Module - VI

Robot Programming - Programming methods, Robot language classification, Robot language structure, element and its functions. Motion, End-effector and Sensor commands in VAL programming language. Simple programs.
Industrial applications of robots in material handling and assembly. Mobile robots, Recent developments in Robotics.
Total no of hrs assigned : 7 hrs.
Total no of hrs taken : 11 hrs.

Robot Programming methods :-

1. Teach Method
2. Lead Through
3. Off-line programming

Teach Method (Powered)

* Logic for the pgm - menu based s/m / using a text editor
* Main chara - robot is taught the positional data.
* Teach pendant + controls to drive the robot in a no: of different coordinate s/ms is used to manually drive the robot to the desired locations.
* Locations are stored with names that can be used within the robot program.

Coordinate s/ms are :

* Joint co-ordinates - Robot joints are driven independently in either direction.
* Global co-ordinates - Tool centre point of the robot can be driven along X, Y & Z axes of the robots global axes s/m.

Tool coordinates (similar to global co-ordinates)

Axes of this one are attached to the tool centre pt of the robot & them move with it.. useful when the tool is near to the workpiece

Work piece co-ordinates

* Useful where small adjustments to the pgm are reqd. as it is s to make them along a major axis of the coordinate s/m. than along a general line. Similar to moving the position and orientation of the global coordinate s/m.

This method - simple to use where simple movements are reqd.
Disadvantage - robot can be out of prodn for a long time during reprogeamming.
Advantageous - robots do same task for their entire life.
Eg:- Robotic welding s/ms

Lead Through (Manual)

Paint spraying robots. Robot is progeammed by being physically moved through the task by an operator. Difficult for large robots. Any hesitations / inaccuracies that are introduced into the pgm cannot be edited out easily without reprogeamm the whole task. Robot controller records the joint positions at a fixed time interval and plays it back.

Off-line progeamming

CAD s/ms are being used to generate NC pgms for milling machines. Pgm robots from CAD data. Pgm structure is built up as for teach progeamming but intelligent tools allows the CAD data to be used to generate sequences of location & process inf'm..

Advantages

1. Reduced down time for programming & m improved
2. Pgming tools make pgming easier
3. Enables concurrent engg. & reduces product lead time.
4. Assists cell design and allows process optimisation

Acc: to British Automation and Robot Association (BARA) - 90% of robots are pgmd using Teach pendant.

To pgm the robot, operator moves it from pt-to-point using buttons on the pendant to move it around and save each position individually When whole pgm has been learned, robot can play back the pts at full speed.

Advantages

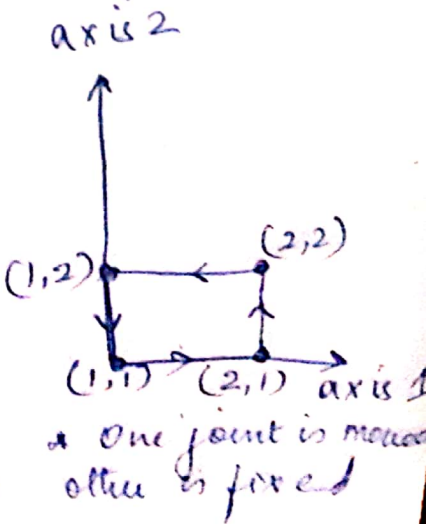
1. allow precise positioning as robot can be programmed using numerical coordinates in every coordinate sys ms.
2. used for simple, painting in a st line / large flat surface.

Disadvantages

1. During pgmming, all ops are halted (Teach mode)
2. Training to learn & pgm.
3. Skilled crafts peopled - not familiar with programming

Program 1 - Rectangular work space.

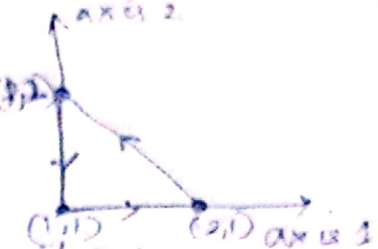
Step	Move	Comments
1	1, 1	Move to lower left corner.
2	2, 1	" lower right "
3	2, 2	" upper right "
4	1, 2	" upper left "
5	1, 1	" back to start posn



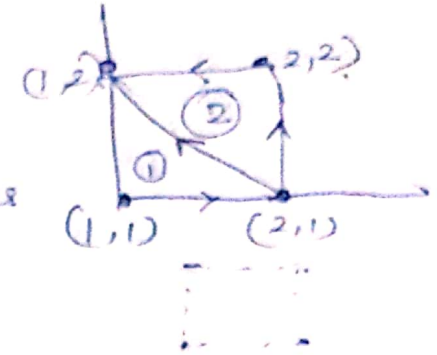
* one joint is moved other is fixed

Program 2

Step	Move	Comment
1	1,1	Move to lower left corner
2	2,2	" lower right "
3	1,2	" upper left "
4	1,1	Move back to start position



* Both joints are to be moved.
* Plan the motion path correctly.
* Limited - sequence non servo robots.
  Pgmd using manual set up procedures than lead-through methods.
* Move both joints at the same time
* Slew motion (diagonal)



* Limited sequence..
  * move in sequence. (border)
  * Move one axis at a time

Robot Language Classification

First generation language :-
[command statements]

* off line progeamming + progeamming thru robot pendan teaching. eg:- VAL
* sensory data handling is limited.
* comm with other computers
* Branching, i/o interfacing, commands leadering to a sequence of movements of arm & body & opening & close of end-effector is possible.

MOVE — motion sequence of manipulator
WAIT, SIGNAL - i/o capabilities
BRANCH - subroutines.

Limitations

* inability to specify complex arithmetic computations for use during program execution.
* inability to make use of complex sensors & sensor data
* less capable to communicate with computers.

Second Generation Languages.

* Overcome limitations in 1st generation & more complex tasks could be done. — Structured programming languages
* They possess structured ctrl constructs used in computer programming languages. eg:- AML, RAIL, MCL, VAL
* similar to computer programming — skill is needed.
* Teach pendant to define locations in the workspace

Features & capabilities

1. Motion control - same as for 1st generation (complex motion also)
~~Advanced Sensor~~ MCL language - based on APT lines, circles, planes, cylinders & other defined in APT & MCL
2. Advanced sensor capabilities - more than simple on-off & control by means of sensor data.
   - use of analog & digital. Communicate with the devices
   - Control of the gripper with enhanced sensor capabilities
     1st generation — involves commands to open/close the grippers.
     2nd " — sensored grippers to measure forces
3. Limited Intelligence - ability to utilize information received abt the work environment to modify s/m behaviour in a programmed manner.

If error – 1st generation – stop functioning.
– 2nd " – sensor data – recover by activating some other progeams.
Just it should be progeammed into the robot controller.
Decision making progeams.

4. Communication and data processing
* for the purpose of keeping records, generating reports and controlling activities in the work cell.

5. Extensibility
to handle future appln, future sensing device & future robots.
Language is expanded by developing commands, subroutines & macros statements (pass parameter values from the main pgm) not in initial Instn set.

Future Generation Language.

1. World modelling – 3D world knowledge, develop its own step by step procedure to perform a task
Offline robot progeamming their world modelling and a high level object oriented commanad will be obeyed by robots.
(Eg:- TIGHTEN THE NUT)

Robot Language structure.

* 2nd generation language - current state of the art in textual languages.

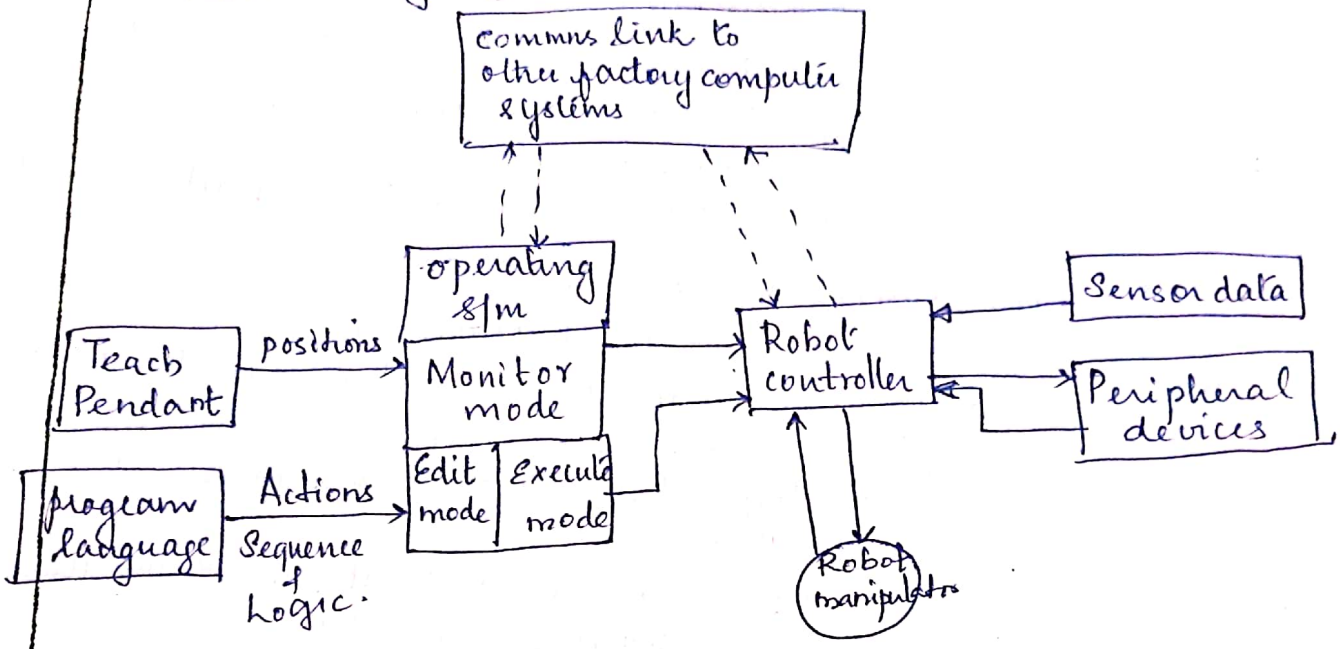


Diagram of robot system showing various components of the s/m that must be coordinated by means of the language.

* Support data comm with other computer s/ms
* " programming of the robot.
* control of the manipulator, and interface with peripherals & sensors

Operating systems

1. Programmer - CRT monitor, an alphanumeric keyboard, teach pendant storage, (magnetic tape/disk)
2. permits the user to determine whether to write a new pgm, edit a program, run a pgm etc. - OS
3. OS - to describe the s/w that supports the internal operation. facilitate the operation of the computer by the user & to maximize the performance and efficiency of the s/m. and associated peripheral devices.

A robot language OS – (1) Monitor mode
(2) Run mode
(3) Edit mode

Monitor mode – (1) overall supervisory control (supervisor mode)
(2) user can define locations in space using teach pendant.
(3) set speed control for the robot,
(4) store programs
(5) transfer programs from storage back into ctrl m/m
(6) switch b/w run & edit.

Run mode – for executing a robot program.
(1) perform sequence of instructions in the program
(2) When testing a new program in run mode. can employ debugging procedures to develop a correct pgm.

Edit mode – (1) provides an Instn set that allows the user to write new programs / edit pgm.

Robot language pgm is processed by OS using either an interpreter / compiler.

Interpreter – pgm in OS that executes each instn of the source (Robot Language Pgm) pgm one at a time.

Compiler – pgm in OS that passes through entire source pgm and pretranslates all of the instns into m/c level code than can be read and executed by the robot controller.
MCL – robot language that is processed by a compiler.

Robot Language Elements and Functions.

Motion Commands

MOVE and Related Statements

Robot language vs computer progeamming language = Manipulator motion ctrl.

MOVE A1 // Move end of the arm (end-effector) from its present position to the pt A1, position and end effector causes the arm to move with a joint interpolated motion.

MOVES A1 // VAL-II language for Straight line interpolation. Straight line trajectory from the current position to the pt A1 and causes the robot arm to follow that trajectory.

MOVE A1 VIA A2 // To control the trajectory, End effector passes through some intermediate pt as it moves from the present posn to next pt via point.
(when obstacles and clearances occur)
To move its arm to point A1 via point A_2

For material handling applications :-
Before pick up, gripper is to be moved to some intermediate location above the part. - approach.
Assume gripper is open.

APPRO A1, 50 // End effector is moved near to pt A1, but offset from the pt along x axis in the negative direction (above the part) by a distance of 50mm.

MOVES A1 // From this location move straight to A1

SIGNAL (to close gripper) // Closes its gripper around the part.

DEPART 50 // Robot moves away from pickup pt along x axis to a distance of 50mm.

APPROS, DEPARTS - using straight line interpolation (Absolute move)

* Incremental move :- direction & distance of the move is defined. or delta move.

DMOVE(1,10) // joint to be moved & distance of the move
unit – inches / mm – prismatic.
– degrees – revolute.

DMOVE(<4,5,6>, <30,-60,90>) //

MOVE ARM2 to A1 // in AL language, arm no: 2 is moved from current position to A1 pt

SPEED control

define velocity with which the robot's arm is moved.

When speed command is given in monitor mode, eg:-

SPEED 60 IPS // speed of the end-effector during progeam execution is 60 inches per second.

SPEED 75 // robot should operate at 75% of normal speed during pgm execution.

If SPEED 60 IPS specified under monitor command & following statement appeared in the progeam SPEED 75. then it means that subsequent statements should be performed at a speed ie. 75% of 60 IPS (45 IPS)

To define points in work space.

* usually done by means of a teach pendant

HERE A1 // That point location is named A1

<50.526, 236.003, 14.581, 25, 090, 125.750>
x-y-z coordinates in world space. wrist rotation angles

DEFINE A1 = POINT <50.526, 236.003, 14.581, 25.090, 125.750>

Paths & Frames

Several points are connected together to define a path in workspace.

DEFINE PATH1 = PATH (A1, A2, A3, A4) // path 1 consists of series of points A1, A2, A3 and A4 defined relative to the robot's world space. All points should be predefined

MOVE PATH1 // robot arm would move through the sequence of positions defined in PATH1 using a joint interpolated motion b/w the pts.

MOVES PATH1 // S: line interpolation.

A reference frame is a cartesian coordinate s/m that may have other points / paths defined relative to it.

Eg:- A part replicated but diff: position & orientation in design

Routing of same pattern at several locations around the contour of a curved plate. Program the routing path relative to a reference frame & redefine the reference frame for each location on the part.

Eg:- Nine identical patterns to be made. each one with a different reference frame.

FRAME 1 ..... FRAME 9. A routing path, ROUTE

DEFINE ROUTE : FRAME 1 = PATH (P1, P2 ... P7) // series of points P1 through P7 defines the routing pattern at the first position on the part identified by FRAME 1.

MOVES ROUTE : FRAME 1
To repeat the same path, only relocating and reorienting it relative to successive reference frames.
MOVES ROUTE : FRAME 2

If MOVES ROUTE // world space coordinate s/m ⇒ reference frame.
The path would be transformed to robots world cartesian coordinate s/m.

END EFFECTOR AND SENSOR COMMANDS

SIGNAL - to initiate o/p signals.
WAIT - to await i/p signals.

End-effector Operation

SIGNAL - To open / Close gripper.
OPEN & CLOSE // action to occur during the execution of the next motion.
OPENI & CLOSEI // action to occur immediately without waiting for the next motion to begin which can be defined by a parameter setting in VAL-II
CLOSE 40MM or CLOSE 1.575 IN. // for a nonservoed gripper, close the gripper to an opening of 40mm.
Some grippers also have tactile / force sensors built into the fingers Permit the robot to sense the presence of the object & to apply a measured force to the object during gripping.

CLOSE 3.0 LB // used to apply a 3 lb gripping force (pound)
CENTER - to centre its arm around object 1 pound = 0.45 kg
For end-effectors that are powered tools than grippers, robot must be able to position the tool and operate it.

OPERATE TOOL (SPEED = 125 RPM) } or.
OPERATE TOOL (TORQUE = 5 IN LB) }
OPERATE TOOL (TIME = 10 SEC) // after 10 sec the operation will terminate.

Eg:- powered screwdriver

Sensor Operation

SIGNAL - turn ON/OFF o/p signal.

SIGNAL 3, ON.
:
SIGNAL 3, OFF

Signal from o/p port 3 to be turned on at one point in the program and turned off at another point in the pgm.

o/p port no: 105
SIGNAL 105, 4.5 // o/p of 4.5 units (V) within the range of o/p sl.

WAIT - ON-OFF conditions (await i/p signal).

To provide power to some ext. devices.
(1) Verify that the device has been turned on before permitting the pgm to continue.
(2) Later in the pgm, robot turns off the device and device signals back that it has been turned off before the pgm continues.

```
SIGNAL 5, ON
WAIT 15, ON → acknowledge
      :
SIGNAL 5, OFF
WAIT 15, OFF .
```

Instead of port no:, label a name to the port.

```
DEFINE MOTOR 1 = OUTPORT 5.
DEFINE SENSR 3 = INPORT 15.
```

So it can be changed within the pgm.

```
SIGNAL MOTOR 1, ON
WAIT   SENSR 3, ON
         :
         :
SIGNAL MOTOR 1, OFF
WAIT   SENSR 3, OFF
```

```
DEFINE VOLT 1 = OUTPORT 105.

   SIGNAL VOLT 1
```

REACT statement

Continuously monitor an incoming signal and to respond to a change in the signal in some manner.

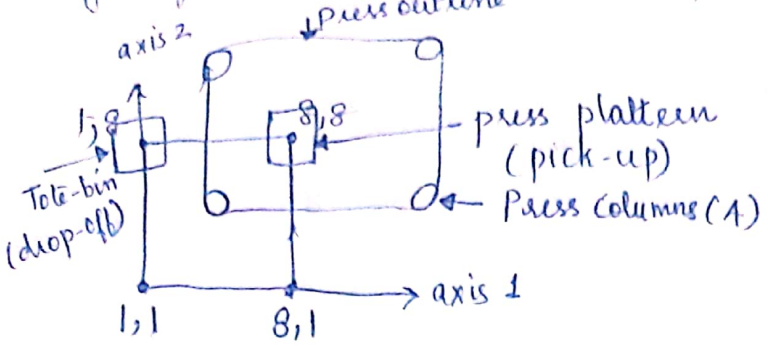
```
REACT 17, SAFETY // i/p line 17 is to be continuously monitored and when a change in its signal value occurs, branch to a subroutine SAFETY.

REACT I 17, SAFETY
```

Simple pick and place operation

```
1.  .PROGRAM move.Parts()
2.  ; Pick up parts at location "pick" and put them down
                                    at "place"
3   parts =100; Number of parts to be processed.
4.  height 1 =25; Approach | depart height at "pick"
5.  height 2= 50;
6.  PARAMETER.HAND.TIME =16; Set up for slow hand
7.  OPEN; Make sure hand is open
8.  MOVE start; Move to safe starting location
9.  For i=1 TO Parts; Process the parts
10. APPRO pick, height 1; Go toward the pick up.
11. MOVES pick; Move to the part
12. CLOSEI; Close the hand.
13. DEPARTS height 1; Back away
14. APPRO place, height2; Go toward the put-down
15. MOVES place; Move to the destination.
16. OPENI; Release the part.
17. DEPARTS height 2; Back away.
18. END; Loop for the next part.
19. TYPE "ALL DONE.", /I3, parts, " parts processed"
20. STOP; END of the progeam.
21. .END
```

2. Pgm for to accomplish the press unloading task

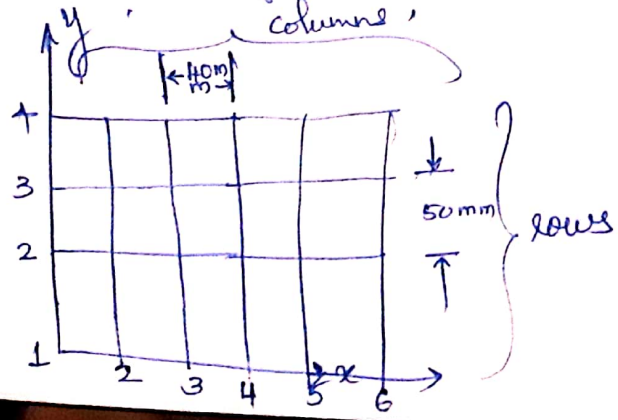


Sequence begins with the gripper in the open position.

Step	Move or signal	Comments
0	1,1	Start at home position.
1	8,1	Move to wait "
2	WAIT 11	Wait for press to open.
3	8,8	Move to pick up point
4	SIGNAL 5	Signal gripper to close. → wait for gripper to close
	→ DELAY 1SEC	
5	8,1	Move to safe position
6	SIGNAL 4	Signal press to actuate.
7	1,1	Move around press column
8	1,8	Move to tote-pan
9	SIGNAL 6	Signal gripper to open
	→ DELAY 1SEC.	→ wait for gripper to open.
10.	1,1	Move to safe position.

Assuming gripper is either opened / closed.

3. Palletizing operation.



Procedure :- Pick up parts from an incoming chute and deposit them onto a pallet. Pallet has 4 rows, 50mm apart & 6 columns 40mm apart. Plane of the pallet is assumed to be parallel to the xy plane. Objects to be picked up are about 25mm tall (1")

Variables : ROW, COLUMN, X, Y

Location Constants : PICK UP, CORNER

Location Variables : DROP

```
PROGRAM  PALLETIXE
DEFINE PICK UP = JOINTS (1;2,3,4,5)
DEFINE  CORNER = JOINTS (1,2,3,4,5)
DEFINE  DROP = COORDINATES (X,Y)
OPENI
ROW = 0          Initialize ROW
10  Y = ROW * 50.0   Compute y for drop off. point.
    COLUMN = 0       Initialize column.
20  X = COLUMN * 40.0   compute x for drop off point.
    DROP = CORNER + (X,Y)   Define drop for each iteration
    APPRO PICKUP, 50        Pickup Sequence.
    MOVES PICKUP            "
                            "
    CLOSEI                  "
    DEPART 50               Drop off sequence.
    APPRO DROP, 50          "
    MOVES DROP              "
                            "
    OPENI
    DEPART 50
    COLUMN = COLUMN + 1     Increment column variable
    IF COLUMN LT 6 GOTO 20  Check if column limit is reached
```

```
ROW = ROW + 1        Increment Row variable.
IF ROW LT H, GO TO 10   Check if row limit is reached.
END PROGRAM.
```

4. The same program is generalized. (for any pallet sizes)

```
PROGRAM PALLETIZE
DEFINE PICKUP = JOINTS(1,2,3,4,5)
DEFINE CORNER = JOINTS(1,2,3,4,5)
DEFINE DROP = COORDINATES(X,Y)
OPENI
SIGNAL 1
WAIT 11
CALL PALLET (MAXCOL=6, MAXROW=4, XSPACE=40, YSPACE=50)
SIGNAL 2
WAIT 12
GO TO 5
END PROGRAM
SUBROUTINE PALLET(MAXCOL, MAXROW, XSPACE, YSPACE)
ROW=0
Y = ROW * YSPACE
COLUMN=0
X = COLUMN * XSPACE
DROP = CORNER+(X,Y)
APPRO PICKUP, 50
MOVES PICK UP
CROSEI
DEPART 50
APPRO DROP, 50
MOVES DROP
OPENI
DEPART 50
COLUMN = COLUMN+1
IF COLUMN LT MAXCOL GO TO 20
ROW = ROW+1
IF ROW LT MAXROW GOTO 10.
END SUBROUTINE
```