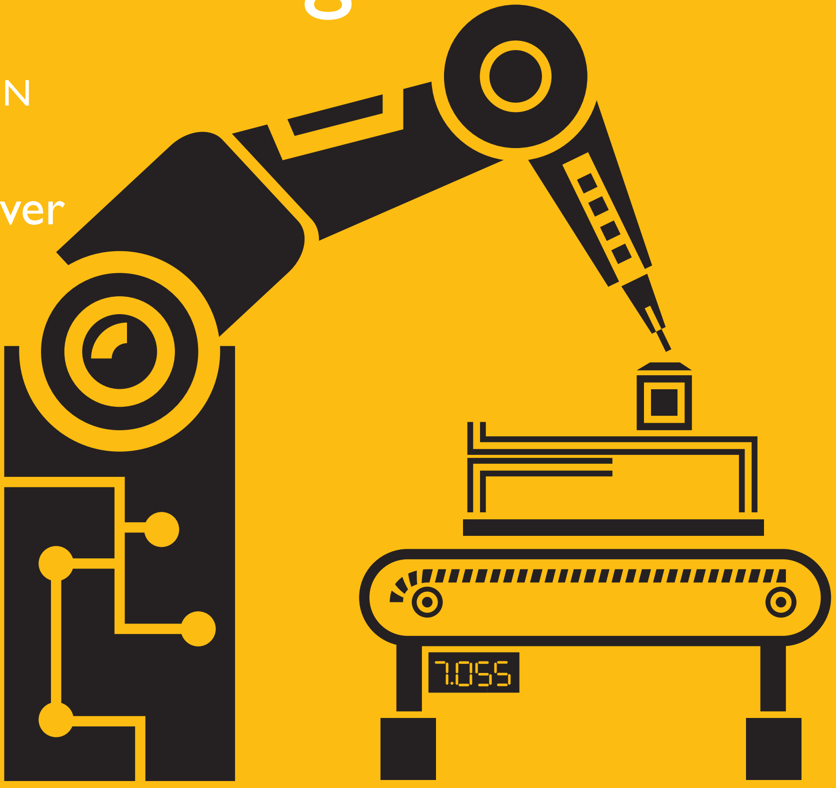
GLOBAL EDITION

Automation, Production Systems, and Computer-Integrated Manufacturing

FOURTH EDITION

Mikell P. Groover



ALWAYS LEARNING

PEARSON

Abbreviations Used in This Book

Abbreviation	Unabbreviated Unit(s)
A	amps
C	Celsius, Centigrade
cm	centimeters
F	Fahrenheit
hp	horsepower
hr	hour, hours
Hz	hertz $(\text{sec})^{-1}$
in	inch, inches
lbf	pounds force
m	meters
min	minute, minutes
mm	millimeters
MPa	megapascals (N/mm^2)
mV	millivolts
N	newtons
ops	operations
Pa	pascals (N/m^2)
pc	pieces, parts
rad	Radians
rev	revolutions
sec	second, seconds
V	volts
W	watts
wk	week, weeks
yr	year, years
μ-in	microinches
μm	microns, micrometers
μ-sec	microseconds
μV	microvolts
Ω	ohms

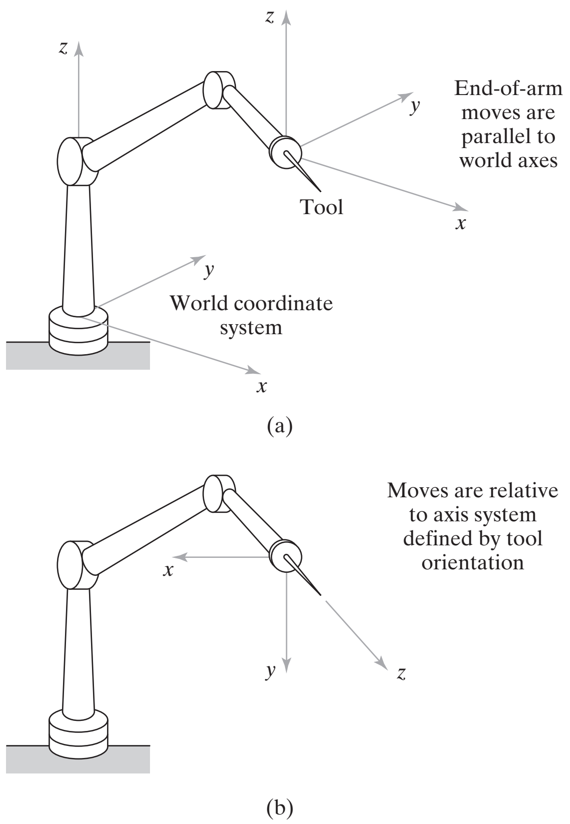

z
z
y
End-of-arm moves are parallel to world axes
Tool
x
y
World coordinate system
x
(a)
Moves are relative to axis system defined by tool orientation
x
y
z
(b)

Figure 8.13 (a) World-coordinate system. (b) Tool-coordinate system.

joints (types R, T, and V). Accomplishing straight line motion requires manipulators with these types of joints to carry out a linear interpolation process. In ***straight line interpolation***, the control computer calculates the sequence of addressable points in space through which the wrist end must move to achieve a straight line path between two points.

Other types of interpolation are available. More common than straight line interpolation is joint interpolation. When a robot is commanded to move its wrist end between two points using ***joint interpolation***, it actuates each of the joints simultaneously at its own constant speed such that all of the joints start and stop at the same time. The advantage of joint interpolation over straight line interpolation is that usually less total motion energy is required to make the move. This may mean that the move could be made in slightly less time. It should be noted that in the case of a Cartesian coordinate robot, joint interpolation and straight line interpolation result in the same motion path.

Still another form of interpolation is used in manual-leadthrough programming. In this case, the robot must follow the sequence of closely spaced points that are defined during the programming procedure. In effect, this is an interpolation process for a path that usually consists of irregular smooth motions, such as in spray painting.

The speed of the robot is controlled by means of a dial or other input device, located on the teach pendant and/or the main control panel. Certain motions in the work cycle should be performed at high speed (e.g., moving parts over substantial distances in the cell), while other motions require low speed (e.g., motions that require high precision in positioning the work part). Speed control also permits a given program to be tried out at a safe slow speed and then used at a higher speed during production.

Advantages and Disadvantages. The advantage offered by the leadthrough methods is that they can be readily learned by shop personnel. Programming the robot by moving its arm through the required motion path is a logical way for someone to teach the work cycle. It is not necessary for the robot programmer to possess knowledge of computer programming. The robot languages described in the next section, especially the more advanced languages, are more easily learned by someone whose background includes computer programming.

There are several inherent disadvantages of the leadthrough programming methods. First, regular production must be interrupted during the leadthrough programming procedures. In other words, leadthrough programming results in downtime of the robot cell or production line. The economic consequence of this is that the leadthrough methods are most appropriate for relatively long production runs and are less appropriate for small batch sizes.

Second, the teach pendant used with powered leadthrough and the programming devices used with manual leadthrough are limited in terms of the decision-making logic that can be incorporated into the program. It is much easier to write logical instructions using a computer-like robot language than a leadthrough method.

Third, because the leadthrough methods were developed before computer control became common for robots, these methods are not readily compatible with modern computer-based technologies such as CAD/CAM, manufacturing databases, and local communications networks. The capability to readily interface the various computer-automated subsystems in the factory for transfer of data is considered a requirement for achieving computer integrated manufacturing.

8.5.2 Robot Programming Languages

The use of textual programming languages became an appropriate programming method as digital computers took over the control function in robotics. Their use has been stimulated by the increasing complexity of the tasks that robots are called on to perform, with the concomitant need to imbed logical decisions into the robot work cycle. These computer-like programming languages are really a combination of on-line and off-line methods, because the robot must still be taught its locations using the leadthrough method. Textual programming languages for robots provide the opportunity to perform the following functions that leadthrough programming cannot readily accomplish:

- Enhanced sensor capabilities, including the use of analog as well as digital inputs and outputs
- Improved output capabilities for controlling external equipment
- Program logic that is beyond the capabilities of leadthrough methods
- Computations and data processing similar to computer programming languages
- Communications with other computer systems.

This section reviews some of the capabilities of the robot programming languages. Many of the language statements are taken from commercially available robot languages.

Motion Programming. Motion programming with robot languages usually requires a combination of textual statements and leadthrough techniques. Accordingly, this method of programming is sometimes referred to as *on-line/off-line programming*. The textual statements are used to describe the motion, and the leadthrough methods are used to define the position and orientation of the robot during and/or at the end of the motion. To illustrate, the basic motion statement is

MOVE P1

which commands the robot to move from its current position to a position and orientation defined by the variable name P1. The point P1 must be defined, and the most convenient way to define P1 is to use either powered leadthrough or manual leadthrough to place the robot at the desired point and record that point into memory. Statements such as

HERE P1

or

LEARN P1

are used in the leadthrough procedure to indicate the variable name for the point. What is recorded into the robot's control memory is the set of joint positions or coordinates used by the controller to define the point. For example, the aggregate

$$(236, 158, 65, 0, 0, 0)$$

could be utilized to represent the joint positions for a six-axis manipulator. The first three values (236, 158, 65) give the joint positions of the body-and-arm, and the last three values (0, 0, 0) define the wrist joint positions. The values are specified in millimeters or degrees, depending on the joint types.

There are variants of the MOVE statement. These include the definition of straight line interpolation motions, incremental moves, approach and depart moves, and paths. For example, the statement

MOVES P1

denotes a move that is to be made using straight line interpolation. The suffix S on MOVE designates straight line motion.

An incremental move is one whose endpoint is defined relative to the current position of the manipulator rather than to the absolute coordinate system of the robot. For example, suppose the robot is presently at a point defined by the joint coordinates (236, 158, 65, 0, 0, 0), and it is desired to move joint 4 (corresponding to a twisting motion of the wrist) from 0 to 125. The following form of statement might be used to accomplish this move:

DMOVE (4, 125)

The new joint coordinates of the robot would therefore be given by (236, 158, 65, 125, 0, 0). The prefix D is interpreted as delta, so DMOVE represents a delta move, or incremental move.

Approach and depart statements are useful in material handling operations. The APPROACH statement moves the gripper from its current position to within a certain distance of the pickup (or drop-off) point, and then a MOVE statement positions the end effector at the pickup point. After the pickup is made, a DEPART statement moves the gripper away from the point. The following statements illustrate the sequence:

```
APPROACH P1, 40 MM
MOVE P1
(command to actuate gripper)
DEPART 40 MM
```

The destination is point P1, but the APPROACH command moves the gripper to a safe distance (40 mm) above the point. This might be useful to avoid obstacles such as other parts in a tote pan. The orientation of the gripper at the end of the APPROACH move is the same as that defined for point P1, so that the final MOVE P1 is really a spatial translation of the gripper. This permits the gripper to be moved directly to the part for grasping.

A path in a robot program is a series of points connected together in a single move. The path is given a variable name, as illustrated in the following statement:

```
DEFINE PATH123 = PATH(P1, P2, P3)
```

This is a path that consists of points P1, P2, and P3. The points are defined in the manner described above using HERE or LEARN statements. A MOVE statement is used to drive the robot through the path.

```
MOVE PATH123
```

The speed of the robot is controlled by defining either a relative velocity or an absolute velocity. The following statement represents the case of relative velocity definition:

```
SPEED 75
```

When this statement appears within the program, it is typically interpreted to mean that the manipulator should operate at 75% of the initially commanded velocity in the statements that follow in the program. The initial speed is given in a command that precedes the execution of the robot program. For example,

```
SPEED 0.5 MPS
EXECUTE PROGRAM1
```

indicates that the program named PROGRAM1 is to be executed by the robot at a speed of 0.5 m/sec.

Interlock and Sensor Commands. The two basic interlock commands (Section 5.3.2) used for industrial robots are WAIT and SIGNAL. The WAIT command is used to implement an input interlock. For example,

```
WAIT 20, ON
```

would cause program execution to stop at this statement until the input signal coming into the robot controller at port 20 was in an "on" condition. This might be used in a situation where the robot needed to wait for the completion of an automatic machine cycle in a loading and unloading application.

The SIGNAL statement is used to implement an output interlock. This is used to communicate to some external piece of equipment. For example,

```
SIGNAL 21, ON
```

would switch on the signal at output port 21, perhaps to actuate the start of an automatic machine cycle.

The above interlock commands represent situations where the execution of the statement occurs at the point in the program where the statement appears. There are other situations in which it is desirable for an external device to be continuously monitored for any change that might occur. This would be useful, for example, in safety monitoring where a sensor is set up to detect the presence of humans who might wander into the robot's work volume. The sensor reacts to the presence of the humans by signaling the robot controller. The following type of statement might be used for this case:

```
REACT 25, SAFESTOP
```

This command would be written to continuously monitor input port 25 for any changes in the incoming signal. If and when a change in the signal occurs, regular program execution is interrupted, and control is transferred to a subroutine called SAFESTOP. This subroutine would stop the robot from further motion and/or cause some other safety action to be taken.

Although end effectors are attached to the wrist of the manipulator, they are actuated very much like external devices. Special commands are usually written for controlling the end effector. In the case of grippers, the basic commands are

```
OPEN
```

and

```
CLOSE
```

which cause the gripper to actuate to fully open and fully closed positions, respectively, where fully closed is the position for grasping the object in the application. Greater control over the gripper is available in some sensored and servo-controlled hands. For grippers with force sensors that can be regulated through the robot controller, a command such as

```
CLOSE 2.0 N
```

controls the closing of the gripper until a 2.0-N force is encountered by the gripper fingers. A similar command used to close the gripper to a given opening width is

```
CLOSE 25 MM
```

A special set of statements is often required to control the operation of tool-type end effectors, such as spot welding guns, arc welding tools, spray painting guns, and powered spindles (e.g., for drilling or grinding). Spot welding and spray painting controls are typically simple binary commands (e.g., open/close and on/off), and these commands would be similar to those used for gripper control. In the case of arc welding and powered spindles, a greater variety of control statements is needed to control feed rates and other parameters of the operation.

Computations and Program Logic. Many robot languages possess capabilities for performing computations and data processing operations that are similar to computer programming languages. Most robot applications do not require a high level of computational power. As the sophistication of applications increases in the future, the computing and data processing duties of the controller will also increase for functions such as calculating complex motion paths, decision making, and integrating with other computer systems.

Many of today's robot applications require the use of branches and subroutines in the program. Statements such as

GO TO 150

and

IF (logical expression) GO TO 150

cause the program to branch to some other statement in the program (e.g., to statement number 150 in the above illustrations).

A subroutine in a robot program is a group of statements that are to be executed separately when called from the main program. In a preceding example, the subroutine SAFESTOP was named in the REACT statement for use in safety monitoring. Other uses of subroutines include making calculations or performing repetitive motion sequences at a number of different places in the program. Using a subroutine is more efficient than writing the same steps several places in the program.

8.5.3 Simulation and Off-Line Programming

The trouble with leadthrough methods and textual programming techniques is that the robot must be taken out of production for a certain length of time to accomplish the programming. Off-line programming permits the robot program to be prepared at a remote computer terminal and downloaded to the robot controller for execution without interrupting production. In true off-line programming, there is no need to physically locate the positions in the workspace for the robot as required with present textual programming languages. Some form of graphical computer simulation is required to validate the programs developed off-line, similar to the off-line procedures used in NC part programming.

The off-line programming procedures that are commercially available use graphical simulation to construct a three-dimensional model of the robot cell for evaluation and off-line programming. The cell might consist of the robot, machine tools, conveyors, and other hardware. The simulator displays these cell components on the graphics monitor and shows the robot performing its work cycle in animated computer graphics. After the program has been developed using the simulation procedure, it is then converted into the textual language corresponding to the particular robot employed in the cell. This is a step in off-line robot programming that is equivalent to post-processing in NC part programming.

In off-line programming, some adjustment must be performed to account for geometric differences between the three-dimensional model in the computer and the actual physical cell. For example, the position of a machine tool chuck in the physical layout might be slightly different than in the model used to do the off-line programming. For the robot to reliably load and unload the machine, it must have an accurate location of the load/unload point recorded in its control memory. A calibration procedure is used to correct the three-dimensional computer model by substituting actual location data from the cell for the approximate values developed in the original model. The disadvantage of calibrating the cell is that some production time is lost in performing this procedure.