

FUNDAMENTALS ON ROBOTICS

Use of COSIMIR: Palletization with RV-M2

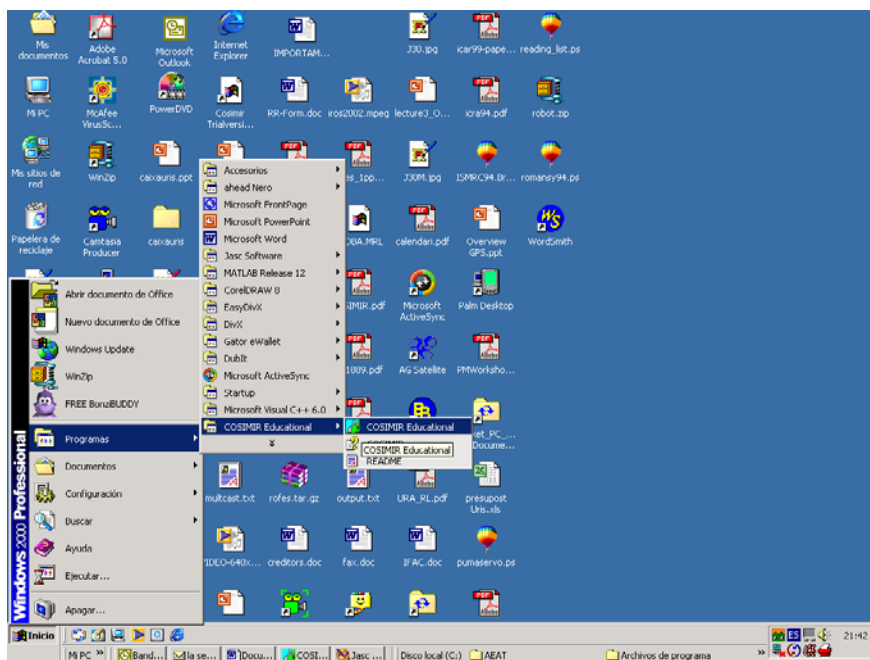
1. Objectives

The main objective of this first lesson is to become familiar with the Simulation Tool COSIMIR, and with the RV-M2 Mitsubishi Robot. We will learn how to move the robot using the *Teaching Box*, and programming using the Movemaster Command language (MRL).

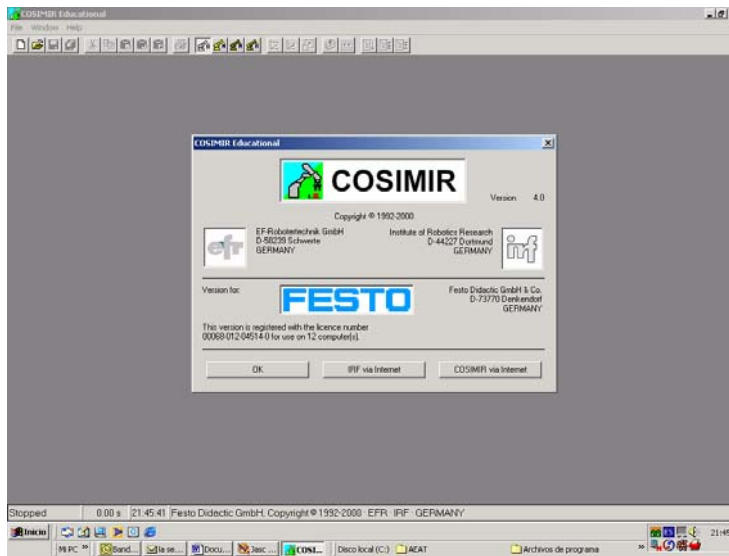
The COSIMIR Simulator will be used to simulate the execution of a simple palletization task with the Mitsubishi Movemaster RV-M2 robot. At the end of the practice a MRL program will be generated, which will also be used to test the task on the real Mitsubishi Movemaster RV-M1 robot. Although the real robot RV-M1 is not the same model available on the simulation environment (a RV-M2 model is used), both instructions sets are perfectly commutable. Therefore, the obtained program on the simulation environment should operate on the real robot (just beware of some few features that are only useful on the COSIMIR virtual environment).

2. COSIMIR Environment

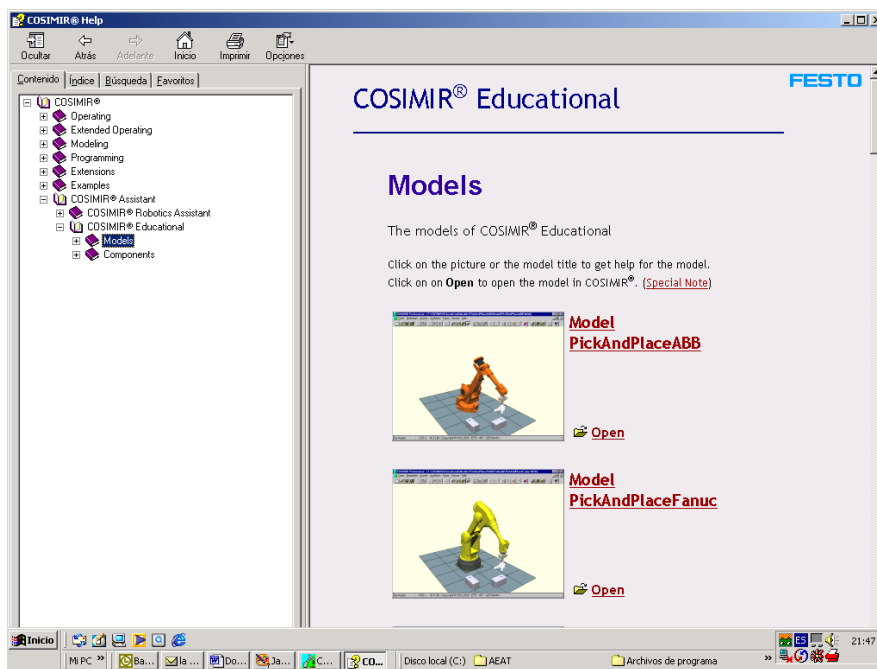
2.1. Start the application: Inicio->Programas->COSIMIR Educativa->COSIMIR Educativa



2.2. Type OK.

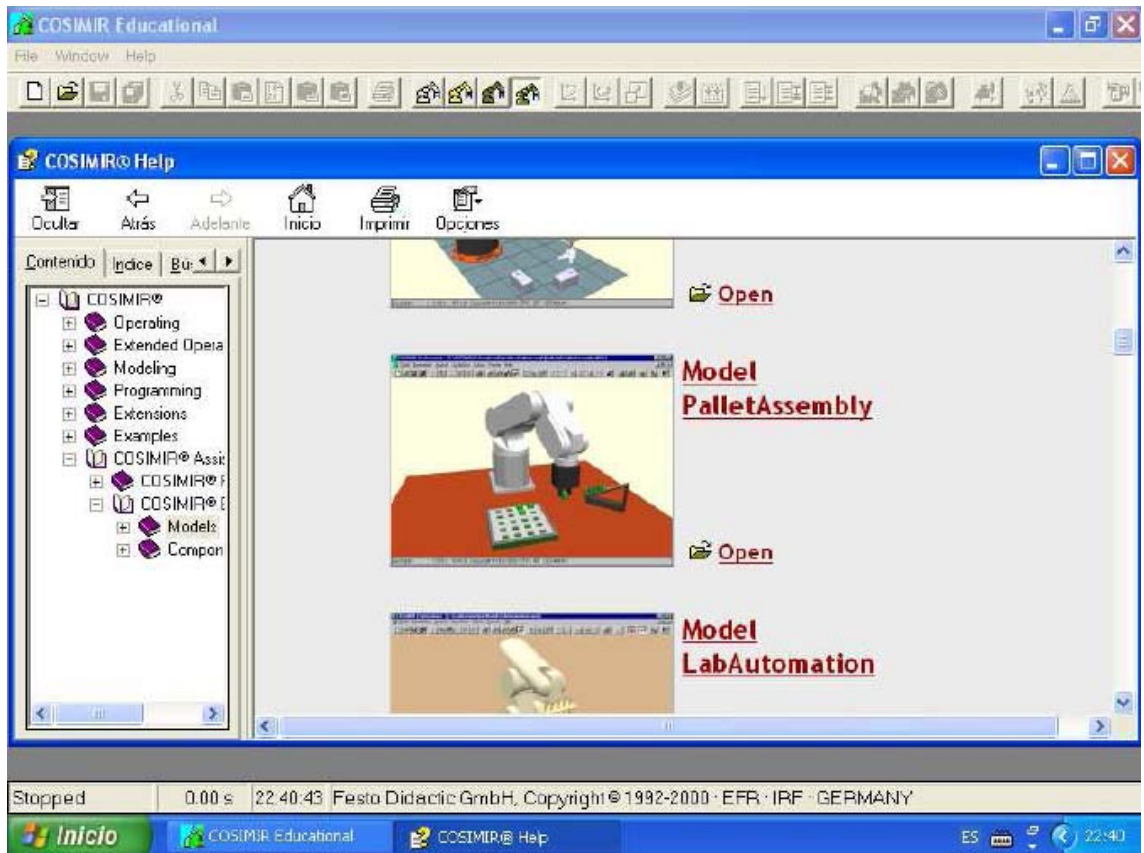


2.3. A HELP file will appear, that allows us to open the different available production cells.




Note that we can choose either to open a new cell or a predefined one. The whole set of predefined models is available by clicking on the right, although it is not restricted to only Mitsubishi robots. The scroll bar allows choosing the right model to perform the desired simulation.

2.4. In this practice, the “PalletAssembly” model should be selected by clicking the “Open” button:



This operation opens a new image which shows the virtual working area. It is composed by a table, the part feeder, and a 5x4 grid with the insertion spaces spatially arranged in an equidistant way.

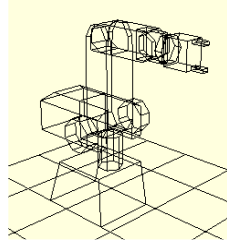


When you open PalletAssembly Model the program that appears at the Program window is not a MRL program. However, you can execute  this program for seeing what you have to do as a practical exercise.

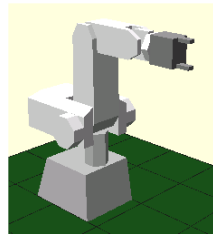
2.5. The 3D visualization can be done using different models. It depends of the choice:



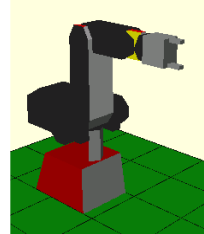
From left to right, it is possible to see the different representations of the scene:



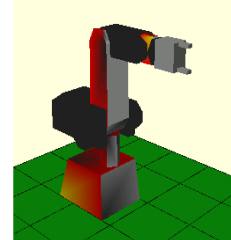
Wire Frame



Filled

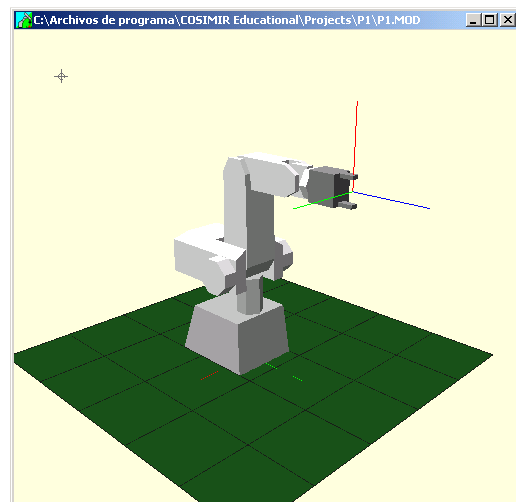
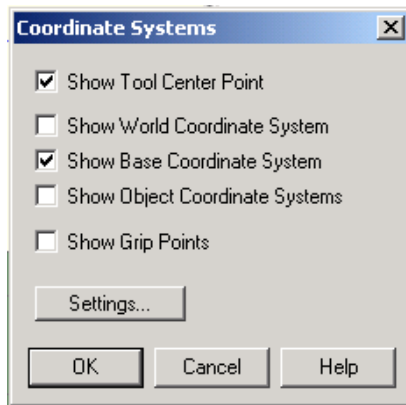


Flat Shaded

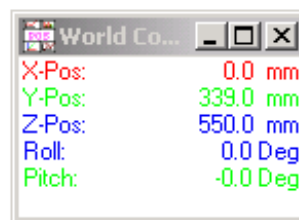
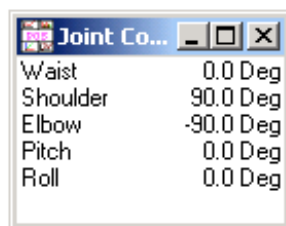


Smooth Shaded

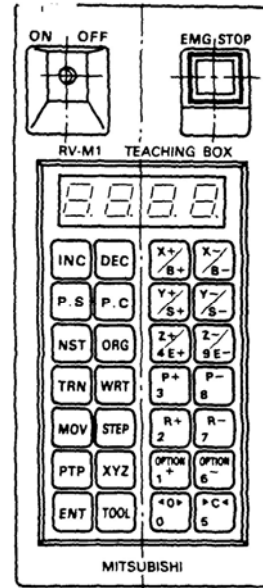
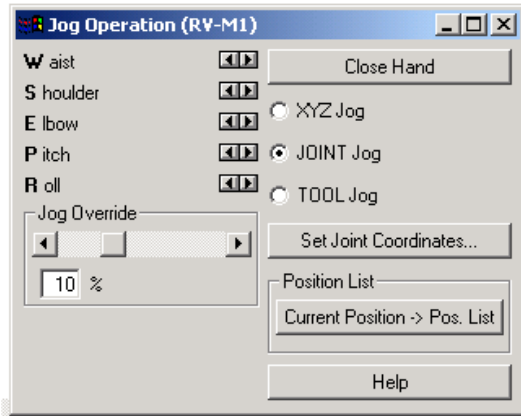
2.6. Next, we will set the visualization of the coordinate systems (base of the robot and end-effector, TCP Tool Center Point). We will select Extras->Coordinate Systems.



2.7. We set the visualization of the robot position using robot coordinates (Extras->Robot Position -> Show Joint Coordinates), and in world coordinates (Extras->Robot Position-> Show World Coordinates).



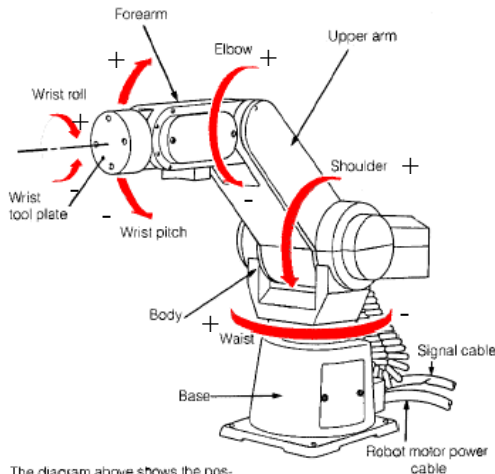
2.8. In order to start working with the robot, it is possible to set the Teaching Box (Extras->Teach in). It will appear a window that provides us the same operational facilities that the Teaching Box.



There are 3 different types of movements of the robot:

2.8.1. Joint Movement.

Select the option JOINT Jog. You can use the controls to move the different joints of the robot (Waist, Shoulder, Elbow, Pitch) and Roll. You can use the displacement bar (Jog Override) to change the speed.

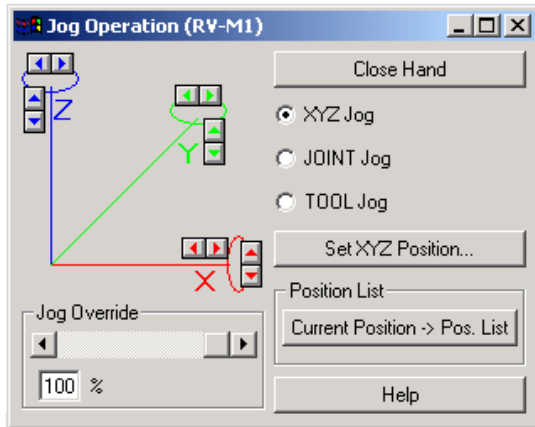


The diagram above shows the possible movements of each individual axis.

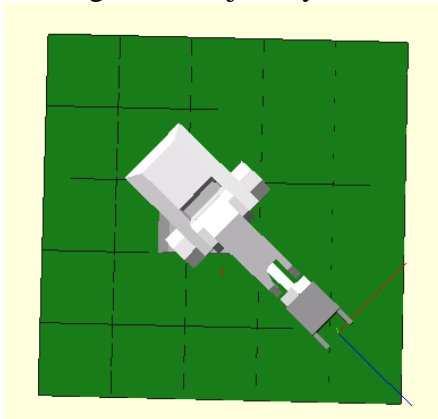
- Check that the signs of the angles are the same that the ones showed in the picture
- Compute the maximum and minimum values of the joint variable for each joint.
- Open and close the gripper.
- Locate the robot at position (0,0,0,0,0). You can use the control Set Joint Position.
- Save this robot position as a new position in the list. You can use the control Current Position->Pos. List.




2.8.2. Cartesian Movement with regarding to the base.

In this mode, the robot moves linearly following the axis of the base coordinate system. The rotations are produced regarding to these axis.



- Set the tracking mode of the end-effector (Extras->TCP Tracking). This option allows the program to show a trail when the robot moves. This option makes easiest the following of the trajectory.



- Use the position controls , orientation  and zoom , for obtaining a planar view of the robot.
- Build a squared trajectory in XY coordinates of the base.
- Locate the robot at the position $x=0$, $y=380$, $z=300$, Pitch=-70, Roll=-40. Use the control Set XYZ Position.
- Save this position as a position 2 of the list. Use the control Current Position->Pos. List.

2.8.3. Cartesian Movement regarding the end-effector system.

In this case a Cartesian movement regarding at the hand axis is produced.

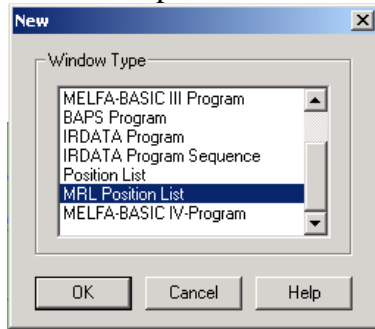
- Build a squared trajectory regarding to the hand axis.
- Check the rotations regarding the axis of the hand.

2.9. Programming with MRL.

In many robotics languages, programming means to obtain a position list and define the transitions between the positions.

2.9.1 Generation of a list of positions

To generate a new position list: *File->New->MRL Position List.*

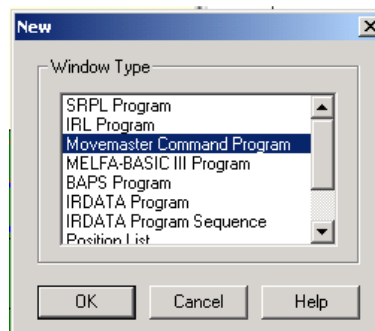


However, note that you do not need to perform this point since you have already defined a position list (similar to :)

No	Position	Orientation	Comment
1	339.0, 0.0, 550.0	0, 90,R,A	Start position
2	240.0, 260.0, 50.0	-130, 150,R,A	Grip cube
3	327.0, 350.0, 122.0	-130, 150,R,A	Release cube
4	272.0, 17.0, 35.0	-90, 180,R,A	Pallet start point
5	272.0, -22.0, 35.0	-90, 180,R,A	Pallet end point A
6	389.0, 17.0, 35.0	-90, 180,R,A	Pallet end point B
7	389.0, -22.0, 35.0	-90, 180,R,A	Pallet diagonal point
8	0.0, 0.0, 888.0	0, 0,R,A	
9	0.0, 0.0, 100.0	0, 0,R,A	

- To manually set a new position / displacement, right button click and select insert position.
- In case of error introducing a position, select the position again and click (right button) and select Properties in order to modify it.
- You can check the positions by double clicking them.
- Note that displacements (like pos number 9) could not be accessed by double clicking it.



2.9.2 Generation of file *.mrl: *File->New->Move Master Command Program*

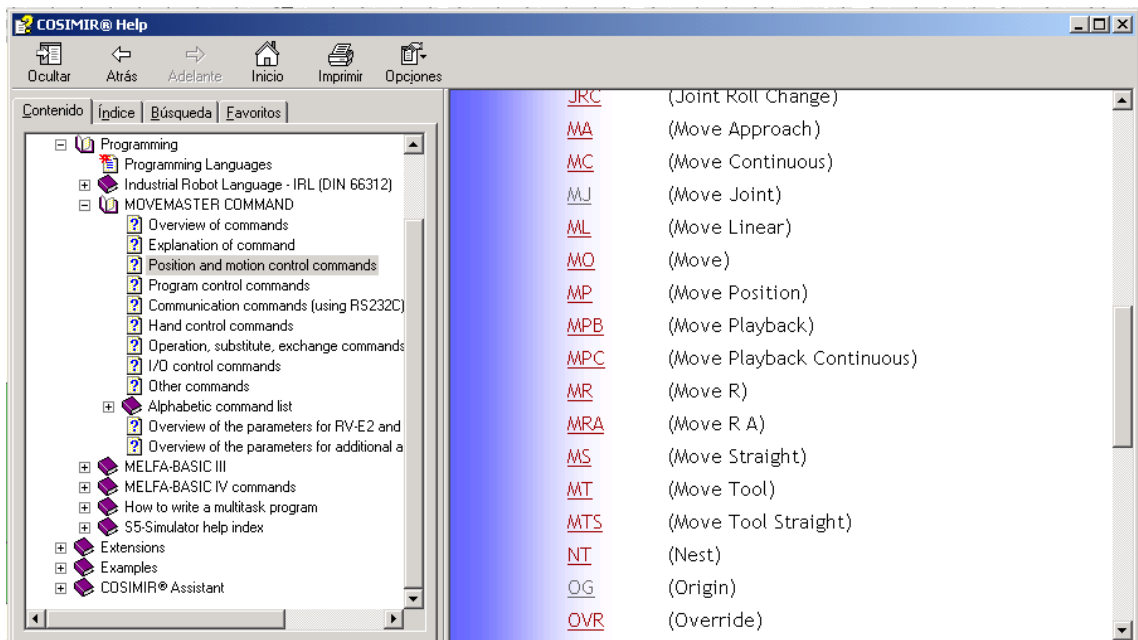


An empty window appears where we are able to introduce the code for programming the robot.

- Enter your first program:

```
[RV-2AJ] C:\D...
10 MO 1
20 MO 2
30 GC
40 MO 4,C
50 GO
60 MO 1
70 ED
```

- Compile: Execute-> Compile + Link or click .
- Step by step execution: Execute->Next Step (or Shift+F10 or click .
- Check the functionality of each instruction (please, read the instruction manual: Help->Contents->Contenido->MOVMASTER COMMAND->Position and Motion Commands). Note that not all the instructions are suitable for the robot RV-M2.



- Can you see any problem in the above implementation? After re-initializing the cell (reset the workcell, “Edit -> Reset Workcell”), let’s try this new code. Do you see the differences?

```
[RV-2AJ] C:\Doc...
10 MO 1
20 MT 2, -50
30 MS 2
40 GC
50 TI 5
60 MT 2, -50, C
70 MA 4, 9, C
80 MS 4, C
90 GO
100 TI 5
110 MA 4, 9
120 MO 1
130 ED
```


3. Practice Development

Once the working environment has been analyzed, it is time for the robot to start working! The desired task is simply a palletization operation: picking the parts up from the feeder and placing them into the grid in an arranged way. Note that while only 8 green cubes are awaiting in the feeder to be placed, the grid disposes of 4x5 available spaces. So, it is not necessary to fill the entire pallet.

To complete the operation, some instructions related to the pallet assembly operation will be needed, as well as some jump instructions and conditional evaluation (CP, LG, SM, NE ...). Provided the COSIMIR environment is equipped with a complete help system, further information can be found in the Instructions Manual of the RV-M1, which is available in the laboratory.

3.1. Useful Instructions (or at least, the most common ones):

Movements:

- Normal Movement (MO)
- Movement of the Tool (MT)
- Movement on a Straight line (MS)
- Changing the speed (SP): high speed to positioning (above the position), and low speed to approaching (to position)

Grip Control:

- Grip Close (GC)
- Grip Open (GO)

Program control:

```
WHILE (COUNTER_2<=5) DO  
  A  
ENDWHILE  
  
100 CP 2  
110 LG 5, 150  
  ' INSTRUCTIONS A  
130 IC 2 (don't forget it)  
140 GT 100  
150 ' ENDWHILE
```

```
REPEAT  
  A  
UNTIL (COUNTER_3>6) DO  
100 'BEGIN REPEAT  
110 'INSTRUCTIONS A  
120 CP 3  
130 SM 6, 100  
140 EO 6. 100
```

```
FOR N=1 TO 10 DO  
  A  
ENDFOR  
100 RC 10  
110 'INSTRUCTIONS A  
120 NX
```

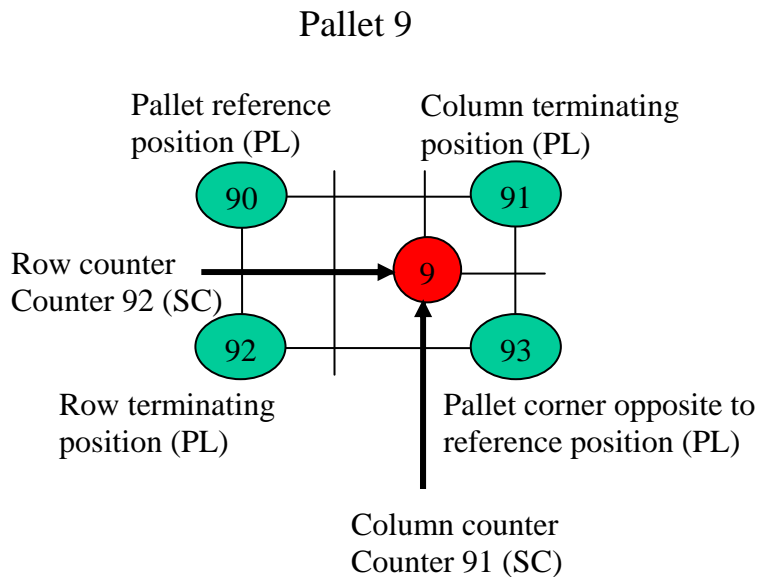
```
IF COMPTADOR_1=4  
THEN  
  A  
ELSE  
  B  
ENDIF  
  
100 CP 1  
110 EQ 4, 140  
120 ' INSTRUCTIONS B  
130 GT 150  
140 ' INSTRUCTIONS A  
150 ' ENDIF
```

Timing:

- Halt motion (TI)... for grasping properly the part.

Pallet definition:

- Define corners of the pallet (PL). The first argument is the pallet predefined corner, and the second one is the real position.
- Pallet initialization (PA): Number of pallet, rows and columns
- Position inside the pallet is controlled using COUNTERS (SC)
- Pallet position computation (PT)



```
25 'CARREGAR LES CANTONADES
30 PL 90,2
40 PL 91,1
50 PL 92,3
60 PL 93,4
65 ' PALET 9, 4 COL I 3 FIL
70 PA 9,4,3
75 ' COL=3, FIL=2
80 SC 91,3
90 SC 92,2
100 PT 9
110 MO 9
120 TI 20
130 MO 12
```

I/O Control Instructions:

- Setting the output state of a bit (bit 1, in this specific case) of the I/O board of Unit Control (OB+1, OB-1, set and reset respectively). For controlling external devices
- Note that the feeder is not a passive component. Therefore, when a part has picked been up the next one has to move to the free space, and so on. In this sense, every time a part is picked up the output 1 from the I/O port has to be enabled and immediately disabled (please, analyze how the instruction OB is working).

Some useful tricks:

- It is not necessary to write the numbers before each line: Edit -> Renumber (or ctrl+r).
- To reset the workcell, Edit -> Reset Workcell
- To save the program, you should save the code (.mrl) and the position list (.pos).