SKEM4153
ROBOT TECHNOLOGY FOR AUTOMATION

CHAPTER 7
Work Cell and Robot Programming

Prof. Dr. Shamsudin H.M. Amin
Ir. Dr. Mohd Ridzuan Ahmad
(mdridzuan@utm.my)

Contents:

- Work Cell Controller Programming
- Programming Sequential Cell Activity
- Robot Language Development
- Language Classification
- Robot Program Fundamentals
  - V+ Programming Language
  - Program Creation, Location Creation
  - Creating and Altering Programs
  - Motion and Cycle Times
  - Relative Locations
  - Sample Programs

1. Work Cell Controller Programming

The 3 categories of work cell control software

- Software Developed In-house
- Application Enabler Software
- OSI (Open System Interconnected) Solution

1. Work Cell Controller Programming
- Software Developed In-house

- Written by end user using C or VB.
- Advantage – provide opportunity for tight integration of information and data.
- Disadvantage – development time and cost is high, inability to change the software easily when the cell hardware or configuration changes.

1. Work Cell Controller Programming
- Application Enabler Software

- Enabler software provides a set of software productivity tools for the development of control programs for CIM cells.
- Products such as Plantworks from IBM, Industrial Precision Tool Kit from HP etc. help reduce the difficulty in developing cell control and management applications.
- The enablers have a library of driver programs to permit exchange of data and information.
- In addition, they offer LAN and serial data communication support etc.
- Advantage – tenfold improvement in cell control and ease of program development.
- Disadvantage – the cell control is tied to a third party software solution.

1. Work Cell Controller Programming

- OSI (Open System Interconnected) Solution

- MMS (manufacturing message specification) is most often used. It is an ISO 9506, for network communication between intelligent devices in a production environment.
- MMS has 3 parts: service spec, protocol spec, robot interface & protocol spec.
- Advantage – MMS is a common communication standard, not a third party vendor.
- Disadvantage – only a limited number of equipment vendors agreed to support the standard.

2. Programming Sequential Cell Activity

In most applications, sequential control is performed by PLCs.

In some cases, the robot controller provides the control function.

2. Programming Sequential Cell Activity

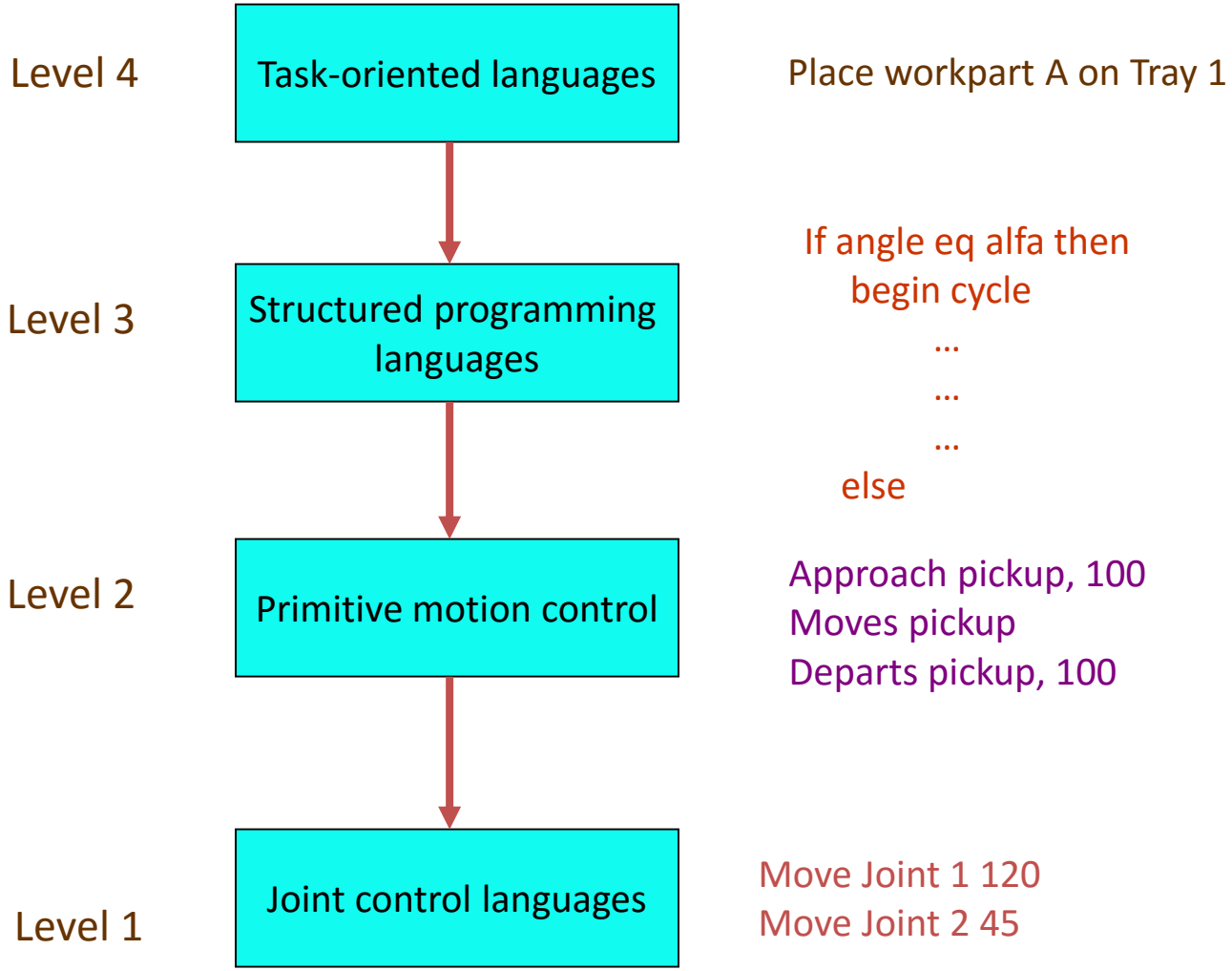
Sequential programming languages:

- Ladder logic – used with PLCs
- Instruction list – based on assembly language
- Structured text – similar to C
- Sequential function chart – a structured language based on the French GRAFCET language
- Function block diagrams – looks like electrical schematics with ladder logic elements

3. Language Classification

- Programming Levels
  - Level 1 Joint control languages
  - Level 2 Primitive motion languages
  - Level 3 Structured programming languages
  - Level 4 Task-oriented languages
- Industry practice
  - Manual lead-through programming
  - Powered lead-through programming
  - Textual languages

3. Language Classification



Programming Language Levels

3. Language Classification

Brief Survey on Programming Languages by Level

Origin	Level2	Level 3	Level 4
ABB Adept		RAPID V V+	
Cincinnati Milacron GMFanuc IBM	T3	 KARL AML, AML/E	 AUTOPASS
Kawasaki McDonnell Douglas Panasonic Rhino Sankyo Seiko Unimation	 RoboTalk VAL	AS MCL PARL-1 Sankyo language DARL II VAL II	

3. Language Classification

Industry practice

Manual lead-through programming

Powered lead-through programming

Textual languages

3. Language Classification

Manual lead-through programming

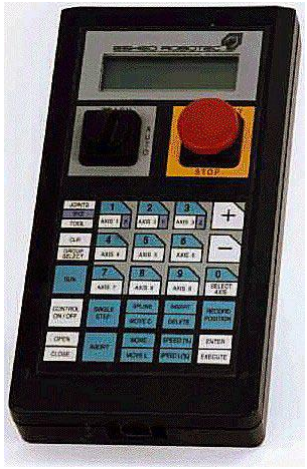
requires the programmer to physically hold the robot arm and end-effector, and manually move it through the desired motion cycle.

A down sized model of large and heavy robot is used to manipulate the motion. Teach buttons located near the wrist, are depressed and the brakes will be released to enable robot movement for teaching the robot.

3. Language Classification

Powered lead-through programming

Most commonly implemented today.



Teach pendant/ teach box is used to power drive and control the robot joints. The motion points are stored in the memory for playback during the work cycle.

3. Language Classification

Textual languages

Use an English-like language to establish the logic and sequence of the work cycle

Program instructions are entered via keyboard. The teach pendant is used to define the locations of the various points on the work space.

The robot programming language names the points as symbols in the program, and these symbols are subsequently defined by showing the robot the locations.

Advantages: permits the computations, allows more detailed logic flows, subroutines are allowed in the programs, and the greater use of sensors and communication.

4. Robot Program Fundamentals

We will now attempt to learn the
V+ Advanced Robot Programming Language

V+ Language

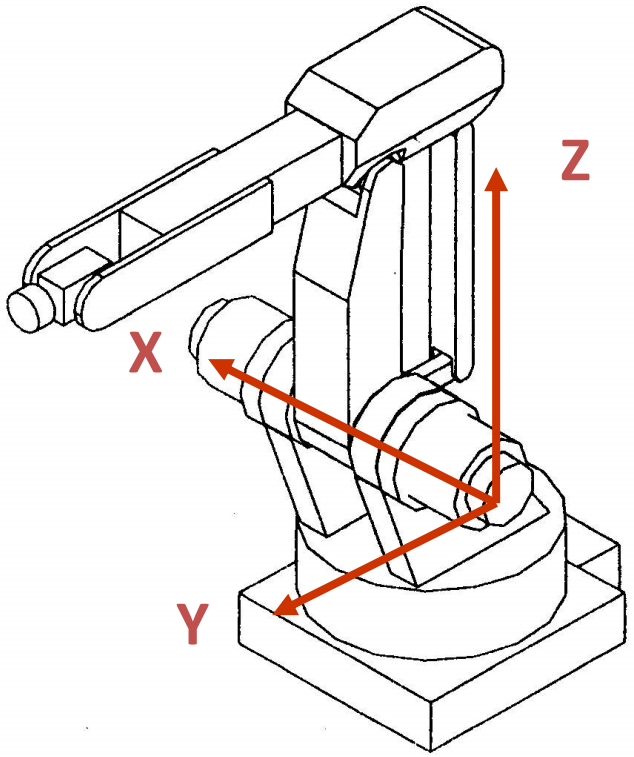
Robot States

World State

All movement is parallel to the World coordinates in the direction of X, Y, and Z.

It is also possible to rotate about the world coordinates by using the keys RX, RY, and RZ.

The gripper is selected by pressing T1 and operated by pressing either the + or - Speed Bar keys.



World coordinate system

V+ Language

Robot States

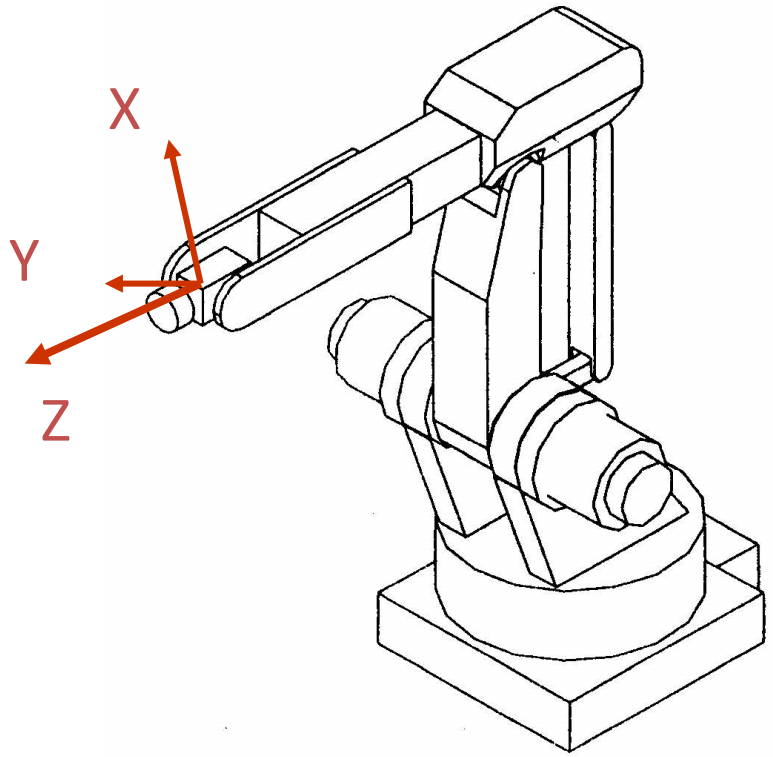
Tool State

All movement is parallel to the Tool coordinates in the direction of X, Y, and Z.

It is also possible to rotate about the Tool coordinates by using the keys RX, RY, and RZ.

The gripper is selected by pressing T1 and operated by pressing either the + or - Speed Bar keys.

NOTE: Tool X is in the direction as indicated by the joint 6 reference mark



Tool coordinate system

V+ Language

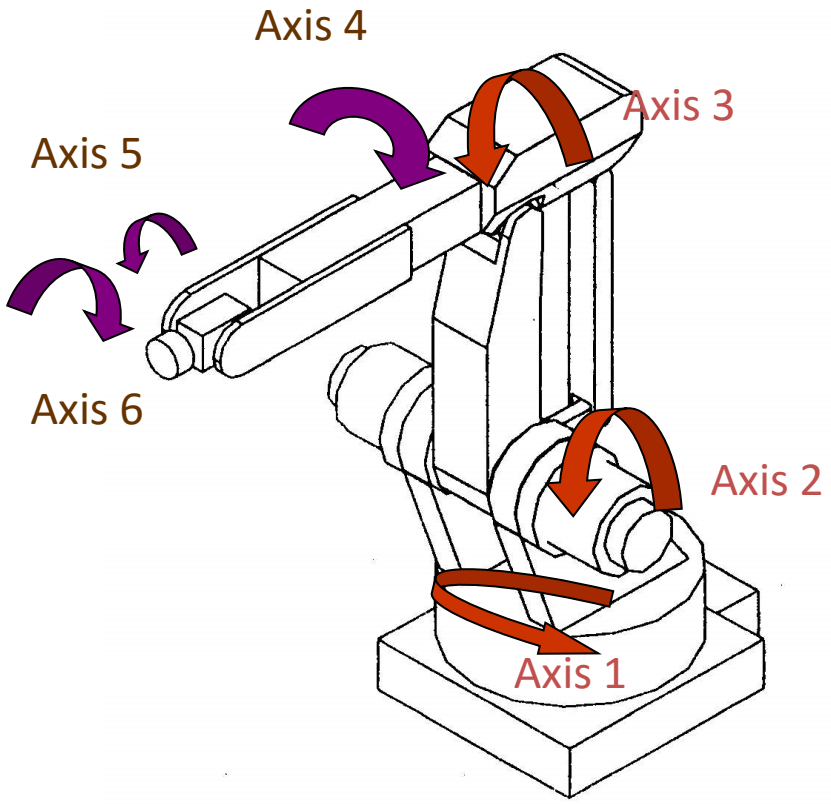
Robot States

Joint State

Individual joint rotation on axis 1 to 6.

Rotation can be either positive or negative in direction.

The gripper is selected by pressing T1 and operated by pressing either the + or - Speed Bar keys.



Program Creation

Types of Programs

V+ program is a collection of instructions that the Adept operating system follows to move a device, activate external/internal signals, perform computations, record data and display information.

- Robot Control Program
  - directly controls the robot
- Process Control Program
  - Can execute asynchronously, independently, and concurrently along with the robot control program. Often used to monitor and control external processes via external digital signal lines.
- Monitor Command Program
  - Consists entirely of monitor commands rather than program instructions.

Location Creation

Location Types

- Transformation
- Precision Points

Location Creation

Location Types

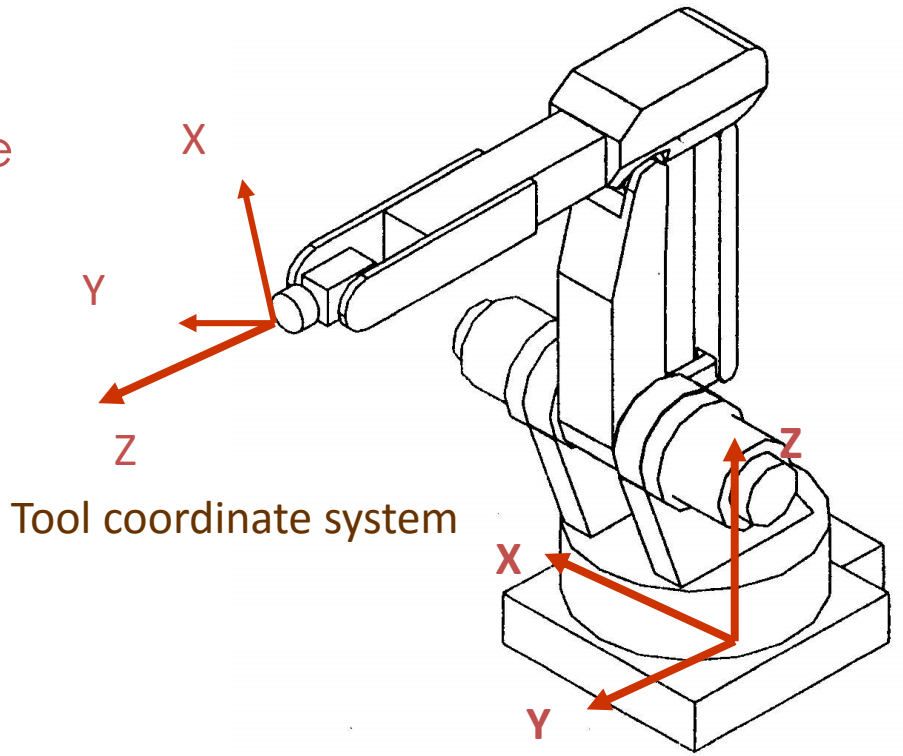
- Transformation

Six values of a transformation are labelled below, with the lower case characters representing "y=yaw", "p=pitch", "r=roll".

x	y	z	y	p	r
0	950	1500	0	120	30

Position (Cartesian space)

Orientation Yaw, Pitch, Roll (end-effector)



Transformation Values

Location Creation

Location Types

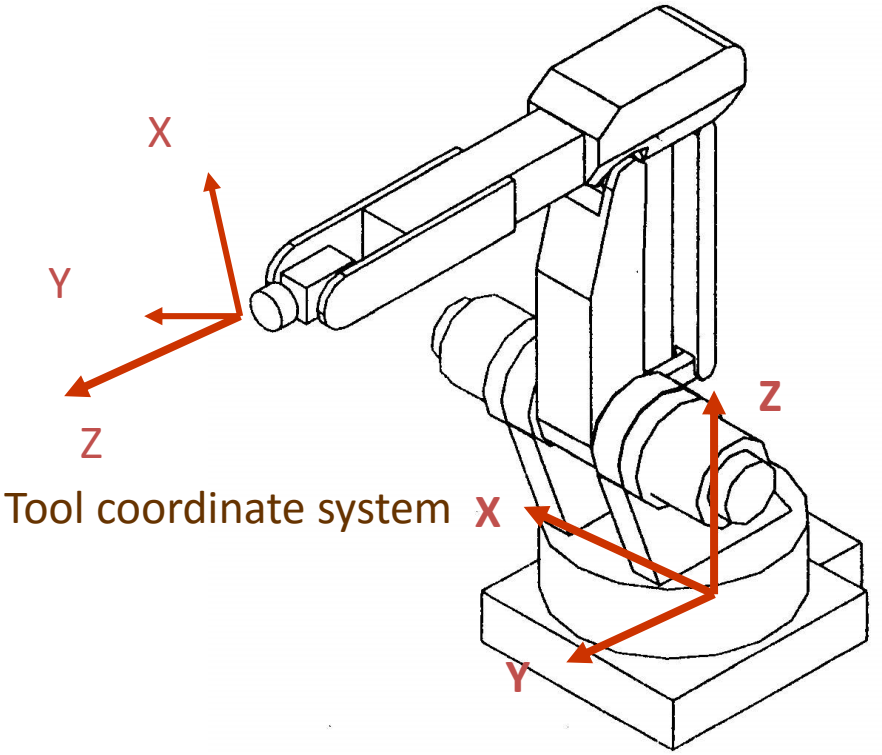
- Precision Points

For applications where joint orientation is critical, where exact trace of joints, elbow, wrist is critical.

For a 6 DOF robot, the six values of the joints are:

Jt1	Jt2	Jt3	Jt4	Jt5	Jt6	
0	0	0	0	0	0	(Pose 1)
90	0	0	0	0	0	(Pose 2)
90	0	-90	0	0	0	(Pose 3)
90	90	-90	0	0	0	(Pose 4)

Joint Values for Precision Points



Creating and Altering Location Variables

Commands to be typed in creating and altering location variables

HERE Monitor Command
First, move the robot (via Teach Pendant) to the desried location, then type

HERE loc_name
e.g. HERE pickuppt
HERE hole#1

TEACH Monitor Command
Often used to create multiple locations in a sequence and placed into a one dimensional array.

TEACH loc_name
e.g. TEACH pickuppoint

Each time "REC/DONE" button on the Teach Pendant (MCP-Manual Control Pendant) is pressed, a location is created. Continue the procedure (i.e. move the robot to a location, the press the "REC/DONE" button) until you are finished. Then when finished, press the RETURN key.

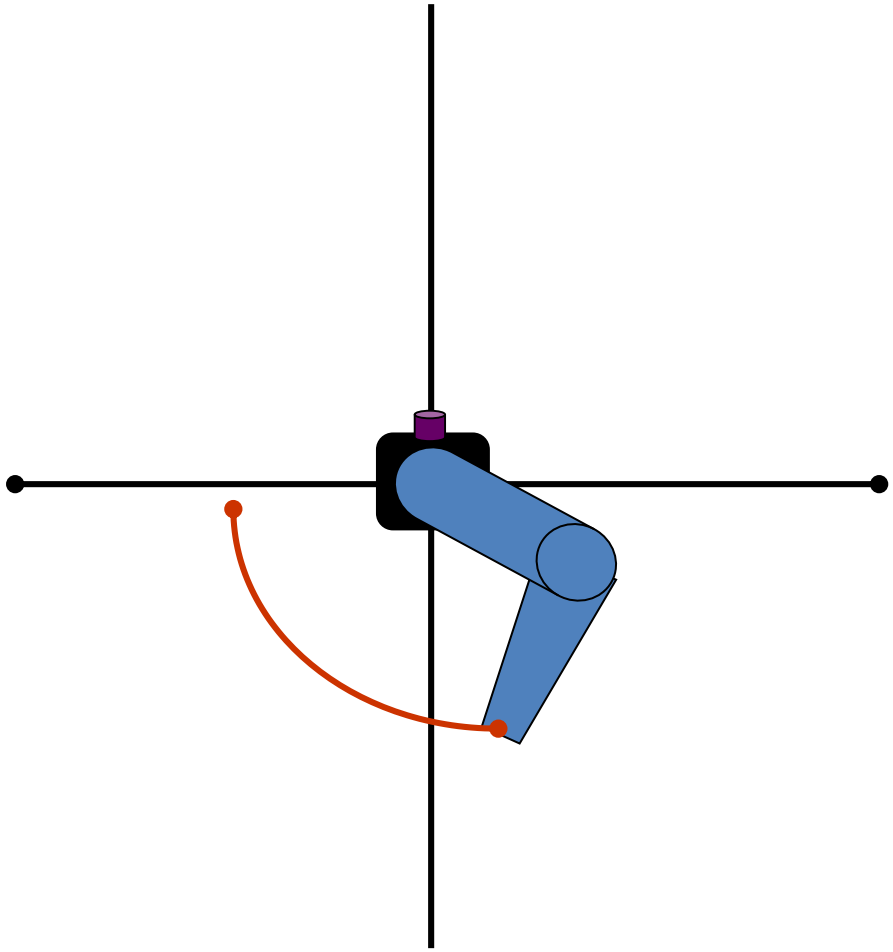
Motion and Cycle Times

Motion Path Types

Joint Interpolated Motions

The new joint angles are calculated, and the motion is executed by driving those angles without regard to any specific path.

The motion is very fast. This is also called jogging. E.g. 11m/s for RX90



Motion and Cycle Times

Motion Instructions

APPRO

This instruction initiates an approach to a specified location

APPRO loc_name, destination_height

It can either be a straight line motion, APPROS or a joint interpolated motion,

APPRO loc_name can be precision point (i.e. #loc_name) or transformation (loc_name) that currently resides in memory.

destination_height in mm, can be above (+) or below (-) the location along the Tool Z-axis
of when the location was defined.

Example: APPRO #toolpt1, 100
(approach a location stored as precision point #toolpt1, a distance 100mm above the Z-axis of the Tool when the location was defined)

Motion and Cycle Times

Motion Instructions

MOVE

This instruction initiates a move to a specified location

MOVE loc_name
MOVES loc_name

It can either be a straight line motion, MOVES or a joint interpolated motion, MOVE loc_name can be precision point (i.e. #loc_name) or transformation (loc_name) that currently resides in memory. The robot will assume the position an orientation of the location as it was created. Example:

MOVES loc_name
(moving in a straight line to a location stored as loc_name)

MOVE #loc_name
(move in joint interpolated motion i.e. jogging to precision point #loc_name this time using exactly the values of joint angles stored as precision point)

Motion and Cycle Times

Motion Instructions

DEPART

This instruction initiates a depart from a specified location

DEPARTS destination_height

It can either be a straight line motion, DEPARTS or a joint interpolated motion, DEPART destination_height in mm, can be above (+) or below (-) the location along the Tool Z-axis of when the location was defined.

Example: DEPARTS 100

(departing in a straight line motion from a specified location where the robot is currently at, to a distance 100mm, above the Z-axis of the Tool when the location was defined)

Motion and Cycle Times

Motion Speed

SPEED speed_factor

The speed factor values possible are:

Minimum	= 0.000001
Maximum	= use extreme care with values over 100
Normal full speed	= 100
Default at start-up	= 50

Example:
```
SPEED  5
FINE
```
(defines the speed to be 5, and next commands the robot to do FINE speed, i.e. slow)

Motion and Cycle Times

Cycle Times

There are 16 built-in timers (in Adept).

TIMER(timer_number) = time_value

Example 1:

Setting a timer to zero and timing the move to "pickpoint" then

```
BREAK
TIMER(1) = 0
MOVE pickpoint
BREAK
t = TIMER (1)
```

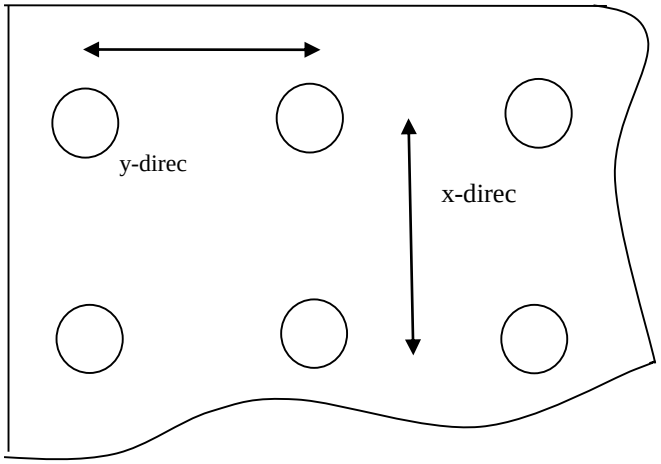
Example 2:

Setting timer to zero and then waiting for 5 sec

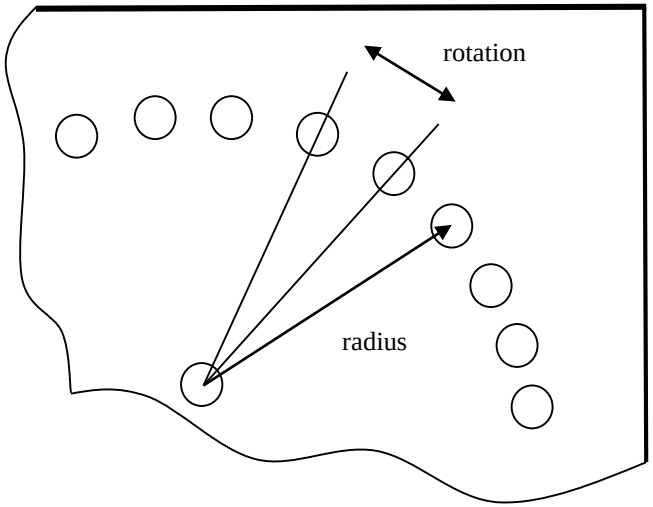
move to "pickpoint"

```
BREAK
TIMER(1) = 0
        WAIT TIMER(1) >= 5
MOVE pickpoint
```

Relative locations for Palletizing Program



Rectangular Pallet



Circular Pallet

An Example Program for Clock Assembly using V+ Programmimg Language

```
.PROGRAM clock()
   1        SPEED 75 ALWAYS
            RESET
            CALL gettool5()
   2        WAIT SIG(1048)
            IF count == 12 GOTO 3
            CALL woodbase()
            SIGNAL 48
            WAIT SIG(-1048)
            SIGNAL -48
            WAIT SIG(1048)
            CALL plate()
            CALL puttool5()
            CALL getsucker()
   3        CALL puttool5()
            MOVE apprtoolstnd
.END
```

```
.PROGRAM gettool5()
            SPEED 75 ALWAYS
            WAIT SIG(-1036)
            WAIT SIG(1038)
            SIGNAL 34
            MOVE apprtoolstnd
            MOVE overtool5
            SPEED 45
            MOVES griptool5
            WAIT SIG(1036)
            CLOSEI
            SIGNAL -34, 33
            SPEED 45
            MOVES tool5pin
            WAIT SIG(-1038)
            MOVES frnttool5
            MOVE apprtool5
            MOVE apprtoolstnd
            RETURN
.END
```

An Example Program for Clock Assembly using V+ Programmimg Language

Locations as defined by the robot controller

```
apprtoolstnd    -0.932732344 0.359892339 -0.022089828 0.354187518 0.925977468
                 0.130831733  0.067540027 0.114207059 -0.991158485 89.412101746
                 865.715270996 154.966278076
overtool5       0.350352436 0.925255001 0.145452425 0.935987949 -0.35156399 -
                 0.018145595  0.03434654 0.142499074 -0.989198864 164.168502808
                 1141.583618164 -110.121070862
griptool5       0.35035795 0.925256193 0.145431384 0.935985684 -0.351570785 -
                 0.018132156 0.034352537 0.142474443  -0.9892022164.1690979
                 1141.580444336 -182.095916748
```

TEXT AND REFERENCE BOOKS

- **Textbook:**

1. James A. Rehg: Introduction to Robotics in CIM Systems. Fifth Edition, Prentice-Hall. 2003.

- **Reference book:**

1. Mikell P. Groover: Automation, Production Systems, and Computer Integrated Manufacturing, Second Edition. 2004.
2. Mikell P. Groover, Mitchell Weiss, Roger N. Nagel, Nicholas G. Odrey: Industrial Robotics: Technology, Programming, and Applications, McGraw-Hill. 1986.
3. Farid M. L. Amirouche: Computer-Aided Design and Manufacturing. Prentice-Hall.
4. Richard K. Miller, Industrial Robot Handbook. Van Nostrand Reinhold, N.Y. (1987).

THANK YOU