Motion Coordinate System

1756-HYD02, 1756-M02AE, 1756-M02AS, 1756-M03SE, 1756-M08SE, 1756-M16SE, 1768-M04SE
Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards. Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.

| WARNING: | Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss. |
| ATTENTION: | Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence. |
| IMPORTANT | Identifies information that is critical for successful application and understanding of the product. |

Labels may also be on or inside the equipment to provide specific precautions.

| SHOCK HAZARD: | Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present. |
| BURN HAZARD: | Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures. |
| ARC FLASH HAZARD: | Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE). |
Summary of changes

This manual contains new and updated information. Use these reference tables to locate new or changed information.

Grammatical and editorial style changes are not included in this summary.

Global changes

This table contains a list of topics changed in this version, the reason for the change, and a link to the topic that contains the changed information.

<table>
<thead>
<tr>
<th>Topic Name</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the SCARA Independent J1J2J3J6 Coordinate System on page 67</td>
<td>Added section to configure a SCARA Independent J1J2J3J6 Coordinate System.</td>
</tr>
</tbody>
</table>
# Table of Contents

**Summary of changes**
- Before you begin ................................................................. 9  
- Sample projects ................................................................. 9  
- Additional resources ......................................................... 10

**Preface**

**Create and configure a coordinate system**

**Chapter 1**
- Create a Coordinate System .............................................. 11  
- Coordinate System Properties dialog box ....................... 13  
- Edit Coordinate System properties .................................. 13  
  - Geometry tab .................................................................. 14

**Chapter 2**
- Program coordinate system with no orientation ................ 17  
- Blended moves and termination types with MCLM or MCCM .. 17  
  - Example ladder diagram for blended instructions .......... 18  
  - Bit States at transition points of blended move by using actual tolerance or no settle ........................................ 19  
  - Bit States at transition points of blended move by using no decel .. 20  
  - Bit states at transition points of blended move by using command tolerance .................................................. 21  
  - Bit states at transition points of blended move by using follow contour velocity constrained or unconstrained .......... 22  
  - Choose a termination type ............................................. 22

**Chapter 3**
- Configure an Articulated Independent robot ......................... 33  
- Establish reference frame for an articulated independent robot .. 33  
- Methods to establish a reference frame for an articulated independent robot ............................................................... 35  
  - Method 1 - Establish a reference frame ......................... 36  
  - Method 2 - Establish a reference frame using a MRP instruction ... 36  
- Configuration parameters for Articulated Independent robot ... 37  
- Link lengths for Articulated Independent robot .................. 38  
- Base Offsets ...................................................................... 39  
- End-Effector Offsets for Articulated Independent robot .......... 39  
- Configure Delta robot geometries ..................................... 40  
- Configure a Delta Three-dimensional robot ....................... 41  
- Establish the reference frame for a Delta Three-dimensional robot ................................................................. 41  
- Calibrate a Delta Three-dimensional robot ........................ 42  
- Alternate method for calibrating a Delta Three-dimensional robot 43
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure Zero Angle Orientations for Delta Three-dimensional robot</td>
<td>43</td>
</tr>
<tr>
<td>Identify the work envelope for a Delta Three-dimensional robot</td>
<td>44</td>
</tr>
<tr>
<td>Define configuration parameters for a Delta Three-dimensional robot</td>
<td>45</td>
</tr>
<tr>
<td>Configure a Delta Two-dimensional robot</td>
<td>46</td>
</tr>
<tr>
<td>Establish the reference frame for a Delta Two-dimensional robot</td>
<td>47</td>
</tr>
<tr>
<td>Calibrate a Delta Two-dimensional robot</td>
<td>47</td>
</tr>
<tr>
<td>Identify the work envelope for a Delta Two-Dimensional robot</td>
<td>48</td>
</tr>
<tr>
<td>Define configuration parameters for a Delta Two-dimensional robot</td>
<td>48</td>
</tr>
<tr>
<td>Configure a SCARA Delta robot</td>
<td>49</td>
</tr>
<tr>
<td>Establish the reference frame for a SCARA Delta robot</td>
<td>49</td>
</tr>
<tr>
<td>Calibrate a SCARA Delta robot</td>
<td>50</td>
</tr>
<tr>
<td>Identify the work envelope for a SCARA Delta robot</td>
<td>50</td>
</tr>
<tr>
<td>Define configuration parameters for a SCARA Delta robot</td>
<td>50</td>
</tr>
<tr>
<td>Configure a Delta robot with a Negative X1b offset</td>
<td>51</td>
</tr>
<tr>
<td>Arm solutions</td>
<td>52</td>
</tr>
<tr>
<td>Left-arm and right-arm solutions for two-axes robots</td>
<td>53</td>
</tr>
<tr>
<td>Solution mirroring for three-dimensional robots</td>
<td>53</td>
</tr>
<tr>
<td>Change the robot arm solution</td>
<td>54</td>
</tr>
<tr>
<td>Plan for singularity</td>
<td>55</td>
</tr>
<tr>
<td>Encounter a no-solution position</td>
<td>55</td>
</tr>
<tr>
<td>Error conditions</td>
<td>55</td>
</tr>
<tr>
<td>Configure an Articulated Dependent robot</td>
<td>56</td>
</tr>
<tr>
<td>Reference frame for Articulated Dependent robots</td>
<td>56</td>
</tr>
<tr>
<td>Methods to establish a reference frame for an articulated independent robot</td>
<td>58</td>
</tr>
<tr>
<td>Method 1 - Establish a reference frame using zero angle orientation</td>
<td>59</td>
</tr>
<tr>
<td>Method 2 - Establish a reference frame</td>
<td>60</td>
</tr>
<tr>
<td>Work envelope for articulated independent robot</td>
<td>60</td>
</tr>
<tr>
<td>Configuration parameters for Articulated Dependent robot</td>
<td>61</td>
</tr>
<tr>
<td>Link lengths for Articulated Dependent robot</td>
<td>61</td>
</tr>
<tr>
<td>Base offsets for Articulated Independent robot</td>
<td>62</td>
</tr>
<tr>
<td>Configure a Cartesian Gantry robot</td>
<td>63</td>
</tr>
<tr>
<td>Introduction</td>
<td>63</td>
</tr>
<tr>
<td>Establish the reference frame for a Cartesian Gantry robot</td>
<td>63</td>
</tr>
<tr>
<td>Identify the work envelope for a Cartesian Gantry robot</td>
<td>64</td>
</tr>
<tr>
<td>Define configuration parameters for a Cartesian Gantry robot</td>
<td>64</td>
</tr>
</tbody>
</table>
# Chapter 4

Configure a Cartesian H-bot robot .......................................................... 65
   Establish the reference frame for a Cartesian H-bot ....................... 66
   Identify the work envelope for a Cartesian H-bot ............................ 66
   Define configuration parameters for a Cartesian H-bot robot ....... 66

Configure a SCARA Independent Robot ............................................ 66

Configure the SCARA Independent J1J2J3J6 Coordinate System ........... 67

Configuration Parameters for the Robot ........................................... 67
   Link Lengths for SCARA Independent J1J2J3J6 Robot .................. 68
   Zero Angle Orientations for SCARA Independent J1J2J3J6 Robot ... 68
   Ball Screw Coupling for SCARA Independent J1J2J3J6 Robot ....... 70

Robot Configuration for SCARA Independent J1J2J3J6 Robot .......... 74
   Robot Configuration in MCPM instruction .................................. 75
   Robot Configuration in MCTPO instruction ............................... 75
   Robot Configuration Example .................................................. 76

Identify the Work Envelope for the Robot ........................................ 78

Maximum Joint Limits condition for SCARA Independent J1J2J3J6 robot ................................................................. 79
   Configure the Joint Limits ....................................................... 79

Work and Tool Frame offset limits for SCARA Independent J1J2J3J6 robot ................................................................. 79

Sample Project for SCARA Independent J1J2J3J6 Robot ............... 80

## Appendix A

Coordinate system attributes ............................................................ 81

## Appendix B

Solution mirroring for three-dimensional robots ............................. 87

Change arm solution ................................................................. 88
   Change arm solution example .................................................. 88

Singularity ...................................................................................... 88

Index
This manual provides information on how to configure various coordinated motion applications. Use the following table to choose a motion coordinated instruction. Information about the coordinate instructions can be found in the Logix5000™ Controllers Motion Instruction Reference Manual, publication MOTION-RM002.

<table>
<thead>
<tr>
<th>If you want to</th>
<th>Use this instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiate a single or multi-dimensional linear coordinated move for the specified axes within a Cartesian coordinate system.</td>
<td>Motion Coordinated Linear Move (MCLM)</td>
</tr>
<tr>
<td>Initiate a two- or three-dimensional circular coordinated move for the specified axes within a Cartesian coordinate system.</td>
<td>Motion Coordinated Circular Move (MCCM)</td>
</tr>
<tr>
<td>Initiate a change in path dynamics for coordinate motion active on the specified coordinate system.</td>
<td>Motion Coordinated Change Dynamics (MCCD)</td>
</tr>
<tr>
<td>Stop the axes of a coordinate system or cancel a transform.</td>
<td>Motion Coordinated Stop (MCS)</td>
</tr>
<tr>
<td>Initiate a controlled shutdown of all of the axes of the specified coordinate system.</td>
<td>Motion Coordinated Shutdown (MCSD)</td>
</tr>
<tr>
<td>Start a transform that links two coordinate systems together.</td>
<td>Motion Coordinated Transform (MCT)</td>
</tr>
<tr>
<td>Start a transform that links to coordinate systems together. The MCTO instruction incorporates translation and orientation in its position transformation.</td>
<td>Motion Coordinated Transform with Orientation (MCTO)</td>
</tr>
<tr>
<td>Calculate the position of one coordinate system with respect to another coordinate system.</td>
<td>Motion Calculate Transform Position (MCTP)</td>
</tr>
<tr>
<td>Calculate the position of a point in one coordinate system to the equivalent point in a second coordinate system.</td>
<td>Motion Coordinated Transform Position with Orientation (MCTPO)</td>
</tr>
<tr>
<td>Initiate a reset of all of the axes of the specified coordinate system from the shutdown state to the axis ready state and clear the axis faults.</td>
<td>Motion Coordinated Shutdown Reset (MCSR)</td>
</tr>
<tr>
<td>Start a single or multi-dimensional linear coordinated path move (CP) for the specified axes within a Cartesian coordinate system.</td>
<td>Motion Coordinated Path Move (MCPM)</td>
</tr>
</tbody>
</table>

(1) Instruction cannot be used with SoftLogix™ controllers.

(2) Instruction only available for Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Before you begin

This manual is a redesigned manual from publication LOGIX-UM002. A companion manual is available called the SERCOS and Analog Motion Configuration and Start-Up User Manual, publication MOTION-UM001. For CIP motion configuration information, see the CIP Motion Configuration and Startup User Manual, publication MOTION-UM003. If you have any comments or suggestions, please see the back cover of this manual.

Sample projects

The Rockwell Automation sample project’s default location is:

c:\Users\Public\Public Documents\Studio 5000\Sample\ENU\<current_release>\Rockwell Automation

There is a PDF file name Vendor Sample Projects that explains how to work with the sample projects. Free sample code is available at http://samplecode.rockwellautomation.com/.

The Vendor Sample Projects.pdf default location is:
Additional resources

These documents contain additional information concerning related Rockwell Automation products. You can view or download publications at [http://literature.rockwellautomation.com](http://literature.rockwellautomation.com).

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sercos and Analog Motion Configuration and Startup User Manual,</td>
<td>Describes how to configure a motion application and to start up your motion solution by using Logix5000 motion modules.</td>
</tr>
<tr>
<td>publication MOTION-UM001</td>
<td></td>
</tr>
<tr>
<td>&gt;5k&lt; Controllers Motion Instructions Reference Manual, publication</td>
<td>Provides a programmer with details about motion instructions for a Logix-based controller.</td>
</tr>
<tr>
<td>MOTION-RM002</td>
<td></td>
</tr>
<tr>
<td>Integrated Motion on the Ethernet/IP Network: Configuration and Startup</td>
<td>Describes how to configure an integrated motion application and to start up your motion solution by using Studio 5000 Logix Designer® application.</td>
</tr>
<tr>
<td>User Manual, publication MOTION-UM003</td>
<td></td>
</tr>
<tr>
<td>Logix5000 Controllers Common Procedures, publication 1756-PM001</td>
<td>Provides detailed and comprehensive information about how to program a Logix5000 controller.</td>
</tr>
<tr>
<td>Logix5000 Controllers General Instructions Reference Manual, publication</td>
<td>Provides a programmer with details about general instructions for a Logix-based controller.</td>
</tr>
<tr>
<td>1756-RM003</td>
<td></td>
</tr>
<tr>
<td>Logix5000 Controllers Process and Drives Instructions Reference</td>
<td>Provides a programmer with details about process and drives instructions for a Logix-based controller.</td>
</tr>
<tr>
<td>Manual, publication 1756-RM006</td>
<td></td>
</tr>
<tr>
<td>ControlLogix System User Manual, publication 1756-UM001</td>
<td>Describes the necessary tasks to install, configure, program, and operate a ControlLogix® system.</td>
</tr>
<tr>
<td>ControlLogix 5580 and GuardLogix 5580 Controllers User Manual, publication</td>
<td>Provides complete information on how to install, configure, select I/O modules, manage communication, develop applications, and troubleshoot the ControlLogix 5580 and GuardLogix 5580 controllers.</td>
</tr>
<tr>
<td>1756-UM543</td>
<td></td>
</tr>
<tr>
<td>CompactLogix 5370 Controllers User Manual, publication 1756-UM021</td>
<td>Describes the necessary tasks to install, configure, program, and operate a CompactLogix™ system.</td>
</tr>
<tr>
<td>GuardLogix Controllers User Manual, publication 1756-UM020</td>
<td>Describes the GuardLogix®-specific procedures you use to configure, operate, and troubleshoot the controller.</td>
</tr>
<tr>
<td>GuardLogix 5570 and Compact GuardLogix 5370 Controller Systems Safety</td>
<td>Contains detailed requirements for achieving and maintaining SIL 3/PL2 with the GuardLogix 5570 or CompactLogix 5370 controller safety system, using the Studio 5000 Logix Designer application.</td>
</tr>
<tr>
<td>Reference Manual, publication 1756-RM099</td>
<td></td>
</tr>
<tr>
<td>GuardLogix 5580 and Compact GuardLogix 5380 Controller Systems Safety</td>
<td>Provides information on safety application requirements for GuardLogix 5580 and Compact GuardLogix 5380 controllers in Studio 5000 Logix Designer® applications.</td>
</tr>
<tr>
<td>Reference Manual, publication 1756-RM012</td>
<td></td>
</tr>
<tr>
<td>Industrial Automation Wiring and Grounding Guidelines, publication 1770-</td>
<td>Provides general guidelines for installing a Rockwell Automation industrial system.</td>
</tr>
<tr>
<td>6.1</td>
<td></td>
</tr>
</tbody>
</table>
Create and configure a coordinate system

Create a Coordinate System

Use the Coordinate System tag to set the attribute values used by the Multi-Axis Coordinated Motion instructions in motion applications. Create the Coordinate System tag before executing any of the Multi-Axis Coordinated Motion instructions.

The Coordinate System tag:

- Defines the COORDINATE_SYSTEM data type
- Associates the Coordinate System to a Motion Group
- Associates the axes to the Coordinate System
- Sets the dimension
- Defines the values used by the operands of the Multi-Axis Motion Instructions

Configuring the Coordinate System tag defines the values for Coordination Units, Maximum Speed, Maximum Acceleration, Maximum Deceleration, Actual Position Tolerance, and Command Position Tolerance.

To create a coordinate system:

1. In the Controller Organizer, right-click the motion group and select New Coordinate System.
The New Tag dialog box opens.

2. In Name, enter the name of the coordinate system.
3. [optional] In Description, type a description of the coordinate system.
4. In Type, select the type of tag to create. For a coordinate system, the only valid choices are:
   - Base - Refers to a normal tag and is the default
   - Alias - Refers to a tag that references another tag with the same definition
5. In Data Type, select COORDINATE_SYSTEM.
6. In External Access, select whether the tag has None, Read/Write, or Read Only access from external applications such as HMIs.
7. Select Constant to prevent executing logic from writing values to the tag. Refer to the online help for more information about the Constant check box.
8. Select Open COORDINATE_SYSTEM to open the Coordinate System Wizard after creating the tag.
   Once the tag is created, double-click the coordinate system to open the Coordinate System Properties dialog box to edit the coordinate system tag.
9. Select Create to create the tag.
See also

Coordinate System Properties dialog box on page 13

Use the Coordinate System Wizard or Coordinate System Properties dialog box to configure the Coordinate System tag. The dialog box contains tabs for configuring different facets of the Coordinate System.

<table>
<thead>
<tr>
<th>Wizard/Coordinate System Properties tab</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>The General tab is used to:</td>
</tr>
<tr>
<td></td>
<td>• Associate the tag to a Motion Group.</td>
</tr>
<tr>
<td></td>
<td>• Select the coordinate system type.</td>
</tr>
<tr>
<td></td>
<td>• Select the coordinate definition for the geometry type.</td>
</tr>
<tr>
<td></td>
<td>• If applicable, specify the number of dimensions and transform dimensions for the geometry type.</td>
</tr>
<tr>
<td></td>
<td>• Enter the associated axis information.</td>
</tr>
<tr>
<td></td>
<td>• Select whether to update Actual Position values of the coordinate system automatically during operation.</td>
</tr>
<tr>
<td>Geometry</td>
<td>The Geometry tab configures key attributes related to non-Cartesian geometry and shows the bitmap of the associated geometry.</td>
</tr>
<tr>
<td>Offset</td>
<td>The Offset tab configures the offsets for the base and end effector. This tab shows the bitmaps for the offsets related to the geometry.</td>
</tr>
<tr>
<td>Units</td>
<td>The Units tab defines the Coordination Units and the Conversion Ratios.</td>
</tr>
<tr>
<td>Dynamics</td>
<td>The Dynamics tab configures the Vector, Actual and Command Position Tolerance, and Orientation values for a Cartesian coordinate system.</td>
</tr>
<tr>
<td>Joints</td>
<td>The Joints tab defines the Joints Conversion ratios.</td>
</tr>
<tr>
<td>Motion Planner</td>
<td>The Motion Planner tab enables or disables Master Delay Compensation or Master Position Filter.</td>
</tr>
<tr>
<td>Tag</td>
<td>The Tag tab is used to rename the tag, edit the description, and review the Tag Type, Data Type, and Scope information.</td>
</tr>
</tbody>
</table>

Edit Coordinate System properties

Use the Coordinate System Properties dialog box to modify an existing Coordinate System or configure the Coordinate System.

To edit the Coordinate System properties:

1. In the Controller Organizer, expand the Motion Group folder, and double-click the Coordinate System, or right-click the Coordinate System and select Properties.
2. Use the tabs in the Coordinate System Properties dialog box to make the appropriate changes. An asterisk appears on the tab to indicate that changes have been made but not implemented.
3. Click Apply to save the changes. To exit without saving any changes, click Cancel.

See also

Coordinate System Properties dialog box on page 13
Geometry tab

The Geometry tab of the Coordinate System Properties is where you can specify the link lengths and zero angle orientation values for articulated robotic arms.

The graphic displayed on this tab shows a typical representation of the type of coordinate system you selected on the General tab. Your robot should look similar to the one shown in the graphic, but may be somewhat different depending on your application.

Link Lengths box

The Link Lengths box displays boxes to let you specify a value for the length of each link in an articulated robotic arm (coordinate system). The measurement units for the articulated coordinate system are defined by the measurement units configured for the affiliated Cartesian coordinate system. The two coordinate systems are linked or affiliated with each other by an MCT instruction.

When specifying the link length values, be sure that the values are calculated by using the same measurement units as the linked Cartesian coordinate system. For example, if the manufacturer specifies the robot link lengths by using millimeter units and you want to configure the robot by using inches, then you must convert the millimeter link measurements to inches and enter the values in the appropriate Link Length boxes.

IMPORTANT Be sure that the link lengths specified for an articulated coordinate system are in the same measurement units as the affiliated Cartesian coordinate system. Your system will not work properly if you are using different measurement units.

The number of boxes available for configuration in the Link Lengths box is determined by values entered on the General tab for the type of coordinate system, total coordinate system dimensions, and transform dimensions. The link identifiers are L1 and L2 in the corresponding graphic. These boxes are not configurable for a Cartesian coordinate system.
Zero Angle Orientations box

The Zero Angle Orientation box is the rotational offset of the individual joint axes. If applicable, enter the offset value in degrees for each joint axis. The number of available boxes is determined by the coordinate dimension value entered on the General tab. The angle identifiers are Z1, Z2, and Z3 in the corresponding graphic.
Cartesian coordinate system

Use this information to configure a Cartesian coordinate system.

See also

Program coordinate system with no orientation on page 17

Use these multi-axis coordinated motion instructions to perform linear and circular moves in single and multidimensional spaces. A Cartesian coordinate system with no orientation in the Logix Designer application can include one, two, or three axes.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion Coordinated Linear Move (MCLM)</td>
<td>Use the MCLM instruction to start a single or multidimensional linear coordinated move for the specified axes within a Cartesian coordinate system.</td>
</tr>
<tr>
<td>Motion Coordinated Circular Move (MCCM)</td>
<td>Use the MCCM instruction to initiate a two or three-dimensional circular coordinated move for the specified axes within a Cartesian coordinate system.</td>
</tr>
<tr>
<td>Motion Coordinated Transform (MCT)</td>
<td>Use the MCT instruction to start a transform that links two coordinate systems together.</td>
</tr>
<tr>
<td>Motion Calculate Transform Position (MCTP)</td>
<td>Use the MCTP instruction to calculate the position of a point in one coordinate system to the equivalent point in a second coordinate system.</td>
</tr>
</tbody>
</table>

See the Logix 5000 Motion Controllers Instructions Reference Manual, publication MOTION-RM002, for more information about the MCLM, MCCM, MCT, and MCTP instructions.

To blend two MCLM or MCCM instructions, start the first one and queue the second one. The tag for the coordinate system gives two bits for queuing instructions.

- MovePendingStatus
- MovePendingQueueFullStatus

For example, the following ladder diagram uses coordinate system cs1 to blend Move1 into Move2.

See also

Example ladder diagram for blended instructions on page 18
If Step = 1, then:

Move1 starts and moves the axes to a position of 5, 0.

and once Move1 is in process, and there is room to queue another move, then:

Step = 2.

If Step = 2, then:

Move1 is already happening.

Move2 goes into the queue and waits for Move1 to complete.

When Move1 is complete:

Move2 moves the axes to a position of 10, 5.

And once Move2 is in process and there is room in the queue:

Step = 3.

When an instruction completes, it is removed from the queue and there is space for another instruction to enter the queue. Both bits always have the same value because you can queue only one pending instruction at a time. If the application requires several instructions to be executed in sequence, the bits are set by using these parameters.

<table>
<thead>
<tr>
<th>When</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>One instruction is active and a second instruction is pending in the queue</td>
<td>• MovePendingStatus bit = 1</td>
</tr>
<tr>
<td></td>
<td>• MovePendingQueueFullStatus bit = 1</td>
</tr>
<tr>
<td></td>
<td>• You cannot queue another instruction</td>
</tr>
</tbody>
</table>
When an active instruction completes and leaves the queue:

- MovePendingStatus bit = 0
- MovePendingQueueFullStatus bit = 0
- You can queue another instruction

The termination type operand for the MCLM or MCCM instruction specifies how the currently executing move gets terminated. These illustrations show the states of instruction bits and coordinate system bits that get affected at various transition points (TP).

The termination types are:

- 0 - Actual tolerance
- 1 - No Settle
- 2 - Command Tolerance
- 3 - No Decel
- 4 - Follow Contour Velocity Constrained
- 5 - Follow Contour Velocity Unconstrained
- 6 - Command Tolerance Programmed

See also

Termination types on page 22

Bit States at transition points of blended move by using actual tolerance or no settle

This topic lists the bit states at transition points of Blended Move by using Actual Tolerance or No Settle.

This table shows the bit status at the various transition points shown in the preceding graph with termination type of Actual Tolerance or No Settle.
### Bit States at transition points of blended move by using no decel

This lists the bit states at transition points of blended move by using no decel.

<table>
<thead>
<tr>
<th>Bit</th>
<th>TP1</th>
<th>TP2</th>
<th>TP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move1.DN</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Move1.IP</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Move1.AC</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Move1.PC</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Move2.DN</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Move2.IP</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>Move2.AC</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>Move2.PC</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>cs1.MoveTransitionStatus</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>cs1.MovePendingStatus</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>cs1.MovePendingQueueFullStatus</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

This table shows the bit status at the various transition points shown in the preceding graph with termination type of No Decel. For No Decel termination type distance-to-go for transition point TP2 is equal to deceleration distance for the Move1 instruction. If Move 1 and Move 2 are collinear, then Move1.PC will be true at TP3, which is the programmed end-point of first move.
Bit states at transition points of blended move by using command tolerance

This lists the bit states at transition points of Blended Move by using Command Tolerance.

This table shows the bit status at the various transition points shown in the preceding graph with termination type of Command Tolerance. For Command Tolerance termination type distance-to-go for transition point TP2 is equal to Command Tolerance for the coordinate system cs1.

<table>
<thead>
<tr>
<th>Bit</th>
<th>TP1</th>
<th>TP2</th>
<th>TP3</th>
<th>TP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move1.DN</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Move1.IP</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Move1.AC</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Move1.PC</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Move2.DN</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Move2.IP</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Move2.AC</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Move2.PC</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>cs1.MoveTransitionStatus</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>cs1.MovePendingStatus</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>cs1.MovePendingQueueFullStatus</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

linear → linear move
This lists the bit states at transition points of blended move by using follow contour velocity constrained or unconstrained.

This table shows the bits status at the transition points.

<table>
<thead>
<tr>
<th>Bit</th>
<th>TP1</th>
<th>TP2</th>
<th>TP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move1.DN</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Move1.IP</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Move1.AC</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Move1.PC</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Move2.DN</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Move2.IP</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>Move2.AC</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>Move2.PC</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>cs1.MoveTransitionStatus</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>cs1.MovePendingStatus</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>cs1.MovePendingQueueFullStatus</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Choose a termination type

The termination type determines when the instruction is complete. It also determines how the instruction blends its path into the queued MCLM or MCCM instruction, if there is one.
## To choose a termination type:

<table>
<thead>
<tr>
<th>If you want the axes to (vector speeds)</th>
<th>And you want the instruction to complete when</th>
<th>Then use this Termination Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>stop between moves.</td>
<td>The following occurs:</td>
<td>0 - Actual Tolerance</td>
</tr>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
<td>• Command position equals target position.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The vector distance between the target and actual positions is less than or equal to the Actual Position Tolerance of the Coordinate System.</td>
<td></td>
</tr>
<tr>
<td>keep the speed constant except between moves.</td>
<td>The command position equals the target position.</td>
<td>1 - No Settle</td>
</tr>
<tr>
<td><img src="image2.png" alt="Diagram" /></td>
<td>1 - No Settle</td>
<td></td>
</tr>
<tr>
<td>transition into or out of a circle without stopping.</td>
<td>The axes get to the point at which they must decelerate at the deceleration rate.</td>
<td>2 - Command Tolerance</td>
</tr>
<tr>
<td><img src="image3.png" alt="Diagram" /></td>
<td>2 - Command Tolerance</td>
<td></td>
</tr>
<tr>
<td>accelerate or decelerate across multiple moves.</td>
<td>4 - Follow Contour Velocity Constrained</td>
<td>3 - No Decel</td>
</tr>
<tr>
<td><img src="image4.png" alt="Diagram" /></td>
<td>4 - Follow Contour Velocity Constrained</td>
<td></td>
</tr>
<tr>
<td>use a specified Command Tolerance</td>
<td>The command position gets within the Command Position Tolerance of the coordinate system.</td>
<td>5 - Follow Contour Velocity Unconstrained</td>
</tr>
<tr>
<td><img src="image5.png" alt="Diagram" /></td>
<td>5 - Follow Contour Velocity Unconstrained</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6 - Command Tolerance Programmed</td>
<td></td>
</tr>
</tbody>
</table>
To make sure that this is the right choice for you:

- Review the tables below.

<table>
<thead>
<tr>
<th>Termination Type</th>
<th>Example Path</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0 - Actual Tolerance | ![Graph](image1.png) | The instruction stays active until both of these happen:  
  - Command position equals target position.  
  - The vector distance between the target and actual positions is less than or equal to the Actual Position Tolerance of the coordinate system.  
  At that point, the instruction is complete and a queued MCLM or MCCM instruction can start.  
  **Important:** Make sure that you set the Actual Tolerance to a value that your axes can reach. Otherwise the instruction stays in process. |
| 1 - No Settle | ![Graph](image2.png) | The instruction stays active until the command position equals the target position. At that point, the instruction is complete and a queued MCLM or MCCM instruction can start. |
| 2, 6 - Command Tolerance | ![Graph](image3.png) | The instruction stays active until the command position gets within the Command Tolerance of the Coordinate System. At that point, the instruction is complete and a queued MCLM or MCCM instruction can start. If you don't have a queued MCLM or MCCM instruction, the axes stop at the target position. |

**The Logix Designer application compares**

<table>
<thead>
<tr>
<th>To the</th>
<th>And uses the</th>
<th>For the</th>
</tr>
</thead>
<tbody>
<tr>
<td>100% of the configured length of the first instruction using a Command Tolerance termination type</td>
<td>configured Command Tolerance for the Coordinate System</td>
<td>shorter of the two lengths command Tolerance length used for the <strong>first</strong> instruction</td>
</tr>
<tr>
<td>100% of the configured length of the last move instruction using a Command Tolerance termination type</td>
<td>configured Command Tolerance for the Coordinate System</td>
<td>shorter of the two lengths command Tolerance length used for the <strong>next to last</strong> instruction</td>
</tr>
<tr>
<td>50% of each of the lengths of all other move instructions</td>
<td>configured Command Tolerance for the Coordinate System</td>
<td>shorter of the two lengths command Tolerance length used for <strong>each individual</strong> instruction</td>
</tr>
<tr>
<td>Termination Type</td>
<td>Example Path</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 3 - No Decel                     | ![Diagram](image1.png)  | The instruction stays active until the axes get to the deceleration point. At that point, the instruction is complete and a queued MCLM or MCCM instruction can start.  
                        | ![Diagram](image2.png)  | • The deceleration point depends on whether you use a trapezoidal or S-curve profile.  
                        | ![Diagram](image3.png)  | • If you don’t have a queued MCLM or MCCM instruction, the axes stop at the target position.  
                                                                                                                                                                                                                                                                                                                                                     |
| 4 - Follow Contour Velocity      | ![Diagram](image4.png)  | The instruction stays active until the axes get to the target position. At that point, the instruction is complete and a queued MCLM or MCCM instruction can start.  
                        | ![Diagram](image5.png)  | • This termination type works best with tangential transitions. For example, use it to go from a line to a circle, a circle to a line, or a circle to a circle.  
                        | ![Diagram](image6.png)  | • The axes follow the path.  
                        | ![Diagram](image7.png)  | • The length of the move determines the maximum speed of the axes. If the moves are long enough, the axes will not decelerate between moves. If the moves are too short, the axes decelerate between moves.  
                                                                                                                                                                                                                                                                                                                                                     |
| 5 - Follow Contour Velocity      | ![Diagram](image8.png)  | This termination type is similar to the contour velocity constrained. It has these differences:  
                        | ![Diagram](image9.png)  | • Use this termination type to get a triangular velocity profile across several moves. This reduces jerk.  
                        | ![Diagram](image10.png) | • To avoid position overshoot at the end of the last move, you must calculate the deceleration speed at each transition point during the deceleration-half of the profile.  
                        | ![Diagram](image11.png) | • You must also calculate the starting speed for each move in the deceleration half of the profile.  
                                                                                                                                                                                                                                                                                                                                                     |

**Important Considerations**

If you stop a move (that is, using an MCS or by changing the speed to zero with an MCCD) during a blend and then resume the move (that is, by reprogramming the move or by using another MCCD), it will deviate from the path that you would have seen if the move had not been stopped and resumed. The same phenomenon can occur if the move is within the decel point of the start of the blend. In either case, the deviation will most likely be a slight deviation.
**Velocity Profiles for Collinear Moves**

Collinear moves are those that lie on the same line in space. Their direction can be the same or opposite. The velocity profiles for collinear moves can be complex. This section provides you with examples and illustrations to help you understand the velocity profiles for collinear moves programmed with MCLM instructions.

**Velocity Profiles for Collinear Moves with Termination Type 2 or 6**

This illustration shows the velocity profile of two collinear moves using a Command Tolerance (2) termination type. The second MCLM instruction has a lower velocity than the first MCLM instruction. When the first MCLM instruction reaches its Command Tolerance point, the move is over and the .PC bit is set.

**Velocity Profile of Two Collinear Moves When the Second Move has a Lower Velocity than the First Move and Termination Type 2 or 6 is Used**

This illustration shows the velocity profile of two collinear moves using a Command Tolerance (2) termination type. The second MCLM instruction has a higher velocity than the first MCLM instruction. When the first MCLM instruction reaches its Command Tolerance point, the move is over and the .PC bit is set.
Velocity Profile of Two Collinear Moves When the Second Move has a Higher Velocity than the First Move and Termination Type 2 or 6 is Used

Velocity Profiles for Collinear Moves with Termination Types 3, 4, or 5

This illustration shows a velocity profile of two collinear moves. The second MCLM instruction has a lower velocity than the first MCLM instruction and one of these termination types are used:

- No Decel (3)
- Follow Contour Velocity Constrained (4)
- Follow Contour Velocity Unconstrained (5)

When the first MCLM instruction reaches the deceleration point, it decelerates to the programmed velocity of the second move. The first move is over and the .PC bit is set.

Velocity Profile of Two Collinear Moves When the Second Move has a Lower Velocity than the First Move and Termination Type 3, 4, or 5 is Used
This illustration shows a velocity profile of two collinear moves. The second MCLM instruction has a **higher** velocity than the first MCLM instruction and one of these termination types are used:

- No Decel (3)
- Follow Contour Velocity Constrained (4)
- Follow Contour Velocity Unconstrained (5)

The .PC bit is set when the first move reaches its programmed endpoint.

**Velocity Profile of Two Collinear Moves When the Second Move has a Higher Velocity than the First Move and Termination Type 3, 4, or 5 is Used**

![Diagram showing velocity profile of two collinear moves](image)

**Symmetric Profiles**

Profile paths are symmetric for all motion profiles.

Programming the velocity, acceleration, and deceleration values symmetrically in the forward and reverse directions generates the same path from point A to point C in the forward direction, as from point C to point A in the reverse direction.

While this concept is most easily shown in a two-instruction sequence, it applies to instruction sequences of any length provided that they are programmed symmetrically.
Refer to this Example of a Symmetric Profile for more details.

- MCLM 1 (point A to point B) is followed by MCLM 2 (point B to point C).
- MCLM 3 (point C to point B) is followed by MCLM 4 (point B to point A).
- The acceleration of MCLM 1 must be equal to the deceleration of MCLM 4.
- The deceleration of MCLM 1 must be equal to the acceleration a MCLM 4.
- The acceleration of MCLM 2 must be equal to the deceleration of MCLM 3.
- The deceleration of MCLM 2 must be equal to the acceleration of MCLM 3.

MCLM 1 (Pos = [2,0], Accel = 1, Decel = 2)
MCLM 2 (Pos = [2,1], Accel = 3, Decel = 4)
MCLM 3 (Pos = [2,0], Accel = 4, Decel = 3)
MCLM 4 (Pos = [0,0], Accel = 2, Decel = 1)

IMPORTANT

We recommend that you terminate any sequence of moves by either Termination Type 0 or 1, that is, TT0 or TT1.

To guarantee that your trajectory is symmetric, you must terminate any sequence of moves by either Termination Types 0 or 1. You should also use a Termination Type of 0 or 1 at the Reversal Point of a profile that moves back on itself.

Using a TT2, TT3, TT4, TT5, or TT6 as the last move in a profile (or the reversal point) is safe. However, the resulting trajectory from A to B may not always be the same as that from B to A. Explicit termination of the sequence of moves helps the controller to optimize the velocity profile, reduce the CPU load, and guarantee a symmetric profile.
How To Get a Triangular Velocity Profile

If you want to program a pick and place action in four moves, minimize the Jerk rate, and use a triangular velocity profile.

Then, use termination type 5. The other termination types may not let you get to the speed you want.

<table>
<thead>
<tr>
<th>Termination Types 2, 3, 4, or 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>You want to get to this speed…</td>
</tr>
<tr>
<td>…but the axes have to decelerate before they get there.</td>
</tr>
</tbody>
</table>

The length of each move determines its maximum speed. As a result, the axes will not reach a speed that causes them to overshoot the target position during deceleration.
Blending Moves at Different Speeds

You can blend MCLM and MCCM instructions where the vector speed of the second instruction is different from the vector speed of the first instruction.

<table>
<thead>
<tr>
<th>If the next move is</th>
<th>And the Termination Type of the first move is</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slower</td>
<td>2 - Command Tolerance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 - No Decel</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 - Contour Velocity Constrained</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 - Contour Velocity Unconstrained</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6 - Command Tolerance Programmed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vector speed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Target position of first move</td>
<td></td>
</tr>
<tr>
<td>Faster</td>
<td>2 - Command Tolerance</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 - No Decel</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6 - Command Tolerance Programmed</td>
<td></td>
</tr>
</tbody>
</table>

4 - Contour Velocity Constrained
5 - Contour Velocity Unconstrained

![Diagram](Image)
Chapter 3

 Cartesian coordinate system examples

Configure an Articulated Independent robot

Use these guidelines when configuring an Articulated Independent robot.

**WARNING:** Before turning ON the Transform and/or establishing the reference frame, be sure to do the following for the joints of the target coordinate system.

- Set and enable the soft travel limits.
- Enable the hard travel limits.

Failure to do this can allow the robot to move outside of the work envelope causing machine damage and/or serious injury or death to personnel.

See also

- Establish reference frame for an Articulated Independent robot on page 33
- Methods to establish a reference frame for Articulated Independent robot on page 35
- Work envelope for Articulated Independent robot on page 60
- Define configuration parameters for Articulated Independent robot on page 37

Establish reference frame for an articulated independent robot

The reference frame is the Cartesian coordinate frame that defines the origin and the three primary axes (X1, X2, and X3). These axes measure the real Cartesian positions.

**WARNING:** Failure to properly establish the correct reference frame for your robot can cause the robotic arm to move to unexpected positions causing machine damage and/or injury or death to personnel.

The reference frame for an Articulated Independent robot is located at the base of the robot as shown in this figure.
Before establishing the Joint-to-Cartesian reference frame relationship, it is important to know some information about the Kinematic mathematical equations used in the Logix controllers. The equations are written as if the Articulated Independent robot joints are positioned as shown in the following illustration.

**Illustration 2 - Side view**

- +J1 is measured counterclockwise around the +X3 axis starting at an angle of J1=0 when L1 and L2 are both in the X1-X2 plane.
- +J2 is measured counterclockwise starting with J2=0 when L1 is parallel to X1-X2 plane.
- +J3 is measured counterclockwise with J3=0 when L2 is aligned with link L1.

When the robot is physically in this position, the Logix Designer application Actual Position tags for the axes must be:

- J1 = 0.
- J2 = 0.
- J3 = 0.
When the robot is physically in the above position, the Logix Designer application Actual Position tags for the axes must be:

- J1 = 0.
- J2 = 90.
- J3 = -90.

If the physical position and joint angle values of the robot cannot match those shown in the preceding illustrations, use one of the Alternate Methods for Establishing the Joint-to-Cartesian reference frame relationship.

**See also**

[Methods for establishing a reference frame for an articulated independent robot](#) on page 35

Use the following methods to establish a reference frame for the robot.

<table>
<thead>
<tr>
<th>For each:</th>
<th>Use one of these methods to establish the reference frame:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental axis</td>
<td>Each time the power for the robot is cycled.</td>
</tr>
<tr>
<td>Absolute axis</td>
<td>Only to establish absolute home.</td>
</tr>
</tbody>
</table>

- Method 1 - Establishes a Zero Angle Orientation and allows the configured travel limits and home position on the joint axes to remain operational. Use this method when operating the axes between the travel limits determined prior to programming a Motion Redefine Position (MRP) instruction and want these travel limits to stay operational.

- Method 2 - Uses a MRP instruction to redefine the axes position to align with the joint reference frame. This method may require the soft travel limits to be adjusted to the new reference frame.

**See also**

[Method 1 for an incremental axis](#) on page 59
Method 1 - Establish a reference frame

Each axis for the robot has the mechanical hard stop in each of the positive and negative directions. Manually move or press each axes of the robot against its associated mechanical hard stop and redefine it to the hard limit actual position provided by the robot manufacturer. J1 is the axis at the base of the robot that rotates around X3.

When the robot is moved so that Link1 is parallel to the X3 axis and Link2 is parallel to X1 axis as shown in Articulated Dependent, the Logix Designer application values for the Actual Position tags should be:

- \( J1 = 0 \).
- \( J2 = 90^\circ \).
- \( J3 = 0^\circ \).

If the Logix Designer application Actual Position tags do not show these values, configure the Zero Angle Orientation for the joint or joints that do not correspond.

<table>
<thead>
<tr>
<th>If the Logix Designer application read-out values are:</th>
<th>Set the Zero Angle Orientations on the Coordinate System Properties dialog box to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1 = 10</td>
<td>Z1 = -10</td>
</tr>
<tr>
<td>J2 = 80</td>
<td>Z2 = 10</td>
</tr>
<tr>
<td>J3 = 5</td>
<td>Z3 = -5</td>
</tr>
</tbody>
</table>

Setting the Zero Angle Orientations

Position the robot so that:

- L1 is parallel to the X3 axis.
- L2 is parallel to X1 axis.
Program a Motion Redefine Position (MRP) instruction for all three axes with the following values:

- \( J_1 = 0 \)
- \( J_2 = 90^\circ \)
- \( J_3 = -90^\circ \)

The Joint-to-Cartesian reference frame relationship is automatically established by the Logix controller after the Joint coordinate system parameters, which are the link lengths, base offsets, and end-effector offsets, are configured and the MCT instruction is enabled.

**See also**

[Method 1 - Establish a reference frame using zero angle orientation on page 35](#)

**Configuration parameters for Articulated Independent robot**

Configure the Logix Designer application to control robots with varying reach and payload capacities. The configuration parameter values for the robot include:

- Link lengths
- Base offset
- End effector offsets

The configuration parameter information is available from the robot manufacturer.

**IMPORTANT** Verify that the values for the Link Lengths, Base Offsets, and End-Effector Offsets are entered in the Coordinate System Properties dialog box using the same measurement units.

This example illustrates the typical configuration parameters for an Articulated Independent robot.
If the robot is two-dimensional, then X3b and X3e are X2b and X2e.

See also

Link lengths for Articulated Independent robot on page 38
Base offsets for Articulated Independent robot on page 62
End effector offsets for Articulated Independent robot on page 39

Link lengths for Articulated Independent robot

<table>
<thead>
<tr>
<th>For an articulated independent robot with</th>
<th>The length of</th>
<th>Is equal to the value of the distance between</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 dimensions</td>
<td>L1, L2</td>
<td>J1 and J2, J2 and the end-effector</td>
</tr>
<tr>
<td>3 dimensions</td>
<td>L1, L2</td>
<td>J2 and J3, J3 and the end-effector</td>
</tr>
</tbody>
</table>

Enter the link lengths on the Geometry tab in the Coordinate System Properties dialog box.

See also

Base offset for Articulated Independent robot on page 62
End effector offsets for Articulated Independent robot on page 39
Chapter 3  Cartesian coordinate system examples

Configuration parameters for Articulated Independent robot on page 27

The Base Offset is a set of coordinate values that redefines the origin of the robot. The correct base offset values are typically available from the robot manufacturer. Type the values for the base offsets in the $X_{1b}$ and $X_{3b}$ boxes of the Coordinate System Properties dialog box.

**Base Offsets**

Type the Base Offset values.

For the robot shown in our example, the Base Offset values are:

- $X_{1b} = 3.0$
- $X_{3b} = 4.0$

**End-Effector Offsets for Articulated Independent robot**

The robot can have an end effector attached to the end of robot link L2. If there is an attached end effector, configure the End-Effector Offset value on the Offsets tab in the Coordinate System Properties dialog box. The End-Effector Offsets are defined with respect to the tool reference frame at the tool tip.

Some robots also have an offset defined for the J3 joint. Account for this value when computing the $X_{3e}$ end effector offset value. If the value for $X_{3e}$ offset is
entered as the sum of $X_{3e1} + X_{3e2} = -3 + 1.5 = -1.5$, the configured value for $X_{3e}$ is $-1.5$.

See also

- **Configuration parameters for Articulated Independent robot** on page 37
- **Link Lengths for Articulated Independent robot** on page 38
- **Base Offsets for Articulated Independent robot** on page 62

**Configure Delta robot geometries**

The Logix Designer application supports three types of geometries that are often called parallel manipulators.

- Three-dimensional Delta
- Two-dimensional Delta
- SCARA Delta

In these geometries, the number of joints is greater than the degrees of freedom, and not all the joints are actuated (motor driven). These un-actuated joints are typically spherical joints.
Configure a Delta Three-dimensional robot

This illustration shows a four axes Delta robot that moves in three-dimensional Cartesian (X1, X2, X3) space. This type of robot is often called a spider or umbrella robot.

The Delta robot in this illustration is a three-degree of freedom robot with an optional fourth degree of freedom used to rotate a part at the tool tip. In the Logix Designer application, the first three-degrees of freedom are configured as three joint axes (J1, J2, J3) in the robots coordinate system. The three joint axes are:

- Directly programmed in joint space.
- Automatically controlled by the embedded Kinematics software in the Logix Designer application from instructions programmed in a virtual Cartesian coordinate system.

This robot contains a fixed top plate and a moving bottom plate. The fixed top plate is attached to the moving bottom plate by three link-arm assemblies. All three of the link-arm assemblies have a single top link arm (L1) and a parallelogram two-bar link assembly (L2).

As each axis (J1, J2, J3) is rotated, the TCP of the gripper moves correspondingly in (X1, X2, X3) direction. The gripper remains vertical along the X3 axis while its position is translated to (X1, X2, X3) space by the mechanical action of the parallelograms in each of the forearm assemblies. The mechanical connections of the parallelograms via spherical joints ensures that the top and bottom plates remain parallel to each other.

Program the TCP to an (X1, X2, X3) coordinate, then the Logix Designer application computes the commands necessary for each of the joints (J1, J2, J3) to move the gripper linearly from the current (X1, X2, X3) position to the programmed (X1, X2, X3) position, at the programmed vector dynamics.

When each top link (L1) moves downward, its corresponding joint axis (J1, J2, or J3) is assumed to be rotating in the positive direction. The three joint axes of the robot are configured as linear axes.

To rotate the gripper, configure a fourth axis as a linear or rotary, independent axis.
Establish the reference frame for a Delta Three-dimensional robot

The reference frame for the Delta geometries is located at the center of the top fixed plate. Joint 1, Joint 2, and Joint 3 are actuated joints. If the Delta coordinate system in the Logix Designer application is configured with the joints homed at $0^\circ$ in the horizontal position, then $L_1$ of one of the link pairs will be aligned along the $X_1$ positive axis as shown. Moving in the counterclockwise direction from Joint 1 to Joint 2, the $X_2$ axis will be orthogonal to the $X_1$ axis. Based on the right hand rule, $X_3$ positive will be the axis pointing up (out of the paper).

Calibrate a Delta Three-dimensional robot

Use these steps to calibrate the robot.

To calibrate a Delta Three-dimensional robot:

1. Obtain the angle values from the robot manufacturer for $J_1$, $J_2$, and $J_3$ at the calibration position. Use these values to establish the reference position.
2. Move all joints to the calibration position by jogging the robot under programmed control or manually moving the robot when the joint axes are in an open loop state.

3. Do one of the following:
   a. Use the Motion Redefine Position (MRP) instruction to set the positions of the joint axes to the calibration values obtained in step 1.
   b. Set the configuration value for the joint axes home position to the calibration values obtained in step 1 and execute a Motion Axis Home (MAH) instruction for each joint axis.

4. Move each joint to an absolute position of 0.0. Verify that each joint position reads 0 degrees and the respective L1 is in a horizontal position.
   If L1 is not in a horizontal position, see the alternate method for calibrating a Delta three-dimensional robot.

See also

Alternate method for calibrating a Delta Three-dimensional robot on page 43

Rotate each joint to a position so that the respective link is at a horizontal position. Perform one of the following:

- Use an MRP instruction to set all the joint angles to 0° at this position.
- Configure the values for the Zero Angle Offsets on the Geometry tab in the Coordinate System Properties dialog box equal to the values of the joints in a horizontal position.

Configure Zero Angle Orientations for Delta Three-dimensional robot

For Delta robot geometries, the internal transformation equations in the Logix Designer application are written assuming that:

- Joints are at 0° when link L1 is horizontal.
- As each top link (L1) moves downward, its corresponding joint axis (J1, J2, or J3) is rotating in the positive direction.

If you want the joint angular position when L1 is horizontal to be at any other value than 0°, then configure the zero angle orientation values on the Geometry tab on Coordinate System Properties dialog box to align the joint angle positions with the internal equations.

For example, if the Delta robot is mounted so that the joints attached at the top plate are homed at 30° in the positive direction below horizontal and you
want the Logix Designer application readout values to be zero in this position, then configure the Zero Angle Orientation values to -30° on the Geometry tab on the Coordinate System Properties dialog box.

**Delta Robot with Joints Homed at 30°**

[Diagram of a Delta robot with joints homed at 30°]

**Configuring Delta robot Zero Angle orientation**

The work envelope is the three-dimensional region of space that defines the reaching boundaries for the robot arm. The typical work envelope for a Delta robot looks similar to plane in the upper region, with sides similar to a hexagonal prism and the lower portion similar to a sphere. For more information regarding the work envelope of Delta three-dimensional robots, see the documentation provided by the robot manufacturer.

Program the robot within a rectangular solid defined inside the robot’s work zone. The rectangular solid is defined by the positive and negative dimensions...
of the X1, X2, X3 virtual source axes. Be sure that the robot position does not go outside the rectangular solid. Check the position in the event task.

To avoid issues with singularity positions, the MCT instruction internally calculates the joint limits for the Delta robot geometries. When an MCT instruction is invoked for the first time, the maximum positive and maximum negative joint limits are internally calculated based upon the link lengths and offset values entered on the Geometry and Offsets tabs in the Coordinate System Properties dialog box.

**Delta three-dimensional Configuration Systems Properties dialog box**

- Geometry and Offsets tabs

During each scan, the joint positions in the forward and inverse kinematics routines are checked to ensure that they are within the maximum and minimum negative joint limits.

Homing or moving a joint axis to a position beyond a computed joint limit and invoking a MCT instruction results in an error 67 (Invalid Transform position). For more information regarding error codes, see Logix 5000 Controllers Motion Instructions Reference Manual, publication MOTION-RM002.

Define configuration parameters for a Delta Three-dimensional robot

Configure the Logix Designer application to control robots with varying reach and payload capacities. The configuration parameter values for the robot include:

- Link lengths
- Base offsets
- End-effector offsets

The configuration parameter information is available from the robot manufacturer.
Configure a Delta Two-dimensional robot

This illustration shows a two-dimensional Delta robot that moves in two-dimensional Cartesian space.

This robot has two rotary joints that move the gripper in the (X1, X2) plane. Two forearm assemblies attach a fixed top plate to a movable bottom plate. A gripper is attached to the movable bottom plate. The bottom plate is always orthogonal to the X2 axis and its position is translated in Cartesian space (X1, X2) by mechanical parallelograms in each forearm assembly. The two joints, J1, and J2, are actuated joints. The joints between links L1 and L2 and between L2 and the base plate are unactuated joints.

Each joint is rotated independently to move the gripper to a programmed (X1, X2) position. As each joint axis (J1 or J2 or J1 and J2) is rotated, the TCP of the gripper moves correspondingly in the X1 or X2 direction or X1 and X2 direction. Program the TCP to a (X1, X2) coordinate, then the Logix Designer application uses internal vector dynamic calculations to compute the proper commands needed for each joint to move the gripper linearly from the current (X1, X2) position to the programmed (X1, X2) position.

The two joint axes (J1 and J2) of the robot are configured as linear axes.

To rotate the gripper, configure a third axis as a linear or rotary, independent axis.

See also

- Establish the reference frame for a Delta Two-dimensional robot on page 47
- Calibrate a Delta Two-dimensional robot on page 47
- Identify the work envelope for a Delta Two-dimensional robot on page 47
Define configuration parameters for a Delta Two-dimensional robot on page 48

Establish the reference frame for a Delta Two-dimensional robot

The reference frame for the two-dimensional Delta geometry is located at the center of the fixed top plate. When the angles of joints J1 and J2 are both at 0°, each of the two L1 links is along the X1 axis. One L1 link is pointing in the positive X1 direction, the other in the negative X1 direction.

When the right-hand link L1 moves downward, joint J1 is assumed to be rotating in the positive direction and when L1 moves upward, the J1 is assumed to be moving in the negative direction. When the left-hand link L1 moves downward, joint J2 is assumed to be rotating in the positive direction and when left-hand L1 moves upward, the J2 is assumed to be moving in the negative direction.

See also

Calibrate a Delta Two-dimensional robot on page 47

Calibrate a Delta Two-dimensional robot

Calibrate a Delta two-dimensional robot using the same method for calibrating a Delta three-dimensional robot. Obtain the angle values from the robot manufacturer for J1 and J2 at the calibration position. Use these values to establish the reference position.

See also

Calibrate a Delta Three-dimensional robot on page 42
Identify the work envelope for a Delta Two-Dimensional robot

The work envelope is the two-dimensional region of space that defines the reaching boundaries for the robot arm. The typical working envelope for a two-dimensional Delta robot is a boundary composed of circular arcs.

Program the parameters for the two-dimensional Delta robot within a rectangle, dotted lines in the illustration, inside the robots work zone. Define the rectangle by the positive and negative dimensions of the X1, X2 virtual source axes. Be sure that the robot position does not go outside the rectangle. Check the position in the event task.

To avoid problems with singularity positions, the Logix Designer application internally calculates the joint limits for the Delta robot geometries. When an MCT instruction is invoked for the first time, the maximum positive and maximum negative joint limits are internally calculated based upon the link lengths and offset values entered on the Geometry and Offsets tabs of the Coordinate System Properties dialog box.

For more information about maximum positive and negative joint limits, see Maximum positive joint limit condition and Maximum negative joint limit condition.

Homing or moving a joint axis to a position beyond a computed joint limit and then invoking an MCT instruction, results in an error 67 (Invalid Transform position). For more information regarding error codes see the Logix 5000 Controllers Motion Instructions Reference Manual, publication MOTION-RM002.

Configure the Logix Designer application to control robots with varying reach and payload capacities. The configuration parameter values for the robot include:

- Link lengths
- Base offsets
- End-effector offsets

The configuration parameter information is available from the robot manufacturer.
Configure a SCARA Delta robot

The SCARA Delta robot geometry is similar to a two-dimensional Delta robot geometry except that the X1-X2 plane is tilted horizontally with the third linear axis in the vertical direction (X3).

![Diagram of SCARA Delta robot]

Establish the reference frame for a SCARA Delta robot

The reference frame for the SCARA Delta robot is located at the center of the base plate.

When the angles of joints J1 and J2 are both at 0°, the two L1 links is along the X1 axis. One L1 link is pointing in the positive X1 direction, the other in the negative X1 direction.

When the right-hand link L1 moves in the clockwise direction (looking down on the robot), joint J1 is assumed to be rotating in the positive direction. When the right-hand link L1 moves counterclockwise, joint J1 is assumed to be moving in the negative direction.

When left-hand link L1 moves in the clockwise direction, joint J2 is assumed to be moving in the negative direction. When the left-hand link L1 moves in the counterclockwise direction, joint J2 is assumed to be rotating in the positive direction.

Based on the right hand rule, X3 positive will be orthogonal to the X1-X2 plane pointing up. The linear axis will always move in the X3 direction.

When configuring a SCARA Delta robot in the Logix Designer application, observe these guidelines:

See also

- Establish the reference frame for a SCARA Delta robot on page 49
- Calibrate a SCARA Delta robot on page 50
- Identify the work envelope for a SCARA Delta robot on page 50
- Define configuration parameters for a SCARA Delta robot on page 51
- Configure a Delta robot with a Negative X1b offset on page 51
Chapter 3  Cartesian coordinate system examples

- Configure the source and the target coordinate system with a transform dimension of two.
- The linear axis configured as a third axis must be the same for both the source and target coordinate systems.

Calibrate a SCARA Delta robot using the same method for calibrating a Delta three-dimensional robot. For more information about calibration, see Calibrate a Delta Three-dimensional Robot.

See also

Calibrate a Delta Three-dimensional Robot on page 42

Identify the work envelope for a SCARA Delta robot

The work envelope for a SCARA Delta robot is similar to the two-dimensional Delta robot in the X1-X2 plane. The third linear axis extends the work region making it a solid region. The maximum positive and negative limits of the linear axis defines the height of the solid region.

It is recommended to program the SCARA Delta robot within a rectangular solid defined inside the work zone of the robot. Define the rectangular solid by the positive and negative dimensions of the X1, X2, X3 virtual source axes. Be sure that the robot position does not go outside the rectangular solid.

Check the position in the event task.

To avoid problems with singularity positions, the Logix Designer application internally calculates the joint limits for the Delta robot geometries. For more information about maximum positive and negative joint limits, see Maximum positive joint limit condition and Maximum negative joint limit condition.

Homing or moving a joint axis to a position beyond a computed joint limit, and invoking an MCT instruction, results in an error 67 Invalid Transform position. For more information regarding error codes, see Logix 5000.
Define configuration parameters for a SCARA Delta robot

The Logix Designer application can be configured for control of robots with varying reach and payload capacities. The configuration parameter values for the robot include:

- Link lengths
- Base offset
- End-effector offset

The configuration parameter information is available from the robot manufacturer.

See also

- Link length for SCARA Delta robot
- Base Offset for SCARA Delta robot
- End Effector Offset for SCARA Delta robot

Configure a Delta robot with a Negative X1b offset

Beginning with version 17 of the application, you can use negative offsets for the X1b base offset on 2D and 3D delta geometries. For example, a mechanical 2D delta robot using a negative X1b offset has a mechanical configuration as shown in the diagram.

See also

- Maximum positive joint limit condition
- Maximum negative joint limit condition
The base offset $X_{1b}$ is the value equal to the distance from the origin of the robot coordinate system to one of the actuator joints. In the previous figure, one of the actuator joints (P1), is on the negative side of X1. The base offset $X_{1b}$ is -10 units from the origin of the coordinate system (X1 - X2 intersection) to P1.

The Logix Designer application coordinate system configuration for the offset tab used with the preceding example is shown in the following example.

This negative offset description also applies for Delta 3D and SCARA-Delta configurations.

**Arm solutions**

A kinematic arm solution is the position of all joints on the robot that correspond to a Cartesian position. When the Cartesian position is inside the workspace of the robot, then at least one solution always exists. Many of the geometries have multiple joint solutions for a single Cartesian position.

- Two axis robots - two joint solutions typically exist for a Cartesian position.
- Three axis robots - four joint solutions typically exist for a Cartesian position.

**See also**

- Left-arm and right-arm solutions for two-axes robots on page 53
- Solution mirroring for three-dimensional robots on page 53
- Change the robot arm solution on page 54
- Plan for singularity on page 55
- Encounter a no-solution position on page 55
**Left-arm and right-arm solutions for two-axes robots**

A robot having an arm configuration has two kinematics solutions when attempting to reach a given position. Point A is shown in the following illustration. One solution satisfies the equations for a right-armed robot, the other solution satisfies the equations for a left-armed robot.

See also

[Arm solutions on page 52](#)

**Solution mirroring for three-dimensional robots**

For a three-dimensional Articulated Independent robot, there are four solutions for the same point:

- Left-arm
- Right-arm
- Left-arm mirror
- Right-arm mirror

For example, consider the Cartesian point XYZ (10,0,15). The joint position corresponding to this point has four joint solutions. Two of the solutions are the same as the solutions for the two-dimensional case. The other solutions are mirror image solutions where J1 is rotated 180°.
You can switch the robot from a left-arm solution to a right-arm solution or vice versa. This is done automatically when a joint move is programmed forcing a left/right change to occur. After the change is performed, the robot stays in the new arm solution when Cartesian moves are made. If required, the robot arm solution changes again when another joint move is made.

**Example:** Suppose, you want to move the robot from position A \((x_1,y_1)\) to position B \((x_2,y_2)\) as shown in the following figure. At position A, the system is in a left arm solution. When programming a Cartesian move from A \((x_1,y_1)\) to B \((x_2,y_2)\), the system moves along the straight line from A to B while maintaining a left arm solution. If you want to be at position B in a right-arm solution, you must make a joint move in J1 from \(\theta_1\) to \(\theta_2\) and a joint move in J2 from \(\alpha_1\) to \(\alpha_2\).
Plan for singularity

A singularity occurs when an infinite number of joint positions (mathematical solutions) exist for a given Cartesian position. The Cartesian position of a singularity is dependent on the type of the robot geometry and the size of the link lengths for the robot. Not all robot geometries have singularity positions.

For example, singularities for an Articulated Independent robot occur when:

- The robot manipulator folds its arm back onto itself and the Cartesian position is at the origin.
- The robot is fully stretched at or very near the boundary of its workspace.

An error condition is generated when a singularity position is reached.

**WARNING:** Avoid programming the robot towards a singularity position when programming in Cartesian mode. The velocity of the robot increases rapidly as it approaches a singularity position and can result in injury or death to personnel.

See also

Arm solutions on page 52

Encounter a no-solution position

When a robot is programmed to move beyond its work envelope, there is no mathematical joint position for the programmed Cartesian position. The system forces an error condition.

For example, if an Articulated Independent robot has two 10-inch arms, the maximum reach is 20 inches. Programming to a Cartesian position beyond 20 inches produces a condition where no mathematical joint position exists.

**WARNING:** Avoid programming the robot towards a no-solution position when programming in Cartesian mode. The velocity of the robot increases rapidly as it approaches this position and can result in injury or death to personnel.

See also

Arm solutions on page 52

Error conditions

Kinematics error conditions are detected:

- Upon activation of a transformation by executing an MCT instruction.
- In some movement conditions.

Errors can occur for certain movement conditions for either the source or target coordinate system after a transformation has been established. These types of errors are reported in the MCT instruction error codes. Singularity and other movement error conditions are also reported in the MCT error codes.
• Computing an invalid position via an MCTP instruction.

For a list and description of error codes, see Logix5000 Controllers Motion Instructions Reference Manual, publication MOTION-RM002.

Configure an Articulated Dependent robot

The Articulated dependent robot contains motors for the elbow and the shoulder at the base of the robot. The dependent link controls J3 at the elbow. Follow these guidelines when configuring an Articulated Dependent robot.

WARNING: Before turning ON the Transform and/or establishing the reference frame, do the following for the joints of the target coordinate system:

• Set and enable the soft travel limits.
• Enable the hard travel limits.

Failure to perform these steps can cause robotic arm to move to unexpected positions causing machine damage and/or injury or death to personnel.

See also

Reference frame for Articulated Dependent robots on page 56

Methods to establish a reference frame for an articulated dependent robot

Work envelope for Articulated Dependent robot

Define configuration parameters for Articulated Dependent robot on page 61

Reference frame for Articulated Dependent robots

The reference frame is the Cartesian (typically the source) coordinate frame that defines the origin and the primary axes, X1, X2, and X3. These are used to measure the real Cartesian positions.

WARNING: Failure to properly establish the correct reference frame for the robot can cause the robotic arm to move to unexpected positions causing machine damage and/or injury or death to personnel.
Example 1: Articulated Dependent robot 1

This diagram illustrates the reference frame for an Articulated Dependent robot at the base of the robot.

These equations represent the Articulated Dependent robot joint positioning shown in Articulated Dependent robot 1 diagram.

- $+J_1$ is measured counterclockwise around the $+X_3$ axis starting at an angle of $J_1=0$ when $L_1$ and $L_2$ are both in the $X_1$-$X_2$ plane.
- $+J_2$ is measured counterclockwise starting with $J_2=0$ when $L_1$ is parallel to $X_1$-$X_2$ plane.
- $+J_3$ is measured counterclockwise with $J_3=0$ when $L_2$ is parallel to the $X_1$-$X_2$ plane.

When the robot is in this position, the Logix Designer application Actual Position tags for the axes must be:

- $J_1 = 0$.
- $J_2 = 0$.
- $J_3 = 0$.

Example 2: Figure 79 - Articulated Dependent 2

When the robot is in this position, the Logix Designer application Actual Position tags for the axes must be:

- $J_1 = 0$.
- $J_2 = 90$.
- $J_3 = -90$. 
Example 3: Articulated Dependent 3

If the position and joint angle values of the robot are unable to match the Articulated Dependent 2 or in Articulated Dependent 3 examples, use a methods outlined in the Method to Establish a Reference Frame for an articulated dependent robot topic to establish the Joint-to-Cartesian reference frame relationship.

Use the following methods to establish a reference frame for the robot.

<table>
<thead>
<tr>
<th>For each:</th>
<th>Use one of these methods to establish the reference frame:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental axis</td>
<td>Each time the power for the robot is cycled.</td>
</tr>
<tr>
<td>Absolute axis</td>
<td>Only to establish absolute home.</td>
</tr>
</tbody>
</table>

- Method 1 - Establishes a Zero Angle Orientation and allows the configured travel limits and home position on the joint axes to remain operational. Use this method when operating the axes between the travel limits determined prior to programming a Motion Redefine Position (MRP) instruction and want these travel limits to stay operational.
- Method 2 - Uses a MRP instruction to redefine the axes position to align with the joint reference frame. This method may require the soft travel limits to be adjusted to the new reference frame.

See also

Method 1 for an incremental axis on page 59
Method 2 for an absolute axis on page 36
Each axis for the robot has the mechanical hard stop in each of the positive and negative directions. Manually move or press each axes of the robot against its associated mechanical hard stop and redefine it to the hard limit actual position provided by the robot manufacturer. J1 is the axis at the base of the robot that rotates around X3.

When the robot is moved so that Link1 is parallel to the X3 axis and Link2 is parallel to X1 axis, the values for the Actual Position tags for the axes in the Logix Designer application should be:

- J1 = 0
- J2 = 90°
- J3 = 0°

If the Actual Position tags do not show these values, configure the Zero Angle Orientation parameters in the Coordinate System Properties dialog box for the joint or joints that do not correspond.

<table>
<thead>
<tr>
<th>If the Logix Designer application read-out values are:</th>
<th>Set the Zero Angle Orientations on the Coordinate System Properties dialog box to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1 = 10</td>
<td>Z1 = -10</td>
</tr>
<tr>
<td>J2 = 80</td>
<td>Z2 = 10</td>
</tr>
<tr>
<td>J3 = 5</td>
<td>Z3 = -5</td>
</tr>
</tbody>
</table>

The Joint-to-Cartesian reference frame relationship is automatically established by the Logix controller after the Joint coordinate system parameters (link lengths, base offsets, and end effector offsets) are configured and the MCT instruction is enabled.
Method 2 - Establish a reference frame

Position the robot so that:
- L1 is parallel to the X3 axis.
- L2 is parallel to X1 axis.

Program a Motion Redefine Position (MRP) instruction for all the three axis to with the following values 0, 90, and 0°.

The Joint-to-Cartesian reference frame relationship is automatically established by the 1756-L6xx controller after the Joint coordinate system parameters, which are link lengths, base offsets, and end-effector offsets, are configured and the MCT instruction is enabled.

The work envelope is the three-dimensional region of space that defines the reaching boundaries for the robot arm. The work envelope for an articulated robot is ideally a complete sphere with an inner radius equal to L1 - L2 and outer radius equal to L1 + L2. Due to the range of motion limitations on individual joints, the work envelope may not be a complete sphere.

### Work envelope for articulated independent robot

<table>
<thead>
<tr>
<th>If the range-of-motion values for the articulated robot are:</th>
<th>Typically, the work envelope is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1 = ± 170</td>
<td><img src="image1.png" alt="Top view" /></td>
</tr>
<tr>
<td>J2 = 0 to 180</td>
<td><img src="image2.png" alt="Side view" /></td>
</tr>
<tr>
<td>J3 = ± 60</td>
<td></td>
</tr>
<tr>
<td>L1 = 10</td>
<td></td>
</tr>
<tr>
<td>L2 = 12</td>
<td></td>
</tr>
</tbody>
</table>

Top view - Depicts the envelope of the tool center point sweep in J1 and J3 while J2 remains at a fixed position of 0°.

Side view - Depicts the envelope of the tool center point sweep in J2 and J3 while J1 remains at a fixed position of 0°.
See also

Configuration parameters for articulated independent robot on page 37

Configure an articulated independent robot on page 33

Configure the Logix Designer application to control robots with varying reach and payload capacities. Be sure to have these configuration parameter values for the robot:

- Link lengths
- Base offsets
- End-effector offsets

The configuration parameter information is available from the robot manufacturer.

IMPORTANT Verify that the values for the link lengths, base offsets, and end-effector offsets are entered into the Configuration Parameters dialog box using the same measurement units.

This example illustrates the typical configuration parameters for an Articulated Dependent robot.

If the robot is two-dimensional, the X3b and X3e are X2b and X2e.

See also

Link lengths for Articulated Dependent robot on page 62
Link lengths are the rigid mechanical bodies attached at joints.

### Link lengths for Articulated Dependent robot

Link lengths are the rigid mechanical bodies attached at joints.

<table>
<thead>
<tr>
<th>For an articulated dependent robot with</th>
<th>The length of</th>
<th>Is equal to the value of the distance between</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 dimensions</td>
<td>L1</td>
<td>J1 and J2</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>J2 and the end-effector</td>
</tr>
<tr>
<td>3 dimensions</td>
<td>L1</td>
<td>J2 and J3</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>J3 and the end-effector</td>
</tr>
</tbody>
</table>

Enter the link lengths on the **Geometry** tab in the **Coordinate System Properties** dialog box.

Type the **Link Length** values.
The **Link Length** values in this example are:
- L1 = 10.0
- L2 = 12.0

See also

**Configuration parameters for Articulated Dependent robot** on page 61

### Base offsets for Articulated Independent robot

The base offset is a set of coordinate values that redefines the origin of the robot. The correct base-offset values are typically available from the robot.
manufacturer. Type the values for the Base Offsets in the $X_{1b}$ and $X_{3b}$ boxes on the Geometry tab in the Coordinate System Properties dialog box.

See also
- Link Lengths on page 38
- End Effector Offsets on page 39
- Configuration parameters for Articulated Independent robots on page 37

Configure a Cartesian Gantry robot
Use these guidelines when configuring a Cartesian Gantry robot.

See also
- Establish the reference frame for a Cartesian Gantry robot on page 63
- Identify the work envelope for a Cartesian Gantry robot on page 64
- Define configuration parameters for a Cartesian Gantry robot on page 64

Introduction
Establish the reference frame for a Cartesian Gantry robot

Use these guidelines when configuring a Cartesian Gantry robot.

For a Cartesian Gantry robot, the reference frame is an orthogonal set of $X_1$, $X_2$, and $X_3$ axes positioned anywhere on the Cartesian robot. All global coordinate measurements (points) are relative to this reference frame.
Typically, the reference frame is aligned with the X1, X2, and X3 axes of the machine.

To establish a Local coordinate system with axes positions different from the reference frame, use the Motion Redefine Position (MRP) instruction to reset the position register. Also use the Offset Vector in the MCT transform instruction to establish an offset between the Local coordinate system and the reference frame.

For more information about Motion Instructions, see Logix 5000 Controllers Motion Instructions Reference Manual, publication MOTION-RM002.

The work envelope for a Cartesian Gantry robot is typically a solid rectangle of length, width, and height that is equal to the axis travel limits.

Defining the link lengths, base offset, or end-effector offset configuration parameters is not required for a Cartesian Gantry robot.

Identify the work envelope for a Cartesian Gantry robot
Define configuration parameters for a Cartesian Gantry robot
Configure a Cartesian H-bot

The H-bot is a special type of Cartesian two-axis gantry robot. This type of machine has three rails positioned in the form of a letter H. Two motors are positioned at the end of each leg of the robot. Unlike a standard gantry robot, neither motor is riding on top of the moving rails. Use these guidelines when configuring a Cartesian H-bot.

In the Cartesian H-bot illustration, the X1 and X2 axes are the real axes on the robot. X1 Virt and X2 Virt are configured as the virtual axes.

The configuration of the H-bot mechanical linkages enable it to move at a 45° angle to the axes when motor A or motor B is rotated.

For example, when:

- Motor A (X1 axis) is rotated, the robot moves along a straight line at +45° angle.
- Motor B (X2 axis) is rotated, the machine moves at an angle of -45°.
- Motors A and B are rotated clockwise at the same speed, then the machine moves along a horizontal line.
- Motors A and B are rotated counterclockwise at the same speed then, the machine moves along a vertical line.

Any X,Y position can be reached by properly programming the two motors.

For example, a move of (X1 = 10, X2 = 0) causes the X1X2 axes to move to a position of (X1=7.0711, X2=7.0711). A move to (X1=10, X2 =10) causes the robot to move to a position of (X1=0, X2=14.142).

Utilizing the Logix Designer application Kinematics function configured with two Cartesian coordinate systems and a -45° rotation performs the function.
To configure two Cartesian coordinate systems:

Coordinate System 1 (CS1) and Coordinate System 2 (CS2) each contain two linear axes.

1. Configure CS1 to contain the virtual X1 and X2 axes.
2. Configure CS2 to contain the real X1 and X2 axes.
3. Configure the Orientation vector of the MCT instruction as (0,0,-45), a negative degree rotation around the X3 axis.
4. Configure the Translation vector as (0,0,0).
5. Link the CS1 and CS2 by using a MCT instruction.
6. Home the H-bot and then program all moves in CS1.

The machine moves the tool center point (TCP) to the programmed coordinates in CS2. The \(-45^\circ\) rotation introduced by the Kinematics, counteracts the \(45^\circ\) rotation introduced by the mechanics of the machine and the H-bot moves to the CS1 configured coordinates. As a result, a programmed move of X1\(_{\text{virt}}\)=10, X2\(_{\text{virt}}\)=5 moves to a real mechanical position of X1=10, X2=5.

See also

- Establish the reference frame for a Cartesian H-bot robot on page 66
- Identify the work envelope for a Cartesian H-bot robot on page 66
- Define configuration parameters for a Cartesian H-bot robot on page 66

Establish the reference frame for a Cartesian H-bot

For a Cartesian H-bot, the Base coordinate system is an orthogonal set of X1, X2 axes postponed anywhere on the Cartesian H-bot. The angular rotation of the reference frame may not be rotated for this robot since the angular rotation vector is used to achieve the \(45^\circ\) rotation required for the mechanical operation.

Identify the work envelope for a Cartesian H-bot

The work envelope for a Cartesian H-bot is a rectangle of length and width equal to the axis soft travel limits.

Define configuration parameters for a Cartesian H-bot robot

Defining the link lengths, base offset, or end-effector offset configuration parameters is not required for a Cartesian H-bot robot.

Configure a SCARA Independent Robot

The typical SCARA Independent robot has two revolute joints and a single prismatic joint. This robot is identical to the Articulated Independent two dimensional robot except that the X1-X2 plane is tilted horizontally with a third linear axis in the vertical direction. Use these guidelines when configuring a SCARA Independent robot.
Configure the SCARA Independent J1J2J3J6 Coordinate System

This illustration shows a SCARA Independent J1J2J3J6 coordinate system robot. The typical SCARA Independent J1J2J3J6 robot has three revolute joints and one prismatic joint. From base frame, Link 1 (L1) is rigid arm which connects Joint J1/J2 and Link 2 (L2) is also a rigid arm connecting J2/J3/J6. Two independent motors producing coordinated motion at Joint 1 (J1) and Joint 2 (J2) respectively to control the SCARA’s X-Y motion. Joint 3 (J3) and Joint 6 (J6) produce Z-Rz motion at the end of arm.

Some of the SCARA geometries have ball screw spline assembly. This assembly can provide linear and rotary motion as well as combined spiral motion, where J3 controls the linear motion in the Z axis and J6 controls the rotational motion.

Use these guidelines when configuring a SCARA Independent J1J2J3J6 robot.

See also

- Configuration Parameters for the Robot on page 67
- Robot Configuration for SCARA Independent J1J2J3J6 Robot on page 74
- Maximum Joint Limits condition for SCARA Independent J1J2J3J6 robot on page 79
- Sample Project for SCARA Independent J1J2J3J6 Robot on page 80

Configuration Parameters for the Robot

Configure the Logix Designer application, to control robots with varying reach and payload capacities. The configuration parameter values for the robot includes:

- Link Lengths
- Zero Angle Orientations
• Ball Screw Lead

The configuration parameter information is available from the robot manufacturer.

Tip: Base offsets and end-effector offsets do not apply to a SCARA Independent J1J2J3J6 robot.

### Link Lengths for SCARA Independent J1J2J3J6 Robot

Link lengths are the rigid mechanical bodies attached to the joints. Configure **Link Lengths L1** and **L2** in the **Geometry** tab of **Coordinate System Properties** dialog box.

### Zero Angle Orientations for SCARA Independent J1J2J3J6 Robot

For SCARA robot geometries, the internal transformation equations in the Logix Designer application assume:

- J1 and J2 are at 0° when link L1 is aligned to L2 along with X axis of the base frame.
- J6 axis of rotation is aligned with Z axis of End of Arm frame (Z axis of End of Arm frame pointing down with respect to base frame) or in parallel with Z axis of base frame when J6 is at 0.

To have joints J1, J2, and J6 angular positions be any value other than 0, configure the **Zero Angle Orientation** values on the **Geometry** tab in the **Coordinate System Properties** dialog box to align the joint angle positions with the internal equations.

For example:

- Joint J1 is moved by 30° and J2 is moved by 15° from their default home positions and this is the new Home position for J1 and J2. If you need the readout values in the application to be zero in this new position, enter -30° in Z1 and -15° in Z2 parameter on the Geometry tab.
• The Z6 offset is used to set J6 axis home position other than the default 0 position. In this example, the Joint J6 is moved by -90° from its default home position. To get the new home position for J6, we need to set Z6 to -90°.

The first diagram shows the top view with Zero Angle orientation. The second diagram shows the side view of J6 with Zero Angle Offset before and after -90° rotation.
In Some SCARA robots Ball screw and spline mechanism is used to get rotation and linear movement using a single shaft setup.

In general, as shown in this image, to control the position and orientation of the Shaft, the Ball Screw Nut and Ball Spline Nut need to work together.

The Ball Screw Nut only introduces linear motion of the shaft (up and down, no rotation), the direction of the movement depends on the thread types of the ball screws. The J3 motor is producing the linear motion by rotating the ball screw nut.

For the Ball Spline Nut, it introduces the rotation of the shaft, and the linear position of the shaft also changes The Ball Spline Nut is rotated by the J6 motor.
In many cases, you would use Ball Screw Nut and Ball Spline Nut together to compensate the linear movement for each other, to introduce the rotation only movement of the Shaft.

For the SCARA robot, in the Logix firmware, J3 is associated with the Ball Screw Nut; and J6 is associated with the Ball Spline Nut.

As shown in the image above, J3 performs linear movement to change the Cartesian Z position of Shaft. To change the linear position of the Shaft only, J3 is used.

In the image above, J6 rotation introduces the rotation of the shaft, which also causes the linear movement.

The distance of the linear movement caused by the rotation of the shaft is calculated by the Lead parameter, the formula is

\[
\text{Lead} = \frac{\text{Linear Movement Distance}}{\text{One Revolution of the rotation}}
\]
As shown in the image above, J3 can perform linear movement to change the Cartesian Z position of Shaft. To change the linear position of the shaft only, J3 is used.

![Diagram showing J3 and J6 axes](image)

As shown in the image above, to rotate the shaft only without the linear movement. Move both J3 and J6.

In Studio 5000 Logix Designer application, when only the Rz move is programmed and Z remains the same, the Kinematics transformations in the controller compensate the upward or downward motion caused by the mechanical coupling of the J6 axis by generating opposite movement for J3 axis. The net effect is that you observe only the rotational Rz movement.

These examples address the three scenarios shown in the images.

Assuming **Lead** is 36 mm/revolution, and J3, J6, Z and Rz are all set to 0.

**Example 1: Moving J3 only:**

If J3 moves up 3 mm, $J_3 = -3$ mm

\[
Z = -J_3 \\
= 3 \text{ mm}
\]

**Example 2: Moving J6 only:**

If J6 is rotated 30 degree in clockwise.

\[
R_z = -J_6 = -30 \\
Z = -J_6 \times \text{Lead} \\
= -30 \times \frac{36}{360} \\
= -3 \text{ mm}
\]

**Example 3: Moving Rz only:**

If Rz is rotated by 30 deg in clockwise direction.

\[
R_z = -30
\]
then

\[ J6 = -R_z = 30 \]

Since J6 is moved by 30 degrees, it produces linear movement on Z axis. To compensate this linear move effect J3 needs to move in the opposite direction.

\[ J3 = -J6 \times \text{Lead} \]
\[ = -30 \times \frac{36}{360} \]
\[ = -3 \text{mm} \]

so

\[ Z = 0 \]

Means there is no linear movement.

These three examples are included in the table.

<table>
<thead>
<tr>
<th>Joint Configuration (Lead=36 mm/rev)</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>Rx</th>
<th>Ry</th>
<th>Rz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Setting, J3 = 0, J6 = 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Example 1: J3 = -3</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Example 2: J6 = 30</td>
<td>0</td>
<td>0</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>-30</td>
</tr>
<tr>
<td>Example 3: J3 = -3, J6 = 30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-30</td>
</tr>
</tbody>
</table>

The three examples are shown in Studio 5000 Logix Designer.

A SCARA 4 Axis example is shown here.

First, the **Lead** parameter is set to 36.0 Coordination Unit per Revolution.

And currently as shown in the figure above, in Joint space, J3 = 0 and J6 = 0.

And in cartesian space, Z = 0 and Rz = 0.
First we move J3 to -3 position.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Controller</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rz, CommandPosition</td>
<td>Controller</td>
<td>30.0</td>
</tr>
<tr>
<td>J6, CommandPosition</td>
<td>Controller</td>
<td>30.0</td>
</tr>
<tr>
<td>J3, CommandPosition</td>
<td>Controller</td>
<td>0.0</td>
</tr>
<tr>
<td>Z, CommandPosition</td>
<td>Controller</td>
<td>-3.0</td>
</tr>
</tbody>
</table>

Now Z = -J3 = 3, shown in the figure above.

Then, reset all the parameters to 0 and move J6 to 30.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Controller</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rz, CommandPosition</td>
<td>Controller</td>
<td>30.0</td>
</tr>
<tr>
<td>J6, CommandPosition</td>
<td>Controller</td>
<td>30.0</td>
</tr>
<tr>
<td>J3, CommandPosition</td>
<td>Controller</td>
<td>0.0</td>
</tr>
<tr>
<td>Z, CommandPosition</td>
<td>Controller</td>
<td>-3.0</td>
</tr>
</tbody>
</table>

Now in the figure above, Rz = -30 and Z = -3, which is consistent with the results of Example 2.

Reset all the parameters again and move J3 to 3 and J6 to 30.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Controller</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rz, CommandPosition</td>
<td>Controller</td>
<td>30.0</td>
</tr>
<tr>
<td>J6, CommandPosition</td>
<td>Controller</td>
<td>30.0</td>
</tr>
<tr>
<td>J3, CommandPosition</td>
<td>Controller</td>
<td>3.0</td>
</tr>
<tr>
<td>Z, CommandPosition</td>
<td>Controller</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Now in the figure above, Z = 3 and Rz = -30, which is consistent to the Example 3.

**Robot Configuration for SCARA Independent J1J2J3J6 Robot**

The SCARA robot has two kinematics solutions when attempting to reach a given position.

While achieving a given target position, if J2 is moving in the negative direction with respect to the frame at the end of link L1 (J2 angle is negative), the configuration is considered Lefty Configuration. If J2 is moving in a positive direction with respect to the frame at the end of the link L1 (J2 angle is positive), the configuration is considered Righty Configuration.
Chapter 4  Configure a Cartesian H-bot

The illustration below shows the same cartesian position achieved by Righty and Lefty configuration.

- When looking at the EOA:
  - If the elbow is to the right, the configuration is Righty.
  - If the elbow is to the left, the configuration is Lefty.

- When MCTO is initiated, it sets Robot Configuration based on current J2 position and while MCTO is active, it remains in the same configuration.

- If MCPM continuous path (CP) move is programmed with a robot configuration parameter that is different from the robot configuration set by the MCTO instruction, it gives error 136.

For Error codes and instruction details refer to the MCPM instruction section.

In MCTPO, Bit 0 of the Robot Configuration is ignored. Robot Configuration parameter is input and output parameter for MCTPO instruction which depends on Transform Direction used.

- If MCTPO Transform direction is set to "Forward Transform", then the system computes the Robot Configuration for the user and updates to tag data.

- If MCTPO Transform direction is set to "Inverse Transform" then the user provides Robot Configuration as an input tag.

Robot Configuration is DINT datatype tag and its definition is shown in this table:

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Flip (1)/ No Flip (0)</td>
<td>Above (1)/ Below (0)</td>
<td>Left (1)/ Right (0)</td>
<td>Change (1)/ Same (0)</td>
</tr>
<tr>
<td>Robot configured as Right</td>
<td>N/A</td>
<td>N/A</td>
<td>0</td>
<td>x</td>
</tr>
</tbody>
</table>
Bit Position | 3 | 2 | 1 | 0  
---|---|---|---|---
Robot configured as Left | N/A | N/A | 1 | x  

Notations:

N/A: Not applicable for SCARA J1J2J3J6 Robot.

×: Value is ignored.

For more Error codes and instruction details refer to the MCTPO instruction section.

For an example, suppose we have L1 and L2 of length 350 units each. The SCARA robot needs to move to the EOA at cartesian coordinates x=450, y=130. The two solutions are shown in this image.

The Studio 5000 Logix Designer application detects a certain Cartesian Position and needs to know the Joint positions with respect to a certain Robot configuration.

This example illustrates an MCTPO instruction with Transform Direction as Inverse, where the user feeds Cartesian Position and Robot Configuration for Left Configuration as input. The instruction computes the corresponding target joint angle positions and writes the value to the Transform Position parameter as the output.

MCPTO1

This example illustrates an MCTPO instruction with Transform Direction as Inverse, where the user feeds Cartesian Position and Robot Configuration for Right Configuration as input. The instruction computes the corresponding
target joint angle positions and writes the value to the **Transform Position** parameter as the output.

The application knows the Joint Positions and would like to know the Cartesian Position and Robot configuration associated to that Robot position.

This example illustrates the MCTPO instruction with Transform Direction as Forward. The target positions configured are guided into the Reference position operand as input. The instruction computes the corresponding Cartesian positions and Robot Configuration as the output. In this example target positions are evaluated as Left configuration. The target positions configured are guided into the Reference position operand as input. The instruction computes the corresponding Cartesian positions and Robot Configuration as the output. In this example target positions are evaluated as Right configuration.

The Studio 5000 Logix Designer application knows the Joint Positions and would like to know the Cartesian Position and Robot configuration associated to that Robot position.

This example illustrates the MCTPO instruction with Transform Direction as Forward. The target positions configured are guided into the Reference position operand as input. The instruction computes the corresponding Cartesian positions and Robot Configuration as the output. In this example target positions are evaluated as Left configuration. The target positions configured are guided into the Reference position operand as input. The instruction computes the corresponding Cartesian positions and Robot Configuration as the output. In this example target positions are evaluated as Right configuration.
Cartesian positions and Robot Configuration as the output. In given example target positions are evaluated as Left configuration.

This example illustrates the MCTPO instruction with Transform Direction as Forward. The target positions configured are guided into the Reference position operand as input. The instruction computes the corresponding Cartesian positions and Robot Configuration as the output. In given example target positions are evaluated as Right configuration.

**Identify the Work Envelope for the Robot**

The work envelope is the three-dimensional region of space that defines the reaching boundaries for the SCARA Independent J1J2J3J6 robot arm. The work envelope for the SCARA Independent J1J2J3J6 Robot is a hollow cylinder with:

- A height (H) equal to the travel limit of the J3 axis.
- An inner radius (R1) equal to |L1-L2|.
- An outer radius (R2) equal to |L1+L2|.

Due to the limited range of motion on individual joints J1 and J2, the work envelope may not be a complete cylinder.

The work envelope for the SCARA Independent J1J2J3J6 robot varies if the tool is attached to the robot. The tool shape and dimensions may modify the work envelope.
Maximum Joint Limits
condition for SCARA
Independent J1J2J3J6 robot

- The maximum joint limits for configuring Joint 1(J1) and Joint 2(J2) axes is +/-179°.
- The Joint 3(J3) is a linear axis and does not have any kinematics limits. J3 range depends on the stroke length value provided by manufacturers.
- The Joint 6(J6) axis is the rotational axis that can have multiple turns. The maximum number of turns supported is +/-127. Maximum positive and negative range is checked based on number of turns supported on J6.

These limits are set as a Soft Travel Limit on the Scaling tab in the Axis Properties dialog box.

Homing or moving a joint axis to a position beyond a computed joint limit and invoking a MCTO instruction results in an error 151 ("Joint Angle beyond limit"). For more information regarding error codes, see Logix 5000 Controllers Motion Instructions Reference Manual, publication MOTION-RM002.

Work and Tool Frame offset limits for SCARA
Independent J1J2J3J6 robot

The Work and Tool Frame offset values defined in the MCTO and MCTPO instruction. SCARA Independent J1J2J3J6 Robot geometry has orientation limitations at the end of the arm, so Work and Tool frame offset values are limited up to reachable work envelope.

These offset values are allowed for Work and Tool frames. The MCTO and MCTPO instructions generates error 148 for invalid offset values.

- Offset values on X, Y, Z and Rz axis are allowed for the Work Frame offsets. Rx and Ry offsets are restricted and must be set to 0. Specify these offsets through the Work Frame parameter in the MCTO and MCTPO instructions.
- Offset values on X, Y, Z and Rz axis are allowed for the Tool Frame offsets. Rx and Ry offsets are restricted and must be set to 0. Specify these offsets through the Tool Frame parameter in the MCTO and MCTPO instructions.
Sample Project for SCARA Independent J1J2J3J6 Robot

To use the Kinematic sample project on configuring a SCARA Independent J1J2J3J6 Robot, on the Help menu, select Vendor Sample Projects and then select the Motion category.

The Rockwell Automation sample project’s default location is:

c:\Users\Public\Public Documents\Studio 5000\Sample\ENU\v<current_release>\Rockwell Automation
## Coordinate system attributes

### Attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Data Type</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accel Status</td>
<td>BOOL</td>
<td>Tag</td>
<td>Use the Accel Status bit to determine if the coordinated (vectored) motion is currently being commanded to accelerate. The acceleration bit is set when a coordinated move is in the accelerating phase due to the current coordinated move. It clears when the coordinated move stops or the coordinated move is in the decelerating phase.</td>
</tr>
<tr>
<td>Actual Pos Tolerance Status</td>
<td>BOOL</td>
<td>Tag</td>
<td>Use the Actual Pos Tolerance Status bit to determine when a coordinate move is within the Actual Position Tolerance. The Actual Pos Tolerance Status bit is set for AT term type only. The bit is set when interpolation is complete and the actual distance to programmed endpoint is less than the configured AT value. The bit remains set after an instruction completes. The bit is reset if a new instruction is started or the axis moves so that the actual distance to programmed endpoint is greater than the configured AT value.</td>
</tr>
<tr>
<td>Actual Position</td>
<td>REAL[8]</td>
<td>Tag</td>
<td>Array of actual position of each axis associated to this motion coordinate system in Coordinate Units.</td>
</tr>
<tr>
<td>Actual Position Tolerance</td>
<td>REAL</td>
<td>GSV</td>
<td>Coordination Units</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SSV</td>
<td>The Actual Position Tolerance attribute value is a distance unit used when instructions, such as MCLM and MCCM, specify a Termination Type of Actual Position.</td>
</tr>
<tr>
<td>Axes Configuration Faulted</td>
<td>DINT</td>
<td>Tag</td>
<td>Shows which axes in this coordinate system have a configuration fault.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If this bit is on</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Axes Inhibited Status</td>
<td>DINT</td>
<td>Tag</td>
<td>Shows which axes in this coordinate system are inhibited.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If this bit is on</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Axes Servo On Status</td>
<td>DINT</td>
<td>Tag</td>
<td>Shows which axes in this coordinate system are on (via MS0).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If this bit is on</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>
### Appendix A  Coordinate system attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Data Type</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accel Status</strong></td>
<td>BOOL</td>
<td>Tag</td>
<td>Use the Accel Status bit to determine if the coordinated (vectored) motion is currently being commanded to accelerate. The acceleration bit is set when a coordinated move is in the accelerating phase due to the current coordinated move. It clears when the coordinated move stops or the coordinated move is in the decelerating phase.</td>
</tr>
<tr>
<td><strong>Axes Shutdown Status</strong></td>
<td>DINT</td>
<td>Tag</td>
<td>Shows which axes in this coordinate system are shutdown.</td>
</tr>
<tr>
<td><strong>Axis Fault</strong></td>
<td>DINT</td>
<td>Tag</td>
<td>The Axis Fault Bits attribute is a roll-up of all of the axes associated to this motion coordinate system. A bit being set indicates that one of the associated axes has that fault.</td>
</tr>
</tbody>
</table>
| **Axis Inhibit Status**    | BOOL      | Tag    | If this bit is:  
  - ON — An axis in the coordinate system is inhibited.  
  - OFF — None of the axis in the coordinate system are inhibited.                                             |
| **Command Pos Tolerance Status** | BOOL | Tag | Use the Command Position Tolerance Status bit to determine when a coordinate move is within the Command Position Tolerance.  
The Command Position Tolerance Status bit is set for all term types whenever the distance to programmed endpoint is less than the configured CT value. The bit will remain set after an instruction completes. The bit is reset when a new instruction is started. |
| **Command Position Tolerance** | REAL | GSV | Coordination Units  
The Command Position Tolerance attribute value is a distance unit used when instructions, such as MCLM and MCCM, specify a Termination Type of Command Position. |
| **Config Fault**           | BOOL      | Tag    | The Configuration Fault bit is set when an update operation targeting an axis configuration attribute of an associated motion module has failed. Specific information concerning the Configuration Fault may be found in the Attribute Error Code and Attribute Error ID attributes associated with the motion module. |
| **Coordinate Motion Status** | DINT | Tag | Lets you access the motion status bits for the coordinate system in one 32-bit word.                                                                                                                        |

<table>
<thead>
<tr>
<th>Status</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accel Status</td>
<td>0</td>
</tr>
<tr>
<td>Decel Status</td>
<td>1</td>
</tr>
<tr>
<td>Actual Pos Tolerance Status</td>
<td>2</td>
</tr>
<tr>
<td>Command Pos Tolerance Status</td>
<td>3</td>
</tr>
<tr>
<td>Stopping Status</td>
<td>4</td>
</tr>
<tr>
<td>Reserved</td>
<td>5</td>
</tr>
<tr>
<td>Move Status</td>
<td>6</td>
</tr>
<tr>
<td>Transition Status</td>
<td>7</td>
</tr>
<tr>
<td>Attribute</td>
<td>Data Type</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Accel Status</td>
<td>BOOL</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Coordinate System Auto Tag</td>
<td>SINT</td>
</tr>
<tr>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>Coordinate System Status</td>
<td>DINT</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Decel Status</td>
<td>BOOL</td>
</tr>
<tr>
<td>Dynamics Configuration Bits</td>
<td>DINT</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Coordinate system attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Data Type</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accel Status</td>
<td>BOOL</td>
<td>Tag</td>
<td>Use the Accel Status bit to determine if the coordinated (vectored) motion is currently being commanded to accelerate. The acceleration bit is set when a coordinated move is in the accelerating phase due to the current coordinated move. It clears when the coordinated move stops or the coordinated move is in the decelerating phase.</td>
</tr>
<tr>
<td>Reduced S-Curve Velocity Overshoots</td>
<td></td>
<td></td>
<td>You can cause a coordinate system to overshoot its programmed speed if you decrease the acceleration jerk while the coordinate system is accelerating. This change keeps to overshoot to no more than 50% of the programmed speed.</td>
</tr>
<tr>
<td>Maximum Acceleration</td>
<td>REAL</td>
<td>GSV</td>
<td>Coordination Units / Sec^2 The Maximum Acceleration attribute value is used by motion instructions, such as MCLM and MCCM, to determine the acceleration rate to apply to the coordinate system vector when the acceleration is specified as a percent of the maximum.</td>
</tr>
<tr>
<td>Maximum Deceleration</td>
<td>REAL</td>
<td>GSV</td>
<td>Coordination Units / Sec^2 The Maximum Deceleration attribute value is used by motion instructions, such as MCLM and MCCM, to determine the deceleration rate to apply to the coordinate system vector when the deceleration is specified as a percent of the maximum.</td>
</tr>
<tr>
<td>Maximum Pending Moves</td>
<td>DINT</td>
<td>GSV</td>
<td>The Maximum Pending Moves attribute is used to determine how many Move Pending queue slots should be created as part of the create service for the coordinate system. Limited to a queue of one.</td>
</tr>
<tr>
<td>Maximum Speed</td>
<td>REAL</td>
<td>GSV</td>
<td>Coordination Units / Sec The value of the Maximum Speed attribute is used by various motion instructions, such as MCLM and MCCM, to determine the steady-state speed of the coordinate system vector when the speed is specified as a percent of the Maximum.</td>
</tr>
<tr>
<td>Module Fault</td>
<td>BOOL</td>
<td>Tag</td>
<td>The Module Fault bit attribute is set when a serious fault has occurred with the motion module associated with the selected axis. Usually a module fault affects all axes associated with the motion module. A module fault generally results in the shutdown of all associated axes. Reconfiguration of the motion module is required to recover from a module fault condition.</td>
</tr>
<tr>
<td>Modules Faulted</td>
<td>DINT</td>
<td>Tag</td>
<td>Shows which axes in this coordinate system have a module fault. If this bit is on Then this axis has a module fault</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Motion Status</td>
<td>BOOL</td>
<td>Tag</td>
<td>The Motion Status bit attribute is set indicating that at least one Coordinate Motion instruction is active and the Coordinate System is connected to its associated axes.</td>
</tr>
<tr>
<td>Move Pending Queue Full Status</td>
<td>BOOL</td>
<td>Tag</td>
<td>The move pending queue full bit is set when there is no room in the instruction queue for the next coordinated move instruction. Once there is room in the queue, the bit is cleared.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Data Type</td>
<td>Access</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------</td>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Accel Status</strong></td>
<td>BOOL</td>
<td>Tag</td>
<td>Use the Accel Status bit to determine if the coordinated (vectored) motion is currently being commanded to accelerate. The acceleration bit is set when a coordinated move is in the accelerating phase due to the current coordinated move. It clears when the coordinated move stops or the coordinated move is in the decelerating phase.</td>
</tr>
<tr>
<td><strong>Move Pending Status</strong></td>
<td>BOOL</td>
<td>Tag</td>
<td>The move pending bit is set once a coordinated motion instruction is queued. Once the instruction has begun executing, the bit will be cleared, provided no subsequent coordinated motion instructions have been queued in the mean time. In the case of a single coordinated motion instruction, the status bit may not be detected in Logix Designer application since the transition from queued to executing is faster than the coarse update. The real value of the bit comes in the case of multiple instructions. As long as an instruction is in the instruction queue, the pending bit will be set. This provides the Logix Designer programmer a means of stream-lining the execution of multiple coordinated motion instructions. Ladder logic containing coordinated motion instructions can be made to execute faster when the programmer allows instructions to be queued while a preceding instruction is executing. When the MovePendingStatus bit is clear, the next coordinated motion instruction can be executed (that is, setup in the queue).</td>
</tr>
<tr>
<td><strong>Move Status</strong></td>
<td>BOOL</td>
<td>Tag</td>
<td>The move bit is set when coordinated motion is generating motion for any associated axes. Once coordinated motion is no longer being commanded, the move bit is cleared.</td>
</tr>
<tr>
<td><strong>Move Transition Status</strong></td>
<td>BOOL</td>
<td>Tag</td>
<td>The move transition bit is set once the blend point between two successive coordinated moves has been reach. The bit remains set while the blend of the two moves into one is in process. Once the blend is complete, the move transition bit is cleared.</td>
</tr>
<tr>
<td><strong>Physical Axes Faulted</strong></td>
<td>DINT</td>
<td>Tag</td>
<td>Shows which axes in this coordinate system have a servo axis fault.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>If this bit is on</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td><strong>Physical Axis Fault</strong></td>
<td>BOOL</td>
<td>Tag</td>
<td>If the Physical Axis Fault bit is set, it indicates that there is one or more fault conditions that have been reported by the physical axis. The specific fault conditions can then be determined through access to the fault attributes of the associated physical axis.</td>
</tr>
<tr>
<td><strong>Ready Status</strong></td>
<td>BOOL</td>
<td>Tag</td>
<td>The Ready bit is set when all associated axes are enabled. It is cleared after an MCSD, MGSD or a fault on any of the associated axes.</td>
</tr>
<tr>
<td><strong>Shutdown Status</strong></td>
<td>BOOL</td>
<td>Tag</td>
<td>The Coordinate System bit will be set after an MCSD or MGSD is executed and all associated axes have stopped. An MCSR or a MGSR will reset the coordinate system and clear the bit. Coordinated moves cannot be initiated while this bit is set.</td>
</tr>
<tr>
<td><strong>Stopping Status</strong></td>
<td>BOOL</td>
<td>Tag</td>
<td>The stopping bit is set when an MCS instruction is executed. The bit will remain set until all coordinated motion is stopped. The bit is cleared when all coordinated motion has stopped.</td>
</tr>
<tr>
<td><strong>Transform Source Status</strong></td>
<td>BOOL</td>
<td>Tag</td>
<td>If the bit is: • ON — The coordinate system is the source of an active transform. • OFF — The coordinate system isn’t the source of an active transform.</td>
</tr>
<tr>
<td><strong>Transform Target Status</strong></td>
<td>BOOL</td>
<td>Tag</td>
<td>If the bit is: • ON — The coordinate system is the target of an active transform. • OFF — The coordinate system is not the target of an active transform.</td>
</tr>
</tbody>
</table>
For a three-dimensional Articulated Independent robot, there are four solutions for the same point:

- Left-arm
- Right-arm
- Left-arm mirror
- Right-arm mirror

For example, consider the Cartesian point XYZ (10,0,15). The joint position corresponding to this point has four joint solutions. Two of the solutions are the same as the solutions for the two-dimensional case. The other solutions are mirror image solutions where J1 is rotated 180°.
You can switch the robot from a left-arm solution to a right-arm solution or vice versa. This is done automatically when a joint move is programmed forcing a left/right change to occur. After the change is performed, the robot stays in the new arm solution when Cartesian moves are made. If required, the robot arm solution changes again when another joint move is made.

**Change arm solution example**

Use this example to move the robot from position A (x1,y1) to position B (X2,Y2). At position A, the system is in a left arm solution. When programming a Cartesian move from A (X1,Y1) to B (X2,Y2), the system moves along the straight line from A to B while maintaining a left arm solution. If you want to be at position B in a right-arm solution, move joint J1 from $\theta_1$ to $\theta_2$ and move joint J2 from $\alpha_1$ to $\alpha_2$.

**Singularity**

A singularity occurs when an infinite number of joint positions and mathematical solutions exist for a given Cartesian position. The Cartesian position of a singularity is dependent on the type of the robot geometry, and the size of the link lengths for the robot. Not all robot geometries have singularity positions.

For example, singularities for an Articulated Independent robot occur when:

- The robot manipulator folds its arm back onto itself, and the Cartesian position is at the origin.
- The robot is fully stretched at or very near the boundary of its workspace.

An error condition is generated when a singularity position is reached.
WARNING: Avoid programming your robot towards a singularity position when programming in Cartesian mode. The velocity of the robot increases rapidly as it approaches a singularity position and can result in injury or death to personnel.
Index

A
Arm Solution
   definition of configure 52
Articulated Dependent 56
   define configuration parameters 61
   link lengths 62
   reference frame 56
Articulated Independent 33
   configuration parameters 37
   end effector offsets 39
   establish reference frame 33
   link lengths 38
   work envelope 60

C
Cartesian coordinate system
   program with no orientation 17
Cartesian Gantry 63
   configuration parameters 64
   establish reference frame 63
   identify the work envelope 64
Cartesian H-bot 65
   configuration parameters 66
   establish reference frame 66
   identify the work envelope 66
Collinear Moves
   velocity profiles
   termination types 22
Coordinate System Properties
   Editing 13
Coordinate System Properties dialog box 13

D
Delta Robot
   Maximum Negative Joint Limit Condition 44
   Maximum Positive Joint Limit Condition 44, 45, 46, 47, 48, 49, 50, 51

Delta three-dimensional 41
   calibrate 42
   configuration parameters 45
   maximum negative joint limit condition 44
   maximum positive joint limit condition 44
   reference frame 42
   work envelope 44
Delta two-dimensional 46
   calibrate 47
   configuration parameters 48
   establish the reference frame 47
   work envelope 48

G
Geometries with no orientation support
   Articulated Dependent 56
   Articulated Independent 33
   Cartesian Gantry 63
   Cartesian H-bot 65
   configure 52
   Delta three-dimensional 41
   Delta two-dimensional 46
   SCARA Delta 49
   SCARA Independent 66

M
Motion Calculate Transform Position (MCTP) 17
Motion Coordinated Transform (MCT) 17
Multi-Axis Coordinated Motion Instructions 17
   Motion Calculate Transform Position (MCTP) 17
   Motion Coordinated Change Dynamics (MCCD) 22
   Motion Coordinated Circular Move (MCMM) 17, 18, 22
   Motion Coordinated Linear Move (MCLM) 17, 22
   Motion Coordinated Stop (MCS) 22
   Motion Coordinated Transform (MCT) 17

P
program with no orientation 17
   blended moves with MCLM and MCCM 17
   example ladder diagram 18
S

sample projects  9

SCARA Delta  49
  configuration parameters  51
  establish the reference frame  49
  identify the work envelope  50
  negative X1b offset  51

SCARA Independent  66

Singularity
  planning for
    definition of  88

T

termination type  22
### Rockwell Automation support

Use these resources to access support information.

<table>
<thead>
<tr>
<th>Technical Support Center</th>
<th>Find help with how-to videos, FAQs, chat, user forums, and product notification updates.</th>
<th>rok.auto/support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledgebase</td>
<td>Access Knowledgebase articles.</td>
<td>rok.auto/knowledgebase</td>
</tr>
<tr>
<td>Local Technical Support Phone Numbers</td>
<td>Locate the telephone number for your country.</td>
<td>rok.auto/phonesupport</td>
</tr>
<tr>
<td>Literature Library</td>
<td>Find installation instructions, manuals, brochures, and technical data publications.</td>
<td>rok.auto/literature</td>
</tr>
<tr>
<td>Product Compatibility and Download Center (PCDC)</td>
<td>Get help determining how products interact, check features and capabilities, and find associated firmware.</td>
<td>rok.auto/pcdc</td>
</tr>
</tbody>
</table>

### Documentation feedback

Your comments help us serve your documentation needs better. If you have any suggestions on how to improve our content, complete the form at rok.auto/docfeedback.

### Waste Electrical and Electronic Equipment (WEEE)

At the end of life, this equipment should be collected separately from any unsorted municipal waste.

Rockwell Automation maintains current product environmental information on its website at rok.auto/pec.