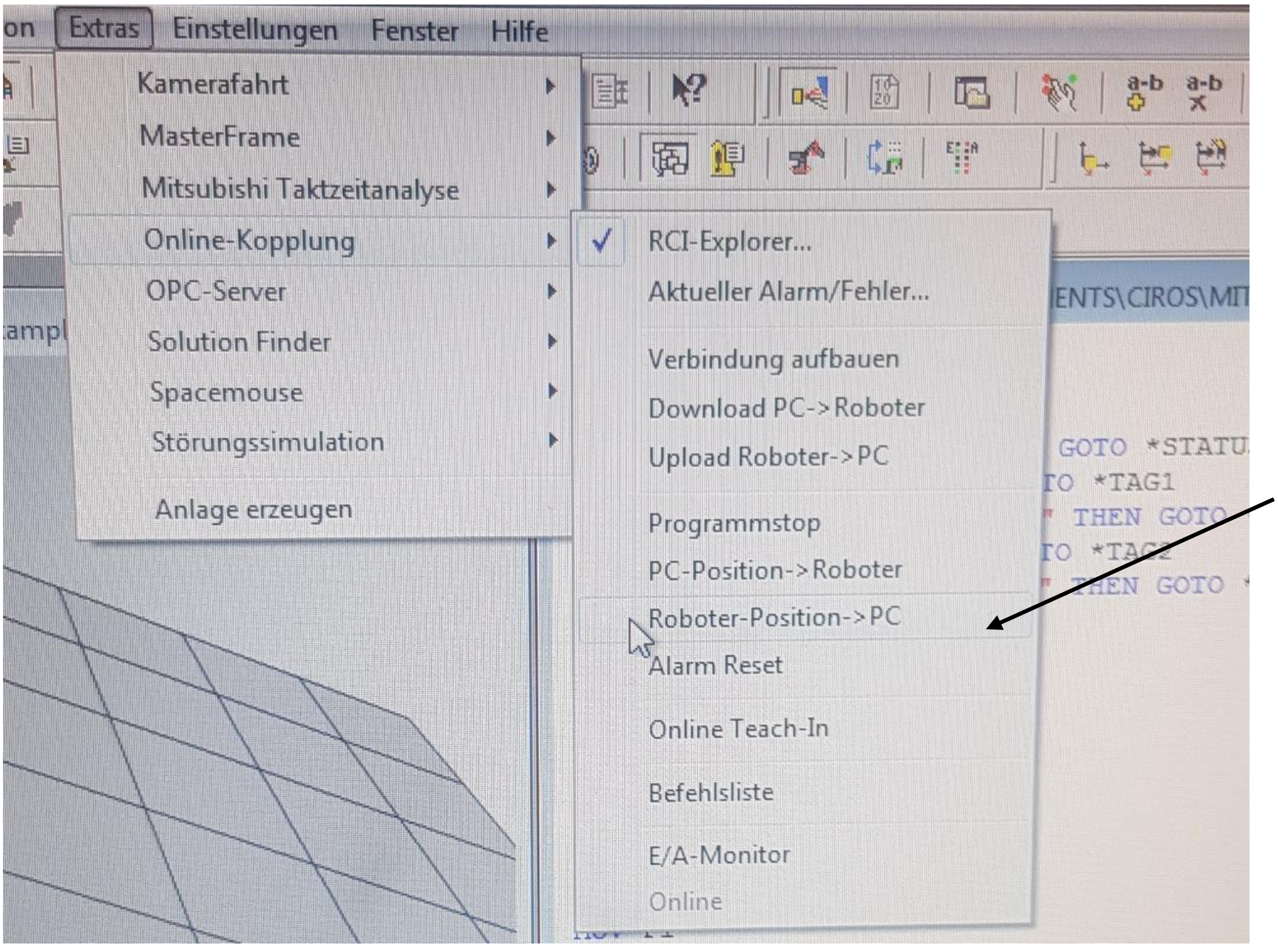
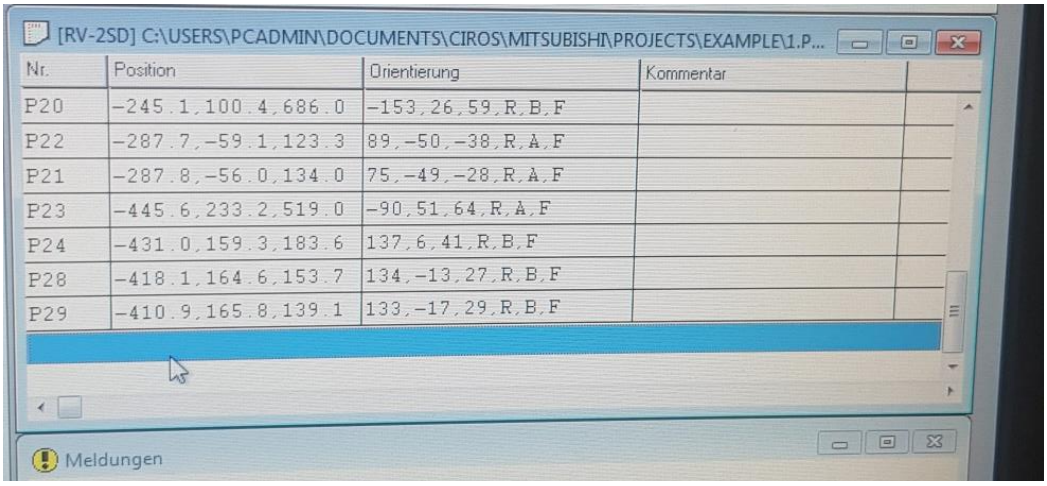


*Figure 5-12: Feeding Current Position of Robot into CIROS Studio*

Correct storage of real time position of the robot can also be verified through simulation window of CIROS Studio as the robot also moves to exact position into simulation window as well. Then in order to store position coordinates, an empty space on position list is reserved by clicking on it, then **SHIFT** and **F2** are pressed simultaneously from keyboard to store the coordinate as a new entry on position list.

These position lists are reference coordinates for the robot because in Manual mode, the robot is moved to exact same coordinates and when the positions are stored and fed to the robot, it remembers the coordinates and follows the path according to movement commands.



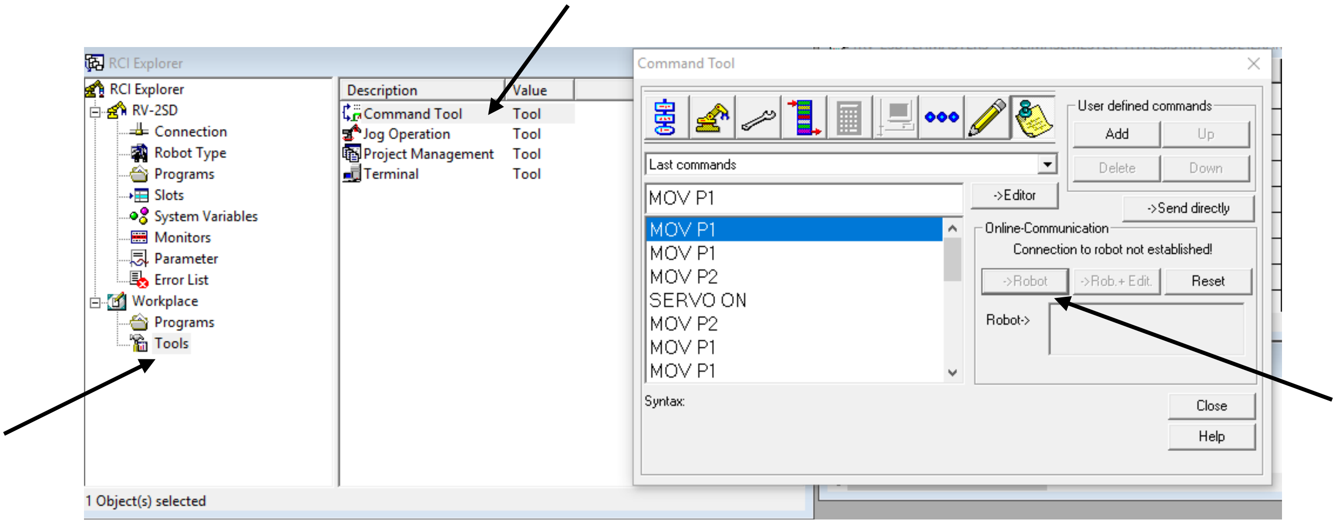
[RV-2SD] C:\USERS\PCADMIN\DOCUMENTS\CIROS\MITSUBISHI\PROJECTS\EXAMPLE\1.P...

| Nr. | Position | Orientierung | Kommentar |
|---|---|---|---|
| P20 | -245.1,100.4,686.0 | -153,26,59,R,B,F | |
| P22 | -287.7,-59.1,123.3 | 89,-50,-38,R,A,F | |
| P21 | -287.8,-56.0,134.0 | 75,-49,-28,R,A,F | |
| P23 | -445.6,233.2,519.0 | -90,51,64,R,A,F | |
| P24 | -431.0,159.3,183.6 | 137,6,41,R,B,F | |
| P28 | -418.1,164.6,153.7 | 134,-13,27,R,B,F | |
| P29 | -410.9,165.8,139.1 | 133,-17,29,R,B,F | |

Figure 5-13: Setting Real Time Coordinates on Position List

## 5.7. Sending Online Commands

In account of moving the robot directly from CIROS Studio or to send online commands directly, the robot must be set to Automatic Mode. After the robot has been set to automatic, the connection is established though TCP/IP configuration as explained earlier. To send direct commands through CIROS Studio, built in tools of CIROS Studio are used. In RCI Explorer, one of the tools is ***Command Tool***, that is used for direct commands communication and user commands modification.



| Description | Value |
|---|---|
| Command Tool | Tool |
| Jog Operation | Tool |
| Project Management | Tool |
| Terminal | Tool |

1 Object(s) selected

Command Tool

Last commands

```
MOV P1

MOV P1
MOV P1
MOV P2
SERVO ON
MOV P2
MOV P1
MOV P1
```

Connection to robot not established!

Figure 5-14: Command Tool of CIROS Studio

In the Command Tool window, many pre-defined commands are stored. But before using command tool, it must be made sure that the robot is in Automatic mode and the Servo is on. The servo can also be turned on by sending command through command tool. So, after storing the positions of arm through Manual operation and finalizing the position list, the program and position list is downloaded into the robot. It is to verify that the robot knows the positions P1, P2, P3 and so on through position list.

Now the robot can be moved directly form command tool by following steps:

- Servo is turned on by sending ***SERVO ON*** command through RCI Explorer
- ***MOV P1*** command would move the robot to the position P1 that was stored by Manual operation.
- ***HOPEN 1*** and ***HCLOSE 1*** commands would open or close the gripper respectively if the Air pressure is provided correctly.

There are so many commands that can be sent through RCI Explorer in order to move the robot directly from CIROS Studio. Then there is also an option in RCI Explorer for customized user commands that a user can define in programming window and can execute later directly from RCI Explorer.

## 5.8. Creating a Program using T/B

A MELFA Basic IV or V program can also be written, stored, edited and executed by using T/B. To create a program, following steps are followed:

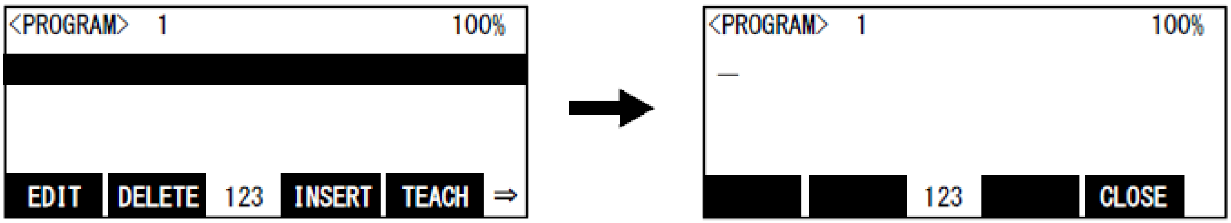
Take a simple program as a sample:

*1 Mov P1*

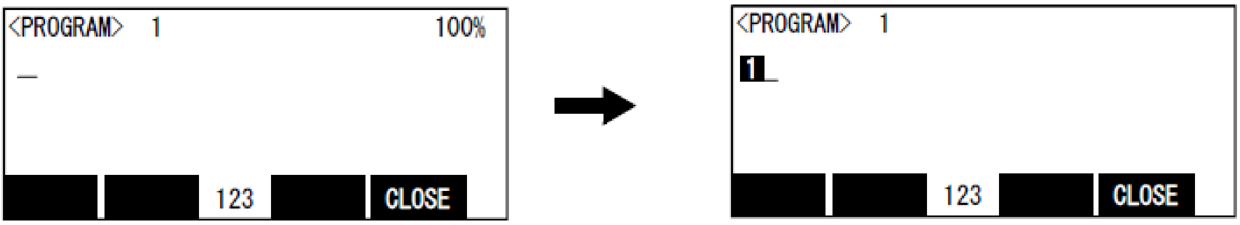
*2 Mov P2*

*3 End*

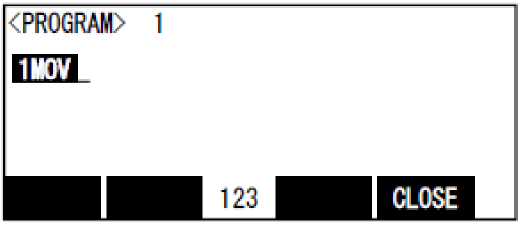
- Function key [F3] corresponding to “INSERT” in the command edit screen is pressed.



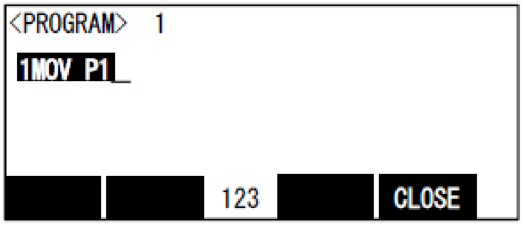
- [CHARACTER] key is pressed, number input mode is selected, and a name is given to the program such as ‘1’. The space between step number and command is omissible.



- [CHARACTER] key is pressed again, and character input mode is selected. [MNO] is pressed in order to write M, o, v.
- Similarly position number is entered such as P1, P2 or other number if there’s another position data.

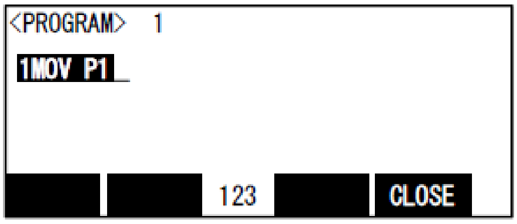


```
<PROGRAM>  1
1MOV
```

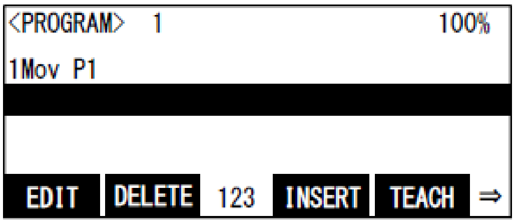


```
<PROGRAM>  1
1MOV P1
```

- [EXE] key is pressed to register the movement commands that have been just entered.

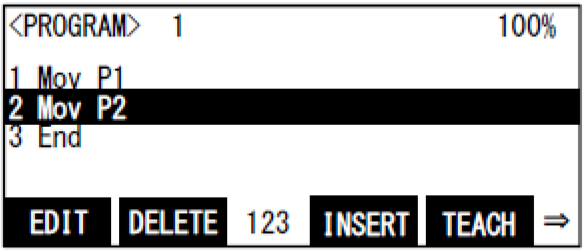


```
<PROGRAM>  1
1MOV P1
```



```
<PROGRAM>  1                    100%
1Mov P1
```

- Similarly, the other commands are entered.



```
<PROGRAM>  1                    100%
1 Mov P1
2 Mov P2
3 End
```

This is how the input of commands are completed using T/B. the function key which corresponds to “CLOSE” is pressed then, the program will be created and saved. If the “CLOSE” is not indicated, [FUNCTION] key is pressed to display it.

## 5.9. Precautions for Using the Robot

The safety measures for using the robot are specified in the "Labor Safety and Sanitation Rules". An outline of the rules is given below.

1. ***Robot installation***

- Secure and enough workspace is required to safely perform work such as teaching, and maintenance related to the robot.
- Installation of the controller is performed outside the robot's motion space. (If a safety fence is provided, install outside the fence.)
- Installation of the controller is done where the entire robot operation can be viewed.
- Installation of display lamps is performed to indicate the robot's operation state.

- The robot arm is fixed securely onto the fixing table with the designated bolts.

### 2. *Prevention of contact with operator*

- Installation of a safety fence or enclosure is done so that the operator cannot easily enter the robot's motion space.
- Installation of an interlock function is performed that will stop the robot if the safety fence or enclosure door is opened.

### 3. *Work procedures*

- Creating and observing work procedures for the robot teaching, operation, inspection and emergencies.
- Creation of hand signals to be followed when several operators are working together.
- Creation of displays such as "Teaching in Progress" and "Inspection in Progress" to be put up when an operator is in the robot's motion space so that other operators will not operate the operation panel (controller, control panel).

### 4. *Training*

- Training the operators about the operations, maintenance and safety required for the robot work.
- Only trained and registered operators must operate the robot.
- Participation in the "Special training for industrial robots" sponsored by the Labor Safety and Sanitation Committee etc. is recommended for safety training.

### 5. *Daily inspection and periodic inspection*

- The robot is inspected always before starting daily operations and confirmed that there are no abnormalities.
- Periodic inspection standards are set in view of the robot's ambient environment and operation frequency and perform periodic inspections.
- Records are made when periodic inspections and repairs have been done and the records are stored for three or more years.

### 6. *Precautions for handling:*

- RV-2SDB has brakes of all axes. The precision of the robot may drop, looseness may occur, and the reduction gears may be damaged if the robot is moved with force with the brakes applied. Moreover, when the axis without the brake is servo-off, take care to falling by the self-weight.
- Avoid moving the robot arm by hand. When unavoidable, gradually move the arm. If moved suddenly, the accuracy may drop due to an excessive backlash, or the backed-up data may be destroyed. Note that depending on the posture, even when within the movement range, the section could interfere with the base section. Take care to prevent interference during jog.

- The robot arm is configured of precision parts such as bearings. Grease is used for lubricating these parts. When cold starting at low temperatures or starting operation after long-term stoppage, the position accuracy may drop, or servo alarms may occur. If these problems occur, perform a 5 to 10-minute running-in operation at a low speed (about a half of normal operating speed).
- The robot arm and controller must be grounded with Class D grounding to secure the noise resistance and to prevent electric shocks.
- The items described in these specifications are conditions for carrying out the periodic maintenance and inspections described in the instruction manual.
- When using the robot arm on a mobile axis or elevating table, the machine cables enclosed as standard configuration may break due to the fixed installation specifications. In this case, use the machine cable extension (for flexed)" factory shipment special specifications or options.
- If this robot interferes with the workpiece or peripheral devices during operation, the position may deviate, etc. Take care to prevent interference with the workpiece or peripheral devices during operation.
- Do not attach a tape or a label to the robot arm and the controller. If a tape or a label with strong adhesive power, such as a packaging tape, is attached to the coated surfaces of the robot arm and controller, the coated surface may be damaged when such tape or label is peeled off.
- If the robot is operated with a heavy load and at a high speed, the surface of the robot arm gets very hot. It would not result in burns; however, it may cause secondary accidents if touched carelessly.
- Do not shut down the input power supply to stop the robot. If the power supply is frequently shut down during a heavy load or high-speed operation, the speed reducer may be damaged, backlash may occur, and the program data may be destroyed.
- During the robot's automatic operation, a break is applied to the robot arm when the input power supply is shut down by a power failure, for instance. When a break is applied, the arm may deviate from the operation path predetermined by automatic operation and, as a result, it may interfere with the mechanical stopper depending on the operation at shutdown. In such a case, take an appropriate measure in advance to prevent any dangerous situation from occurring due to the interference between the arm and peripheral devices.
- Example) Installing a UPS (uninterruptible power supply unit) to the primary power source in order to reduce interference.
- Do not conduct an insulated voltage test. If conducted by mistake, it may result in a breakdown.

- Fretting may occur on the axis which moving angle or moving distance move minutely, or not moves. Fretting is that the required oil film becomes hard to be formed if the moving angle is small, and wear occurs. The axis which not moved is moving slightly by vibration etc. To make no fretting recommends moving these axes about once every day the 30 degree or more, or the 30mm or more.
- The United Nations' Recommendations on the Transport of Dangerous Goods must be observed for transborder transportation of lithium batteries by air, sea, and land. The lithium batteries (Q6BAT, ER6) used in Mitsubishi industrial robots contain less than 1 g of lithium and are not classified as dangerous goods. However, if the quantity of lithium batteries exceeds 24 batteries for storage, etc., they will be classified as Class 9: Miscellaneous dangerous substances and articles. Shipping less than 24 batteries is recommended to avoid having to carry out transport safety measures as the customer' s consignor [14]. Note that some transportation companies may request an indication that the batteries are not dangerous goods be included on the invoice. For shipping requirement details, please contact your transportation company.

### 5.10. Safety

***Safety measures for Automatic operation:***

- Installation of safety fences is done so that operators will not enter the operation area during operation and indicate that automatic operation is in progress with lamps, etc.
- Creation of signals to be given when starting operation, a person can be assigned to give the signal, and make sure that the operator follows the signals.

***Safety measures for teaching:***

The following measures are observed when teaching in the robot's operation range.

- To Specify and follow items such as procedures related to teaching work, etc.
- To take measures so that operation can be stopped immediately in case of trouble, and measures so that operation can be restarted.
- To take measures with the robot start switch, etc., to indicate that teaching work is being done.
- To always inspect that stop functions such as the emergency stop device before starting the work.
- To immediately stop the work when trouble occurs and correct the trouble.
- To take measures so that the work supervisor can immediately stop the robot operation when trouble occurs.

- The teaching operator must have completed special training regarding safety. (Training regarding industrial robots and work methods, etc.)
- Create signals to be used when several operators are working together.

***Safety measures for maintenance and inspections:***

The power is turned OFF and measures are taken to prevent operators other than the relevant operator from pressing the start switch when performing inspections, repairs, adjustments, cleaning or oiling. If operation is required, take measures to prevent hazards caused by unintentional or mistaken operations.

- To specify and follow items such as procedures related to maintenance work, etc.
- To take measures so that operation can be stopped immediately in case of trouble, and measures so that operation can be restarted.
- To take measures with the robot start switch, etc., to indicate that work is being done.
- To take measures so that the work supervisor can immediately stop the robot operation when trouble occurs.
- The operator must have completed special training regarding safety. (Training regarding industrial robots and work methods, etc.)
- Create signals to be used when several operators are working together.

# 6. Modeling and Simulation

This chapter elaborates the project model through integrated flowchart and Petri Net models. It also shows the experimental analysis of the whole project from the beginning to the final stage.

## 6.1. Petri Net Model

A Petri Net is a bipartite graph consisting of two types of nodes, namely places (represented by a circle) and transitions (represented by a rectangle) [15].
Referring to the following example, the nodes labeled P1 and P2 are called *places* and the node labeled A is called a *transition*. A place can be connected to a transition and vice-versa by means of directed edges called arcs. The notion of a token gives a Petri Net its behavior. Tokens reside in places and often represent either a status, an activity, a resource or an object. The distribution of tokens in a Petri net is called its marking or state. Transitions represent events of a system. The occurrence of an event is defined by the notion of transition enabling and firing. A transition is enabled if all its input places have at least one token each. When an enabled transition fires it consumes one token from each input place and produces one token in each output place, i.e. changes the state.

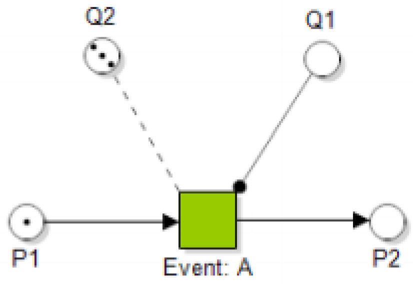


*Figure 6-1: Petri Net*

The state of a Petri net $N = (P, T, F)$ is determined by its marking which represents the distribution of tokens over places of the net. A marking m of a Petri net N is a bag over its places P, i.e. $m \in B(P)$.
A transition $t \in T$ is enabled in some marking m if and only if:

$$\bullet t(p) \leq m(p), \text{ for all places } p \in P$$

If N is an inhibitor net then for the enabling of transition t from marking m, we also require that $m(p) = 0$ for all places $p \in \iota(t)$. An enabled transition t from marking m may fire, resulting in a new marking $m'(p)$ with:

$$m'(p) = m(p) - \bullet t(p) + t\bullet(p); \qquad \text{if } p \in P \setminus \rho(t)$$

and,

$$m'(p) = 0; \quad \text{if } p \in \rho(t)$$

and, is denoted by:

$$m\ (t) \rightarrow m'$$

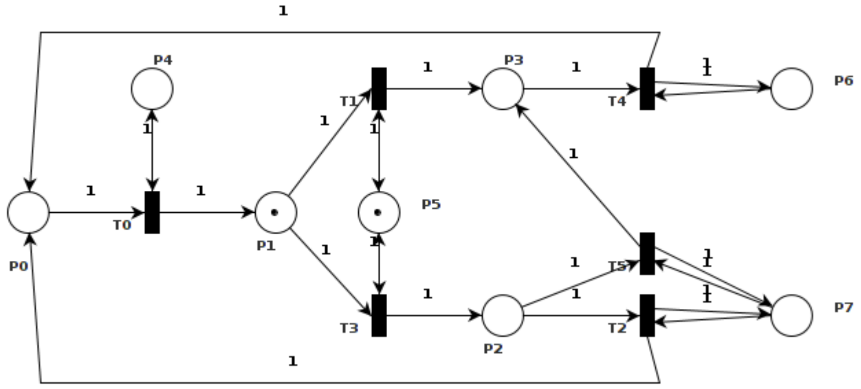


*Figure 6-2: States, Transitions and Firing of a Marked Petri Net*

An example of a Marked Petri Net is depicted in above figure, where graphically:

- Places are represented by circles,
- Transitions are represented by filled rectangles,
- Arcs are represented by arrows,
- Tokens are represented by dots inside places.

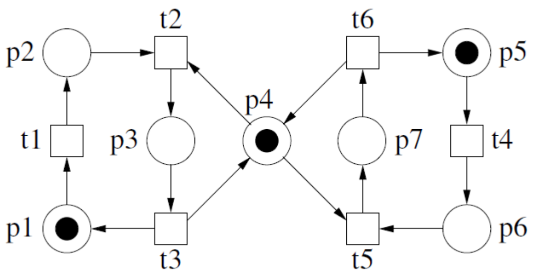
The evolution through different states (or markings) of the Marked Petri Net is achieved by firing enabled transitions. Although more than one transition can be enabled at the same time, only one will fire.

**Properties of a Petri Net:** The properties of a Petri Net are classified as either behavioral or structural [16]. Some behavioral properties include:

- Boundedness
- Reachability
- Reversibility
- Coverability
- Persistence
- Liveness
- Synchronic Distance
- Fairness

The analysis methods for Petri nets may be classified as reachability, matrix-equation approach, refinement and reduction techniques. The techniques for refinement and reduction facilitate the analysis of complex systems by means of behaviour preserving transformations. The former refines an abstract model into a more detailed model in a hierarchical manner, while the latter reduces a system model into a simpler one, while preserving behavioural properties like liveness and boundedness. So, the former is a synthesis technique while the latter is applied to ease system analysis. The structural analysis of Petri Nets relies on the concepts of **place/transition invariants**.

Let N be a net and C its incidence matrix. A natural solution of the equation $C^T x = 0$ such that $x \neq 0$ is called a place invariant (or: P-invariant) of N. Notice that a P-invariant is a vector with one entry for each place.



For instance, in above Petri Net, $x1 = (1\ 1\ 1\ 0\ 0\ 0\ 0)^T$, $x2 = (0\ 0\ 1\ 1\ 0\ 0\ 1)^T$, and $x3 = (0\ 0\ 0\ 0\ 1\ 1\ 1)^T$ are all P-invariants. A P-invariant indicates that the number of tokens in all reachable markings satisfies some linear invariant.

The liveness property of a Petri Net states that "Something good will eventually occur" contrasting a safety property which states that "Something bad does not occur". A liveness property cannot be violated in a finite execution of a distributed system because the "good" event might still occur at some later time. All properties can be expressed as the intersection of safety and liveness properties. Several forms of liveness can be recognized. For instance, *Freedom from deadlock* or *Freedom from starvation (finite bypass)*. Let's take the following example.

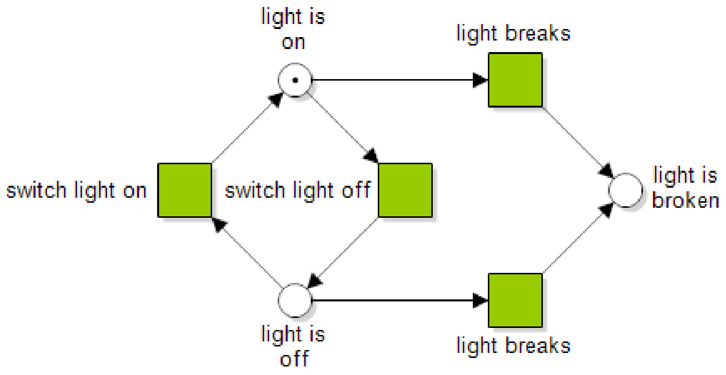


Figure 6-3: Reference Petri Net

In the example above, the marking has only one token, in *light is on*. Only two transitions can fire: the upper *light breaks* and *light is off*. If *light breaks* fires, we end up in the marking that has one token in *light is broken* and no tokens elsewhere. After that, no transition can ever fire again. If instead, *switch light off* fires, we end up in the marking that has one token in *light is off* and no tokens elsewhere. In that marking, two transitions can fire, namely the lower *light breaks* and *switch light on*. If *light breaks* fires, we end up in the marking that has one token in *light is broken* and no tokens elsewhere. After that, no transition can ever fire again. If instead, *switch light on* fires, we end up in the marking that we started with. With that out of the way, we can describe ***quasi-liveness*** and ***liveness***:

A transition is said to be ***quasi-live*** if it can be fired at least once. That is, from the present marking, either it can fire directly, or by firing transitions it is possible to arrive in a marking in which it can fire directly. In other words: in the present situation, there is a way in which this change can occur, either now or in the future. In the example above, every transition is quasi-live.

A marking is said to be *dead*, *a* ***deadlock***, if no transitions can fire. *No* changes can ever occur. This is the case for the marking we reach after firing one of the *light breaks* transitions.

A transition is said to be ***live*** if it is quasi-live in every possible marking. That is to say: in *any* situation, that change can still occur, either right away or later. In the example above, no transition is live: we have found a dead marking, in which none of them can fire. A Petri net is said to be ***live*** if all its transitions are live. In any situation, all the changes can still occur. Clearly this is not the case in the example above.

Petri Nets have unlimited applications in different fields of science. They are used to build models of the system and to integrate the finite state machine or to represent a robotic system into its development states. The designed Petri Net model of our system is shown as below.

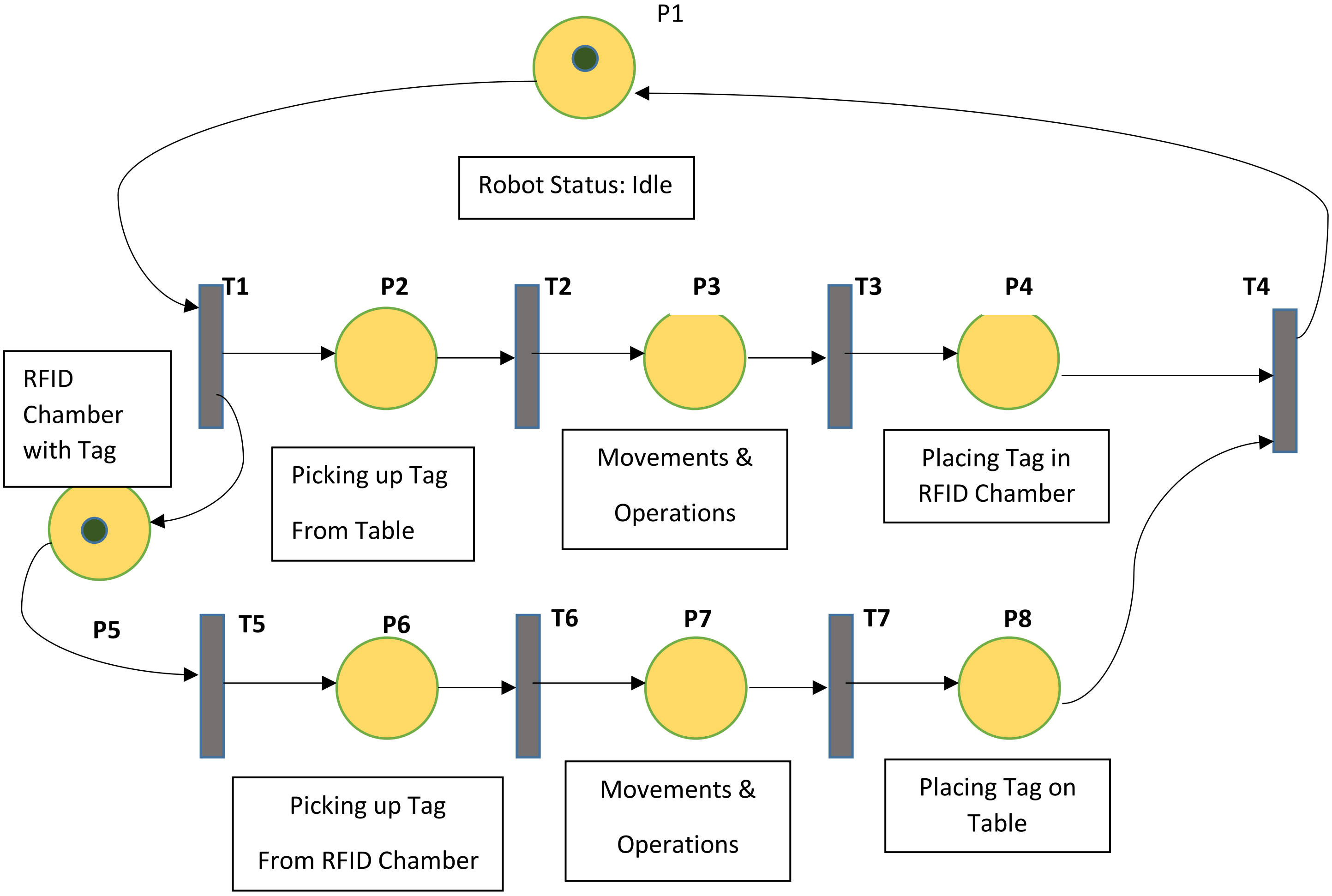


*Figure 6-4: Petri Net Model of the Project*

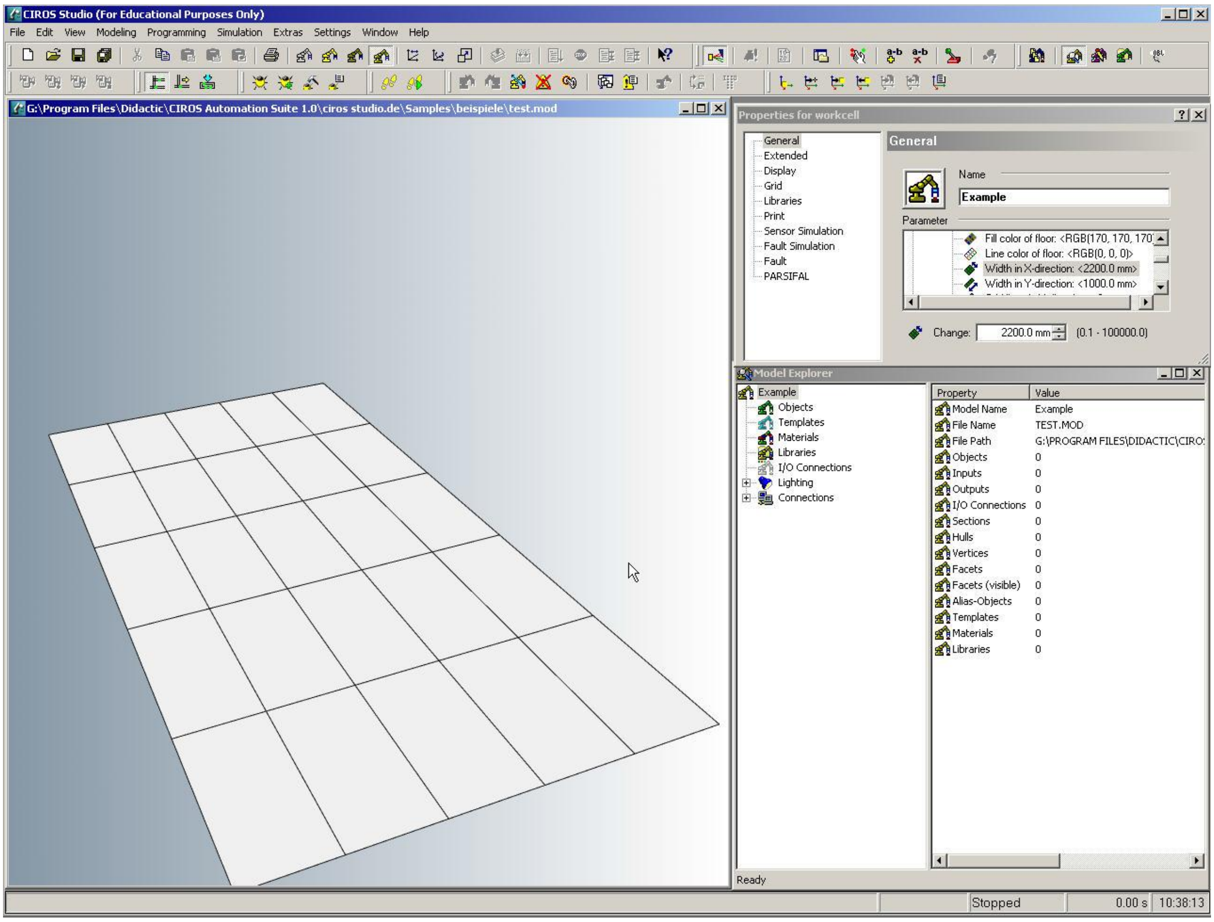
Here, an example of one tag is considered. The robot's safe position is an idle state where it returns after each task execution i.e. P1. From idle state P1, it has two possibilities and the further movements of robot depend on the commands it has been given. When T1 is fired, either it can go to pick a tag from the table i.e. state P2, or it can go to RFID chamber to pick a tag from the chamber i.e. P5. After T2 is fired, that there is a series of coupled movements. And then the robot decides the task according to initial command i.e. P3. If it had picked a tag from the table, then T3 is fired and it places the tag in RFID measurement cabinet and goes to state P4, on the contrary, T5 is fired and it picks the tag from chamber and goes to P6. T6 is fired and after a series of adjustments, it goes to P7. Then T7 is fired and it places the on the right position on the table ending up into state P8. Then finally, it goes back to safe state i.e. P1 after T4 has been fired.

The model presented here is not based on architectural level. Of course, there are a lot of internal complex structures that have been not shown here in this model. But this model provides an overall complete idea of the task under consideration.

## 6.2. Modeling on CIROS Studio

Let's have an example of our system with just one tag. Now before actual experiment in real time, the system of a simple pick and drop scenario is modeled and tested on CIROS Studio. For that purpose, a simple work cell is designed as follows:

- In CIROS Studio, command **New Work** cell from the **File** menu is chosen to create a new work cell. The filename (e. g. "Example. mod") for the new work cell is specified. After creating the new work cell, a different work cell name as well as properties for the work cell (e. g. background colour, floor colour, and floor size) can be specified. The dialog **Properties for Work Cell** is used to change the properties of Work Cell. X-values, Y-values and other attributes can be changed in order to adjust the length, width or height of the work cell.



Name: Example

Fill color of floor: <RGB(170, 170, 170

Line color of floor: <RGB(0, 0, 0)>

Width in X-direction: <2200.0 mm>

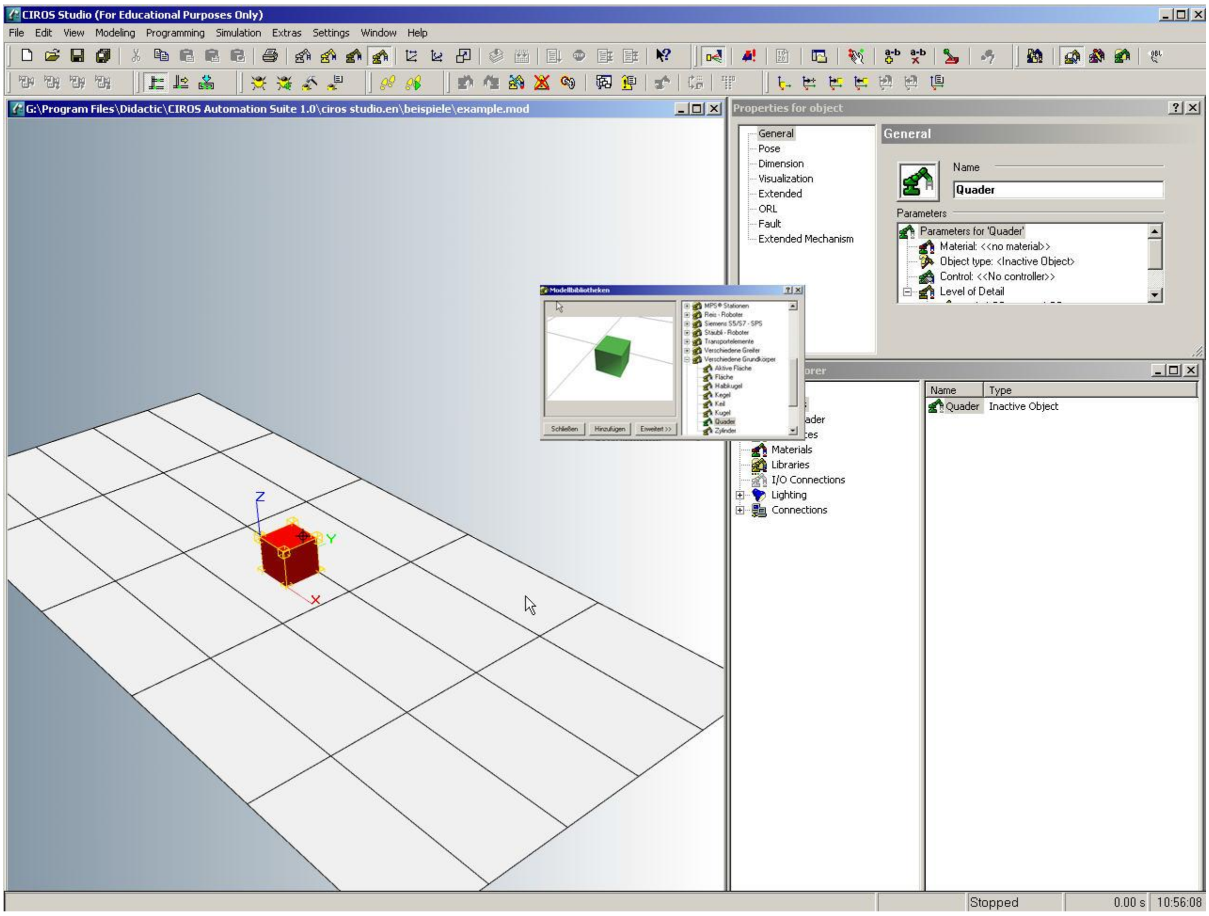
Width in Y-direction: <1000.0 mm>

Change: 2200.0 mm (0.1 - 100000.0)

| Property | Value |
|---|---|
| Model Name | Example |
| File Name | TEST.MOD |
| File Path | G:\PROGRAM FILES\DIDACTIC\CIRO: |
| Objects | 0 |
| Inputs | 0 |
| Outputs | 0 |
| I/O Connections | 0 |
| Sections | 0 |
| Hulls | 0 |
| Vertices | 0 |
| Facets | 0 |
| Facets (visible) | 0 |
| Alias-Objects | 0 |
| Templates | 0 |
| Materials | 0 |
| Libraries | 0 |

*Figure 6-5: Work Cell creation in CIROS Studio*

- From **Modeling** menu, **Model Libraries** are selected to add different items in the work cell. A box is added from **Miscellaneous Primitive** model library. The box is an analogue to the tag in real time. Element properties can be edited in **Properties for Object** dialog.



Name: Quader

Parameters for 'Quader'
Material: <<no material>>
Object type: <Inactive Object>
Control: <<No controller>>
Level of Detail

| Name | Type |
|---|---|
| Quader | Inactive Object |

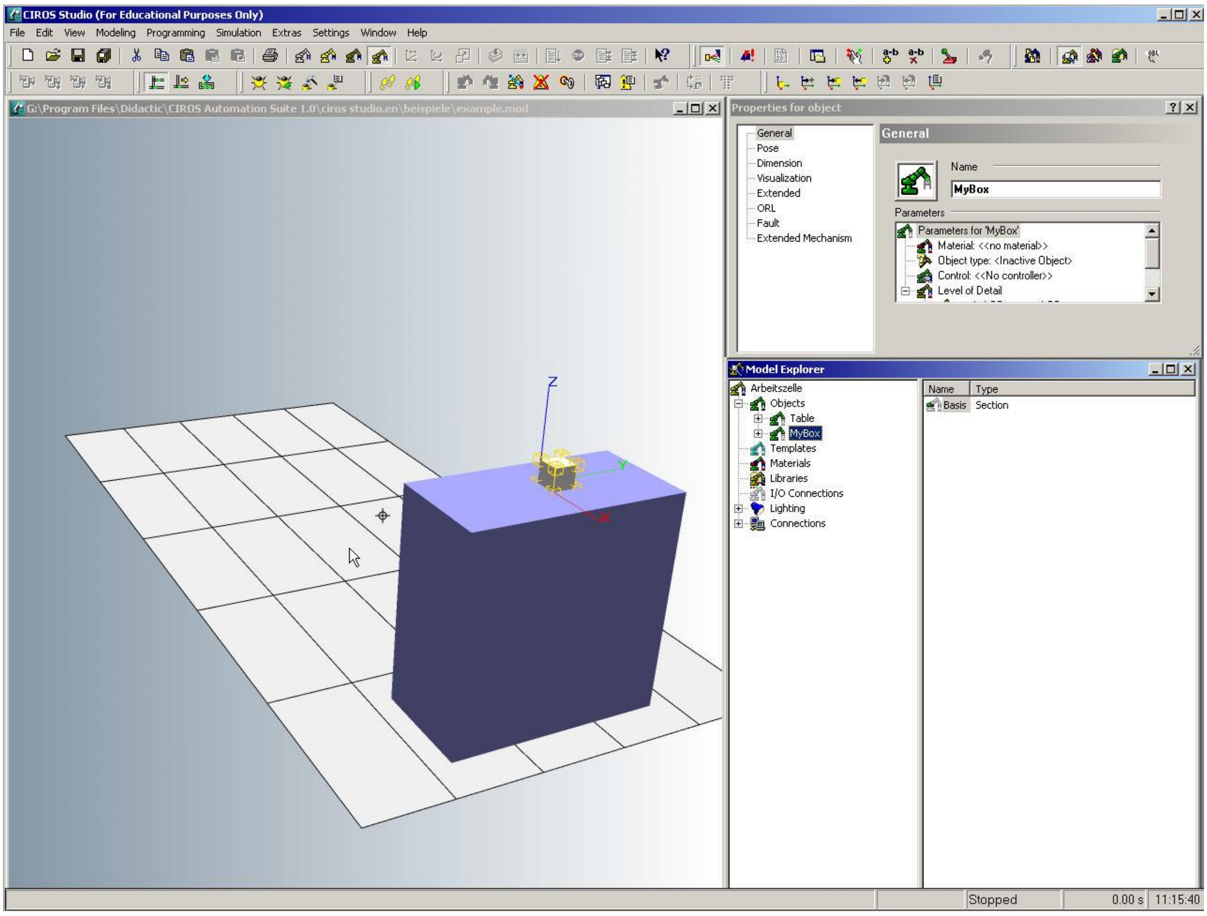
Stopped 0.00 s 10:56:08

*Figure 6-6: Table Addition in Work Cell*

- For instance, in order to make it look more like a working environment or a table to put the tags on, following properties may be adjusted.
    - Pose (x, y, z)
    - Dimension (x, y, z)
    - Visualization

  After setting these properties, the object is renamed from “Box” to “Table” in model explorer.
- A second box is added in the work cell and named as Tag/MyBox and following properties of the new box are edited:

| | |
|---|---|
| Pose (x, y, z) | 850mm, 0mm, 500mm |
| Dimension (x, y, z) | 50mm, 50mm, 50mm |
| Visualization | Light Grey |


CIROS Studio (For Educational Purposes Only)
File Edit View Modeling Programming Simulation Extras Settings Window Help
G:\Program Files\Didactic\CIROS Automation Suite 1.0\ciros studio.en\beispiele\example.mod
Z
Y
X
Properties for object
General
Pose
Dimension
Visualization
Extended
ORL
Fault
Extended Mechanism
Model Explorer
Arbeitszelle
Objects
Table
MyBox
Templates
Materials
Libraries
I/O Connections
Lighting
Connections

General

Name: MyBox

Parameters

Parameters for 'MyBox'
- Material: <<no material>>
- Object type: <Inactive Object>
- Control: <<No controller>>
- Level of Detail

| Name | Type |
|---|---|
| Basis | Section |

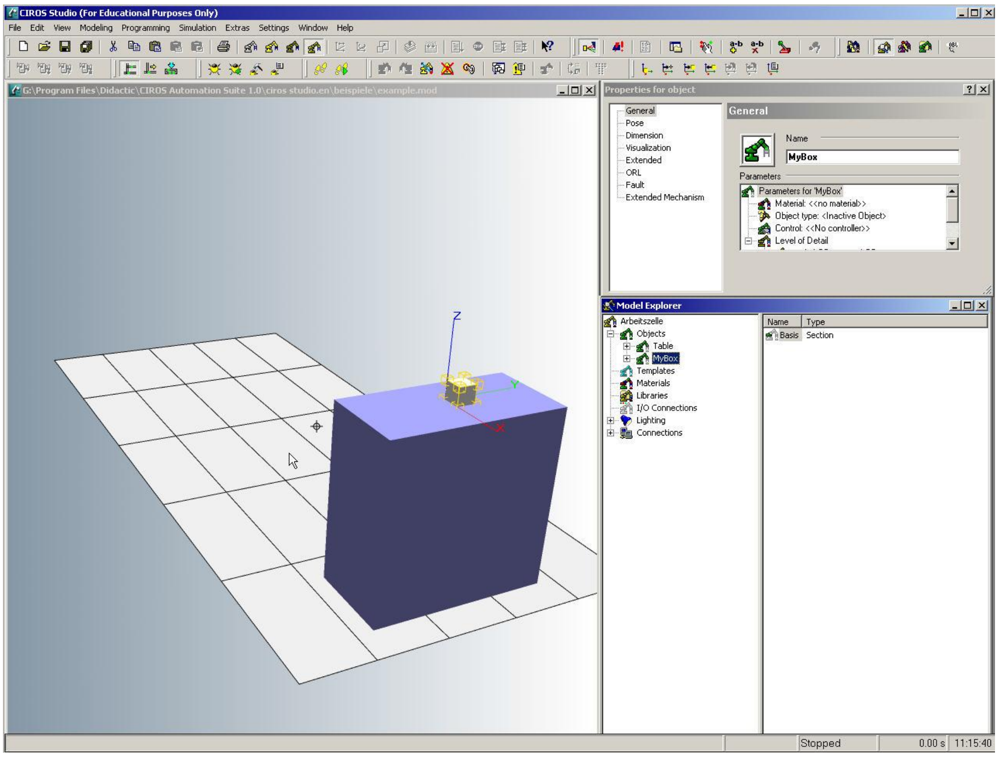
Stopped 0.00 s 11:15:40

Figure 6-7: Tag/Box Addition in Work Cell

- To let a tag/box be grabbed by the robot, a grip point is assigned to the workpiece. **Base** group of the object MyBox is selected and in the context menu, command **New** -> **Grip point** is selected.

  Grip point is renamed as **Workpiece** and is moved in the centre of the object by altering the coordinates relative to the section coordinate system:

| | |
|---|---|
| Pose (x, y, z) | 25mm, 25mm, 25mm |
| (R, P, Y) | 180 °, 0 °, 180 ° |



General

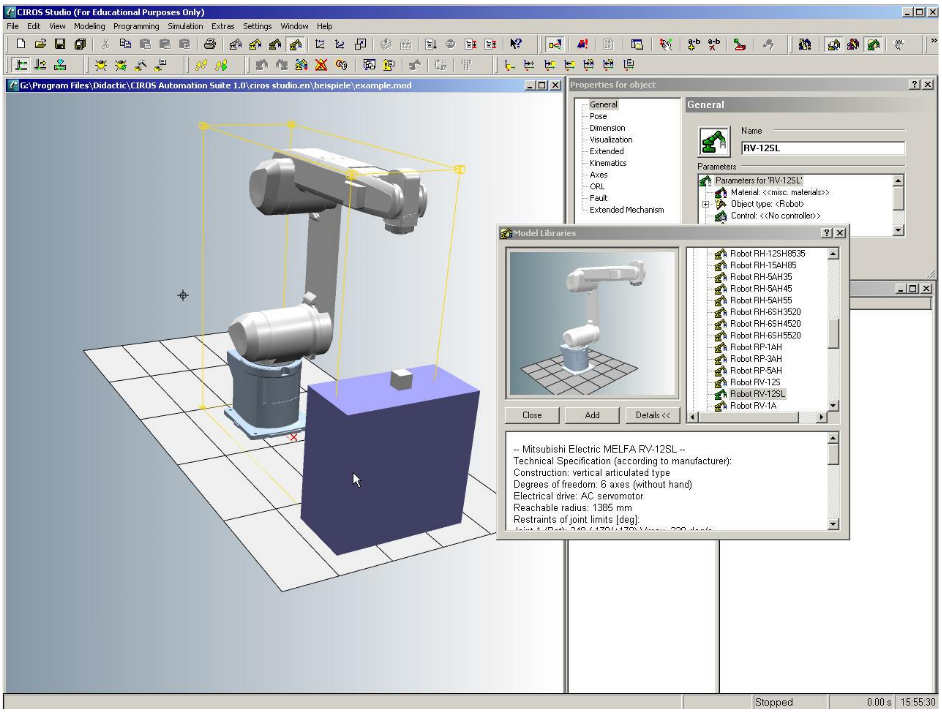
Name: MyBox

Parameters

Parameters for 'MyBox'
Material: <<no material>>
Object type: <Inactive Object>
Control: <<No controller>>
Level of Detail

*Figure 6-8: Grip point Addition*

- Now, the robot is added to the work cell by selecting a Mitsubishi or any other robot under experiment from the library **Mitsubishi Robots**. In order to have better understanding of the example, let's take the robot RV-12SL and add it from Mitsubishi Robots.



General

Name: RV-12SL

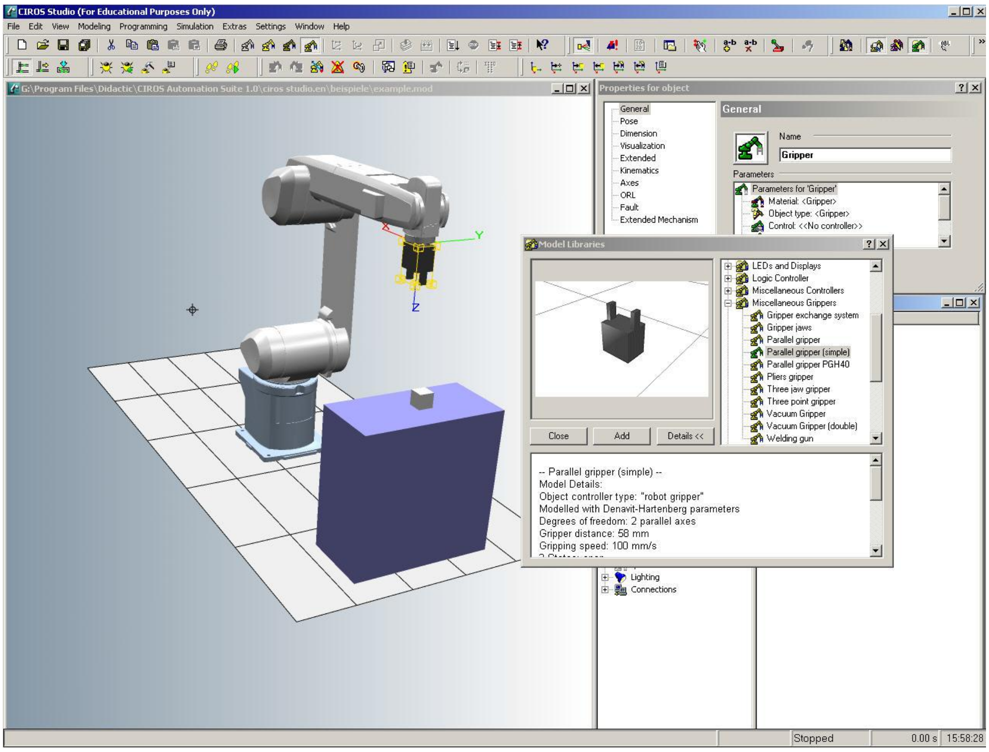
Parameters

Parameters for 'RV-12SL'
Material: <<misc. materials>>
Object type: <Robot>
Control: <<No controller>>

-- Mitsubishi Electric MELFA RV-12SL --
Technical Specification (according to manufacturer):
Construction: vertical articulated type
Degrees of freedom: 6 axes (without hand)
Electrical drive: AC servomotor
Reachable radius: 1385 mm
Restraints of joint limits [deg]:

*Figure 6-9: Robot Addition*

- Parallel Gripper (simple) from Library **Miscellaneous Grippers** in dialog box Model Libraries is added to the work cell. The gripper is then attached to the robot automatically.



General

Name: Gripper

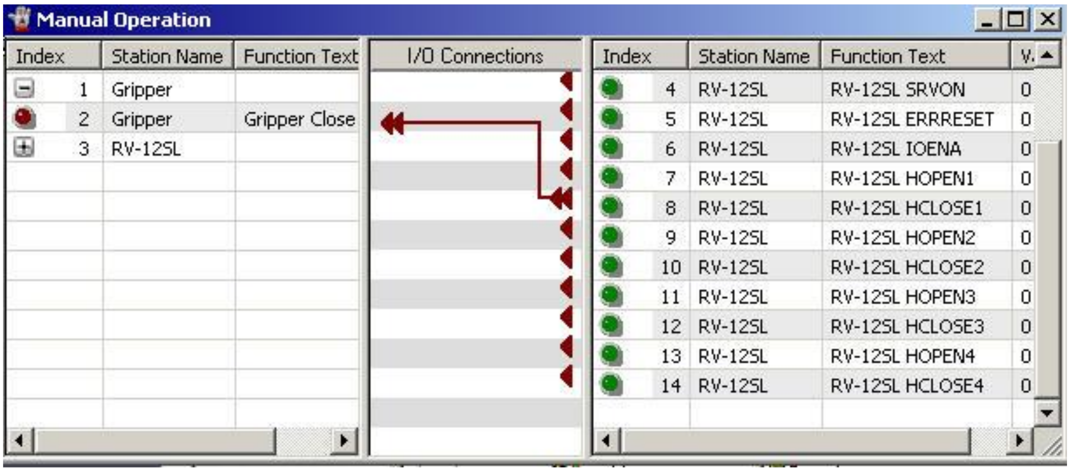
Parameters

Parameters for 'Gripper'
Material: <Gripper>
Object type: <Gripper>
Control: <<No controller>>

-- Parallel gripper (simple) --
Model Details:
Object controller type: "robot gripper"
Modelled with Denavit-Hartenberg parameters
Degrees of freedom: 2 parallel axes
Gripper distance: 58 mm
Gripping speed: 100 mm/s

*Figure 6-10: Gripper Addition*

- To simulate the electrical connection between the robot controller and the gripper, **Manual Operation** window is opened from Modeling command. The window shows the inputs on the left side and outputs on the right. In between the I/O connection will be shown. The arrow next to the output **HCLOSE 1** of the robot in area I/O connections is clicked and a connection line to the input **Gripper Close** of the gripper is drawn. The modeling is completed now, and the work cell is saved.



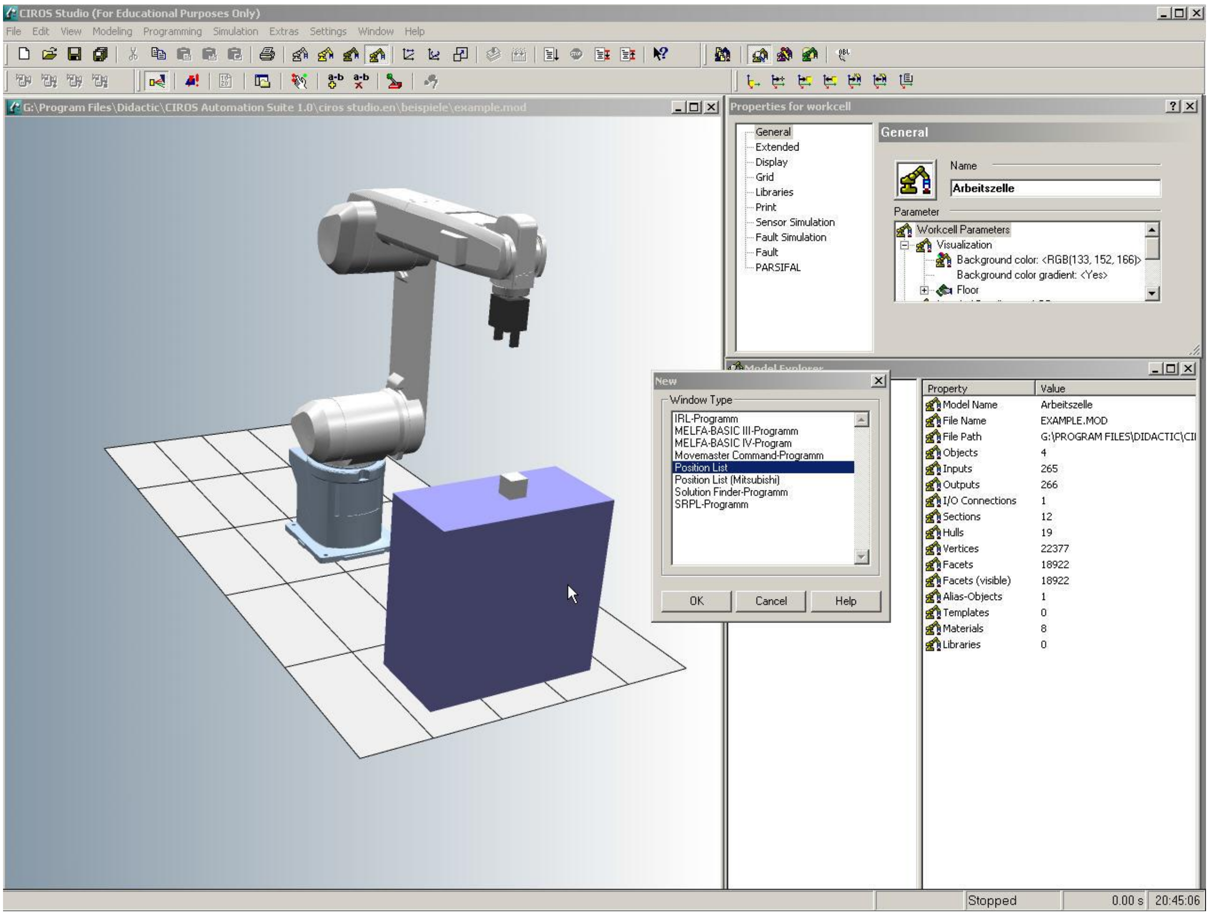
| Index | Station Name | Function Text |
|---|---|---|
| 1 | Gripper | |
| 2 | Gripper | Gripper Close |
| 3 | RV-12SL | |

| Index | Station Name | Function Text | V. |
|---|---|---|---|
| 4 | RV-12SL | RV-12SL SRVON | 0 |
| 5 | RV-12SL | RV-12SL ERRRESET | 0 |
| 6 | RV-12SL | RV-12SL IOENA | 0 |
| 7 | RV-12SL | RV-12SL HOPEN1 | 0 |
| 8 | RV-12SL | RV-12SL HCLOSE1 | 0 |
| 9 | RV-12SL | RV-12SL HOPEN2 | 0 |
| 10 | RV-12SL | RV-12SL HCLOSE2 | 0 |
| 11 | RV-12SL | RV-12SL HOPEN3 | 0 |
| 12 | RV-12SL | RV-12SL HCLOSE3 | 0 |
| 13 | RV-12SL | RV-12SL HOPEN4 | 0 |
| 14 | RV-12SL | RV-12SL HCLOSE4 | 0 |

*Figure 6-11: Electrical connection of Gripper*

## 6.3. Programming of the Model

To program the model, we developed in previous paragraph, a new position list is created first by selecting **New** from **File** menu and selecting Position List.



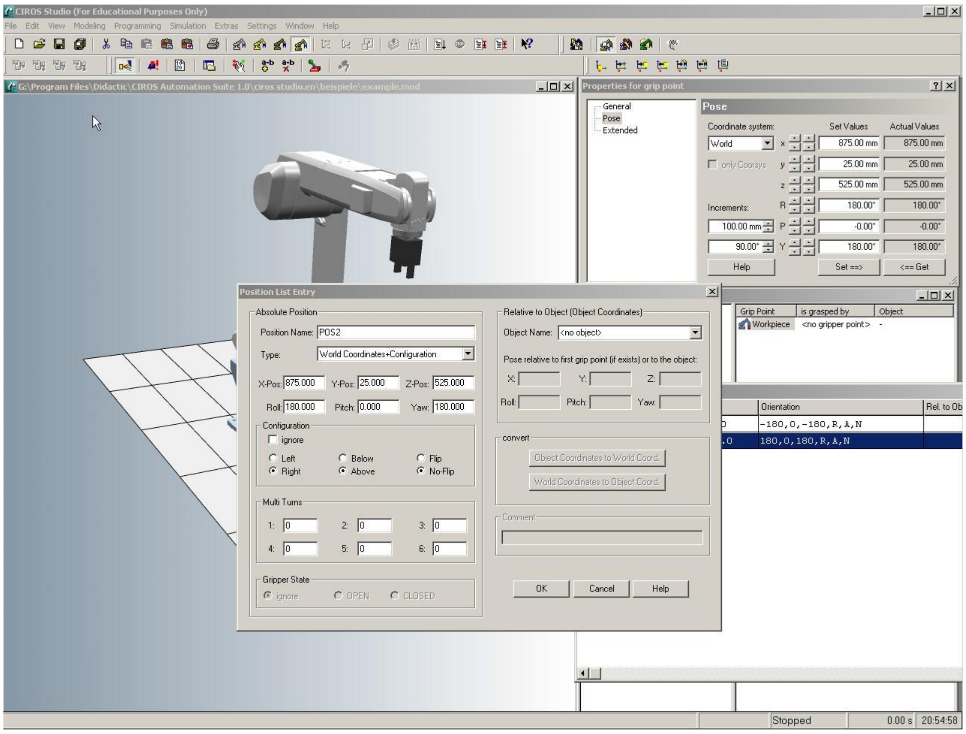
Name: Arbeitszelle

Workcell Parameters
  Visualization
    Background color: <RGB(133, 152, 166)>
    Background color gradient: <Yes>
  Floor

| Property | Value |
|---|---|
| Model Name | Arbeitszelle |
| File Name | EXAMPLE.MOD |
| File Path | G:\PROGRAM FILES\DIDACTIC\CII |
| Objects | 4 |
| Inputs | 265 |
| Outputs | 266 |
| I/O Connections | 1 |
| Sections | 12 |
| Hulls | 19 |
| Vertices | 22377 |
| Facets | 18922 |
| Facets (visible) | 18922 |
| Alias-Objects | 1 |
| Templates | 0 |
| Materials | 8 |
| Libraries | 0 |

*Figure 6-12: Position List Creation*

- Initial position of the robot is accepted as first position (POS1) of the position List by pressing **SHIFT + F2**. Grip point of the object MyBox is selected in Model Explorer and from the property **Pose**, World coordinate system is selected as reference coordinate system. Position is selected by clicking on the entry with left mouse button and Position List Entry dialog is opened. Position data of the grip point is transferred to POS2 and dialog box is closed by clicking OK.


CIROS Studio (For Educational Purposes Only)
File Edit View Modeling Programming Simulation Extras Settings Window Help
Properties for grip point
General
Pose
Extended
Help
Set ==>
<== Get
Position List Entry
OK
Cancel
Help
Object Coordinates to World Coord.
World Coordinates to Object Coord.
Stopped

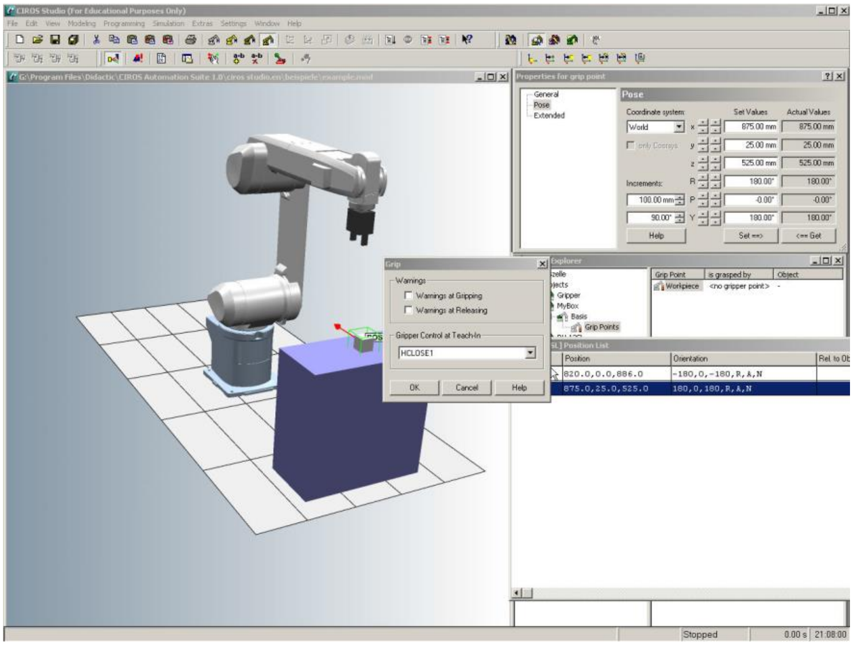
Position List Entry

Absolute Position
Position Name: POS2
Type: World Coordinates+Configuration
X-Pos: 875.000 Y-Pos: 25.000 Z-Pos: 525.000
Roll: 180.000 Pitch: 0.000 Yaw: 180.000
Configuration
☐ ignore
○ Left ○ Below ○ Flip
● Right ● Above ● No-Flip
Multi Turns: 1: 0 2: 0 3: 0 4: 0 5: 0 6: 0
Gripper State: ● ignore ○ OPEN ○ CLOSED

Relative to Object (Object Coordinates)
Object Name: <no object>

Figure 6-13: Position Data Transfer

- In the menu Settings – Grip is used to configure the grip control via Teach-In (F8) window. Close Hand and Open Hand are pressed in Teach-In window in order to close and open the gripper of the robot respectively. If there is any warning of no object near the gripper, it is confirmed, or these can also be switched off.

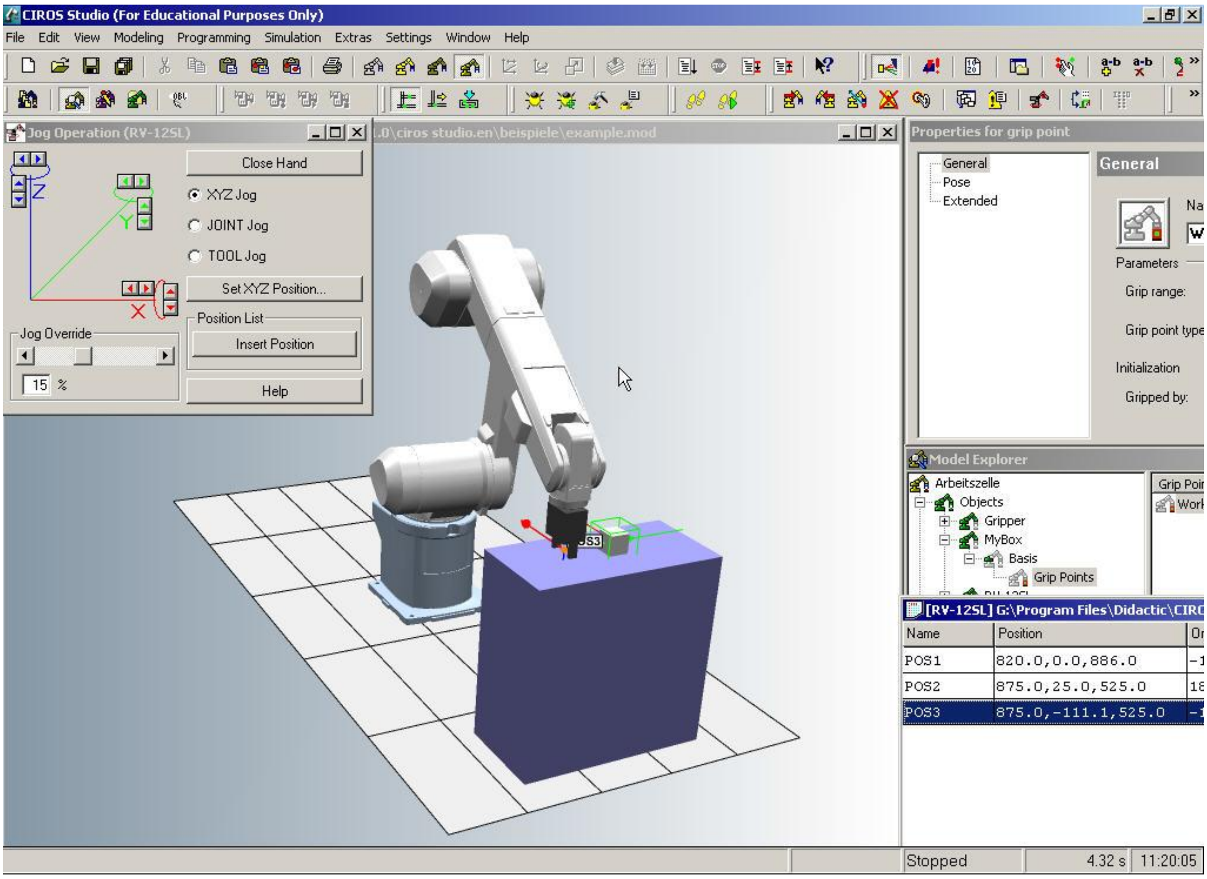


Grip

Warnings: ☐ Warnings at Gripping ☐ Warnings at Releasing
Gripper Control at Teach-In: HCLOSE1

Figure 6-14: Hand Opening and Closing

- The robot is then moved to POS2 by double clicking on the position entry of position list. Modus XYZ-Jog (world coordinates) in the teach-In window is selected and by clicking on the respective button, the robot is moved towards negative Y-direction. A new position is inserted as POS3 in the position list. This position list is then saved as Example.psl. now, from menu **Simulation**, work cell is reset by clicking on **Reset Work Cell**.



● XYZ Jog

Jog Override: 15 %

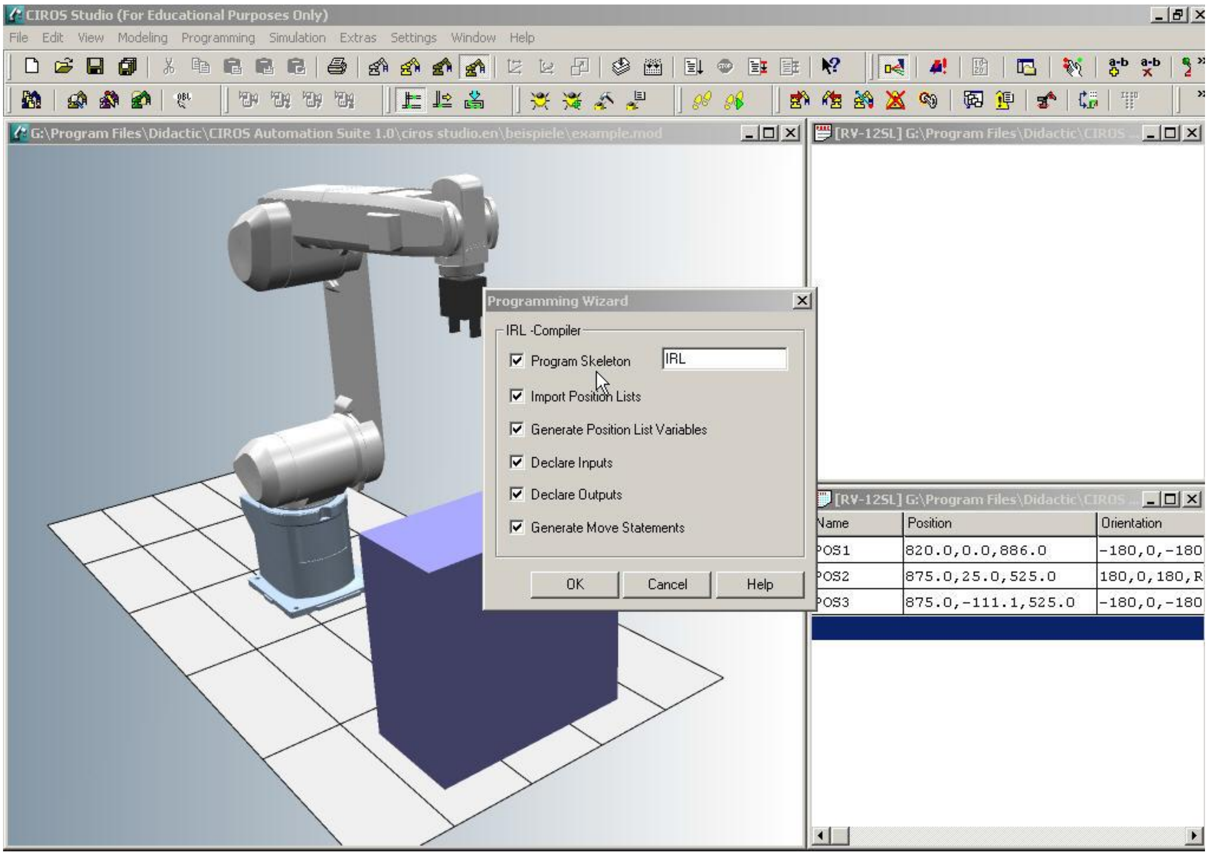
[RV-12SL] G:\Program Files\Didactic\CIRO

| Name | Position | Or |
|---|---|---|
| POS1 | 820.0,0.0,886.0 | -1 |
| POS2 | 875.0,25.0,525.0 | 18 |
| POS3 | 875.0,-111.1,525.0 | -1 |

Stopped | 4.32 s | 11:20:05

*Figure 6-15: Moving the Robot to Next Position*

- Now, in order to program the modeled system, a new program file is generated. It could be in any language from MELFA Basic IV, V or it could be .IRL file. Let's take an example of .IRL file here. From menu command **File** – **New**, a new IRL program is generated. Program is saved as Example.IRL. Program Wizard is used to generate a simple body of the program by activating programming window from **Programming** – **Programming Wizard**. All options are selected and OK is pressed.


CIROS Studio (For Educational Purposes Only)
File Edit View Modeling Programming Simulation Extras Settings Window Help
G:\Program Files\Didactic\CIROS Automation Suite 1.0\ciros studio.en\beispiele\example.mod
[RV-12SL] G:\Program Files\Didactic\CIROS ...
Programming Wizard
IRL -Compiler
OK
Cancel
Help

☑ Program Skeleton IRL
☑ Import Position Lists
☑ Generate Position List Variables
☑ Declare Inputs
☑ Declare Outputs
☑ Generate Move Statements

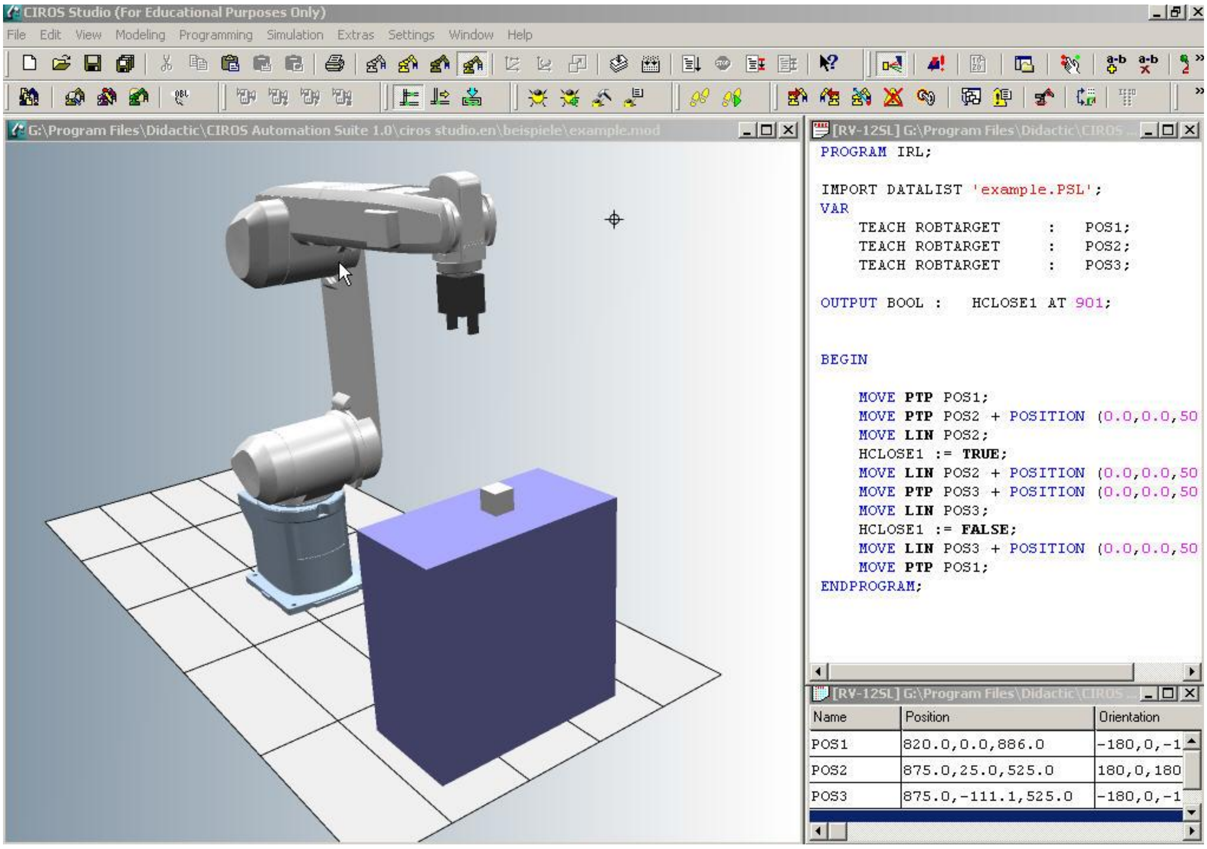
| Name | Position | Orientation |
|---|---|---|
| POS1 | 820.0,0.0,886.0 | -180,0,-180 |
| POS2 | 875.0,25.0,525.0 | 180,0,180,R |
| POS3 | 875.0,-111.1,525.0 | -180,0,-180 |

Figure 6-16: .IRL Program Creation

- The program is then changed to a simple pick and place task where the safety positions are 50 mm above picking and placing position. All the inputs and outputs are then deleted except the output **HCLOSE 1**.
- Sample Program:

```
PROGRAM IRL;
IMPORT DATALIST 'Example.PSL';
VAR
TEACH ROBTARGET : POS1;
TEACH ROBTARGET : POS2;
TEACH ROBTARGET : POS3;
OUTPUT BOOL : Grasp AT 0;
BEGIN
MOVE PTP POS1;
MOVE PTP POS2 + POSITION (0.0,0.0,50.0);
MOVE LIN POS2;
GRASP := TRUE;
MOVE LIN POS2 + POSITION (0.0,0.0,50.0);
MOVE PTP POS3 + POSITION (0.0,0.0,50.0);
MOVE LIN POS3;
```

```
GRASP := FALSE;
MOVE LIN POS3 + POSITION (0.0,0.0,50.0);
MOVE PTP POS1;
ENDPROGRAM;
```


CIROS Studio (For Educational Purposes Only)
File
Edit
View
Modeling
Programming
Simulation
Extras
Settings
Window
Help
G:\Program Files\Didactic\CIROS Automation Suite 1.0\ciros studio.en\beispiele\example.mod
[RV-12SL] G:\Program Files\Didactic\CIROS ...
[RV-12SL] G:\Program Files\Didactic\CIROS ...

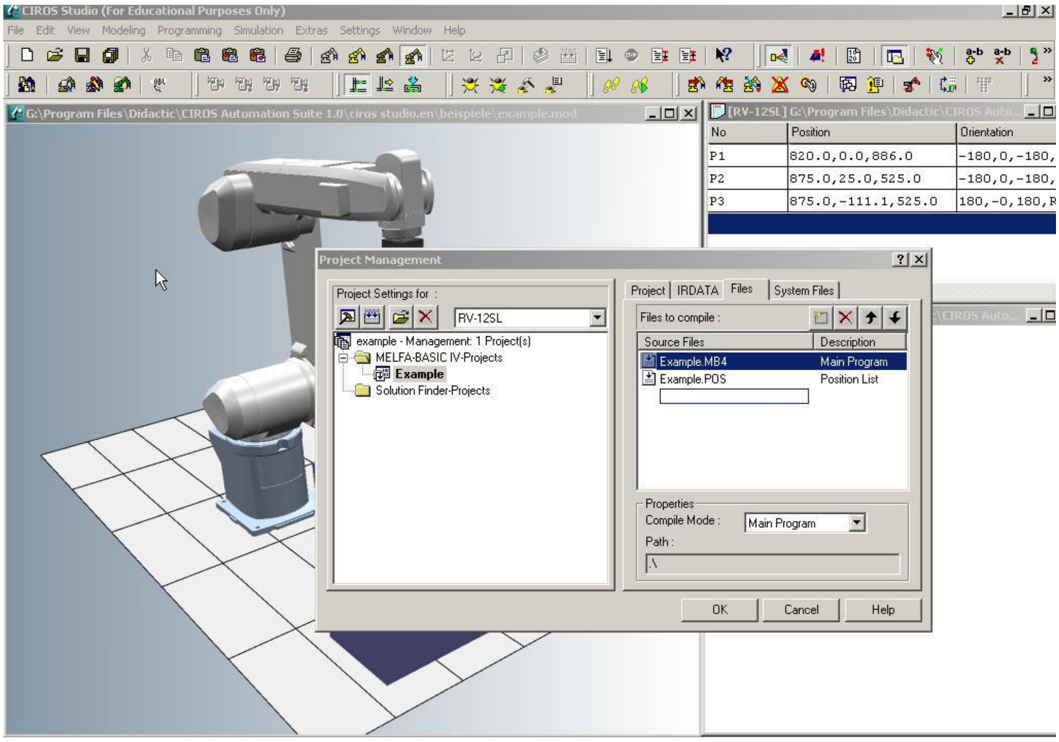
```
PROGRAM IRL;

IMPORT DATALIST 'example.PSL';
VAR
    TEACH ROBTARGET    :   POS1;
    TEACH ROBTARGET    :   POS2;
    TEACH ROBTARGET    :   POS3;

OUTPUT BOOL :   HCLOSE1 AT 901;


BEGIN

    MOVE PTP POS1;
    MOVE PTP POS2 + POSITION (0.0,0.0,50
    MOVE LIN POS2;
    HCLOSE1 := TRUE;
    MOVE LIN POS2 + POSITION (0.0,0.0,50
    MOVE PTP POS3 + POSITION (0.0,0.0,50
    MOVE LIN POS3;
    HCLOSE1 := FALSE;
    MOVE LIN POS3 + POSITION (0.0,0.0,50
    MOVE PTP POS1;
ENDPROGRAM;
```

| Name | Position | Orientation |
|---|---|---|
| POS1 | 820.0,0.0,886.0 | -180,0,-1 |
| POS2 | 875.0,25.0,525.0 | 180,0,180 |
| POS3 | 875.0,-111.1,525.0 | -180,0,-1 |

*Figure 6-17: Program and Simulation Window*

- Program is saved then and from the menu **Programming**, the code is compiled by selecting **Compile+Link**. The syntax will be checked and IRDATA code will be generated and downloaded in the virtual robot controller. IRDATA is a standardized low-level code needed to download robot programs created in different programming languages [17]. CIROS processes IRDATA code created by compilers for different high-level programming languages.
- Similarly, MELFA Basic IV/V program can be generated too. For that purpose, a new MRL-position list is opened and MELFA Basic IV/V code is generated. The new files are named as Example.pos and Example.mb4 (Example.mb5) respectively. For simulation purposes, the new program and position list is integrated into MELFA Basic IV/V project. From **Project Management** dialog in Execute menu, **Add Project** is chosen to add this new project and named as Mitsubihsi.prj. On page **Files** of dialog box, the program and position list are then added to the project by selecting

those files. **Mitsubishi.mb4** is declared as main program then. The program is the integrated into current project.


CIROS Studio (For Educational Purposes Only)
File Edit View Modeling Programming Simulation Extras Settings Window Help
G:\Program Files\Didactic\CIROS Automation Suite 1.0\ciros studio.en\beispiele\example.mod
[RV-12SL] G:\Program Files\Didactic\CIROS Auto...
Project Management
Project Settings for :
RV-12SL
example - Management: 1 Project(s)
MELFA-BASIC IV-Projects
Example
Solution Finder-Projects
Project
IRDATA
Files
System Files
Files to compile :
Properties
OK
Cancel
Help

| No | Position | Orientation |
|---|---|---|
| P1 | 820.0,0.0,886.0 | -180,0,-180, |
| P2 | 875.0,25.0,525.0 | -180,0,-180, |
| P3 | 875.0,-111.1,525.0 | 180,-0,180,R |

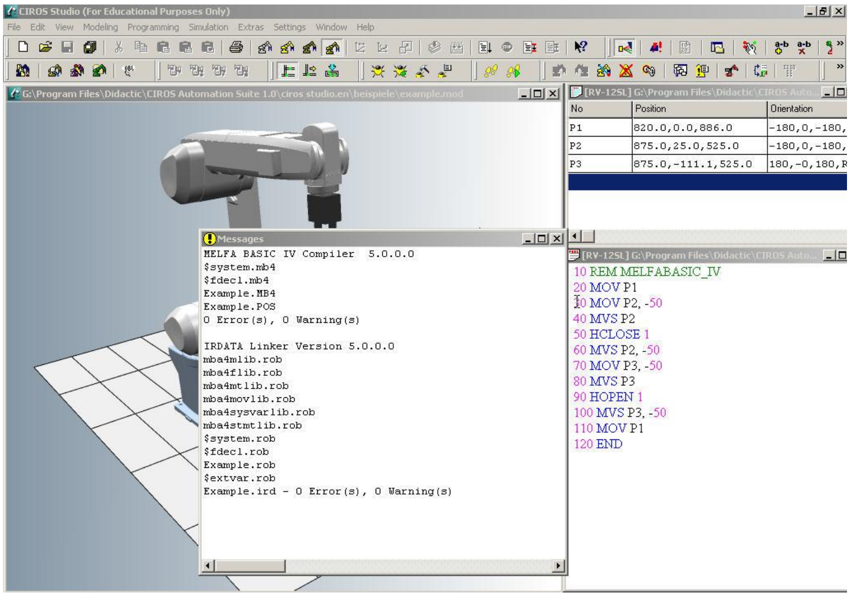
| Source Files | Description |
|---|---|
| Example.MB4 | Main Program |
| Example.POS | Position List |

Compile Mode : Main Program

Path : .\

Figure 6-18: MELFA Basic IV Program Creation

- From **Programming** menu, click on **Compile+Link** after activating programming window by clicking on it. The syntax is then checked and IRDATA code is generated. An executable robot program for the work cell is then developed


CIROS Studio (For Educational Purposes Only)
File Edit View Modeling Programming Simulation Extras Settings Window Help
G:\Program Files\Didactic\CIROS Automation Suite 1.0\ciros studio.en\beispiele\example.mod
[RV-12SL] G:\Program Files\Didactic\CIROS Auto...
Messages

| No | Position | Orientation |
|---|---|---|
| P1 | 820.0,0.0,886.0 | -180,0,-180, |
| P2 | 875.0,25.0,525.0 | -180,0,-180, |
| P3 | 875.0,-111.1,525.0 | 180,-0,180,R |

```
MELFA BASIC IV Compiler  5.0.0.0
$system.mb4
$fdecl.mb4
Example.MB4
Example.POS
0 Error(s), 0 Warning(s)

IRDATA Linker Version 5.0.0.0
mba4mlib.rob
mba4flib.rob
mba4mtlib.rob
mba4movlib.rob
mba4sysvarlib.rob
mba4stmtlib.rob
$system.rob
$fdecl.rob
Example.rob
$extvar.rob
Example.ird - 0 Error(s), 0 Warning(s)
```

```
10 REM MELFABASIC_IV
20 MOV P1
30 MOV P2, -50
40 MVS P2
50 HCLOSE 1
60 MVS P2, -50
70 MOV P3, -50
80 MVS P3
90 HOPEN 1
100 MVS P3, -50
110 MOV P1
120 END
```

Figure 6-19: Compilation of MELFA Basic IV Code

- To start the simulation, from **Simulation** window, **Start** is pressed. The program is simulated step by step and simulation time is displayed in the status bar. Currently executed command is highlighted in the program window. Before starting the simulation second time, the work cell needs to be reset from Simulation window by clicking on Reset Work Cell. This command resets all the objects as well as the robot.

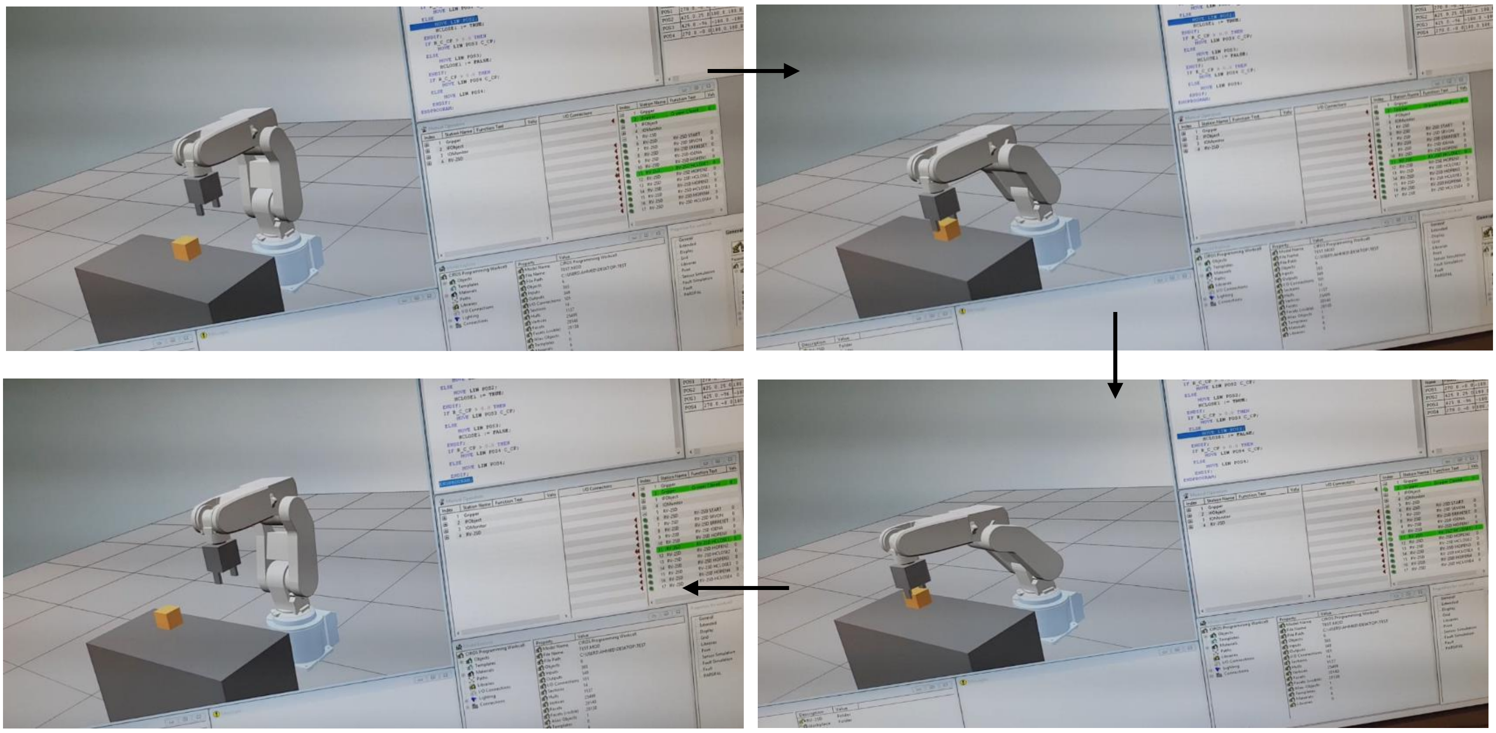


*Figure 6-20: Simulation of the Code*

## 6.4. Modifications in the Actual System

The RFID tags that we have in DigiLab4U are placed on some substrates or material which are in a form of tiny plates. So, for robot to pick those tags up and place them in chamber, some system parameters are improvised, and a new feasible solution is adapted. Some holders were designed by using the same material as the interior of RFID Chamber in order to avoid noise or hinderance in the experiment. A base of the same material is also designed for each tag so that there would not be any error in placement of tag. Such necessary adaptations enable us to have better precision in each task of the robot and allows the correct placement after each operation. There is no need to adjust the tags manually because the robot moves precise enough to drop the tag at exact position from where it picked the tag.

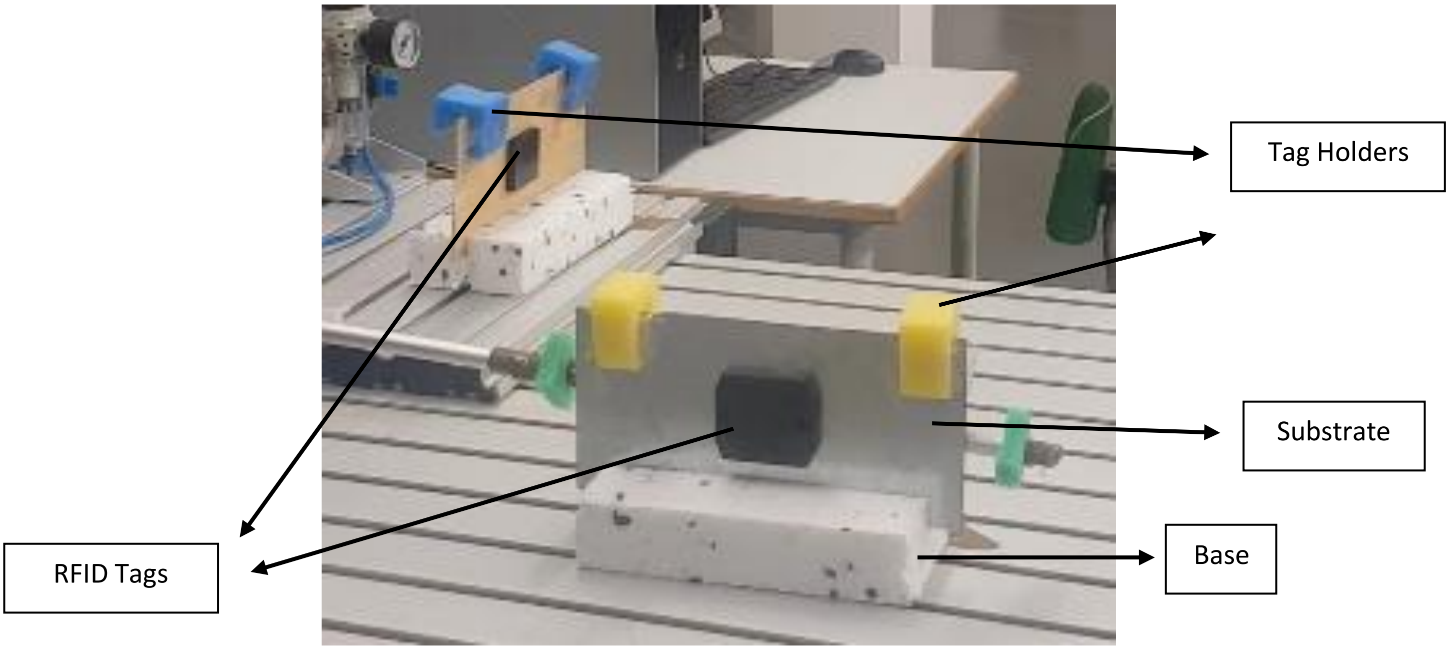


*Figure 6-21: RFID Tags Improvement*

A tool is also mounted on the robot so that it could easily pick up the tag and access the right place inside the chamber. A stick is mounted in a combination with the gripper to increase length and to pick the tag up.

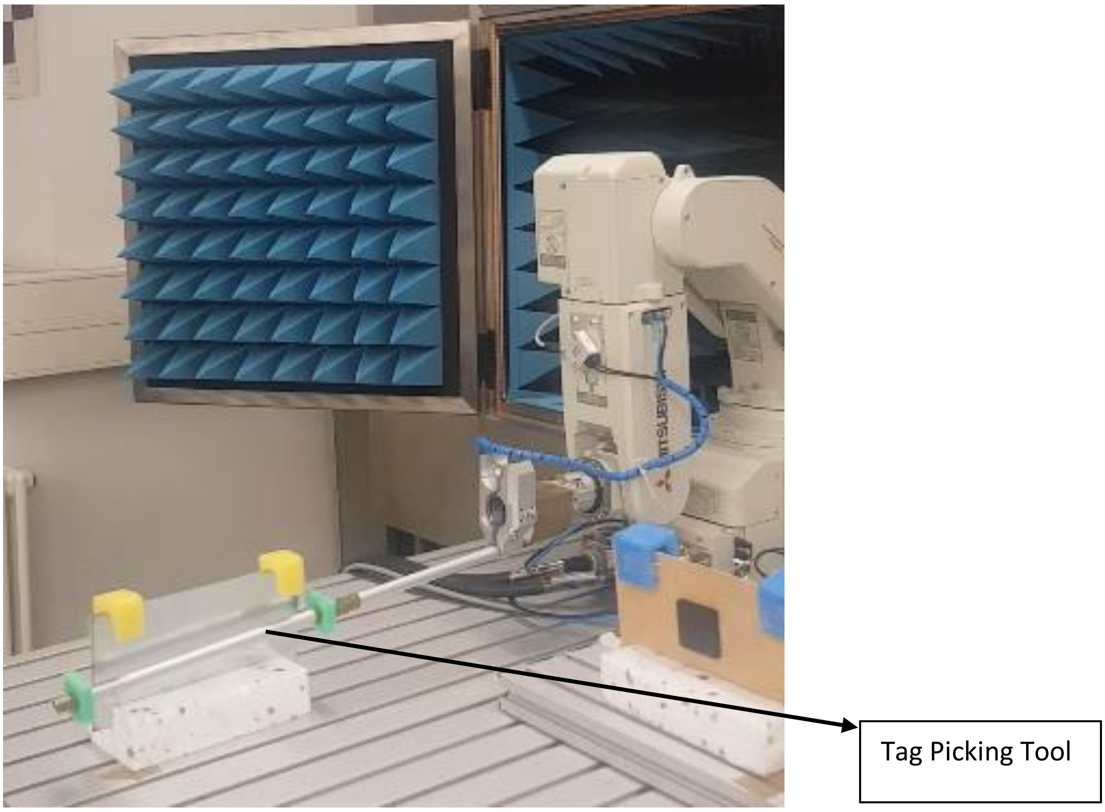


*Figure 6-22: Tag Picking Mechanism*

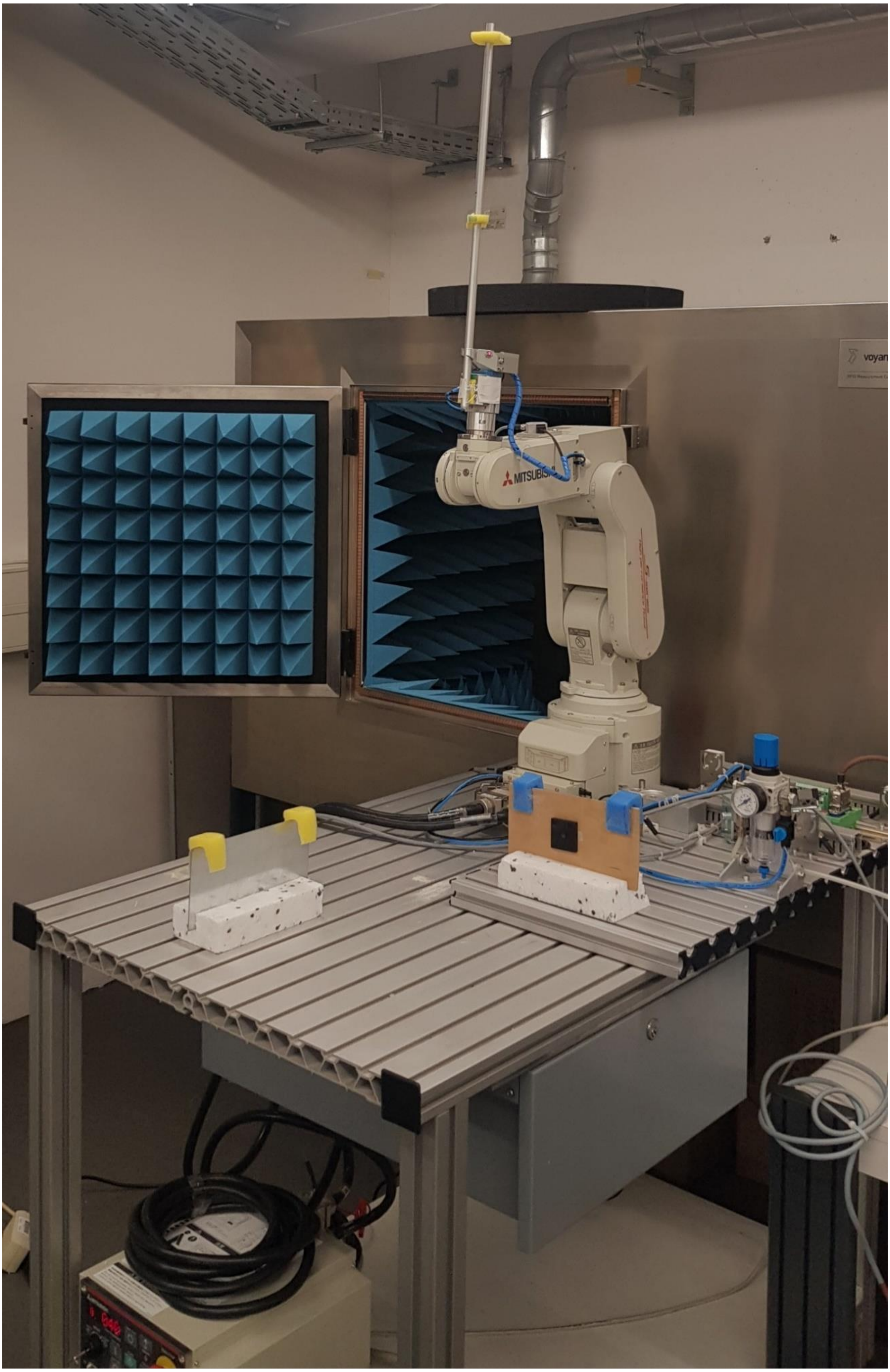
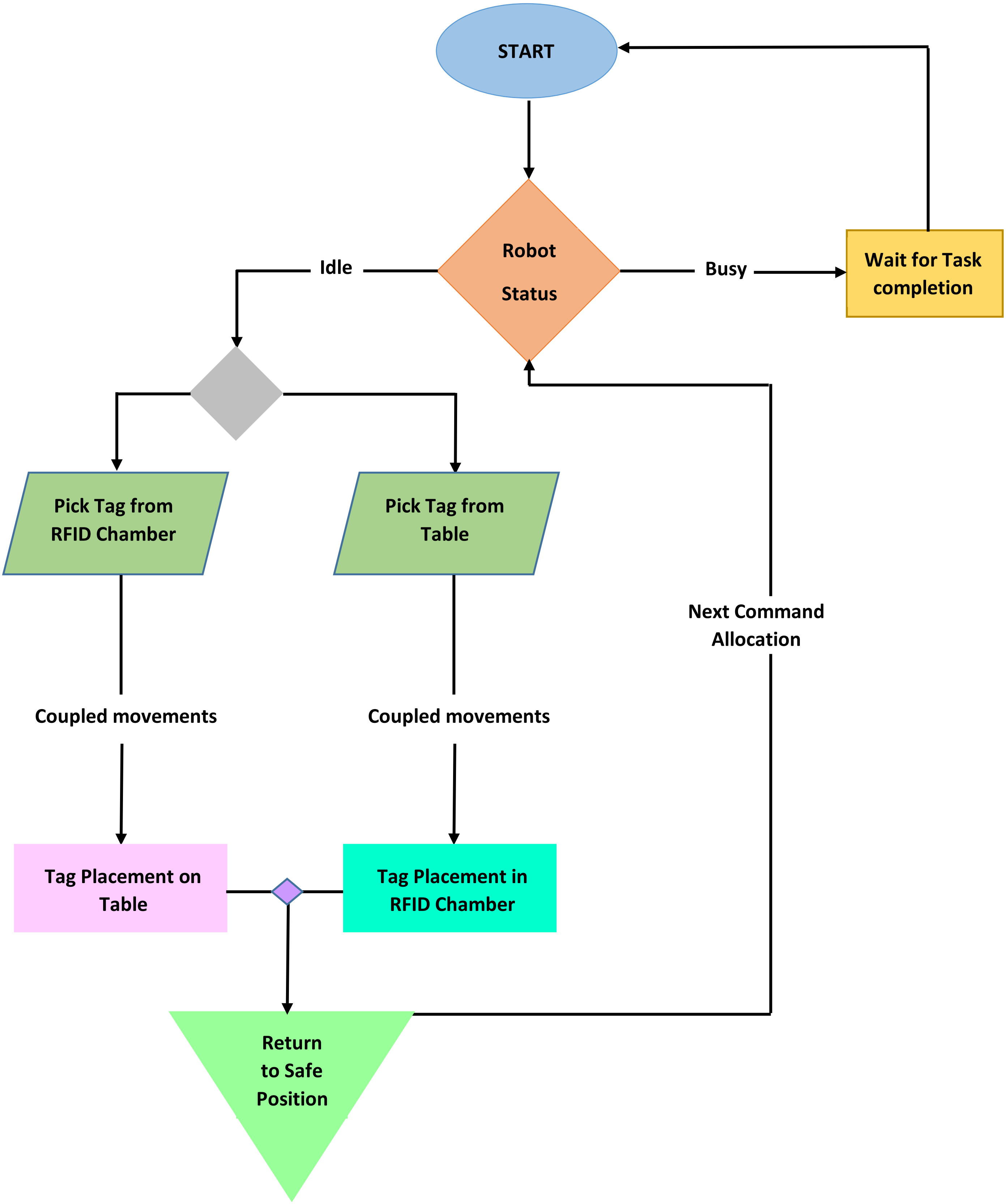


*Figure 6-23: Mitsubishi RV-2SDB, RFID Tags and RFID Measurement Cabinet*

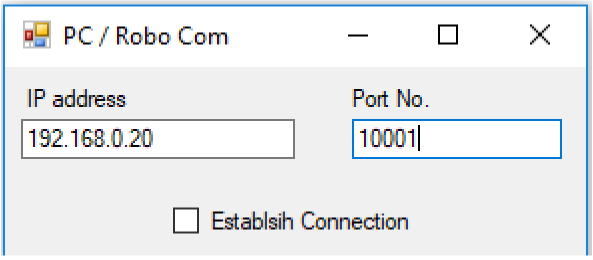
## 6.5. Experimental Analysis

In this section the complete analysis of the experiment is shown through the help of Flowchart and pictures from the experiment.



Once the code is written on CIROS Studio, it is compiled and downloaded into the robot controller. Then the following steps are performed:

- Robo / PC Com application is opened, and Connection is established between controller and PC.



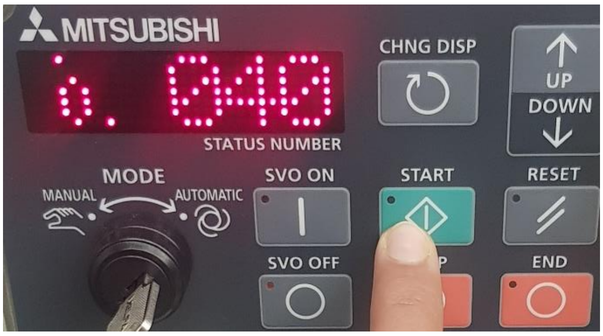
PC / Robo Com

IP address: 192.168.0.20

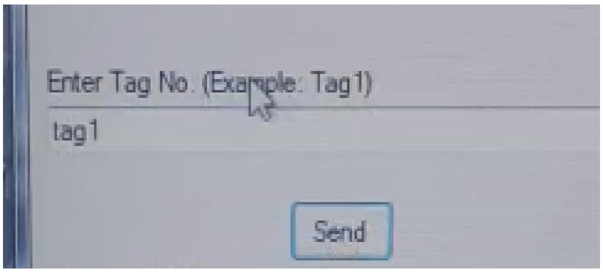
Port No.: 10001

☐ Establsih Connection

- After the connection, program in the robot controller is started by pressing "Start" button on the controller.



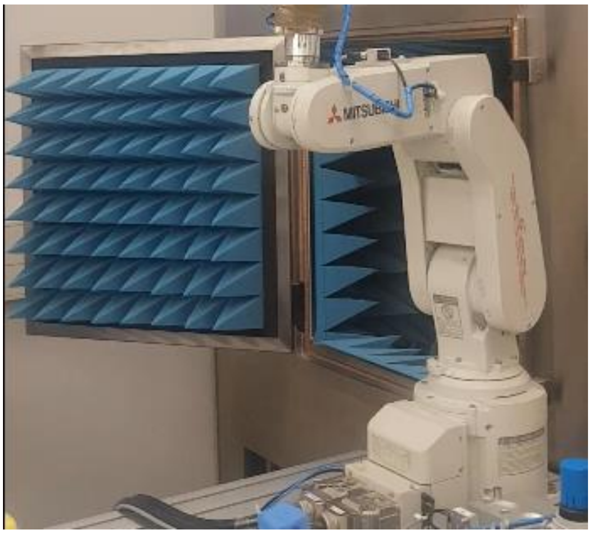
- Then a command is sent from the Robo / PC Com application. It could be a command of picking any tag from the table and placing that into chamber. The syntax of command is like "tag1", "tag2" etc.

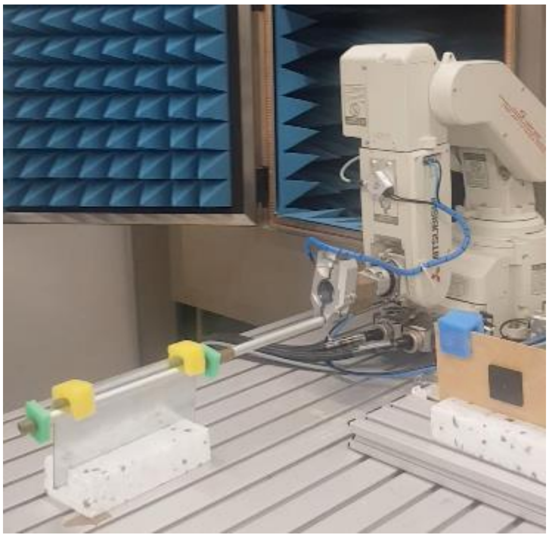


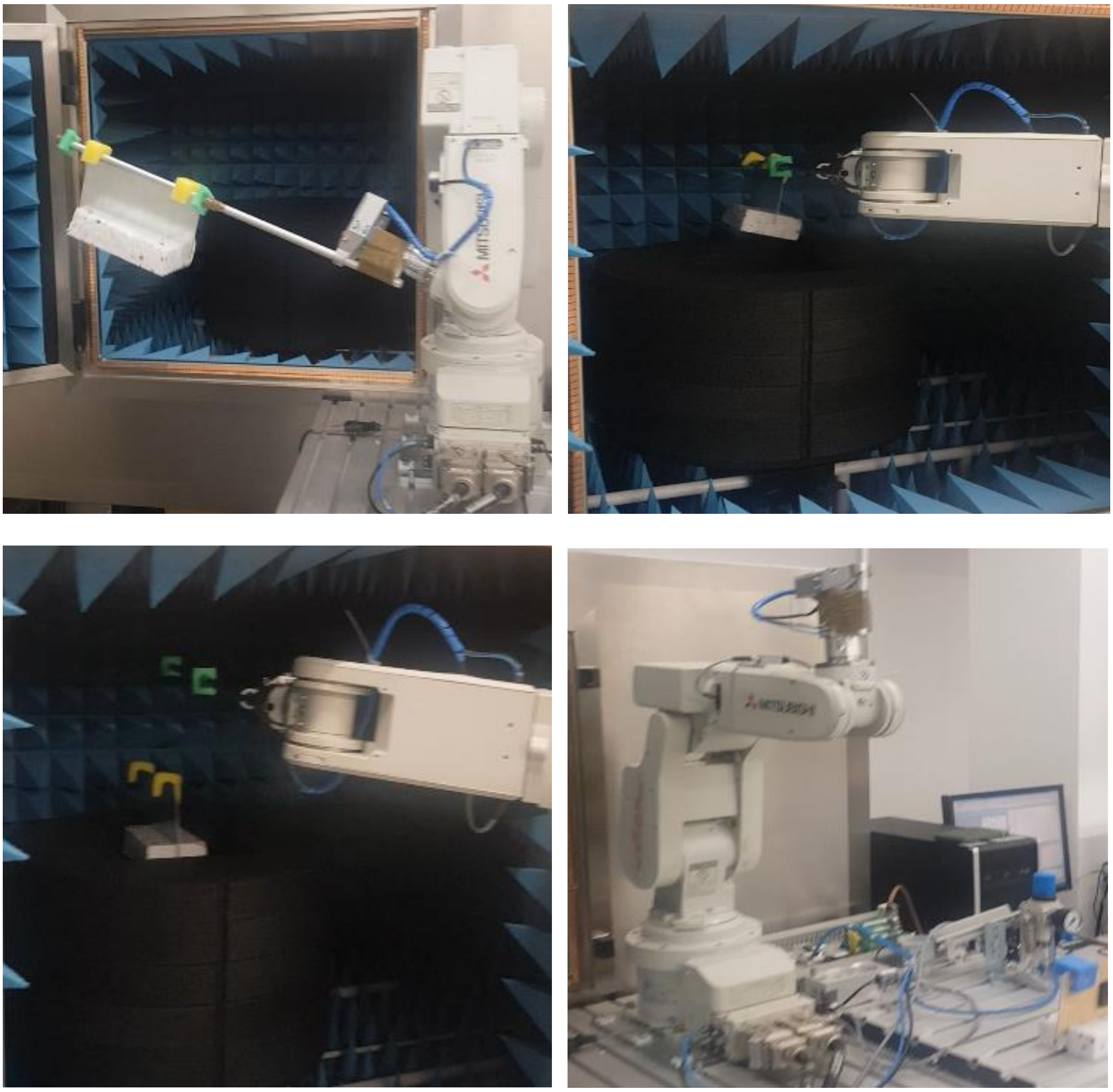
Enter Tag No. (Example: Tag1)

tag1

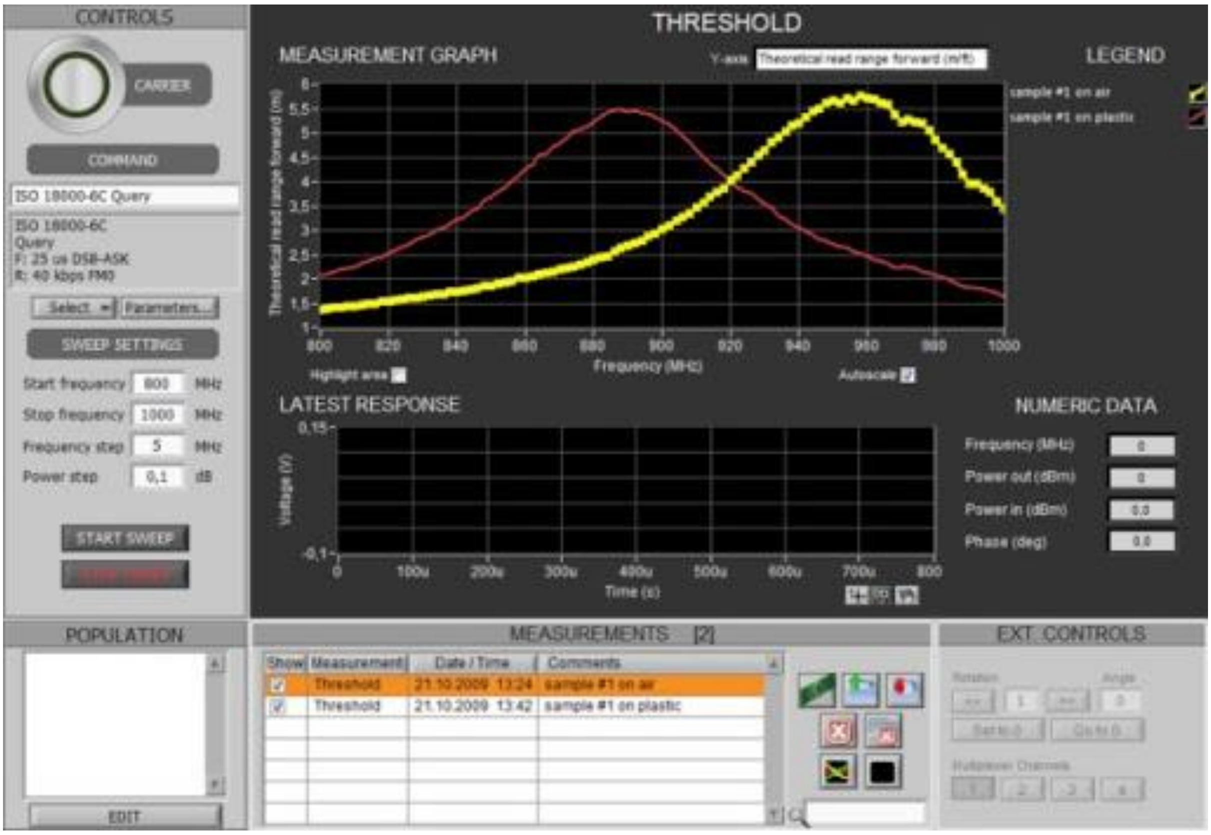
- After the command is entered and sent, the robot starts moving from its safe position to pick up the tag and place it into the chamber.







- After placing the tag, the robot returns to its original safe position as shown above. Now the next stage of experiment begins. The experiment on tag is performed eventually by using Tagformance software and the data is analyzed. There are quite versatile options to analyze different types of data on Tagformance. There is also an option to save the graphs and the obtained graphical data as an excel file consisted of numerical values so that it is easy to analyze in future.



ISO 18000-6C Query

ISO 18000-6C
Query
F: 25 us DSB-ASK
R: 40 kbps FM0

Start frequency 800 MHz
Stop frequency 1000 MHz
Frequency step 5 MHz
Power step 0,1 dB

Y-axis Theoretical read range forward (m/ft)

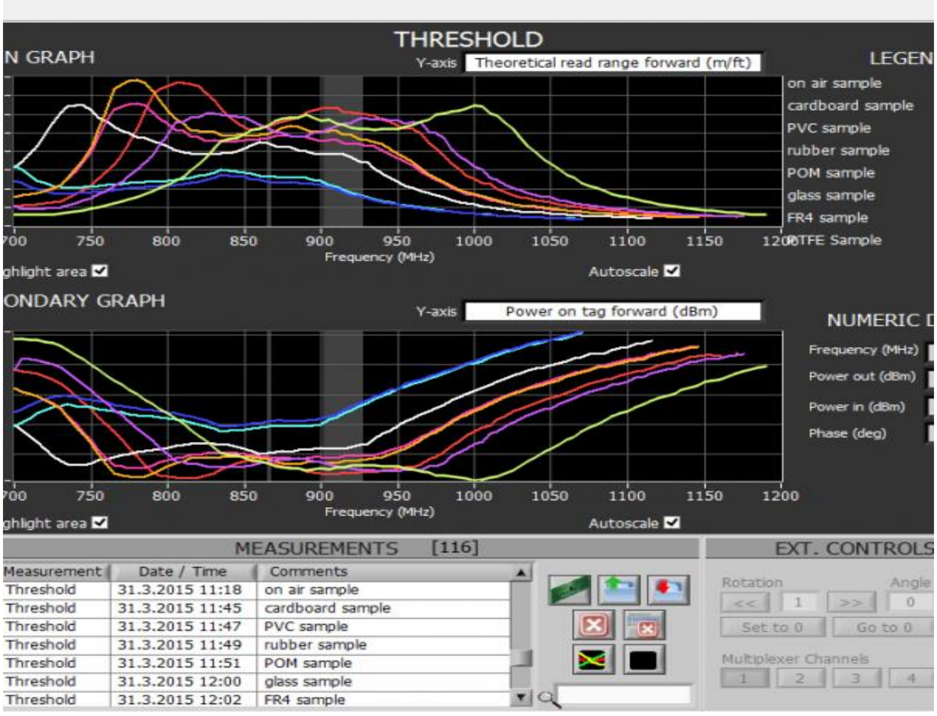
sample #1 on air
sample #1 on plastic

Frequency (MHz) 0
Power out (dBm) 0
Power in (dBm) 0.0
Phase (deg) 0.0

| Show | Measurement | Date / Time | Comments |
|---|---|---|---|
| ☑ | Threshold | 21.10.2009 13:24 | sample #1 on air |
| ☑ | Threshold | 21.10.2009 13:42 | sample #1 on plastic |

Figure 6-24: Tagformance Software Analysis

- ➢ Among other attributes, the Threshold graphs of different materials (Rubber, PVC, Cardboard or wood etc.) are shown below.



Y-axis Theoretical read range forward (m/ft)

on air sample
cardboard sample
PVC sample
rubber sample
POM sample
glass sample
FR4 sample
PTFE Sample

Y-axis Power on tag forward (dBm)

Frequency (MHz)
Power out (dBm)
Power in (dBm)
Phase (deg)

| Measurement | Date / Time | Comments |
|---|---|---|
| Threshold | 31.3.2015 11:18 | on air sample |
| Threshold | 31.3.2015 11:45 | cardboard sample |
| Threshold | 31.3.2015 11:47 | PVC sample |
| Threshold | 31.3.2015 11:49 | rubber sample |
| Threshold | 31.3.2015 11:51 | POM sample |
| Threshold | 31.3.2015 12:00 | glass sample |
| Threshold | 31.3.2015 12:02 | FR4 sample |

Figure 6-25: Threshold Graphs of Different Materials

- Similarly, Orientation Sensitivity of a tag can also be checked. The detailed analysis on Tagformance software is not done or provided here because it was out of the scope of thesis work. Only a reference examples are given here.

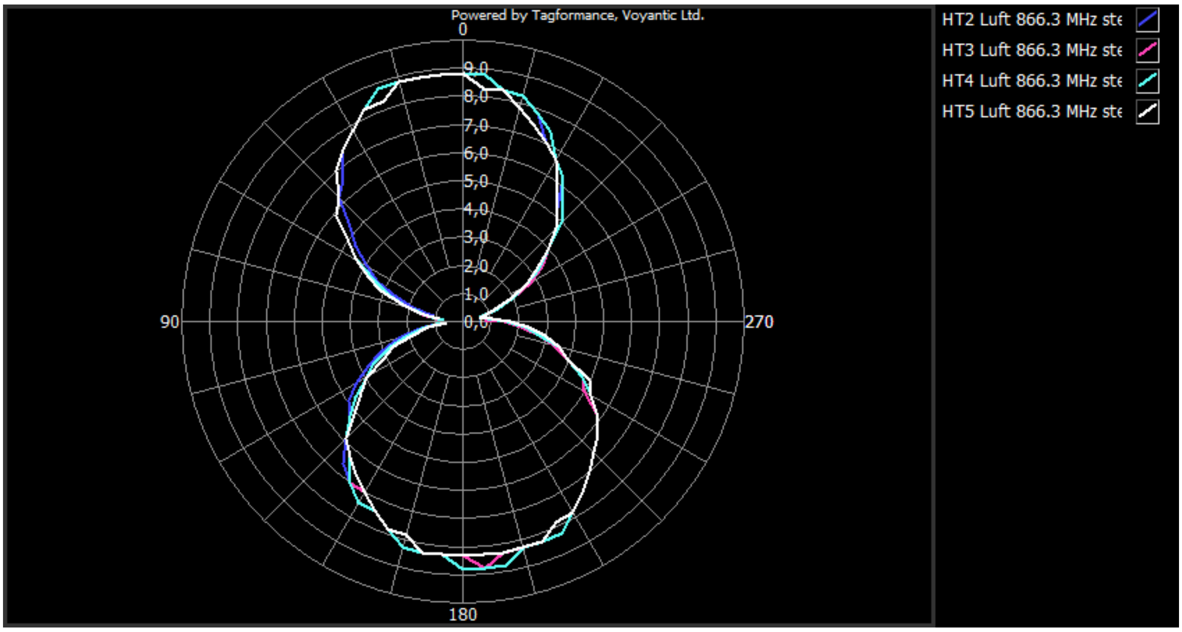
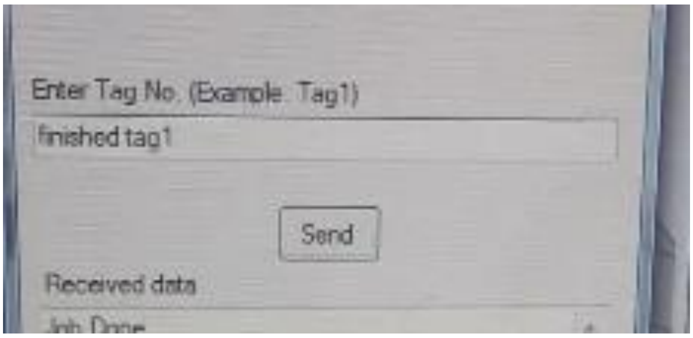


Figure 6-26: Orientation Sensitivity Graph

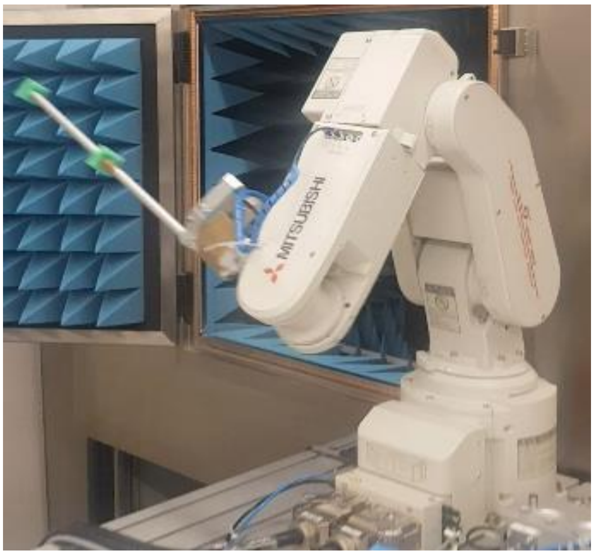
- After the completion of experiment and storage of graphical data obtained from the tag, another command like "finished tag1" is sent to the robot. The robot then goes into the chamber, picks up the tag and put it on the exact position from where it had picked it up.

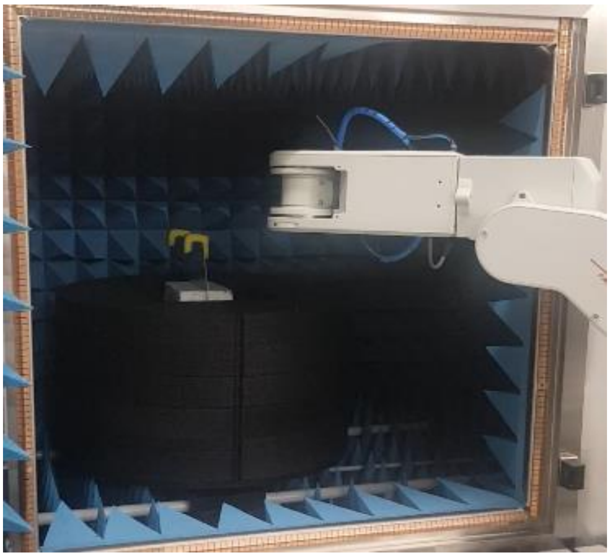

Enter Tag No. (Example: Tag1)

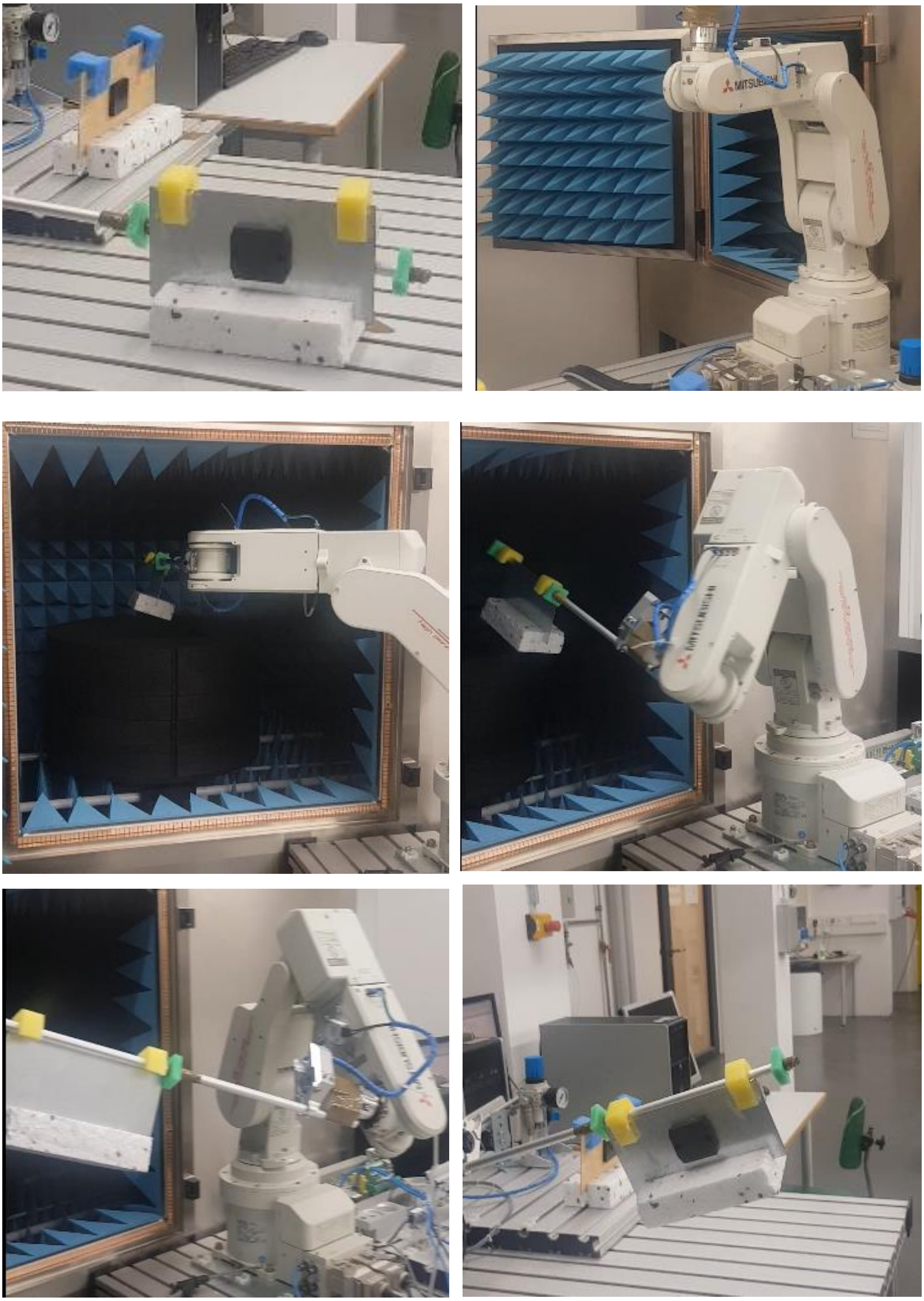
finished tag1

Received data

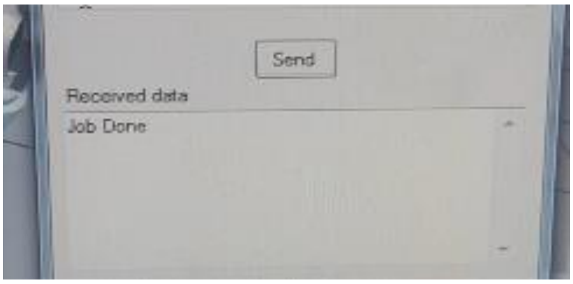
Job Done







- At the end it returns to safe position and send a “Job Done” message. Which means that it is now ready for the next task/job. Now, a user can again send the command for the same tag or it could be any other tag. And the robot repeats the series of movements as earlier and is always ready for the next job. This whole experiment does not take much time. The time dependency is only related to the Tagformance experiment part.



Job Done

## 6.6. Possible Errors or Risks

The sensitive and costly lab equipment requires utter safety and care, but risks and errors are always a part of a project. Some categories of errors that were faced or might happen are shown as followed:

- Position Data Loss
- Battery Power Loss
- Variables, I/O, Com Link Errors
- Safety Risks

Position Data of robotic arm could be lost because as explained earlier, the batteries at the back of motor cover which store the real time data of robotic arm may dry over time. Similarly, the battery of robotic controller which stores the programs etc. can also lose power with the passage of time. Then there are some errors related to declaration of correct variables, definitions of correct I/O’s or unsuccessful connection of Robot and Com application. Considering all these errors, the risk of safety cannot be ignored. A sincere precaution is required while working with laboratory equipment.

# 7. Future Work and Conclusion

## 7.1. Conclusion

The focus of this thesis was to integrate different laboratories structures together in such a way that the experiments or projects could be accessed remotely. Among the projects, this thesis specifically addressed the automation of an industrial robot (Mitsubishi MELFA RV-2SDB) with RFID measurement cabinet. The study focused on developing control schemes through which the robot could be controlled remotely by anticipating TCP/IP configuration. For that purpose, an application was designed using socket programming by which it was quite handy to send online commands and control the robot remotely. In short, to present a web-centered, cross-institutional and research-oriented remote laboratory (DigiLab4U) comprised of an automated industrial robot and necessary hardware/software infrastructure required to make available for incoming students or public internet usage.

Based on experimental analysis of the work related to MELFA RV-2SDB robot, it can be concluded that this robot can be used for very complex and precise tasks, performed under high speed. In this thesis, the robot successfully managed to do the task of identifying the correct tag and flawless placement of that tag. MELFA RV-2SDB can be used for a wide range of tasks, which demand both high precision and speeds. In the scope of this work, every part of applications is being utilized in a distributed control schemes such that the control of robot is integrated in flexible assembly. I identified a set of requirements that apply to all web-based remote laboratories and focused on solutions to these requirements. Specifically, I presented solutions to interface, control and design difficulties in the client and server-side software of the robot when implementing a remote laboratory architecture. The combination of shared physical hardware and shared middleware software allows for experiments that build upon and compare against results on the same platform and in the same environment for common tasks. I described how researchers or future students can interact with the PC / Robo Com application and its environment remotely through a network of same TCP/IP ranged interface, as well as develop similar interfaces to visualize and run experiments remotely. The remote laboratory was designed for remote development as a primary use case. However, another significant use is for educational purposes. A DigiLab4U remote laboratory available for students could improve distance education, the accessibility, and the cost of running laboratories.

There are several steps that can be considered to make these experiments and project more effective. Some aspects of that are discussed in the followings section.

## 7.2. Future Work

Despite all the design and integration considerations, some parts of the system still require modifications and enhancements. A few among them are highlighted here.

- ***Automation of door system:*** The door of RFID cabinet is still an issue because currently, a person should open or close the door manually before or after the experiment respectively. In future, some measures are advised such that the door of RFID cabinet would be made autonomous and it would be controlled via a relay or some other kind of signaling device. The user would be able to send a signal remotely to open the door and once the robot has finished placing the tag inside the chamber, it would be sent a signal again to shut the door automatically. This task is of critical importance and will be solved shortly in near future.
- ***Video Surveillance:*** Cameras are advised to be planted in the whole lab so that all the experiments can be visualized remotely through a video surveillance. The video monitoring is part of the basic experiment that is considered to be deployed soon.
- ***Safety Fence or Perimeter Security:*** The idea is to build/install a security fence around experimental premises. The fact that the whole apparatus is expensive plus the robot is sensitive to handle requires some safety measures. In the future, it is thought to install a security fence or other safety mechanism such that when there would be some kind of interruption during the experiment because of mankind, like if the robot is operational and some person happens to be in the premises of experiment and opens the fence door, the robot would automatically stop. This is to ensure the safety of both, the robot and the person.
- ***Precision Enhancement of the Tag:*** Right now, the experiment was comprised of a couple of specified tags and the positions of those tags were already taught to the robot. In the future, when there would be some addition of multiple tags, the robot would be programmed in such a way that it would be able to configure the position of those newly added tags precisely and automatically.

These are some precautions or future tasks that are under consideration in the scope of this project.

# 8. REFERENCES

[1] Wirtschaft Digital 2017 | Monitoring Report-Compact https://www.bmwi.de/Redaktion/DE/Publikationen/Digitale-Welt/monitoring-report-wirtschaft-digital.html

[2] MITSUBISHI “Mitsubishi industrial robot SD Series, RV-2SD/2SDB Standard Specifications Manual (CR1DA-771 Controller)”

[3] IoT agenda [Online] https://internetofthingsagenda.techtarget.com/definition/RFID-radio-frequency-identification

[4] RFID Lab (HFT) [Online] https://digilab4u.com/rfid-lab/

[5] IoT agenda [Online] https://internetofthingsagenda.techtarget.com/definition/RFID-radio-frequency-identification

[6] RFID Lab (HFT) [Online] https://digilab4u.com/rfid-lab/

[7] RV-2SD-SQ Brochure Ver. B [Online]
RV-2SD/2SDB Series – Mitsubishi Electric

[8] CIROS Studio [Online]
https://www.festo-didactic.com/ov3/media/customers/1100/ciros_studio_manual_1.pdf

[9] CIROS Studio – Creating virtual learning environments [Online] https://www.festo-didactic.com/int-en/learning-systems/software-e-learning/ciros/ciros-studio-creating-virtual-learning-environments.htm?fbid=aW50LmVuLjU1Ny4xNy4xOC4xMTEwLjgxODY

[10] Daniel Bolla (FESTO DC-EC): *Melfa-Basic V. Handbook*, 2013

[11] JADLOVSKÝ, J. – LACIŇÁK S. – CHOVAŇÁK J. - ILKOVIČ J. 2010. Proposal for distributed control system of flexible production line, Journal of Cybernetics and Informatics, vol. 11, Košice, Slovakia, ISSN: 1336-4774

[12] CRnQ/CRnD Controller Instruction Manual Troubleshooting.pdf http://meltrade.hu/download.php?f=9618-crnqcrnd-instruction-manual-(troubleshooting)-bfp-a8662-v-(07.12)

[13] RV-2SD, 2SDB Robot Arm Setup and Maintenance.pdf http://meltrade.hu/open_pdf.php?f=9641-rv-2sd-instruction-manual-

[14] RV-2SDB Standard Specification Manual.pdf https://robotics.ee.uwa.edu.au/courses/robotics/project/festo/MPS_TD_V2.4_EN/English/06_Robot/RV-2SDB/Mitsubishi%20manuals/bfp-a8790b.pdf

[15] Petri nets for modeling robots - Pure – Eindhoven University https://pure.tue.nl/ws/files/3989333/780942.pdf

[16] Petri nets for modeling robots - Pure – Eindhoven University https://pure.tue.nl/ws/files/3989333/780942.pdf

[17] Ciros_Studio_Manual_1.pdf
https://needoc.net/ciros-studio-manual-1

# APPENDICES

## System’s Data – CIROS Studio File

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Automation Code of MELFA RV-2SDB %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

Def Char COMMAND           % Variable declaration %
OPEN "COM1:" AS #1         % Communication Line opening as File 1%
*LOOP:INPUT #1, COMMAND    % Input from Com Application %
IF COMMAND = "Status?" THEN GOTO *STATUS
IF COMMAND = "tag1" THEN GOTO *TAG1
IF COMMAND = "finished tag1" THEN GOTO *PICK1
IF COMMAND = "tag2" THEN GOTO *TAG2
IF COMMAND = "finished tag2" THEN GOTO *PICK2
PRINT #1, "INVALID COMMAND"
GOTO *LOOP
*STATUS:        % Status check of robot %
PRINT #1, "OK"
GOTO *LOOP
*TAG1:          % Picking of tag1 and placing in RFID chamber %
SERVO ON
DLY 1
JOVRD 60
MOV P1
DLY 0.5
MOV P2
DLY 0.5
MOV P3
DLY 0.5
JOVRD 30
MOV P4
DLY 0.5
JOVRD 5
MOV P5
DLY 0.5
JOVRD 60
MOV P6
DLY 0.5
MOV P7
DLY 0.5
MOV P8
```

```
DLY 0.5
MOV P9
DLY 0.5
JOVRD 5
MOV P10
DLY 0.5
MOV P11
DLY 0.5
JOVRD 60
MOV P9
DLY 0.5
MOV P7
DLY 0.5
MOV P6
DLY 0.5
MOV P1
DLY 1
SERVO OFF
PRINT #1, "Job Done"
GOTO *LOOP
*PICK1:  % Picking of tag1 from RFID chamber and placing on table %
SERVO ON
DLY 0.5
JOVRD 60
MOV P6
DLY 0.5
MOV P7
DLY 0.5
MOV P8
DLY 0.5
MOV P9
DLY 0.5
MOV P11
DLY 1
JOVRD 5
MOV P10
DLY 1
MOV P9
DLY 1
JOVRD 60
MOV P8
DLY 0.5
```

```
MOV P7
DLY 0.5
MOV P6
DLY 0.5
MOV P5
DLY 0.5
JOVRD 5
MOV P21
DLY 1
MOV P22
DLY 0.5
JOVRD 60
MOV P3
DLY 0.5
MOV P1
DLY 1
SERVO OFF
PRINT #1, "Job Done"
GOTO *LOOP
*TAG2:    % Picking of tag2 and placing in RFID chamber %
SERVO ON
DLY 1
JOVRD 60
MOV P1
DLY 0.2
MOV P17
DLY 0.5
MOV P18
DLY 0.5
JOVRD 20
MOV P19
DLY 1
MOV P20
DLY 0.5
JOVRD 60
MOV P23
DLY 0.5
MOV P6
DLY 0.5
JOVRD 60
MOV P7
DLY 0.5
```

```
MOV P8
DLY 0.5
MOV P9
DLY 0.5
JOVRD 5
MOV P10
DLY 0.5
MOV P11
DLY 0.5
JOVRD 60
MOV P9
DLY 0.5
MOV P8
DLY 0.5
MOV P7
DLY 0.5
MOV P6
DLY 0.5
MOV P1
DLY 1
SERVO OFF
PRINT #1, "Job Done"
GOTO *LOOP
*PICK2:  % Picking of tag2 from RFID chamber and placing on table %
SERVO ON
DLY 1
JOVRD 60
MOV P6
DLY 0.5
MOV P7
DLY 0.5
MOV P8
DLY 0.5
MOV P9
DLY 0.5
MOV P11
DLY 1
JOVRD 5
MOV P10
DLY 1
MOV P9
DLY 1
```

```
JOVRD 60
MOV P8
DLY 0.5
MOV P7
DLY 0.5
MOV P6
DLY 0.5
MOV P23
DLY 0.5
MOV P20
DLY 0.5
MOV P24
DLY 1
JOVRD 5
MOV P28
DLY 1
MOV P29
DLY 0.5
JOVRD 60
MOV P17
DLY 0.5
MOV P1
DLY 1
SERVO OFF
PRINT #1, "Job Done"
GOTO *LOOP
END
```