■ Teaching Robot Movement

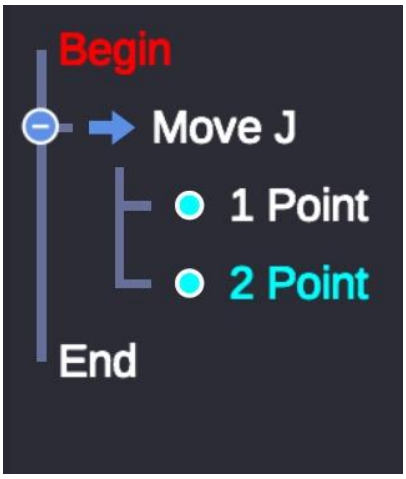
The basic robot teaching functions are 

Move and 

Point. Both icons are on the top bar when using the **Make** screen.

- **Move**: Defines motion property. Generates a movement command for the robot arm. Requires points to be defined.
- **Point**: A sub-function of **Move**. Defines a destination position for each movement.

After using the **Move** and **Point** functions in an empty program, the script field in the UI will look like the following.



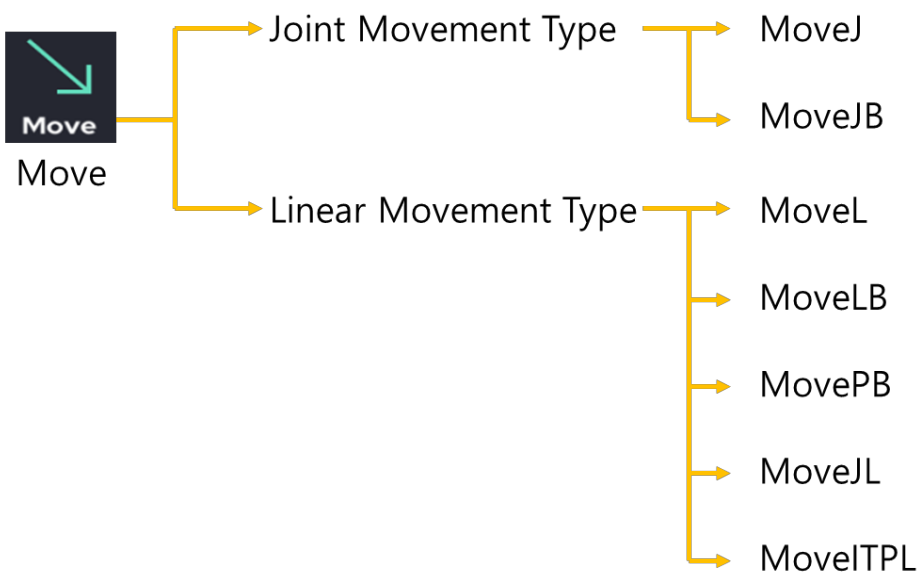
← motion property

← 1st destination

← 2nd destination

Details on each of the **Move** and **Point** functions follow on the pages below.

■ Move Function



Move sets the robot arm's motion properties. The two primary types of movements are **Joint** and **Linear**. These types are further broken down into commands, as shown in the figure above.

■ Joint Movement Commands

The Joint Movement Commands generate movement by setting the angular value of each individual joint (in degrees).

▷ MoveJ (Move Joint) :

Sets each joint angle to the values contained within the target **Point**.
Note: The movement speeds for all joints are slowed relative to the joint that requires the most movement time.

▷ MoveJB (Move Joint Blend) :

Starting from the initial arm configuration, the arm will move smoothly between each **Point** without stopping by using the Move J method.

■ Linear Movement Commands

The Linear Movement Commands generate movement by setting the position of the TCP in the Cartesian coordinate system. These commands use Cartesian coordinates (x,y,z coordinate values and rotations) as the target values for the movement.

▷ MoveL (Move Linear) :

Moves the TCP linearly (using x, y, and z) from the current position to the position contained within the target **Point** (in mm). Will also rotate the TCP (using Rx, Ry, and Rz) based on the orientation contained within the target **Point** (in degrees).

▷ MoveLB (Move Linear Blend) :

Starting from the initial arm configuration, the arm will move smoothly between each **Point** without stopping by using the Move L method. This method will generate an arc-shaped path.

For each **Point**, the user must specify a Blend Radius. This Blend Radius determines how far away the TCP will be from the **Point** when moving along the path.

If the Blend Radius is set to 0, the path will be the same as only using the Move L method.

The Blend Radius has a maximum value, which is half of the distance between the initial **Point** and the destination **Point**. This ensures that the arm will maintain a blended movement.

Move LB has two modes, Constant and Intended.

- Constant mode maintains the first **Point's** TCP orientation (Rx, Ry, and Rz) during movement, only changing the tool's position (x, y, and z) through the movements.
- Intended mode changes both the orientation and position the TCP as the arm moves.

▷ MovePB (Move Point Blend) :

MovePB is similar to MoveLB, but it is more universally available. For each **Point**, the user can set the blend amount in either distance or percentage (%). The speed can also be set separately for each point.

▷ MoveJL (Move J with Linear Input) :

Like MoveL, the Cartesian value of the target point is used as input. However, instead of going straight to the point, it uses MoveJ's method. When the Cartesian coordinate system input is received, it is converted into the target joint angle through inverse kinematics and inputted again to MoveJ.

▷ MoveITPL (Move Interpolation) :

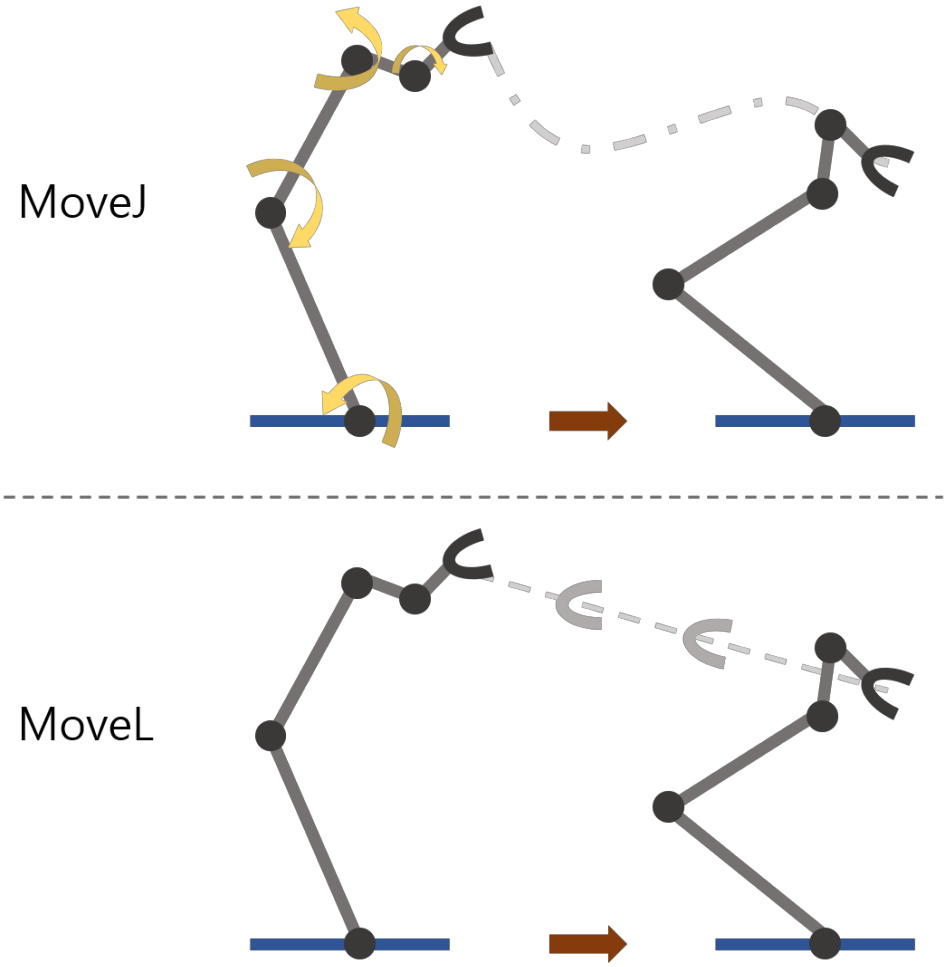
Starting from the starting point (the current position), move smoothly between the points without any stops using the Move L method.

MoveLB or PB blends across (blend) each waypoint, but MoveITPL moves along the trajectory exactly past each waypoint. So, there is no separate blend setting.

MoveITPL has two modes. Constant mode is to move the tool orientation while maintaining the starting point value. Intended mode is to change the orientation of each tool.

The speed can be set separately for each intermediate waypoint.

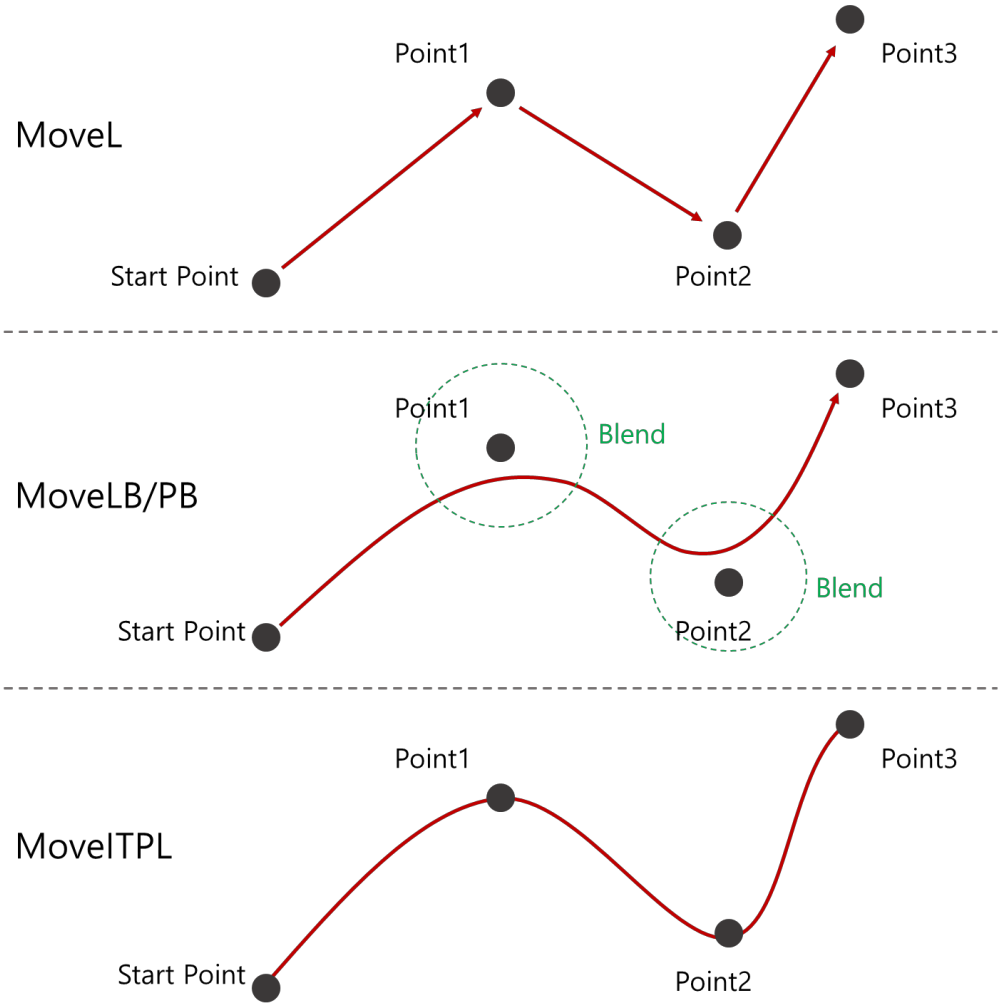
■ Difference between **MoveJ** and **MoveL**



MoveJ does not consider the movement trajectory of the terminal (TCP). It is an operation that only uses the joint angle information of the starting point and the joint angle of the target point. The driving speed of other joints are adjusted to the joints that require the most driving time.

MoveL is a mode that uses inverse kinematics to move the trajectory of the terminal (TCP) linearly from the starting point to the target point. 6 Cartesian coordinate values (x, y, z, Rx, Ry, Rz) are the inputs for the target point value.

■ Difference between **MoveL**, **MoveLB/PB**, and **MoveITPL**



MoveL moves in a straight, linear path between the start and destination points. The arm will arrive at each sequential arrival Point, stop, and then continue to the next Point.

MoveLB/PB starts at the initial Point, uses each intermediate Point as a waypoint, then stops at the final Point. The arm will not stop at the specified waypoints. Instead, it will arc around each point according to the blend distance, and then continue without stopping.

MoveITPL, the points other than the arrival point move to the waypoint, creating a trajectory that passes exactly through the waypoint. The trajectory is created without stopping and a separate speed setting is possible for each waypoint.

Warning

Warning:

1) The five linear motion commands (MoveL, MoveLB, MovePB, MoveJL, MoveITPL) move the robot using inverse kinematics calculations. Therefore, movement may be limited in singularity positions where inverse kinematics calculations are not possible.

2) Certain joints may move faster or be restricted in motion while in the dead zone of the robot. Further information about dead zones can be found in Section 1.7.