



[Instruction Set](#) > Multi-Axis Coordinated Motion Instructions

Search



- ▷ [Quick Start Steps](#)
- ▷ [Logix Designer](#)
- ▷ [Module Information](#)
- ◀ [Instruction Set](#)

[Logix 5000 Controllers](#)

[Instruction and Application Considerations](#)

[Logix Designer Application Instruction Set](#)

[Interpret the Attribute Tables](#)

[Array Concepts](#)
- ▷ [CIP Axis Attributes](#)
- ▷ [Module Configuration Attributes](#)
- [Bit Addressing](#)
- [Common Attributes](#)
- [Data Conversions](#)
- [Elementary data types](#)
- [LINT data types](#)
- [Floating Point Values](#)
- [Immediate values](#)
- [Index Through Arrays](#)
- [Math Status Flags](#)
- [Motion Error Codes \(.ERR\)](#)
- [Structures](#)
- ▷ [Equipment Sequence instructions](#)
- ▷ [Equipment Phase Instructions](#)
- ▷ [Alarm Instructions](#)
- ▷ [Advanced Math Instructions](#)
- ▷ [Array \(File\)/Misc Instructions](#)
- ▷ [Array \(File\)/Shift Instructions](#)
- ▷ [ASCII Conversion Instructions](#)
- ▷ [ASCII Serial Port Instructions](#)
- ▷ [ASCII String Instructions](#)
- ▷ [Bit Instructions](#)
- ▷ [Compare Instructions](#)
- ▷ [Debug Instructions](#)
- ▷ [Drives Instructions](#)
- ▷ [Drive Safety Instructions](#)
- ▷ [For/Break Instructions](#)
- ▷ [Filter Instructions](#)
- ▷ [Function Block Attributes](#)
- ▷ [Structured Text Attributes](#)
- ▷ [Compute/Math Instructions](#)
- ▷ [Move/Logical Instructions](#)
- ▷ [Input/Output Instructions](#)
- ▷ [License Instructions](#)
- ▷ [Math Conversion Instructions](#)
- ▷ [Metal Form Instructions](#)
- ▷ [Motion Configuration](#)

Multi-Axis Coordinated Motion Instructions

Use the Multi-Axis Coordinated Motion Instructions to move up to six axes in a coordinate system.

Available Instructions

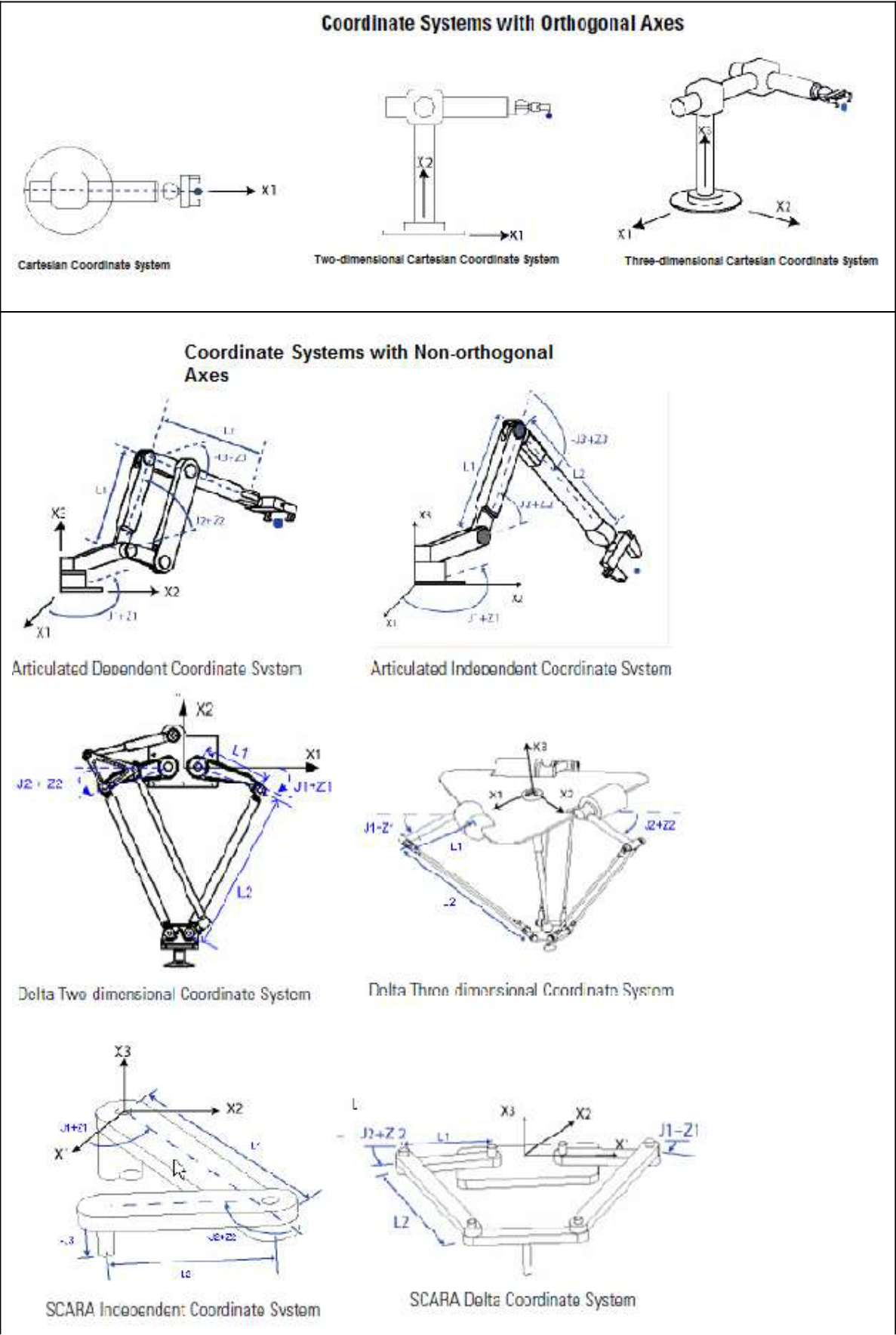
Ladder Diagram and Structured Text

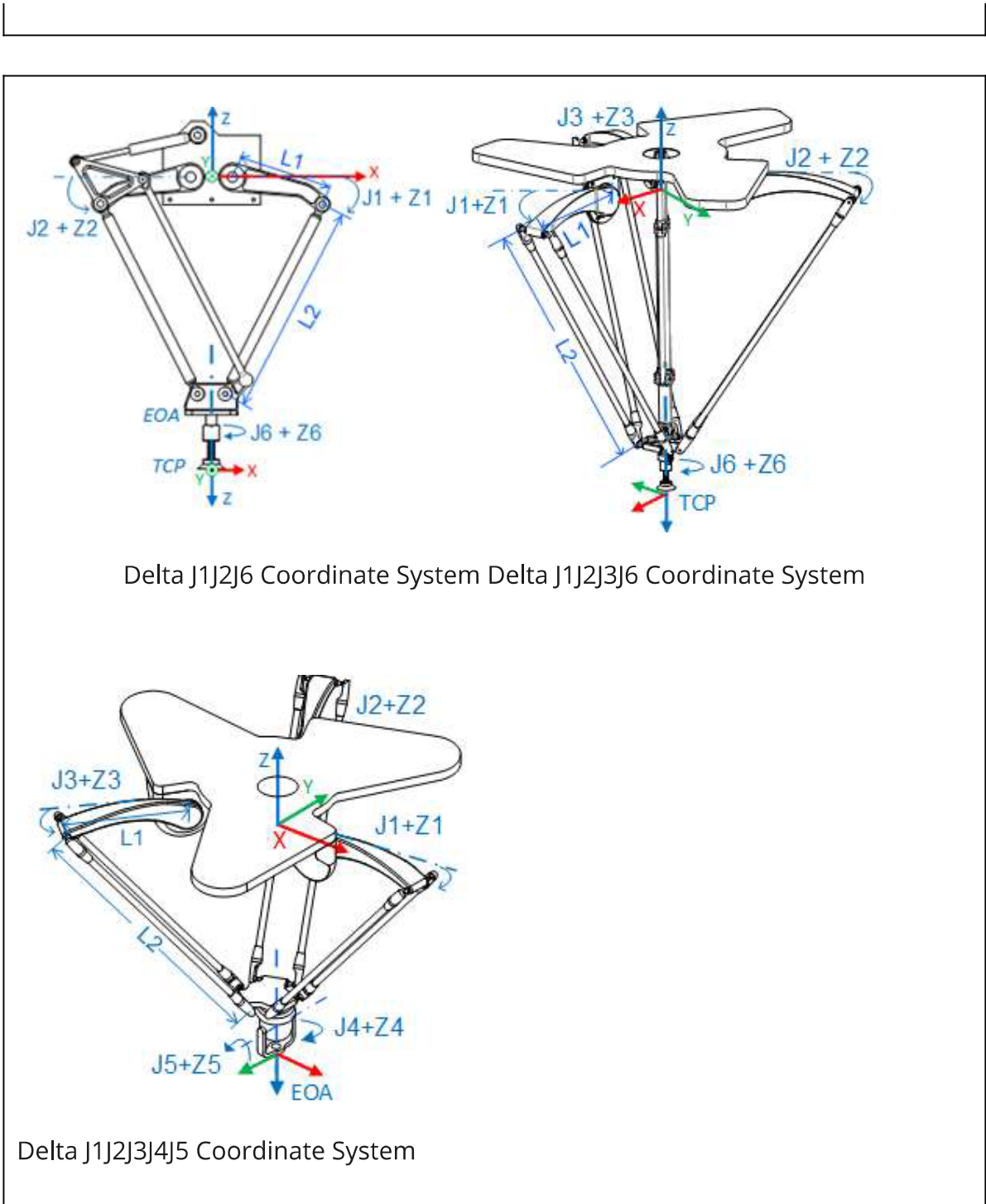
MCS	MCLM	MCCM	MCCD	MCSD	MCSR	MCT	MCTP
MDCC	MCTPO	MCPM	MCTO				

Function Block

Not available

The following are coordinate system examples.





The multi-axis coordinated motion instructions are:

If:	Use this instruction:
Stopping the axes of a Coordinate System or cancel a transform.	MCS
Initiating a single or multi-dimensional linear Coordinated move for the specified axes within a Cartesian Coordinate System.	MCLM
Initiating a two- or three-dimensional circular Coordinated move for the specified axes within a Cartesian Coordinate System.	MCCM
Initiating a change in path dynamics for Coordinate motion active on the specified Coordinate System.	MCCD
Initiating a controlled shutdown of all of the axes of the specified Coordinate System.	MCSD
Initiating 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.	MCSR
Starting a transform that links two Coordinate Systems together. This instruction is only available on supported controllers.	MCT
Calculating the position of one Coordinate System with respect to another Coordinate System.	MCTP

Instructions

- ▷ [Motion Event Instructions](#)
- ▷ [Motion Group Instructions](#)
- ▷ [Motion Move Instructions](#)
- ▷ [Motion State Instructions](#)
- ▲ [Multi-Axis Coordinated Motion Instructions](#)
 - [Master Driven Coordinated Control \(MDCC\)](#)
 - [Motion Calculate Transform Position \(MCTP\)](#)
 - [Motion Coordinated Change Dynamics \(MCCD\)](#)
 - [Motion Coordinated Circular Move \(MCCM\)](#)
 - [Motion Coordinated Transform with Orientation \(MCTO\)](#)
 - [Motion Coordinated Path Move \(MCPM\)](#)
 - [Motion Calculate Transform Position with Orientation \(MCTPO\)](#)
 - [Motion Coordinated Linear Move \(MCLM\)](#)
 - [Motion Coordinated Shutdown \(MCSD\)](#)
 - [Motion Coordinated Shutdown Reset \(MCSR\)](#)
 - [Motion Coordinated Stop \(MCS\)](#)
 - [Motion Coordinated Transform \(MCT\)](#)
 - [Speed, acceleration, deceleration, and jerk enumerations for coordinated motion](#)
 - [Status Bits for Motion Instructions \(MCLM, MCCM\) when MDCC Is Active](#)
 - [Change between master driven and time driven modes for Coordinated Motion instructions](#)
 - [Choose a Termination Type](#)
 - [Common Action Table for Slave Coordinate System and Master Axis](#)
 - [Input and Output](#)
 - [Parameters Structure for Coordinate System Motion Instructions](#)
 - [Returned Calculated Data Parameter for Coordinated System Motion Instruction](#)
- ▷ [Logical and Move Instructions](#)

This instruction is only available on supported controllers.	
Defining a Master and Slave relationship between a Master Axis and a Coordinate System. Coordinate motion instructions MCLM and MCCM executed on a Slave Coordinate System will be synchronized to a Master Axis.	MDCC
Calculating the position of a point in one coordinate system to the equivalent point in a second coordinate system.	MCTPO
Starting a multi-dimensional coordinated path move for the specified axes within a Cartesian coordinate system.	MCPM
Starting a transform that links two coordinate systems together with orientation control.	MCTO

- [Register more instructions](#)
- ▷

[Program Control Instructions](#)
- ▷

[Sequencer Instructions](#)
- ▷

[Special Instructions](#)
- ▷

[Timer and Counter Instructions](#)
- ▷

[Trigonometric Instructions](#)
- ▷

[Process Control Instructions](#)
- ▷

[Select/Limit Instructions](#)
- ▷

[Sequential Function Chart \(SFC\) Instructions](#)
- ▷

[Statistical Instructions](#)
- ▷

[Safety Instructions](#)
- ▷

[Studio 5000 Logix Designer Glossary](#)

Using Different Termination Types When Blending Instructions

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

- MovePendingStatus
- MovePendingQueueFullStatus

For example, the following ladder diagram uses Coordinate System cs1 to blend Move1 into Move2.

Example Ladder Diagram for Blended Instructions

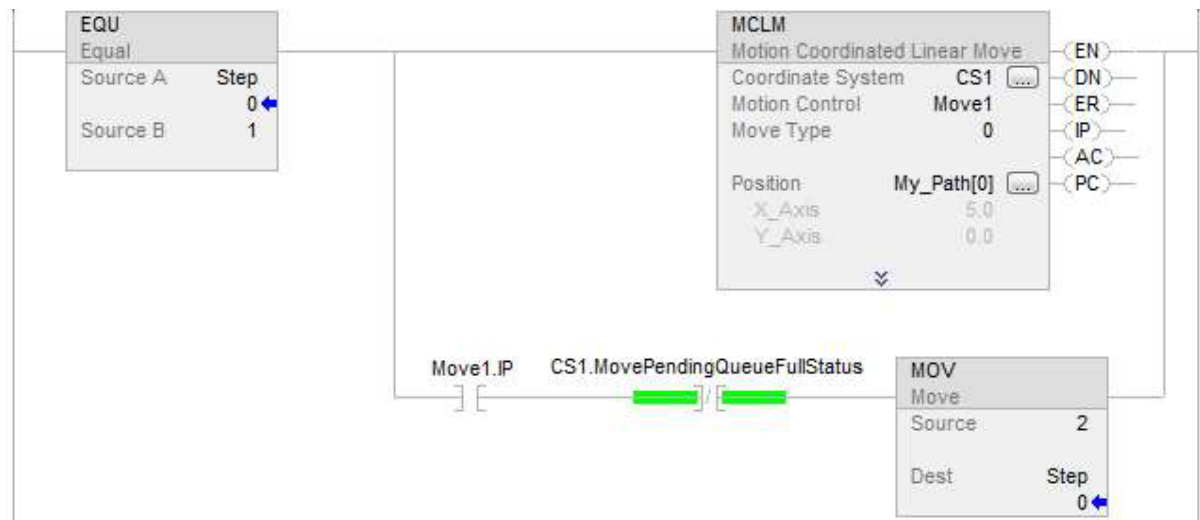
If Step=1 then

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

Step=2



If Step=2 then

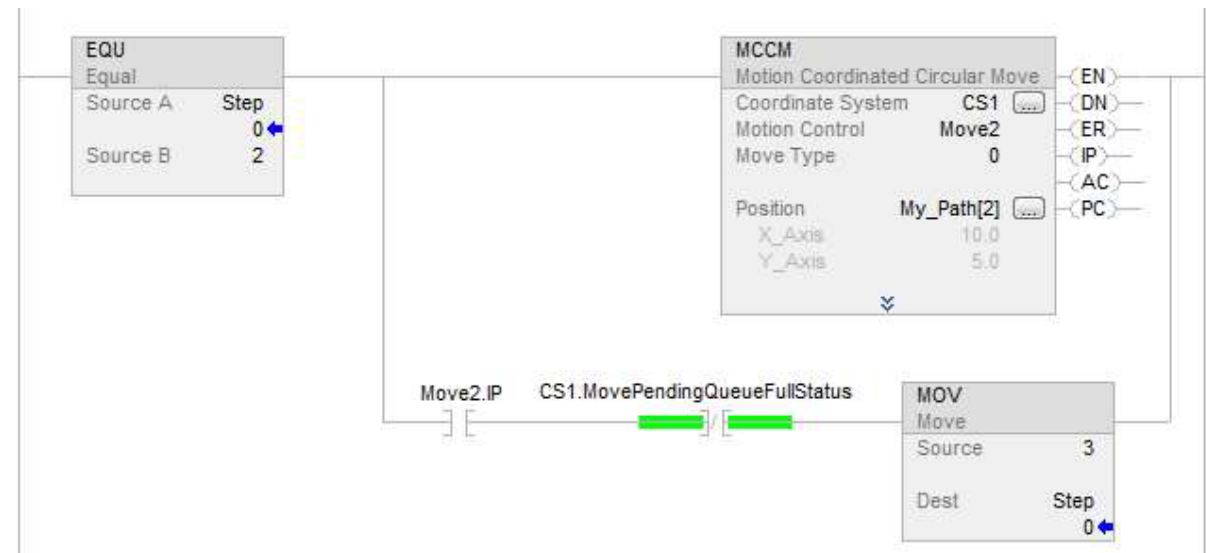
Move1 is already happening.

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.



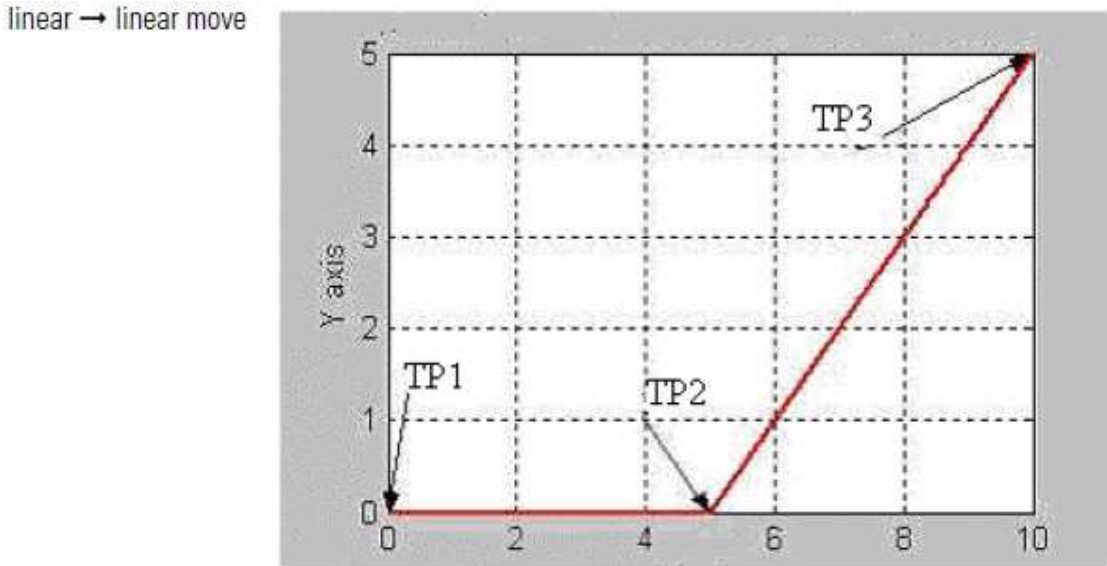
When an instruction completes, it is removed from the queue and there is space for another instruction to enter the queue. Both bits always have the same value because you can queue only one pending instruction at a time. If the application requires several instructions to be executed in sequence, then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

Bit Parameters

When	Then
One instruction is active and a second instruction is pending in the queue.	<ul style="list-style-type: none">MovePendingStatus bit = 1MovePendingQueueFullStatus bit = 1You can't queue another instruction
An active instruction completes and leaves the queue.	<ul style="list-style-type: none">MovePendingStatus bit = 0MovePendingQueueFullStatus bit = 0You can queue another instruction

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

Bit States at Transition Points of Blended Move Using Actual Tolerance or No Settle



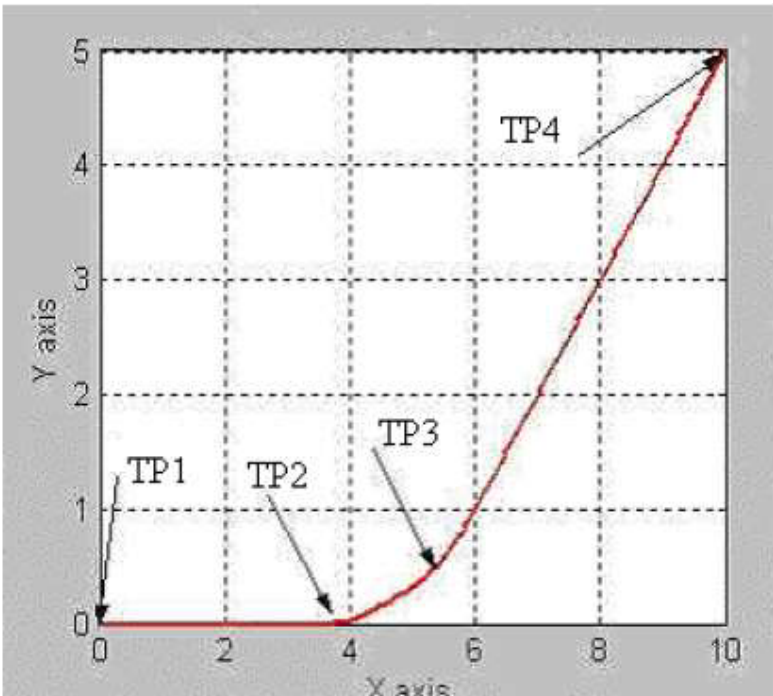
The following table shows the bit status at the various transition points shown in the preceding graph with termination type of either Actual Tolerance or No Settle.

Bit	TP1	TP2	TP3
-----	-----	-----	-----

Move1.DN	T	T	T
Move1.IP	T	F	F
Move1.AC	T	F	F
Move1.PC	F	T	T
Move2.DN	T	T	T
Move2.IP	T	T	F
Move2.AC	F	T	F
Move2.PC	F	F	T
cs1.MoveTransitionStatus	F	F	F
cs1.MovePendingStatus	T	F	F
cs1.MovePendingQueueFullStatus	T	F	F

Bit States at Transition Points of Blended Move Using No Decel

linear → linear move



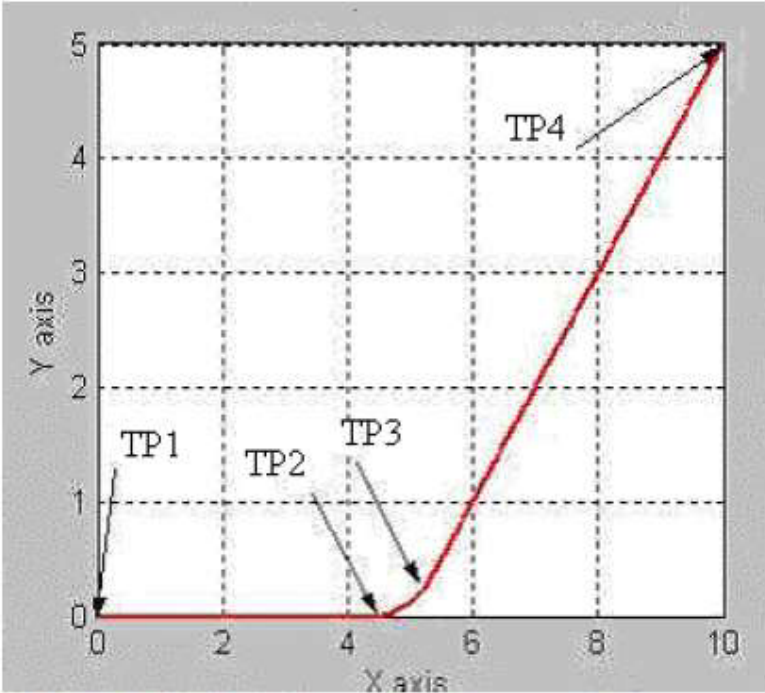
The following table shows the bit status at the various transition points shown in the preceding graph with termination type of No Decel. For No Decel termination type distance-to-go for transition point TP2 is equal to deceleration distance for the Move1 instruction. If Move 1 and Move 2 are collinear, then Move1.PC will be true at TP3 (the programmed end-point of first move).

Bit	TP1	TP2	TP3	TP4
Move1.DN	T	T	T	T
Move1.IP	T	F	F	F
Move1.AC	T	F	F	F
Move1.PC	F	T	T	T

Move1.PC	F	T	T	T
Move2.DN	T	T	T	T
Move2.IP	T	T	T	F
Move2.AC	F	T	T	F
Move2.PC	F	F	F	T
cs1.MoveTransitionStatus	F	T	F	F
cs1.MovePendingStatus	T	F	F	F
cs1.MovePendingQueueFullStatus	T	F	F	F

Bit States at Transition Points of Blended Move Using Command Tolerance

linear → linear move



The following 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.

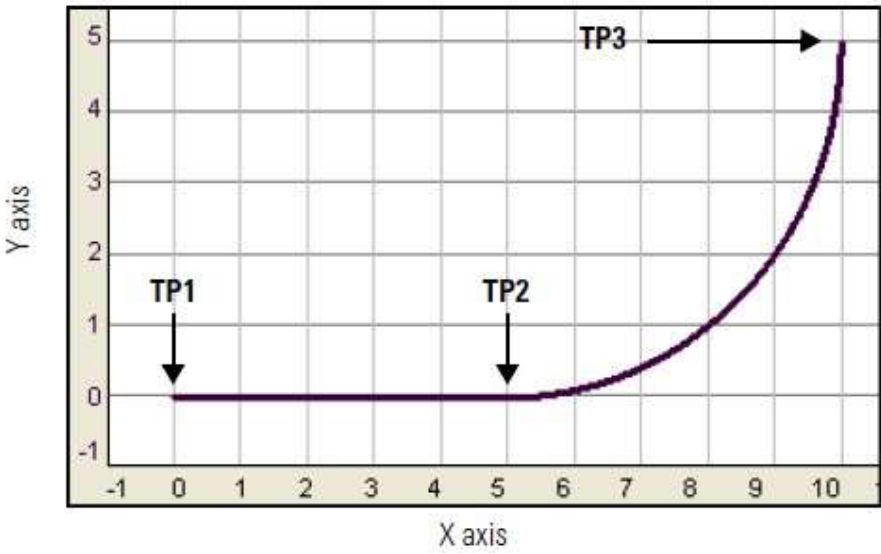
Bit	TP1	TP2	TP3	TP4
Move1.DN	T	T	T	T
Move1.IP	T	F	F	F
Move1.AC	T	F	F	F
Move1.PC	F	T	T	T
Move2.DN	T	T	T	T
Move2.IP	T	T	T	F
Move2.AC	F	T	T	F
Move2.PC	F	F	F	T

--	--	--	--	--

cs1.MoveTransitionStatus	F	T	F	F
cs1.MovePendingStatus	T	F	F	F
cs1.MovePendingQueueFullStatus	T	F	F	F

Bit States at Transition Points of Blended Move Using Follow Contour Velocity Constrained or Unconstrained

linear → circular move



The following table shows the bits status at the transition points.

Bit	TP1	TP2	TP3
Move1.DN	T	T	T
Move1.IP	T	F	F
Move1.AC	T	F	F
Move1.PC	F	T	T
Move2.DN	T	T	T
Move2.IP	T	T	F
Move2.AC	F	T	F
Move2.PC	F	F	T
cs1.MoveTransitionStatus	F	F	F
cs1.MovePendingStatus	T	F	F
cs1.MovePendingQueueFullStatus	T	F	F

See also

[Choose a Termination Type](#)

[Speed, Acceleration, Deceleration, and Jerk Enumerations for Coordinated Motion](#)

CS1, Bit 6, Motion Control, Bit 6 (MCM, MCMN, L, MDC1, A1)

[Status Bits for Motion Instructions when \(MCLM, MCLM\) when MDCC is Active](#)

[Input and Output Parameters Structure for Coordinate System Motion Instructions](#)

[Changing Between Master Driven and Time Driven Modes for Coordinated Motion Instructions](#)

Copyright © 2019 Rockwell Automation Technologies, Inc. All Rights Reserved.

[How are we doing?](#)