# 10. ROBOT PROGRAMMING

**EXAMPLE CODE 10.1**

Consider an example of a robot in pick-up action. Let the pick-up point be P1. Gripper is assumed to be open.

* First the gripper is moved near to the point P1 but away from it 50 away from point P1. **(APPRO P1,50)**
* After that the gripper is moved along a straight line to point P1 for picking the part placed in P1. **(MOVES P1)**
* After grabbing the part, gripper is closed **(SIGNAL)**
* Then, gripper is moved 50mm away from P1 **(DEPART P1,50)**
* **APPROS** and **DEPARTS** allows approach and depart through straight line motion.

**EXAMPLE CODE 10.2**

Consider an example where a robot manipulator needs to move to A, then move to B in a straight line and then move to D via C. The corresponding program can be:

**MOVE A**

**MOVES B**

**MOVE D VIA C**

**EXAMPLE CODE 10.3**

Consider a simple instance of a pick and place robot which picks up an object from a chute and place them in a container. The gripper is assumed to be open.

The series of activities can be listed as:

1. The end effector to move to a location near to the part to be picked.

2. Move to the part to be picked.

3. Close the jaws of the gripper to grab the part.

4. Lift the part from the chute.

5. Carry the part to a location away from the point where it is to be dropped.

6. Put the part into the container to be dropped.

7. Open the gripper jaws.

8. Move away from the box.

Program can be written as:

APPRO PICK, 70

MOVES PART

CLOSEI

DEPARTS 170

APPROS BOX, 220

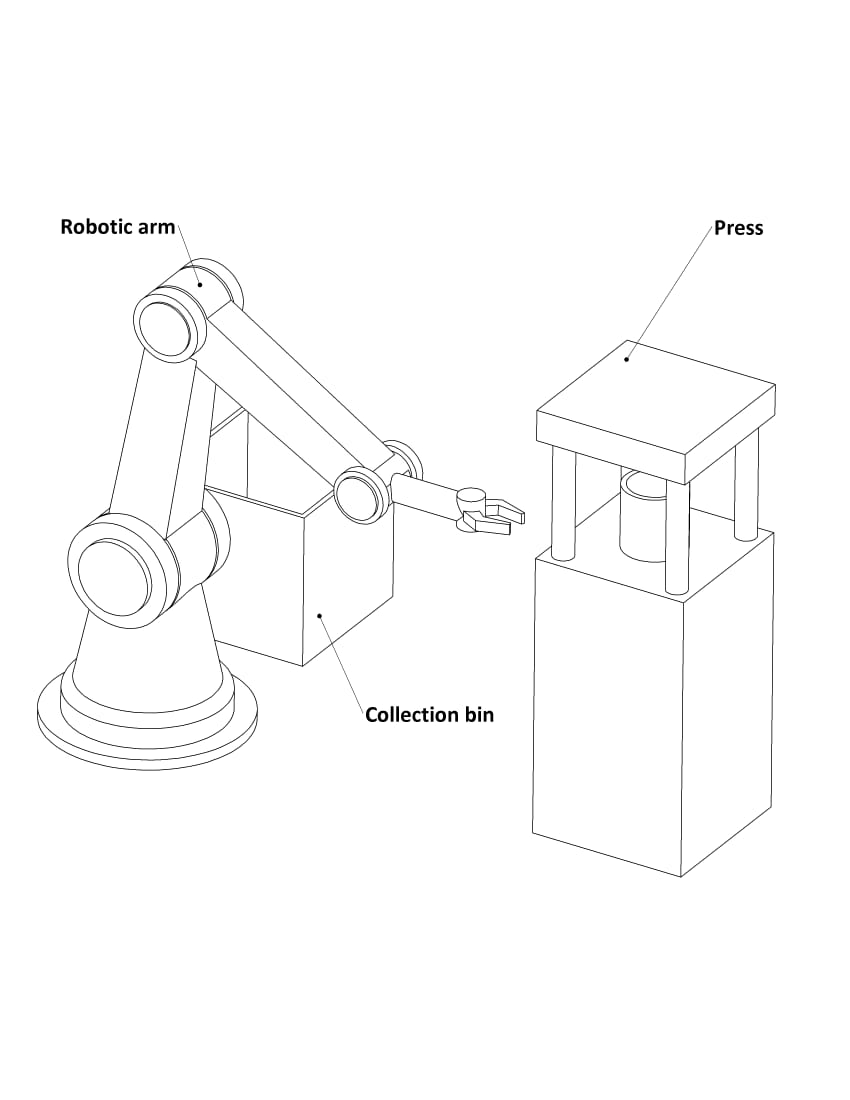
MOVE BOX

OPENI

DEPART

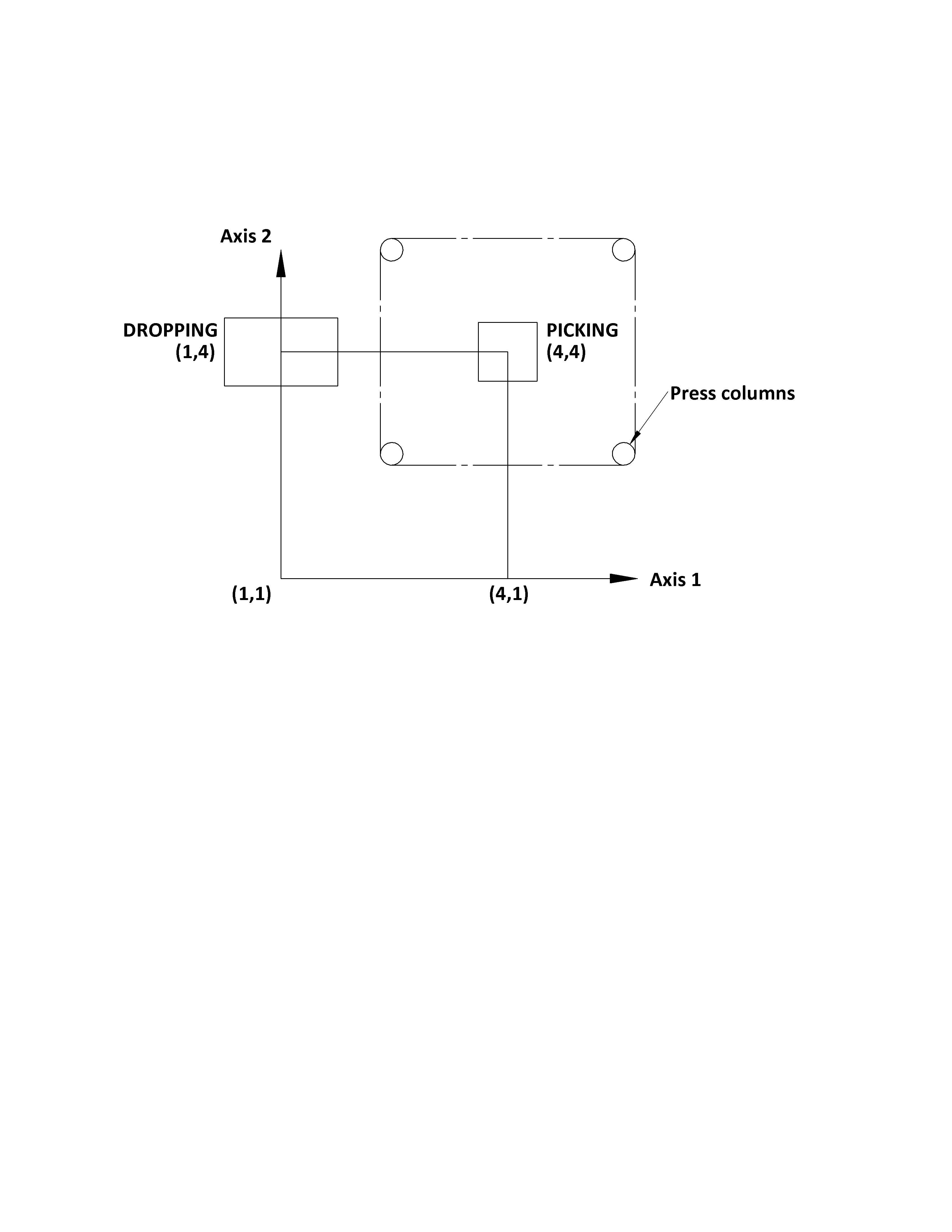
**EXAMPLE CODE 10.4**

Consider an unloading action of a two-axis robot in a press. The working environment and workcell layout is given in Figure 10.3 and 10.4. Parts have to be picked up and dropped at positions (4,4,) and (1,4) respectively. Press columns have been marked in figure. Though pick-up and drop- down points are in a straight line, arm cannot make the direct movement due to the presence of a press column in between and will collide with it. Robotic arm can wait at (4,1) till the press gets opened to grab the part from pick-up point.



*Figure 10.3 Working environment of a robot in an unloading procedure in a press*

Microcontroller is provided with various input-output ports. Ports 1 to 10 of the controller can be used to give output signals and 11 to 20 can be assigned for getting inputs. Here, we need 3 output lines, say, port 4 to actuate the press to get it opened, port 5 to open the gripper and port 6 to close the gripper. They are the signals to be sent from the controller. Input port say port 12 is used to get signal from press to know that it has been opened.



*Figure 10.4 Layout of the workcell of a robot in an unloading procedure in a press*

The series of actions can be listed as:

1. Start from initial position (1,1)
2. Move to position (4,1) to wait for the press to open
3. Wait for the input signal from press i.e. port 12
4. Instruct gripper to open through port 5
5. Move to pick-up point (4,4,)
6. Instruct gripper to close through port 6 (now the part is picked by the gripper)
7. Move to position (4,1)
8. Press is instructed to actuate through port 4
9. Arm is moved to initial position (1,1)
10. Arm is moved to the point where part is to be dropped i.e. (1,4)
11. Instruct the gripper to open through port 5 (now the part is dropped in the container)
12. Move back to initial position (1,1)

Positions are named as:

|  |  |
| --- | --- |
| SAFE | (1,1) |
| INTER | (4,1) |
| PICKUP | (4,4) |
| DROP | (1,4) |

Program can be written as:

MOVE SAFE

MOVE INTER

WAIT 12

SIGNAL 5

MOVE PICKUP

SIGNAL 6

MOVE INTER

SIGNAL 4

MOVE SAFE

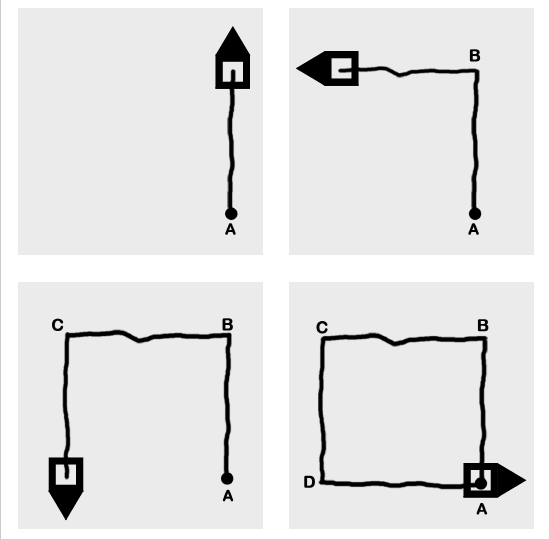
MOVE DROP

SIGNAL 5

MOVE SAFE

**EXAMPLE CODE 10.5**

To follow a square path as shown in Figure 10.5, robot should move from A to B, then similar motion B to C, C to D and D back to A. Same motion to be repeated four times.



*Figure 10.5 Robot following a square path starting from point A*

**Programming in Python**

To move from A to B,

*VWStraight (600,400): To move for a distance of 600mm at a speed of 400mm/second*

*VWWait()*

*VWTurn (90, 80):*

*VWWait()*

To move from A to B, VWStraight() is used to move it over a distance and VWTurn() is used to turn by an angle. VWWait() is added after each driving command. If VWWait() is not added, control will come back to the program after each command and new command will override the previous one. The program will go through all commands without execution and will run only the last command. In this case, it will only execute VWTurn (90,80) and will turn by an angle of 90 degrees, without moving from the initial position.

To make a square ABCD, the above code should be added in a loop and should be executed 4 times*.*

*for x in range (0,4):*

*VWStraight (600,400)*

*VWWait()*

*VWTurn (90, 80)*

*VWWait()*

**Programming in C/C++**

This code can also be simply written in C language as:

*#include “eyebot.h”*

*int main()*

*{*

*for (int i=0; i<4; i++) // run 4 sides*

*{*

*VWStraight(600, 400); // drive straight 600mm at a speed of 400mm/second*

*VWWait(); // wait until finished*

*VWTurn(90, 80); // turn 90 degrees*

*VWWait(); // wait until finished*

*}*

*}*