Logix 5000 Controllers Motion Instructions

1756 ControlLogix, 1756 GuardLogix, 1769 CompactLogix, 1769 Compact GuardLogix, 1789 SoftLogix, 5069 CompactLogix, 5069 Compact GuardLogix, Studio 5000 Logix Emulate

Rockwell Automation Publication MOTION-RM002M-EN-P-November 2023
Supersedes MOTION-RM002L-EN-P-November 2022

Rockwell Automation

Reference Manual

Original Instructions

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).

Rockwell Automation recognizes that some of the terms that are currently used in our industry and in this publication are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

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

Global changes

This table identifies changes that apply to all information about a subject in the manual and the reason for the change. For example, the addition of new supported hardware, a software design change, or additional reference material would result in changes to all of the topics that deal with that subject.

None in this release.

New or enhanced features

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.

Subject	Reason
Motion Coordinated Circular Move on page 432	Added information about the REF_To type.
Motion Coordinated Change Dynamics on page 409	Added information about the REF_To type.
Motion Coordinated Linear Move on page 456	Added information about the REF_To type.
Motion Coordinated Path Move on page 394	Added information about the REF_To type.
Motion Coordinated Stop on page 485	Added information about the REF_To type.
Motion Coordinated Shutdown on page 477	Added information about the REF_To type.
Motion Coordinated Shutdown Reset on page 481	Added information about the REF_To type.
Motion Coordinated Transform on page 494	Added information about the REF_To type.
Motion Coordinated Transform with Orientation on page 381	Added information about the REF_To type.
Motion Calculate Transform Position with Orientation on page 422	Added information about the REF_To type.
Motion Calculate Transform Position on page 374	Added information about the REF_To type.
Master Driven Coordinated Control on page 368	Added information about the REF_To type.
Motion Coordinated Circular Move on page 432	Added note regarding action on references.
Motion Coordinated Linear Move on page 456	Added note regarding action on references.
Motion Arm Output Cam on page 261	Added information about the REF_To type.
Motion Arm Registration on page 261	Added information about the REF_To type.
Motion Arm Watch on page 250	Added information about the REF_To type.
Motion Disarm Output Cam on page 307	Added information about the REF_To type.
Motion Disarm Registration on page 269	Added information about the REF_To type.
Motion Disarm Watch on page 256	Added information about the REF_To type.
Motion Apply Axis Tuning on page 316	Added information about the REF_To type.
Motion Apply Hookup Diagnostics on page 334	Added information about the REF_To type.
Motion Run Axis Tuning on page 324	Added information about the REF_To type.
Motion Run Hookup Diagnostics on page 341	Added information about the REF_To type.

Subject	Reason
Motion Axis Gear on page 115	Added information about the REF_To type.
Motion Axis Home on page 84	Added information about the REF_To type.
Motion Axis Jog on page 91	Added information about the REF_To type.
Motion Axis Move on page 101	Added information about the REF_To type.
Motion Axis Position Cam on page 158	Added information about the REF_To type.
Motion Axis Stop on page 74	Added information about the REF_To type.
Motion Axis Time Cam on page 193	Added information about the REF_To type.
Motion Change Dynamics on page 128	Added information about the REF_To type.
Motion Redefine Position on page 139	Added information about the REF_To type.
Motion Axis Fault Reset on page 27	Added information about the REF_To type.
Motion Axis Shutdown on page 32	Added information about the REF_To type.
Motion Axis Shutdown Reset on page 37	Added information about the REF_To type.
Motion Direct Drive Off on page 42	Added information about the REF_To type.
Motion Direct Drive On on page 47	Added information about the REF_To type.
Motion Drive Start on page 53	Added information about the REF_To type.
Motion Servo Off on page 61	Added information about the REF_To type.
Motion Servo On on page 66	Added information about the REF_To type.

Use this locator to find the applicable Logix5000 controllers instruction manual for each instruction.

Logix5000 Controllers General Instructions Reference Manual 1756-RM003	Logix5000 Controllers Advanced Process Control and Drives and Equipment Phase and Sequence Instructions Reference Manual 1756-RM006	Logix5000 Controllers Motion Instructions Reference Manual MOTION-RM002	PlantPAx Process Control Instructions PROCES-RM215
Absolute Value (ABS)	Alarm (ALM)	Master Driven Coordinated Control (MDCC)	Process Analog HART (PAH)
Add (ADD)	Attach to Equipment Phase (PATT)	Motion Apply Axis Tuning (MAAT)	Process Analog Input (PAI)
Analog Alarm (ALMA)	Attach to Equipment Sequence (SATT)	Motion Apply Hookup Diagnostics (MAHD)	Process Dual Sensor Analog Input (PAID)
Always False (AFI)	Coordinated Control (CC)	Motion Arm Output Cam (MAOC)	Process Multi Sensor Analog Input (PAIM)
Arc Cosine (ACS, ACOS)	D Flip-Flop (DFF)	Motion Arm Registration (MAR)	Process Analog Output (PAO)
Arc Sine (ASN, ASIN)	Deadtime (DEDT)	Motion Arm Watch (MAW)	Process Boolean Logic (PBL)
Arc Tangent (ATN, ATAN)	Derivative (DERV)	Motion Axis Fault Reset (MAFR)	Process Command Source (PCMDSRC)
ASCII Chars in Buffer (ACB)	Detach from Equipment Phase (PDET)	Motion Axis Gear (MAG)	Process Discrete 2-, 3-, or 4-State Device (PD4SD)
ASCII Clear Buffer (ACL)	Detach from Equipment Sequence (SDET)	Motion Axis Home (MAH)	Process Deadband Controller (PDBC)
ASCII Handshake Lines (AHL)	Discrete 3-State Device (D3SD)	Motion Axis Jog (MAJ)	Process Discrete Input (PDI)
ASCII Read (ARD)	Discrete 2-State Device (D2SD)	Motion Axis Move (MAM)	Process Discrete Output (PDO)
ASCII Read Line (ARL)	Enhanced PID (PIDE)	Motion Axis Position Cam (MAPC)	Process Dosing (PDOSE)
ASCII Test for Buffer Line (ABL)	Enhanced Select (ESEL)	Motion Axis Stop (MAS)	Process Analog Fanout (PFO)
ASCII Write (AWT)	Equipment Phase Clear Failure (PCLF)	Motion Axis Time Cam (MATC)	Process High or Low Selector (PHLS)
ASCII Write Append (AWA)	Equipment Phase Command (PCMD)	Motion Axis Shutdown (MASD)	Process Interlocks (PINTLK)
Bit Field Distribute (BTD)	Equipment Phase External Request (PXRQ)	Motion Axis Shutdown Reset (MASR)	Process Lead Lag Standby Motor Group (PLLS)
Bit Field Distribute with Target (BTDT)	Equipment Phase Failure (PFL)	Motion Calculate Cam Profile (MCCP)	Process Motor (PMTR)
Bit Shift Left (BSL)	Equipment Phase New Parameters (PRNP)	Motion Coordinated Path Move (MCPM)	Process Permissives (PPERM)
Bit Shift Right (BSR)	Equipment Phase Override Command (POVR)	Motion Calculate Slave Values (MCSV)	Process Proportional + Integral + Derivative (PPID)
Bitwise And (AND)	Equipment Phase Paused (PPD)	Motion Coordinated Transform with Orientation (MCTO)	Process Pressure/Temperature Compensated Flow (PPTC)
Bitwise (NOT)	Equipment Sequence Assign Sequence Identifier (SASI)	Motion Calculate Transform Position (MCTP)	Process Restart Inhibit (PRI)
Bitwise (OR)	Equipment Sequence Clear Failure (SCLF)	Motion Calculate Transform Position with Orientation (MCTPO)	Process Run Time and Start Counter (PRT)
Boolean AND (BAND)	Equipment Sequence command (SCMD)	Motion Change Dynamics (MCD)	Process Tank Strapping Table (PTST)
Boolean Exclusive OR (BXOR)	Equipment Sequence Override (SOVR)	Motion Coordinated Change Dynamics (MCCD)	Process Valve (PVLV)
Boolean NOT (BNOT)	Function Generator (FGEN)	Motion Coordinated Circular Move (MCCM)	Process Valve Statistics (PVLVS)
Boolean OR (BOR)	High Pass Filter (HPF)	Motion Coordinated Linear Move (MCLM)	
Break (BRK)	High/Low Limit (HLL)	Motion Coordinated Shutdown (MCSD)	
Breakpoints (BPT)	HMI Button Control (HMIBC)	Motion Coordinated Shutdown Reset (MCSR)	
Clear (CLR)	Integrator (INTG)	Motion Coordinated Stop (MCS)	
Compare (CMP)	Internal Model Control (IMC)	Motion Coordinated Transform (MCT)	
Convert to BCD (TOD)	JK Flip-Flop (JKFF)	Motion Direct Drive Off (MDF)	

Logix5000 Controllers General Instructions Reference Manual 1756-RM003	Logix5000 Controllers Advanced Process Control and Drives and Equipment Phase and Sequence Instructions Reference Manual 1756-RM006	Logix5000 Controllers Motion Instructions Reference Manual MOTION-RM002	PlantPAx Process Control Instructions PROCES-RM215
Convert to Integer (FRD)	Lead-Lag (LDLG)	Motion Direct Drive On (MDO)	
Copy File (COP), Synchronous Copy File (CPS)	Low Pass Filter (LPF)	Motion Direct Start (MDS)	
Cosine (COS)	Maximum Capture (MAXC)	Motion Disarm Output Cam (MDOC)	
Compute (CPT)	Minimum Capture (MINC)	Motion Disarm Registration (MDR)	
Count down (CTD)	Modular Multivariable Control (MMC)	Motion Disarm Watch (MDW)	
Count up (CTU)	Moving Average (MAVE)	Motion Group Shutdown (MGSD)	
Count up/down CTUD	Moving Standard Deviation (MSTD)	Motion Group Shutdown Reset (MGSR)	
Data Transition (DTR)	Multiplexer (MUX)	Motion Group Stop (MGS)	
Degrees (DEG)	Notch Filter (NTCH)	Motion Group Strobe Position (MGSP)	
Diagnostic Detect (DDT)	Phase State Complete (PSC)	Motion Redefine Position (MRP)	
Digital Alarm (ALMD)	Position Proportional (POSP)	Motion Run Axis Tuning (MRAT)	
DINT To String (DTOS)	Proportional + Integral (PI)	Motion Run Hookup Diagnostics (MRHD)	
Divide (DIV)	Pulse Multiplier (PMUL)	Motion Servo Off (MSF)	
End of Transition (EOT)	Ramp/Soak (RMPS)	Motion Servo On (MSO)	
Equal to (EQU)	Rate Limiter (RLIM)		
File Arithmetic (FAL)	Reset Dominant (RESD)		
File Bit Comparison (FBC)	Scale (SCL)		
FIFO Load (FFL)	S-Curve (SCRV)		
FIFO Unload (FFU)	Second-Order Controller (SOC)		
File Average (AVE)	Second-Order Lead Lag (LDL2)		
File Standard Deviation (STD)	Select (SEL)		
File Fill (FLL)	Selected Negate (SNEG)		
File Sort (SRT)	Selected Summer (SSUM)		
Find String (FIND)	Set Dominant (SETD)		
For (FOR)	Split Range Time Proportional (SRTP)		
File Search and Compare (FSC)	Totalizer (TOT)		
Get System Value (GSV) and Set System Value (SST)	Up/Down Accumulator (UPDN)		
Greater Than or Equal to (GEQ)			
Greater than (GRT)			
Insert String (INSERT)			
Immediate Output (IOT)			
Is Infinity (IsINF)			
Is Not a Number (IsNAN)			
Jump to Label (JMP) and Label (LBL)			
Jump to Subroutine (JSR), Subroutine (SBR), and Return (RET)			
Jump to External Routine (JXR)			
Less Than (LES)			
Less Than or Equal to (LEQ)			
LIFO Load (LFL)			
LIFO Unload (LFU)			
License Validation (LV)			

Logix5000 Controllers General Instructions Reference Manual 1756-RM003	Logix5000 Controllers Advanced Process Control and Drives and Equipment Phase and Sequence Instructions Reference Manual 1756-RM006	Logix5000 Controllers Motion Instructions Reference Manual MOTION-RM002	PlantPAx Process Control Instructions PROCES-RM215
Limit (LIM)			
Log Base (LOG)			
Lower to Case (LOWER)			
Masked Move (MVM)			
Masked Move with Target (MVMT)			
Master Control Reset (MCR)			
Masked Equal to (MEQ)			
Message (MSG)			
Middle String (MID)			
Modulo (MOD)			
Move (MOV)			
Multiply (MUL)			
Natural Log (LN)			
Negate (NEG)			
Not Equal to (NEQ)			
No Operation (NOP)			
One Shot (ONS)			
One Shot Falling (OSF)			
One Shot Falling with Input (OSFI)			
One Shot Rising (OSR)			
One Shot Rising with Input (OSRI)			
Output Energize (OTE)			
Output Latch (OTL)			
Output Unlatch (OTU)			
Proportional Integral Derivative (PID)			
Radian (RAD)			
Real to String (RTOS)			
Reset (RES)			
Reset SFC (SFR)			
Return (RET)			
Retentive Timer On (RTO)			
Retentive Timer On with Reset (RTOR)			
Pause SFC (SFP)			
Size In Elements (SIZE)			
Sequencer Input (SQI)			
Sequencer Load (SQL)			
Sequencer Output (SQO)			
Sine (SIN)			
Square Roost (SQR/SQRT)			
String Concatenate (CONCAT)			
String Delete (DELETE)			
String to DINT (STOD)			
String to REAL (STOR)			

Logix5000 Controllers General Instructions Reference Manual 1756-RM003	Logix5000 Controllers Advanced Process Control and Drives and Equipment Phase and Sequence Instructions Reference Manual 1756-RM006	Logix5000 Controllers Motion Instructions Reference Manual MOTION-RM002	PlantPAx Process Control Instructions PROCES-RM215
Swap Byte (SWPB)			
Subtract (SUB)			
Tangent (TAN)			
Timer Off Delay (TOF)			
Timer Off Delay with Reset (TOFR)			
Timer On Delay (TON)			
Timer On Delay with Reset (TONR)			
Temporary End (TND)			
Tracepoints (TPT)			
Trigger Event Task (EVENT)			
Truncate (TRN)			
Unknown Instruction (UNK)			
Upper Case (UPPER)			
User Interrupt Disable (UID)/User Interrupt Enable (UIE)			
X to the Power of Y (XPY)			
Examine if Closed (XIC)			
Examine If Open (XIO)			
Bitwise Exclusive (XOR)			

Summary of changes
Instruction Locator
Preface
Studio 5000 environment 15
Additional resources 15
Legal Notices 16

Chapter 1

Understand Instruction Timing
Process Type Instructions 20
Write a Motion Application Program 22

Chapter 2

Motion State Instructions
Motion State Instructions 23
- Motion Axis Fault Reset (MAFR) 27
- MAFR Flow Chart (True) 32
- Motion Axis Shutdown (MASD) 32
- MASD Flow Chart (True) 37
- Motion Axis Shutdown Reset (MASR) 37
- MASR Flow Chart (True) 42
- Motion Direct Drive Off (MDF) 42
- MDF Flow Chart (True) 47
- Motion Direct Drive On (MDO) 47
- MDO Flow Chart (True) 53
- Motion Drive Start (MDS) 53
- Motion Servo Off (MSF) 61
- MSF Flow Chart (True) 65
- Motion Servo On (MSO) 66
- MSO Flow Chart (True) 71

Chapter 3

Motion Move Instructions
Motion Move Instructions 73
- Motion Axis Stop (MAS) 74
- Motion Axis Home (MAH) 84
- MAH Flow Chart (True) 91
- Motion Axis Jog (MAJ) 91
- Motion Axis Move (MAM) 101
- Motion Axis Gear (MAG) 115
- MAG Flow Chart (True) 128
- Motion Change Dynamics (MCD) 128
- MCD Flow Chart (True) 139
- Motion Redefine Position (MRP) 139
- MRP Flow Chart (True) 146
- Motion Calculate Cam Profile (MCCP) 146

Motion Calculate Slave Values (MCSV) 154
Motion Axis Position Cam (MAPC) 158
MAPC Flow Chart (True) 193
Motion Axis Time Cam (MATC) 193
MATC Flow Chart (True) 224

Chapter 4

Motion Group Instructions

Motion Group Instructions 225
Motion Group Stop (MGS) 226
MGS Flow Chart (True) 233
Motion Group Shutdown (MGSD) 233
MGSD Flow Chart (True) 239
Motion Group Shutdown Reset (MGSR) 239
MGSR Flow Chart (True) 244
Motion Group Strobe Position (MGSP) 244
MGSP Flow Chart (True) 248

Chapter 5

Motion Event Instructions

Motion Event Instructions 249
Motion Arm Watch (MAW) 250
MAW Flow Chart (True) 255
Understand a Programming example 255
Motion Disarm Watch (MDW) 256
MDW Flow Chart (True) 261
Motion Arm Registration (MAR) 261
MAR Flow Chart (True) 269
Motion Disarm Registration (MDR) 269
Motion Arm Output Cam (MAOC) 273
Scheduled Output Module 294
Specifying the Output Cam 300
Specifying Output Compensation 304
MAOC Flow Chart (True) 307
Motion Disarm Output Cam (MDOC) 307
MDOC Flow Chart (True) 313

Chapter 6

Motion Configuration Instructions

Motion Configuration Instructions 315
Motion Apply Axis Tuning (MAAT) 316
MAAT Flow Chart (True) 324
Motion Run Axis Tuning (MRAT) 324
MRAT Flow Chart (True) 334
Motion Apply Hookup Diagnostics (MAHD) 334

MAHD Flow Chart (True) 341
Motion Run Hookup Diagnostics (MRHD) 341
MRHD Flow Chart (True) 356
Modify Motion Configuration Parameters 357

Chapter 7

Multi-Axis Coordinated Motion Instructions

Multi-Axis Coordinated Motion Instructions 359
Master Driven Coordinated Control (MDCC) 368
Motion Calculate Transform Position (MCTP) 374
Motion Coordinated Transform with Orientation (MCTO) 381
Motion Coordinated Path Move (MCPM) 394
Motion Coordinated Change Dynamics (MCCD) 409
Motion Calculate Transform Position with Orientation (MCTPO) 422
Motion Coordinated Circular Move (MCCM) 432
Motion Coordinated Linear Move (MCLM) 456
Motion Coordinated Shutdown (MCSD) 477
Motion Coordinated Shutdown Reset (MCSR) 481
Motion Coordinated Stop (MCS) 485
Motion Coordinated Transform (MCT) 494
Speed, acceleration, deceleration, and jerk enumerations 508
Returned Calculated Data Parameter 513
Status Bits for MCLM, MCCM when MDCC Is Active 514
Change between master and time driven modes 516
Choose a Termination Type 517
Common Action Table for Slave Coordinate System and Master Axis 526
Input and Output Parameters Structure 527
Returned Calculated Data Parameter 538

Chapter 8

Master Driven Speed Control Functionality

Master Driven Axis Control (MDAC) 541
Change between Master Driven and Time Driven Modes for Single Axis Motion instructions 550
Action Table for Slave and Master Axis 552
Input and Output Parameters Structure 554
Speed, Acceleration, Deceleration, and Jerk Enumerations 560
Time Based Planning 569
Status Bits for Motion Instructions (MAM, MATC, MAJ) When MDAC Is Active 572

Chapter 9

Program a velocity profile and jerk rate and tune an S-Curve Profile

Definition of Jerk 575
Choose a Profile 576
Velocity Profile Effects 577
Tune an S-Curve Profile 577

Chapter 10

Motion Error Codes, faults, and attributes

Motion Error Codes (.ERR) 581
Handle Motion Faults 594
Motion Attributes 595
Motion Status and Configuration Parameters 639
Troubleshoot Axis Motion 639

Chapter 11

Overview of motion-related data types

CAM Structure 657
CAM_PROFILE Structure 657
MOTION_GROUP Structure 658
MOTION_INSTRUCTION Data Type 660
OUTPUT_CAM Structure 662
OUTPUT_COMPENSATION Structure 664

Chapter 12

Overview of Structured Text Programming

Structured Text Syntax 665
ST Components: Assignments 666
Specify a non-retentive assignment 667
ASCII character to string member 668
ST Components: Expressions 668
Use operators and functions 669
Use relational operators 670
Use logical operators 672
Use bitwise operators 672
Determine the order of execution 673
ST Components: Instructions 673
ST Components: Constructs 674
Character string literals 675
String Types 676
IF_THEN 676
CASE_OF 679
FOR_DO 681
WHILE_DO 684
REPEAT_UNTIL 686
ST Components: Comments 689

Chapter 13

Common attributes for Motion instructions

Common Attributes 691
Index Through Arrays 691
Immediate values 692
Floating Point Values 693
Elementary data types 694
Data Conversions 697
Math Status Flags 700
Bit Addressing 702

Index

This manual provides a programmer with details about the available General, Motion, Process, and Drives instruction set for a Logix-based controller.

If you design, program, or troubleshoot safety applications that use GuardLogix controllers, refer to the GuardLogix Safety Application Instruction Set Safety Reference Manual, publication 1756-RM095.

This manual is one of a set of related manuals that show common procedures for programming and operating Logix 5000 controllers.

For a complete list of common procedures manuals, refer to the Logix 5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.

The term Logix 5000 controller refers to any controller based on the Logix 5000 operating system.

Studio 5000 environment

The Studio 5000 Automation Engineering & Design Environment® combines engineering and design elements into a common environment. The first element is the Studio 5000 Logix Designer® application. The Logix Designer application is the rebranding of RSLogix 5000® software and will continue to be the product to program Logix 5000™ controllers for discrete, process, batch, motion, safety, and drive-based solutions.

The Studio 5000® environment is the foundation for the future of Rockwell Automation® engineering design tools and capabilities. The Studio 5000 environment is the one place for design engineers to develop all elements of their control system.

Additional resources

These documents contain additional information concerning related Rockwell Automation products.

Resource	Description

Resource	Description
Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1	Provides general guidelines for installing a Rockwell Automation industrial system.
Product Certifications webpage, available at http://ab.rockwellautomation.com	Provides declarations of conformity, certificates, and other certification details.

View or download publications at http://www.rockwellautomation.com/literature. To order paper copies of technical documentation, contact the local Rockwell Automation distributor or sales representative.

Legal Notices

Rockwell Automation publishes legal notices, such as privacy policies, license agreements, trademark disclosures, and other terms and conditions on the Legal Notices page of the Rockwell Automation website.

Software and Cloud Services Agreement

Review and accept the Rockwell Automation Software and Cloud Services Agreement here.

Open Source Licenses

The software included in this product contains copyrighted software that is licensed under one or more open source licenses. Copies of those licenses are included with the software. Corresponding Source code for open source packages included in this product are located at their respective web site(s).

Alternately, obtain complete Corresponding Source code by contacting Rockwell Automation via the Contact form on the Rockwell Automation website:
http://www.rockwellautomation.com/global/about-us/contact/contact.page
Please include "Open Source" as part of the request text.

A full list of all open source software used in this product and their corresponding licenses can be found in the OPENSOURCE folder. The default installed location of these licenses is `C:\Program Files (x86)\Common Files\Rockwell\Help\<Product Name>\Release Notes\OPENSOURCE\index.htm`.

Understand Instruction Timing

Motion instructions use three types of timing sequences:

Description	Timing Type
The instruction completes in one scan.	immediate on page 17
The instruction completes over several scans because the instruction sends messages to the servo module.	message on page 18
The instruction could take an indefinite amount of time to complete.	process on page 20

Immediate type motion instructions execute to completion in one scan. If the controller detects an error during the execution of these instructions, the error status bit sets and the operation ends.

Examples of immediate type instructions include the:

- Motion Change Dynamics (MCD) instruction
- Motion Group Strobe Position (MGSP) instruction

Immediate instructions work as follows:

1. When the rung that contains the motion instruction becomes true, the controller:
   - Sets the enable (EN) bit.
   - Clears the done (DN) bit.
   - Clears the error (ER) bit.
2. The controller executes the instruction completely.

If the controller:	Then:
Does not detect an error when the instruction executes	The controller sets the .DN bit.
Detects an error when the instruction executes	The controller sets the .ER bit and stores an error code in the control structure.

3. The next time the rung becomes false after either the .DN or .ER bit sets, the controller clears the .EN bit.
4. The controller can execute the instruction again when the rung becomes true.

Message type motion instructions send one or more messages to the servo module.

Examples of message type instructions include the:

- Motion Direct Drive On (MDO) instruction
- Motion Redefine Position (MRP) instruction

Message type instruction work as follows:

1. When the rung that contains the motion instruction becomes true, the controller:
   - Sets the enable (EN) bit.
   - Clears the done (DN) bit.
   - Clears the error (ER) bit.
2. The controller begins to execute the instruction by setting up a message request to the servo module.

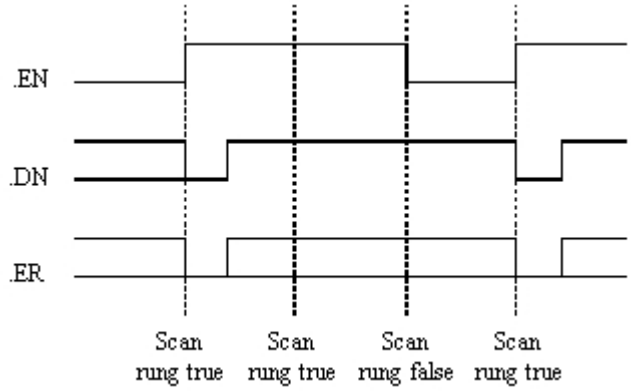
   Tip: The remainder of the instruction executes in parallel to the program scan.
3. The controller checks if the servo module is ready to receive a new message.
4. The controller places the results of the check in the message status word of the control structure.
5. When the module is ready, the controller constructs and transmits the message to the module.

   Tip: This process may repeat several times if the instruction requires multiple messages.
6. The instruction executes.

If the controller:	Then:
Does not detect an error when the instruction executes	The controller sets the .DN bit.
Detects an error when the instruction executes	The controller sets the .ER bit and stores an error code in the control structure.

7. The next time the rung becomes false after either the .DN or .ER bit sets, the controller clears the .EN bit.

8. When the rung becomes true, the controller can execute the instruction again.



See also

Motion Direct Drive On (MDO) on page 47

Motion Redefine Position (MRP) on page 139

Process Type Instructions

Process type motion instructions initiate motion processes that can take an indefinite amount of time to complete.

Examples of process type instructions include the:

- Motion Arm Watch Position (MAW) instruction
- Motion Axis Move (MAM) instruction

Process type instructions work as follows:

1. When the rung that contains the motion instruction becomes true, the controller:
   - Sets the enable (.EN) bit.
   - Clears the done (.DN) bit.
   - Clears the error (.ER) bit.
   - Clears the process complete (.PC) bit.
2. The controller initiates the motion process.

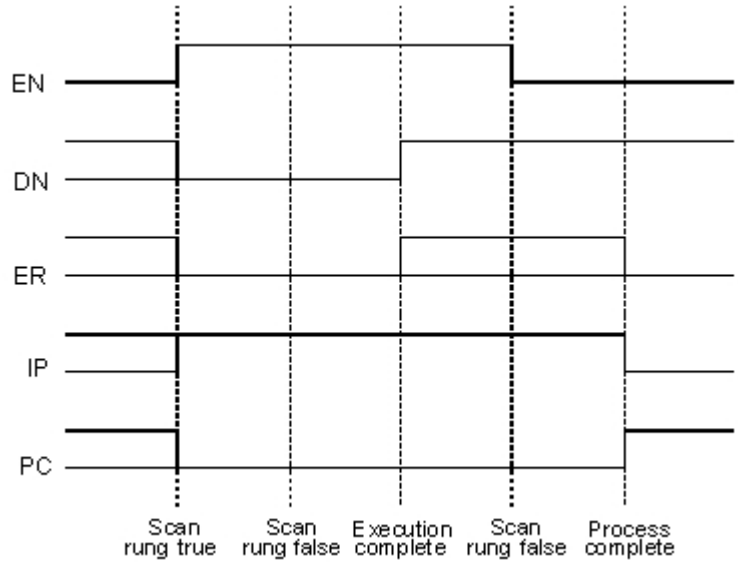
If:	Then the controller:
The controller does not detect an error when the instruction executes	• Sets the .DN bit. • Sets the in process (.IP) bit.
The controller detects an error when the instruction executes	• Sets the .ER bit. • Stores an error code in the control structure. • Does not change the .IP and .PC bits.
The controller detects another instance of the motion instruction	Clears the .IP bit for that instance.
The motion process reaches the point where the instruction can be executed again	Sets the .DN bit. **Tip:** For some process type instructions, like MAM, this will occur on the first scan. For others, like MAH, the .DN bit will not be set until the entire homing process is complete.
One of the following occurs during the motion process: • The motion process completes • Another instance of the instruction executes • Another instruction stops the motion process • A motion fault stops the motion process	• Sets the .DN bit. • Sets the .PC bit. • Clears the .IP bit.

3. Once the initiation of the motion process completes, the program scan can continue.

   Tip: The remainder of the instruction and the control process continue in parallel with the program scan.

4. The next time the rung becomes false after either the .DN bit or the .ER bit sets, the controller clears the .EN bit.

5. When the rung becomes true, the instruction can execute again.



Write a Motion Application Program

To write a motion application program, you can insert motion instructions directly into the ladder diagram application program. The motion instruction set consists of five groups of motion instructions:

- Motion state instructions on page 23
- Motion move instructions on page 73
- Motion group instructions on page 225
- Motion event instructions on page 249
- Motion configuration instructions on page 315

These instructions operate on one or more axes. Identify and configure axes before you can applying axes. For more information about configuring axes, see the Integrated Motion on the EtherNet/IP Network: Configuration and Startup publication MOTION-UM003.

See also

Understand Motion Status and Configuration Parameters on page 639

Modify Motion Configuration Parameters on page 357

Handle Motion Faults on page 594

Understand a Programming Example on page 255

Motion State Instructions

Motion State Instructions

Motion state control instructions directly control or change the operating states of an axis. These are the motion state instructions.

Available Instructions

Ladder Diagram and Structured Text

MSO	MSF	MASD	MASR	MDO	MDF	MDS	MAFR

Function Block

Not available

IMPORTANT Tags used for the motion control attribute of instructions should only be used once. Re-use of the motion control tag in other instructions can cause unintended operation. This may result in damage to equipment or personal injury.

Motion state control instructions directly control or change the operating states of an axis. The motion state instructions are:

If you want to:	Use this instruction:
Enable the servo drive and activate the axis servo loop.	MSO
Disable the servo drive and deactivate the axis servo loop.	MSF
Force an axis into the shutdown operating state. Once the axis is in the shutdown operating state, the controller will block any instructions that initiate axis motion.	MASD
Change an axis from an existing shutdown operating state to an axis ready operating state. If all of the axes of a servo module are removed from the shutdown state as a result of this instruction, the OK relay contacts for the module will close.	MASR
Enable the servo drive and set the servo output voltage of an axis.	MDO
Deactivate the servo drive and set the servo output voltage to the output offset voltage.	MDF
Activate the drive control loops for the specified axis and run the motor at the specified speed.	MDS

Clear all motion faults for an axis.	MAFR

The five operating states of a non-CIP axis are:

Operating State	Description
Axis Ready	This is the normal power-up state of the axis. In this state: • The servo module drive enable output is inactive. • Servo action is disabled. • No servo faults are present.
Direct Drive Control	This operating state allows the servo module DAC to directly control an external drive. In this state: • The servo module drive enable output is active. • Position servo action is disabled.
Servo Control	This operating state allows the servo module to perform closed loop motion. In this state: • The servo module drive enable output is active. • Servo action is enabled. • The axis is forced to maintain the commanded servo position.
Axis Faulted	In this operating state, a servo fault is present, and the status of the drive enable output, the action of the servo, and the condition of the OK contact depend on the faults and fault actions that are present.
Shutdown	This operating state allows the OK relay contacts to open a set of contacts in the E- string of the drive power supply. In this state: • The servo module drive enable output is inactive. • Servo action is disabled. • The OK contact is open.

The 16 operating states of a CIP axis are:

Operating State	As Shown in the Logix Designer Programming Application	Description
Initializing	0	During the Initializing State, the drive first initializes all attributes to their factory default values, that is, resets all active faults. The drive then waits for the controller to establish connections to it. Once connections are established, the controller sets configuration attributes in the drive to values stored in the controller. If the drive supports synchronous operation, the controller then synchronizes with the drive. Once this process has been completed successfully, the drive and all its associated axis instances transition to the Pre-charge state. If a problem is found during the initializing process, an Initialization Fault is generated. An Initialization Fault is an unrecoverable fault. You can only clear the fault can via a power cycle or a drive reset. If the connection to the drive closes for any reason during operation, the drive returns to the Initializing State
Pre-Charge	1	The drive is waiting for the DC Bus to fully charge, that is, the DC Bus Up status bit is cleared. Once the DC Bus reaches an operational voltage level, that is, DC Bus Up status bit is set, the axis transitions to the Stopped state. The drive's power structure is always disabled in this state, that is, the Power Structure Enabled status bit is cleared. Any attempt to enable the drive via the Axis Control mechanism while in this state is reported back to the controller as an error in the Response Status and the axis remains in the Pre-charge state.

Stopped	2	In the Stopped state, the drive's inverter power structure should either be disabled and free of torque, that is, the Power Structure Enabled status bit is cleared, or held in a static condition via an active control loop, that is, Power Structure Enabled status bit is set. The drive cannot initiate motion in the Stopped state nor can the drive respond to a planner generated command reference, that is, the Tracking Command status bit is cleared. In general, the axis should be at rest. However, if you apply an external force or torque to the load, a brake may be needed to maintain the rest condition. In the Stopped state, main power is applied to the drive and the DC Bus are at an operational voltage level. If there are any Start Inhibited conditions detected while in this state, the axis transitions to the Start Inhibited state. If an Enable request or one of the Run Test service requests is applied to an axis in the Stopped state, the motion axis transitions to the Starting state.
Starting	3	When an Enable request is given to an axis in the Stopped or Stopping state while it is performing a Flying Start, the axis immediately transitions to the Starting state. In this state, the drive checks the following conditions before transitioning to the Running state. • Brake Release delay time • Induction Motor flux level The drive control and power structures are activated during the Starting state, that is, the Power Structure Enabled status bit is set. But the command reference is set to a local static value and does not track the command reference derived from the motion planner, that is, the Tracking Command status bit is cleared. If all the starting conditions are met, the axis state transitions to either the Running state or the Testing state.
Running	4	The drive's power structure is active, that is, the Power Structure Enabled status bit is set. Additionally, the selected Control Mode is enabled and actively tracking command data from the controller-based or drive-based motion planner output to affect axis motion, that is, the Tracking Command status bit is set.
Testing	5	When any one of the Run Test request services is sent to the motion axis while in the Stopped state, that is, services that require an active power structure to execute, the axis immediately transitions to the Starting state, that is, the Power Structure Enabled status bit is set. Then once the Starting state conditions are met, the axis transitions to the Testing state. Like the Running state, in the Testing state, the drive's power structure is active. The motion axis remains in this state for the duration of the requested test procedure and then returns to the Stopped state. The motion axis can also exit the Testing state by either a fault or an explicit Axis Control request.
Stopping	6	When a Disable request is issued to an axis in the Running or Testing state, the axis immediately transitions to the Stopping state. In this state, the axis is in the process of stopping and no longer tracks command data from the motion planner, that is, the Tracking Command status bit is cleared. Once the selected Stopping Mode procedure has completed, the axis transitions to the Stopped state.
Aborting	7	When a Major Fault occurs in the drive while the axis is in either the Running or Testing states, the motion axis immediately transitions to the Aborting state. In this state, the axis is in the process of stopping and no longer tracks command data from the motion planner, that is, the Tracking Command status bit is cleared. The Aborting state executes the appropriate stopping action as specified by the drive. As with the Stopping state, in the Aborting state the power structure remains active, that is, the Power Structure Enabled status bit is set, for as long as the stopping action takes to complete. Once the stopping procedure is complete the axis transitions to the Faulted state. When faults conditions are detected in the controller that are not visible to the drive, or when the drive reports a Minor Fault condition, the controller brings the axis to a stop, either directly via an Axis Control state change request or motion planner stop, or indirectly via a fault handler in the user program. If the Axis State reported by the driv e is Stopping, the controller sets the CIP Axis State to Aborting based on the presence of the fault condition.

Faulted	8	The faulted state is identical to the Stopped state or the Shutdown state with the exception that there are one or more faults active. Faults are latched conditions. Therefore, a Fault Reset is required to clear the faults and, assuming the original fault condition has been removed, the axis transitions to the Axis State of the drive. There are many different sources of faults: • CIP Initialization Faults - Faults that occur when the drive transitions out of the Initializing state. These faults can apply to a specific axis or the entire drive. • CIP Axis Faults - Faults that apply to a specific axis and are the direct result of Axis Exceptions configured to generate a Fault response. Axis exceptions are run-time conditions that are related to Motor, Inverter, Converter, Bus Regulator, and Feedback components. • Safety Fault: Faults that apply to a specific axis and are generated by a fault condition detected in the drive's safety monitor functionality. A Safety Fault always results in the axis transitioning to the Stopped state. • Motion Fault: Faults generally associated with fault conditions generated by the motion planner function. These faults can include conditions related to the input, for example, actual position, or output, for example, command position, signals. • Module Fault: Faults that apply to the entire drive and affect all axes associated with that drive. Module faults include all node faults reported by the drive and also communication fault conditions detected on the controller side of the motion connection. • Group Fault: Faults related to the motion group object function and affect all axes associated with the motion group. Group Fault conditions are detected by controller and are associated with the time synchronization function common to all axes in the motion group. • Configuration Fault: Fault generated anytime there is an error when sending configuration data to the drive.
Start Inhibited	9	This state is the same as the Stopped state with the exception that the axis has one or more 'start inhibit' conditions that prevent it from successfully transitioning to the Starting state. Once corrected, the axis state automatically transitions back to the Stopped state.
Shutdown	10	When a Shutdown request is issued to the drive or a Shutdown fault action is executed by the drive, the targeted axis immediately transitions to the Shutdown state. The Shutdown state has the same basic characteristics of the Stopped state except that it can be configured via the Shutdown Action attribute to drop the DC Bus power to the drive's power structure. Regardless of whether DC Bus power is disconnected, this state requires an explicit Shutdown Reset request from the controller to transition to the Pre-charge state. If the drive is configured to keep DC Bus power active while in the Shutdown state, then the motion axis transitions through the Pre-charge state to the Stopped state. In the case where a Shutdown fault action is initiated by the drive in response to an exception condition that is configured to be a major fault, the drive executes the Shutdown action. However, the axis goes to the Faulted state, not the Shutdown state. Similarly, when the axis is in the Shutdown state and a major fault condition occurs, the axis transitions to the Faulted state. A Fault Reset request from the controller clears the fault and, assuming the original fault condition has been removed, the axis transitions to the Shutdown state. A Shutdown Reset request from the controller, however, both clears the fault and performs a shutdown reset so, assuming the original fault condition has been removed, the axis transitions to the Pre-charge state.
Axis Inhibited	11	If you inhibit the axis, the associated instance in the CIP Motion connection is eliminated and the axis state transitions to the Axis Inhibited state. If this is the only instance supported by the CIP Motion connection, the connection itself is closed. The Axis Inhibited state is a controller-only sub state of the Self-test state[1]. The Axis Inhibited condition is checked during the controller Self-test state as qualification for transition to the Initializing state. If currently Axis Inhibited, you must execute an Un-Inhibit operation to transition to the Initializing state and restore axis function.
Not Grouped	12	If a CIP Motion axis is created and not associated with a Motion Group, the axis state is set to the Not Grouped state. A CIP Motion axis must be assigned to a Motion Group for the axis to be updated by the periodic Motion Task and carry out its function. This condition is checked during the controller Self-test state as qualification for transition to the Initializing state. For this reason, the Not Grouped state is considered a controller-only sub state of the Self-test state.

No Device	13	If the CIP Motion axis in the controller is created, but not currently associated with a drive, the axis state indicates the No Device state. A CIP Motion axis must be associated with a physical drive to function. This condition is checked during the controller Self-test state as qualification for transition to the Initializing state. For this reason, the No Device state is considered a controller-only sub state of the Self Test state.

(1) The Self-test state is a drive state. This state does not appear in the Logix Designer programming application as an operating state of a CIP axis. Instead, self-test is represented as the Initializing state for a CIP axis.

See also

Motion Configuration Instructions on page 315

Motion Move Instructions on page 73

Multi-Axis Coordinated Motion Instructions on page 359

Motion Event Instructions on page 249

Motion Group Instructions on page 225

Motion Axis Fault Reset (MAFR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

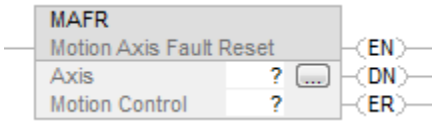
Use the Motion Axis Fault Reset (MAFR) instruction to clear all motion faults for an axis. This is the only method for clearing axis motion faults.

IMPORTANT The MAFR instruction removes the fault status, but does not perform any other recovery, such as enabling servo action. In addition, when the controller removes the fault status, the condition that generated the fault(s) may still exist. If the condition is not corrected before using the MAFR instruction, the axis immediately faults again.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MAFR(Axis,MotionControl);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description
Axis	AXIS_CIP_DRIVE	AXIS_CIP_DRIVE AXIS_GENERIC AXIS_GENERIC_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE **Tip:** AXIS_GENERIC is supported by the ControlLogix 5570 and the GuardLogix 5570 controllers only.	Tag	Name of the axis on which to perform the operation. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

See Structured Text Syntax for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axis' servo action has been successfully disabled and the drive enable and servo active status bits have been cleared.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MAFR instruction directly clears the specified fault status on the specified axis. It does not correct the condition that caused the error. If the condition is not corrected prior to executing the MAFR instruction the axis could immediately fault again giving the appearance that the fault status was not reset.

This instruction is most commonly used as part of a fault handler program, which provides application specific fault action in response to various potential motion control faults. Once the appropriate fault action is taken, the MAFR instruction can be used to clear all active fault status bits.

Important:	The .DN bit is not set immediately. It will be set once the appropriate Motion Module or Drive has completed its required resets, which could take up to several seconds. Once set the axis is in the Ready state, but only after the request is completed.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults. A major fault can occur if an uninitialized reference or reference of the incorrect type is passed to the Axis operand.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, and .ER are cleared to false.

Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes. If the EN bit is set to false, then there is no action taken,
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in Ladder Diagram table
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in Ladder Diagram table.

Error Codes

See *Motion Error Codes (.ERR)* for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. See *Motion Error Codes (.ERR)* for Motion Instructions.

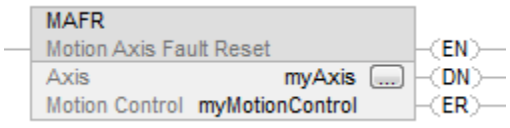
MAFR Changes to Status Bits

None

Examples

When the input conditions are true, the controller clears all motion faults for myAxis.

Ladder Diagram



Structured Text

MAFR(myAxis,myMotionControl);

See also

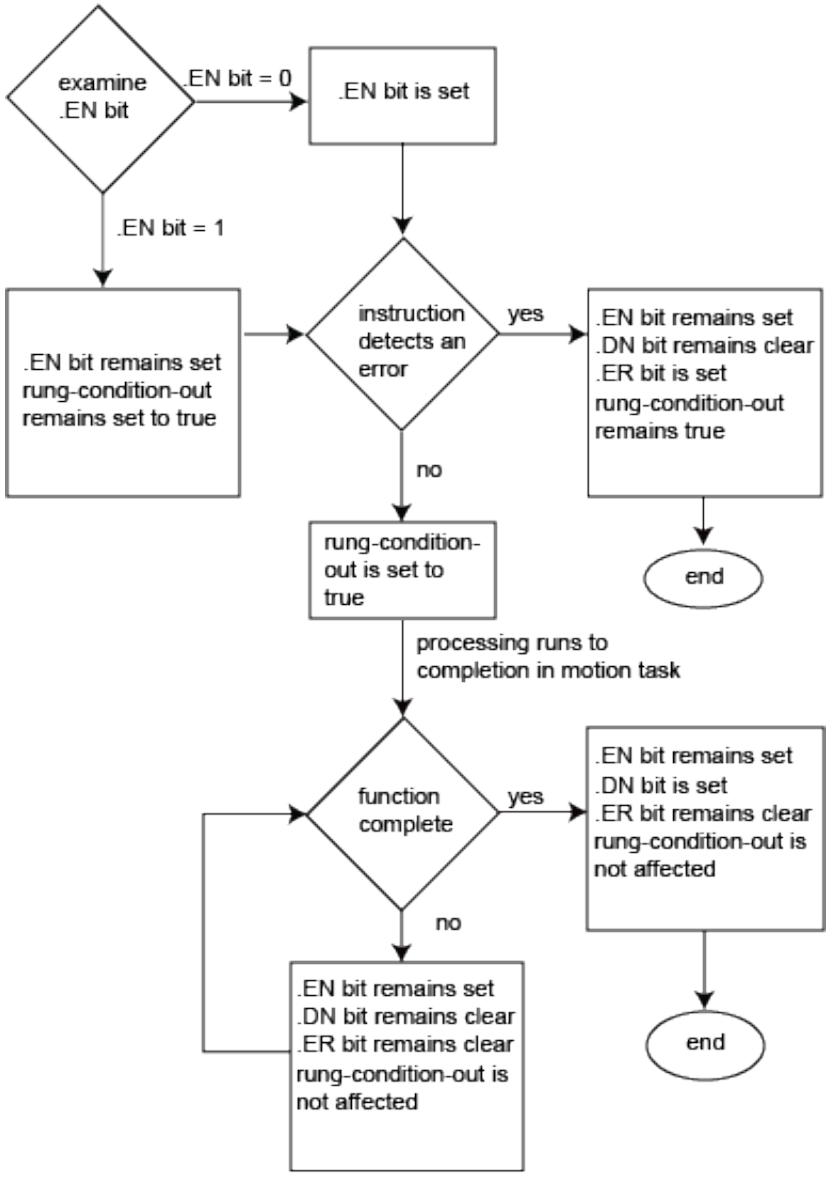
Common Attributes on page 691

Structured Text Syntax on page 665

Motion Error Codes (.ERR) on page 581

MAFR Flow Chart on page 31

MAFR Flow Chart (True)



Motion Axis Shutdown (MASD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Use the Motion Axis Shutdown (MASD) instruction to force a specified axis into the Shutdown state. The Shutdown state of an axis is the condition where the drive output is disabled, servo loop deactivated, and any available or associated OK solid-state relay contacts open. The axis remains in the Shutdown state until either an Axis or Group Shutdown Reset is executed.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block.

Structured Text

MASD(Axis,MotionControl);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL	AXIS_CIP_DRIVE AXIS_GENERIC AXIS_GENERIC_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE AXIS_VIRTUAL **Tip:** AXIS_GENERIC is supported by the ControlLogix 5570 and the GuardLogix 5570 controllers only.		Name of the axis on which to perform the operation. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion	MOTION_INSTRUCTION			Structure used to access instruction status parameters.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axes have been successfully set to Shutdown state.
.ER (Done) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MASD instruction directly and immediately disables drive output, disables the servo loop, and opens any associated OK contacts. This action places the axis into the Shutdown state.

Another action initiated by the MASD instruction is the clearing of all motion processes in progress and the clearing of all the motion status bits. Associated with this action, the command also clears all motion instruction IP bits that are currently set for the targeted axis.

The MASD instruction forces the targeted axis into the Shutdown state. One of the unique characteristics of the Shutdown state is that, when available, the OK solid state relay contact for the motion module or drive is Open. Where available this feature can be used to open up the E-Stop string that controls main power to the drive system. Note that there is typically only one OK contact per motion module which means that execution of an MASD instruction for either axis associated with a given module opens the OK contact.

Another characteristic of the Shutdown state is that any instruction that initiates axis motion is blocked from execution. Attempts to do so result in an execution error. Only by executing one of the Shutdown Reset instructions can motion be successfully initiated.

The axis remains in the shutdown state until a Motion Axis Shutdown Reset (MASR), a Motion Group Shutdown Reset (MGSR), or a Motion Coordinate Shutdown Reset (MCSR) instruction executes. If the axis is associated with a Coordinate System, the axis will be reset if the Motion Coordinate Shutdown Reset (MCSR) instruction executes.

IMPORTANT The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

Additionally, for CIP motion, the MASD instruction supports canceling the Motion Drive Start (MDS) instruction. This includes clearing the MDS In Process (.IP) bit, and clearing the DirectVelocityControlStatus and the DirectTorqueControlStatus bit in the Motion Status attribute.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Master Driven Speed Control (MDSC) and the MASD Instruction

When the axis is shut down:

- The IP bit of the Master Driven Axis Control (MDAC) instruction is reset on an axis that is shutdown.
- The AC bit of the MDAC instruction is reset when the axis is stopped as it is shutdown.
- The MASD instruction clears the pending Master Axis for all future single motion instructions.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults. A major fault can occur if an uninitialized reference or reference of the incorrect type is passed to the Axis operand.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, and .ER are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes. If the EN bit is set to false, then there is no action taken.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See *Motion Error Codes (.ERR)* for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. See *Motion Error Codes (.ERR)* for Motion Instructions.

MASD Changes to Single Axis Status Bits

Axis Status Bits

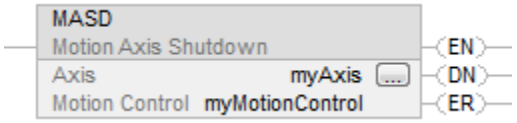
Bit Name	State	Meaning
ServoActionStatus	FALSE	The axis is in the Servo Off state with the servo loop inactive.
DriveEnableStatus	FALSE	The drive enable output is inactive.
ShutdownStatus	TRUE	The axis is in the shutdown state.

Motion Status Bits

Bit Name	State	Meaning
AccelStatus	FALSE	Axis is not Accelerating.
DecelStatus	FALSE	Axis is not Decelerating.
MoveStatus	FALSE	Axis is not Moving.
JogStatus	FALSE	Axis is not Jogging.
GearingStatus	FALSE	Axis is not Gearing.
HomingStatus	FALSE	Axis is not Homing.
StoppingStatus	FALSE	Axis is not Stopping.
PositionCamStatus	FALSE	Axis is not Position Camming.
TimeCamStatus	FALSE	Axis is not Time Camming.
PositionCamPendingStatus	FALSE	Axis does not have a Position Cam Pending.
TimeCamPendingStatus	FALSE	Axis does not have a Time Cam Pending.
GearingLockStatus	FALSE	Axis is not in a Gear Locked condition.
PositionCamLockStatus	FALSE	Axis is not in a Cam Locked condition.
DirectVelocityControlStatus	FALSE	Axis is not under Direct Velocity Control.
DirectTorqueControlStatus	FALSE	Axis is not under Direct Torque Control.

Examples

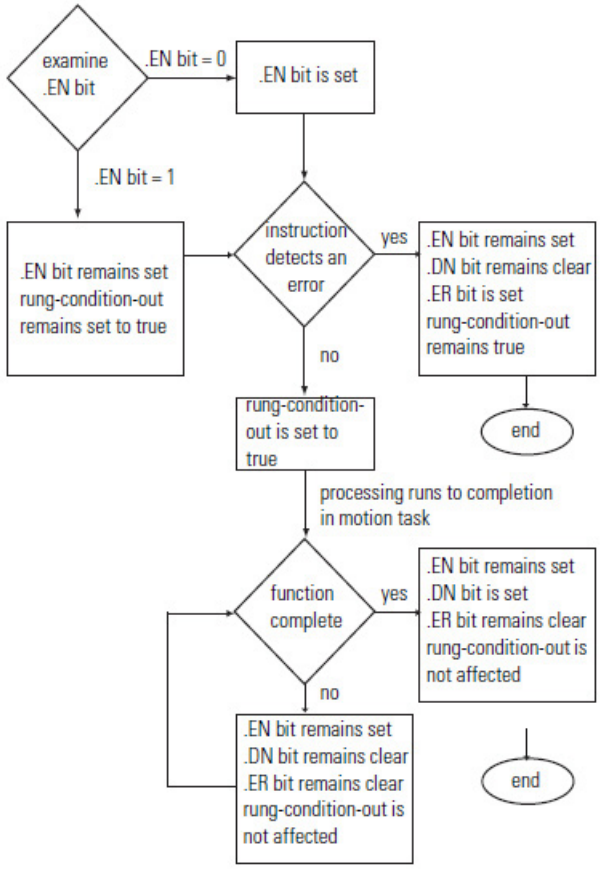
Ladder Diagram



Structured Text

MASD(myAxis, myMotionControl);

MASD Flow Chart (True)



Motion Axis Shutdown Reset (MASR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

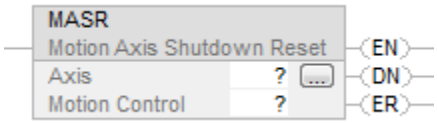
Use the Motion Axis Shutdown (MASR) instruction to transition an axis from an existing Shutdown state to an Axis Ready state. All faults associated with the specified axis are automatically cleared. If, as a result of this instruction,

all axes of the associated motion module are no longer in the Shutdown condition, the OK relay contacts for the module close.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MASR(Axis,MotionControl);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description

Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL	AXIS_CIP_DRIVE AXIS_GENERIC AXIS_GENERIC_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE AXIS_VIRTUAL **Tip:** AXIS_GENERIC is supported by the ControlLogix 5570 and the GuardLogix 5570 controllers only.	Tag	Name of the axis on which to perform the operation. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axes is successfully reset from Shutdown state.
.ER (Done) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MASR instruction clears all axis faults and takes the specified axis out of the Shutdown state. If the motion module supports an OK contact, and no other module axis is in the Shutdown state, the MASR instruction results in closure of the module's OK solid-state relay contact. Regardless of the OK contact condition, execution of the MASR places the axis into the Axis Ready state.

Just as the Motion Axis Shutdown (MASD) instruction forces the targeted axis into the Shutdown state, the MASR instruction takes the axis out of the Shutdown state into the Axis Ready state. One of the unique characteristics of the Shutdown state is that any associated OK solid state relay contact for the motion module is Open. If, as a result of an MASR instruction there are no axes associated with a given motion module in the Shutdown state, the OK relay contacts close as a result of the MASR. This feature can be used to close the E-Stop string that controls main power to the drive system and, thus, permit the customer to reapply power to the drive. Note that there is typically

only one OK contact per motion module which means that execution of the MASR instruction may be required for all axes associated with a given module for the OK contact to close.

The MASR instruction is a procedure type command that is processed from the Logix controller, through the associated motion module, and to the associated drives.

IMPORTANT The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

In this transitional instruction, the relay ladder, toggle the rung-condition-in from cleared to set each time the instruction should execute.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults. A major fault can occur if an uninitialized reference or reference of the incorrect type is passed to the Axis operand.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, and .ER are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes. If the .EN bit is set to false, no action is taken.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in Ladder Diagram table.

Error Codes

See *Motion Error Codes (.ERR)* for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. See *Motion Error Codes (.ERR)* for Motion Instructions.

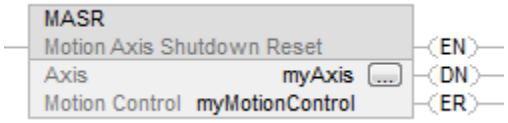
Status Bits

Bit Name	State	Meaning
ShutdownStatus	FALSE	The axis is not in the shutdown state.

Examples

When the input conditions are true, the controller resets axis1 from a previous shutdown operating state into an axis ready operating state.

Ladder Diagram



Structured Text

MASR(myAxis, myMotionControl);

See also

Motion State Instructions on page 23

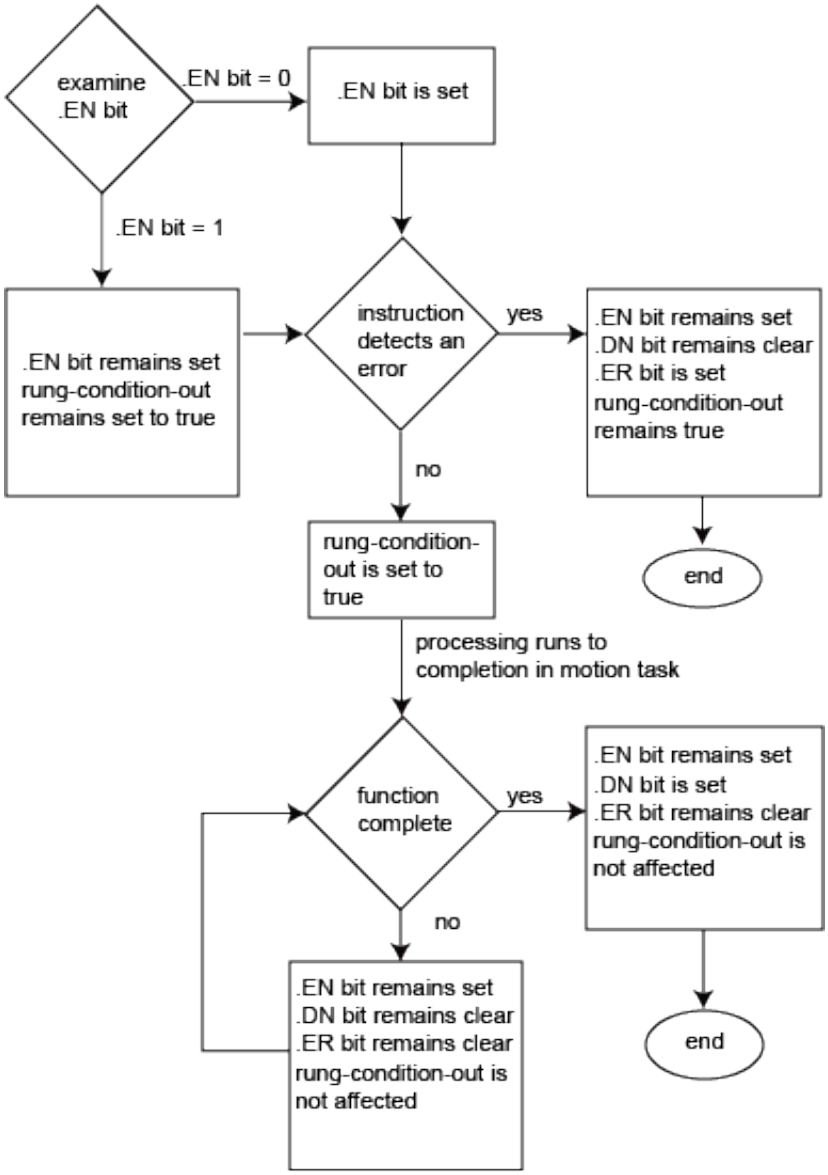
Common Attributes on page 691

Structured Text Syntax on page 665

Motion Error Codes (.ERR) on page 581

MASR Flow Chart on page 41

MASR Flow Chart (True)



Motion Direct Drive Off (MDF)

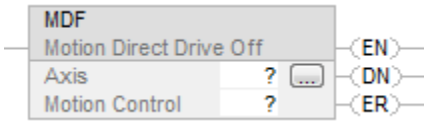
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, and ControlLogix 5580 controllers.

Use the Motion Direct Drive Off (MDF) instruction to deactivate the servo drive and to set the servo output voltage to the output offset voltage. The output offset voltage is the output voltage that generates zero or minimal drive motion. You can specify this value during axis configuration.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MDF(Axis,MotionControl);

Operands

Ladder Diagram and Structured Text

Operand	Type ControlLoigx 5570, ControlLogix 5580, GuardLogix 5570 and GuardLogix 5580	Format	Description
Axis	AXIS_SERVO	Tag	Motion Axis of data type AXIS_SERVO only. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Structure

Mnemonic	Description

.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axis' drive signals have been successfully disabled and the drive enable status bit is cleared.
.ER (Done) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

For motion module's with an external servo drive interface, the MDF instruction directly disables the motion module Drive Enable output of the specified physical axis and also zeroes the modules' servo output to the external drive by applying the configured Output Offset value.

The MDF instruction is used to stop motion initiated by a preceding Motion Direct Drive On (MDO) instruction and transition the axis from the Direct Drive Control state back to the Axis Ready state.

To successfully execute an MDF instruction, the targeted axis must be configured as a Servo axis. Otherwise, the instruction errors.

IMPORTANT The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the EanbleIn false to true each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or reference of the incorrect type is passed to the Axis operand. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, and .ER are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes. If the EN bit is set to false, then there is no action taken,
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. See *Motion Error Codes (.ERR)* for Motion Instructions.

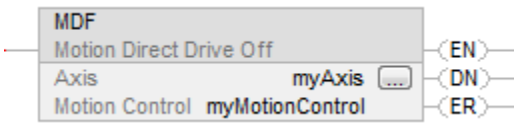
MDF Changes to Single Axis Status Bits

Axis Status Bits

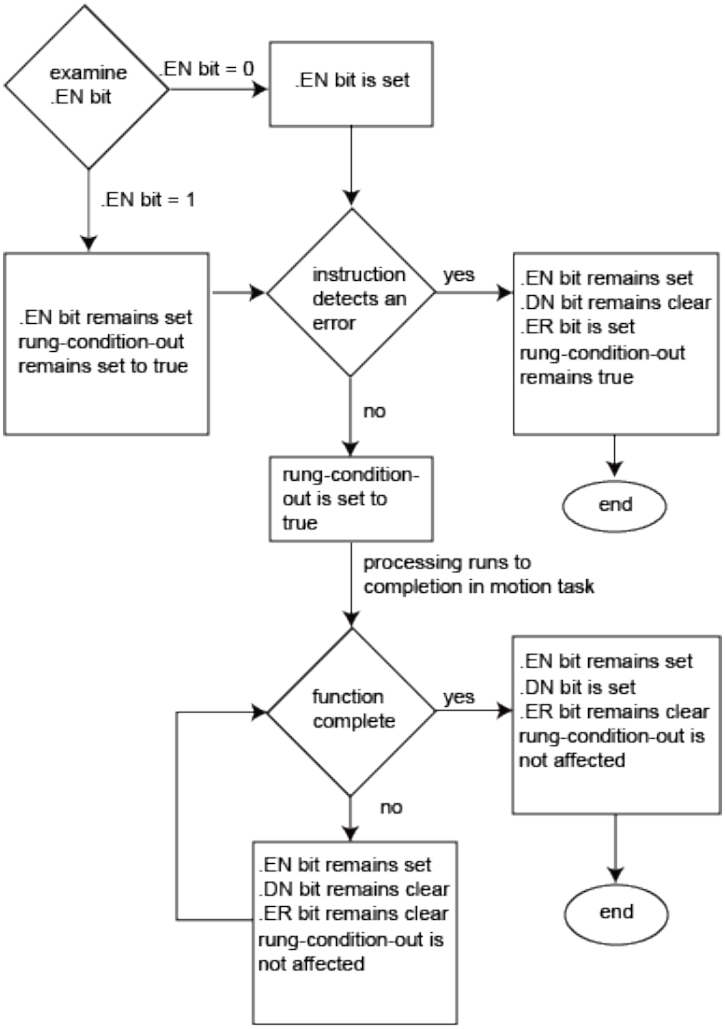
Bit Name	State	Meaning
DriveEnableStatus	TRUE	Axis is in Axis Ready state with the Drive Enable output now active.

Example

Ladder Diagram



MDF Flow Chart (True)



Motion Direct Drive On (MDO)

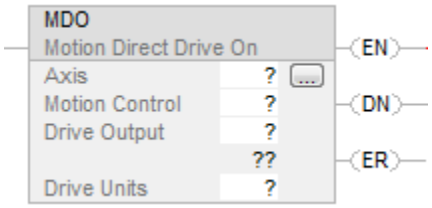
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, and ControlLogix 5580 controllers.

Use the Motion Direct Drive On (MDO) instruction in conjunction with motion modules that support an external analog servo drive interface. This instruction activates the Drive Enable of the module, enabling the external servo drive, and also sets the output voltage of the drive of the servo module to the specified voltage level. The value for Drive Output may be specified in Volts or % of maximum axis' Output Limit.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MDO(Axis,MotionControl, DriveOutput,DriveUnits);

Operands

Ladder Diagram and Structured Text

Operand	Type ControlLoigx 5570, ControlLogix 5580, GuardLogix 5570, and GuardLogix 5580	Format	Description
Axis	AXIS_SERVO	Tag	Motion Axis of data type AXIS_SERVO only. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Drive Output	SINT INT DINT REAL	Immediate or Tag	Voltage to output in % of servo Output Limit or in Volts.
Drive Units	Boolean	Immediate	How do you want to interpret the drive output? 0 = Volts 1 = Percent

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

Enter your selection for the operands that require you to select from available options:

This Operand	Has These Options Which You...	
	Enter As Text	Or Enter As a Number
DriveUnits	Volts Percent	0 1

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axis' drive enable bit is activated and the specified analog output is successfully applied.
.ER (Done) Bit 28	It is set to indicate that the instruction detected an error, such as if you entered a Drive Output value that was too large.

Description

For motion modules with an external servo drive interface, the MDO instruction can be used to directly enable the Drive Enable output of the axis and set the analog output to a specified level determined by the Drive Output parameter. The Drive Output parameter can be expressed as a voltage, or as a percent of the maximum configured output voltage value given by the Output Limit attribute.

The MDO instruction can only be used on a physical axis whose Axis Type is configured for Servo. The instruction only executes when the axis' is in the Axis Ready state (for example, servo action is OFF). The resulting state of the axis is referred to as the Drive Control state.

The MDO instruction automatically enables the specified axis by activating the appropriate Drive Enable output before setting the servo module's analog output to the specified voltage value. There is, typically, a 500 msec delay between the activation of the drive enable output and the setting of the analog output to the specified level to allow the drive's power structure to stabilize. To minimize drift during this drive enabling delay, the output voltage to the drive is set to the Output Offset attribute value (default is zero). Thereafter the output voltage is given by the specified Drive Output value of the MDO instruction and indicated by the Servo Output status attribute value.

The 16-bit DAC hardware associated with various Logix servo modules limits the effective resolution of the Direct Drive Motion Control to 305 μV or 0.003%. In the case of Direct Drive operation, the module's servo loop is inactive and bypassed. The Motion Direct Drive On instruction is only

affected by the Servo Output Polarity configuration bit, the Output Offset, and Output Limit attributes for the axis. In the case where Output Limit configuration value is reduced below the current output voltage value, the Servo Output value is automatically clamped to the Output Limit value.

The most common use of this instruction is to provide an independent programmable analog output as an open loop speed reference for an external drive or for testing an external servo drive for closed loop operation.

To successfully execute a MDO instruction, the targeted axis must be configured as a Servo axis and be in the Axis Ready state, with servo action off. If these conditions are not met the instruction errs.

IMPORTANT The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle the rung-condition-in from false to true each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition.

Loss of Feedback When Using an MDO Instruction

If you experience a loss of feedback when issuing an MDO instruction and need to move the axis with an MDO instruction, follow these steps:

1. Set the Feedback Fault Actions to Status Only.
2. When a feedback fault occurs, issue an MSF instruction to turn the servo off.
3. Issue an MAFR instruction to clear the feedback fault status.

The MDO instruction executes without another feedback fault shutting down the system. However, the feedback fault status remains feedback fault condition exists.

IMPORTANT
- Keep this in mind when using the previous steps:
- Once feedback has been lost, the reported position may not be valid. To re-establish a valid position, perform another home operation.
- If you need to issue another MDO instruction, such as to modify the output voltage, when the first MDO instruction is executing, you must first issue an MSF instruction to stop the servo and then issue an MAFR instruction to clear the feedback fault.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or reference of the incorrect type is passed to the Axis operand. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, and .ER are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes. If the EN bit is set to false, there is no action taken,
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. See Error Codes (ERR) for Motion Instructions.

These Extended Error codes help to pinpoint the problem when the MDO instruction receives a Servo Message Failure (12) error message.

Extended Error Code (decimal)	Associated Error Code (decimal)	Meaning
Object Mode conflict (12)	SERVO_MESSAGE_FAILURE (12)	Axis is shutdown.

Status Bits

MDO Changes to Status Bits

Bit Name	State	Meaning

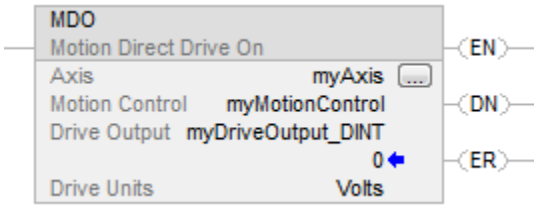
DriveEnableStatus	TRUE	Axis is in Drive Control state with the Drive Enable output active.

Examples

Example 1

The Drive Output operand is a DINT tag and Drive Units is "Volts"

Ladder Diagram



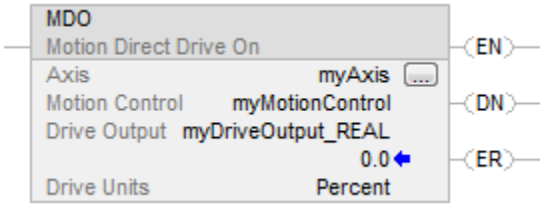
Structured Text

MDO(myAxis, myMotionControl, myDriveOutput_DINT, volts);

Example 2

The Drive Output operand is a REAL tag and Drive Units is "Percent"

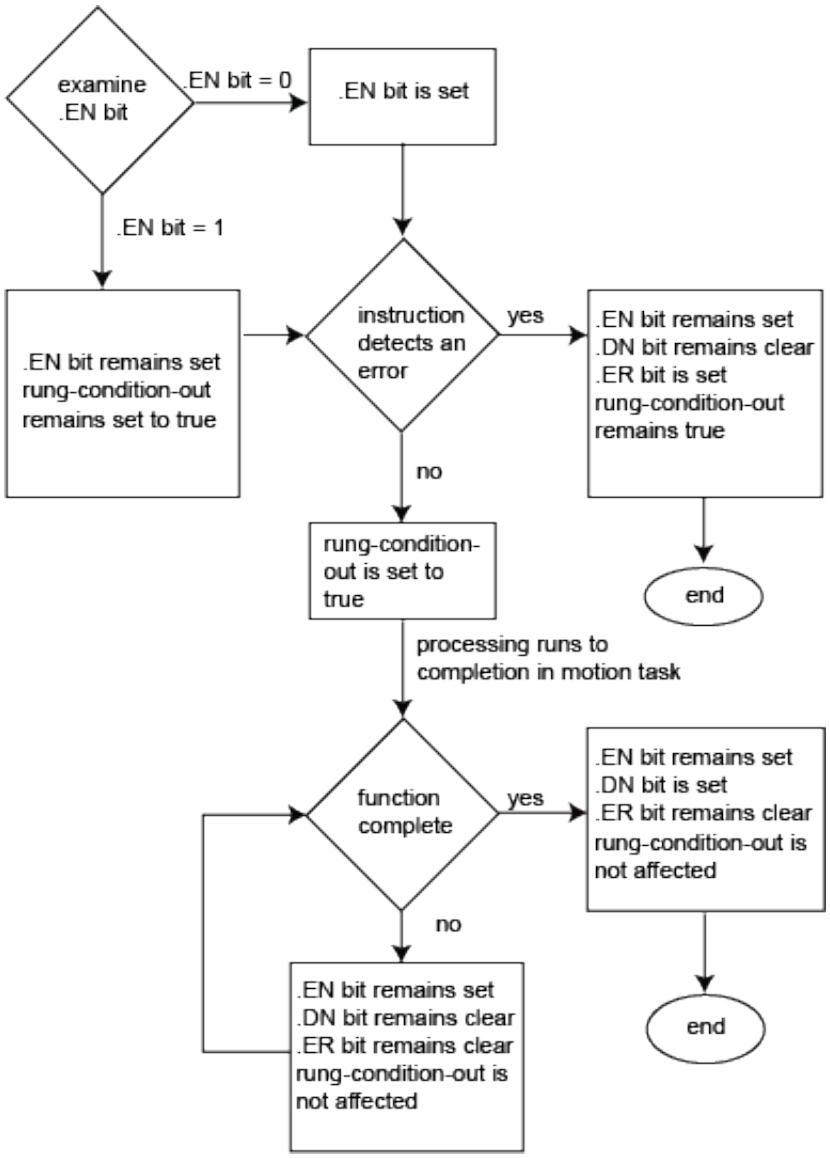
Ladder Diagram



Structured Text

MDO(myAxis, myMotionControl, myDriveOutput_REAL, percent);

MDO Flow Chart (True)



Motion Drive Start (MDS)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Use the Motion Drive Start (MDS) instruction to activate the drive control loops for the specified axis and run the motor at the specified speed.

IMPORTANT The MDS instruction is validated if the CIP drive device supports the S ramp attributes, including:

- RampAcceleration
- RampDeceleration
- RampVelocity - Positive
- RampVelocity - Negative
- RampJerk - Control

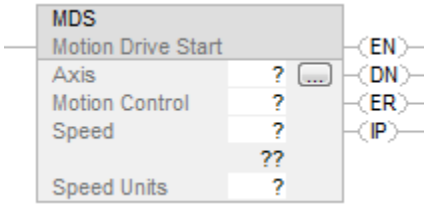
This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.
-

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MDS(Axis,MotionControl,Speed,Unitspersec);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description

Axis	AXIS_CIP_DRIVE	AXIS_CIP_DRIVE	Tag	Motion Axis of data type AXIS_CIP_DRIVE only. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Structure used to control execution of the motion instruction.
Speed	SINT INT DINT REAL	SINT INT DINT REAL	Immediate or Tag	Defines the initial speed for the DirectVelocityControlStatus Command attribute.
Speed Units	SINT INT DINT	SINT INT DINT	Immediate	Which units do you want to use for the speed? 0 = % of Maximum 1 = Units per Sec

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the rung makes a true-to-false transition and drive messaging is complete.
.DN (Done) Bit 29	It is set when the drive has been successfully enabled and remains set until the rung makes a false-to-true transition.
.ER (Error) Bit 28	It is set when the instruction encounters an error condition. The error condition can be a direct result of the initial conditions or may result during the instruction's execution. The bit remains set until the rung makes a false-to-true transition.
.IP (In process) Bit 26	It is set when the instruction has been successfully initiated and remains set until: another MDS instruction supersedes the initial instruction another instruction terminates the initial instruction a drive fault occurs.
STATE	Reflects the state of the instruction. 0 = Sending a request to the drive module to turn the drive on 1 = Waiting for the drive enable and servo action status bits to be set

Description

The MDS instruction:

- Is only valid for the AXIS_CIP_DRIVE axis data type.
- Performs a drive enable if the axis is not in the Running state.
- Applies desired DirectVelocityControlStatus Command attribute and/or the DirectTorqueControlStatus Command attributes.
- Presents the DirectVelocityControlStatus Command attributes and/or the DirectTorqueControlStatus Command attributes.

- Is activated on a Rung False-to-True transition.

The MDS instruction is used to activate the direct control of velocity or torque for a specified axis. The instruction performs an axis enable sequence and then presets the DirectVelocityControlStatus Command attribute and/or the DirectTorqueControlStatus Command attribute if the selected drive supports direct control. The most common usage of the MDS instruction is the flying start application, where the following attributes directly control the motion dynamics:

- RampAcceleration
- RampDeceleration
- RampVelocity - Positive
- RampVelocity - Negative
- RampJerk - Control

The DirectVelocityControlStatus Command attribute is applied by taking the value in the speed field in the instruction configuration and copying it into the DirectVelocityControlStatus Command attribute. The DirectVelocityControlStatus Command attribute is then scaled and summed onto the commanded output to the drive device. The DirectVelocityControlStatus Command attribute value can be modified directly via the MOV instruction.

The DirectTorqueControlStatus Command attribute is applied by taking the value in speed field in the instruction configuration and copying it into the DirectTorqueControlStatus Command attribute. The DirectTorqueControlStatus Command attribute is then sent directly to the drive via the placeholder in the CIP Controller to Drive Connection.

A combination of the DirectVelocityControlStatus Command attribute and the DirectTorqueControlStatus Command attribute can be applied in applications that require Speed-limited Adjustable Torque (SLAT) modes. SLAT operation mode provides automatic speed control under certain conditions.

The SLAT Configuration is an enumerated attribute that determines how the drive controls torque for this axis instance. To support applications that require SLAT control, the Min/Max torque control enumerations provide a feature to automatically switch to and from speed control under certain conditions. In either SLAT mode the drive operates in one of the states described in the following table.

Enumeration	Mode	Description
0	SLAT disabled	SLAT function is disabled. This is the normal Velocity Loop operation.
1	SLAT Min Speed/Torque	Drive automatically switches from torque control to speed control if Velocity Error > SLAT set point and switches back to torque control if Speed Error < 0.

2	SLAT Max Speed Torque	Drive automatically switches from torque control to speed control if Velocity Error < -SLAT set point and switches back to torque control if Speed Error > 0.

When you execute the MDS instruction and the drive is configured for velocity control, the acceleration and deceleration ramp to the specified speed is controlled by the drives based on the Acceleration Limit and Deceleration Limit attributes. The Motion Planner takes the value from the Direct Command Velocity attribute and sums it into the axis output before sending the command to the drive. The most common use of this instruction is to perform a Drive Start application into a spinning motor, also known as a Flying Start application.

The axis remains in DirectVelocityControlStatus Command attribute or DirectTorqueControlStatus Command attribute modes until canceled by one of the following instructions:

- Motion Axis Stop (MAS)
- Motion Axis Shutdown (MASD)
- Motion Coordinated Shutdown (MCSD)
- Motion Group Shutdown (MGSD)
- Motion Servo Off (MSF)

Depending on how the fault action is configured, an axis fault can also cancel the MDS instruction.

Execution of the MDS instruction has no effect on motion group or coordinate system objects. However, the instruction affects axis objects as follows:

When the MDS instruction is initiated without errors, the DirectVelocityControlStatus bit of the MotionStatus axis attribute is set, indicating the DirectVelocityControlStatus bit is active on the axis.

The DirectVelocityControlStatus bit remains set until it is made inactive via an MAS or MASR instruction, or via an axis fault.

Also, when the MDS instruction is initiated without errors, the DirectTorqueControlStatus bit attribute of the MotionStatus axis attribute is set, indicating the DirectTorqueControlStatus Command attribute is active on the axis.

The DirectTorqueControlStatus bit remains set until it is made inactive via an MAS or MASR instruction, or via an axis fault.

Some fault actions impact the execution of the MDS instruction.

Fault Action	Description
Ignore	Ignore instructs the device to completely ignore the exception condition. For some exceptions that are fundamental to the operation of the axis, it may not be possible to Ignore the condition.
Alarm	Alarm action instructs the device to set the associated bit in the Axis Alarm word but to otherwise not affect axis behavior. For some exceptions that are fundamental to the operation of the device, it may not be possible to select this action or any other action that leaves device operation unaffected.

Fault Action	Description
Fault Status Only	Fault Status Only instructs the device to set the associated bit in the Axis Faults word but to otherwise not affect axis behavior. It is up to the controller to programmatically bring the axis to a stop in this condition. For some exceptions that are fundamental to the operation of the device, it may not be possible to select this action or any other action that leaves device operation unaffected.
Stop Planner	Stop Motion instructs the device to set the associated bit in the Axis Faults word and instructs the Motion Planner to perform a controlled stop of all planned motion at the configured Max Decel rate but otherwise not affect axis behavior. This allows the axis to be subsequently moved via the Motion Planner without first clearing the fault. For some exceptions that are fundamental to the operation of the device, it may not be possible to select this action or any other action that leaves device enabled.
Stop Drive	The Stop Drive action results in the device both setting the associated bit in the Axis Faults word and bringing the axis to a stop based on the factory set "best" available stopping method. This "best" stopping method includes both the method of decelerating the motor to a stop and the final state of the axis given the expected level of control still available. The level of axis control available depends on the specific exception condition and on the configured control mode. The available deceleration methods are defined by the Stopping Mode attribute. Standard stopping modes, listed in decreasing levels of deceleration control, are as follows: Ramp Decel Current Limit Decel Coast In general, the "best" stopping mode is the most controlled deceleration method still available given the exception condition. The final state of the axis in response to the Major Fault exception action can be any one of the following states that are listed in decreasing levels of control functionality: Hold (Stopped state with Holding Torque) Disable (Stopped state with Power Structure Disabled) Shutdown (DC Bus Power Disabled) The "best" final state of the axis is the state with the most control functionality still available given the exception condition. But in all these final states a fault reset must be executed before the axis can be restored to enabled operation and commanded to move. If the application requires exception action that is a more severe stopping action than the factory set "best" method, the controller must initiate that action. If the application requires exception action that is less severe than the factory set "best" method, the controller must configure the device axis instance for a Minor Fault exception action and handle the fault directly. This may put device and motor components at risk and should only be allowed by the device when there is an opportunity, albeit temporal, for the device to remain operational. This is important in applications where the value of the product is higher than the value of the motor or device. When multiple major faults occur with different stopping actions, the most severe of the associated stopping actions is applied, i.e. the stopping action that requires the lowest level of control functionality. This rule also applies to the stopping action associated with the Stopping Mode associated with a Disable Request.
Shutdown	Shutdown forces the axis into the Shutdown state, immediately disabling the drive's power structure. If Shutdown Action is configured to do so, this action also drops DC Bus power to the drive's power structure. Therefore, the Shutdown action overrides the drive's best stopping method. An explicit Shutdown Reset is required to restore the drive to an operational state

This is a transitional instruction:

- In relay ladder, toggle the Rung-condition-in from false to true each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults. A major fault can occur if an uninitialized reference or reference of the incorrect type is passed to the Axis operand.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, and .ER are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes. If the EN bit is set to false, there is no action taken,
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See *Motion Error Codes (.ERR)* for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. See *Motion Error Codes (.ERR)* for Motion Instructions.

MDS Changes to Status Bits

Axis Status Bits

Bit Name	State	Meaning

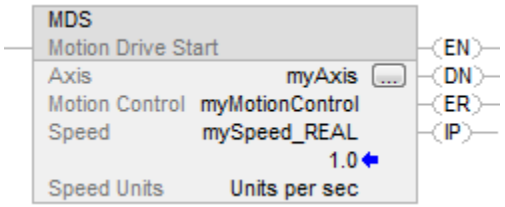
DriveVelocityControlStatus	FALSE	Axis is not under Direct Velocity Control.
DirectTorqueControlStatus	FALSE	Axis is not under Direct Torque Control.

Examples

Example 1

The Speed operand is a REAL tag and Speed Units is "Units per sec"

Ladder Diagram



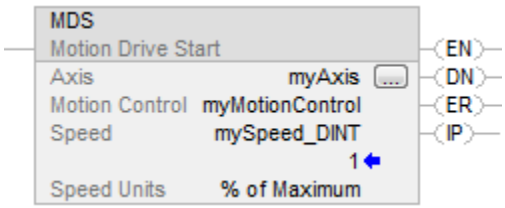
Structured Text

MDS(myAxis,myMotionControl,mySpeed_REAL,Unitspersec);

Example 2

The Speed operand is a DINT tag and Speed Units is % of Maximum

Ladder Diagram



Structured Text

MDS(myAxis,myMotionControl,mySpeed_DINT,%ofMaximum);

Motion Servo Off (MSF)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Use the Motion Servo Off (MSF) instruction to deactivate the drive output for the specified axis and to deactivate the axis' servo loop.

IMPORTANT If you execute an MSF instruction while the axis is moving, the axis coasts to an uncontrolled stop.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block.

Structured Text

MSF (Axis,MotionControl);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description
Axis	AXIS_CIP_DRIVE	AXIS_CIP_DRIVE AXIS_GENERIC AXIS_GENERIC_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE **Tip:** AXIS_GENERIC is supported by the ControlLogix 5570 and the GuardLogix 5570 controllers only.	Tag	Name of the axis on which to perform the operation. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axis' servo action has been successfully disabled and the drive enable and servo active status bits have been cleared.
.ER (Done) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MSF instruction directly and immediately turns off drive output and disables the servo loop on any physical servo axis. With non-CIP motion, this places the axis in the Axis Ready state described in Motion State Instructions. With CIP motion, this places the axis in the Stopped state described in Motion State Instructions. The MSF instruction also disables any motion planners that may be active at the time of execution. The MSF instruction requires no parameters – simply enter or select the desired axis.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

You can use the MSF instruction to turn servo action off when you must move the axis by hand. Since the position continues to be tracked even with servo action off. When the servo loop is turned on again, by the Motion Servo On (MSO) instruction, the axis is again under closed-loop control, at the new position.

The axis stopping behavior varies depending upon the type of drive. In some cases the axis coasts to a stop and in other cases the axis decelerates to a stop using the drive's available stopping torque.

For non-CIP motion, to execute an MSF instruction successfully, the targeted axis must be configured as a Servo axis. If this condition is not met, the instruction errs. If you have an Axis Type of Virtual the instructions errors because with a Virtual Axis the servo action and drive enable status are forced to always be true. A Consumed axis data type also errors because only the producing controller can change the state of a consumed axis.

IMPORTANT The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

Additionally, for CIP motion, the MSF instruction supports canceling the Motion Drive Start (MDS) instruction. This includes clearing the MDS In Process (.IP) bit, and clearing the DirectVelocityControlStatus bit and the DirectTorqueControlStatus bit in the Motion Status attribute.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Master Driven Speed Control (MDSC) and the MSF Instruction

If an MSF is issued in Master Driven Mode, the system shuts the servo off.

The IP bit of the Master Driven Axis Control (MDAC) instruction does not change on an axis that has its servos turned off.

The AC bit of the MDAC instruction resets when the axis stops as the servos are turned off.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or reference of the incorrect type is passed to the Axis operand. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition	Ladder Diagram Action
Prescan	The .EN, .DN, and .ER bits are cleared. The rung-condition-out is set to false.
Rung-condition-in is false	The .EN bit is cleared if the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is cleared to false if the .DN or .ER bit is true
Postscan	The rung-condition-out is set to false.

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See *Motion Error Codes (.ERR)* for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions.

MSF Changes to Status Bits

Axis Status Bits

Bit Name	State	Meaning
ServoActionStatus	FALSE	Axis is in Servo Off state with the servo loop inactive.
DriveEnableStatus	FALSE	Axis Drive Enable output is active.

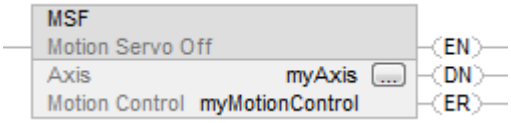
Motion Status Bits

Bit Name	State	Meaning
AccelStatus	FALSE	Axis is not Accelerating.
DecelStatus	FALSE	Axis is not Decelerating.
MoveStatus	FALSE	Axis is not Moving.
JogStatus	FALSE	Axis is not Jogging.
GearingStatus	FALSE	Axis is not Gearing.
HomingStatus	FALSE	Axis is not Homing.
StoppingStatus	FALSE	Axis is not Stopping.
PositionCamStatus	FALSE	Axis is not Position Camming.
TimeCamStatus	FALSE	Axis is not Time Camming.
PositionCamPendingStatus	FALSE	Axis does not have a Position Cam Pending.
TimeCamPendingStatus	FALSE	Axis does not have a Time Cam Pending.
GearingLockStatus	FALSE	Axis is not in a Gear Locked condition.
PositionCamLockStatus	FALSE	Axis is not in a Cam Locked condition.
DirectVelocityControlStatus	FALSE	Axis is not under Direct Velocity Control.
DirectTorqueControlStatus	FALSE	Axis is not under Direct Torque Control.

Example

When the input conditions are true, the controller disables the servo drive and the axis servo loop configured by Axis0.

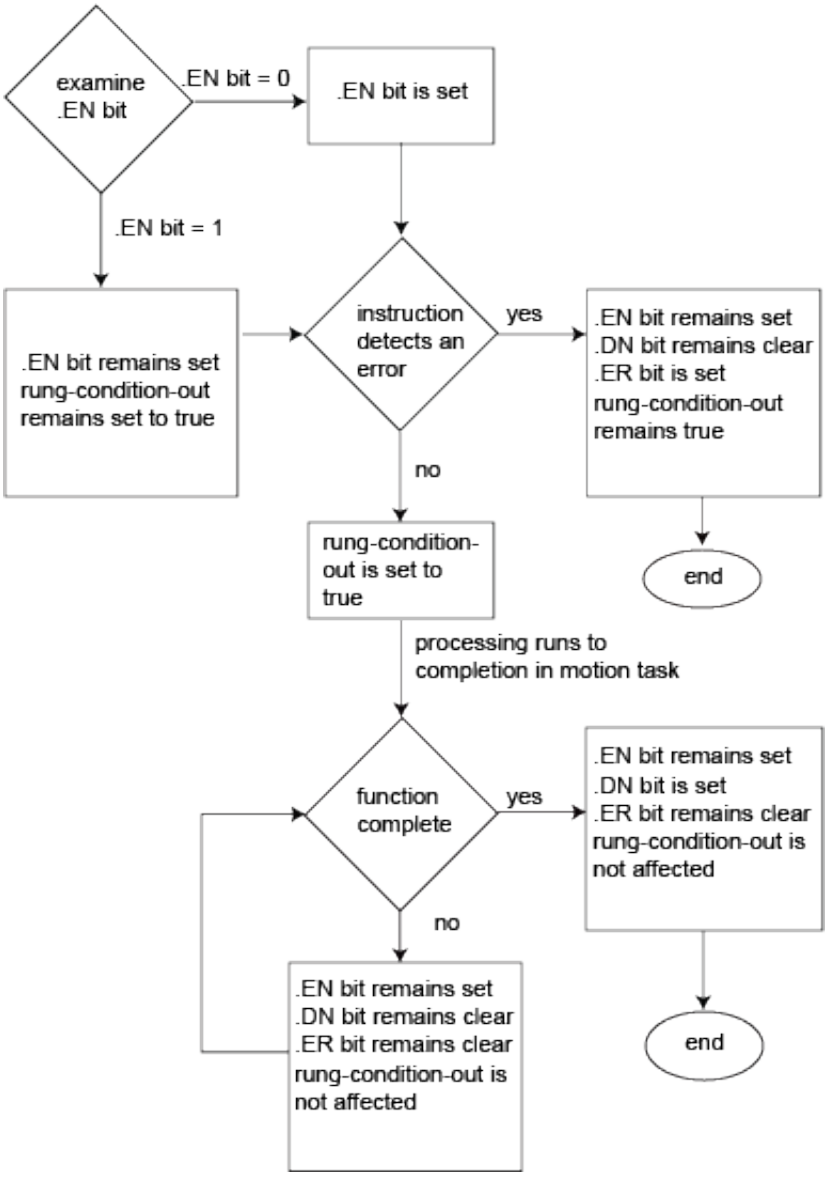
Ladder Diagram



Structured Text

MSF(myAxis, myMotionControl);

MSF Flow Chart (True)



Motion Servo On (MSO)

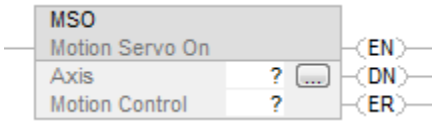
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Use the Motion Servo On (MSO) instruction to activate the drive amplifier for the specified axis and to activate the axis' servo control loop.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MSO(Axis,MotionControl);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description
Axis	AXIS_CIP_DRIVE	AXIS_CIP_DRIVE AXIS_GENERIC AXIS_GENERIC_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE **Tip:** AXIS_GENERIC is supported by the ControlLogix 5570 and the GuardLogix 5570 controllers only.	tag	Name of the axis on which to perform the operation. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION		tag	Structure used to access instruction status parameters.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

Mnemonic	Description

.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the axis' servo action has been successfully enabled and the drive enable and servo active status bits have been set.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MSO instruction directly activates the drive and enables the configured servo loops associated with a physical servo axis. It can be used anywhere in a program, but should not be used while the axis is moving. If this is attempted, the MSO instruction generates an Axis in Motion error.

The MSO instruction automatically enables the specified axis by activating the drive and by activating the associated servo loop. With a non-CIP axis, the resulting state of the axis is referred to as the Servo Control state. With a CIP axis, the resulting state of the axis is referred to as the Running state.

The most common use of this instruction is to activate the servo loop for the specified axis in its current position in preparation for commanding motion.

IMPORTANT The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or reference of the incorrect type is passed to the Axis operand. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, and .ER are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is true.

Rung-condition-in is true	The .EN bit is set to true and the instruction executes. If the EN bit is set to false, there is no action taken,
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See *Motion Error Codes (.ERR)* for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. See *Motion Error Codes (.ERR)* for Motion Instructions. These Extended Error codes help to pinpoint the problem when the MSO instruction receives a Servo Message Failure (12) error message.

Extended Error Code (decimal)	Associated Error Code (decimal)	Meaning
Object Mode conflict (12)	SERVO_MESSAGE_FAILURE (12)	Axis is shutdown.
Process terminated on request (15)	SERVO_MESSAGE_FAILURE (12)	Enable input switch error. (SERCOS)
Device in wrong state (16)	SERVO_MESSAGE_FAILURE (12)	Device State not correct for action. (SERCOS)

MSO Changes to Status Bits

Axis Status Bit

Bit Name	State	Meaning
ServoActionStatus	TRUE	Axis is in Servo Control state with the servo loop active.
DriveEnableStatus	TRUE	The axis drive enable output is active.

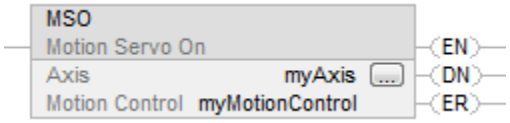
Motion Status Bits

None

Examples

Example 1

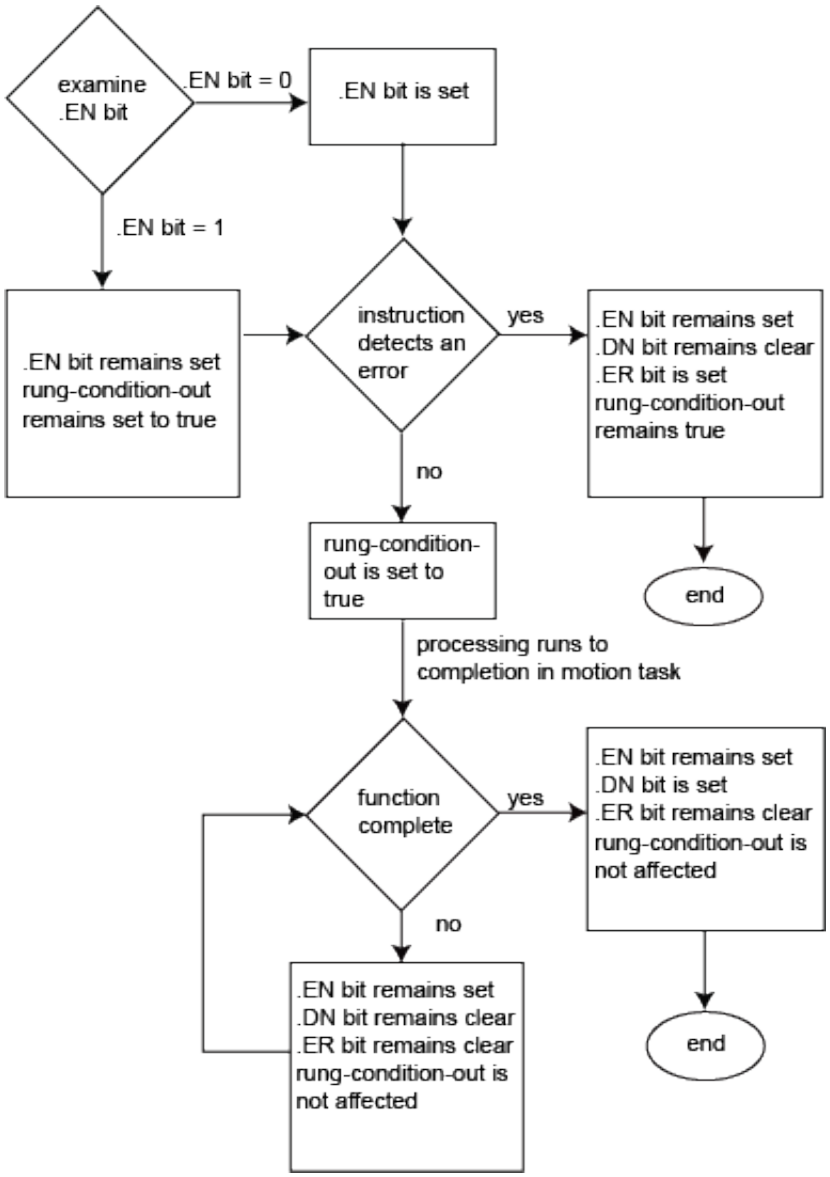
Ladder Diagram



Structured Text

MSO(myAxis, myMotionControl);

MSO Flow Chart (True)



Motion Move Instructions

Motion Move Instructions

Use the Motion Move instructions to control axis position.

Available Instructions

Ladder Diagram and Structured Text

MAS	MAH	MAJ	MAM	MAG	MCD	MRP	MCCP	MCSV	MAPC	MATC	MDAC

Function Block

Not available

IMPORTANT Tags used for the motion control attribute of instructions should only be used once. Re-use of the motion control tag in other instructions can cause unintended operation. This may result in damage to equipment or personal injury.

The Motion Move instructions are:

If you want to:	Use this instruction:
Stop any motion process on an axis.	MAS
Home an axis.	MAH
Jog an axis.	MAJ
Move an axis to a specific position.	MAM
Start electronic gearing between two axes.	MAG
Change the speed, acceleration, or deceleration of a move or a jog that is in progress.	MCD
Change the command or actual position of an axis.	MRP
Calculate a Cam Profile based on an array of cam points.	MCCP
Calculate the slave value, slope, and derivative of the slope for a cam profile and master value.	MCSV
Start electronic camming between two axes.	MAPC
Start electronic camming as a function of time.	MATC
Define a Master/Slave relationship between two motion axes and select which type of move instructions.	MDAC

See also

Motion Configuration Instructions on page 315

Motion State Instructions on page 23

Multi-Axis Coordinated Motion Instructions on page 359

Motion Event Instructions on page 249

Motion Group Instructions on page 225

Motion Axis Stop (MAS)

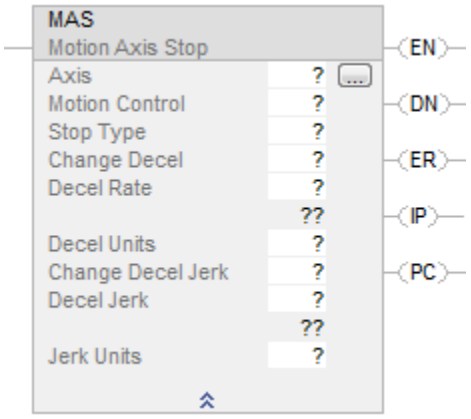
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Use the Motion Axis Stop (MAS) instruction to stop a specific motion process on an axis or to stop the axis completely.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available for function block.

Structured Text

MAS(Axis,MotionControl,StopType,ChangeDecel,DecelRate,DecelUnits,ChangeDecelJerk,DecelJerk,Jerk Units)

Operands

Ladder Diagram

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description	
Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_GENERIC AXIS_GENERIC_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE **Tip:** AXIS_GENERIC is supported by the ControlLogix 5570 and the GuardLogix 5570 controllers only.	Tag	Name of the axis. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.	
Motion Control	MOTION_INSTRUCTION		Tag	Control tag for the instruction.	
Stop Type	DINT		Immediate	**To Stop**	**Choose this Stop Type**

				All motion in process for this axis.	All = 0 With this selection, the instruction stops all motion on an axis. The stops takes any coordinated motion on the axis into account when it computes the Decel rate and stops that component of the coordinated motion. The other axes components of the coordinated motion are unaffected and continue. If this instruction stops a Motion Drive Start (MDS) instruction, the Direct Control Feature is disabled and the affected axis is decelerated to a stop using the instruction parameters.
				Only a certain type of motion but leave other motion processes running.	Choose the type of motion that you want to stop:
					Jog = 1
					Move = 2
					Gear = 3
					Home = 4
					Tune = 5
					Test = 6
					Time Cam = 7
					Position Cam = 8
					Master Offset Move = 9
					Direct Control = 10[1]

					The axis could still be moving when the MAS instruction is complete.
Change Decel	DINT		Immediate	**If you want to**	**Then Choose**
				Use the Maximum Deceleration rate of the axis.	No = 0
				Specify the deceleration rate.	Yes = 1
Decel Rate	REAL		Immediate or Tag	**Important:** The axis could overshoot its target position if you reduce the deceleration while a move is in process. Deceleration rate of the axis. The instruction uses this value only if Change Decel is Yes.	
Decel Units	DINT		Immediate	Which units do you want to use for the Decel Rate? • Units per sec^2 (0) • % of Maximum (1)	
Change Decel Jerk	DINT		Immediate	**If you want to**	**Then Choose**
				Use the Maximum Deceleration Jerk rate of the axis.	No (0)
				Program the deceleration jerk rate.	Yes (1)
Decel Jerk	REAL		Immediate or Tag	**Important:** The axis could overshoot its target position	

Jerk Units	DINT		Immediate	if you reduce the deceleration jerk while a move is in process. You **must** always enter a value for the Decel Jerk operand. This instruction only uses the value if the Profile is configured as S-curve. Decel Jerk is the deceleration jerk rate of the axis. Use these values to get started. Decel Jerk = 100 (% of Time) 0 = Units per sec^3 1 = % of Maximum 2 = % of Time (use this value to get started)

(1) When the MAS instruction is used to with either an All or DirectVelocityControlStatus Command Stop Type, the selection also clears the MDS In Process (.IP) bit, and clears the Axis DirectVelocityControlStatus bit in the Motion Status attribute

Structured Text

This Operand	Has These Options Which You	
	Enter as Text	**Or Enter as a Number**
Stop Type	all jog move gear home tune test timecam positioncam masteroffsetmove directcontrol	0 1 2 3 4 5 6 7 8 9 10
Change Decel	no yes	0 1
Decel Units	Units per sec^2 %ofmaximum	0 1
Change Decel Jerk	no yes	0 1
Decel Jerk	no enumeration	You **must** always enter a value for the Decel Jerk operand. This instruction only uses the value if the Profile is configured as S-curve. Decel Jerk is the deceleration jerk rate of the axis. Use this value to get started. Decel Jerk = 100 % of Time (2)

Jerk Units	unitspersec^3	0
	% of Maximum	1
	% of Time	2

See Structured Text Syntax for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Structure

To See If	Check To See If This Bit Is Set To	Data Type	Notes
A false-to-true transition caused the instruction to execute.	EN	BOOL	The EN bit stays set until the process is complete and the rung goes false.
The stop was successfully initiated.	DN	BOOL	
An error happened.	ER	BOOL	
The axis is stopping.	IP	BOOL	Any of these actions end the MAS instruction and clear the IP bit: • The axis is stopped • Another MAS instruction supersedes this MAS instruction • Shutdown command • Fault Action
The axis is stopped.	PC	BOOL	The PC bit stays set until the rung makes a false-to-true transition.

Description

Use the MAS instruction when you want a decelerated stop for any controlled motion in process for the axis. The instruction stops the motion without disabling the servo loop. A trapezoidal profile is always used for MAS with Stop Type=ALL for the deceleration regardless of the programmed profile type. Use the instruction to:

- stop a specific motion process such as jogging, moving, or gearing
- stop the axis completely
- abort a test or tune process initiated by a Motion Run Hookup Diagnostics (MRHD) instruction or Motion Run Axis Tuning (MRAT) instruction.

If the Stop Type is	Then the MAS Instruction Uses This Profile
Jog	Same type of profile as the Motion Axis Jog (MAJ) instruction that started the jog.
Move	Same type of profile as the Motion Axis Move (MAM) instruction that started the move.
None of the above	Trapezoidal.

When MAS (Stop Type = All) is used on any axis associated with a coordinate system and a coordinated motion instruction is running on it, Coordinate system's maximum deceleration is used to stop remaining axes. If the coordinate system contains orientation axes, Coordinate system's Orientation maximum deceleration is used for stopping remaining Rx, Ry or Rz axes.

Example

Suppose you use a Motion Axis Jog (MAJ) instruction with an S-curve profile to start a jog. Then you use an MAS instruction with a Stop Type of Jog to stop the jog. In that case, the MAS instruction uses an S-curve profile to stop the jog.

Programming Guidelines

WARNING: **Risk of Velocity and/or End Position Overshoot**

If you change move parameters dynamically by any method, that is by changing move dynamics [Motion Change Dynamics (MCD) instruction or Motion Coordinated Change Dynamics (MCCD)] or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-curve velocity profile can overshoot if:

- Maximum deceleration is decreased while the move is decelerating or close to the deceleration point; or
- Maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

For more information, see Troubleshooting Axis Motion.

Guidelines	Details	
In ladder diagram, toggle the rung condition each time you want to execute the instruction.	This is a transitional instruction: • In ladder diagram, toggle the rung-condition-in from cleared to set each time you want to execute the instruction.	
In structured text, condition the instruction so that it only executes on a transition.	In structured text, instructions execute each time they are scanned. • Condition the instruction so that it only executes on a transition. Use either of these methods: • Qualifier of an SFC action • Structured text construct For more information, see Structured Text Syntax.	
Choose whether to stop all motion or only a specific type of motion.	**If You Want to Stop**	**Then Choose This Stop Type**
	All motion in process for this axis.	All The instruction uses a trapezoidal profile and stops the axis.
	Stop only a certain type of motion but leave other motion processes running.	The type of motion that you want to stop The axis could still be moving when the MAS instruction is complete. The instruction uses an S-curve profile to stop the axis only if: • The Stop Type is Jog or Move, and • The jog or move used an S-curve profile.
	Example: Suppose your axis is executing both a jog and a move at the same time. And suppose you want to stop only the jog but leave the move running. In that case, choose a Stop Type of Jog.	

To stop gearing or camming, select the slave axis.	To stop a gearing or position camming process, enter the **slave** axis to turn off the specific process and stop the axis. If the master axis is a servo axis, you can stop the master axis which in turn stops the slave without disabling the gearing or position camming. **Tip:** If the Master axis is moved manually while the Master axis was in a disabled state, the actual position of the slave axis continues to track the Master's position, regardless whether the MasterReference of the MAPC instruction is set to Actual or Command.
To stop a Master Offset move, enter the slave axis but use master units.	To stop an Absolute or Incremental Master Offset move: • For Axis, enter the slave axis. • For Deceleration and Jerk, enter the values and units for the master axis.
Be careful if the instruction changes motion parameters.	When you execute an MAS instruction, the axis uses the new deceleration and jerk rates for the motion that's already in process. This can cause an axis to overshoot its speed, overshoot its end position, or reverse direction. S-curve profiles are more sensitive to parameter changes. For more information, see *Troubleshoot Axis Motion*.
Use the jerk operands for S-curve profiles.	Use the jerk operands when • The Stop Type is Jog or Move. • The jog or move uses an S-curve profile. Under those conditions, the instruction uses an S-curve profile to stop the axis. The instruction uses a constant deceleration rate for all other types of stops. You must fill in the jerk operands regardless of the type of stop.
Use % of Time for the easiest programming and tuning of jerk.	For an easy way to program and tune jerk, enter it as a % of the acceleration or deceleration time. For more information, see *Tune an S-curve Profile*.

Master Driven Speed Control (MDSC) and the MAS Instruction

If the MAS is executed (goes IP) on the Slave Axis in Master Driven Mode, the MDSC link is immediately broken. The Slave always stops in Time Driven Mode, regardless if the active mode is Time Driven or Master Driven.

Stop Type for MAS

The behavior of the MAS instruction depends on the Stop Type parameter:

- Time CAM, stop type moves, and jogs - When you invoke an MAS with Stop Type = Time Cam on an axis that is running in Master Driven Mode, the axis terminates Master Driven Mode, changes to Time Driven Mode, and stops as specified.
- Master Offset Move - When you invoke an MAS with Stop Type = Master Offset Move on an axis that is running in Master Driven Mode and controlling a Position Cam, the offset axis terminates Master Driven Mode, changes to Time Driven Mode, and stops at the specified deceleration and deceleration jerk.
- All - When you invoke an MAS with Stop Type = All, the MDSC link is broken immediately. All motion planners on the specified axis terminate Master Driven Mode and axes are stopped at the specified deceleration and deceleration jerk in Time Driven Mode.

If the axes are involved in any coordinated motion (in Master Driven Mode), then the axes break from the coordinate motion planner and the other axes in the coordinate system continue. The IP bit is cleared on the Master Driven Axis Control (MDAC).

Note that MAS ALL on the Master axis does not break the MDSC link. For all other stop types in which you execute an MAS instruction on an axis that is being controlled in Master Driven Mode, the motion planner terminates Master Driven Mode on the requested axis for the requested Stop Type, changes to Time Driven Mode, and stops at the specified deceleration and deceleration jerk in Time Driven Mode. Other stop types do not clear the IP bit on the MDAC.

The AC bit of the MDAC instruction resets when the axis is stopped.

MAS All reset the IP bit of the MDAC. Other stop types do not reset the IP bit.

Note that if a stop is issued very close to the programmed endpoint, the actual stop may be beyond the programmed endpoint, especially if run in Master Driven Mode.

The status bit CalculatedDataAvailable in an active motion instruction status word for an MAS instruction resets when an MAS is executed (goes IP). The CalculatedData is not recomputed.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis operand. See Common Attributes for operand related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in Ladder Diagram table
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in Ladder Diagram table.

Error Codes

See Motion Error Codes (.ERR).

Extended Error Codes

Use Extended Error Codes (.EXERR) for more instructions about an error. See Motion Error Codes (.ERR).

If ERR is	And EXERR is	Then		
		Cause	Corrective Action	
13	Varies	An operand is outside its range.	The EXERR is the number of the operand that is out of range. The first operand is 0. For example, if EXERR = 4, then check the Decel Rate.	
			EXERR	MAS Operand
			0	Axis
			1	Motion Control
			2	Stop Type
			3	Change Decel
			4	Decel Rate

MAS Changes to Single Axis Status Bits

Motion Status Bits

If the Stop Type Is	Then		
NOT All	The instruction clears the Motion Status bit for the motion process that you stopped.		
All	The instruction clears all Motion Status bits.		
	Bit Name	State	Meaning
	MoveStatus	FALSE	Axis is not Moving.
	JogStatus	FALSE	Axis is not Jogging.
	GearingStatus	FALSE	Axis is not Gearing.
	HomingStatus	FALSE	Axis is not Homing.
	StoppingStatus	TRUE	Axis is Stopping.
	PositionCamStatus	FALSE	Axis is not Position Camming.
	TimeCamStatus	FALSE	Axis is not Time Camming.
	PositionCamPendingStatus	FALSE	Axis does not have a Position Cam Pending.

	TimeCamPendingStatus	FALSE	Axis does not have a Time Cam Pending.
	GearingLockStatus	FALSE	Axis is not in a Gear Locked condition.
	PositionCamLockStatus	FALSE	Axis is not in a Cam Locked condition.
	DirectVelocityControlStatus	FALSE	Axis is not under Direct Velocity Control.
	DirectTorqueControlStatus	FALSE	Axis is not under Direct Torque Control.

Motion Axis Home (MAH)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Use the Motion Axis Home (MAH) instruction to home an axis. Two different homing modes can be selected during axis configuration: Active or Passive. If an Active homing sequence is selected, the axis executes the configured Home Sequence Type and establishes an absolute axis position. If Passive homing is selected, however, no specific homing sequence is executed and the axis is left waiting for the next marker pulse to establish the home position.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block

Structured Text

MAH(Axis,MotionControl);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_GENERIC_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to perform operation on. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control		MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

See Structured Text Syntax for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when axis home has been successfully completed or is aborted.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.IP (In Process) Bit 27	It is set on positive rung transition and cleared after the Motion Home Axis is complete, or terminated by a stop command, shutdown, or a servo fault.
.PC (Process Complete) Bit 26	It is set when axis home is successfully completed.

Description

The MAH instruction is used to calibrate the absolute position of the specified axis. For axes that are configured as type Servo, the axis can be homed using Active, Passive, or Absolute Homing Mode configuration. For Feedback Only axes, only Passive and Absolute homing modes are available. Absolute Homing Mode requires the axis to be equipped with an absolute feedback device.

Important:	Absolute Homing mode is not available with a CIP axis. However, any successful Home sequence establishes an absolute position.

For CIP axes only, software overtravels are disabled if the Home status bit transitions to the FALSE state.

The MAH instructions sets the HomedStatus bit upon successful completion of the configured homing sequence. This bit indicates that an absolute machine reference position has been established. When this bit is set, operations that require a machine reference, such as Software Overtravel checking can be meaningfully enabled.

For non-CIP Drive axis data types, the HomedStatus bit is cleared under the following conditions:

- Download
- Control power cycle
- Reconnection to Motion Module
- Feedback Loss Fault
- Shutdown

Active Homing

When the axis Homing Mode is configured as Active, the physical axis is first activated for servo operation. As part of this process, all other motion in process is canceled and appropriate status bits are cleared. The axis is then homed using the configured Home Sequence, which may be Immediate, Switch, Marker, or Switch-Marker. The latter three Home Sequences result in the axis being jogged in the configured Home Direction and then, after the position is re-defined based on detection of the home event, the axis is automatically moved to the configured Home Position.

Passive Homing

When the axis Homing Mode is configured as Passive, the MAH instruction re-defines the actual position of a physical axis on the next occurrence of the encoder marker. Passive homing is most commonly used to calibrate Feedback Only axes to their markers, but can also be used on Servo axes. Passive homing is identical to active homing to an encoder marker, except that the motion controller does not command any axis motion.

After initiating passive homing, the axis must be moved past the encoder marker for the homing sequence to complete properly. For closed-loop Servo axes, this may be accomplished with a MAM or MAJ instruction. For physical Feedback Only axes, motion cannot be commanded directly by the motion controller, and must be accomplished via other means.

Absolute Homing

If the motion axis hardware supports an absolute feedback device, Absolute Homing Mode may be used. The only valid Home Sequence for an absolute Homing Mode is "immediate". In this case, the absolute homing process establishes the true absolute position of the axis by applying the configured Home Position to the reported position of the absolute feedback device. Prior to execution of the absolute homing process via the MAH instruction, the axis must be in the Axis Ready state with the servo loop disabled.

To successfully execute a MAH instruction on an axis configured for Active homing mode, the targeted axis must be configured as a Servo Axis Type. To successfully execute an MAH instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. If any of these conditions are not met, the instruction errors.

Important:	The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Master Driven Speed Control (MDSC) and the MAH Instruction

When either an MDAC or MDCC is active:

- If a MAH is executed (goes IP) on a Master axis and the Master axis is not moving, the MDAC or MDCC remains active. If MDAC or MDCC is active while the MAH is executed (goes IP), the slave moves.
- If an MAH is executed (goes IP) on a Master axis and the Master axis is moving, the MAH will error. The MDAC or MDCC state remains unchanged.
- If an MAH is executed (goes IP) on a Slave axis and the Slave axis is not moving, the MDAC or MDCC is canceled.
- If an MAH is executed (goes IP) on a Slave axis while it is moving in MDSC Driven or Time Driven modes, the MAH will error. The state of MDAC or MDCC remains unchanged.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis operand. See Common Attributes for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in Ladder Diagram table
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in Ladder Diagram table.

Error Codes

See Motion Error Codes (.ERR) for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. See Motion Error Codes (.ERR) for Motion Instructions.

The following Extended Error codes help to pinpoint the problem when the MAH instruction receives a Servo Message Failure (12) error message or Illegal Homing Configuration (41).

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	Process terminated on request (1)	Home execution followed by an instruction to shutdown/disable drive, or a motion stop instruction or a Processor change requests a cancel of Home.

SERVO_MESSAGE_FAILURE (12)	No Resources (2)	Not enough memory resources to complete request. (SERCOS)
SERVO_MESSAGE_FAILURE (12)	Object Mode conflict (12)	Axis is in shutdown.
SERVO_MESSAGE_FAILURE (12)	Permission denied (15)	Enable input switch error. (SERCOS)
SERVO_MESSAGE_FAILURE (12)	Device in wrong state (16)	Redefine Position, Home, and Registration 2 are mutually exclusive (SERCOS), device state not correct for action. (SERCOS)
ILLEGAL_HOMING_CONFIG (41)	Home sequence (4)	The Home Sequence is incompatible with the Home Mode.
ILLEGAL_HOMING_CONFIG (41)	Home speed of zero (6)	Home speed cannot be zero.
ILLEGAL_HOMING_CONFIG (41)	Home return speed of zero (7)	The Home Return Speed cannot be zero.

For the Error Code 54 – Maximum Deceleration Value is Zero, if the Extended Error returns a positive number (0-n) it is referring to the offending axis in the coordinate system. Go to the Coordinate System Properties General Tab and look under the Brackets ([ ])column of the Axis Grid to determine which axis has a Maximum Deceleration value of 0. Click on the ellipsis button next to the offending axis to access the Axis Properties screen. Go to the Dynamics tab and make the appropriate change to the Maximum Deceleration Value. If the Extended Error number is -1, this means the Coordinate System has a Maximum Deceleration Value of 0. Go to the Coordinate System Properties Dynamics Tab to correct the Maximum Deceleration value.

Status Bits

MAH Changes to Single Axis Status Bits

Bit Name	State	Meaning
HomingStatus	TRUE	Axis is Homing.
JogStatus	FALSE	Axis is no longer Jogging.*
MoveStatus	FALSE	Axis is no longer Moving.*
GearingStatus	FALSE	Axis is no longer Gearing.
StoppingStatus	FALSE	Axis is no loner Stopping.

*During portions of the active homing sequence these bits may be set and cleared. The MAH instruction uses the Move and Jog motion profile generators to move the axis during the homing sequence. This also means that any disruption in the Move or Jog motion profiles due to other motion instructions can affect the successful completion of the MAH initiated homing sequence.

If in Passive homing mode, the MAH instruction simply sets the Homing Status bit.

See also

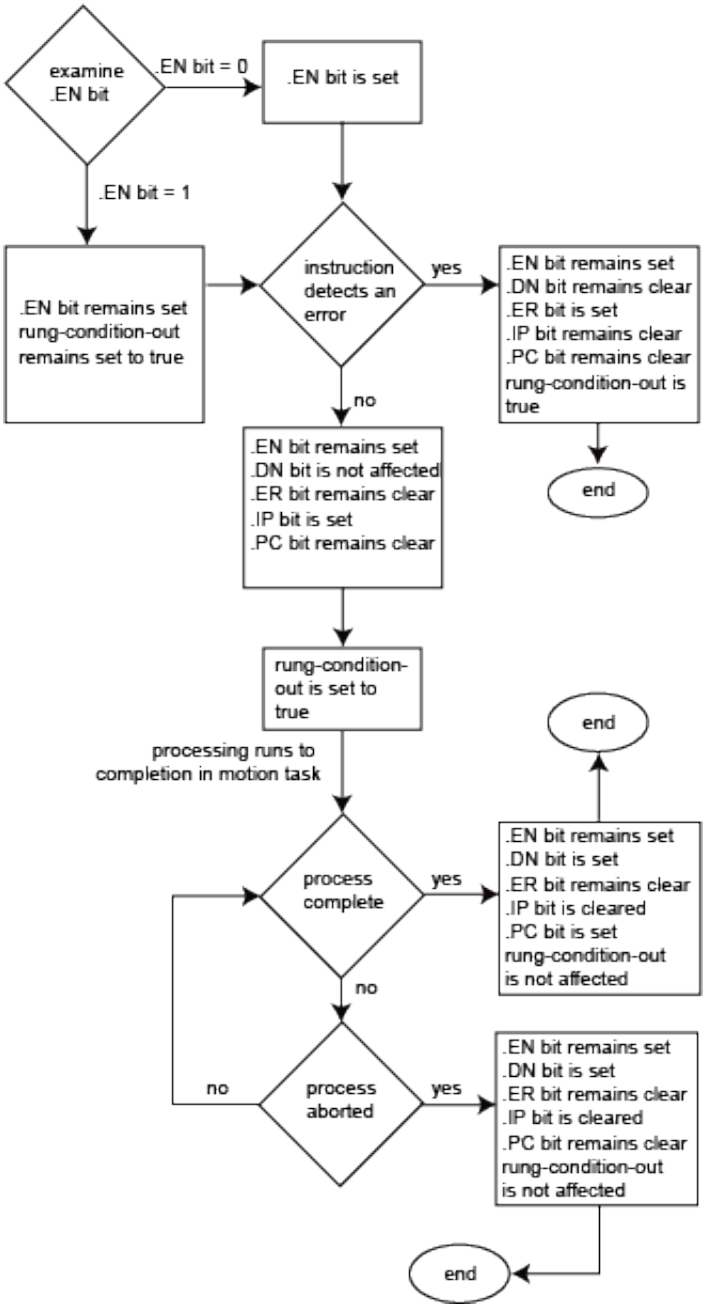
Structured Text Syntax on page 665

MAH Flow Chart (True) on page 90

Motion Error Codes (.ERR) on page 581

Common Attributes on page 691

MAH Flow Chart (True)



Motion Axis Jog (MAJ)

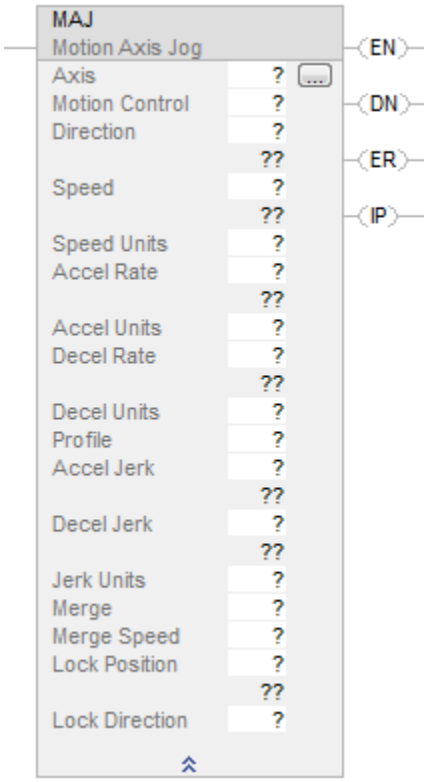
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Use the Motion Axis Jog (MAJ) instruction to move an axis at a constant speed until you tell it to stop.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MAJ(Axis,MotionControl, Direction,Speed, SpeedUnits, AccelRate, AccelUnits, DecelRate, DecelUnits, Profile,
AccelJerk, DecelJerk, JerkUnits, Merge, MergeSpeed LockPosition, LockDirection);

Operands

Operand	**Type** **CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480**	**Type** **ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers**	**Format**	**Description**

Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_GENERIC_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis to jog. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Control tag for the instruction.
Direction	DINT	DINT	Immediate	**For This Jog Direction** / **Enter** Forward / 0 Reverse / 1
Speed	REAL	REAL	Immediate or Tag	Speed to move the axis in Speed Units.
Speed Units	DINT	DINT	Immediate	Which units do you want to use for the Speed? • Units per sec (0) • % of Maximum (1) • Units per MasterUnits (4)
Accel Rate	REAL	REAL	Immediate or Tag	Acceleration rate of the axis in Accel Units.
Accel Units	DINT	DINT	Immediate	Which units do you want to use for the Accel Rate? • Units per sec^2 (0) • % of Maximum (1) • Units per MasterUnit^2 (4)
Decel Rate	REAL	REAL	Immediate or Tag	Deceleration rate to the axis in Deceleration Units.
Decel Units	DINT	DINT	Immediate	Which units do you want to use for the Decel Rate? • Units per sec^2 (0) • % of Maximum (1) • Units per MasterUnit^2 (4)
Profile	DINT	DINT	Immediate	Select the velocity profile to run the jog: • Trapezoidal (0) • S-curve (1)
Accel Jerk	REAL	REAL	Immediate or Tag	You **must** always enter values for the Accel and Decel Jerk operands. This instruction only uses the values if the Profile is
Decel Jerk	REAL	REAL	Immediate or Tag	

Jerk Units	DINT	DINT	Immediate	configured as S-curve. • Accel Jerk is the acceleration jerk rate for the axis • Decel Jerk is the deceleration jerk rate for the axis. Use these values to get started. • Accel Jerk = 100 (% of Time) • Decel Jerk = 100 (% of Time) • Jerk units = 2 Enter the jerk rates in these Jerk Units. 0 = Units per sec^3 1 = % of Maximum 2 = % of Time (use this value to get started) 4 = Units per MasterUnit^3 6 = % of Time-Master Driven
Merge	DINT	DINT	Immediate	Do you want to turn all current axis motion into a pure jog governed by this instruction regardless of the motion instructions currently in process? • NO — Choose Disabled (0) • YES — Choose Enabled (1)
Merge Speed	DINT	DINT	Immediate	If Merge is Enabled, which speed do you want to jog at? • Speed of this instruction — Choose Programmed = 0 • Current speed of the axis — Choose Current = 1
Lock Position	REAL	REAL	Immediate or Tag	Position on the Master Axis where a Slave should start following the master after the move has been initiated on the Slave Axis. See the Structure section below for more information.
Lock Direction	UINT32	UINT32	Immediate or Tag	Specifies the conditions when the Lock Position should be used. See the Structure section below for more information.

Structured Text

This Operand	Has These Options Which You	
	Enter as Text	Or Enter as a Number

Axis	No enumeration	Tag
MotionControl	No enumeration	Tag
Direction	No enumeration	Immediate
Speed	No enumeration	Immediate or Tag
SpeedUnits	units per sec % of maximum unitspermasterunit	0 1 4
AccelRate		
AccelUnits	units per sec^2 % of maximum unitspermasterunit^2	0 1 4
DecelRate		
DecelUnits	units per sec^2 % of maximum unitspermasterunit^2	0 1 4
Profile	Trapezoidal S-curve	0 1
AccelJerk	No enumeration	Immediate or Tag You **must** always enter a value for the Accel and Decel Jerk operands. This instruction only uses the values if the Profile is configured as S-curve. Use these values to get started. • Accel Jerk = 100 (% of Time) • Decel Jerk = 100 (% of Time)
DecelJerk	No enumeration	
Jerk Units	units per sec^3 %ofmaximum %oftime seconds unitspermasterunit^3 %oftimemasterdriven masterunits	0 1 2 (use this value to get started) 3 4 5 6
Merge	disabled enabled	0 1
Merge Speed	programmed current	0 1
Lock Position	No enumeration	Immediate or Tag
Lock Direction	none immediateforwardonly immediatereverseonly positionforward positionreverse	0 1 2 3 4

See Structured Text Syntax for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Structure

To See If	Check To See If This Bit Is Set To	Data Type	Notes

A false-to-true transition caused the instruction to execute.	EN	BOOL	The EN bit stays set until the process is complete and the rung goes false.
The jog was successfully initiated.	DN	BOOL	
An error happened.	ER	BOOL	
The axis is jogging.	IP	BOOL	Any of these actions stop this jog and clear the IP bit: • Another MAJ instruction supersedes this MAJ instruction. • Motion Axis Stop (MAS)instruction. • Merge from another instruction. • Shutdown command. • Fault Action.

Description

Use the MAJ instruction to move an axis at a constant speed without regard to position.

Programming Guidelines

Important: If you change move parameters dynamically by any method, that is by changing move dynamics [Motion Change Dynamics (MCD) instruction or Motion Coordinated Change Dynamics (MCCD)] or by starting a new instruction before the last one has completed, be aware of the risk of velocity or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-curve velocity profile can overshoot if:

- Maximum deceleration is decreased while the move is decelerating or close to the deceleration point; or
- Maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

For more information, see Troubleshooting Axis Motion.

Guidelines	Details
In ladder diagram, toggle the rung condition each time you want to execute the instruction.	This is a transitional instruction: In ladder diagram, toggle the rung-condition-in from cleared to set each time you want to execute the instruction.
In structured text, condition the instruction so that it only executes on a transition.	In structured text, instructions execute each time they are scanned. Condition the instruction so that it only executes on a transition. Use either of these methods: • qualifier of an SFC action • structured text construct For more information, see Structured Text Syntax.
Use the jerk operands for S-curve profiles.	Use the jerk operands when the instruction uses an S-curve profile. You must fill in the jerk operands regardless of the profile.
Use % of Time for the easiest programming and tuning of jerk.	For an easy way to program and tune jerk, enter it as a % of the acceleration or deceleration time. For more information, see Tune an S-curve Profile.
Use Merge to cancel the motion of other instructions.	How you want to handle any motion that's already in process?

	If You Want To	And You Want To	Then Set
	Add the jog to any motion already in process		Merge = Disabled Merge Speed = Programmed The instruction ignores Merge Speed but you must fill it in anyway.
	End the motion from other instructions and just jog	Jog at the speed that you set in this instruction	Merge = Enabled Merge Speed = Programmed
		Jog at the speed that the axis is already moving	Merge = Enabled Merge Speed = Current The instruction ignores the value that you put in the Speed operand.
Be careful if you start another jog while the axis is already jogging.	If you start a new MAJ instruction while one is already in process, you can cause: • an accelerating axis to overshoot its speed • a decelerating axis to reverse (revision 15 and earlier) This happens if the MAJ instructions use an S-curve profile. The new MAJ instruction cancels the old MAJ instruction. The axis uses the speed, acceleration, deceleration, and jerk of the new instruction. For more information, see Troubleshoot Axis Motion.		
Use an MAS instruction to stop the jog.			
Use an MCD instruction to change the speed while jogging.			

When MAJ (Merge = Enabled) is used on any axis associated with a coordinate system and a coordinated motion instruction is running on it, Coordinate system's maximum deceleration is used to stop remaining axes. If the coordinate system contains orientation axes, Coordinate system's Orientation maximum deceleration is used for stopping remaining Rx, Ry or Rz axes.

Structure

See Input and Output Parameters Structure for Single Axis Motion Instructions for the input and output parameters that are available for the MAJ instruction via the Master Driven Speed Control (MDSC) function. Before any of these parameters is active, you must execute an MDAC instruction and it must be active (IP bit is set).

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis operand. See Common Attributes for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See Rung-condition-in is false, followed by Rung-condition-in is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See Motion Error Codes (.ERR) for Motion Instructions.

Runtime Error Condition

The slave move must start at rest if Speed Units = Seconds or Master Units. This condition may occur when the MAJ with Speed = Seconds or Master Units is started while another MAJ is in progress (merging or replacement mode).

Extended Error Codes

Use Extended Error Codes (EXERR) for more instruction about an error.

If ERR is	And EXERR is	Then		
		Cause	Corrective Action	
13	Varies	An operand is outside its range.	The EXERR is the number of the operand that is out of range. The first operand is 0. For example, if EXERR = 3, then check the Speed.	
			EXERR	MAS Operand
			0	Axis
			1	Motion Control
			2	Direction
			3	Speed

15	-1	The coordinated system has a Maximum Deceleration of 0.	Go to the Properties for the coordinate system axis and set a Maximum Deceleration.
	0 or more	An axis in the coordinate system has a Maximum Deceleration of 0.	1. Open the Properties for the axis. 2. Use the EXERR value to see which axis has the Maximum Deceleration of 0. 3. The axis that you are jogging has a deceleration rate of 0.

Changes to Status Bits

Motion Instruction Predefined Data Type Status Bits

See Status Bits for Motion Instructions (MAM, MATC, MAJ) When MDAC Is Active.

MAJ Changes to Instruction Status Bits

Bit Name	**Meaning**	
MotionStatus	The motion status bit for your axis.	
	Bit Number	**Meaning**
AccelStatus	0	The axis is not accelerating (FALSE state).
DecelStatus	1	The axis is not decelerating (FALSE state).
MoveStatus	2	The axis is not moving (FALSE state).
JogStatus	3	The axis is not jogging (FALSE state).
GearingStatus	4	The axis is not gearing (FALSE state).
HomingStatus	5	The axis is not homing (FALSE state).
StoppingStatus	6	The axis is stopping (TRUE state).
AxisHomedStatus	7	The axis is not homed (FALSE state).
PositionCamStatus	8	The axis is not position camming (FALSE state).
TimeCamStatus	9	The axis is not time camming (FALSE state).
PositionCamPendingStatus	10	The axis does not have a Position Cam Pending (FALSE state).
TimeCamPendingStatus	11	The axis does not have a Time Cam Pending (FALSE state).
GearingLockStatus	12	The axis is not in a Gear Locked condition (FALSE state).
PositionCamLockStatus	13	The axis is not in a Position Cam Locked condition (FALSE state).
TimeCamLockStatus	14	The axis is not in a Time Cam Locked condition (FALSE state).
MasterOffsetMoveStatus	15	The axis is offset (TRUE state).
CoordinatedMotionStatus	16	Sets when the MDAC instruction executes (TRUE state). Clears when the instruction completes (FALSE state).
TransformStateStatus	17	The axis is part of an active transform (TRUE state).
ControlledByTransformStatus	18	The axis is moving because of a transform (TRUE state).
DirectVelocityControlStatus	19	The axis is not under Direct Velocity Control (FALSE state).
DirectTorqueControlStatus	20	The axis is not under Direct Torque Control (FALSE state).

JogLockStatus	24	MAJ is Locked to Master in MDSC Mode (TRUE state). The bit is cleared when a MGS, MGSD, MAS, or MASD is executed. If either the Slave or Master axis (or both) is paused by changing its speed to 0, then the JogLockStatus bit stays set. **Master Driven Mode** The bit is set when the Lock Direction request is satisfied. The bit is not used when the enumeration is NONE. For the enumerations Immediate Forward Only and Immediate Reverse Only, the JogLockStatus bit is set immediately when the MAJ is initiated. For the enumeration Position Forward Only and Position Reverse Only, the bit is set when the Master Axis crosses the Master Lock Position in the specified direction. The JogLockStatus bit is cleared when the Master Axis reverses direction and the Slave Axis stops following the Master Axis. The JogLockStatus bit is set again when the Slave Axis resumes following the Master Axis. **Time Driven Mode** The bit is not used when the enumeration is NONE.
MasterOffsetMoveLockStatus	26	Master offset Move is Locked to master in MDSC Mode (TRUE state).
MaximumSpeedExceeded	27	Sets when the maximum axis speed that is specified in the axis configuration is exceeded during a move (TRUE state). Clears when the velocity is reduced below the limit (FALSE state).

MAJ Changes to Single Axis Status Bits

If Merge Is	Then the Instruction Changes These Bits		
	Bit Name	**State**	**Meaning**
Disabled	JogStatus	TRUE	Axis is Jogging.
Enabled	JogStatus	TRUE	Axis is Jogging.
	MoveStatus	FALSE	Axis is no longer Moving.
	GearingStatus	FALSE	Axis is no longer Gearing.

Master Driven Speed Control (MDSC) and Motion Direct Command Support

The Motion Direct commands are not available in the instruction tree for the MDAC or MDCC instruction. You must program an MDAC in one of the supported programming languages before you execute an MAJ in Time Driven Mode. A runtime error will occur if an MDAC is not previously executed in an MAM or MAJ in Master Driven Mode.

The Motion Direct Command supports the MDSC enumerations speed, acceleration, deceleration, and Jerk for MAJ.

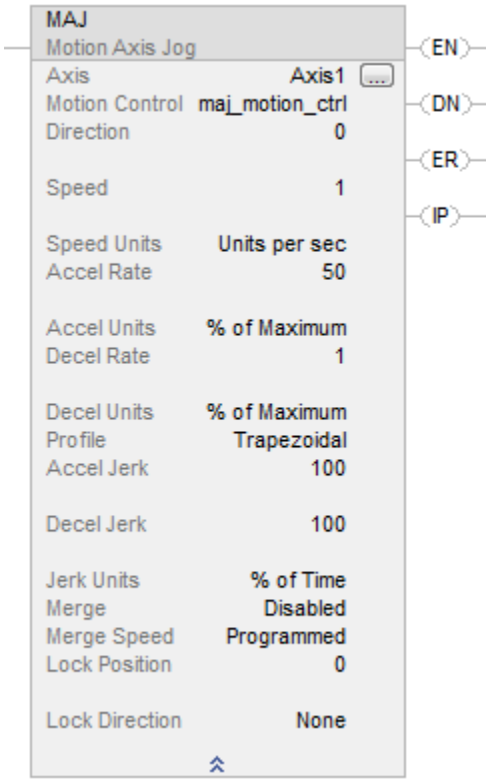
Note that Event Distance and Calculated Data are not supported parameters for the MAJ and Motion Direct Command.

Master Driven Speed Control (MDSC) and CIP Axis Manual Tune and Motion Generator

Event Distance and Calculated Data parameters are not supported for MAJ.

Examples

Ladder Diagram



See also

Troubleshoot Axis Motion on page 639

Structured Text Syntax on page 665

Status Bits for Motion Instructions (MAM,MATC,MAJ) When MDAC Is Active on page 572

Motion Error Codes (.ERR) on page 581

Common Attributes on page 691

Motion Axis Move (MAM)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380,

CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Use the Motion Axis Move (MAM) instruction to move an axis to a specified position.

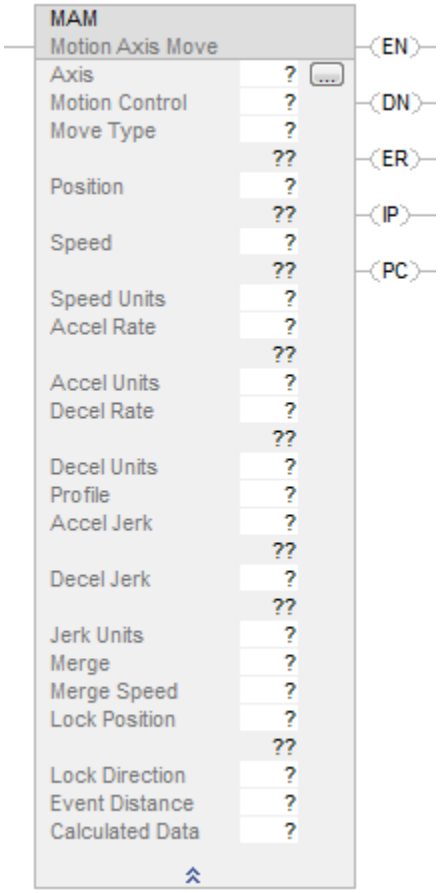
This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MAM(Axis, MotionControl, MoveType, Position, Speed, SpeedUnits, AccelRate, AccelUnits, DecelRate, DecelUnits, Profile, AccelJerk, DecelJerk, JerkUnits, Merge, MergeSpeed, LockPosition, LockDirection, EventDistance, CalculatedData);

Operands

Ladder Diagram

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description		
Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_GENERIC_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis. For an Absolute or Incremental Master Offset move, enter the slave axis. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.		
Motion Control	MOTION_ INSTRUCTION	MOTION_ INSTRUCTION	Tag	Control tag for the instruction		
Move type	DINT	DINT	Immediate or Tag	**To**	**Use This Move Type**	**And Enter**
				Move an axis to an absolute position	Absolute	0
				Move an axis a specified distance from where it is now	Incremental	1
				Move a Rotary axis to an absolute position in the shortest direction regardless of its current position	Rotary Shortest Path	2
				Move a Rotary axis to an absolute position in the **positive** direction regardless of its current position	Rotary Positive	3
				Move a Rotary axis to an absolute position in the **negative** direction regardless of its current position	Rotary Negative	4
				Offset the master value of a position cam to an absolute position	Absolute Master Offset	5
				Offset the master value of a position cam by an incremental distance	Incremental Master Offset	6
				See Choose a Move Type for a Rotary Axis below for more information about rotary moves.		
Position	REAL	REAL	Immediate or Tag	Absolute position or incremental distance for the move		
				For This Move Type	**Enter This Position Value**	

				Absolute	Position to Move to
				Incremental	Distance to Move
				Rotary Shortest Path	Position to move to. Enter a positive value that is less than the Position Unwind value.
				Rotary Positive	
				Rotary Negative	
				Absolute Master Offset	Absolute Offset Position
				Incremental Master Offset	Incremental Offset Distance
Speed	REAL	REAL	Immediate or Tag	Speed to move the axis in Speed Units	
Speed Units	DINT	DINT	Immediate	Which units do you want to use for the Speed? Units per sec (0) % of Maximum (1) Time (3) Units per MasterUnit (4) Master Units (7)	
Accel Rate	REAL	REAL	Immediate or Tag	Acceleration rate of the axis in Accel Units	
Accel Units	DINT	DINT	Immediate	Which units do you want to use for the Accel Rate? Units per sec^2 (0) % of Maximum (1) Time (3) Units per $MasterUnit^2$ (4) Master Units (7)	
Decel Rate	REAL	REAL	Immediate or Tag	Deceleration rate of the axis in Deceleration Units.	
Decel Units	DINT	DINT	Immediate	Which units do you want to use for the Decel Rate? Units per sec^2 (0) % of Maximum (1) Time (3) Units per $MasterUnit^2$ (4) Master Units (7)	
Profile	DINT	DINT	Immediate	Select the velocity profile to run for the move: • Trapezoidal (0) • S-Curve (1)	
Accel Jerk	REAL	REAL	Immediate or Tag	The instruction only uses the jerk operands if the Profile is S-curve. You must always fill them in however. • Accel Jerk is the acceleration jerk rate for the axis. • Decel Jerk is the deceleration jerk rate for the axis. Use these values to get started. • Accel Jerk = 100 • Decel Jerk = 100 • Jerk Units = 2 (% of Time) You can also enter the jerk rates in these Jerk Units. • Units per sec^3 (0) • % of Maximum (1) • % of Time (2) • Time (3) • Units per $MasterUnit^3$ (4) • % of Time-Master Driven (6) • Master Units (7)	
Decel Jerk	REAL	REAL	Immediate or Tag		
Jerk Units	DINT	DINT	Immediate		

Merge	DINT	DINT	Immediate	Do you want to turn all current axis motion into a pure move governed by this instruction regardless of the motion instructions currently in process? • NO— Choose Disabled (0) • YES — Choose Enabled (1)
Merge Speed	DINT	DINT	Immediate	If Merge is Enabled, which speed do you want to move at? • Speed of this instruction — Select Programmed (0) • Current speed of the axis — Select Current (1)
Lock Position	REAL	REAL	Immediate or Tag	Position on the Master Axis where a Slave should start following the master after the move has been initiated on the Slave Axis. See the Structure section below for more information.
Lock Direction	SINT, INT, or, DINT	SINT, INT, or, DINT	Immediate	Specifies the conditions when the Lock Position should be used. Valid Values = 0-4 Default = None (Enumeration 1-4 are currently not allowed in Time Driven or Time Based modes.) See the Structure section below for more information.
Event Distance	REAL ARRAY or 0	REAL ARRAY or 0	Array Tag	The position(s) on a move measured from the end of the move. See the Structure section below for more information.
Calculated Data	REAL ARRAY or 0	REAL ARRAY 0	Array Tag	Master Distance(s) (or time) needed from the beginning of the move to the Event Distance point. See the Structure section below for more information.

Structured Text

This Operand	Has These Options Which You	
	Enter as Text	**Or Enter as a Number**
SpeedUnits	unitspersec %ofmaximum time unitspermasterunit masterunits	0 1 3 4 7
AccelUnits	unitspersec^2 %ofmaximum time unitspermasterunit^2 masterunits	0 1 3 4 7
DecelUnits	unitspersec^2 %ofmaximum time unitspermasterunit^2 masterunits	0 1 3 4 7
Profile	trapezoidal scurve	0 1

JerkUnits	unitspersec[3] %ofmaximum %oftime time unitspermasterunit[3] %oftime-masterdriven masterunits	0 1 2 3 4 6 7
Merge	disabled enabled	0 1
MergeSpeed	programmed current	0 1
Lock Position	No enumeration	Immediate, Real, or Tag
Lock Direction	None immediateforwardonly immediatereverseonly positionforward positionreverse	0 1 2 3 4
Event Distance	No enumeration	Array or 0
Calculated Data	No enumeration	Array or 0

See Structured Text Syntax for more information on the syntax of expressions within structured text.

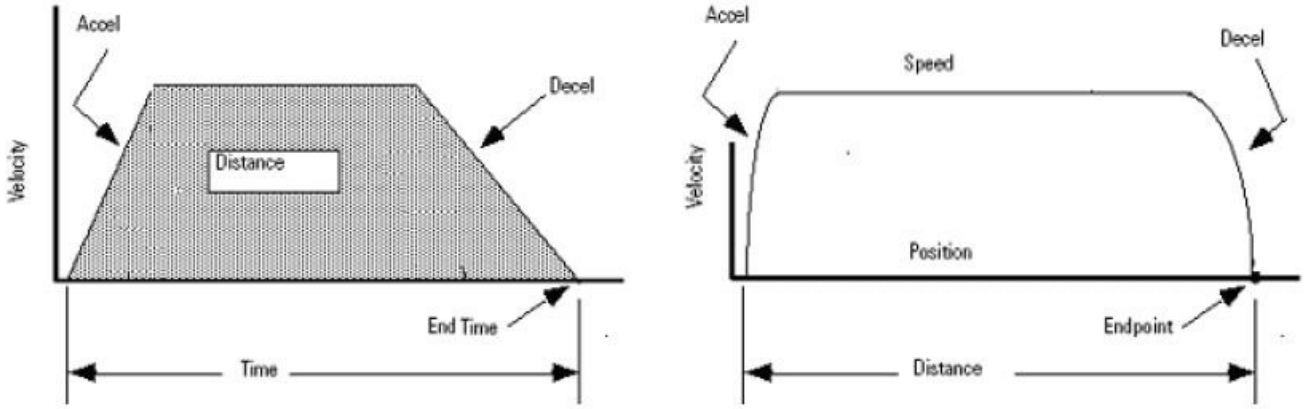
MOTION_INSTRUCTION Data Type

To See If	Check If This Bit Is Set To	Data Type	Notes
A false-to-true transition caused the instruction to execute	EN	BOOL	The EN bit stays set until the process is complete and the rung goes false.
The move was successfully initiated	DN	BOOL	
An error happened	ER	BOOL	
The axis is moving to the end Position	IP	BOOL	Any of these actions stop this move and clear the IP bit: • The axis gets to the end Position • Another MAM instruction supersedes this MAM instruction • MAS instruction • Merge from another instruction • Shutdown command • Fault Action
The axis is at the end Position	PC	BOOL	• The PC bit stays set until the rung makes a false-to-true transition. • The PC bit stays cleared if some other action stops the move before the axis gets to the end Position.

Description

The MAM instruction moves an axis to either a specified absolute position or by a specified incremental distance. The MAM instruction can also produce other special types of moves.

Trapezoidal Move Starting from Standstill



Programming Guidelines

Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is by changing move dynamics [Motion Change Dynamics (MCD)] instruction or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-curve velocity profile can overshoot if:

- maximum deceleration is decreased while the move is decelerating or close to the deceleration point; or
- maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

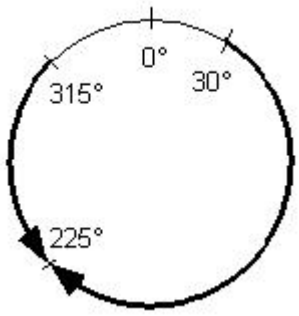
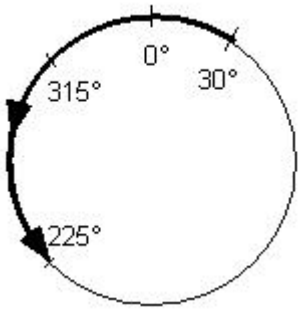
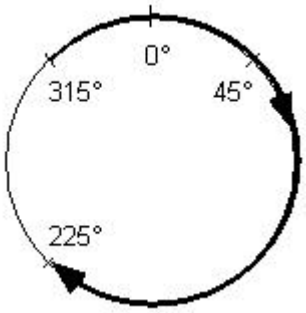
For more information, see Troubleshoot Axis Motion

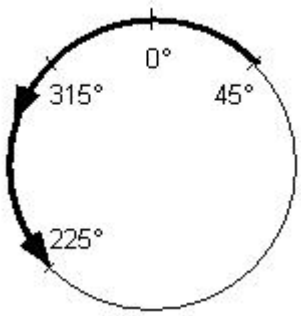
Guideline	Details
In ladder diagram, toggle the rung condition each time you want to execute the instruction.	This is a transitional instruction. In ladder diagram, toggle the rung-condition-in from cleared to set each time you want to execute the instruction.
In structured text, condition the instruction so that it only executes on a transition	In structured text, instructions execute each time they are scanned. Condition the instruction so that it only executes on a transition. Use either of these methods: • qualifier of an SFC action • structured text construct For more information, see Structured Text Syntax.
For a Master Offset move, enter the slave axis but use master units.	Use an Absolute or Incremental Master Offset move to off set the master value of a position cam without actually changing the position of the master axis. This shifts the position cam profile along the master axis. • For Axis, enter the slave axis. • For Position, enter the absolute offset position or incremental offset distance • For Speed, Acceleration, Deceleration, and Jerk, enter them for the master axis. The instruction adds in the offset at the Speed, Acceleration, Deceleration, and Jerk values.
Use % of Time for the easiest programming and tuning of jerk	For an easy way to program and tune jerk, enter it as a % of the acceleration or deceleration time. For more information, see Tune an S-Curve Profile.
Use Merge to cancel the motion of other	How you want to handle any motion that's already in process?

<table>
<tr><td rowspan="5">instructions</td><td>If you want to</td><td>And you want to</td><td>Then set</td></tr>
<tr><td>Add the move to any motion already in process</td><td>—▶</td><td>Merge = Disabled
Merge Speed = Programmed
The instruction ignores Merge Speed but you must fill it in anyway.</td></tr>
<tr><td rowspan="2">End the motion from other instructions and just jog</td><td>Move at the Speed that you set in this instruction</td><td>Merge = Enabled
Merge Speed = Programmed</td></tr>
<tr><td>Move at the speed that the axis is already moving</td><td>Merge = Enabled
Merge Speed = Current
The instruction ignores the value that you put in the Speed operand.</td></tr>
<tr><td colspan="3">Is This an Absolute or Incremental Master Offset Move?
If this is an Absolute or Incremental Master Offset move and Merge is Enabled, then the following is true.
• The move only ends an Absolute or Incremental Master Offset move that is already in process.
• The move does not affect any other motion that is already in process.</td></tr>
<tr><td>Use a second MAM instruction to change one that is already in process.</td><td colspan="3">You can change the position, speed, acceleration, or deceleration. The change immediately takes effect.
To Change the Position of An | Set Up a Second MAM Instruction Like This
Absolute Move | Either:
• Set the Move Type to Absolute and the Position to the new position.
• Set the Move Type to Incremental and set the Position to the distance to change the end position. The new end position is the old end position plus the new incremental distance.
In either case, the axis moves to the new position without stopping at the old position—including any required change of direction.
Incremental Move | Either:
• Set the Move Type to Absolute and the Position to the new position. The axis goes directly to the new position without completing the incremental move.
• Set the Move Type to Incremental and set the Position to the additional distance. The axis moves the total of both incremental moves.</td></tr>
</table>

Combine a move with gearing for complex profiles and synchronization.	You can use a Motion Axis Gear (MAG) instruction together with an MAM instruction. This superimposes the gearing on top of the move or the move on top of the gearing. **Example:** Superimpose an incremental move on top of electronic gearing for phase advance and retard control.

Choose a Move Type for a Rotary Axis

Move Type	Example	Description
Absolute	Absolute move to 225°. The direction depends on the starting position of the axis. 	With an Absolute move, the direction of travel depends on the current position of the axis and is not necessarily the shortest path to the end position. Starting positions less than the end position result in motion in the positive direction, while starting positions greater than the end position result in motion in the negative direction. The specified position is interpreted trigonometrically and can be positive or negative. It can also be greater than the Position Unwind value. Negative position values are equivalent to their corresponding positive values and are useful when rotating the axis through 0. For example, -90° is the same as +270°. When the position is greater than or equal to the Position Unwind value, the axis moves through more than one revolution before stopping at an absolute position.
Incremental		The specified distance is interpreted trigonometrically and can be positive or negative. It can also be greater than the Position Unwind value. When the distance is greater than the Position Unwind value, the axis moves through more than one revolution before stopping.
Rotary Shortest Path	Rotary Shortest Path move from 30° to 225°. 	**Important:** Only use a Rotary Shortest Path move if the Positioning Mode of the axis is Rotary (Rotary axis). A Rotary Shortest Path move is a special type of absolute move for a Rotary axes. The axis moves: • To the specified Position in the **shortest** direction regardless of its current position • Through 0° if needed. With a Rotary Shortest Path move, you: • Can start the move while the axis is moving or standing still • **Cannot** move the axis more than one revolution with a single move.
Rotary Positive	Rotary Positive move from 315° to 225°. 	**Important:** Only use a Rotary Positive move while the axis is standing still and not moving. Otherwise the axis could move in the wrong direction. A Rotary Positive move is a special type of absolute move for a Rotary axis. The axis: • Moves to the specified Position in the **positive** direction regardless of its current position • Moves through 0° if needed You can't move the axis more than one revolution with a single Rotary Shortest Path move.

Rotary Negative	Rotary Negative move from 45° to 225°.  	**Important:** Only use a Rotary Shortest Path move if • The Positioning Mode of the axis is Rotary (Rotary axis). • The axis is standing still and not moving. Otherwise the axis could move in the wrong direction. A Rotary Negative move is a special type of absolute move for a Rotary axis. The axis: • Moves to the specified Position in the **negative** direction regardless of its current position • Moves through 0° if needed You cannot move the axis more than one revolution with a single Rotary Shortest Path move.

When MAM (Merge = Enabled) is used on any axis associated with a coordinate system and a coordinated motion instruction is running on it, Coordinate system's maximum deceleration is used to stop remaining axes. If the coordinate system contains orientation axes, Coordinate system's Orientation maximum deceleration is used for stopping remaining Rx, Ry or Rz axes.

Structure

See Input and Output Parameters Structure for Single Axis Motion Instructions for the input and output parameters that are available for the MAM instruction via the Master Driven Speed Control (MDSC) function. Before any of these parameters is active, you must execute an MDAC instruction and it must be active (IP bit is set).

Affected Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis operand. See Common Attributes for operand-related faults.

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is true.
Rung-condition-out is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See Error Codes (.ERR) for Motion Instructions.

Runtime Error Condition

The slave move must start at rest if Speed Units = Seconds or Master Units. This condition may occur when the MAM with Speed = Seconds or Master Units is started while another MAM is in progress (merging or replacement mode).

Extended Error Codes

Use Extended Error Codes (EXERR) for more instruction about an error. See Error Codes (.ERR) for Motion Instructions.

<table>
<tr><th rowspan="2">If ERR is</th><th rowspan="2">And EXERR is</th><th colspan="3">Then</th></tr>
<tr><th>Cause</th><th colspan="2">Corrective Action</th></tr>
<tr><td rowspan="6">13</td><td rowspan="6">Varies</td><td rowspan="6">An operand is outside its range.</td><td colspan="2">The EXERR is the number of the operand that is out of range. The first operand is 0.
For example, if EXERR = 4, then check the Speed.</td></tr>
<tr><td>EXERR</td><td>Operand</td></tr>
<tr><td>0</td><td>Axis</td></tr>
<tr><td>1</td><td>Motion Control</td></tr>
<tr><td>2</td><td>Move Type</td></tr>
<tr><td>3</td><td>Position</td></tr>
<tr><td>4</td><td>Speed</td></tr>
<tr><td rowspan="2">15</td><td>-1</td><td>The coordinate system has a Maximum Deceleration of 0.</td><td colspan="2">Go to the Properties for the coordinate system axis and set a Maximum Deceleration.</td></tr>
<tr><td>0 or more</td><td>An axis in the coordinate system has a Maximum Deceleration of 0.</td><td colspan="2">1. Open the Properties for the axis.
2. Use the EXERR value to see which axis has the Maximum Deceleration of 0.
3. The axis that you are moving via the MAM instruction has a deceleration rate of 0.</td></tr>
</table>

Changes to Status Bits

Motion Instruction Predefined Data Type Status Bits

See Status Bits for Motion Instructions (MAM, MATC, MAJ) When MDAC Is Active.

MAM Changes to Single Axis Status Bits

Bit Name	Meaning	
MotionStatus	The motion status bit for your axis.	
	Bit Number	**Meaning**
AccelStatus	0	The axis is not accelerating (FALSE state).
DecelStatus	1	The axis is not decelerating (FALSE state).
MoveStatus	2	The axis is not moving (FALSE state).
JogStatus	3	The axis is not jogging (FALSE state).
GearingStatus	4	The axis is not gearing (FALSE state).
HomingStatus	5	The axis is not homing (FALSE state).
StoppingStatus	6	The axis is stopping (TRUE state).
AxisHomedStatus	7	The axis is not homed (FALSE state).
PositionCamStatus	8	The axis is not position camming (FALSE state).
TimeCamStatus	9	The axis is not time camming (FALSE state).
PositionCamPendingStatus	10	The axis does not have a Position Cam Pending (FALSE state).
TimeCamPendingStatus	11	The axis does not have a Time Cam Pending (FALSE state).
GearingLockStatus	12	The axis is not in a Gear Locked condition (FALSE state).
PositionCamLockStatus	13	The axis is not in a Position Cam Locked condition (FALSE state).
TimeCamLockStatus	14	The axis is not in a Time Cam Locked condition (FALSE state).
MasterOffsetMoveStatus	15	The axis is offset (TRUE state).
CoordinatedMotionStatus	16	Sets when the MDAC instruction executes (TRUE state). Clears when the instruction completes (FALSE state).
TransformStateStatus	17	The axis is part of an active transform (TRUE state).
ControlledByTransformStatus	18	The axis is moving because of a transform (TRUE state).
DirectVelocityControlStatus	19	The axis is not under Direct Velocity Control (FALSE state).
DirectTorqueControlStatus	20	The axis is not under Direct Torque Control (FALSE state).

MoveLockStatus	22	MAM is Locked to Master in MDSC Mode (TRUE state). The bit is cleared when a MGS, MGSD, MAS, or MASD is executed (goes IP). If either the Slave or Master axis (or both) is paused by changing its speed to 0, then the MoveLockStatus bit stays set. **Master Driven Mode** The bit is set when the Lock Direction request is satisfied. The bit is not used when the enumeration is NONE. For the enumerations Immediate Forward Only and Immediate Reverse Only, the MamLockStatus bit is set immediately when the MAM is initiated. For the enumeration Position Forward Only and Position Reverse Only, the bit is set when the Master Axis crosses the Master Lock Position in the specified direction. The MoveLockStatus bit is cleared when the Master Axis reverses direction and the Slave Axis stops following the Master Axis. The MoveLockStatus bit is set again when the Slave Axis resumes following the Master Axis. **Time Driven Mode** The bit is not used when the enumeration is NONE.
JogLockStatus	24	The axis is not in a Jog Locked condition (FALSE state).
MasterOffsetMoveLockStatus	26	Master offset Move is Locked to master in MDSC Mode (TRUE state).
MaximumSpeedExceeded	27	Sets when the maximum axis speed that is specified in the axis configuration is exceeded during a move (TRUE state). Clears when the speed is reduced below the limit (FALSE state).

Motion Status Bits

If the Move Type Is	And Merge is	Then the Instruction Changes These Bits		
		Bit Name	State	Meaning
NOT Absolute Master Offset or Incremental Master Offset	Disabled	MoveStatus	TRUE	Axis is Moving.
	Enabled	MoveStatus	TRUE	Axis is Moving.
		JogStatus	FALSE	Axis is no longer Jogging.
		GearingStatus	FALSE	Axis is no longer Gearing.
Absolute Master Offset or Incremental Master Offset	→	MasterOffsetMoveStatus	TRUE	Axis is Offset.

Merging in Incremental Mode

The Merge for MAM operates differently from a merge on a Motion Coordinated Linear Move (MCLM) instruction. For the MAM, any uncompleted motion at the point of the merge remains in the move. For example, assume that you have a single axis MAM programmed in incremental mode from a starting absolute position = 0 and with the programmed incremental distance = 4 units. If a merge occurs at an absolute position of 1 and the merge is another incremental move of 4 units, the move completes at a position = 8.

If this example occurs on a Motion Coordinated Linear Move (MCLM) instruction programmed in incremental mode, the final position = 5.

Master Driven Speed Control (MDSC) Merging and Replacement Mode for MAM

When programmed in units of seconds, the MAM instruction must start at rest (that is, both start velocity and acceleration must be equal to 0.0). If programmed in units of seconds or Master Units, a runtime error will occur for the MAM on the Slave if the instruction is activated when not at rest.

Master Driven Speed Control (MDSC) and Motion Direct Command Support

The Motion Direct commands are not available in the instruction tree for the MDAC instruction. You must program an MDAC in one of the supported programming languages before you execute an MAM in Time Driven Mode. A runtime error will occur if an MDAC is not previously executed in an MAM or MAJ in Master Driven Mode.

The Motion Direct Command supports the MDSC enumerations speed, acceleration, deceleration, and Jerk for MAM.

Note that Event Distance and Calculated Data are not supported parameters for the MAM and Motion Direct Command.

Master Driven Speed Control (MDSC) and CIP Axis Manual Tune and Motion Generator

Event Distance and Calculated Data parameters are not supported for MAM.

See also

Troubleshooting Axis Motion on page 639

Structured Text Syntax on page 665

Motion Error Codes (ERR) on page 581

Motion Move Instructions on page 73

Common Attributes on page 691

Motion Axis Gear (MAG)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The Motion Axis Gear (MAG) instruction provides electronic gearing between any two axes in a specified direction and at a specified ratio. When called, the

specified Slave Axis is geared to the Master Axis at the specified Ratio (for example, 1.345) or Slave Counts to Master Counts (for example, 1:3). The MAG instruction supports specification of the gear ratio in one of two different formats, Real or Fractional, as determined by the Ratio Format input selection. The direction of Slave Axis motion relative to the Master Axis is defined by a very flexible Direction input parameter. The gearing direction may be explicitly set as the Same or Opposite or set relative to the current gearing direction as Reverse or Unchanged.

Tip: The value for Ratio is sign sensitive. The Master Reference selection allows gearing input to be derived from either the Actual or Command position of the Master Axis. When the instruction's Clutch capability is activated the gearing instruction commands the slave axis to accelerate or decelerate at a controlled rate before Locking on to the master axis using the instructions Acceleration value much like the clutch of a car.

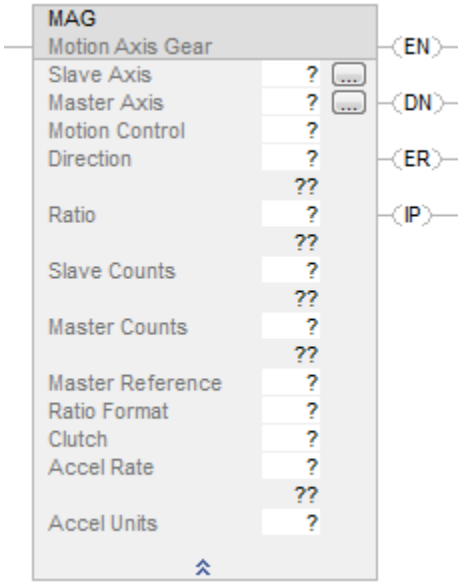
This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.
-

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MAG(SlaveAxis,MasterAxis,MotionControl,Direction,Ratio,SlaveCounts,MasterCounts,MasterReference,RatioFormat,Clutch,AccelRate,AccelUnits);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description

Slave Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_GENERIC_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis on which to perform the operation. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Master Axis	AXIS_CONSUMED AXIS_VIRTUAL AXIS_CIP_DRIVE **Tip:** AXIS_CONSUMED is supported by Compact GuardLogix 5580, CompactLogix 5380, and CompactLogix 5480 controllers only.	AXIS_CONSUMED AXIS_VIRTUAL AXIS_CIP_DRIVE	Tag	The axis that the slave axis follows. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Direction	SINT, INT, or DINT	SINT, INT, or DINT	Immediate	The relative direction that the Slave axis tracks the Master Axis. Select one of following: 0 = slave axis moves in the same direction as the master axis 1 = slave axis moves in the opposite direction of its current direction 2 = slave axis reverses from current or previous 3 = slave axis to continue its current or previous direction.
Ratio	REAL	REAL	Immediate or Tag	Signed Real value establishing the gear ratio in Slave User Units per Master User Unit.
Slave Counts	SINT, INT, or DINT	SINT, INT, or DINT	Immediate or Tag	Integer value representing slave counts used in specifying a Fractional gear ratio.
Master Counts	SINT, INT, or DINT	SINT, INT, or DINT	Immediate or Tag	Integer value representing master counts used in specifying a Fractional gear ratio.

Master Reference	BOOLEAN	BOOLEAN	Immediate	Sets the master position reference to either Command position or Actual position. 0 = Actual – slave axis motion is generated from the current position of the master axis as measured by its encoder or other feedback device. 1 = Command – slave axis motion is generated from the desired or commanded position of the master axis.
Ratio Format	BOOLEAN	BOOLEAN	Immediate	The desired ratio specification format. Select either: 0 = real gear ratio 1 = integer fraction of slave encoder counts to master encoder counts.
Clutch	BOOLEAN	BOOLEAN	Immediate	When Clutch is enabled, motion control ramps the slave axis up to gearing speed at the instruction's defined Acceleration value. If not enabled, the Slave axis immediately locks onto the Master axis. If the Master Axis is currently moving this condition results in an abrupt uncontrolled acceleration event of the Slave Axis which can cause the axis to fault. Select either: 0 = enabled 1 = disabled
Accel Rate	REAL	REAL	Immediate or Tag	Acceleration rate of the Slave Axis in % or Acceleration Units. It is applied when the Clutch feature is enabled.
Accel Units	DINT	DINT	Immediate	The units used to display the Acceleration value. Select either: 0 = units per sec^2 1 = % of maximum acceleration

See Structured Text Syntax for more information on the syntax of expressions within structured text.

For the operands that require you to select from available options, enter your selection as described below.

This Operand	Has These Options Which You	
	Enter as Text	Or Enter as a Number
MasterReference	actual command	0 1

RatioFormat	real fraction_slave_master_counts	0 1
Clutch	enabled disabled	0 1
AccelUnits	unitspersec2 %ofmaximum	0 1

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when axis gear has been successfully initiated.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.IP (In Process) Bit 26	It is set on positive rung transition and cleared if either superseded by another Motion Gear Axes command, or terminated by a stop command, merge, shutdown, or servo fault.

Description

The MAG instruction enables electronic gearing between two axes at a specified ratio. Electronic gearing allows any physical axis to be synchronized to the actual or command position of another physical axis at a precise ratio. It provides a direct edge-to-edge lock between the two axes—no maximum velocity, acceleration, or deceleration limits are used. The speed, acceleration, and deceleration of the slave axis is completely determined by the motion of the master axis and the specified gear ratio.

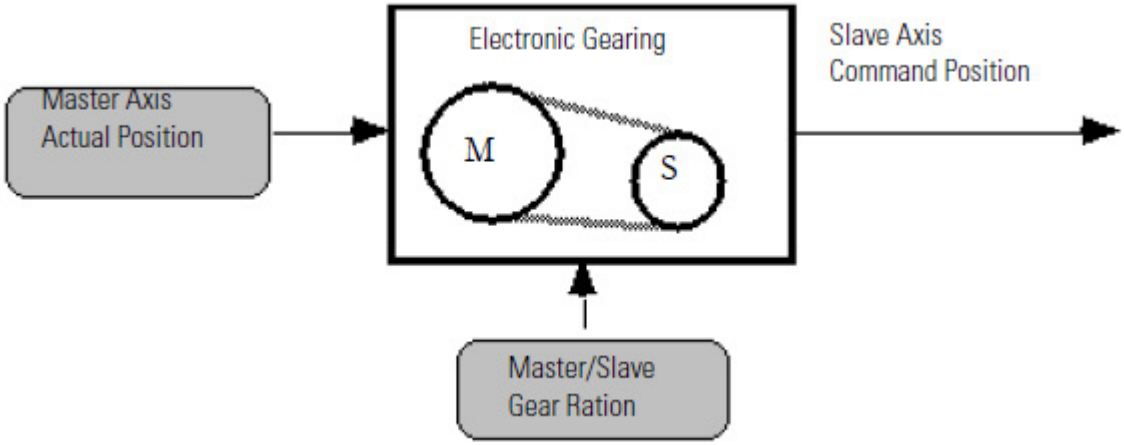
IMPORTANT The maximum velocity, acceleration, or deceleration limits established during axis configuration do not apply to electronic gearing.

Select or enter the desired Master Axis, Slave Axis, and Direction and enter a value or tag variable for the desired ratio. If an axis is dimmed (gray) or not shown in the Slave Axis pop-up menu, the physical axis is not defined for Servo operation.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for servo operation. Use the Tag Editor to create and configure a new axis.

Electronic gearing remains active through any subsequent execution of jog, or move processes for the slave axis. This allows electronic gearing motions to be superimposed with jog, or move profiles to create complex motion and synchronization.

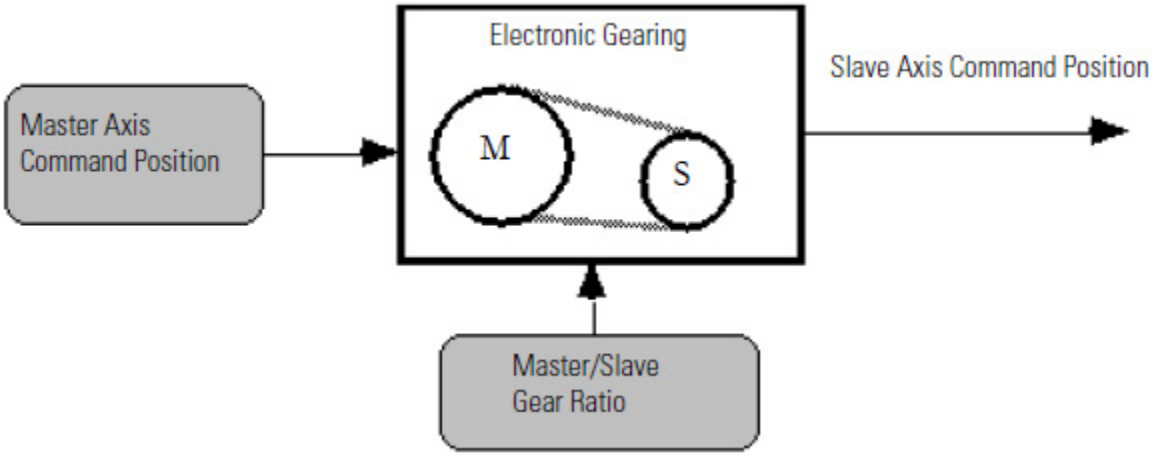
Slaving to the Actual Position When Actual Position is entered or selected as the Master Reference source, the slave axis motion is generated from the actual position of the master axis as shown below.



Actual position is the current position of a physical axis as measured by the axis encoder. This is the only valid selection when the master axis' Axis Type is configured as Feedback Only.

Slave to the Command Position

When Command Position is entered or selected as the Master Reference source, the slave axis motion is generated from the command position of the master axis as shown below.



Command position (only valid when the master axis' Axis Type is configured as Servo) is the current desired or commanded position for the master axis.

Since the command position does not incorporate any associated following error, external position disturbances, or quantization noise, it is a more accurate and stable reference for gearing. When gearing to the command position of the master, the master axis must be commanded to move to cause any motion on the slave axis.

Gear in the Same Direction

When Same is selected or entered as the Direction, the slave axis moves in its positive direction at the specified gear ratio when the master axis moves in its positive direction and vice-versa.

Gear in the Opposite Direction

When Opposite is selected or entered as the Direction, the slave axis moves in its negative direction at the specified gear ratio when the master axis moves in its positive direction and vice-versa.

Change the Gear Ratio

When Unchanged is selected or entered as the Direction, the gear ratio may be changed while preserving the current gearing direction (same or opposite). This is useful when the current direction is not known or not important.

Reverse the Gearing Direction

When Reverse is selected or entered as the Direction, the current direction of the electronic gearing is changed from same to opposite or from opposite to same. This is very useful for winding applications where the gear ratio must be reversed at each end of the wind.

Real Number Gear Ratio

When Ratio Format is selected or entered as Real, the gear ratio is specified as a real number or tag variable with a value between 0.00001 and 9.99999 (inclusive) representing the desired ratio of slave axis position units to master axis position units. A gear ratio expressed this way is easy to interpret since it is defined in the axes' configured position units.

Fraction Gear Ratios

When Ratio Format is selected or entered as Fraction, the gear ratio is specified as a pair of integer numbers or tag variables representing the ratio between the number of slave axis feedback counts and the number of master axis feedback counts. See The Tag variable Builder earlier in this manual for information on tag variables.

IMPORTANT The Conversion Constant entered as part of the axis configuration procedure is not used when the Ratio Format for the MAG instruction is specified as a Fraction.

If your gear ratio cannot be exactly expressed as a real number with a maximum of five digits to the right of the decimal point, use Fraction as the Ratio Format.

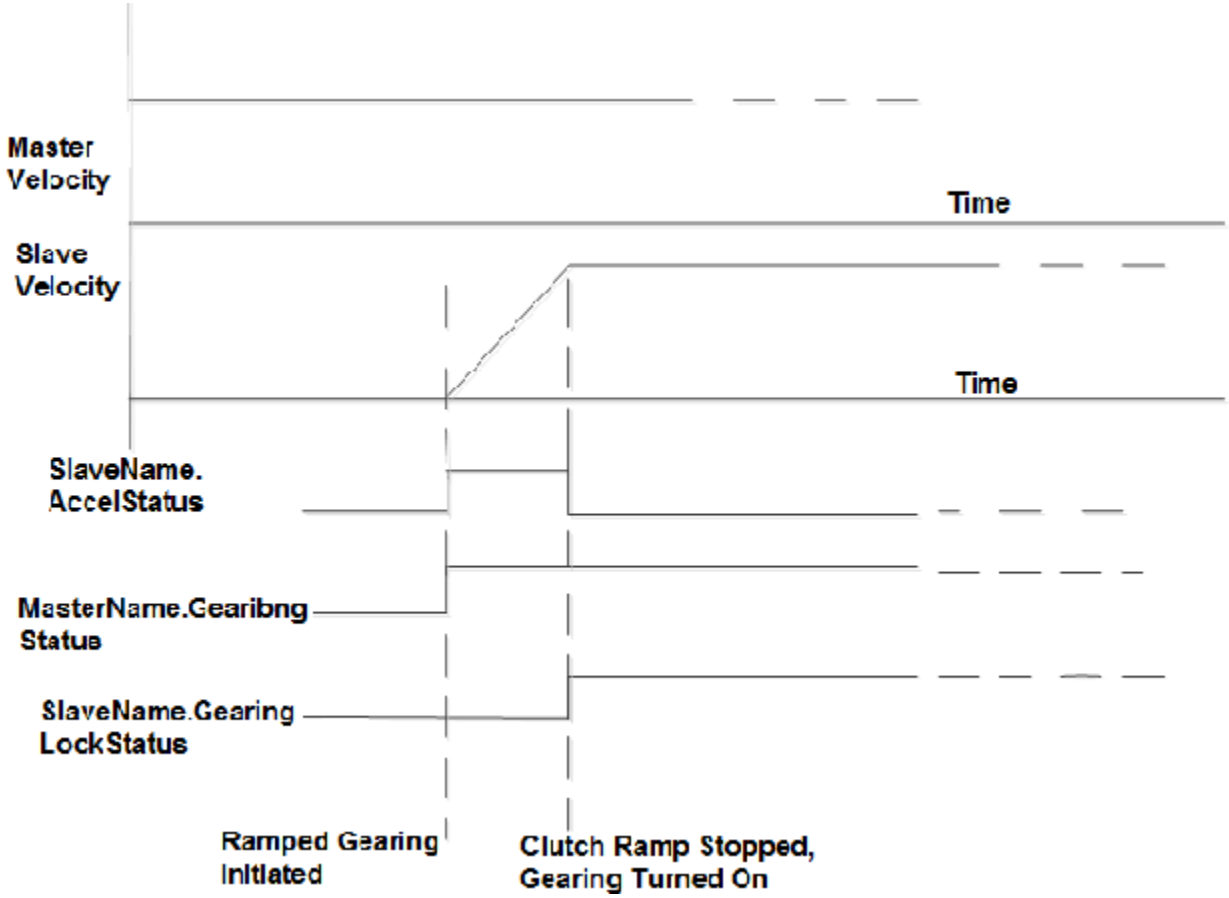
Specifying the gear ratio as a fraction allows the direct implementation of irrational gear ratios (such as 1/3) with no accumulated positioning errors or round off. Since the master and slave count values do not use the axis conversion constants and because they are integers, the actual gear ratio relationship between the slave and master axes exactly match the specified ratio.

For example, the irrational gear ratio of 1/3 can be equivalently specified as 1 slave count to 3 master counts, 10 slave counts to 30 master counts, 3 slave counts to 9 master counts.

Clutch

When the Clutch check box is checked, the slave axis accelerates or decelerates to the speed that it would be moving if it were currently geared to the selected master axis at the specified gear ratio and direction using a trapezoidal velocity profile (linear acceleration or deceleration). Once the slave axis has reached the gearing speed, electronic gearing is automatically activated according to the other selections. Enter the desired Accel Rate as a percentage of the current configured maximum acceleration value or directly in the configured user units for acceleration.

This clutch function works much like the clutch in a car, allowing the slave axis to be smoothly engaged to the master axis as shown below.



Clutch Function

Using the clutch feature avoids the uncontrolled acceleration or deceleration that results when electronic gearing is enabled while the master axis is moving. The clutch feature can also be used to merge gear ratio changes on-the-fly, even changes in direction. The motion controller automatically ramps the slave axis to the speed implied by the master axis at the new ratio and/or direction.

The operation of the clutch ramp generator has no affect on jog or move processes that might be in progress on the slave axis.

Changing Master Axes

The master axis for electronic gearing can be changed at any time, even while gearing is currently enabled. However, since it is possible to have electronic gearing enabled on more than one axis at a time, if a Servo master axis and slave axis are reversed, the axes become cross-coupled and unexpected motion may result.

For example, if you are gearing Axis 0 to Axis 1 (defined as a Servo axis) and then want to change to gearing Axis 1 to Axis 0, you must first disable gearing on Axis 0 (see Disable Gearing later in this section). This is because specifying Axis 1 as the slave axis with Axis 0 as the master axis does not automatically disable Axis 0 from being a slave axis with Axis 1 as the master axis.

Move While Gearing

An incremental MAM instruction may be used for the slave axis (or master axis if the Axis Type is configured as Servo) while the electronic gearing is enabled. This is particularly useful to accomplish phase advance/retard control. The incremental move distance can be used to eliminate any phase error between the master and the slave, or to create an exact non-zero phase relationship. Incremental MAM instruction may also be used in conjunction with electronic gearing to compensate for material slip.

Normally a gear ratio of 1 is used with phase adjustment. A 1:1 ratio ensures that the computed phase error does not change before performing the move to correct it. Electronic gearing is not normally used with absolute moves, since the ultimate endpoint is not predictable.

To successfully execute a MAG instruction, the targeted axis must be configured as a Servo Axis Type and the axis must be in the Servo On state. If any of these conditions are not met than the instruction errors.

IMPORTANT The MAG instruction execution completes in a single scan, thus the Done (.DN) bit and the In Process (.IP) bit are set immediately. The In Process (.IP) bit remains set until the initiated Gear process is superseded by another MAG instruction, or terminated by a Motion Axis Stop command, Merge operation, or Servo Fault Action.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis operand. See Common Attributes for operand-related faults

Execution

Ladder Diagram

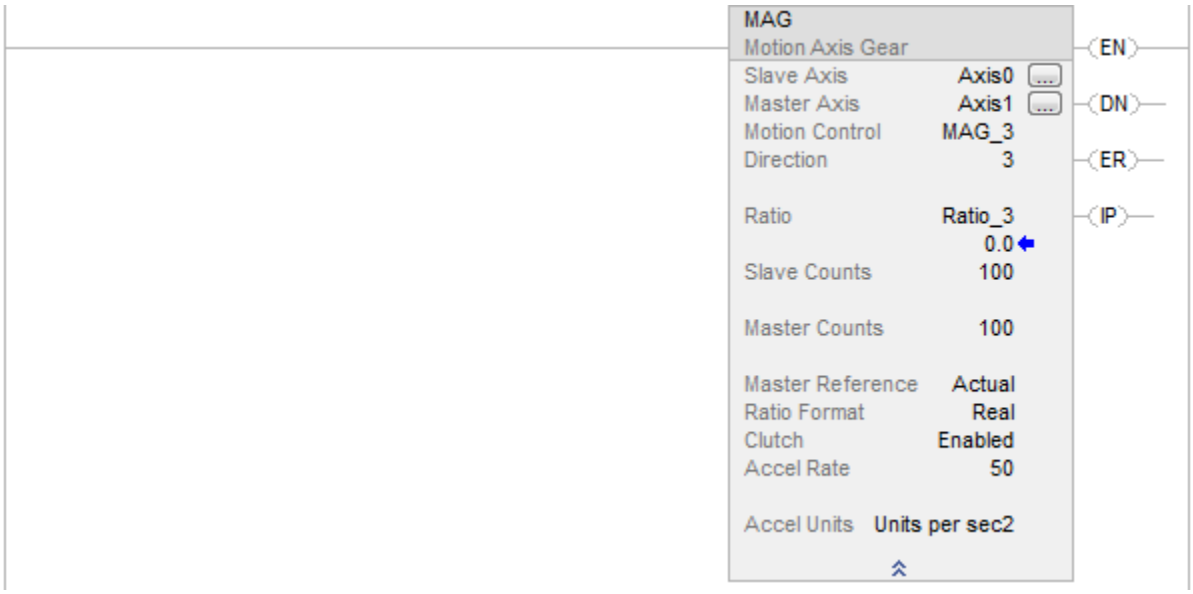
Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in Ladder Diagram table
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in Ladder Diagram table.

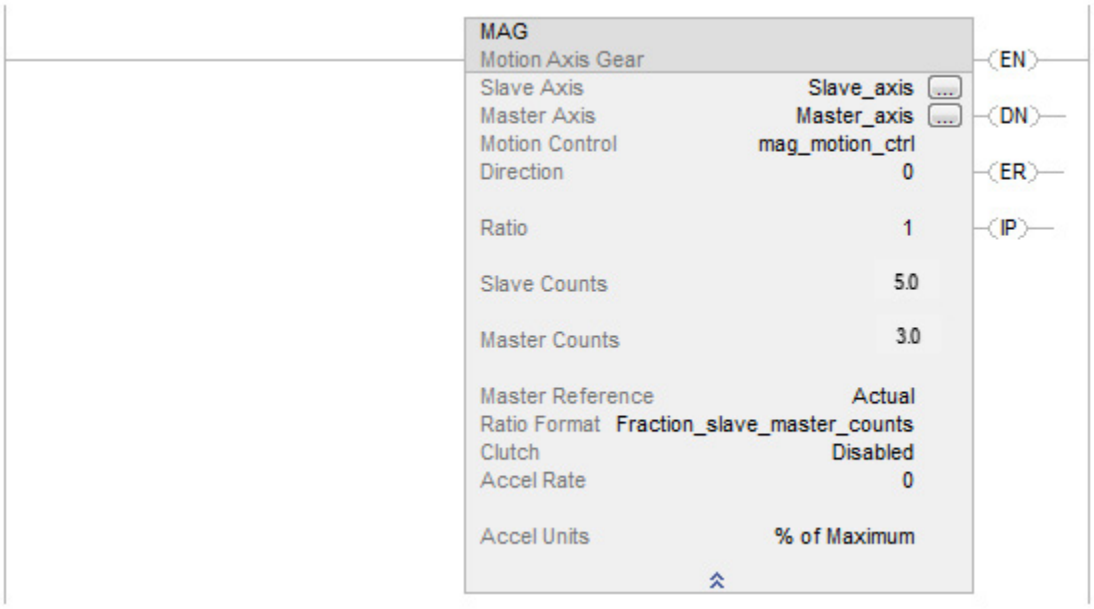
Examples

Example 1

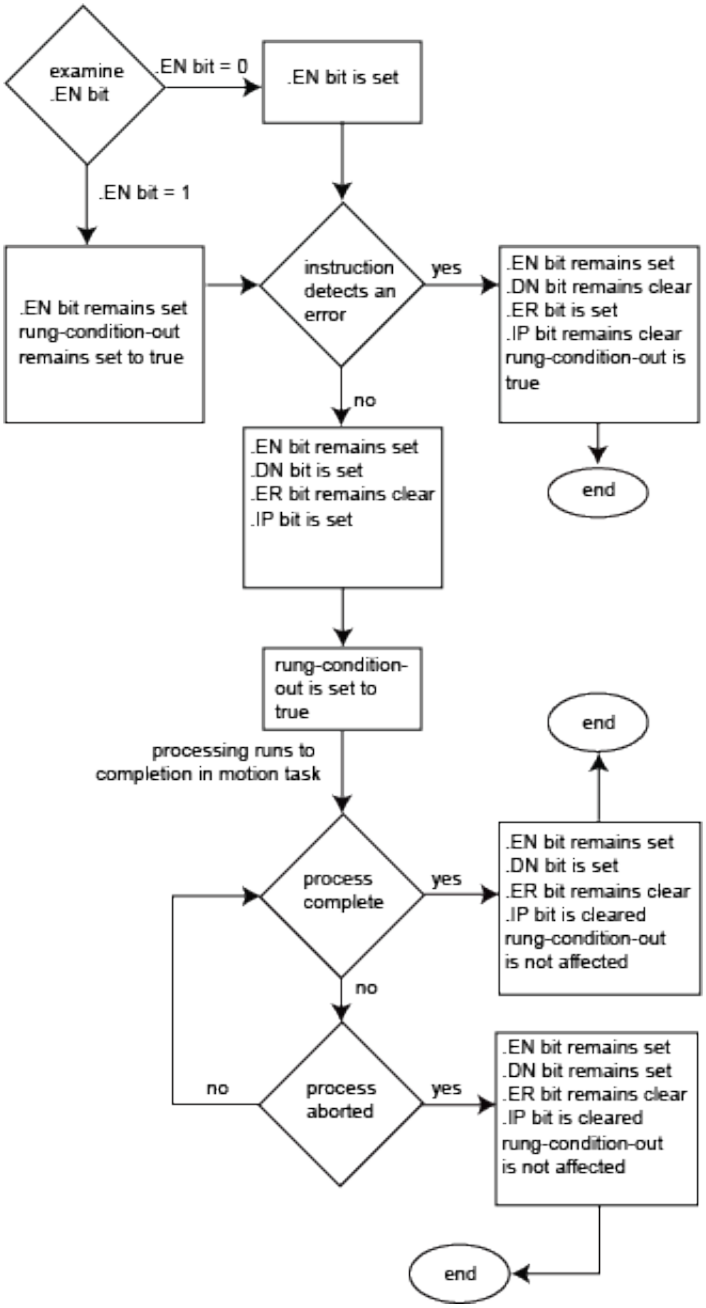


Example 2

This example increases the number of operands that require data conversion.



MAG Flow Chart (True)



Motion Change Dynamics (MCD)

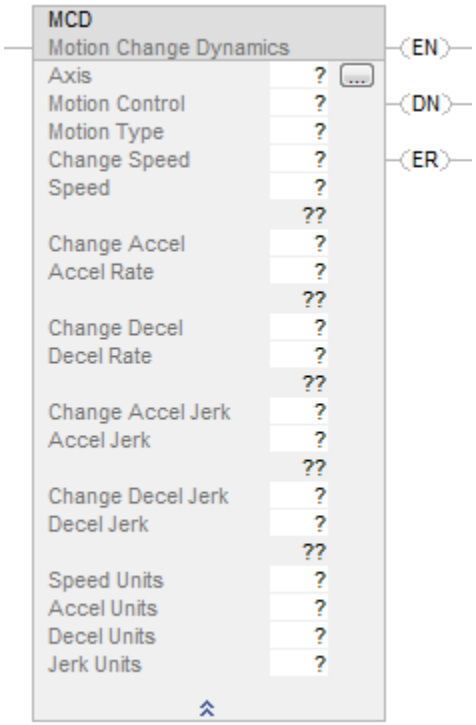
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Use the Motion Change Dynamics (MCD) instruction to selectively change the speed, acceleration rate, or deceleration rate of a move profile or a jog profile in process.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MCD(Axis, MotionControl, MotionType, ChangeSpeed, Speed, ChangeAccel, AccelRate, ChangeAccelJerk, AccelJerk, ChangeDecelJerk, ChangeDecel, DecelRate, SpeedUnits, AccelUnits, DecelUnits, JerkUnits);

Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion.

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_SERVO AXIS_SERVO_DRIVE AXIS_GENERIC_DRIVE	Tag	Name of the axis to perform operation on. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Motion Type	UDINT	UDINT	Immediate	Motion profile (jog or move) to change. Select either: 0 = jog 1 = move
Change Speed	BOOLEAN	BOOLEAN	Immediate	Set to enable a change of speed. Select either: 0 = no 1 = yes
Speed	REAL	REAL	Immediate or Tag	The new Speed to move the axis in % or Speed Units.
Change Accel	BOOLEAN	BOOLEAN	Immediate	Set to enable an acceleration change. Select either: 0 = no 1 = yes
Accel Rate	REAL	REAL	Immediate or Tag	The acceleration rate of the axis in % or Acceleration units.
Change Decel	BOOLEAN	BOOLEAN	Immediate	Set to enable a deceleration change. Select either: 0 = no 1 = yes
Decel Rate	REAL	REAL	Immediate or Tag	The deceleration rate of the axis in % or Deceleration units. The axis could overshoot its target position if you reduce the deceleration while a move is in process.

Change Accel Jerk	SINT, INT, or DINT	SINT, INT, or DINT	Immediate	0 = No 1 = Yes
Accel Jerk	SINT, INT, DINT, or REAL	SINT, INT, DINT, or REAL	Immediate or Tag	You must always enter a value for the Accel Jerk operand. This instruction only uses the value if the Profile is configured as S-curve. Accel Jerk is the acceleration jerk rate for the axis. Use this value to get started. Accel Jerk = 100 (% of Time) Jerk Units = 2
Change Decel Jerk	SINT, INT, or DINT	SINT, INT, or DINT	Immediate	0 = No 1 = Yes
Decel Jerk	SINT, INT, DINT, or REAL	SINT, INT, DINT, or REAL	Immediate or Tag	You must always enter a value for the Decel Jerk operand. This instruction only uses the value if the Profile is configured as S-curve. Decel Jerk is the deceleration jerk rate for the coordinate system. Use these values to get started. Decel Jerk = 100 (% of Time) Jerk Units = 2
Speed Units	DINT	DINT	Immediate	Units used to display the Speed value. Select either: 0 = units per sec 1 = % of maximum speed 4 = units per master unit
Accel Units	DINT	DINT	Immediate	Units used to display the Acceleration value. Select either: 0 = units per sec^2 1 = % of maximum acceleration 4 = units per master unit^2

Decel Units	BOOLEAN	BOOLEAN	Immediate	Units used to display the Deceleration value. Select either: 0 = units per sec^2 1 = % of maximum deceleration 4 = units per master unit^2
Jerk Units	SINT, INT, or DINT	SINT, INT, or DINT	Immediate	0 = Units per sec^3 1 = % of Maximum 2 = % of Time 4 = units per master unit^3 6 = % of Time Master Driven

Structured Text

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

Enter your selection for the operands that require you to select your available options.

This Operand	**Has These Options Which You**	
	Enter as Text	**Or Enter as a Number**
Axis	No enumeration	Tag
MotionControl	No enumeration	Tag
MotionType	jog move	0 1
ChangeSpeed	no yes	0 1
Speed	No enumeration	Immediate or Tag
ChangeAccel	no yes	0 1
AccelRate	No enumeration	Immediate or Tag
ChageDecel	no yes	0 1
DecelRate	No enumeration	Immediate or Tag
ChangeAccelJerk	No enumeration	0 = No 1 = Yes
AccelJerk	No enumeration	Immediate or tag You must always enter a value for the Accel operand. This instruction only uses the value if the Profile is configured as S-curve. Use this value to get started. Accel Jerk = 100
ChangeDecelJerk	No enumeration	0 = No 1 = Yes

DecelJerk	No enumeration	Immediate or tag You must always enter a value Decel Jerk operand. This instruction only uses the value if the Profile is configured as S-curve. Use this value to get started. Decel Jerk = 100
SpeedUnits	unitspersec %ofmaximum unitspermasterunit	0 1 4
AccelUnits	unitspersec^2 %ofmaximum unitspermasterunit^2	0 1 4
DecelUnits	unitspersec^2 %ofmaximum unitspermasterunit^2	0 1 4
JerkUnits	unitspersec^3 %ofmaximum %oftime unitspermasterunit^3 %oftimemasterdriven	0 1 2 4 6

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when axis change dynamics has been successfully initiated. The instruction execution completes in a single scan, and the DN bit is set immediately.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MCD instruction changes the speed of profile moves on-the-fly and the speed, acceleration, and deceleration of trapezoidal profile jogs on-the-fly. Choose the desired physical axis and type of motion and enter values or tag variables for the Speed, Accel, and Decel. Speed, acceleration, and deceleration values can be entered as percentages of the current maximum configured value or directly in the configured speed or acceleration units of the axis.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for servo operation. Use the Tag Editor to create and configure a new axis.

Programming Guidelines

Important:	Risk of Velocity and/or End Position Overshoot If you change move parameters dynamically by any method, that is by changing move dynamics [Motion Change Dynamics (MCD) instruction or Motion Coordinated Change Dynamics (MCCD)] or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot. A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point. An S-curve velocity profile can overshoot if: maximum deceleration is decreased while the move is decelerating or close to the deceleration point; or maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time. For more information, see Troubleshoot Axis Motion.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Changing Move Dynamics

When a Motion type of Move is entered or chosen, the speed, acceleration, and/or deceleration of a Move in progress may be changed to the specified value. The speed change occurs at the specified acceleration rate if the new speed is higher than the current speed or at the specified deceleration rate if the new speed is lower than the current speed.

Pausing Moves

The MCD instruction may be used to temporarily pause a move in progress by changing its speed to zero. Use another MCD instruction with a non-zero speed value to complete the move as originally specified.

Changing Jog Dynamics

When a Motion type of Jog is entered or chosen, the speed, acceleration, and/or deceleration of a Jog in progress may be changed to the specified value. The speed change occurs at the specified acceleration rate if the new speed is higher than the current speed or at the specified deceleration rate if the new speed is lower than the current speed.

Supporting Motion Drive Start (MDS) Instruction

The MCD instruction supports the Motion Drive Start (MDS) instruction. However, the MCD instruction has no affect on the DirectVelocityControlStatus Command feature because the Motion Planner only takes the value from the Direct Command Velocity Attribute and sums it

into the axis output before sending the total command to the drive. After all acceleration and deceleration have been planned, the MCD instruction has no affect on the feature.

Changing Between MDSC and Time Driven Modes in Master Driven Speed Control (MDSC)

You cannot change from Master Driven Mode to Time Driven Mode or vice versa with an MCD instruction. You receive a runtime error if you attempt to change modes.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults. A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis operand.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table
Normal execution	See Rung-condition-in is false, followed by rung-condition-in is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See *Motion Error Codes (.ERR)* for Motion Instructions.

Runtime Error Condition

For the Master Driven Speed Control (MDSC) function, an error will occur at runtime if you attempt to change the mode of the system from Master Driven to Time Driven or from Time Driven to Master Driven.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. See *Motion Error Codes (.ERR)* for Motion Instructions.

Extended Error codes for the Parameter Out of Range (13) error code lists a number that refers to the number of the operand as they are listed in the faceplate from top to bottom with the first operand being counted as zero. Therefore for the MCD instruction, an extended error code of 4 would refer to the Speed operand's value. You would then have to check your value with the accepted range of values for the instruction.

For the Error Code 54 – Maximum Deceleration Value is Zero, if the Extended Error returns a positive number (0-n) it is referring to the offending axis in the coordinate system. Go to the Coordinate System Properties General Tab and look under the Brackets ([ ])column of the Axis Grid to determine which axis has a Maximum Deceleration value of 0. Click on the ellipsis button next to the offending axis to access the Axis Properties screen. Go to the Dynamics tab and make the appropriate change to the Maximum Deceleration Value. If the Extended Error number is -1, this means the Coordinate System has a Maximum Deceleration Value of 0. Go to the Coordinate System Properties Dynamics Tab to correct the Maximum Deceleration value.

MCD Changes to Status Bits

For the Master Driven Speed Control (MDSC) function, when the MCD is executed (goes IP), the status bit (CalculatedDataAvailable (CDA)) is cleared in an MAM instruction, which indicates that the Event Distances has been computed. After the MCD is complete and the Event Distances have been recomputed, the CalculatedDataAvailable status bit is set again. Therefore, look at the CalculatedDataAvailable status bit after the MCD instruction has been completed to determine when to use the recomputed Event Distances.

If a MCD is executed, the CDA bit is cleared. The Calculated Data for the move is recomputed using the new dynamics parameters. The CDA bit is set again

when computations are complete. The Calculated Data that is recomputed is measured from the original Motion Start Point (MSP) to the Event Distance point using the new dynamics parameters as changed by the MCD instruction - not from the point of the MCD.

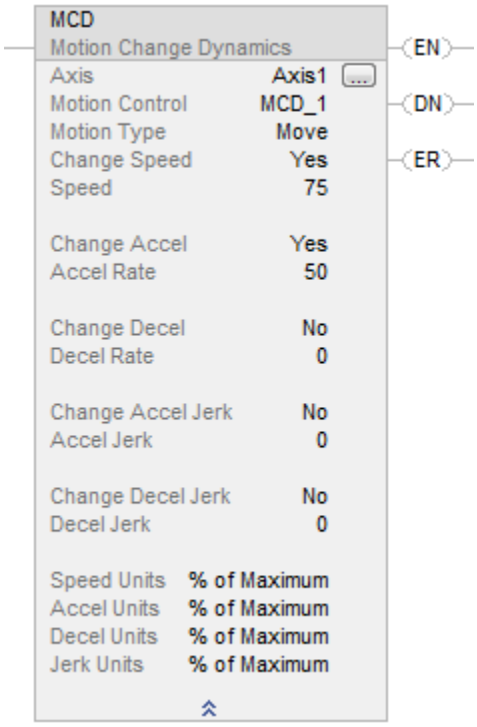
Note that if the MCD changes the speed to 0, the Event Distance is not recomputed; the CDA bit is not set. The Event Distance is however recomputed if a second MCD is issued to restart the motion. The recomputed Calculated Data includes the duration of the stopped motion.

If the Event Distance is set to 0, the Calculated Data is set to equal the position that equals the length of the move. This may be one or two coarse update periods before the PC bit is set because of an internal delay. The end position is typically achieved in the middle of a coarse update period, which adds up to one additional coarse update period to the delay. Therefore, if the master is moved a distance equal to the Calculated Data, you must wait up to 2 iterations more for the PC bit of the slave move to be set.

Examples

Example 1

Ladder Diagram

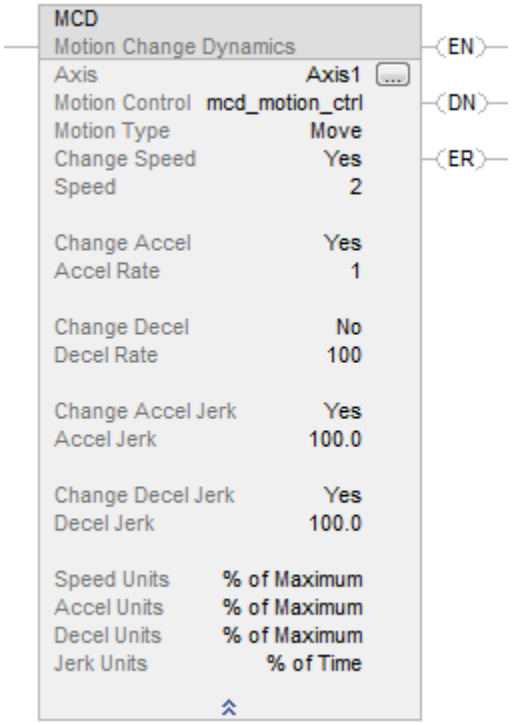


Structured Text

MCD(Axis1,MCD_1,Move,Yes,75,Yes,50,No,0,No,0,No,0,%ofMaximum,%ofMaximum,%ofMaximum,%ofMaximum);

Example 2

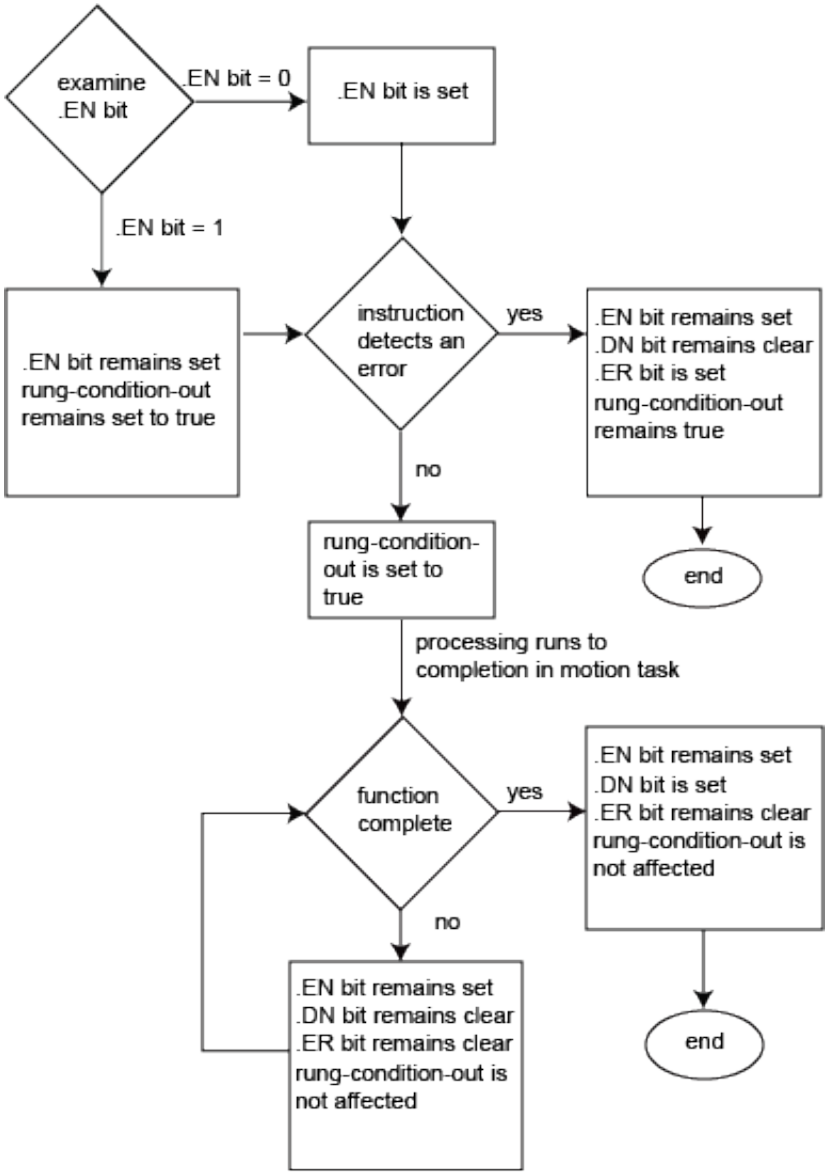
Ladder Diagram



Structured Text

MCD(Axis1,mcd_motion_ctrl,Move,Yes,2,Yes,1,No,100,Yes,100.0,Yes,100.0,%ofMaximum,%ofMaximum,%ofMaximum,%ofTime);

MCD Flow Chart (True)



Motion Redefine Position (MRP)

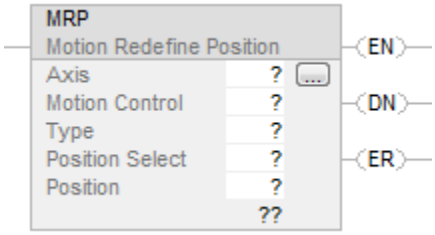
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Use the Motion Redefine Position (MRP) instruction to change the command or actual position of an axis. The value specified by Position is used to update the Actual or Command position of Axis. The position redefinition can be calculated on an Absolute or Relative basis. If Absolute is selected the Position value is assigned to the current Actual or Command position. If Relative is selected the Position value is added as a displacement to the current Actual or Command position. The process of redefining the current axis position has no affect on motion in progress as the instruction preserves the current servo following error during the redefinition process. As a result, axis position can be redefined on-the-fly without disturbing axis motion.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MRP(Axis,MotionControl,Type,PositionSelect,Position);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description

Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis on which to perform operation. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION		Tag	Structure used to access instruction parameters.
Type	BOOLEAN		Immediate	The way you want the redefinition operation to work. Select either: 0 = absolute 1 = relative
Position Select	BOOLEAN		Immediate	Choose what position to perform the redefinition operation on. Select either: 0 = actual position 1 = command position
Position	REAL		Immediate or Tag	The value to use to change the axis position to or offset to current position.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

For the operands that require you to select from available options, enter your selection as:

This Operand	**Has These Options Which You**	
	Enter as Text	**Or Enter as a Number**
Type	Absolute Relative	0 1
PositionSelect	Actual Command	0 1

MOTION_INSTRUCTION Structure

Mnemonic	**Description**
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.

.DN (Done) Bit 29	It is set when the axis' position action has been successfully redefined.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

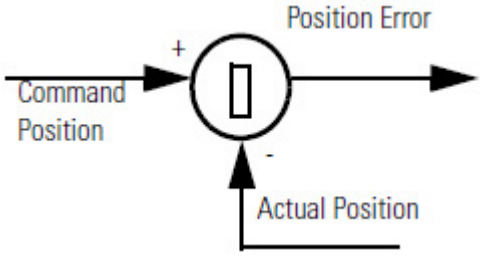
Description

The MRP instruction directly sets the actual or command position of the specified axis to the specified absolute or relative position. No motion is caused by this instruction—the current axis position is simply redefined. Select or enter the desired Axis, Type, Position Selection, and enter a value or tag variable for the desired New Position.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

The MRP instruction may be used while the axis is moving as well as when it is at rest. MRP is used to redefine position on-the-fly for certain registration, slip compensation, and re-calibration applications.

Selection	Description
Absolute Mode	When Absolute is selected or entered as the MRP Type, the New Position specifies the new absolute position of the axis. No motion occurs—the current axis position (actual or command) is simply redefined to be the specified new position. If software overtravel limits are used (refer to Motion Axis Object specification for more information on software overtravel configuration), the new position must be between the Max Positive and Max Negative Travel configuration values. Otherwise a software overtravel fault is generated when the instruction is executed. **Important:** If software overtravel limit checking is in effect, execution of an MRP in Absolute Mode may invalidate the current Max Positive and Max Negative Travel limits in the absolute sense. Exercise caution when redefining the absolute position of an axis that has travel limits. Absolute and relative mode MRP instructions have the same effect when the axis is not moving. When the axis is moving, however, absolute mode introduces a position error equal to the motion of the axis during the time it takes to execute the MRP instruction and assign the new position. Relative mode does not introduce this error and guarantees an exact correction independent of axis speed or position.
Relative Mode	When Relative is selected or entered as the MRP Type, the New Position value is used to offset the current position of the axis. No motion occurs—the current axis position (actual or command) is simply redefined to be the current position plus the specified new position. In relative mode, axis position is redefined in such a way that no position errors are introduced if the axis is moving. It is particularly useful for unwinding axis position under program control rather than using the built-in rotary axis feature. Absolute and relative mode MRP instructions have the same effect when the axis is not moving. When the axis is moving, however, absolute mode introduces a position error equal to the motion of the axis during the time it takes to execute the MRP instruction and assign the new position. Relative mode does not introduce this error and guarantees an exact correction independent of axis speed or position.
Actual Position	When Actual is selected or entered as the MRP Position Selection, the New Position is directly applied to the actual position of the physical axis. The command position of the axis is also adjusted along with the new actual position to preserve any position error which exists. This ensures that there is no unexpected motion of the axis when the positions are redefined. See the Motion Axis Object Specification for more discussion of command position, actual position, and position error.

Command Position	When Command is selected or entered as the MRP Position Selection, the New Position is directly applied to the command position of the servo or imaginary axis. For an axis with a Position Loop type of Feedback Only, the Command Position and the Actual Position are the same. The MRP can be used with ether the Command or Actual Position with the same effect. The actual position of servo axes is also adjusted along with the new command position to preserve any position error which exists. This ensures that there is no unexpected motion of the axis when the positions are redefined. Command position is the desired or commanded position of a servo as generated by any previous motion instructions. Actual position is the current position of a physical or virtual axis as measured by the encoder or other feedback device. Position error is the difference between these two and is used to drive the motor to make the actual position equal to the command position. The Figure below shows the relationship of these three positions.  To successfully execute a MRP instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. Otherwise, the instruction errs. **Important:** The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Master Driven Speed Control (MDSC) and the MRP Instruction

You can execute an MRP on the Master or the Slave axes or coordinate system when an MDSC is active.

The Master axis position is changed when an MRP is executed (goes IP) on the Master while it is moving in MDSC mode; the slave is not affected.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis operand.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See Error Codes (ERR) for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. See *Motion Error Codes (.ERR)* for Motion Instructions. The following Extended Error codes help to pinpoint the problem when the MRP instruction receives a Servo Message Failure (12) error message.

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	Device in wrong state (16)	Redefine Position, Home, and Registration 2 are mutually exclusive.

Extended Error codes for the Parameter Out of Range (13) error code work a little differently. Rather than having a standard enumeration, the number that appears for the Extended Error code refers to the number of the operand as they are listed in the faceplate from top to bottom with the first operand being counted as zero. Therefore for the MRP instruction, an extended error code of 4 would refer to the Position operand's value. You would then have to check your value with the accepted range of values for the instruction.

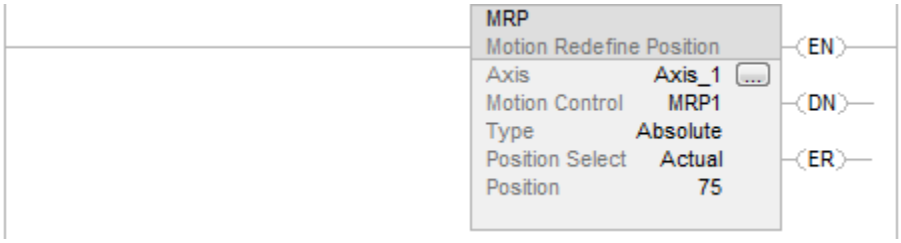
MRP Changes to Single Axis Status Bits

The AxisHomedStatus bit is not impacted by the execution of the MRP instruction. The status is the same before and after the MRP is execution. If the axis has been homed using the absolute home procedure, the AbsoluteReferenceStatus is set. AxisHomedStatus may also be set is the axis has not been subject to a power cycle. AbsoluteReferenceStatus is cleared when the MRP is executed and this is only true to Axis_Servo_Drive axis. This indicates the axis position is no longer referenced to the absolute home position.

Examples

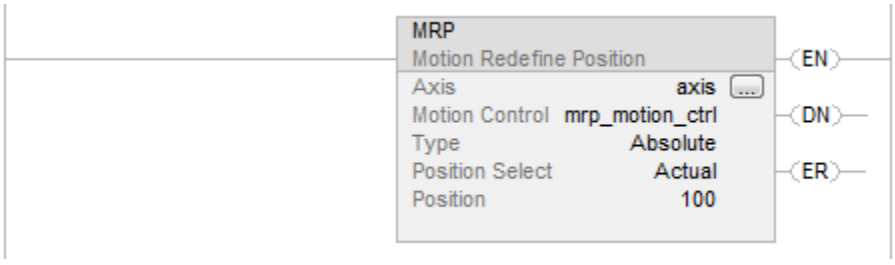
Example 1

Ladder Diagram

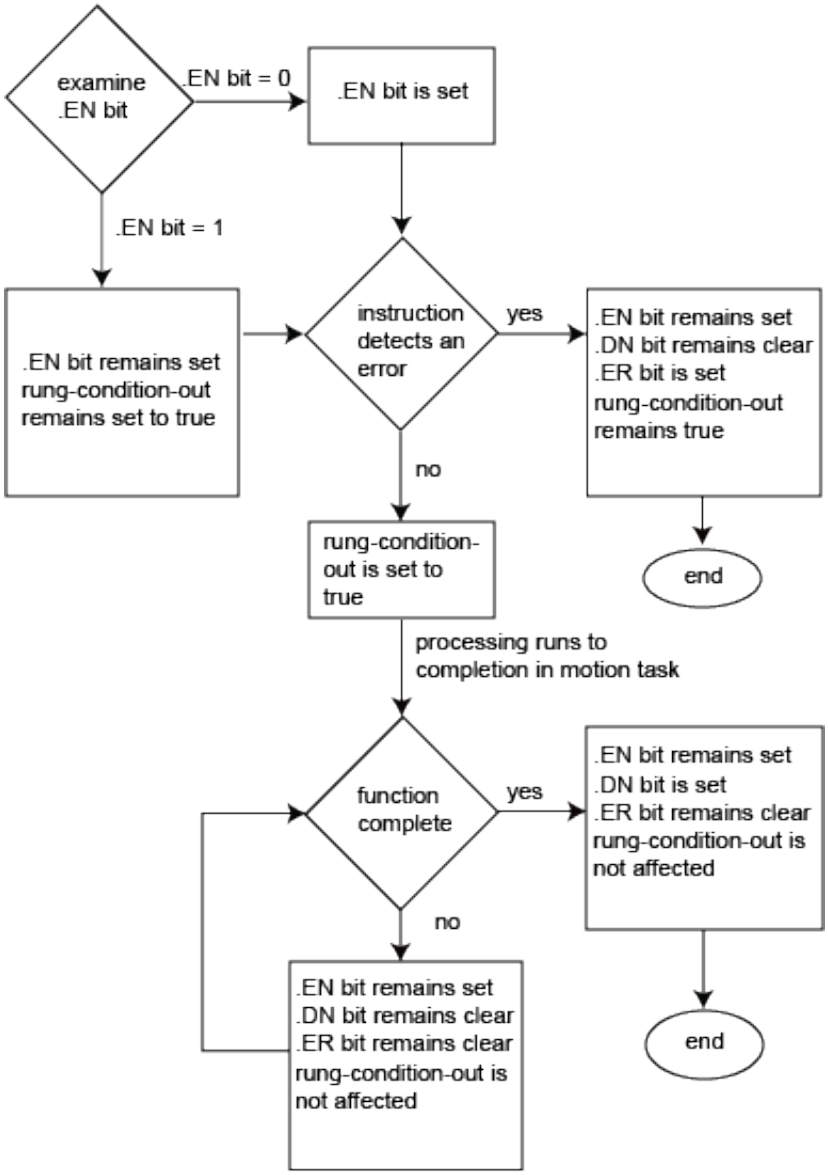


Example 2

Ladder Diagram



MRP Flow Chart (True)



Motion Calculate Cam Profile (MCCP)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

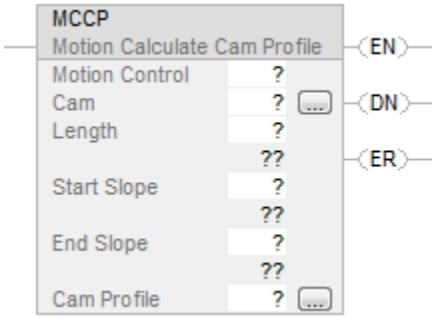
The Motion Calculate Cam Profile (MCCP) instruction calculates a cam profile based on an array of cam points. Establish an array of cam points programmatically or by using the Cam Profile Editor. Each cam point in the cam array consists of a slave position value, a master position (position cam) or time (time cam) value, and an interpolation type (linear or cubic). A Motion Axis Position Cam (MAPC) instruction or Motion Axis Time Cam (MATC) instruction can use the resulting cam profile to govern the motion of a slave axis according to master position or time.

This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.
-

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MCCP(MotionControl,Cam,Length,StartSlope,EndSlope,CamProfile);

Operands

Ladder Diagram

Operand	Type	Format	Description
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access block status parameters.

Cam	CAM, CAM_EXTENDED **Tip:** CAM_EXTENDED is supported by Compact GuardLogix 5580, CompactLogix 5380, and CompactLogix 5480 controllers only.	Array	Tag name of the cam array used to compute the cam profile. The numerical array index indicates the starting cam element in the array used in the cam profile calculation. Ellipsis launches Cam Profile Editor. Use the CAM_EXTENDED type for this operand to enable access to double-precision (64-bit LREAL) cam data members. If used, the Cam Profile operand must be of type CAM_PROFILE_EXTENDED.
Length	UINT	Immediate or Tag	Determines the number of cam elements in the array used in the cam profile calculation.
Start Slope	REAL	Immediate or Tag	This is the boundary condition for the initial slope of the profile. It is valid only for a cubic first segment and specifiesa slope through the first point.
End Slope	REAL	Immediate or Tag	This is the boundary condition for the ending slope of the profile. It is valid only for a cubic last segment and is used to specifies a slope through the last point.
Cam Profile	CAM_PROFILE, CAM_PROFILE_EXTENDED **Tip:** CAM_PROFILE_EXTENDED is supported by Compact GuardLogix 5580, CompactLogix 5380, and CompactLogix 5480 controllers only.	Array	Tag name of the calculated cam profile (Cam Profile) array used as input to MAPC and MATC instructions. Only the zero array element ([0]) is allowed for the Cam Profile array. Ellipsis launches Cam Profile Editor. Use the CAM_PROFILE_EXTENDED type for this operand to enable access to double-precision (64-bit LREAL) cam data members. If used, the Cam operand must be of type CAM_EXTENDED.

Structured Text

Operand	Type	Format	Description
Source	SINT INT DINT REAL STRING structure	tag	Initial element to copy **Important:** The Source and Destination operands should be the same data type, or unexpected results may occur.
Destination	SINT INT DINT REAL STRING LINT structure	tag	Initial element to be overwritten by the Source **Important:** The Source and Destination operands should be the same data type, or unexpected results may occur.
Length	DINT	immediate tag	Number of Destination elements to copy.

See Structured Text Syntax for more information on the syntax of expressions within structured text.

The operands are the same as those for the relay ladder MCCP instruction. For the array operands, you do not have to include the array index. If you do not include the index, the instruction starts with the first element in the array ([0]).

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	The Enable bit is set when the rung transitions from false-to-true and stays set until the done bit is set and the rung goes false.
.DN (Done) Bit 29	The Done bit is set when the calculate cam instruction has been executed and the Cam Profile array calculated.
.ER (Error) Bit 28	The error bit indicates when the instruction detects an error, such as if the cam array is of an illegal length.

Description

The MCCP instruction computes a cam profile based on a given set of points in a specified cam array. Subsequent Motion Axis Position Cam (MAPC) or Motion Axis Time Cam (MATC) camming instructions can use the resultant cam profiles generated by the MCCP instruction to provide complex motion of a slave axis with respect to either a master axis position or with respect to time.

Because cam profiles can be directly calculated by the Logix Designer Cam Profile Editor, the main purpose of the MCCP instruction is to provide a method for calculating cam profiles in real-time based on programmatic changes to the corresponding cam arrays.

Specifying a Cam Array

In order to execute an MCCP instruction, a Cam array tag must be created using the RSLogix Tag Editor or the Cam Profile Editor. The figure below illustrates how the Cam array tags are established and used as input to the MCCP instruction.

The Cam array elements consist of slave (yp) and master (xp) point pairs as well as an interpolation type. Since there is no association with a specific axis position or time, the x and y point values are unitless. The interpolation type may be specified for each point as either linear or cubic.

The MCCP instruction supports two types of Cam array: CAM and CAM_EXTENDED. Use CAM arrays to calculate cam profile tags of type CAM_PROFILE. Use CAM_EXTENDED arrays to calculate cam profile tags of type CAM_PROFILE_EXTENDED, which provide better precision.

Specifying the Cam Profile Tag

To execute a MAPC instruction, a Cam Profile array tag must also be created. Cam Profile array tags may be created by the Logix Designer tag editor or the MAPC/MATC instructions using the built-in Cam Profile Editor.

The data within the Cam Profile array can be modified at compile time by using the Cam Profile Editor, or at run-time with the MCCP instruction. In the case of run-time changes, a Cam array must be created in order to use the MCCP instruction.

The status parameter is used to indicate that the Cam Profile array element has been calculated. If execution of a camming instruction is attempted using any uncalculated elements in a cam profile, the MAPC or MATC instructions error. The type parameter determines the type of interpolation applied between this cam array element and the next cam element.

The MCCP instruction supports two types of Cam Profile array: CAM_PROFILE and CAM_PROFILE_EXTENDED. CAM_PROFILE is calculated from a CAM array. CAM_PROFILE_EXTENDED is calculated from a CAM_EXTENDED array, which provides better precision.

Cam Profile Array Status Member

The Status member of the first element in the cam profile array is special and used for data integrity checks. For this reason, the MCCP must always specify the cam profile with the starting index set to 0. This first cam profile element Status member can have the following values.

Status Variables	Description
0	Cam profile element has not been calculated.
1	Cam profile element is being calculated.
2	Cam profile element has been calculated.
n	Cam profile element has been calculated and is currently being used by (n-2) or MATC instructions.

Linear and Cubic Spline Interpolation

The resultant calculated cam profiles are fully interpolated. This means that if the current master position or time does not correspond exactly with a point in the cam array used to generate the cam profile, the slave axis position is determined by linear or cubic interpolation between adjacent points. In this way, the smoothest possible slave motion is provided. The MCCP instruction accomplishes this by calculating coefficients to a polynomial equation that determines slave position as a function of master position or time.

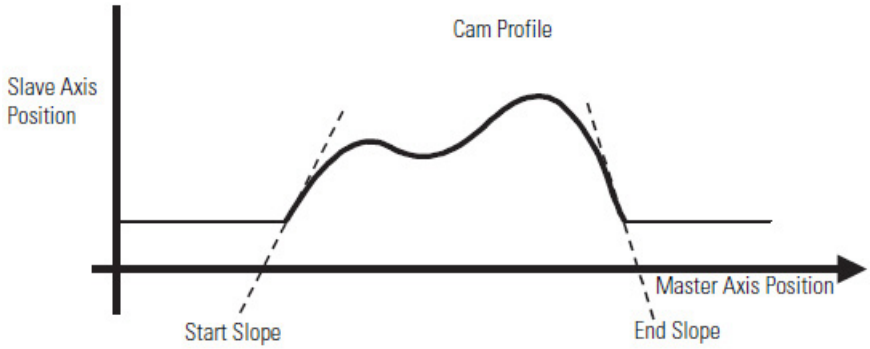
Calculating the Cam Profile

Before calculating a cam profile on a specified axis, the MCCP instructions first checks if the cam profile array has been calculated by checking the value of the first cam profile element's Status member. If the Status value is either 0 or 2, the MCCP proceeds with the calculation of the cam profile. When the cam profile array has been completely calculated, the MCCP instruction sets the first cam profile element's Status value to being calculated, or 1, and then sets the Status value of all other cam profile elements to being calculated. As the calculation proceeds, individual cam profile members' Status values are set to calculated, or 2. When all elements in the cam profile array have been calculated, the first cam profile element's Status value is also set to calculated.

However, if an MCCP instruction is executed with an initial cam profile Status value of 1, then the cam profile is currently being calculated by another MCCP instruction, and the MCCP instruction errors. If the Status value is >2, then the cam profile is being actively used by an MAPC or MATC instruction process, and the MCCP instruction errs.

Start Slope and End Slope

To facilitate a smooth entry into and exit from a cubic cam profile, slope control is provided. The Start Slope and End Slope parameters determine the initial rate of change of the slave relative to the master. These values are used in the cubic spline calculations performed on the cam array. The diagram below the master slave slope relationship.



The default values for Start Slope and End Slope are 0 to facilitate a smooth start and end to the cam profile from rest. However, if the axis is already camming, an appropriate non-zero Start Slope can be specified to match the End Slope of the currently executing cam, to seamlessly blend the two cam profiles together.

The Start Slope and End Slope values are not applicable when starting or ending the cam profile with linear interpolation.

IMPORTANT The MCCP instruction execution completes in a single scan. This instruction should therefore be placed in a separate task to avoid impacting user program scan time.

This is a transitional instruction:

- In relay ladder, toggle Rung-condition-in from false to true each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are not specific enough to help pinpoint the problem. When the MCCP instruction receives an Illegal Cam Length (26) error message to let it know that the length input parameter does not correspond to what the instruction expects, the corresponding Extended Error code provides the number of cams in the Cam Tag provided to the instruction. When the MCCP instruction receives an Illegal Cam Profile Length (27) error

message to let it know that the length input parameter does not correspond to what the instruction expects, the corresponding Extended Error code provides the number of cam points the instruction is attempting to generate. See Error Codes (ERR) for Motion Instructions.

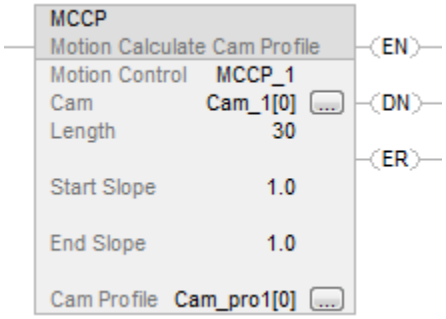
MCCP Changes to Status Bits

No

Examples

Example 1

Ladder Diagram

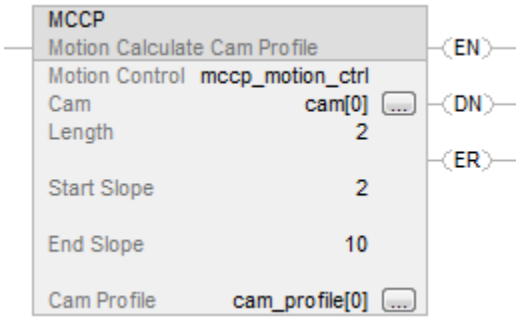


Structured Text

MCCP(MCCP_1,Cam_1[0],30,1.0,1.0,Cam_pro1[0]);

Example 2

Ladder Diagram



Structured Text

MCCP(mccp_motion_ctrl,cam[0],2,2,10,cam_profile[0]);

See also

Motion Move Instructions on page 73

Structured Text Syntax on page 665

Common Attributes on page 691

Motion Error Codes (.ERR) on page 581

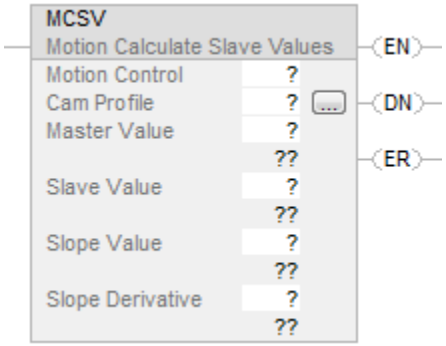
Motion Calculate Slave Values (MCSV)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Use the Motion Calculate Slave Values (MCSV) instruction to calculate the slave value, the slope value, and the derivative of the slope for a given cam profile and a master value.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MCSV(MotionControl,CamProfile,MasterValue,SlaveValue,SlopeValue,Slope Derivative)

Operands

There are data conversion rules for mixed data types within an instruction. See *Data Conversion*.

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access block status parameters.
Cam Profile	CAM_PROFILE, CAM_PROFILE_EXTENDED **Tip:** CAM_PROFILE_EXTENDED is supported by Compact GuardLogix 5580, CompactLogix 5380, and CompactLogix 5480 controllers only.	Array Tag	An array of elements with the array index set to 0. It defines the cam profile used in calculating the slave values. Use the CAM_PROFILE_EXTENDED type for this operand to enable access to double-precision (64-bit LREAL) cam data members.
Master Value	SINT, INT, DINT, or REAL	Immediate or Tag	The exact value along the master axis of the cam profile that is used in calculating the slave values.
Slave Value	REAL	Tag	The value along the slave axis of the cam profile with the master at the specified master value.

Slope Value	REAL	Tag	The first derivative of the value along the slave axis of the cam profile with the master at the specified master value.
Slope Derivative	REAL	Tag	The second derivative of the value along the slave axis of the cam profile with the master at the specified master value.

Structured Text

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

Description

The MCSV instruction determines the slave value, the slope value, and the derivative of the slope for a given cam profile and master value. As an extension to the position and time camming functionality, it supplies the values essential for the recovery from faults during camming operations.

Motion Control

These control bits are affected by the MCSV instruction.

Mnemonic	Description
.EN (Enable) Bit 31	The Enable Bit sets when the rung transitions from false to true. It resets when the rung goes from true to false.
.DN (Done) Bit 29	The Done Bit sets when the slave values have been calculated successfully. It resets when the rung transitions from false to true.
.ER (Error) Bit 28	The Error Bit sets when the slave values have not been calculated successfully. It resets when the rung transitions from false to true.

This is a transitional instruction:

- In relay ladder, toggle Rung-condition-in from false to true each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

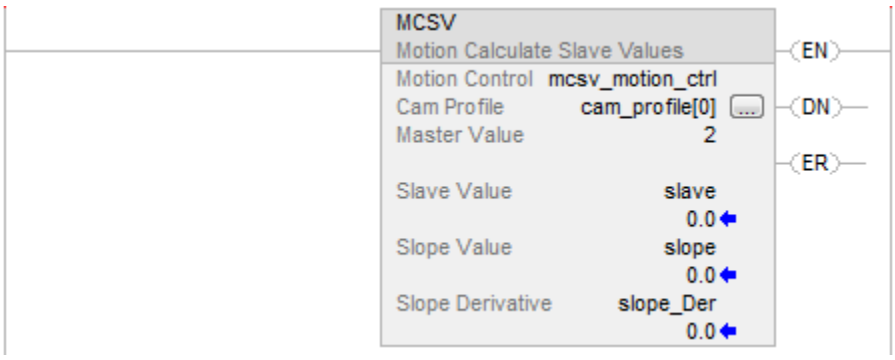
Extended Error Codes

Extended Error codes provide additional instruction-specific information for the error codes that are generic to many instructions. Extended error codes for the Parameter Out of Range (13) error code lists a number that refers to the number of the operand as listed in the faceplate from top to bottom with the first operand being counted as zero. Therefore for the MCSV instruction, an extended error code of 2 would refer to the Master Value operand's value. You would then have to check your value with the accepted range of values for the instruction. See *Motion Error Codes (.ERR)* for Motion Instructions.

MCSV Changes to Status Bits

None

Example



See also

Motion Error Codes (.ERR) on page 581

Structured Text Syntax on page 665

Common Attributes on page 691

Multi-Axis Coordinated Motion Instructions on page 359

Data Conversions on page 697

Motion Axis Position Cam (MAPC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The Motion Axis Position Cam (MAPC) instruction provides electronic camming between any two axes according to the specified Cam Profile.

When executed, the Slave axis is synchronized to the designated Master axis by using a position Cam Profile established by the Studio 5000 Logix Designer application Cam Profile Editor or Motion Calculate Cam Profile (MCCP) instruction.

The instruction also provides an easy means to cancel a running position cam anywhere during the cam execution and replace it with a different profile. The replacement is done immediately or is scheduled at a specific Master position along the running cam.

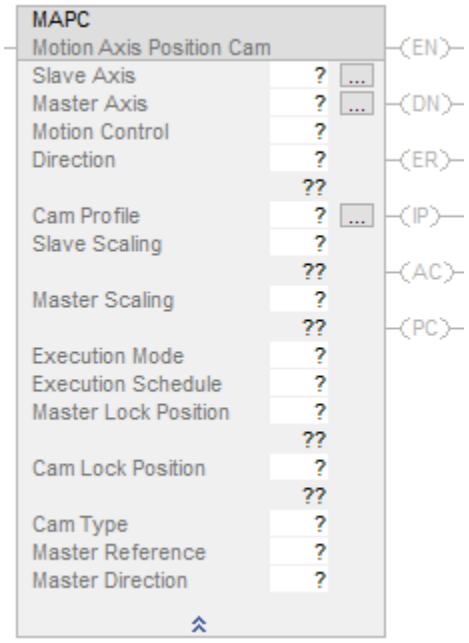
This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MAPC (SlaveAxis, MasterAxis, MotionControl, Direction, CamProfile, SlaveScaling, MasterScaling, ExecutionMode, ExecutionSchedule, MasterLockPosition, CamLockPosition, CamType, MasterReference, MasterDirection);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description
Slave Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL	AXIS_CIP_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE AXIS_GENERIC AXIS_GENERIC_DRIVE AXIS_VIRTUAL	Tag	The axis to which the cam profile applies. Ellipsis launches the Axis Properties dialog. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Master Axis	AXIS_CIP_DRIVE AXIS_CONSUMED AXIS_VIRTUAL AXIS_SERVO AXIS_SERVO_DRIVE **Tip:** AXIS_CONSUMED is supported by Compact GuardLogix 5580, CompactLogix 5380, and CompactLogix 5480 controllers only.	AXIS_CIP_DRIVE AXIS_CONSUMED AXIS_GENERIC_DRIVE AXIS_VIRTUAL AXIS_SERVO AXIS_SERVO_DRIVE	Tag	The axis that the Slave axis follows according to the cam profile. Ellipsis launches the Axis Properties dialog. If Pending is selected as the Execution Schedule, then Master Axis is ignored. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Structure used to access block status parameters.
Direction	DINT	DINT	Immediate	Relative direction of the Slave axis to the Master axis: 0 = Same – The Slave axis position values are in the same sense as the Master's. 1 = Opposite – The Slave axis position values are opposite sense of the Master's. 2 = Reverse – The current or previous direction of the position Cam is reversed on execution. When executed for the first time with Reverse selected, the control defaults the direction to Opposite. 3 = Unchanged – This allows other Cam parameters to be changed without altering the current or previous Camming direction. When executed for the first time with Unchanged selected, the control defaults the direction to Same.
Cam Profile	CAM_PROFILE CAM_PROFILE_EXTENDED **Tip:** CAM_PROFILE_EXTENDED is supported by Compact GuardLogix 5580, CompactLogix 5380, and CompactLogix 5480 controllers only.	CAM_PROFILE CAM_PROFILE_EXTENDED	Array	Tag name of the calculated cam profile array used to establish the master/slave position relationship. Only the zero array element ([0]) is allowed for the Cam Profile array. Ellipsis launches Cam Profile Editor. Use the CAM_PROFILE_EXTENDED type for this operand to enable access to double-precision (64-bit LREAL) cam data members.

Slave Scaling	REAL	REAL	Immediate or Tag	Scales the total distance covered by the slave axis through the cam profile.
Master Scaling	REAL	REAL	Immediate or Tag	Scales the total distance covered by the master axis through the cam profile.
Execution Mode	DINT	DINT	Immediate	Determines if the cam profile is executed only one time or repeatedly: 0 = Once – Once started the slave axis follows the Cam profile until the Cam boundary is crossed. On crossing of the Cam boundary, Cam motion on the Slave axis stops and the .PC bit is set. The Position Cam Status bit in Slave axis' Motion Status word is reset. The Slave motion does not resume if the Master axis moves back into the Cam profile range. 1 = Continuous – Once started the cam profile is executed indefinitely. This feature is useful in rotary applications where it is necessary that the cam position run continuously in a rotary or reciprocating fashion. 2 = Persistent - When the Master axis moves beyond the defined range, cam motion on the Slave axis stops. Cam motion on the Slave resumes in the opposite direction when the Master axis reverses and moves back into the Cam Profile range.
Execution Schedule	DINT	DINT	Immediate	Selects the method used to execute the cam profile. 0 = Immediate – The slave axis is immediately locked to the master axis and the position camming process begins. 1 = Pending – Lets you blend a new position cam execution after an in process position cam is finished. When Pending is selected, these parameters are ignored: Master Axis, Master Lock Position, and Master Reference. 2 = Forward only – The cam profile starts when the master position crosses the Master Lock Position in the forward direction. 3 = Reverse only – The cam profile starts when the master position crosses the Master Lock Position in the reverse direction. 4 = Bidirectional – The cam profile starts when the master position crosses the Master Lock Position in either direction.
Master Lock Position	REAL	REAL	Immediate or Tag	When the Master Offset = 0.0, the Master Lock Position is the Master axis absolute position where the slave axis locks to the master axis. If the Master Offset is X, the Slave axis locks to the Master axis at the absolute master position value of Master Lock Position –X. For example: Assume a Master Lock Position = 50 and a Master Offset Move = 10. Also assume that the Master axis move (MAM) and Master offset move (MOM) start at the same time. Then the Slave will lock to the Master at an absolute Master axis position of 40. This in effect shifts the Cam profile 10 units to the left. If Pending is selected as the Execution Schedule value, then Master Lock Position is ignored.

Cam Lock Position	REAL	REAL	Immediate or Tag	This determines the starting location in the cam profile.
Cam Type	DINT	DINT	Immediate	0 = New Cam - The user wants to start a new Cam. This is the default value and provides backward compatibility for pre-V34 Studio 5000 Logix Designer releases. 1 = Replace and Restart - The replacement Cam will replace the active Cam and start interpolating at a position indicated by the Cam Lock Position. 2 = Replace and Continue – The replacement Cam will replace the active cam and will start interpolating from an internally calculated index which is the point at which the active Cam was replaced. The Cam Lock Position is ignored. This operand is new with V34 Studio 5000 Logix Designer.
Master Reference	DINT	DINT	Immediate	Sets the master position reference to either Command position or Actual position. If Pending is selected for the Execution Schedule value, then Master Reference is ignored. 0 = Actual – slave axis motion is generated from the current position of the master axis as measured by its encoder or other feedback device. 1 = Command – Slave axis motion is generated from the desired or commanded position of the master axis.
Master Direction	DINT	DINT	Immediate	This determines the direction of the master axis that generates slave motion according to the cam profile. Options are: 0 = Bidirectional – slave axis can track the master axis in either direction. 1 = Forward only – Slave axis tracks the master axis in the forward direction of the master axis. 2 = Reverse only – Slave axis tracks the master axis in the opposite direction of the master axis.

Structured Text

MAPC (SlaveAxis, MasterAxis, MotionControl,Direction, CamProfile, SlaveScaling, MasterScaling, ExecutionMode,

ExecutionSchedule, MasterLockPosition, CamLockPosition, CamType, MasterReference, MasterDirection);

The operands are the same as those for the relay ladder MAPC instruction. For the array operands, you do not have to include the array index. If you do not include the array index, the instruction starts with the first element in the array ([0]).

For the operands that require you to select from available options, enter your selection as:

This Operand	Has These Options Which You

	Enter as Text	Or Enter as a Number
ExecutionMode	Once	0
	Continuous	1
	Persistent	2
ExecutionSchedule	Immediate	0
	Pending	1
	ForwardOnly	2
	ReverseOnly	3
	Bidirectional	4
CamType	New Cam	0
	Replace and Restart	1
	Replace and Continue	2
MasterReference	Actual	0
	Command	1
MasterDirection	Bidirectional	0
	ForwardOnly	1
	ReverseOnly	2

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set when the rung makes a False-to-True transition and remains set until the rung goes False.
.DN (Done) Bit 29	It is set when the position cam is initiated.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.IP (In Process) Bit 26	It is set on positive rung transition and cleared if either superseded by another Motion Axis Position Cam command, or canceled by a stop command, merge, shut down, or servo fault.
.AC (Active) Bit 23	It is set when the cam begins interpolation of the Slave axis. It is reset when the Active cam execution is completed or cancelled by a stop command, merge, shut down, or servo fault.
.PC (Process Complete) Bit 27	It is cleared on positive rung transition and set in ONCE Execution Mode, when the position of the Master axis leaves the Master position range defined by the active Cam profile.

Tip: Version 34 and newer: The information in the MOTION_INSTRUCTION Structure Table pertaining to the .AC bit applies to the Compact GuardLogix (5380), CompactLogix (5380), CompactLogix (5480), ControlLogix (5580), GuardLogix (5580) and Logix Emulate series of controllers only. For other controllers, the .AC bit is shown on the instruction faceplate but is inactive and is always equal to False.

Description

The MAPC instruction executes a position cam profile set up by a previous Motion Calculate Cam Profile (MCCP) instruction or, alternatively, by the Logix Designer application Cam Profile Editor. Position cams, in effect,

provide the capability of implementing non-linear electronic gearing relationships between two axes. No maximum velocity, acceleration, or deceleration limits are used. The motion of the Master Axis and designated Cam profile derived from the associated cam table determines the speed, acceleration, and deceleration of the Slave axis.

WARNING: The maximum velocity, acceleration, or deceleration limits established during axis configuration do not apply to electronic camming.

The Direction input parameter determines the direction of Slave axis motion relative to the Master axis. The camming direction, as applied to the Slave, can be explicitly set as the Same or Opposite or set relative to the current camming direction as Reverse or Unchanged.

To accurately synchronize the Slave axis position to Master axis position, an Execution Schedule and an associated Master Lock Position setting can be specified for the Master axis. When the Master axis travels past the Master Lock Position in the direction specified by the Execution Schedule parameter, the Slave axis is locked to the Master axis position according to the specified Cam Profile beginning at the Master Lock Position.

The Cam Profile can also be configured to execute Immediately or Pending completion of a currently executing position cam profile via the Execution Schedule parameter.

The Cam Profile can be executed once, continuous, or in a persistent mode by specifying the desired Execution Mode.

The Master Reference selection allows camming input from the Master to be derived from either the Actual or Command position of the Master axis. A slip clutch feature is available to support applications that require unidirectional motion. It helps to prevent the Slave from backing up when the Master axis reverses direction. The Master Direction parameter controls this feature.

Master and Slave Scaling functionality can be used to scale Slave motion based on a standard cam profile without having to create a cam table and calculate a new cam profile.

The Cam Type selection allows the user to cancel a running cam and replace it with another cam immediately or schedule it to perform at a later position of the Master axis position.

Camming Direction

Cams can be configured to add or subtract their incremental contribution to the Slave axis command position. Control over this behavior is via the Direction parameter.

Camming in the Same Direction

When Same is selected or entered as the Direction for the MAPC instruction, the Slave axis position values computed from the cam profile are added to the command position of the Slave axis. This is the most common operation, as the profile position values are used as entered in the original cam table. Consecutive-increasing profile values result in axis motion in the positive direction and vice versa.

Camming in the Opposite Direction

When Opposite is selected or entered as the Direction, the Slave axis position values computed from the cam profile are subtracted from the command position of the slave axis. Thus, axis motion is in the opposite direction from that implied by the original cam table. Consecutive increasing profile values result in axis motion in the negative direction and vice versa.

Preserving the Current Camming Direction

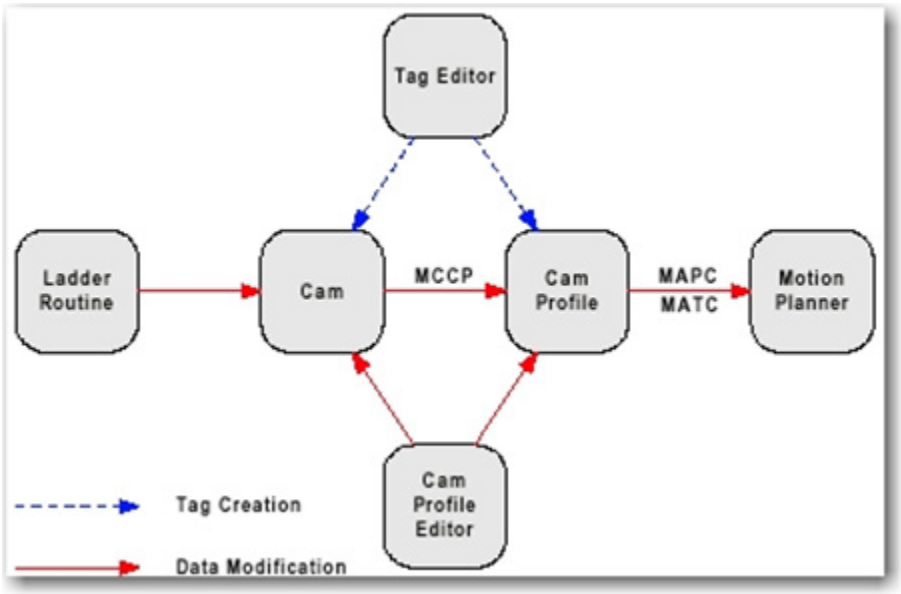
When Unchanged is selected or entered as the Direction, other position cam parameters can be changed while preserving the current or previous camming direction (same or opposite). This is useful when the current direction is not known or not important. For first-time execution of a cam with Unchanged selected, the control defaults the direction to Same.

Reversing the Current Camming Direction

When Reverse is selected the current or previous direction of the position cam is changed from Same to Opposite or from Opposite to Same. For first-time execution of a cam with Reverse selected, the control defaults the direction to Opposite.

Specifying the Cam Profile

To execute a MAPC instruction, a calculated cam profile data array tag must be specified. The user can generate the Cam Profile array tags by using the Logix Designer application tag editor or the MAPC instruction by using the built-in Cam Profile Editor, or by executing a Motion Calculate Cam Profile (MCCP) instruction on an existing Cam array.



The data within the Cam Profile array can be modified at compile time using the Cam Profile Editor, or at run-time with the Motion Calculate Cam Profile (MCCP) instruction. In the case of run-time changes, a Cam array must be created in order to use the MCCP instruction. Refer to the MCCP instruction specification for more detail on converting Cam arrays.

MAPC supports two types of Cam Profile array: CAM_PROFILE and CAM_PROFILE_EXTENDED. CAM_PROFILE is calculated from a CAM array. CAM_PROFILE_EXTENDED is calculated from a CAM_EXTENDED array, which provides better precision.

For CAM_PROFILE array, all but the status and type element of this Cam Profile array element structure are hidden from the Logix Designer application tag editor. These elements are of no value to the user. The Status member is used to indicate that the corresponding Cam Profile array element has been calculated. If execution of a camming instruction is attempted with any uncalculated elements in a cam profile, an error occurs. The type parameter determines the type of interpolation applied between this cam array element and the next cam element (for example, linear or cubic).

For CAM_PROFILE_EXTENDED array, the status, master and slave values, type, and C0, C1, C2, C3 coefficients are visible in the Logix Designer tag editor. Use the status parameter to indicate that the CAM_PROFILE array element has been calculated. If a camming instruction begins to execute with uncalculated elements in a CAM_PROFILE, an error occurs. The master and

slave define the x and y value of the cam element. The type parameter determines the type of interpolation applied between this cam array element and the next cam element (for example, linear or cubic). The C0, C1, C2, and C3 are coefficients that define the shape between two cam elements. They are calculated by the Motion Calculate Cam Profile (MCCP) instruction or Cam Profile Editor and should not be modified. Attempting to modify C0, C1, C2, and C3 coefficients could cause unintended motion.

WARNING: Do not modify the Cam Profile array directly. Modifying the Cam Profile array can cause unintended motion or a motion fault.
Always use Motion Calculate Cam Profile (MCCP) or the Cam Profile editor to adjust the Cam Profile array.

Cam Profile Array Checks

The Status member of the first element in the cam profile array is special and used for data integrity checks. For this reason, the MAPC must always specify the cam profile with the starting index set to 0.

This first cam profile element Status member can have the following values.

Status Variables	Description
0	Cam profile element has not been calculated.
1	Cam profile element is being calculated.
2	Cam profile element has been calculated
n	Cam profile element has been calculated and is being used by (*n*-2) MAPC or MATC instructions.

Before starting a cam on a specified axis, the MAPC instructions check if the Cam Profile array has been calculated by checking the value of the first Cam Profile element's Status member. If the status is 0, 1 then the cam profile has not been calculated yet and the MAPC instruction error flag is set. If the Cam Profile array has been completely calculated (Status > 1), the instruction then increments the Status member indicating that it is in use by this axis.

When the cam completes, or ends, the Status member of the first Cam Profile array element is decremented to keep track of the number of cams that are using the associated cam profile.

Linear and Cubic Interpolation

Position cams are fully interpolated. This means that if the current Master axis position does not correspond exactly with a point in the cam table associated with the cam profile, the slave axis position is determined by linear or cubic interpolation between the adjacent points. In this way, the smoothest possible slave motion is provided.

Each point in the Cam array that was used to generate the Cam Profile can be configured for linear or cubic interpolation.

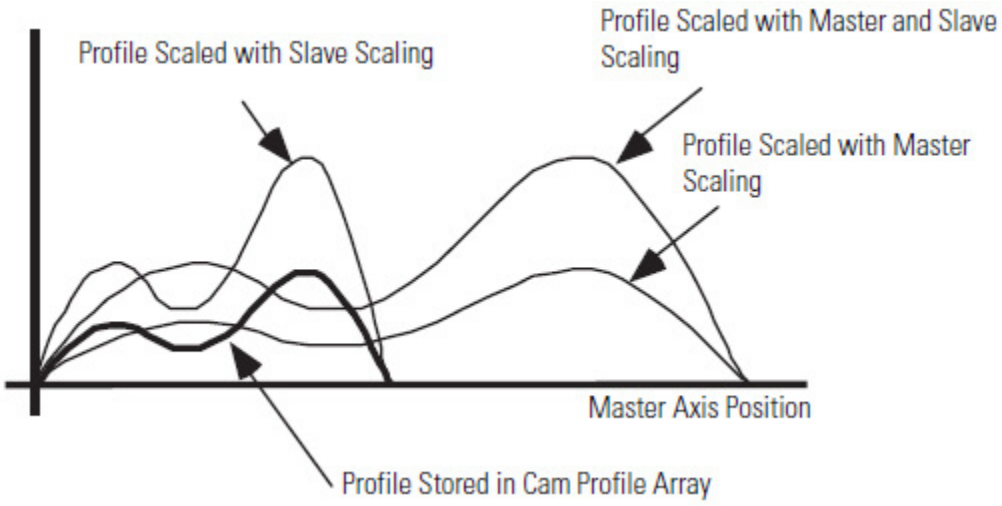
Electronic camming remains active through any subsequent execution of jog, or move processes for the Slave axis. This allows electronic camming motions

to be superimposed with jog, or move profiles to create complex motion and synchronization.

Scaling Position Cams

A position cam profile can be scaled in both the master dimension and slave dimension when it is executed. This scaling feature is useful because it allows the stored cam profile to be used to determine the general form of the motion profile. The scaling parameters are then used to define the total master or slave travel over which the profile is executed, as shown in the illustration below. In this way, one standard cam profile can be used to generate a whole family of specific cam profiles.

When a cam profile array is specified by an MAPC instruction, the master and slave values defined by the cam profile array take on the position units of the master and slave axes respectively. By contrast, the Master and Slave Scaling parameters are unitless values that are used as multipliers to the cam profile.



By default, both the Master Scaling and Slave Scaling parameters are set to 1. To scale a position cam profile, enter a Master Scaling or Slave Scaling value other than 1.

Increasing the master scaling value of a cam profile decreases the velocities and accelerations of the profile, while increasing the slave scaling value increases the velocities and accelerations of the profile. To maintain the velocities and accelerations of the scaled profile approximately equal to those of the unscaled profile, the master scaling and slave scaling values should be equal. For example, if the slave scaling value of a profile is 2, the master scaling value should also be 2 to maintain approximately equal velocities and accelerations during execution of the scaled position cam.

WARNING: Decreasing the Master Scaling value or increasing the Slave Scaling value of a position cam increases the required velocities and accelerations of the profile. This can cause a motion fault if the capabilities of the drive system are exceeded.

Execution Mode

Once

Execution Mode of Once, Continuous, or Persistent can be selected to determine how the cam motion behaves when the Master position moves beyond the start and end points of the profile defined by the original cam table.

When Once is selected (default), the specified cam profile, once started, executes until the Cam boundary is crossed. When the Master axis moves outside the range of the profile, cam motion on the Slave axis stops and the Process Complete (.PC) bit of the MAPC instruction is set. The Slave motion does not resume when and if the Master moves back into the profile range specified by the start and end points.

Continuous

When Continuous mode is selected, the specified cam profile, once started, is executed indefinitely. With continuous operation, the profile's Master and Slave positions are unwound when the position of the Master axis moves outside the profile range, causing the cam profile to repeat. This feature is useful in rotary applications where it is necessary that the position cam run continuously in a rotary or reciprocating fashion. To generate smooth continuous motion by using this technique, however, care must be taken in designing the cam points of the cam table to confirm that there are no position, velocity, or acceleration discontinuities between the start and end points of the calculated cam profile.

Persistent

When Persistent mode is selected, the cam motion of the slave moves if the Master axis is within the cam profile range. When Master axis moves beyond the defined range, the cam motion on the Slave axis stops. When Master axis reverses and moves back into the cam Profile range, Slave axis resumes its cam motion as per cam profile.

Execution Schedule

The Execution Schedule parameter controls the execution of the MAPC instruction.

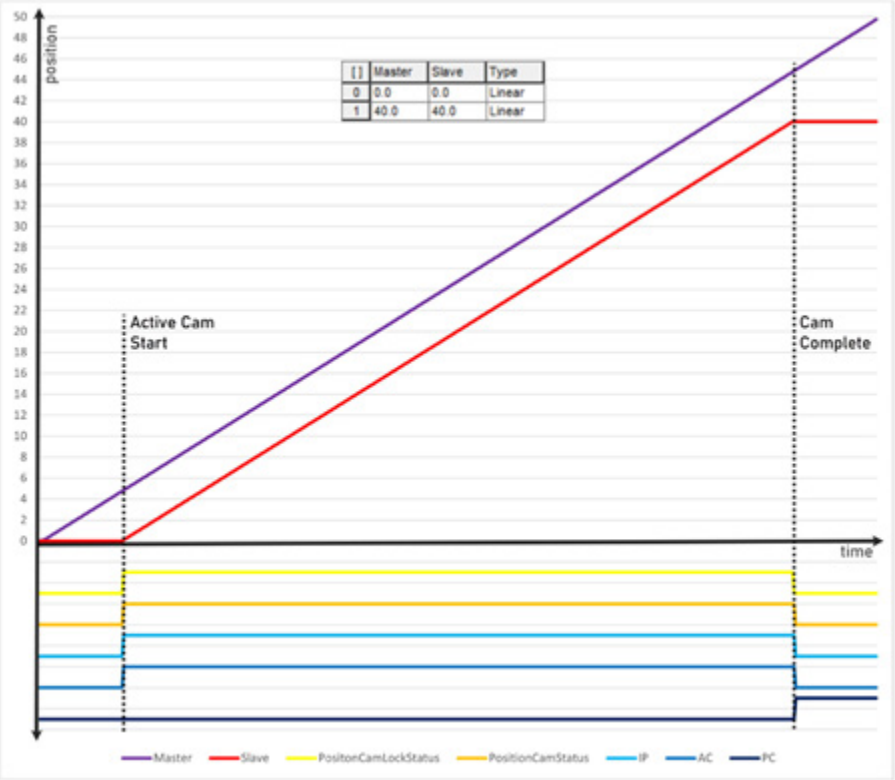
Immediate

By default, the execution schedule of MAPC instruction is Immediate. In Immediate execution, the Slave axis is immediately locked to the master axis and the position camming begins.

Active Cam

In the Immediate Active Cam case, there is no delay in execution of the position camming process. In this case, the Master Lock position parameter is irrelevant.

This figure illustrates Immediate Active Cam execution, with the camming process initiated on the Slave axis. Take notice of the Position Cam Status and Position Cam Lock Status of the Slave axis' Motion status word is set to True. The Slave axis immediately locks to the master axis and follows the Cam profile.

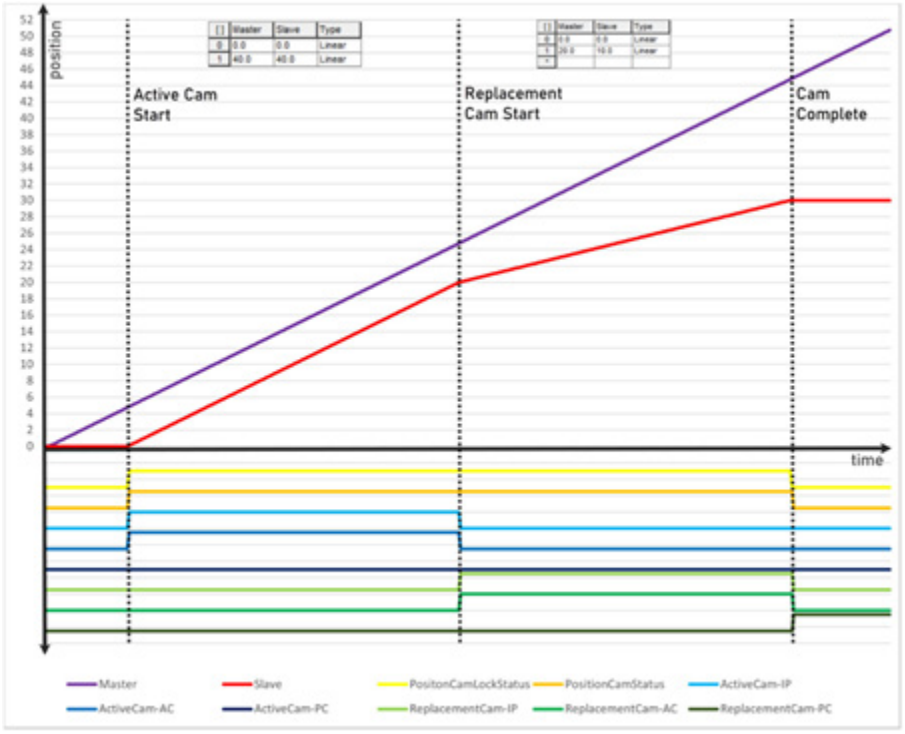


Replace and Restart Cam

In the Immediate Replace and Restart Cam case, the Replacement Cam replaces the Active cam immediately at the programmed Cam Lock Position.

This figure illustrates Immediate Replace and Restart Cam execution. The Active cam initiates the camming process, and the Replacement cam is initiated with execution schedule of Immediate. The replacement happens at the programmed Cam Lock Position of zero, the Master Lock Position is ignored.

The Position Cam Status and Position Cam Lock Status bits of the Slave Axis' Motion Status bits are set.

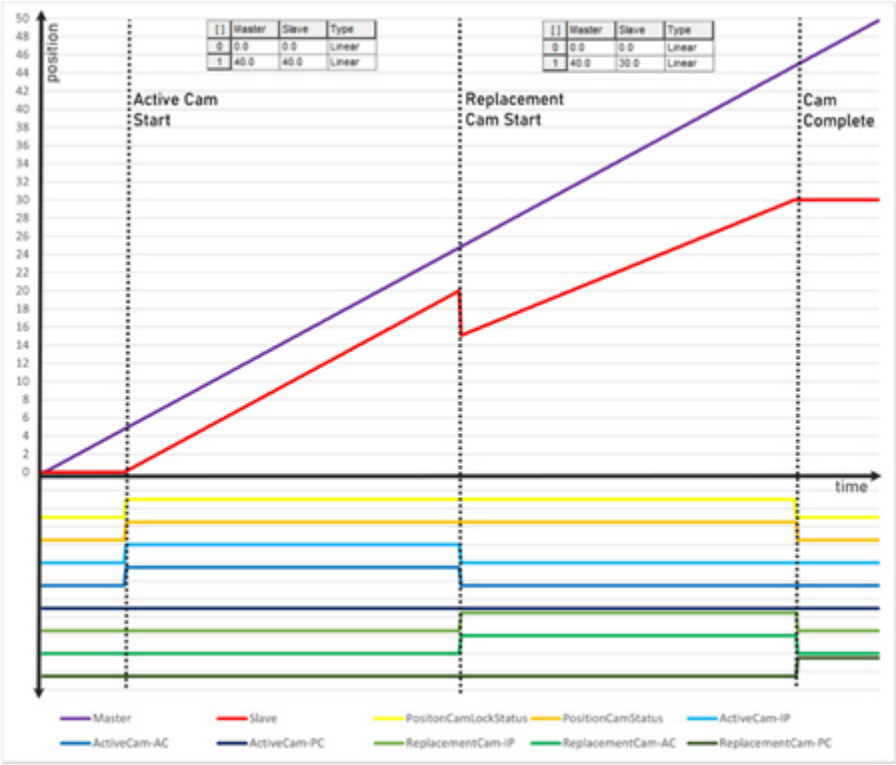


Replace and Continue Cam

In the Immediate Replace and Continue Cam case, the Replacement Cam replaces the Active cam immediately at the point at which the replacement was initiated. The Cam lock Position and Master Lock Position are ignored.

This figure illustrates Immediate Replace and Continue Cam execution. The Active cam initiates the camming process, and the Replacement cam is initiated with execution schedule of Immediate.

The Position Cam Status and Position Cam Lock Status bits of the Slave axis' Motion Status bits are set.



Forward Only, Reverse Only, or Bidirectional

In the case where the Execution Schedule parameter of the instruction is set to Forward Only, Reverse Only, or Bidirectional, the Slave axis is not locked to the Master until the Master axis satisfies the corresponding condition.

Forward Only:

The replacement Cam replaces the Active Cam when the Cam reaches the replacement point while the Master moves in the forward direction.

Reverse Only:

Same as Forward Only except that the Master is moving in the reverse direction.

Bidirectional:

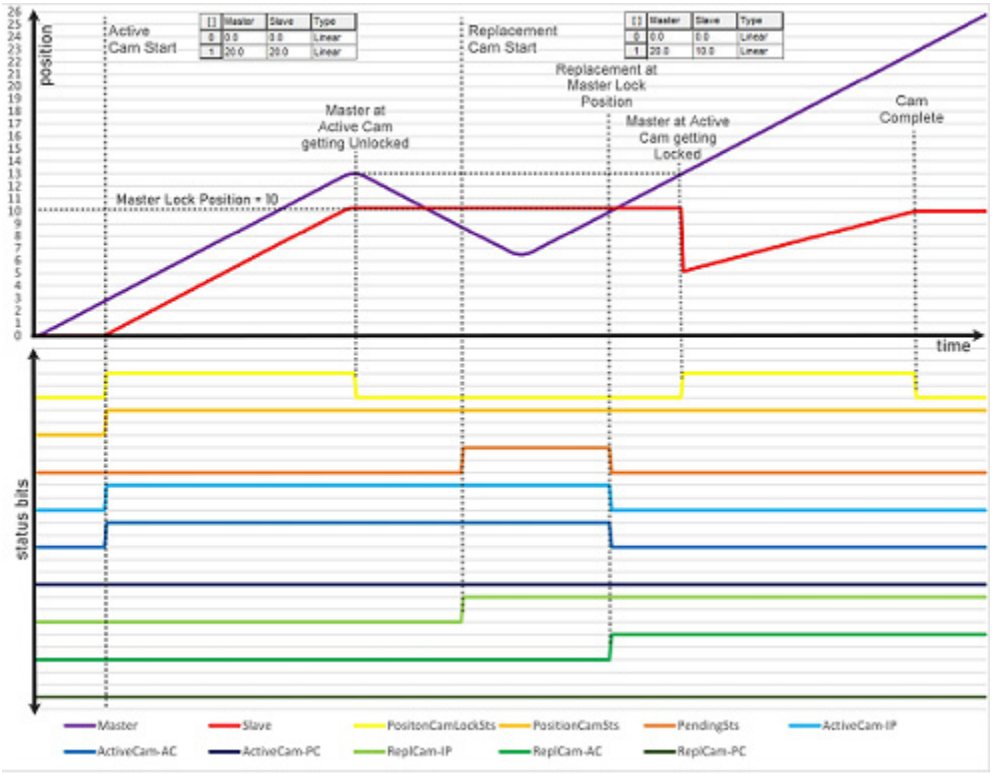
Same as Forward Only except that the Master is moving in either direction.

Replace and Continue Cam with Active Cam Unlocked:

The diagram illustrates when an Active Cam is initiated with Master Direction Forward Only, Execution Mode Immediate, and the Replacement Cam is executed later. The status transitions are indicated by the dotted vertical lines in the diagram.

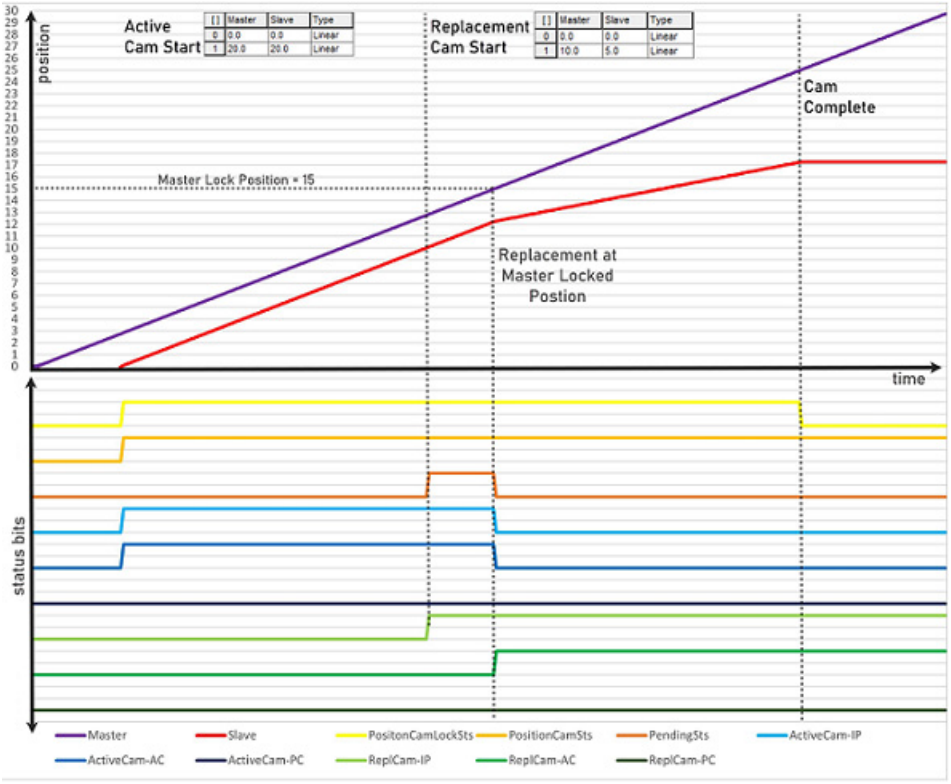
- Active Cam Start: The Slave axis starts following the Master axis in the forward direction. The Position Cam Status and the Position Cam Lock Status of the Slave axis' Motion status word bits are set to True.
- Master at Active Cam getting Unlocked: Due to the Master axis reversal, the Active cam is unlocked, and the Position Cam Lock Status of the Slave axis' Motion status word bit is reset.
- Replacement Cam Start: While the Master Axis is moving in the reverse direction, the Replacement Cam is initiated with Active Cam still unlocked, and with the following configurations:
  - Execution Schedule: Forward Only
  - Master Direction: Forward Only
  - Execution Mode: Persistent (PC bit does not turn to True after Cam Complete)
  - Programmed Master Lock Position: 10 units
- Replacement at Master Lock Position: When the Master axis starts moving in forward direction and reaches Master Lock Position of 10 units, then the AC bit of the Replacement Cam Motion status word is set to True indicating that the Replacement Cam is ready to interpolate the Slave axis.

- Master at Active Cam getting Locked: When the Master axis is moving in forward direction and reaches the Master reversal point of the Active Cam, then the Active Cam is locked, and the Replacement Cam starts driving the Slave axis. Position Cam Lock Status of the Slave axis' Motion status word bit is set to True. The Slave axis starts moving according to the Replacement Cam profile until the Master axis crosses the Cam boundary. The Position Cam Lock Status of the Slave axis' Motion status word bit is reset.



- Replace and Restart Cam with Active Cam Locked: The diagram illustrates when an Active Cam is initiated with Master Direction Forward Only, Execution Mode Immediate, and the Replacement Cam is executed later, the status transitions are indicated by the dotted vertical lines in the figure.
  - Active Cam Start: The Slave axis starts following the Master axis in the forward direction. The Position Cam Status, Position Cam Lock Status of the Slave axis' Motion status word bits are set to True.
  - Replacement Cam Start: While the Master axis is moving in forward direction, the Replacement Cam is initiated with Active Cam still Locked, and with the following configurations:
    - Execution Schedule: Forward Only
    - Master Direction: Forward Only
    - Execution Mode: Persistent (PC bit does not turn to true after Cam Complete)
    - Programmed Master Lock Position: 15 units

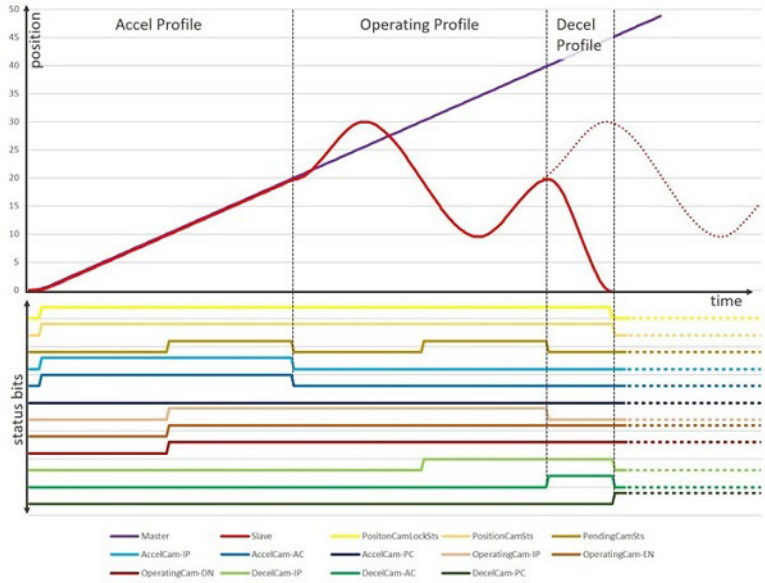
- Replacement at Master Lock Position: When the Master Axis starts moving in the forward direction and reaches Master Lock Position of 15 units, then the AC bit of the Replacement Cam status word is set to True indicating that the Replacement Cam is ready to interpolate the Slave Axis. The Replacement Cam starts moving according to the Replacement Cam profile until the Master Axis crosses the Cam boundary. The Position Cam Lock Status of the Slave Axis' Motion status word bit is set to False then.



Pending Cam

Alternatively, the MAPC instruction's execution can be deferred pending completion of a currently executing position cam. An Execution Schedule selection of Pending can thus be used to blend two position cam profiles without stopping motion.

The Pending execution feature is useful in applications like high-speed packaging when a Slave axis must be locked onto a moving Master axis and accelerate by using a specific profile to the proper velocity. When this acceleration profile is done, it must blend into the operating profile, which is typically executed continuously. To stop the Slave axis, the operating profile is smoothly blended into a deceleration profile such that the axis stops at a known location as shown in the figure.



To confirm smooth motion across the transition, however, the profiles must be designed such that no position, velocity, or acceleration discontinuities exist between the end of the current profile and the start of the new one. This is done by using the Studio 5000 Logix Designer application Cam Profile Editor.

If an Execution Schedule of Pending is selected without a corresponding Active Cam profile in-process, the MAPC instruction executes but no camming motion occurs until another MAPC instruction with a non-pending Execution Schedule is initiated. This allows pending Cam profiles to be preloaded before executing the initial cam. This method addresses cases where immediate Cams would finish before the pending Cam could be reliably loaded.

Replacement Cam effect on a Pending Cam

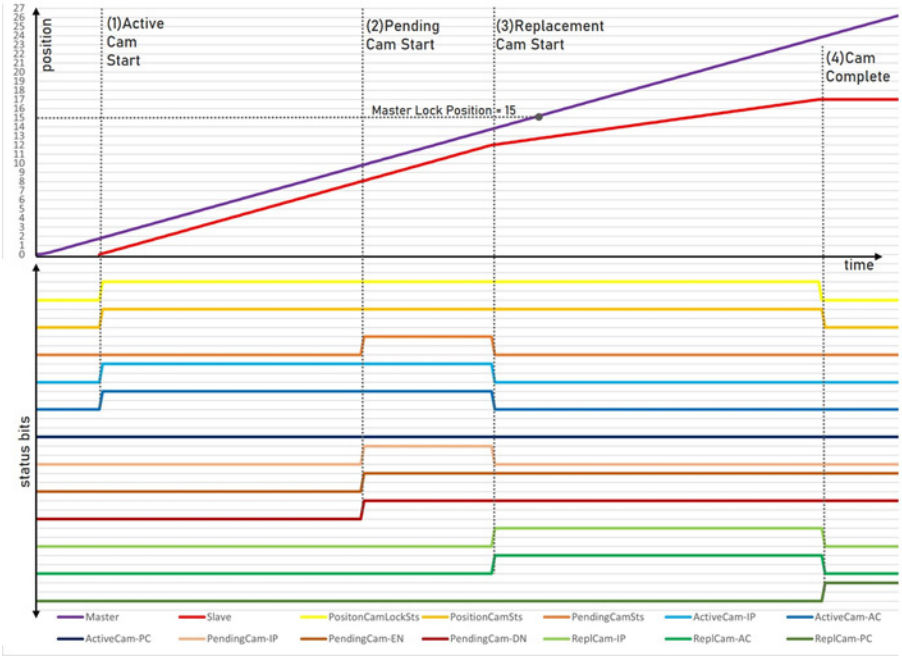
This figure shows an example of how Pending Cam status flags are affected when a Replacement Cam executes before a Pending Cams' start. The dotted vertical lines in the figure indicates the status transitions. When the Replacement Cam executes, the Pending Cam's .IP bit is set and the Replace and Restart Cam Profile runs.

(1) Active Cam starts.

(2) Pending Cam is enabled and waiting for the completion of the Active Cam. The Pending Cam's. IP and .DN bits are set.

(3) Replacement Cam executes before the end of the Active Cam profile completes. The Pending Cam .IP bit resets.

(4) Cam Complete, replacement Cam's .PC bit is set.



Master Lock Position

The programmed Master Lock Position of the replacement Cam is the replacement position (point).

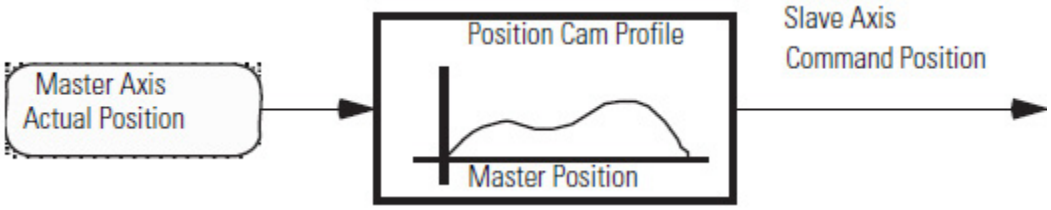
In a Rotary Axis configuration, Master Lock Position is independent of the turns count. The user must confirm that the Master Lock position is within the profile range of the active Cam.

Master Reference

The Master Reference parameter determines the master position source to link to the cam generator. This source can be actual position or command position of the master axis. Smoother motion is derived from command position but in some cases, for example when a physical axis is not controlled by a ControlLogix motion module, actual position is the only practical option.

Slaving to the Actual Position

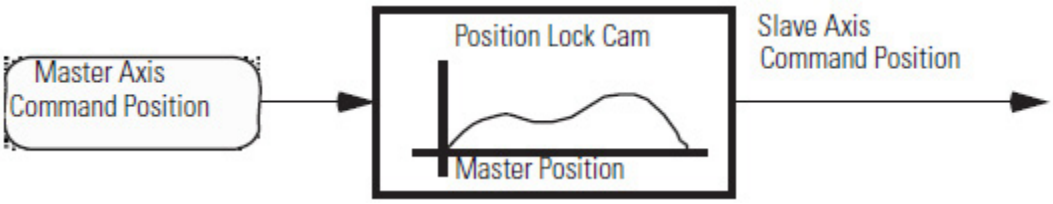
When Actual Position is entered or selected as the Master Reference source, the Slave axis motion is generated from the actual position of the master axis as shown below.



Actual position is the current position of the Master axis as measured by its encoder or other feedback device. The only selection when the Master axis Type is configured as Feedback Only since it is often necessary to synchronize the actual positions of two axes.

Slaving to the Command Position

When Command Position is entered or selected as the Master Reference source, the Slave axis motion is generated from the command position of the Master axis as shown below.



Command position (only available when the Master axis' Axis Type is a Servo or Virtual axis) is the desired or commanded position of the Master axis.

Since the command position does not incorporate any associated following error or external position disturbances, it is a more accurate and stable reference for camming. When camming to the command position of the master, the Master axis must be commanded to move to cause any motion on the Slave axis.

Master Direction

Normally, the Master Direction parameter is set to Bidirectional (default). However, when Forward Only is selected for Master Direction, the lave axis tracks the master axis in the forward direction of the Master axis. When Reverse Only is selected, the Slave axis tracks the Master axis in the reverse direction of the Master axis. If the Master axis changes direction, the Slave axis does not reverse direction, but stays where it was when the master reversed. This Uni-directional feature of position cams is used to provide an

electronic slip clutch, which helps prevent the cam motion generator from moving backward through the cam profile if the master reverses direction.

When the Master axis again reverses, resuming motion in the desired direction, the Slave axis picks up again when the master reaches the position where it initially reversed. In this way, the Slave axis maintains synchronization with the master while motion in the wrong direction is inhibited. This is especially useful where motion in a certain direction can damage the machine or to the product.

Moving While Camming

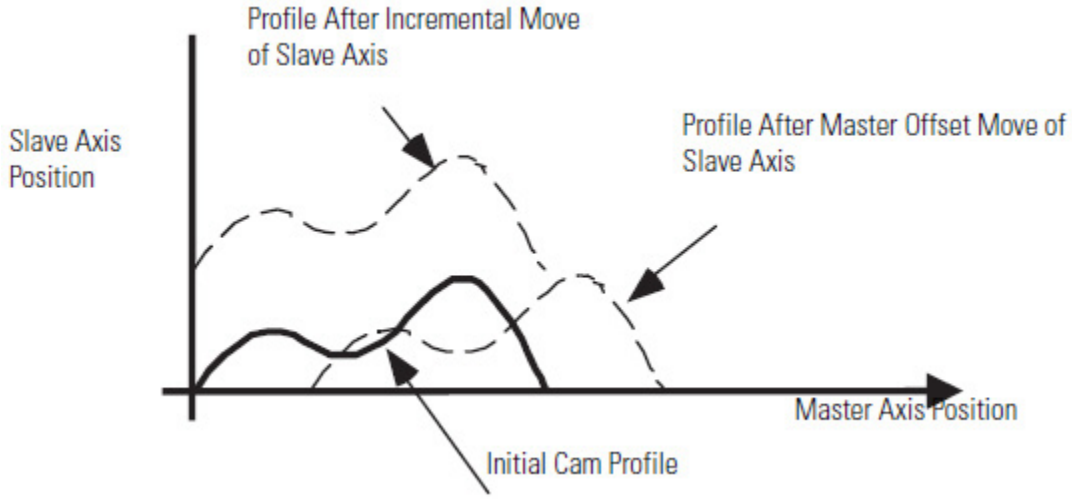
Motion Axis Moves can be performed while camming to provide sophisticated phase and offset control while the Slave axis is running.

Incremental Moves

An Incremental Motion Axis Move (MAM) instruction can be used on the Slave axis (or Master axis if configured for Servo operation) while the position cam is operating. This is useful to accomplish phase advance/retard control. The incremental move distance can be used to eliminate any phase error between the master and the slave, or to create an exact phase relationship.

Master Offset Moves

A MAM instruction can also be used while the position cam is operating to shift the master reference position of the cam on the fly. Unlike an incremental move on the slave axis, a master offset move on the Slave axis shifts the cam profile relative to the Master axis, as shown in the figure below.



When the MAPC instruction (except pending) is initiated, the corresponding active Master Offset Move is disabled and the corresponding Master Offset,

Strobe Offset, and Start Master Offset are reset to zero. In order to achieve the master reference position shift, the MAM instruction must be initiated after the MAPC is initiated.

See the Motion Axis Move (MAM) instruction for more information on Master Offset moves.

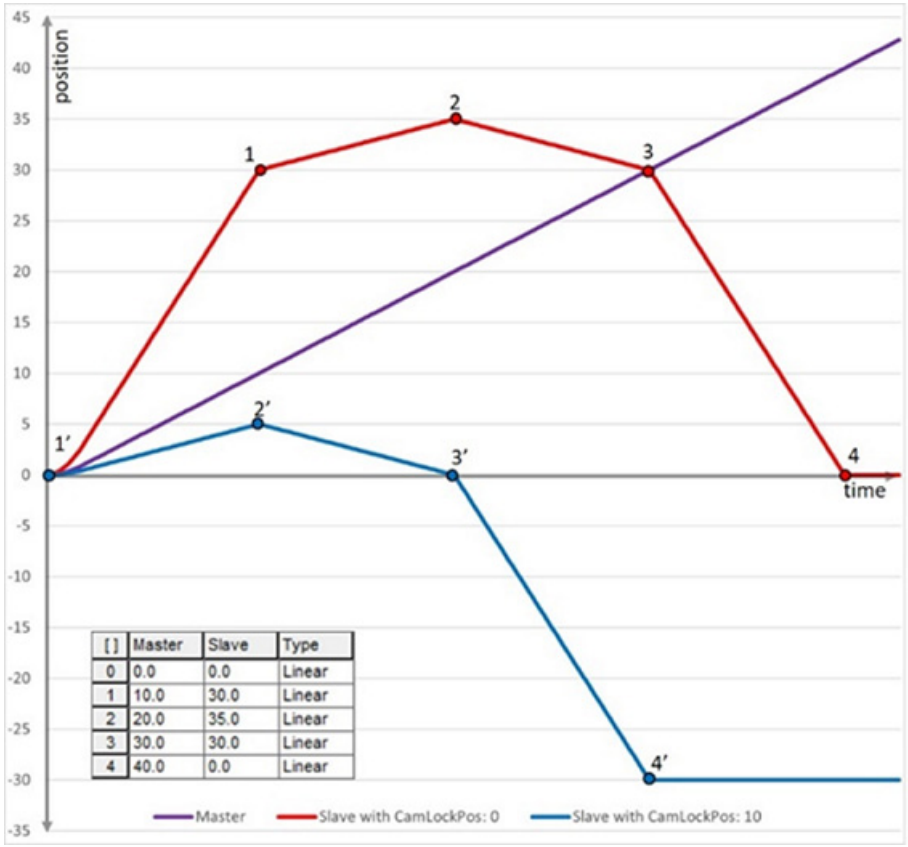
Cam Lock Position

The Cam Lock Position determines the starting location within the Cam Profile when the Slave locks to the Master. Typically, the Cam Lock Position is set to the beginning of the Cam profile. Since the starting point of most cam tables is zero, the Cam Lock Position is typically set to zero. Alternatively, the Cam Lock Position can be set to any position within the Master range of the Cam profile.

The figure illustrates a Cam profile with two different Cam Lock Positions. The red line indicates the Cam profile execution with a Cam Lock Position of zero, and the blue line indicates the same Cam profile execution with a Cam Lock Position of ten. Take notice of how the profile shifts from point 1 to point 1', point 2 to point 2', and so on.

In the figure, the Slave axis is at position of 30 units when the Master is at a position of ten units with a Cam Lock Position of zero as shown in red and indicated by 1. When the Cam Lock Position is changed from zero to ten units, the Slave axis profile is offset by 30 units at each row of the Slave axis position. Shown below is how each row's Slave value is calculated.

[ ]	Master	Slave	Slave'
0	0	0	1' = $S1$ - 30 1' = 30 - 30 = 0
1	10	30	2' = $S2$ - 30 2' = 35 - 30 = 5
2	20	35	3' = $S3$ - 30 3' = 30 - 30 = 0
3	30	30	4' = $S4$ - 30 4' = 0 - 30 = 30
4	40	0	Profile ended '4 = End of Cam Boundary



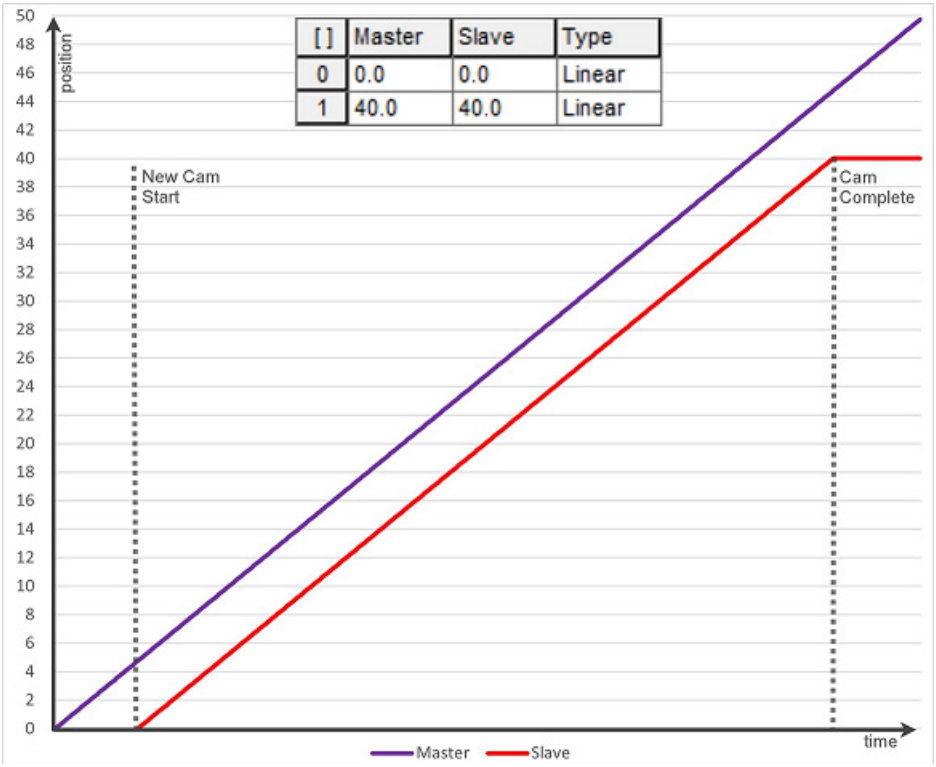
Cam Type

The Cam Type parameter indicates that the programmed cam is a new or replacement cam. The selection allows the user to choose New Cam, Replace and Restart, and Replace and Continue.

New Cam

New Cam should be used when the user wants the programmed cam to be a new cam. This is the default value for the Cam Type parameter and provides backward compatibility for the Studio 5000 Logix Designer application.

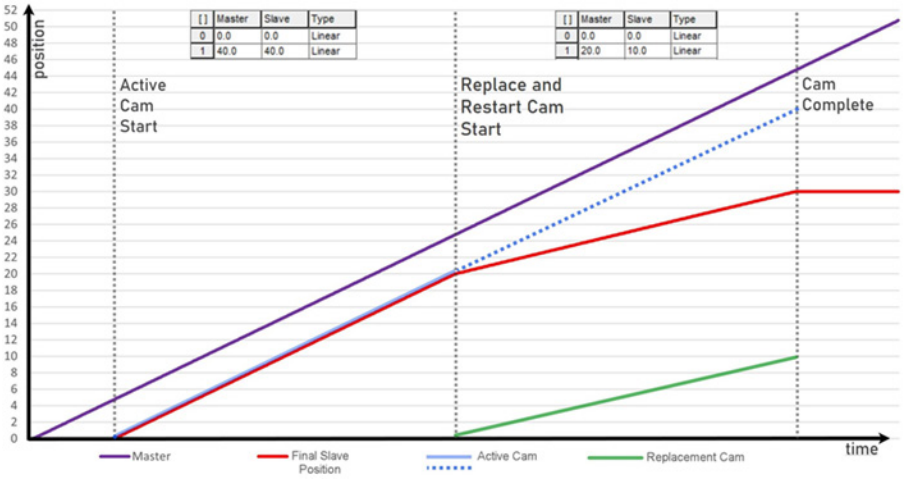
New Cam starts its cam profile at the beginning or when the Pending Cam starts after the Active Cam is complete. This figure illustrates the execution of a New Cam.



Replace and Restart

Use the enumeration Replace and Restart to replace the current running cam. Replacement Cam with Cam Type Replace and Restart starts the cam profile at the programmed Cam Lock Position. The replacement happens without any consideration of velocity, acceleration, jerk disturbances.

The user is responsible for minimizing any velocity, acceleration, and jerk disturbances through proper Cam design. This figure illustrates the execution of Replace and Restart.



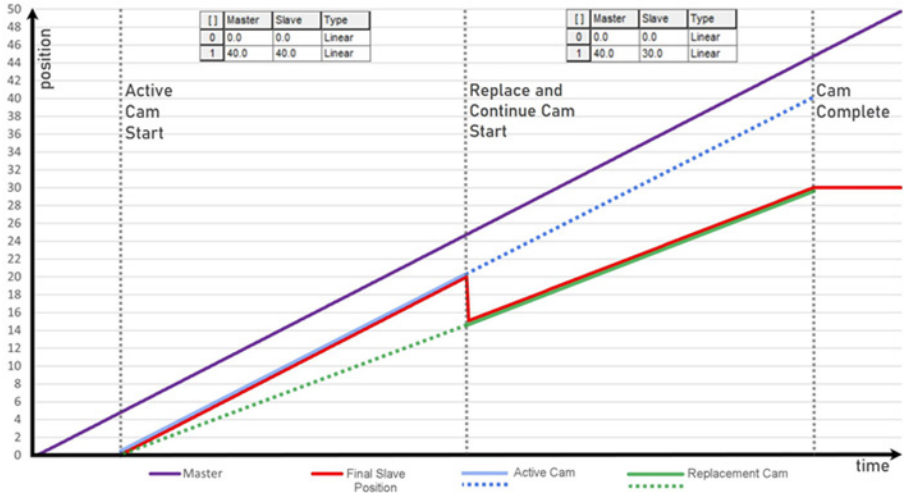
Replace and Continue

Use the enumeration Replace and Continue to replace the Active Cam at any point when it is in progress. At the replacement point, the Replacement cam starts to interpolate the Slave axis. The figure illustrates the execution of a Replace and Continue.

The green dotted line indicates the positions of the Slave axis if the Replacement Cam started at the same time as Active Cam.

The blue dotted line indicates the Slave axis positions if the Active Cam continued execution beyond the replacement point.

The sudden change in Slave Position at the replacement point is due to the Replacement Cam profile now defining the new Slave axis positions.



Stopping a Cam

Like other motion generators (jog, move, gear) active cams must be stopped by the various stop instructions, such as the Motion Axis Stop (MAS) or the Motion Group Stop (MGS). Cam motion must also stop when the ControlLogix processor changes OS modes. The MAS instruction must be able to stop the camming process. This behavior should be identical to the MAS functionality that specifically stops a gearing process.

Merging from a Cam

Like other motion generators (jog, move, gear) active cams must also be compliant with motion merge functionality. Moves and Jogs must be able to merge from active camming. This behavior should be identical to the merge functionality applied to a gearing process.

Fault Recovery

Sometimes it is necessary to respond to an axis fault condition without losing synchronization between a Master and Slave axis that are locked in a cam relationship. With an active cam, there are several ways to handle axis faults.

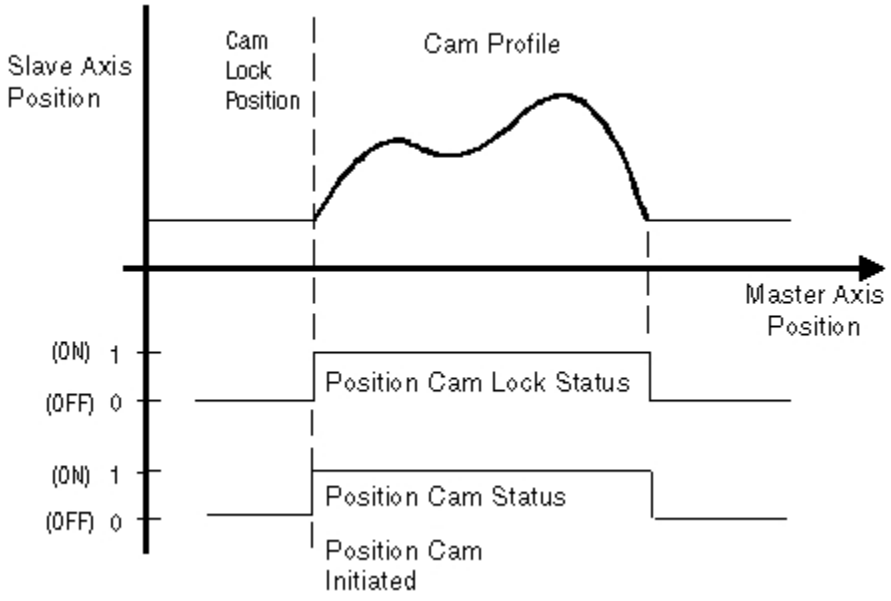
Create a virtual axis and cam everything to it and, if necessary, gear this virtual Master axis to actual Master axis of the machine. Set the various fault actions for all axes to Status Only. When an axis fault occurs (for example, a drive fault) an application program monitoring the axes fault status detects the fault and does a controlled stop of all active axes by stopping the virtual Master axis. At the profiler level, everything is still fully synchronized. Use the following error on faulted axis to determine how far it is out of position. Reset the fault on the faulted axis, bring into position at a controlled speed using the MAM instruction and the computed following error. Finally, start moving virtual Master axis.

Same configuration as above but, in this case, when the Slave axis faults the axis fault action disables the drive. This stops the active cam process on the Slave axis. At this point, the application program stops all other axes via the virtual Master axis. Next, reposition the faulted axis by determining where the Master is, and then calculating where the Slave axis should be had the fault not occurred. Finally, do an immediate lock MAPC to resynchronize with the Cam Lock Position set to the calculated value.

Important:	The MAPC instruction execution completes in a single scan, thus the Done (.DN) bit and the In Process (.IP) bit are set immediately. The In Process (.IP) bit remains set until the initiated PCAM process completes, is superseded by another MAPC instruction, ended by a Motion Axis Stop command, Merge operation, or Servo Fault Action. The Process Complete (.PC) bit is cleared immediately when the MAPC executes and sets when the cam process completes (.PC) when configured for Once Execution Mode.

This is a transitional instruction:

- In relay ladder, toggle EnableIn from False to True each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition. For more information, see Structured Text Syntax.



Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index through Arrays for array-indexing faults. A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis operand.

Execution Conditions

In Structured Text, EnableIn is always True during normal scan. Therefore, if the instruction is in the control path activated by the logic, it executes.

All conditions below the thick solid line can only occur during Normal Scan mode.

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, .IP, .AC, and .PC bits are cleared to False.
Postcan	No action taken.
Enable is False	The .EN bit is cleared to False if either the .DN or .ER bit is True.
EnableIn is True and .EN bit is False	The .EN bit is set to True and the instruction executes according to the latest version of the Motion Instruction PISD document.

EnableIn is True and .EN bit is True	No action taken.

Error Codes

See Error Codes (.ERR) for Motion instructions.

Extended Error Codes

Extended Error Codes provide additional instruction-specific information for the Error Codes that are generic to many instructions.

Extended Error Codes for Axis Not Configured (11) error code are as follows:

- Extended Error Code 1 signifies that the Slave axis is not configured.
- Extended Error Code 2 signifies that the Master axis is not configured.

Extended Error codes for the Parameter Out of Range (13) error code lists a number that refers to the number of the operand as they are listed in the faceplate from top to bottom with the first operand being counted as zero. Therefore for the MAPC instruction, an extended error code of 5 would refer to the Slave Scaling operand's value. You would then have to check your value with the accepted range of values for the instruction.

For the Error Code 54 – Maximum Deceleration Value is Zero, Click the ellipsis button next to the offending axis to access the Axis Properties screen. Go to the Dynamics tab and make the appropriate change to the Maximum Deceleration Value. If the Extended Error number is -1, that means the Coordinate System has a Maximum Deceleration Value of 0. Go to the Coordinate System Properties Dynamics Tab to correct the Maximum Deceleration value.

.SEGMENT field

The .SEGMENT field for the ILLEGAL_CAM_TYPE (28), ILLEGAL_CAM_ORDER (29) or INVALID_CAM_PROFILE_ELEMENT (179) error codes indicate the Cam Profile array element that contains, respectively, an invalid Cam Profile type (not linear or cubic), a non-ascending Master position or an invalid value (such as overflow or not a number). Therefore, INVALID_CAM_PROFILE_ELEMENT with .SEGMENT field value of 3 indicates that the fourth element (or [3]) of the Cam Profile array contains the invalid number. Use the Motion Calculate Cam Profile (MCCP) instruction or Cam Profile editor to recalculate the Cam Profile and confirm that the master and slave values do not contain an invalid value.

Status Bits

Position Cam Status

This Status bit indicates that the Motion Axis Position Cam profile is in progress. It is set when the Motion Axis Position Cam command is initiated on the Slave axis.

It is reset when these conditions are met:

- The Active or Replacement Cam completes with Execution Mode Once.
- The Slave axis is stopped (MAS, MCS) with Stop Type All or Position Cam.
- The Slave axis is shut down (MASD, MGSD, MCSD).

Bit Name	Statement	Execution Mode		
		Once	Persistent	Continuous
Position Cam Status	It is set when the Motion Axis Position Cam command is initiated on the Slave axis.	TRUE	TRUE	TRUE
	It is reset when the Active or Replacement Cam completes execution.	TRUE	FALSE	FALSE

Position Cam Lock Status

The Position Cam Lock Status bit indicates the status when the Master axis starts driving the Slave axis based on the Cam Profile and Slave Direction. It is set when the Master axis satisfies the conditions: Execution Mode, Execution Schedule, Master Direction, and Cam Lock Position.

It is reset when these conditions are met:

- The Master axis crosses the Cam boundary in Execution Mode Once and Persistent.
- The Master axis is moving in the reverse direction as specified in the Master direction.
- The Slave axis is stopped (MAS, MCS) with Stop Type All or Position Cam.
- The Slave axis is shut down (MASD, MGSD, MCSD).

Bit Name	**Statement**	**Execution Mode**		
		Once	**Persistent**	**Continuous**
Position Cam Lock Status	It is set when the Master axis starts driving the Slave axis.	TRUE	TRUE	TRUE
	It is reset when the Master axis crosses the Cam boundary.	TRUE	TRUE	FALSE

Bit Name	Statement	Execution Schedule
Position Cam Lock Status	It is set Immediately. Master Lock and Cam Lock Position are ignored.	Immediate
	It is set when the Master axis satisfies the conditions: Execution Mode, Execution Schedule, Master Direction, and Cam Lock Position.	Forward Only Reverse Only Bidirectional

The figure illustrates Replace and Restart Cam with Active Cam Unlocked. In the figure, an Active Cam is initiated with Execution Mode Persistent, and the Replacement cam is executed later with Active Cam unlocked condition. The dotted vertical lines indicate the status transitions.

(1) Active Cam Start

The Slave axis starts following the Master axis in the forward direction. The Position Cam Status and Position Cam Lock Status bits of the Slave axis' Motion Status bit are set.

(2) Active Cam getting Unlocked

Due to the Master axis crossing the Active Cam profile boundary in forward direction, the Active cam is unlocked, and the Position Cam Lock Status of the Slave axis' Motion status word bit is reset. The Position Cam Status of the Slave axis' Motion Status bit is Unchanged and remains True.

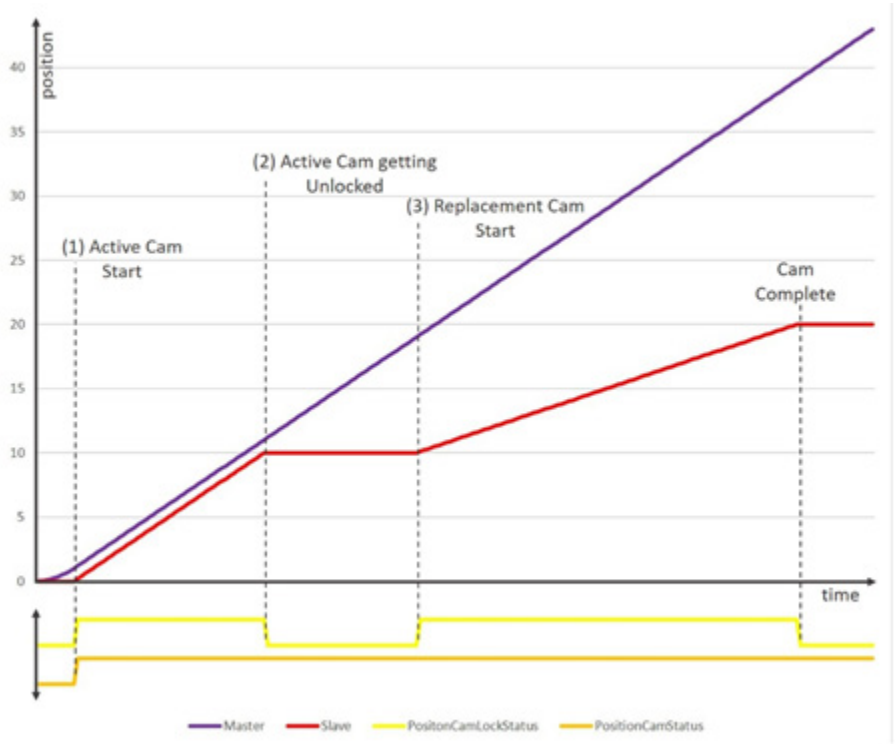
(3) Replacement Cam Start

The Replacement Cam is initiated with Active Cam still unlocked, and with these configurations:

- Execution Schedule: Immediate
- Master Direction: Forward Only
- Execution Mode: Persistent (PC bit does not turn to True after Cam Complete)

The Position Cam Lock Status of the Slave axis' Motion status word bit is set, and the Position Cam Status of the Slave axis' Motion Status bit is Unchanged and remains True.

The Slave axis starts moving according to the Replacement Cam profile until the Master axis crosses the Cam boundary. The Position Cam Lock Status of the Slave axis' Motion status word bit is reset, and the Position Cam Status of the Slave axis' Motion Status bit is Unchanged and remains True.



Position Cam Pending Status

This Status bit indicates that the Motion Axis Position Cam profile is pending the completion of an executing cam profile.

It is set when these conditions are met:

- The Motion Axis Position Cam with Execution Schedule Pending initiates.
- Replacement Cam initiated with Execution Schedule of Forward Only, Reverse Only, or Bidirectional.

It is reset when these conditions are met:

- The current Position Cam profile completes, which starts the pending Cam profile.
- Replacement Cam satisfies the conditions of Execution Schedule and Master Lock Position.
- The Slave axis is stopped (MAS, MCS) with Stop Type All or Position Cam.
- The Slave axis is shut down (MASD, MGSD, MCSD).

Bit Name	Statement	Execution Schedule
Position Cam Pending Status	FALSE	Immediate
	TRUE	Pending
	It is set until the Master Axis satisfies the condition based on Execution Mode, Execution Schedule, Master Direction, and Cam Lock Position	Replacement Cam with: Forward Only Reverse Only Bidirectional

The figure illustrates Replace and Restart Cam execution with Active Cam Locked. In the figure, an Active Cam is initiated with Execution Mode Persistent, and the Replacement cam is executed later with Active Cam locked condition. The status transitions are indicated by the dotted vertical lines.

(1) Active Cam Start

The Slave axis starts following the Master axis in the forward direction. The Position Cam Status and Position Cam Lock Status bits of the Slave axis' Motion Status bit are set. The Position Cam Pending Status bit of the Slave axis' Motion Status bit is reset.

(2) Replacement Cam Start

The Replacement Cam is initiated with the Active Cam still locked, and with these configurations:

- Execution Schedule: Forward Only
- Master Direction: Forward Only
- Execution Mode: Persistent (PC bit does not turn to True after cam Complete)
- Programmed Master Lock Position: 30 Units

The Position Cam Pending Status of the Slave axis' Motion status word bit is set. The Position Cam Status and the Position Cam Lock Status of the Slave axis' Motion Status bit are Unchanged and remains True.

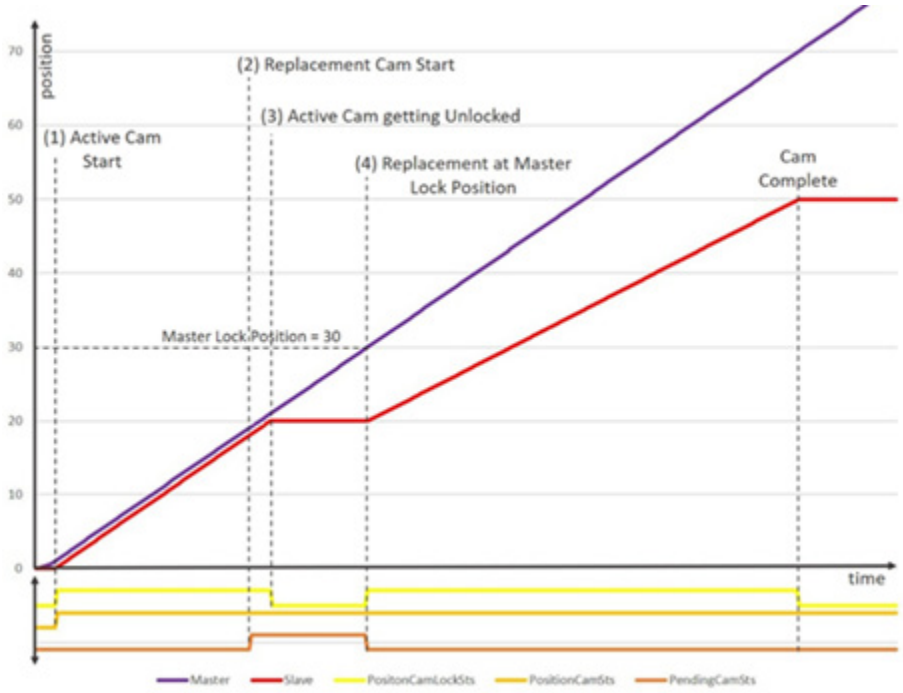
(3) Active Cam getting Unlocked

Due to the Master axis crossing the Active Cam profile boundary in forward direction, the Active cam is unlocked, and the Position Cam Lock Status of the Slave axis' Motion status word bit is reset. The Position Cam Status of the Slave axis' Motion Status bit is Unchanged and remains True.

(4) Replacement at Master Lock Position

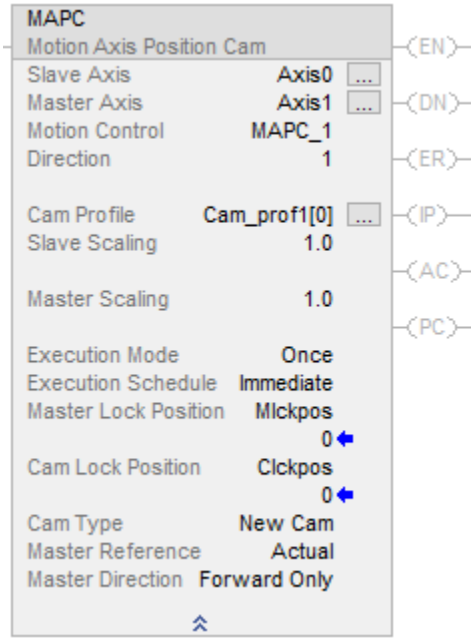
When the Master axis starts moving in forward direction and reaches Master Lock Position of 30 units, the Position Cam Lock Status of the Slave axis' Motion status word bit is set, and the Position Cam Pending Status of the Slave axis' Motion status word bit is reset. The Position Cam Status of the Slave axis' Motion Status bit is Unchanged and remains True.

The Slave axis starts moving according to the Replacement Cam profile until the Master axis crosses the cam boundary. The Position Cam Lock Status of the Slave axis' Motion status word bit is reset.



Example

Relay Ladder Logic



Structured Text

MAPC (Axis0, Axis1, MAPC_1, 1, Cam_prof1[0], 1.0, 1.0, Once, Immediate, Mlckpos, Clckpos, NewCam, Actual, ForwardOnly);

See also

Structured Text Syntax on page 665

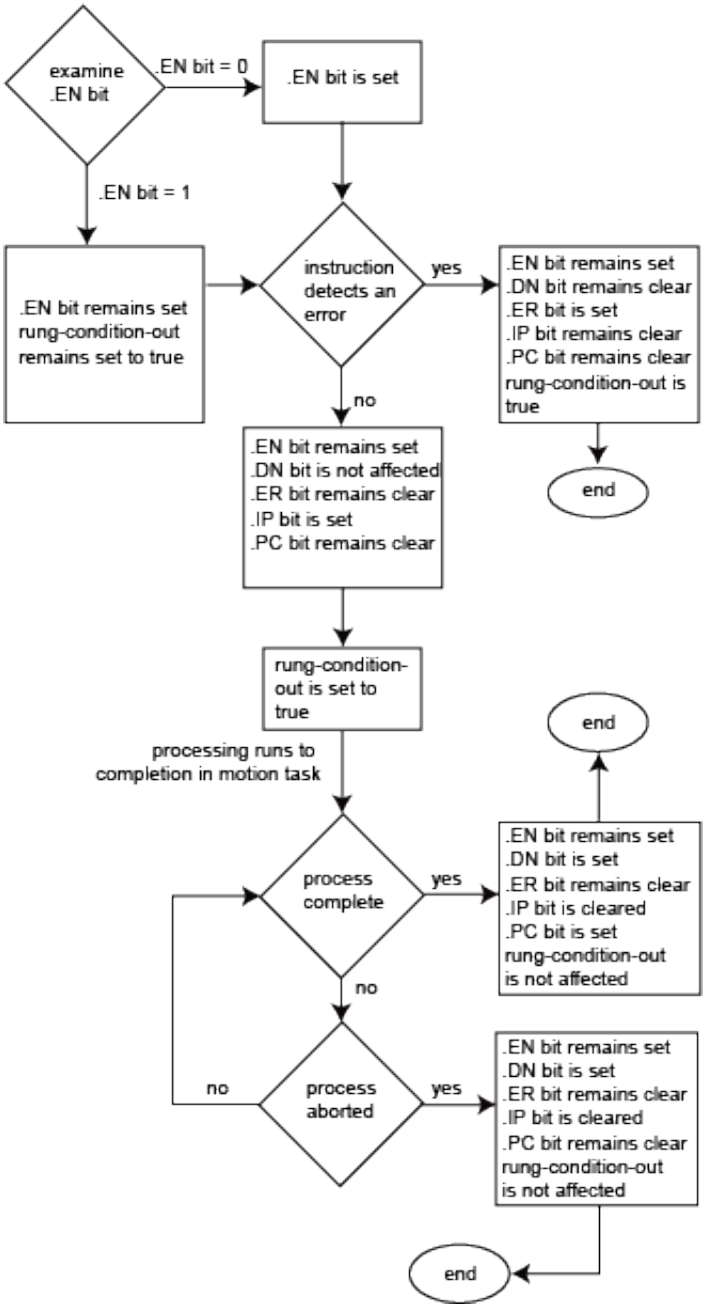
MAPC Flow Chart (True) on page 192

Motion Error Codes (.ERR) on page 581

Motion Move Instructions on page 73

Common Attributes on page 691

MAPC Flow Chart (True)



Motion Axis Time Cam (MATC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The Motion Axis Time Cam (MATC) instruction provides electronic camming of an axis as a function of time or between any two axes in the Master Driven mode.

When executed, the specified axis is synchronized with time or to the Master axis using a Cam Profile established by the Logix Designer Cam Profile Editor, or Motion Calculated Cam Profile (MCCP) instruction.

The instruction also provides an easy means to cancel a running time cam anywhere during the cam execution and replace it with a different profile. The

replacement is done immediately or scheduled at a specific position in Master Driven mode.

The MATC instruction can be configured for either Time Driven or

- Time Driven Mode: When executed, the Slave axis is synchronized with time by using the Cam Profile. Only immediate cam replacement is possible for Time Driven Mode.
- Master Driven Mode: When executed, the Slave axis is synchronized to the Master axis by using the Cam Profile. MDAC instruction is used to set a Master: Slave relationship for Master Driven Mode. Refer to the MDAC Instruction for further reference.

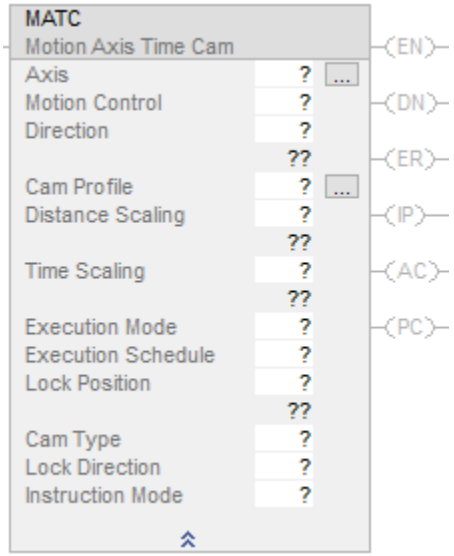
This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MATC(Axis, MotionControl, Direction, CamProfile, DistanceScaling, TimeScaling, ExecutionMode, ExecutionSchedule, LockPosition, CamType, LockDirection, InstructionMode);

Operands

Ladder Diagram

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL	AXIS_CIP_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE AXIS_GENERIC AXIS_GENERIC_DRIVE AXIS_VIRTUAL	Tag	The type of the axis to which the cam profile applies. Ellipsis launches the Axis Properties dialog. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Structure used to access block status parameters.

Direction	DINT	DINT	Immediate	Relative direction of the Slave axis to the aster axis: 0 = Same – The Slave axis position values are in the same sense as the time or the Master axis position values. 1 = Opposite – The Slave axis position values are opposite sense of the time or the Master axis position value. 2 = Reverse – The current or previous direction of the cam is reversed on execution. When executed for the first time with Reverse selected, the control defaults the direction to Opposite.. 3 = Unchanged – this allows other cam parameters to be changed without altering the current or previous camming direction. When executed for the first time with Unchanged selected, the control defaults the direction to Same.
Cam Profile	CAM_PROFILE CAM_PROFILE_EXTENDED **Tip:** CAM_PROFILE_EXTENDED is supported by Compact GuardLogix 5580, CompactLogix 5380, and CompactLogix 5480 controllers only.	CAM_PROFILE CAM_PROFILE_EXTENDED	Array	Tag name of the calculated Cam Profile array that is used to establish the Time/Slave or Master/Slave position relationship. Only the zero array element ([0]) is allowed for the Cam Profile array. Ellipsis launches Cam Profile Editor. Use the CAM_PROFILE_EXTENDED type for this operand to enable access to double precision (64-bit LREAL) cam data members.
Distance Scaling	REAL	REAL	Immediate or Tag	Scales the total distance covered by the Slave axis through the Cam profile.
Time Scaling	REAL	REAL	Immediate or Tag	When the Instruction Mode is Time Driven, scales the time interval covered by the Cam Profile. When the Instruction Mode is Master Driven, scales the total distance covered by the Master axis through the Cam Profile.

Execution Mode	DINT	DINT	Immediate	Determines if the Cam profile is executed only one time or repeatedly. 0 = Once – Once started, the Slave axis follows the Cam profile until the Cam boundary is crossed. On crossing of the Cam boundary, Cam motion on the Slave axis stops and the Process Complete bit is set. The Time Cam Status bit in Slave axis' Motion Status word is reset. The Slave motion does not resume if the time or Master axis moves back into the Cam profile range. 1 = Continuous – Once started the Cam profile is executed indefinitely. This feature is useful in rotary applications where it is necessary that the Cam position run continuously in a rotary or reciprocating fashion.
Execution Schedule	DINT	DINT	Immediate	Selects the method for executing the Cam profile. 0 = Immediate – The Slave axis is immediately locked to the time, or the Master axis and the camming process begins. 1 = Pending – Lets user blend a new Time cam execution after an in-process time cam is finished. When Pending is selected, the Lock Position is ignored.
Lock Position	REAL	REAL	Immediate	When the Instruction Mode is Master Driven, the Lock Position is the Master axis absolute position where the Slave axis locks to the Master axis and starts following the Master axis. If Execution Schedule is Pending or Immediate, Lock Position is ignored. If the Instruction Mode is Time Driven, Lock Position is ignored.

Cam Type	DINT	DINT	Immediate	0 = New Cam – New Cam should be used when the user wants the programmed cam to be a completely new cam. This is the default value for the Cam Type parameter and provides backward compatibility for the Studio 5000 Logix Designer application. 1 = Replace and Restart – The Replacement Cam replaces the Active Cam. The replacement happens without any consideration of velocity, acceleration, or jerk disturbances. 2 = Replace and Continue – The Replacement Cam replaces the Active Cam at any point, when it is in progress. At the replacement point the Replacement Cam starts to interpolate the Slave axis. This operand is new with version 34 of Logix Designer.
Lock Direction	DINT	DINT	Immediate Real or Tag	When the Instruction Mode is Master Driven, this determines the direction of the Master axis that generates Slave motion according to the Cam profile. 0 = None Note: If the Instruction mode is Time Driven, then Lock Direction should be None. 1 = Immediate Forward Only – The Slave axis is immediately locked when the Master axis is moving in the forward direction. 2 = Immediate Reverse Only - The Slave axis is immediately locked when the Master axis is moving in the reverse direction. 3 = Position Forward Only - The Cam profile starts when the Master axis position crosses the Lock Position in the forward direction. 4 = Position Reverse Only - The Cam profile starts when the Master axis position crosses the Lock Position in the reverse direction.

Instruction Mode	DINT	DINT	Immediate	Specifies if a MATC should be executed in: 0 = Time Driven Mode - When executed, the Slave axis is synchronized with time by using the Cam profile. 1 = Master Driven Mode - When executed, the Slave axis is synchronized to the Master axis by using the Cam profile

Structured Text

MATC (Axis, MotionControl, Direction, CamProfile, DistanceScaling, TimeScaling, ExecutionMode, ExecutionSchedule, LockPosition, CamType, LockDirection, InstructionMode);

The operands are the same as the operands for the relay ladder MATC instruction. For the array operands, you do not have to include the array index. If you do not include the array index, the instruction starts with the first element in the array ([0]).

For the array operands, you do not have to include the array index. If you do not include the index, the instruction starts with the first element in the array ([0]).
See Structured Text Syntax for more information on the syntax of expressions within structured text.

For the operands that require you to select from available options, enter your selection as:

This Operand	Has These Options Which You	
	Enter as Text	Or Enter as a Number
ExecutionMode	Once Continuous	0 1
ExecutionSchedule	Immediate Pending	0 1
CamType	New Cam Replace and Restart Replace and Continue	0 1 2
LockDirection	None Immediate Forward Only Immediate Reverse Only Position Forward Only Position Reverse Only	0 1 2 3 4
Instruction Mode	Time Driven Mode Master Driven Mode	0 1

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set when the rung makes a False-to-True transition and remains set until the rung goes False.
.DN (Done) Bit 29	It is set when the time cam is initiated.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.IP (In Process) Bit 26	It is set on positive rung transition and cleared if either superseded by another Motion Axis Time Cam command, or cancelled by a stop command, merge, shut down, or servo fault.
.AC (Active) Bit 23 **Tip:** Supported only on CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers. For other controllers, the .AC bit appears on the instruction faceplate but is inactive and is always equal to False.	It is set when the cam begins interpolation of the Slave axis. It is reset when the Active cam execution completes or cancelled by a stop command, merge, shut down, or servo fault.
.PC (Process Complete) Bit 27	It is cleared on positive rung transition and set in ONCE Execution Mode, when the time in Time Driven Mode or position of the Master axis in Master Driven mode leaves the range defined by the active Cam profile.

Tip: Version 34 and newer: The information in the MOTION_INSTRUCTION Structure Table pertaining to the .AC bit applies to the Compact GuardLogix (5380), CompactLogix (5380), CompactLogix (5480), ControlLogix (5580), GuardLogix (5580) and Logix Emulate series of controllers only. For other controllers, the .AC bit is shown on the instruction faceplate but is inactive and is always equal to False.

Description

The MATC instruction executes a Time Cam Profile set up by using the Motion Calculated Cam Profile (MCCP) instruction or by the Studio 5000 Logix Designer application Cam Profile Editor. No maximum velocity, acceleration, or deceleration limits are used in this instruction. The designated Cam profile derived from the associated cam table determines the speed, acceleration, and deceleration of the Slave axis.

WARNING: The maximum velocity, acceleration, or deceleration limits established during axis configuration do not apply to electronic camming.

The Direction input parameter defines the direction of the Slave axis motion relative to the Master axis. As applied to the Slave axis, the camming direction can be explicitly set as the Same or Opposite or set relative to the current camming direction as Reverse or Unchanged.

The user can use the Distance and Time Scaling functionality to scale Slave motion based on a standard Cam Profile without creating a cam table and calculating a new Cam profile.

The Cam Profile can be executed Once or in Continuous mode by specifying the desired Execution Mode.

The user can also configure the Cam Profile to execute Immediately or Pending completion of a currently running Cam Profile via the Execution Schedule parameter.

To accurately synchronize the Slave axis position to the Master axis position, the user can specify an Execution Schedule setting and an associated Lock Position for the Master axis while the Instruction Mode is Master Driven. If executing in Time-Driven mode, the Lock Position is ignored.

The Cam Type selection allows the user to cancel a running cam and replace it with another cam immediately or schedule it to perform at a specific position in Master Driven mode.

Instruction Mode allows configuring cam execution to either Master Driven or Time Driven mode.

Camming Direction

Cams can be configured to add or subtract their incremental contribution to the axis command position. Control over this behavior is via the Direction parameter.

Camming in the Same Direction

When Same is selected or entered as the Direction for the MATC instruction, the Slave axis position values computed from the Cam Profile are added to the command position of the Slave axis. This is the most common operation, as the profile position values are used as entered in the original cam table. That is, consecutive-increasing profile values result in axis motion in the positive direction and vice versa.

Camming in the Opposite Direction

When Opposite is selected or entered as the Direction, the Slave axis position values computed from the Cam Profile are subtracted from the command position of the Slave axis. Thus, axis motion is in the opposite direction from that implied by the original cam table. In other words, consecutive increasing profile values results in axis motion in the negative direction and vice versa.

Preserving the Current Camming Direction

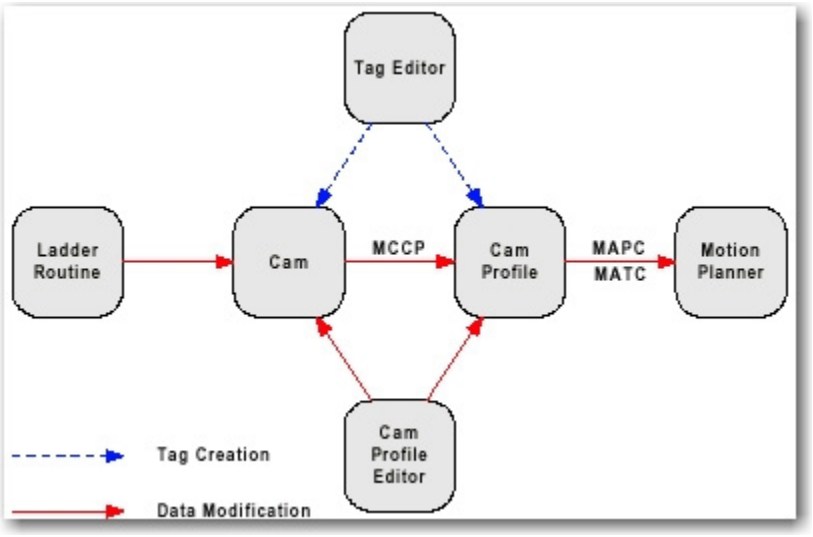
When Unchanged is selected or entered as the Direction, other position cam parameters can be changed while preserving the current or previous camming direction (same or opposite). This is useful when the current direction is not known or not important. For first-time execution of a cam with Unchanged selected, the control defaults the direction to Same.

Reversing the Current Camming Direction

When Reverse is selected, the current or previous direction of the position cam is changed from Same to Opposite or from Opposite to Same. For first-time execution of a cam with Reverse selected, the control defaults the direction to Opposite.

Specifying the Cam Profile

To execute a MATC instruction, a calculated Cam Profile data array tag must be specified. Cam Profile array tags can be created by the Logix Designer tag editor or the MATC instruction by using the built-in Cam Profile Editor, or by executing an Motion Calculate Cam Profile (MCCP) instruction on an existing Cam array.



The data within the Cam Profile array can be modified at compile time by using the Cam Profile Editor, or at runtime with the Motion Calculate Cam Profile (MCCP) instruction. In the case of runtime changes, create a cam array in order to use the MCCP instruction. Refer to the MCCP instruction specification for more detail on converting Cam arrays.

MATC supports two types of Cam Profile array: CAM_PROFILE and CAM_PROFILE_EXTENDED. CAM_PROFILE is calculated from a CAM array. CAM_PROFILE_EXTENDED is calculated from a CAM_EXTENDED array, which provides better precision.

For CAM_PROFILE array, all but the status and type elements of the CAM_PROFILE array element structure are hidden in the Studio 5000 Logix Designer tag editor. Use the status parameter to indicate that the CAM_PROFILE array element has been calculated. If a camming instruction begins to execute with uncalculated elements in a CAM_PROFILE, an error occurs. The type parameter determines the type of interpolation applied between this cam array element and the next cam element (for example, linear or cubic).

For CAM_PROFILE_EXTENDED array, the status, master and slave values, type, and C0, C1, C2, C3 coefficients are visible in the Studio 5000 Logix Designer tag editor. Use the status parameter to indicate that the CAM_PROFILE array element has been calculated. If a camming instruction begins to execute with uncalculated elements in a CAM_PROFILE, an error occurs. The master and slave define the x and y value of the cam element. The type parameter determines the type of interpolation applied between this cam array element and the next cam element (for example, linear or cubic). The C0, C1, C2, and C3 are coefficients that define the shape between two cam elements.

WARNING: Do not modify the Cam Profile array directly. Modifying the Cam Profile array can cause unintended motion or a motion fault.
Always use Motion Calculate Cam Profile (MCCP) or the Cam Profile editor to adjust the Cam Profile array.

Cam Profile Array Checks

The Status member of the first element in the Cam Profile array is special and used for data integrity checks. For this reason, the MATC must always specify the cam profile with the starting index set to 0. This first cam profile element Status member can have the following values:

Status Variables	Description
0	Cam Profile element has not been calculated.
1	Cam Profile element is being calculated.
2	Cam Profile element has been calculated
n	Cam Profile element has been calculated and is currently being used by (n-2) MAPC or MATC instructions.

Before starting a cam on a specified axis, the MATC instructions checks if the Cam Profile array has been calculated by checking the value of the first cam profile element's Status member. If Status is 0 or 1 then the cam profile has not been calculated yet and the MATC instruction errors. If the Cam Profile array has been completely calculated (Status > 1), the instruction then increments the Status member indicating that it is in use by this axis.

When the cam completes, or terminates, the Status member of the first cam profile array element is decremented to maintain track of the number of cams actively by using the associated cam profile.

Linear and Cubic Interpolation

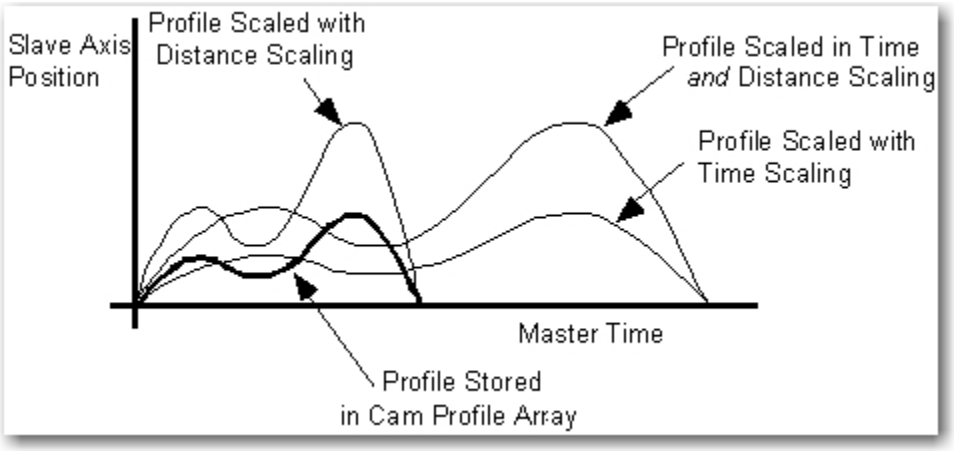
Time cams are fully interpolated. This means that if the current master time value does not correspond exactly with a point in the cam table associated with the cam profile, the Slave axis position is determined by linear or cubic interpolation between the adjacent points. In this way, the smoothest possible slave motion is provided.

Each point in the Cam array that was used to generate the Cam Profile can be configured for linear or cubic interpolation.

Electronic camming remains active through any subsequent execution of jog, or move processes for the slave axis. This allows electronic camming motions to be superimposed with jog, or move profiles to create complex motion and synchronization.

Scaling Time Cams

A time cam profile can be scaled in both time and distance when it is executed. This scaling is useful because it allows the stored cam profile to be used only for the form of the motion with the scaling used to define the time or distance over which the profile is executed.



When MATC instruction specifies a Cam Profile array, the master coordinate values defined by the Cam Profile array take on the time units (seconds) and the slave values take on the units of the Slave axis. By contrast, the Time and Distance Scaling parameters are "unit less" values that are used as multipliers to the Cam profile.

By default, both the Time and Distance Scaling parameters are set to 1. To scale a time Cam profile, enter a Time Scaling or Distance Scaling value other than 1.

Increasing the Time Scaling value of a Cam profile decreases the velocities and accelerations of the profile, while increasing the Distance Scaling value increases the velocities and accelerations of the profile. To maintain the velocities and accelerations of the scaled profile approximately equal to those

of the unscaled profile, the Time Scaling and Distance Scaling values should be equal. For example, if the Distance Scaling value of a profile is 2, the Time Scaling value should also be 2 to maintain approximately equal velocities and accelerations during execution of the scaled time cam.

WARNING: Decreasing the Master Scaling value or increasing the Slave Scaling value of a position cam increases the required velocities and accelerations of the profile. This can cause a motion fault if the capabilities of the drive system are exceeded.

Execution Modes

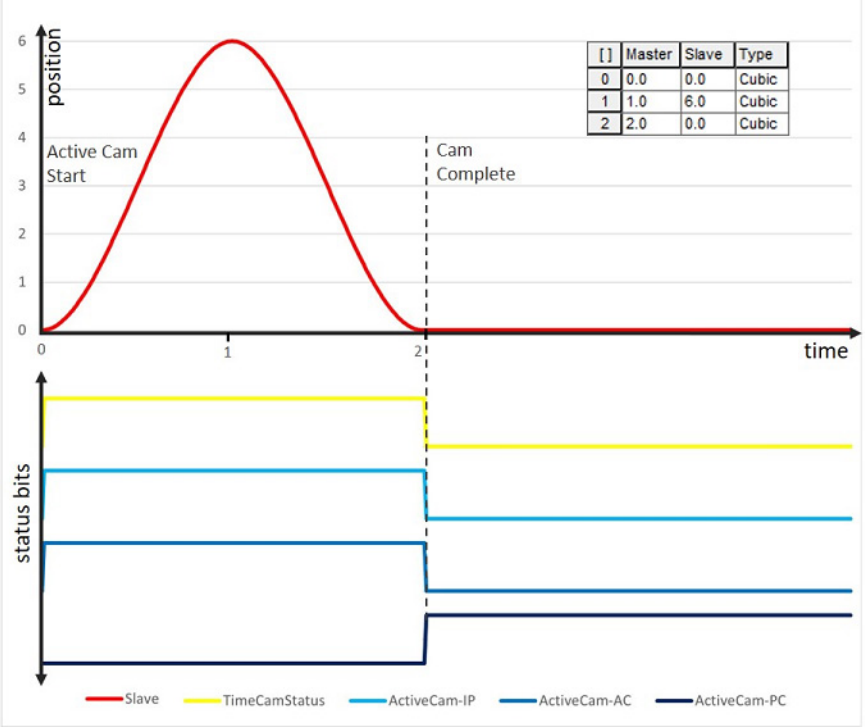
The New Cam and Replacement Cam can have an Execution Mode of Once or Continuous. Execution Mode can be selected to determine how the cam motion behaves when the time moves beyond the end point of the Cam Profile defined by the original cam table.

Once

When Once is selected (default), the specified cam profile, once started, executes till the Cam boundary is crossed. When the Master axis moves outside the range of the profile, cam motion on the Slave axis stops and the Process Complete (. PC) bit of MATC Instruction is set and the Time Cam Status bit in Slave axis' Motion Status word is reset.

The diagram illustrates the execution of New Cam in Time Driven Mode with Execution Schedule of Immediate and Execution Mode of Once.

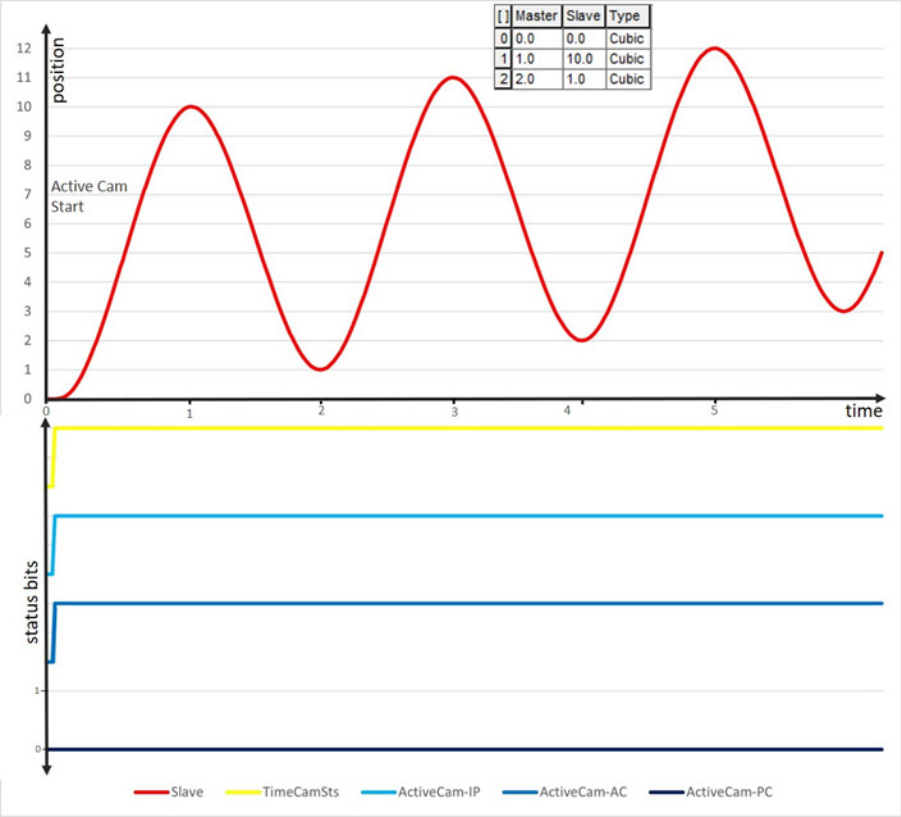
When the Active Cam starts, the .IP and .AC bits are set. The Time Cam Status bit in Slave axis' Motion Status word is also set. When the time moves beyond the range of the Cam profile, the Active Cam .PC bit gets set and the .IP and .AC bits are reset. The Time Cam Status bit in Slave axis' Motion Status word is reset.



Continuous

When Continuous Mode is selected, the Cam profile is executed indefinitely. With continuous operation, the slave positions are reset when the time moves beyond the end of the Cam profile causing the Cam profile to repeat indefinitely. This feature is useful in rotary applications where it is necessary that the time cam run continuously in a rotary or reciprocating fashion. To generate smooth continuous motion by using this technique, however, care must be taken in designing the cam points of the cam table so that there are no position, velocity, or acceleration discontinuities between the start and end points of the calculated Cam profile.

The diagram illustrates the execution of New Cam in Time Driven Mode with Execution Schedule of Immediate and Execution Mode of Continuous. At time value of 2, the Active Cam profile crosses the Cam boundary and the second cycle of the Cam profile starts, at time value of 4 the second cycle of Cam profile completes and the third cycle starts, this goes on indefinitely.



Execution Schedule

The Execution Schedule parameter controls the MATC instruction's execution schedule. An Execution Schedule of Immediate (0) and Pending (1) is used in both Master Driven Mode and Time Driven Mode.

Immediate

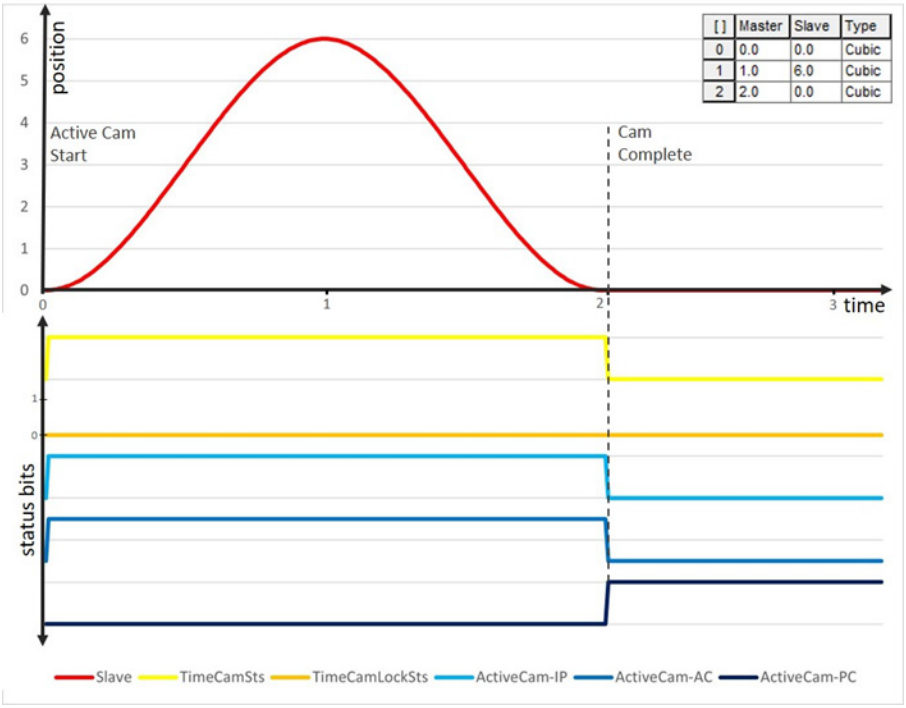
By default, the execution schedule of MATC instruction is Immediate. In Immediate execution, the Slave axis is immediately locked to the time master coordinate, according to the specified Cam Profile. there is no delay in enabling of the time camming process.

Active Cam

In the Immediate Active Cam case, there is no delay in execution of the time cam. In this case, the Lock Position parameter is irrelevant.

The diagram illustrates Immediate Active Cam initiated in Time Driven Mode and Execution Mode of Once. When the camming process is initiated, the

Slave axis synchronized with the time and follows the Cam Profile. The Time Cam Status bit is set at the beginning, and Time Cam Lock Status of the Slave axis' Motion status word remains unchanged as it is unused in Time Driven mode. The Active Cam's .PC bit is set when time moves beyond the range of the Cam Profile. The Time cam Status of the Slave axis' Motion status word bit is reset then.

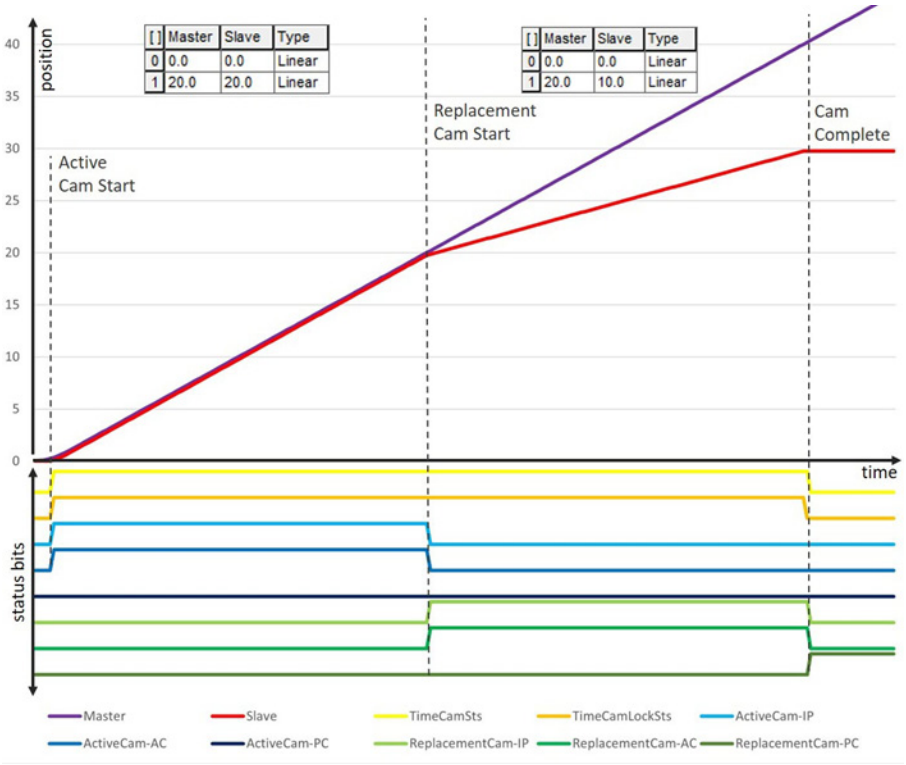


Replace and Restart Cam

In the Immediate Replace and Restart cam case, the replacement cam replaces the Active Cam immediately.

The diagram illustrates Immediate Replace and Restart cam execution in Master Driven Mode, Execution Mode of Once and Lock Direction of Immediate Forward Only. The Active cam initiates the camming process, and the Replacement cam is initiated immediately.

The Time Cam Status and Time Cam Lock Status of the Slave axis' Motion status word is set. The replacement Cam's .PC bit is set when time moves beyond the range of the Replacement Cam Profile.



Replace and Continue Cam

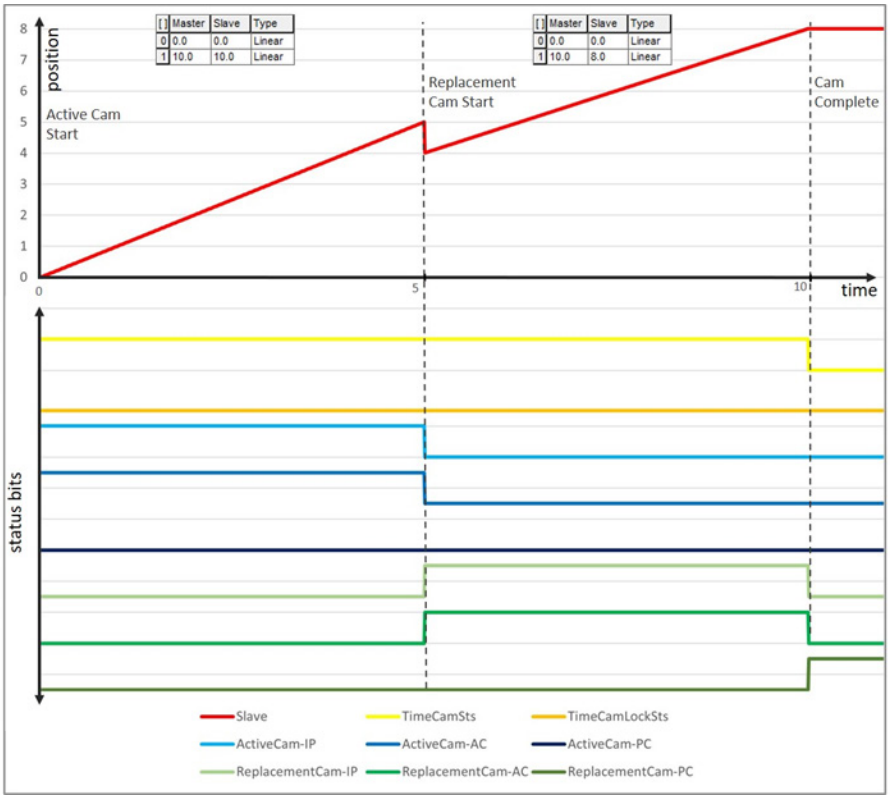
In the Immediate Replace and Continue Cam case, the Replacement Cam replaces the Active Cam immediately at the point at which the replacement was initiated.

The diagram illustrates Immediate Replace and Continue cam execution with this configuration:

- Slave Direction: Same
- Execution Mode: Once
- Execution Schedule: Immediate
- Lock Direction: None
- Lock Position: 0
- Instruction Mode: Time Driven Mode

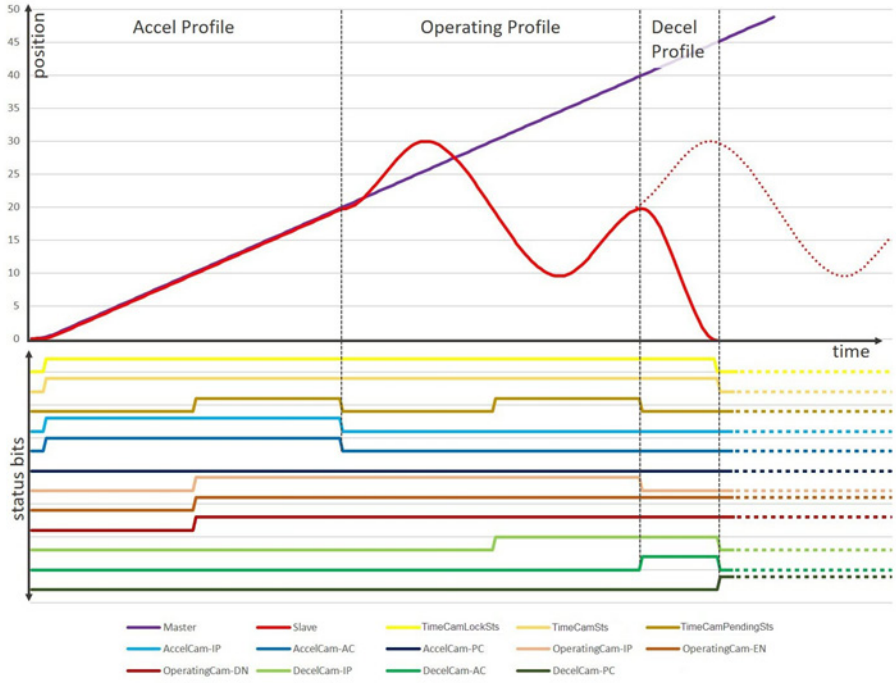
The Active cam initiates the camming process, and the Replacement cam replaces the Active Cam immediately. The Slave axis is synchronized with the Time and follows the Replacement Cam Profile.

The Time Cam Status of the Slave axis' Motion status word is set. The replacement Cam's .PC bit is set when time moves beyond the range of the Replacement Cam Profile.



Pending Cam Execution: Alternatively, the MATC instruction's execution can be deferred pending completion of an executing Time cam. An Execution Schedule selection of Pending is blends two Time Cam profiles without stopping motion.

The Pending execution feature is useful in applications like high-speed packaging when a Slave axis must be locked into a Master axis and accelerated by using a specific profile to the proper velocity. When this acceleration profile is done, it must blend into the operating profile, which is typically executed continuously. To stop the Slave axis, the operating profile is blended into a deceleration profile such that the axis stops at a known location as shown in the diagram.



To confirm smooth motion across the transition, the profiles must be designed such that no position, velocity, or acceleration discontinuities exist between the end of the current profile and the start of the new one. This is done by using the Studio 5000 Logix Designer application Cam Profile Editor.

If an Execution Schedule of Pending is selected without a corresponding Active Cam profile in process, the MATC instruction executes but no camming motion occurs until another MATC instruction with a non-pending Execution Schedule is initiated. This allows pending Cam Profiles to be preloaded prior to executing the initial cam. This method addresses cases where immediate cams would finish before the pending cam could be reliably loaded.

Replacement Cam effect on a Pending Cam :This diagram shows an example of how Pending Cam is affected when a Replacement Cam starts. The status transitions are indicated by the dotted vertical lines in the diagram. When the Replacement Cam executes, the Pending Cam's .IP bit is set and the Replace and Restart Cam Profile runs.

(1) Active Cam starts in Time Driven Mode with Lock Direction of Immediate Forward only.

(2) Pending Cam is enabled and waiting for the completion of the Active Cam. The Pending Cam's. IP and .DN bits are set.

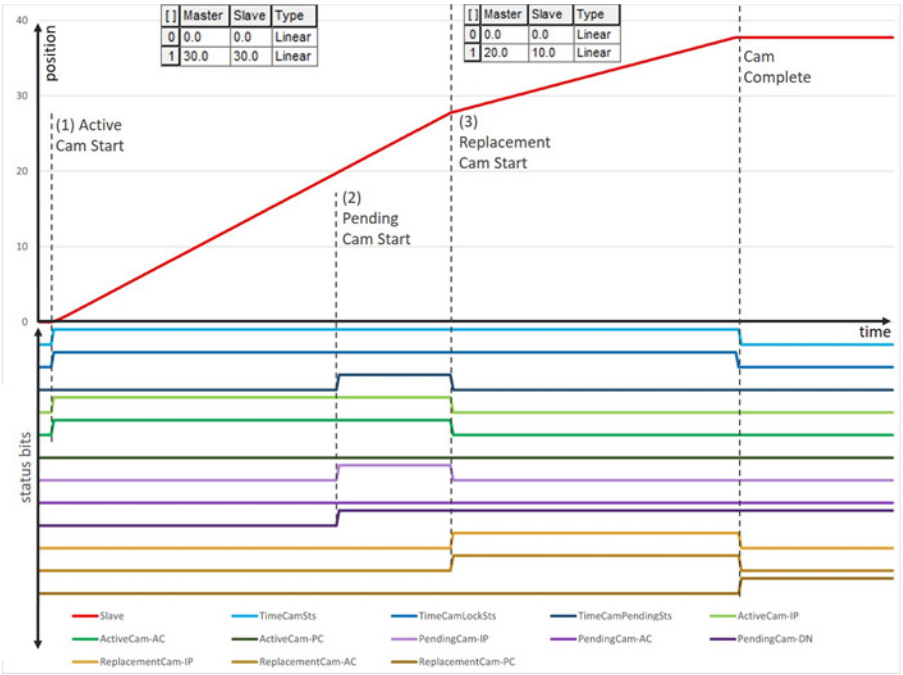
(3) Replacement Cam executes in Time Driven mode before the end of the Active Cam profile completes. The Pending Cam's .IP bit resets and no change in .AC bit.

(4) Cam Complete. The Replacement Cam's .PC bit is set.

If the Cam Type is Replace and Restart or Replace and Continue with the Execution Schedule of Pending, then the MATC instruction's .ER flag is set.

Both the Pending Cam and an Active Cam must be in the same instruction mode - either both must be in Time Driven Mode, or both must be in Master Driven Mode. An instruction that pends a Cam to an Active Cam in a different mode causes the Active Cam's .ER to set.

All pending Cams must use the Immediate Forward or the Immediate Reverse Lock Direction. A pending Cam with a specified Lock Position with a Position Forward Only or Position Reverse Only Lock Direction causes a LOCK_DIRECTION_CONFLICT (95) error. (See MDSC Error Codes for a list of runtime error codes.) A second pending Cam overlays the first pending Cam in memory.



Cam Type

The Cam Type parameter indicates that the programmed cam is a new or replacement cam.

The selection allows the user to choose:

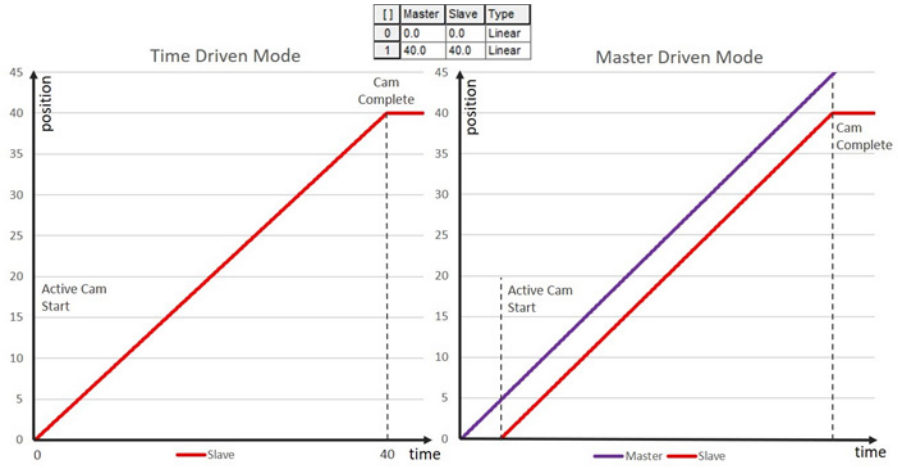
- New Cam
- Replace and Restart
- Replace and Continue

New Cam

New Cam should be used when the user wants the programmed cam to be a completely new cam. This is the default value for the Cam Type parameter and provides backward compatibility for the Studio 5000 Logix Designer application.

New Cam starts its cam profile at the beginning or when the Pending Cam starts after the Active Cam is complete.

The diagram illustrates the execution of a New Cam. The left side shows the execution in Time Driven Mode, where the Slave axis is synchronized with time using cam profile. The right side shows the execution of the same profile in Master Driven Mode where the Slave axis is locked to the Master axis.



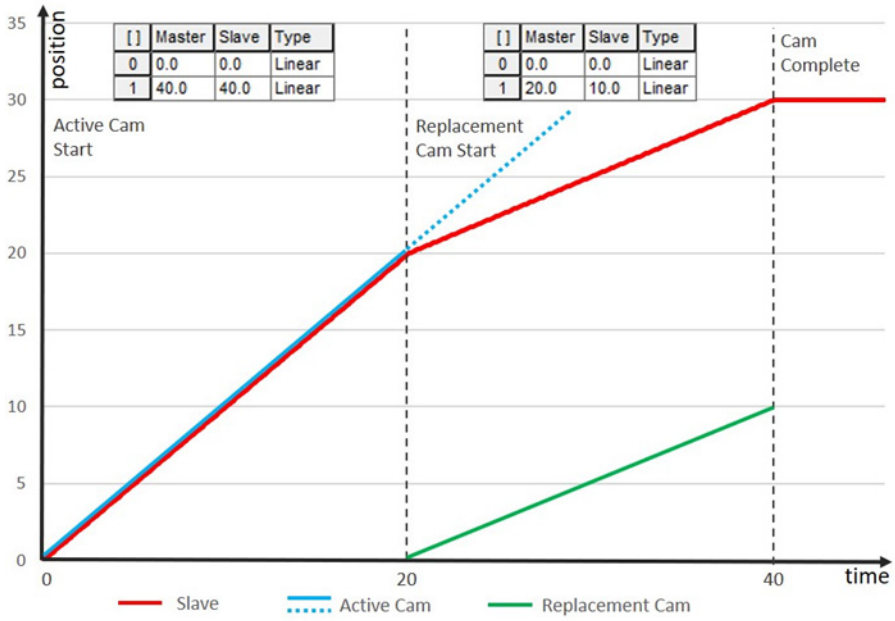
Replace and Restart

Use the enumeration Replace and Restart to replace the current running cam. The replacement happens without any consideration of velocity, acceleration, or jerk disturbances.

The user is responsible for minimizing any velocity, acceleration, or jerk disturbances through proper Cam design.

The diagram illustrates the execution of Replace and Restart in Time Driven Mode with Execution Schedule of Immediate and Execution Mode of Once.

When the Active Cam starts, the Slave axis is synchronized with time by using the cam profile. The Active Cam profile is replaced with the Replacement Cam when it is initiated, and then the Slave axis starts moving according to the Replacement Cam profile synchronized in time until the cam completes.

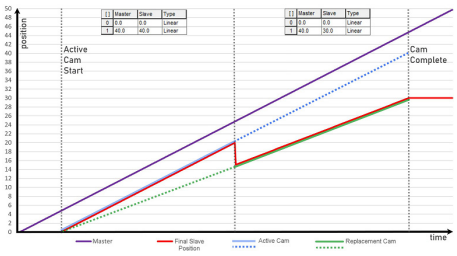


Replace and Continue

Use the enumeration Replace and Continue to replace the Active Cam at any point when it is in progress. At the replacement point, the Replacement Cam starts to interpolate the Slave axis.

The diagram illustrates the execution of a Replace and Continue in Master Driven Mode with Execution Schedule of Immediate and Execution Mode of Continuous.

The green dotted line indicates the positions of Slave axis if the Replacement Cam started at the same time as Active Cam. The blue dotted line indicates the Slave axis positions if the Active Cam continued execution beyond the replacement point. The change in Slave Position at the replacement point is due to the cam profile defined for the Replacement Cam.



Lock Direction

Lock Direction indicates the direction of the Master axis that generates the Slave motion. In Time Driven Mode, time cams are programmed with a Lock Direction of None.

In Master Driven Mode, the time cams are programmed with a following Lock Direction and are used for Cam Type Replace and Restart and Replace and Continue.

Immediate Forward Only: The Replacement Cam replaces the Active Cam immediately when Master axis is moving in the forward direction.

Immediate Reverse Only: The Replacement Cam replaces the Active Cam immediately when Master axis is moving in the reverse direction.

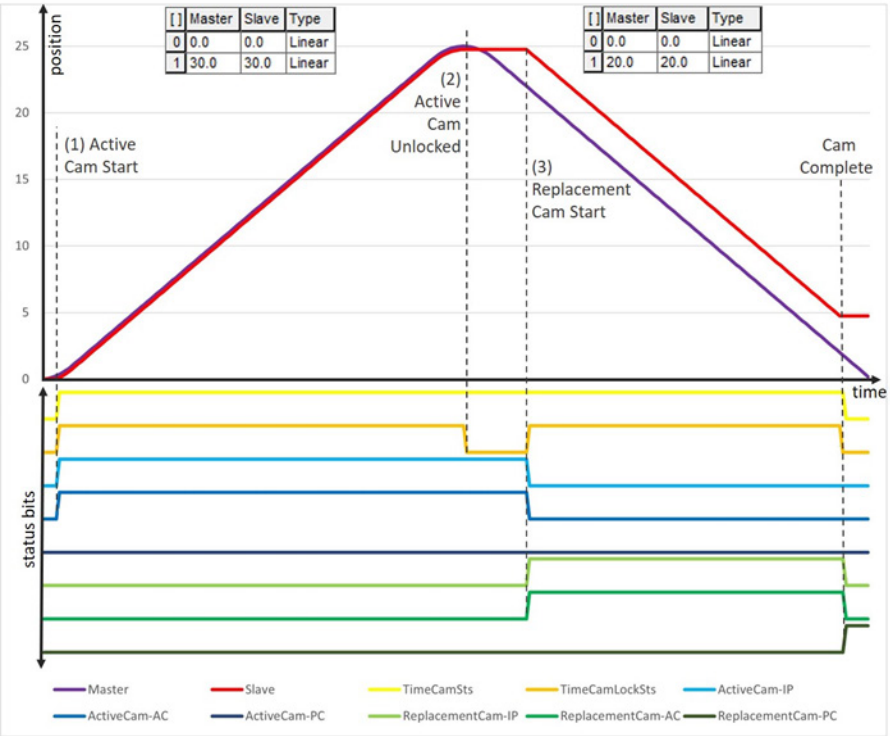
The diagram illustrates Immediate Replace and Restart Cam execution in Master Driven Mode with Execution Mode of Once and Lock Direction is Immediate Reverse Only.

(1) Active Cam Start: The Active Cam with Lock Direction of Immediate Forward only starts when Master axis moves in the forward direction. The Time Cam Status and Time Cam Lock Status of the Slave axis' motion status word bits are set.

(2) Active Cam Unlocked: Due to Master axis reversal, the Active Cam is unlocked, and the Time Cam Lock Status of the Slave axis' Motion status word is reset.

(3) Replacement Cam Start: While Master axis is moving in the reverse direction, the Replacement Cam replaces the Active Cam immediately and Replacement Cam's .AC bit is set indicating that the Replacement Cam is interpolating the Slave axis. The Time Cam Lock Status of the Slave axis' Motion status word is set.

The Replacement Cam's .PC is set when Master axis moves beyond the range of the Replacement Cam Profile. The Time Cam Status and Time Cam Lock Status of the Slave axis's Motion status word bits are reset then.



Position Forward Only: The Replacement Cam replaces the Active Cam when the cam reaches the replacement point, while the Master axis moves in the forward direction.

Position Reverse Only: Same as Position Forward Only except that the Master axis is moving in the reverse direction.

The diagram illustrates Immediate Replace and Continue Cam execution in Master Driven Mode.

(1) Active Cam Start: The Active Cam with Lock Direction of Immediate Forward only starts when Master axis moves in the forward direction. The Time Cam Status and Time Cam Lock Status of the Slave axis' motion status word bits are set.

(2) Active Cam Unlocked: Due to Master axis reversal, the Active Cam is unlocked, and the Time Cam Lock Status of the Slave axis' Motion status word is reset.

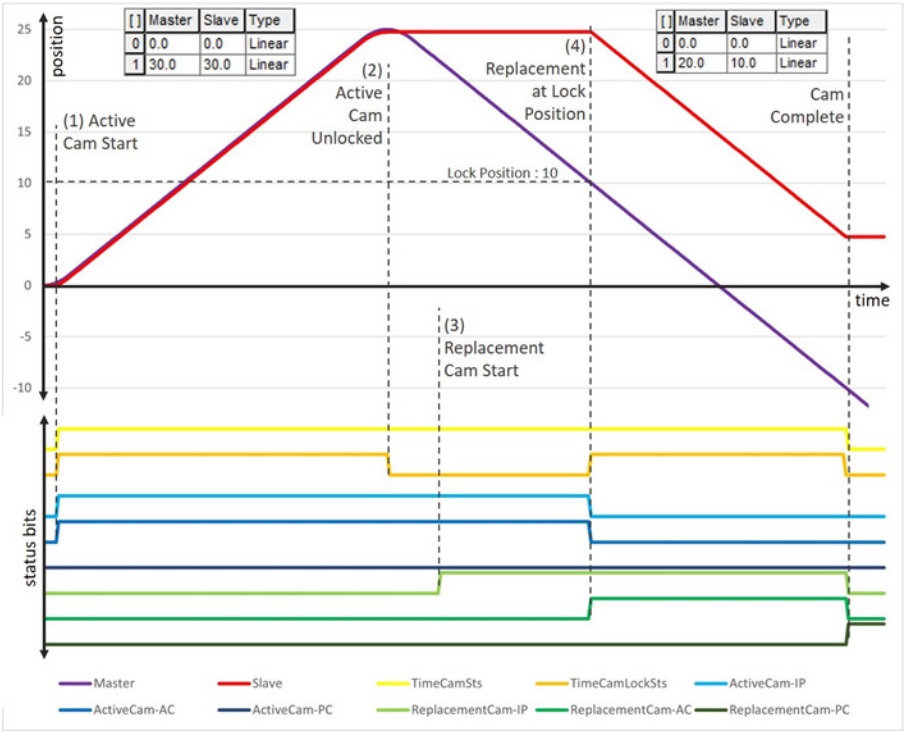
(3) Replacement Cam Start: While the Master axis is moving in the reverse direction, the Replacement Cam is initiated with Active Cam still unlocked with following configuration:

- Execution Schedule: Immediate
- Lock Direction: Position Reverse Only

- Execution Mode: Once
- Lock Position: 10
- Instruction Mode: Master Driven Mode
- Replacement at Lock Position

When Master axis starts moving in reverse direction and reaches the Lock position of 10 units, then the Replacement Cam's .AC bit is set indicating that the Replacement Cam is interpolating the Slave axis.

The Replacement Cam's .PC bit is set when Master axis moves beyond the range of the Replacement cam Profile. The Time Cam Lock Status of the Slave axis's Motion status word bit is reset then.



Stopping a Cam

Like other motion generators (for example, jog, move, gear) Active Cams must be stopped by the various stop instructions, MAS, or MGS. Cam motion must also stop when the ControlLogix processor changes OS modes. The MAS instruction, in particular, must be able to specifically stop the camming process. This behavior should be identical to the MAS functionality that stops a gearing process.

Merging from a Cam

Like other motion generators (for example, jog, move, gear) Active Cams must also be compliant with motion merge functionality. Moves and Jogs, in

particular, must be able to merge from active camming. This behavior should be identical to the merge functionality applied to a gearing process.

IMPORTANT The MATC instruction execution completes in a single scan, thus the Done (.DN) bit and the In Process (.IP) bit are set immediately. The In Process (.IP) bit remains set until the initiated Time Camming process is superseded by another MATC instruction, or stopped by a Motion Axis Stop command, Merge operation, or Servo Fault Action.

Structure

See Input and Output Parameters Structure for Single Axis Motion Instructions for the input and output parameters that are available for the MATC instruction via the Master Driven Speed Control (MDSC) function. Before any of these parameters is active, you must execute an MDAC instruction and it must be active (IP bit is set).

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Index through Arrays for array-indexing faults.A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis operand.

Execution Conditions

In Structured Text, EnableIn is always True during a normal scan. If the instruction is in the control path activated by the logic, it executes.

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, .IP, .AC and .PC bits are cleared to False.
Postscan	No Action taken
EnableIn is false This condition can only occur during Normal Scan mode.	The .EN bit is cleared to false if either the .DN or .ER bit is true.
EnableIn is true and .EN bit is false This condition can only occur during Normal Scan mode.	The .EN bit is set to true and the instruction executes according to the latest version of the Motion Instruction PISD document.

EnableIn is true and .EN bit is true This condition can only occur during Normal Scan mode.	No Action taken

Error Codes

See Motion Error Codes (.ERR) for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction-specific information for the Error Codes that are generic to many instructions. Extended error codes for the PARAMETER_OUT_OF_RANGE(13) error code lists a number that refers to the number of the operands, as they are listed in the faceplate, from top to bottom, with the first operand being counted as zero. Therefore, for the MATC instruction, an extended error code of 5 would refer to the Time Scaling operand's value. You would then have to check your value against the accepted range of values for the instruction.

SEGMENT field

The .SEGMENT field for the ILLEGAL_CAM_TYPE (28), ILLEGAL_CAM_ORDER (29) or INVALID_CAM_PROFILE_ELEMENT (179) error codes indicate the Cam Profile array element that contains, respectively, an invalid Cam Profile type (not linear or cubic), a non-ascending Master position or an invalid value (such as overflow or not a number). Therefore, INVALID_CAM_PROFILE_ELEMENT with .SEGMENT field value of 3 indicates that the fourth element (or [3]) of the Cam Profile array contains the invalid number. Use the Motion Calculate Cam Profile (MCCP) instruction or Cam Profile editor to recalculate the Cam Profile and ensure the master and slave values do not contain an invalid value.

Status Bits

Time Cam Status

This Status bit indicates that the MATC profile is in progress and sets when the MATC command is initiated on the Slave axis.

Resets with either of these conditions:

- The Active or Replacement Cam completes with Execution Mode is Once.
- The Slave axis is stopped (MAS, MCS) with Stop Type All or Time Cam.
- The Slave axis shuts down (MASD, MGSD, MCSD).

Bit Name	Statement	Execution Mode	
		Once	Continuous
Time Cam Status	Sets when the MATC command is initiated on the Slave axis	TRUE	TRUE
	Resets when the Master axis crosses the cam boundary.	TRUE	FALSE

Time Cam Lock Status

The Time Cam Lock Status bit is valid only in Master Driven mode and when the Execution schedule is Immediate. The Time Cam Lock Status bit indicates that the status when the Master axis starts driving the Slave axis based on the cam profile and Slave Direction and sets when the Master axis satisfies these conditions: Execution Mode, Lock Direction, and Lock Position.

Resets with either of these conditions:

- The Master axis crosses the cam boundary when the Execution Mode is Once.
- The Master axis is moving in the reverse direction of that specified in the Lock direction.
- The Slave axis is stopped (MAS, MCS) with Stop Type All or Time Cam.
- The Slave axis shuts down (MASD, MGSD, MCSD).

Bit Name	Statement	Execution Mode	
		Once	Continuous
Time Cam Lock Status	Sets when the Master axis starts driving the Slave axis according to the cam profile.	TRUE	TRUE
	Resets when the Master axis crosses the cam boundary.	TRUE	FALSE

Bit Name	Statement	Lock Direction
Time Cam Lock Status	Sets when the Master axis satisfies these conditions: Execution Mode, Lock Direction. Lock Position is ignored.	Immediate Forward Only Immediate Reverse Only
	Sets when the Master axis satisfies these conditions: Execution Mode, Execution Schedule, Lock Direction, and Lock Position.	Position Forward Only Position Reverse Only

Time Cam Pending Status

This Status bit indicates that the MATC profile is pending the completion of an executing cam profile and sets when these conditions are met:

- The MATC with Execution Schedule Pending is initiated.
- Replacement Cam is initiated with Instruction Mode Master Driven and the Lock Direction.

Resets with either of these conditions:

- The current Time cam process completes (.PC bit) sets.
- Time Cam Lock Status sets.
- The Slave axis is stopped (MAS, MCS) with Stop Type All or Time Cam.
- The Slave axis is shut down (MASD, MGSD, MCSD).

		Execution Schedule	
Bit Name	**Statement**	**Immediate**	**Pending**
Time Cam Pending Status	Bit Status in Execution Schedule	FALSE	TRUE

Bit Name	Statement	Execution Schedule
Time Cam Pending Status	Sets until the Master axis satisfies these conditions based on Execution Mode, Lock Direction, and Lock Position.	Replacement Cam with: Immediate Forward Only Immediate Reverse Only Position Forward Only Position Reverse Only

The diagram illustrates Immediate Replace and Continue Cam execution in Master Driven Mode.

(1) Active Cam Start

The Active Cam with Lock Direction of Immediate Forward only starts when the Master axis moves in the forward direction. The Time Cam Status and Time Cam Lock Status of the Slave axis' motion status word bit sets. The Time Cam Pending Status bit of the Slave axis' Motion Status bit is Unchanged.

(2) Active Cam Unlocked

Due to Master axis reversal, the Replacement Cam unlocks, and the Time Cam Lock Status of the Slave axis' Motion status word resets. The Time Cam Status of the Slave axis' Motion Status bit is Unchanged.

(3) Replacement Cam Start

While the Master axis is moving in the reverse direction, the Replacement Cam initiates with Active Cam still unlocked with this configuration:

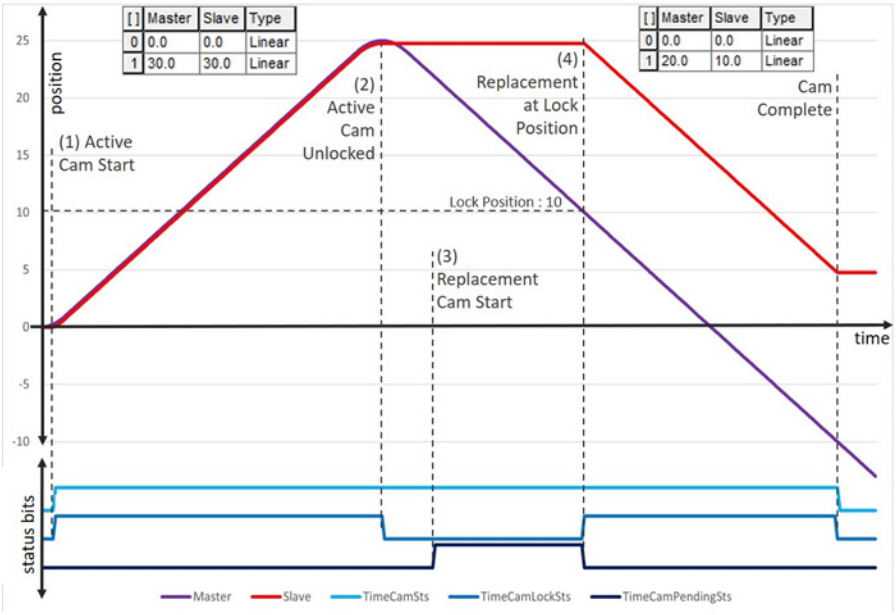
- Execution Schedule: Immediate

- Lock Direction: Position Reverse Only
- Execution Mode: Once
- Lock Position: 10
- Instruction Mode: Master Driven Mode

The Time Cam Pending Status of the Slave axis' Motion status word bit sets. The Time Cam Status and the Time Cam Lock Status of the Slave axis' Motion Status bits are Unchanged.

(4) Replacement at Lock Position

When the Master axis starts moving in the reverse direction and reaches the Lock position of 10 units, the Time Cam Lock Status of the Slave axis' Motion status word bit sets, and the Time Cam Pending Status of the Slave axis' Motion status word bit resets. The Time Cam Status of the Slave axis' Motion Status bit is Unchanged.

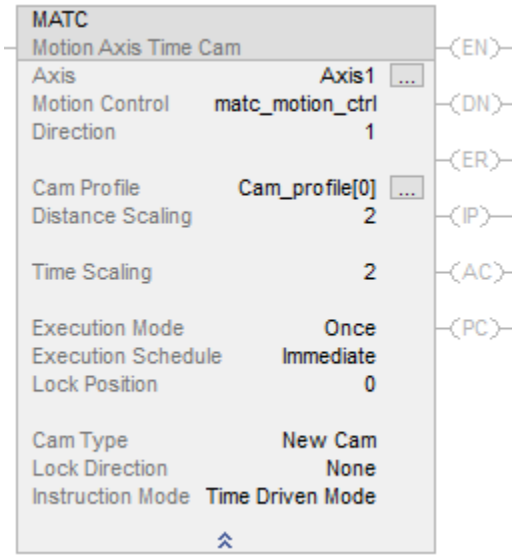


Master Driven Speed Control (MDSC) and Motion Direct Command Support

The Motion Direct commands are not available in the instruction tree for the MATC instruction. You must program an MATC in one of the supported programming languages before you execute an MAM or MAJ in Time Driven Mode. A runtime error occur if an MATC is not previously executed in an MAM and MAJ in Master Driven Mode.

Example

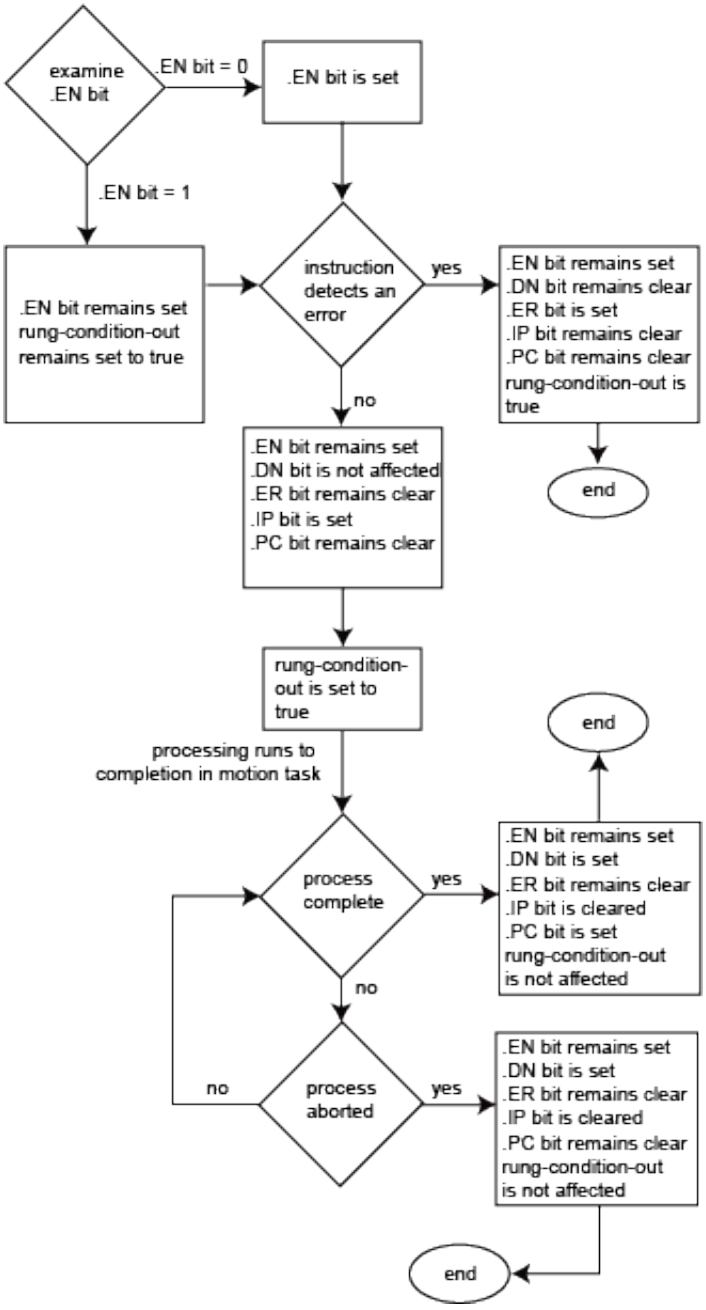
Relay Ladder Logic



Structured Text

MATC(Axis1,matc_motion_ctrl,1,cam_profile[0],2,2,Once,Immediate,0,NewCam,None,TimeDrivenMode);

MATC Flow Chart (True)



Motion Group Instructions

Motion Group Instructions

Group instructions include all motion instructions that operate on all the axes in the specified group. Instructions that are applied to groups include position strobe, shutdown control, and stopping instructions. Only one group is supported for each Logix controller.

These are the motion group instructions.

Available Instructions

Ladder Diagram and Structured Text

MGS	MGSD	MGSR	MGSP

Function Block

Not available

IMPORTANT Tags used for the motion control attribute of instructions should only be used once. Reuse of the motion control tag in other instructions can cause unintended operation. This may result in damage to equipment or personal injury.

Group Control Instructions include all motion instructions that operate on all the axes in the specified group. Instructions that can be applied to groups include position strobe, shutdown control, and stopping instructions. Note that at present only one group is supported per Logix controller.

The motion group instructions are:

If you want to:	Use this instruction:
Initiate a stop of motion on a group of axes.	MGS
Force all axes in a group into the shutdown operating state.	MGSD
Transition a group of axes from the shutdown operating state to the axis ready operating state.	MGSR
Latch the current command and actual position of all axes in a group.	MGSP

See also

Motion Configuration Instructions on page 315

Motion Move Instructions on page 73

Motion State Instructions on page 23

Multi-Axis Coordinated Motion Instructions on page 359

Motion Event Instructions on page 249

Motion Group Stop (MGS)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The Motion Group Stop (MGS) instruction initiates a stop of all motion in progress on all axes in the specified group by a method configured individually for each axis or as a group via the Stop Mode of the MGS instruction. If the MGS Stop Mode is specified as Programmed, each axis in the group is stopped according to the configured Programmed Stop Mode axis attribute. This is the same stopping mechanism that is employed by the Logix Operating System when there is a Logix controller state change. This Programmed Stop Mode attribute currently provides five different methods of stopping an axis:

- Fast Stop
- Fast Disable
- Hard Disable
- Fast Shutdown
- Hard Shutdown

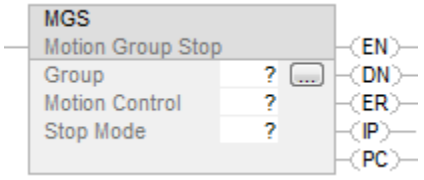
Alternatively, an explicit Stop Mode may be selected by using the MGS instruction. If a Stop Mode of Fast Disable is selected, all axes in the group stop with Fast Disable behavior. When the motion of all the axes in the group has been brought to a stop, the Process Complete (PC) bit is set in the control structure.

This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.
-

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MGS(Group,MotionControl, StopMode);

Operands

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Group	MOTION_GROUP	Tag	Name of the group of axes to perform operation on
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Stop Mode	UDINT	Immediate	Controls how the axes in the group are stopped. Select one of the following methods: 0 = Programmed - each axis is stopped according to how the individual axis has been configured. 1 = Fast Stop - each axis in the group is decelerated at the Maximum Deceleration rate and the stopped axis is left in the Servo Active state. 2 = Fast Disable - each axis in the group is decelerated at the Maximum Deceleration rate and the stopped axis is placed in the Axis Ready state.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

For the operands that require you to select from available options, enter your selection as:

This Operand	Has These Options Which You	
	Enter as Text	Or Enter as a Number

StopMode	programmed	0
	faststop	1
	fastdisable	2

MOTION_INSTRUCTION Structure

Mnemonic	Description
.DN (Done) Bit 29	It is set when the group Programmed Stop has been successfully initiated for all axes in the group.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured group.
.IP (In Process) Bit 26	It is set on positive rung transition and cleared after the Motion Group Stop is complete.
.PC (Process Complete) Bit 27	It is set after all the axes in group have been successfully brought to a stop according to each axis' Programmed Stop Mode configuration.

Description

With the Stop Mode parameter set for Programmed, the MGS instruction brings motion for all of the axes in the specified group to a stop according to the configured Programmed Stop Mode for each axis. If the axis has both single axis motion moves and coordinated moves occurring, the MGS stops the single axis motion using the axis' maximum deceleration rate and stops the coordinated axes motion using the coordinate system's maximum deceleration rate. A trapezoidal profile is always used for the deceleration regardless of the programmed profile type.

The MGS instruction initiates the same programmed stopping action that is automatically applied when the processor's operating system changes operating mode (for example, Run Mode to Program Mode). This is particularly useful in designing custom motion fault handlers.

If the MGS Stop Mode parameter is set to Fast Stop, each axis in the group is forced to perform a Fast Stop process, regardless of the configured Programmed Stop Mode. Each axis in the group is decelerated at the Maximum Deceleration rate and, once stopped, the axis is left in the Servo Active state.

If the MGS Stop Mode parameter is set to Fast Disable, each axis in the group is forced to perform a Fast Disable process, regardless of the configured Programmed Stop Mode. Each axis in the group is decelerated at the Maximum Deceleration rate and, once stopped, placed into the Axis Ready (servo inactive and drive disabled) state.

The MGS instruction currently supports five Programmed Stop Action modes:

- Fast Stop
- Fast Disable
- Hard Disable

- Fast Shutdown
- Hard Shutdown

Each axis may be configured to use any of these five stop modes. This table describes the effect of each these five stopping modes as they apply to an individual axis in the specified group.

Mode	Description
Fast Stop	For an axis configured for a Fast Stop the MGS instruction initiates a controlled stop much like that initiated by an MAS instruction. In this case the Motion Group Stop (MGS) instruction brings the axis motion to a controlled stop without disabling the axis servo loop. It is useful when a fast decelerated stop the axis is desired with servo control retained. The MGS instruction uses the configured Maximum Deceleration of the axis to stop only the single axis motion. The coordinated move portion of the axis uses the coordinated system configured Maximum Deceleration to stop the axis. If the coordinate system contains orientation axes, configured Maximum Orientation Deceleration is used to stop orientation axes. When a Fast Stop is used to stop a Motion Drive Start (MDS) instruction, the Direct Command feature is disabled. Additionally, the affected axis decelerates to a stop using its ramp deceleration.
Fast Disable	For an axis configured for a Fast Disable the MGS instruction initiates a controlled stop much like that initiated by an MAS instruction with the exception that the drive is disabled when the axis comes to a stop. Use MGS when a fast decelerated stop the axis is desired before the drive is disabled. The MGS instruction uses the configured Maximum Deceleration of the axis to stop only the single axis motion. The coordinated move portion of the axis uses the coordinated system configured Maximum Deceleration to stop the axis. If the coordinate system contains orientation axes, configured Maximum Orientation Deceleration is used to stop orientation axes. CIP Motion When a Fast Disable is issued and a Direct Velocity command is issued via the MDS instruction, the CIP axis is disabled when all planned motion is stopped. However, in this case, the drive device continues coasting or stopping according to its Stop Action selection.
Hard Disable	For an axis configured for a Hard Disable the MGS instruction initiates the equivalent of an MSF instruction to the axis. This action immediately turns the appropriate axis drive output off, and disables the servo loop. Depending on the drive configuration, this may result in the axis coasting to a stop but offers the quickest disconnect of drive output power. When a Hard Disable is used to stop a Motion Drive Start (MDS) instruction, the Direct Command feature is disabled. Additionally, the affected axis is immediately disabled.
Fast Shutdown	For an axis configured for a Fast Shutdown, the MGS instruction initiates a Fast Stop and then applies the equivalent of a Motion Axis Shutdown (MASD) instruction to the axis. This action turns the appropriate axis driver output OFF, disables the servo loop, opens any associated motion module's OK contacts, and places the axis into the Shutdown state. When a Fast Shutdown is used to stop a Motion Drive Start (MDS) instruction, the Direct Command feature is disabled. Additionally, the affected axis decelerates to a stop using its ramp deceleration is immediately shutdown.
Hard Shutdown	For an axis configured for a Hard Shutdown the MGS instruction initiates the equivalent of an Motion Axis Shutdown (MASD) instruction to the axis. This action turns the appropriate axis drive output OFF, disables the servo loop, opens any associated motion module OK contacts, and places the axis into the Shutdown state. Depending on the drive configuration, this may result in the axis coasting to a stop but offers the quickest disconnect of Drive power via the OK contacts. To successfully execute a MGS instruction, the targeted group must be configured. When a Hard Shutdown is used to stop a Motion Drive Start (MDS) instruction, the Direct Command feature is disabled. Additionally, the affected axis is immediately shutdown.

IMPORTANT The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

In addition to the ways the various stopping modes affect the Motion Drive Start (MDS) instruction were described in the previous table, all these modes also clear the MDS In Process (.IP) bit and clear the DirectVelocityControlStatus bit in the Motion Status attribute.

This is a transitional instruction:

- In relay ladder, toggle Rung-condition-in from false to true each time the instruction should execute.

Master Driven Speed Control (MDSC) and the MGS Instruction

If an MGS is issued when in Master Driven Mode, a switch is made to Time Driven Mode and the axes are stopped in Time Driven Mode.

The IP bit of the Master Driven Axis Control (MDAC) and Master Driven Coordinate Control (MDCC) instructions are reset.

The AC bit of the MDAC and MDCC instructions are reset when the axis is stopped as the group is shutdown.

The status bit CalculatedDataAvailable in an active motion instruction status word for an MAM, MCLM or MCCM instruction is cleared when an MCS is executed (goes IP). The CalculatedData is not recomputed.

The MGS instruction clears the pending Master Axis for all future single and coordinated motion instructions.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Status Bits

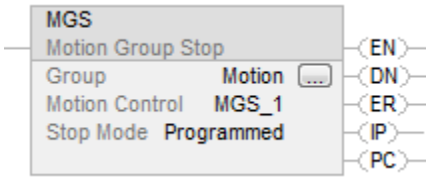
MGS Changes to Single Axis Status Bits

If the Stop Type is	Then		
NOT All	The instruction clears the Motion Status bit for the motion process that you stopped.		
All	The instruction clears all Motion Status bits.		
	Bit Name	**State**	**Meaning**
	MoveStatus	FALSE	Axis is not Moving.
	JogStatus	FALSE	Axis is not Jogging.
	GearingStatus	FALSE	Axis is not Gearing.
	HomingStatus	FALSE	Axis is not Homing.
	StoppingStatus	TRUE	Axis is not Stopping.
	PositionCamStatus	FALSE	Axis is not Position Camming.
	TimeCamStatus	FALSE	Axis is not Time Camming.
	PositionCamPendingStatus	FALSE	Axis does not have a Position Cam Pending.
	TimeCamPendingStatus	FALSE	Axis does not have a Time Cam Pending.
	GearingLockStatus	FALSE	Axis is not in a Gear Locked condition.
	PositionCamLockStatus	FALSE	Axis is not in a Cam Locked condition.
	DirectVelocityControlStatus	FALSE	Axis is not under Direct Velocity Control.
	DirectTorqueControlStatus	FALSE	Axis is not under Direct Torque Control.

Example

When the input conditions are true, the controller stops motion on all axes in group1. After the controller stops all motion, the axes are inhibited.

Ladder Diagram



Structured Text

MGS(Motion,MGG_1,Programmed);

See also

Motion Drive Start (MDS) on page 53

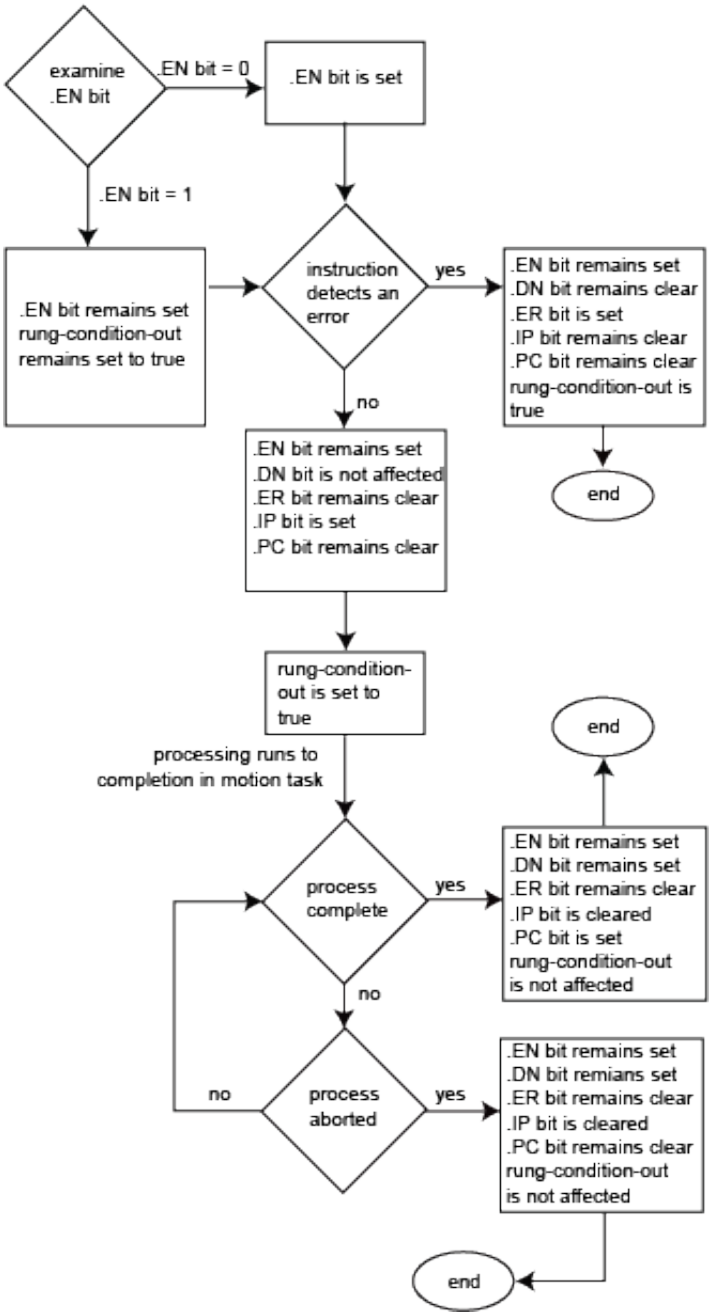
MGS Flow Chart (True) on page 232

Motion Error Codes (.ERR) on page 581

Motion Group Instructions on page 225

Common Attributes on page 691

MGS Flow Chart (True)



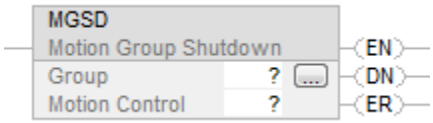
Motion Group Shutdown (MGSD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Use the Motion Group Shutdown (MGSD) instruction to force all axes in the designated group into a Shutdown state. The Shutdown state of an axis is Servo Off, drive output is deactivated, and the motion module's OK solid-state relay contacts, if applicable, are opened. The group of axes remains in the Shutdown state until either Group Shutdown Reset is executed or each axis is individually reset via the Motion Axis Shutdown (MASD) instruction.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MGSD(Group,MotionControl);

Operands

Ladder Diagram

Operand	Type	Format	Description
Group	MOTION_GROUP	Tag	Name of the group of axes to perform operation on
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

Structured Text

Operand	Type	Format	Description
Group	MOTION_GROUP	Tag	Name of the group of axes to perform operation on
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and Rung-condition-in goes false.
.DN (Done) Bit 29	The done bit indicates when the instruction sets the group of axes to the shutdown operating state.

.ER (Error) Bit 28	The error bit indicates when the instruction detects an error, such as if messaging to the servo module failed.

Description

The MGSD instruction turns drive output off, disables the servo loops of all axes in the specified group, and opens any associated OK contacts for all applicable motion modules in the group. This action places all group axes into the Shutdown state. The MGSD instruction takes only one parameter; simply select or enter the desired group to shutdown.

Another action initiated by the MGSD instruction is the clearing of all motion processes in progress and a clearing of all the motion status bits. Associated with this action, the command also clears all motion instruction .IP bits that may currently be set for each axis in the group.

The MGSD instruction forces the targeted group of axes into the Shutdown state. One of the unique characteristics of the Shutdown state is that the OK solid state relay contact for all of the group's motion modules Open. This feature can be used to open up the E-Stop string(s) that control main power to the various drive systems.

Another characteristic of the Shutdown state is that any instruction that initiates axis motion for an axis within the group is blocked from execution. Attempts to do so results in an execution error. Only by executing one of the Shutdown Reset instructions can motion then be successfully initiated.

To successfully execute a MGSD instruction, the targeted group must be created and configured.

IMPORTANT The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

Additionally, the MGSD instruction supports canceling the Motion Drive Start (MDS) instruction. This includes clearing the MDS In Process (.IP) bit, and clearing the DirectVelocityControlStatus bit and the DirectTorqueControlStatus bit in the Motion Status attribute.

This is a transitional instruction:

- In relay ladder, toggle Rung-condition-in from false to true each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition.

Master Driven Speed Control (MDSC) and the MGSD Instruction

When the group shuts down:

- The IP bit of the Master Driven Axis Control (MDAC) and Master Driven Coordinate Control (MDCC) instructions reset as the group is shutdown.
- The AC bit of the MDAC and MDCC instructions are reset when the axis is stopped as the group is shutdown.
- The MGS instruction clears the pending Master Axis for all future single and coordinated motion instructions.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See Common Attributes for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See Motion Error Codes (ERR) for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. See Motion Error Codes (ERR) for Motion Instructions.

Status Bits

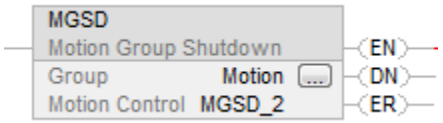
MGSD Changes to Single Axis Status Bits

Bit Name	State	Meaning
ServoActionStatus	FALSE	Axis is in Servo Off state with the servo loop inactive.
DriveEnableStatus	FALSE	Axis Drive Enable output is inactive.
ShutdownStatus	TRUE	Axis is in Shutdown state.
AccelStatus	FALSE	Axis is not Accelerating.
DecelStatus	FALSE	Axis is not Deceleratin.
GearingLockStatus	FALSE	Axis is not locked.
JogStatus	FALSE	Axis is not Jogging.
MoveStatus	FALSE	Axis is not Moving.
GearingStatus	FALSE	Axis is not Gearing.
HomingStatus	FALSE	Axis is not Homing
DirectVelocityControlStatus	FALSE	Axis is not under Direct Velocity Control.
DirectTorqueControlStatus	FALSE	Axis is not under Direct Torque Control.

Example

When the input conditions are true, the controller forces all axes in group1 into a shutdown operating state.

Ladder Diagram



Structured Text

MGSD(Motion,MGSD_2);

See also

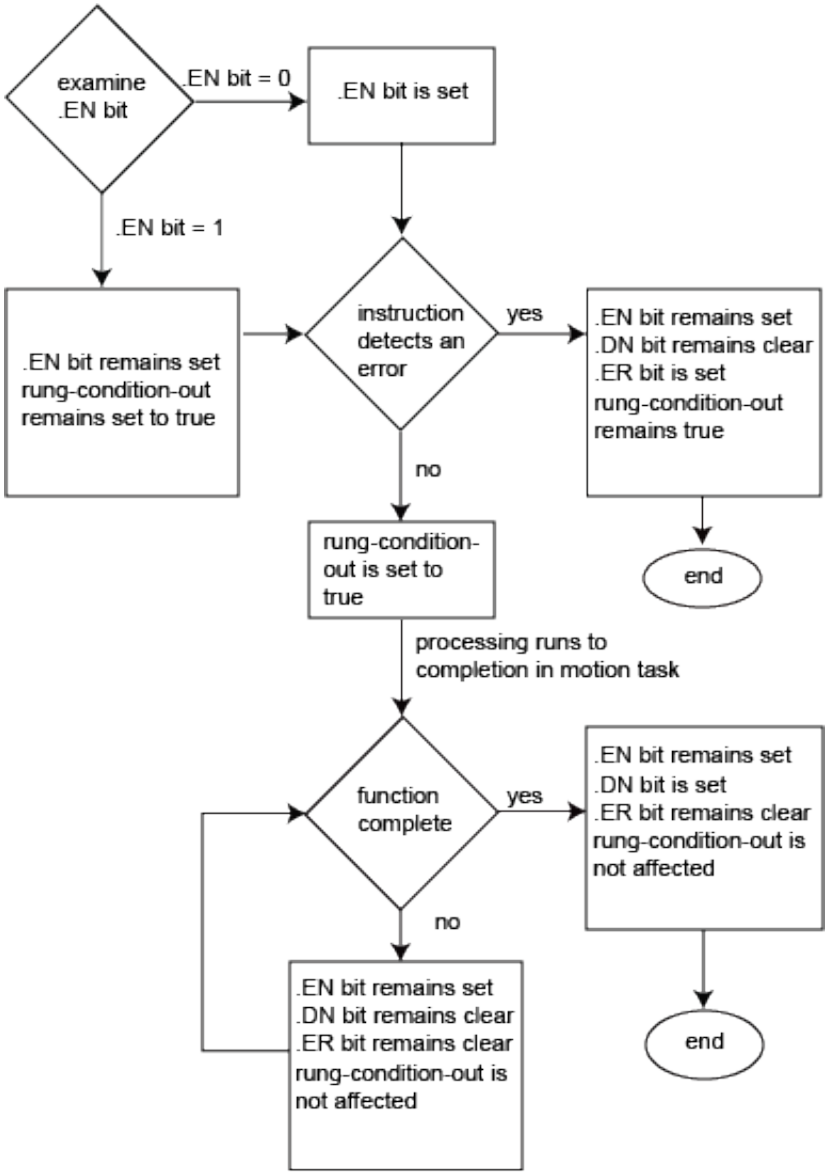
Motion Drive Start (MDS) on page 53

Structured Text Syntax on page 665

Common Attributes on page 691

Motion Error Codes (.ERR) on page 581

MGSD Flow Chart (True)



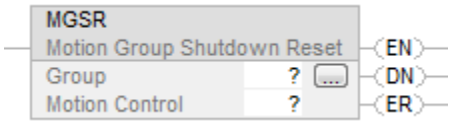
Motion Group Shutdown Reset (MGSR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Use the Motion Group Shutdown Reset (MGSR) instruction to transition a group of axes from the shutdown operating state to the axis ready operating state. As a result of this command, all faults associated with the axes in the group are cleared and any OK relay contacts of motion modules associated with the specified group are closed.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MGSR(Group,MotionControl);

Operands

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Group	MOTION_GROUP	Tag	Name of the group of axes to perform operation on
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and Rung-condition-in goes false.
.DN (Done) Bit 29	The done bit indicates when the instruction resets the group of axes from the shutdown operating state.
.ER (Error) Bit 28	The error bit indicates when the instruction detects an error, such as if messaging to the servo module failed.

Description

The MGSR instruction takes all the axes in the specified group out of the Shutdown state by clearing all axis faults and closing any associated OK solid-state relay contacts for the motion modules within the group. This action places all axes within the motion group in the Axis Ready state.

Just as the Motion Group Shutdown (MGSD) instruction forces all the axes in the targeted group into the Shutdown state. The MGSR instruction takes all the axis in the specified group out of the Shutdown state and into the Axis Ready state. One of the unique characteristics of the Shutdown state is that, if supported, the OK solid state relay contact for each of the group's motion modules is Open. Hence, the result of an MGSR instruction applied to a group of motion modules is that all motion module OK relay contacts close. This feature can be used to close the E-Stop strings that control main power to the various drive systems and permits the customer to reapply power to the drives.

To successfully execute a MGSR instruction, the targeted group must be configured.

IMPORTANT The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle Rung-condition-in from false to true each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken

Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Status Bits

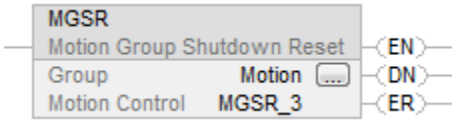
MGSR Changes to Status Bits

Bit Name	State	Meaning
ServoActionStatus	FALSE	Axis in Servo Off state with the servo loop inactive.
DriveEnableStatus	FALSE	Axis Drive Enable output is inactive.
ShutdownStatus	FALSE	Axis is in Shutdown state.

Examples

When the input conditions are true, the controller transitions all axes in group1 from the shutdown operating state to the axis ready operating state.

Ladder Diagram



Structured Text

```
MGSR(Motion,MGSR_3);
```

See also

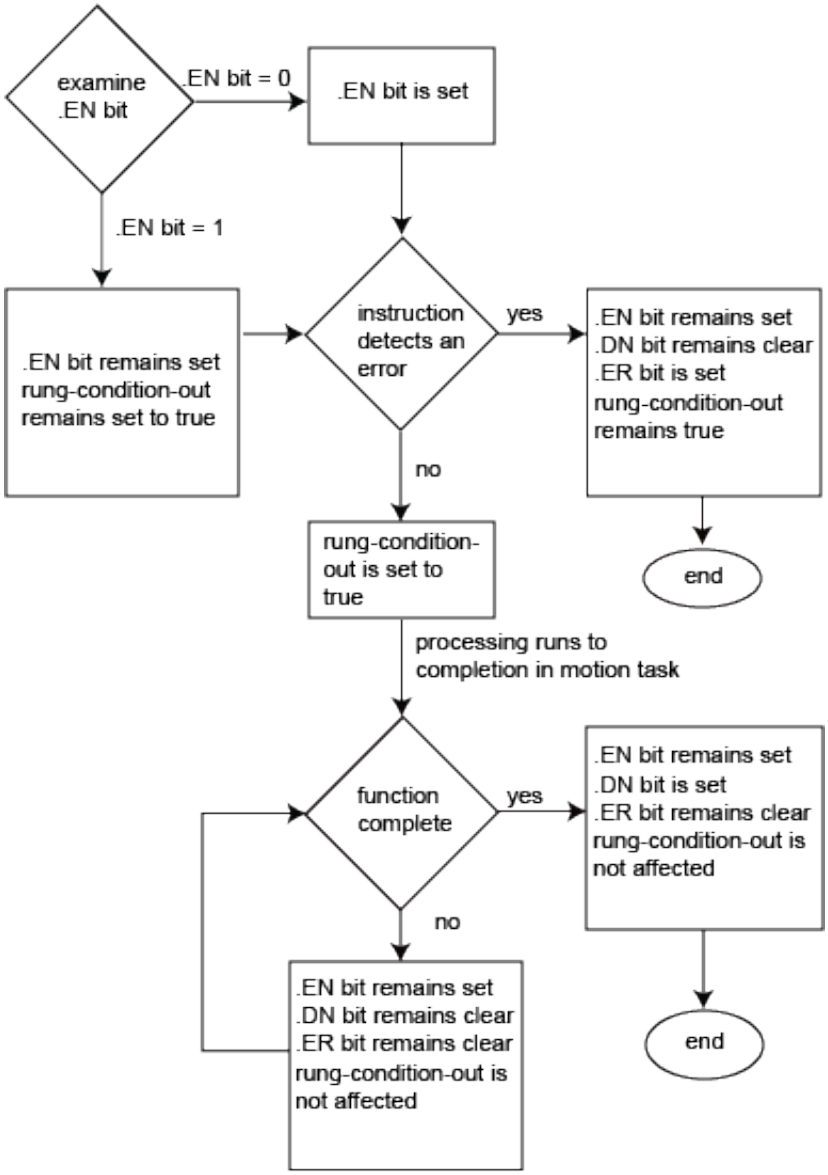
Motion Group Shutdown (MGSD) on page 233

Structured Text Syntax on page 665

Motion Error Codes (ERR) on page 581

Common Attributes on page 691

MGSR Flow Chart (True)



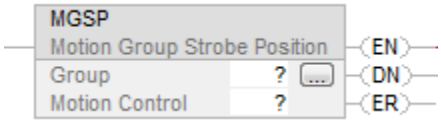
Motion Group Strobe Position (MGSP)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Use the Motion Group Strobe Position (MGSP) instruction to latch the current Command and Actual Position of all axes in the specified group at a single point in time. The latched positions are available in the StrobeActualPosition and StrobeCommandPosition parameters in the Motion Axis Object for each axis configured in the group.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MGSP(Group,MotionControl);

Operands

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Group	MOTION_GROUP	Tag	Name of the group of axes to perform operation on
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set when the group of axes have been successfully set to Shutdown state.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured group.

Description

The MGSP instruction synchronously latches all command and actual position values of all axes in the specified group at the time of execution. The MGSP

instruction takes only one parameter; simply select or enter the desired axis to strobe.

If the targeted group does not appear in the list of available groups, the group has not been configured for operation. Use the Tag Editor to create and configure a new groups.

The MGSP instruction may be used at any time to capture a complete set of command and actual position information for all axes in the specified group. This operation is often required as a precursor to computations involving position values of different axes within the group.

To successfully execute a MGSP instruction, the targeted group must be configured.

IMPORTANT The MGSP instruction execution completes in a single scan, setting the Done .DN bit immediately.

This is a transitional instruction:

- In relay ladder, toggle Rung-condition-in from false to true each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Status Bits

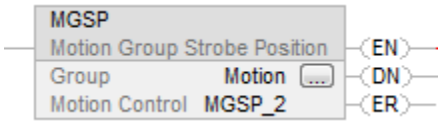
MGSP Changes to Status Bits

The MGSP instruction does not make any changes to the status bits.

Examples

When the input conditions are true, the controller latches the current command and the actual position of all axes in group1.

Ladder Diagram



Structured Text

MGSP(Motion,MGSP_2);

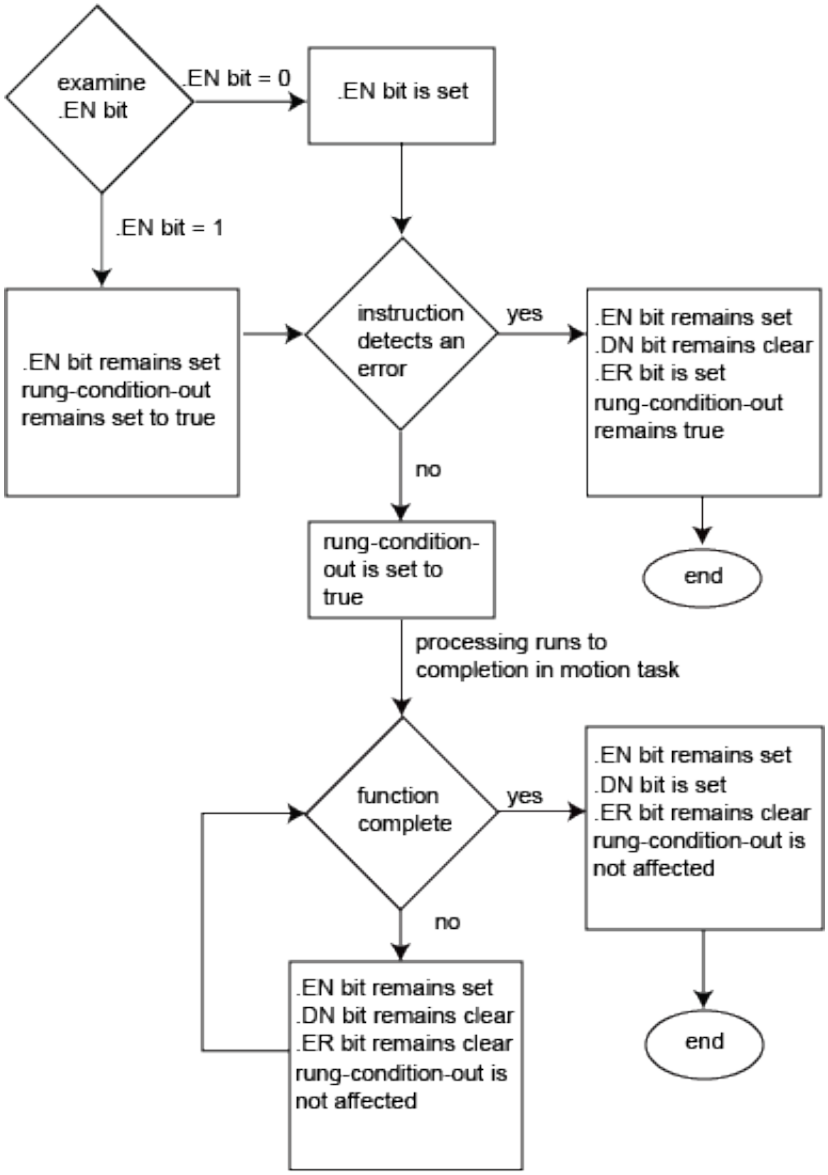
See also

Structured Text Syntax on page 665

Motion Error Codes (.ERR) on page 581

Common Attributes on page 691

MGSP Flow Chart (True)



Motion Event Instructions

Motion Event Instructions

These are the motion event instructions.

Available Instructions

Ladder Diagram and Structured Text

MAW	MDW	MAR	MDR	MAOC	MDOC

Function Block

Not available

Important: Tags used for the motion control attribute of instructions should only be used once. Reuse of the motion control tag in other instructions can cause unintended operation. This may result in damage to equipment or personal injury.

Motion event instructions control the arming and disarming of special event checking functions, such as registration and watch position.

The motion event instructions are:

If you want to:	Use this instruction:
Arm watch-position event-checking for an axis.	MAW
Disarm watch-position event-checking for an axis.	MDW
Arm servo-module registration-event checking for an axis.	MAR
Disarm servo-module registration-event checking for an axis.	MDR
Arm an Output Cam.	MAOC
Disarm an Output Cam	MDOC

See also

Motion Configuration Instructions on page 315

Motion Group Instructions on page 225

Motion Move Instructions on page 73

Motion State Instructions on page 23

Multi-Axis Coordinated Motion Instructions on page 359

Motion Arm Watch (MAW)

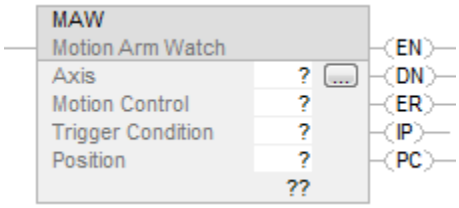
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Use the Motion Arm Watch (MAW) instruction to arm motion module watch position event checking for the specified axis. When this instruction is called, a watch position event is enabled using the watch Position for the Axis and specified Forward or Reverse event condition. After the arming is complete the Actual Position for the Axis is monitored against the Watch Position and when the specified watch event condition is met, the Event (PC) bit is set, and the Watch Event Status bit in the Axis data structure is set.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MAW(Axis,MotionControl, TriggerCondition,Position);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description
Axis	AXIS_CIP_DRIVE	AXIS_CIP_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE AXIS_GENERIC_DRIVE AXIS_GENERIC **Tip:** AXIS_GENERIC is supported by the ControlLogix 5570 and the GuardLogix 5570 controllers only.	Tag	Name of the axis on which to perform the operation. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Trigger Condition	BOOLEAN	BOOLEAN	Immediate	Select the watch-event trigger condition: 0 = forward – the servo module looks for the actual position to change from less than the watch position to greater than the watch position. 1 = reverse – the servo module looks for the actual position to change from greater than the watch position to less than the watch position.
Position	REAL	REAL	Immediate or Tag	The new value for the watch position.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

For the operands that require you to select from available options, enter your selection as:

This Operand	Has These Options Which You	
	Enter as Text	Or Enter as a Number
TriggerCondition	forward reverse	0 1

MOTION_INSTRUCTION Structure

Mnemonic	Description

.EN (Enable) Bit 31	It is set to true when the rung makes a false-to-true transition and remains set to true until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set to true when the axis watch event checking has been successfully armed.
.ER (Error) Bit 28	It is set to true to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.IP (In Process) Bit 26	It is set to true on positive rung transition and cleared to false after the watch event has occurred, or has been superseded by another Motion Arm Watch, or terminated by a Motion Disarm Watch command.
.PC (Process Complete) Bit 27	It is set to true when a watch event occurs.

Description

The MAW instruction sets up a Watch Position event to occur when the specified physical axis reaches the specified Set-point position.

Set Point Position

Watch Position events are useful for synchronizing an operation to a specified axis position while the axis is moving, such as activating a solenoid to push a carton off a conveyor at a certain axis position. Select or enter the desired physical axis, the desired Trigger Condition, and enter a value or tag variable for the desired Watch Position.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

When an Arm Watch Position instruction is executed, the WatchEventStatus bit is set to 0 (FALSE) and the actual position of a physical axis is monitored (at the servo loop update rate) until it reaches the specified Watch Position. After the watch position event occurs, the WatchEventStatus bit for the axis is set to 1 (TRUE).

Multiple watch position events may be active at a given time, however only one may be active at a time for any given physical axis. Each event is monitored independently and may be checked using the appropriate WatchEventStatus bit.

IMPORTANT In large I/O connections, force values can slow down the rate at which the controller processes repetitive watch positions.

To successfully execute a MAW instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. Otherwise, the instruction errs.

IMPORTANT The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis Operand. See Common Attributes for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is set to true. Otherwise, the .EN bit is not affected. The .DN,.ER ,.IP and .PC bits are not affected.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See Motion Error Codes (ERR) for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. The following Extended Error codes help to pinpoint the problem when the MAW instruction receives a Servo Message Failure (12) error message. See *Motion Error Codes (.ERR)* for Motion Instructions.

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning

SERVO_MESSAGE_FAILURE (12)	No Response (2)	Not enough memory resources to complete request. (SERCOS)

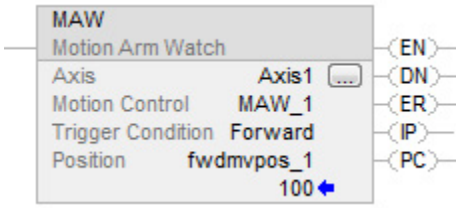
Status Bits

MAW Changes to Status Bits

Bit Name	State	Meaning
WatchEventArmedStatus	TRUE	The axis is looking for a watch position event.
WatchEventStatus	FALSE	The previous watch event is cleared.

Example

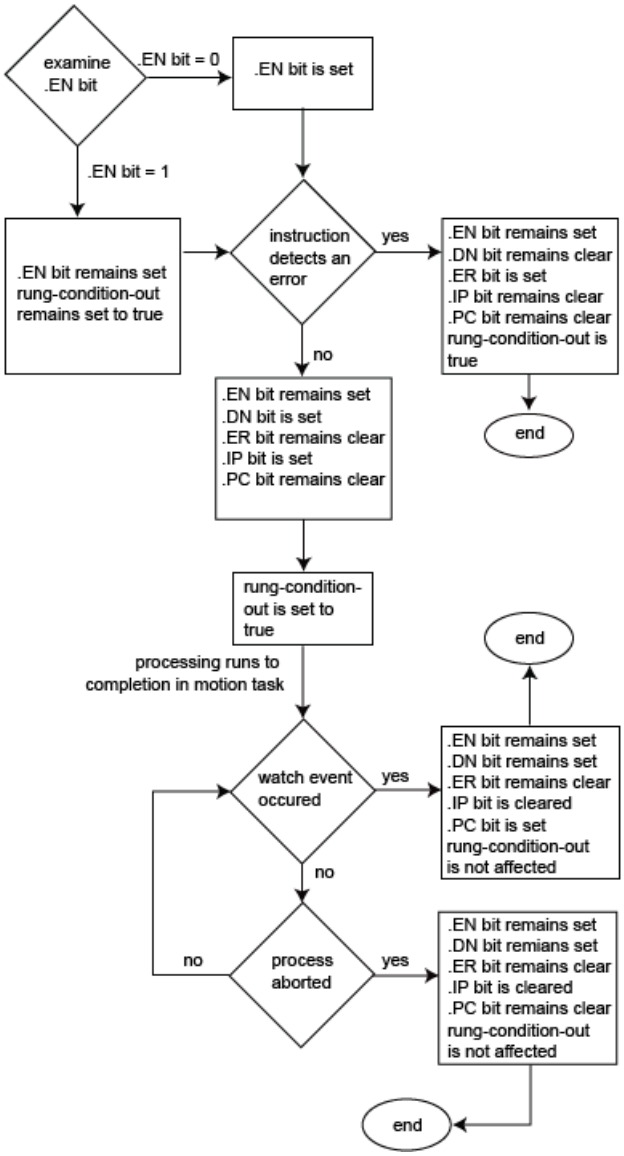
Ladder Diagram



Structured Text

MAW(Axis1,MAW_1,Forward,fwdmvpos_1);

MAW Flow Chart (True)



Understand a Programming example

This figure shows rungs to redefine the axis position of a motion control application program using the Motion Arm Watch (MAW) instruction.

Watch Position Example

This example shows the ladder logic to:

- Create a watch position event
- Enable an output when the trigger condition occurs.

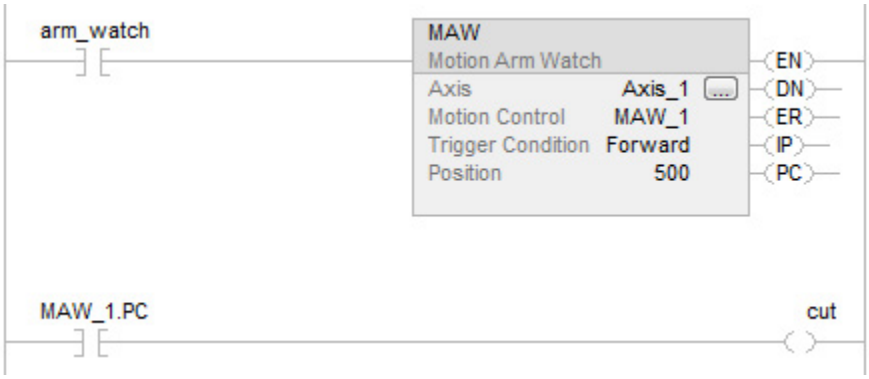
Rung 0:

This rung arms a watch position event for Axis_1 when the arm_ watch bit transitions from false to true

When the MAW instruction begins executing, the Axis_1.WatchEvStatus bit resets. This bit sets when the actual position of 500 in the forward direction.

Rung 1:

This rung enables the cut output when the watch position occurs. (The MAW_1.PC bit sets when the watch position occurs.)



The Motion Disarm Watch (MDW) instruction is used to disarm watch position event checking for the specified axis.

Motion Disarm Watch (MDW)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Use the Motion Disarm Watch (MDW) instruction to disarm watch-position event-checking for an axis. This instruction has the effect of clearing both the Watch Event Status and Watch Armed Status bits in the axis data structure. Executing this instruction also clears the In Process bit associated with the controlling Motion Arm Watch (MAW) instruction.

Available Languages

Ladder Diagram

Use the Motion Disarm Watch (MDW) instruction to disarm watch-position event-checking for an axis. This instruction has the affect of clearing both the Watch Event Status and Watch Armed Status bits in the axis data structure. Executing this instruction also clears the In Process bit associated with the controlling Motion Arm Watch (MAW) instruction.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Function Block

This instruction is not available in function block.

Structured Text

MDW(Axis,MotionControl);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description
Axis	AXIS_CIP_DRIVE	AXIS_CIP_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE AXIS_GENERIC_DRIVE AXIS_GENERIC **Tip:** AXIS_GENERIC is supported by the ControlLogix 5570 and the GuardLogix 5570 controllers only.	Tag	Name of the axis on which to perform the operation. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set to true when the rung makes a false-to-true transition and remains set to true until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set to true when the axis watch event checking has been successfully disarmed.

.ER (Error) Bit 28	It is set to true to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MDW instruction cancels watch position event checking set up by a previous Motion Arm Watch (MAW). The Disarm Watch Position instruction requires no parameters; simply enter or select the desired physical axis.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

To successfully execute a MDW instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. Otherwise, the instruction errs.

Important:	The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis Operand. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See *Motion Error Codes (.ERR)* for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. See *Motion Error Codes (.ERR)* for Motion Instructions.

Status Bits

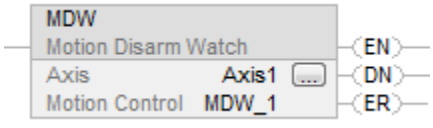
MDW Changes to Status Bits

Bit Name	State	Meaning
WatchEventArmedStatus	FALSE	The axis is not looking for a watch position event.
WatchEventStatus	FALSE	The previous watch event is cleared.

Examples

When the input conditions are true, the controller disarms watch-position event-checking for axis1.

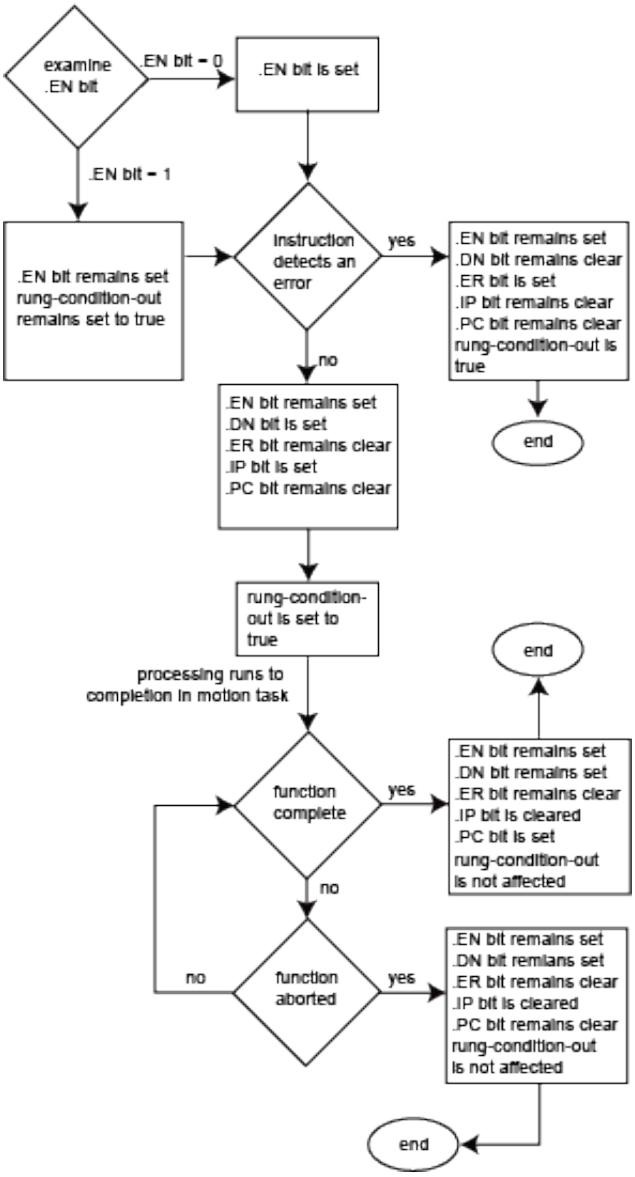
Ladder Diagram



Structured Text

MDW(Axis1,MDW_1);

MDW Flow Chart (True)



Motion Arm Registration (MAR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

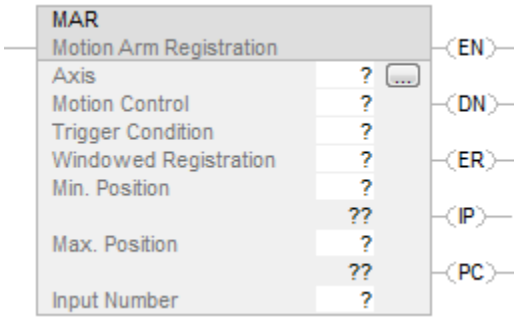
Use the Motion Arm Registration (MAR) instruction to arm registration event checking for the specified axis. When the instruction is called, a registration event is armed based on the selected Registration Input and the specified Trigger Condition. When the specified Registration Input transition satisfies the Trigger Condition, the axis computes the axis position at the moment the event occurred based on hardware latched encoder count data and stores it in the associated Registration Position variable in the axis data structure. Also, the instruction's Event (PC) bit is simultaneously set, as well as the Registration Event Status bit in the axis data structure. If Windowed Registration is selected, only registration events whose computed registration position falls within the Max and Min Position window are accepted. If the

Registration Position falls outside this window the registration event checking is automatically rearmed.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MAR(Axis,MotionControl, TriggerCondition, WindowedRegistration, MinimumPosition, MaximumPosition, InputNumber);

Operands

Ladder Diagram

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description
Axis	AXIS_CIP_DRIVE	AXIS_CIP_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE AXIS_GENERIC_DRIVE AXIS_GENERIC **Tip:** AXIS_GENERIC is supported by the ControlLogix 5570 and the GuardLogix 5570 controllers only.	Tag	Name of the axis on which to perform the operation. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Trigger Condition	BOOLEAN	BOOLEAN	Immediate	Defines the Registration Input transition that defines the registration event. Select either: 0 = trigger on positive edge 1 = trigger on negative edge.
Windowed Registration	BOOLEAN	BOOLEAN	Immediate	Enable (1) if registration is to be Windowed, that is, that the computed Registration Position must fall within the specified Min and Max Position limits to be accepted as a valid registration event. Select either: 0 = disabled 1 = enabled.
Minimum Position	REAL	REAL	Immediate or Tag	Used when Windowed Registration is enabled. Registration Position must be greater than Min. Position limit before registration event is accepted.
Maximum Position	REAL	REAL	Immediate or Tag	Used when Windowed Registration is enabled. Registration Position must be less than Max. Position limit before registration event is accepted.
Input Number	UINT32	UINT32	1 or 2	Specifies the Registration Input to select. 1 = Registration 1 Position 2 = Registration 2 Position.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

For the operands that require you to select from available options, enter your selection as:

This Operand	Has These Options Which You	
	Enter as Text	Or Enter as a Number
TriggerCondition	positive_edge negative_edge	0 1
WindowedRegistration	disabled enabled	0 1

MOTION_INSTRUCTION Structure

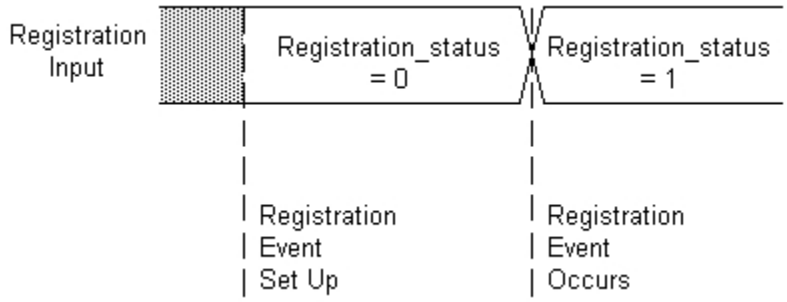
Mnemonic	Description
.EN (Enable) Bit 31	It is set to true when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set to true when the axis registration event checking has been successfully armed.
.ER (Error) Bit 28	It is set to to true to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.IP (In Process) Bit 26	It is set to true on positive rung transition and cleared to false after the registration event has occurred, or has been superseded by another Motion Arm Reg command, or terminated by a Motion Disarm Reg command.
.PC (Process Complete) Bit 27	It is set to true when a registration event occurs.

Description

The MAR instruction sets up a registration event to store the actual positions of the specified physical axis on the specified edge of the selected dedicated high speed Registration input for that axis.

When an MAR instruction is executed, the RegEventStatus bit is set to 0 (FALSE) and the selected Registration input for the specified axis is monitored until a Registration input transition of the selected type (the registration event) occurs. When the registration event occurs, the RegEventStatus bit for the axis is set to 1 (TRUE) and the Actual Position of the axis is stored in the Registration Position variable corresponding to the registration input (for example, Registration 1 Position 1 or Registration 2 Position).

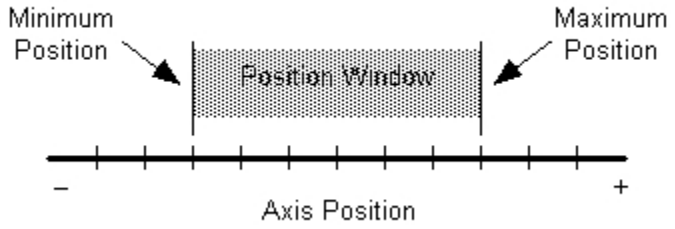
Registration



Multiple registration events may be active at any time for a given axis, but only one may be active per registration input. Each event is monitored independently and may be checked using the appropriate RegEventStatus bit.

Windowed Registration

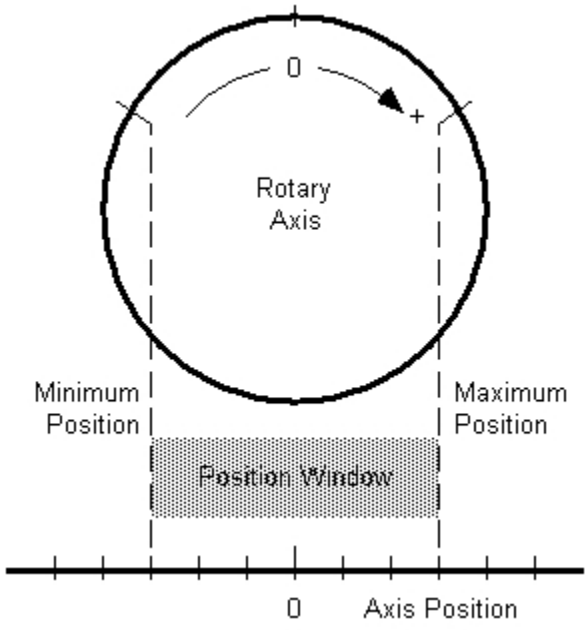
When the Windowed Reg checkbox is checked, the selected trip state only results in a registration event if it occurs when the axis is within the window defined by the minimum and maximum positions as shown below.



Enter values or tag variables for the desired absolute positions that define the position window within which the selected trip state of the Registration input is valid. Windowed registration is useful in providing a mechanism to ignore spurious or random transitions of the registration sensor, thus improving the noise immunity of high-speed registration inputs.

For linear axes, the values can be positive, negative, or a combination. However, the Minimum Position value must be less than the Maximum Position value for the registration event to occur. For rotary axes, both values must be less than the unwind value set in the motion controller's machine setup menu. The Minimum Position value can be greater than the Maximum Position value for registration windows that cross the unwind point of the axis, as shown below.

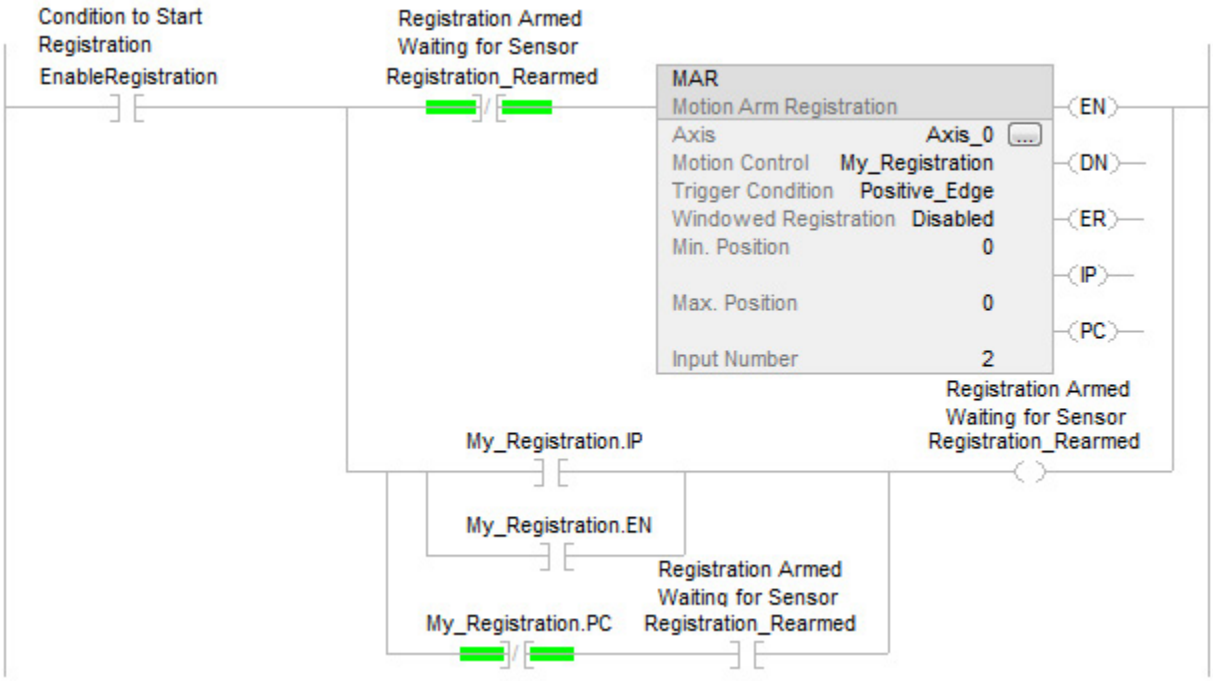
Position Window for Rotary Axis



Rearming an MAR Instruction

If your application requires rapid and continuous detection of a registration sensor, we recommend using this logic:

Ladder Logic for Continuous Registration Detection



To rearm the MAR instruction, the rung must change from false to true. The rate at which this logic functions depends on:

- program scan time
- motion task coarse update rate

IMPORTANT In large I/O connections, force values can slow down the rate at which the controller processes repetitive motion registration.

To successfully execute a MAR instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. Otherwise, the instruction errs.

IMPORTANT The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request completes.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis Operand. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See *Motion Error Codes (.ERR)* for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. The following Extended Error codes help to pinpoint the problem when the MAR instruction receives a Servo Message Failure (12) error message. See *Motion Error Codes (.ERR)* for Motion Instructions.

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	No Response (2)	Not enough memory resources to complete request. (SERCOS)
SERVO_MESSAGE_FAILURE (12)	Invalid value (3)	Registration input provided is out of range.
SERVO_MESSAGE_FAILURE (12)	Device in wrong state (16).	Redefine Position, Home, and Registration 2 are mutually exclusive. (SERCOS)

Extended Error codes for the Parameter Out of Range (13) error code work a little differently. Rather than having a standard enumeration, the number that appears for the Extended Error code refers to the number of the operand as they are listed in the faceplate from top to bottom with the first operand being counted as zero. Therefore for the MAR instruction, an extended error code of 4 would refer to the Min Position value. You would then have to check your value with the accepted range of values for the instruction.

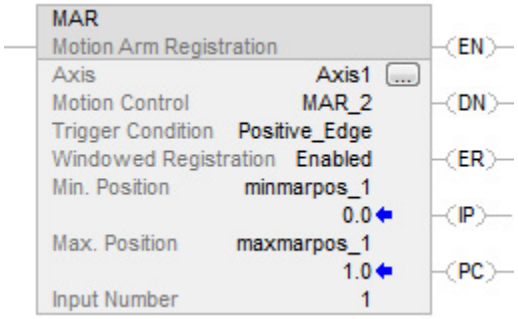
Status Bits

MAR Changes to Status Bits

Bit Name	State	Meaning
RegEvent1ArmedStatus RegEvent2ArmedStatus	TRUE	The axis is looking for a registration event.
RegEvent1Status RegEvent2Status	FALSE	The previous registration event is cleared.

Example

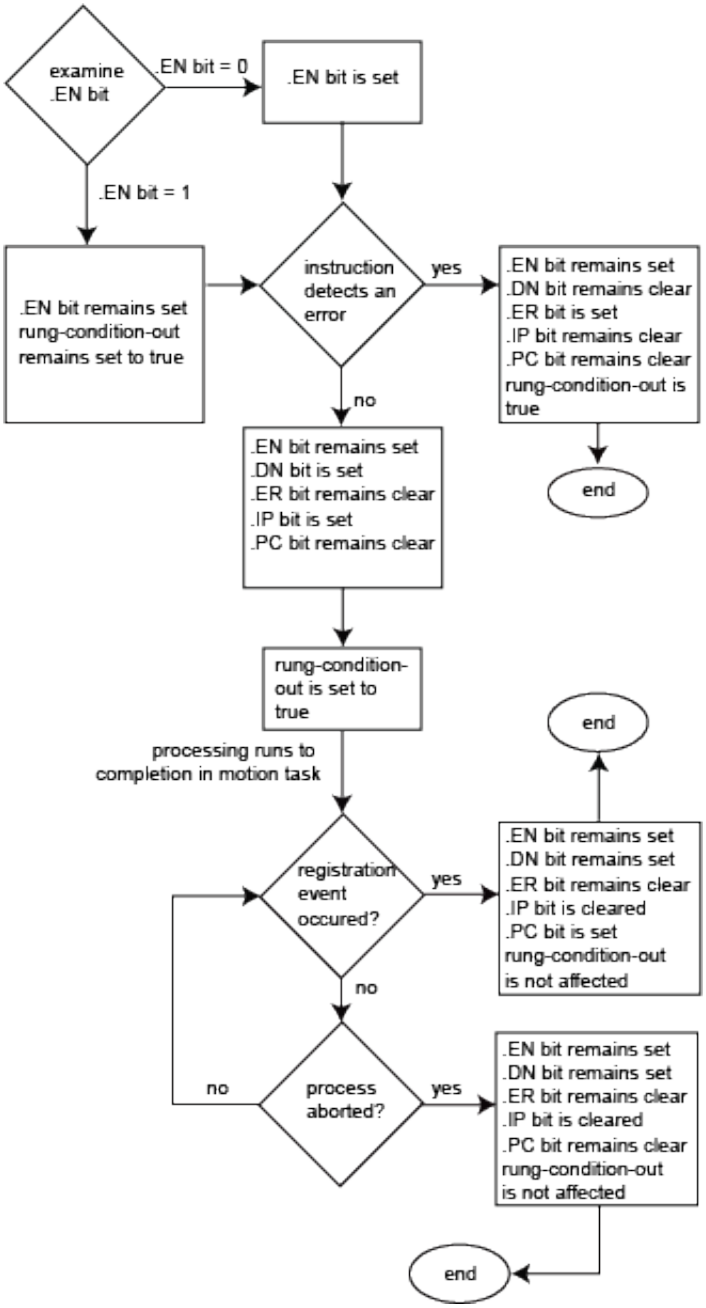
Ladder Diagram



Structured Text

MAR(Axis1, MAR_2, Positive_Edge, enabled, minmarpos_1, maxmarpos_1, 1);

MAR Flow Chart (True)



Motion Disarm Registration (MDR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Use the Motion Disarm Registration (MDR) instruction to disarm the specified motion module registration input event checking for the specified axis. This instruction has the effect of clearing both the RegEventStatus and the RegArmedEventStatus bits. The In Process bit of the controlling Motion Arm Registration instruction, if any, is cleared as a result of executing the MDR instruction.

Architecture	Standard	Safety

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block.

Structured Text

MDR(Axis,MotionControl, InputNumber);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description

Axis	AXIS_CIP_DRIVE	AXIS_CIP_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE AXIS_GENERIC_DRIVE AXIS_GENERIC Tip: AXIS_GENERIC is supported by the ControlLogix 5570 and the GuardLogix 5570 controllers only.	Tag	Name of the axis on which to to perform the operation. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Input Number	UINT32	UINT32	1 or 2	Specifies the Registration Input to select. 1 = Registration 1 Position 2 = Registration 2 Position

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set to true when the rung makes a false-to-true transition and remains set to true until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set to true when axis watch event checking has been successfully disarmed.
.ER (Error) Bit 28	It is set to true to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MDR instruction cancels registration event checking established by a previous Motion Arm Registration instruction. Only the registration checking associated with the specified registration input is disabled.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

To successfully execute a MDR instruction, the targeted axis must be configured as either a Servo or Feedback Only axis. Otherwise, the instruction errors.

IMPORTANT The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request completes.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis Operand. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See *Motion Error Codes (.ERR)* for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. The following Extended Error codes help to pinpoint the problem when the MDR instruction receives a Servo Message Failure (12) error message. See *Motion Error Codes (.ERR)* for Motion Instructions.

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	Invalid value (3)	Registration input provided is out of range.

Extended Error codes for the Parameter Out of Range (13) error code work a little differently. Rather than having a standard enumeration, the number that appears for the Extended Error code refers to the number of the operand as they are listed in the faceplate from top to bottom with the first operand being counted as zero. Therefore for the MDR instruction, an extended error code of 2 would refer to the Input Number operand's value. Input number is limited by the accepted range of values for the instruction and by the drive type. Some CIP drives allow 1 and 2, while other CIP drives will only allow 1.

Status Bits

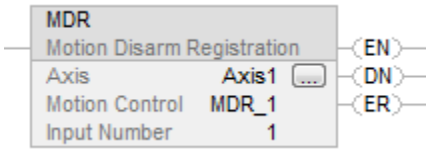
MDR Changes to Status Bits

Bit Name	State	Meaning
RegEventArmedStatus	FALSE	The axis is not looking for a registration event.
RegEventStatus	FALSE	The previous registration event is cleared.

Example

When the input conditions are true, the controller disarms registration-event checking for axis2.

Ladder Diagram



Structured Text

```
MDR(Axis1, MDR_1, 1);
```

Motion Arm Output Cam (MAOC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

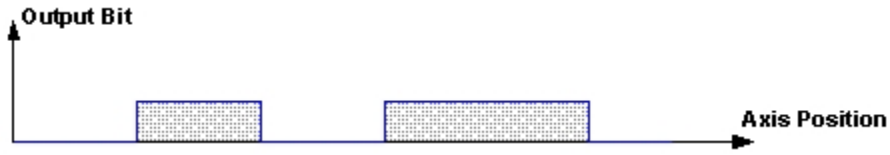
The Motion Arm Output Cam (MAOC) instruction offers the functionality to set and reset output bits based on an axis position.

This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.
-

IMPORTANT The Scheduled Output Module can be associated with one (1) MAOC axis/execution target only.

Motion Arm Functionality



Internally, Output Cam objects handle the Motion arm Output Cam functionality. Each Output Cam object is responsible for one output, which consists of 32 output bits. Each single output bit can be programmed separately with an Output Cam profile, and compensated for position offset and time delay.

The MAOC instruction initiates the arming of a specific Output Cam between the designated axis and output. When executed, the specified output cam bits are synchronized to the designated axis using an Output Cam Profile established by the Logix Designer Output Cam Editor. This relationship can be viewed as a master/slave relationship with the axis representing the master and the output bit representing the slave. Hence, the Output Cam functionality is related to the position cam functionality, which provides a relationship between a master axis and a slave axis.

To accurately synchronize the output cams to the designated axis, an execution schedule and associated axis and cam arm positions are specified. When the axis travels past the axis arm position in the direction specified by the Execution Schedule parameter, the cam position becomes locked to the axis position starting at the specified Cam Arm Position parameter. At this time, the output cam is armed and the Output Cam Armed status is set. The output cam can also be configured via the Execution Schedule parameter to execute Immediately or Pending completion of a currently executing output cam. The output cam can also be executed Once, Continuously or Persistently by specifying the desired Execution Mode. Persistent behavior allows the output cam to become disarmed when the cam position exceeds the output cam range, and rearmed when cam position returns to within range.

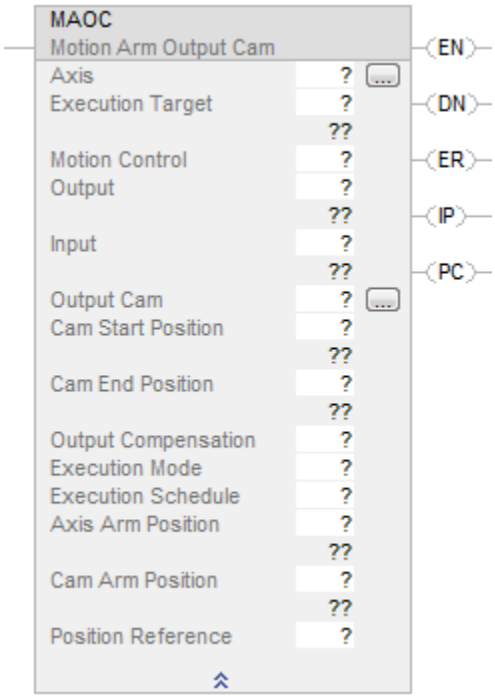
Output Cam range is defined by input parameters CamStartPosition and CamEndPosition. The Master Reference selection allows axis input to be derived from either the Actual or Commanded position of the designated axis.

IMPORTANT Output cams increase the potential for exceeding coarse update rate. This can cause misbehavior if the motion task execution time exceeds the configured group coarse update period. The only way to check on this condition is to monitor the max execution time from the **Motion Group Properties** dialog box.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MAOC(Axis, ExecutionTarget, MotionControl, Output, Input, OutputCam, CamStartPosition, CamEndPosition, OutputCompensation, ExecutionMode, ExecutionSchedule, AxisArmPosition, CamArmPosition, Reference);

Operands

There are data conversion rules for mixed data types within an instruction. See *Data Conversion*.

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description
Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_CONSUMED **Tip:** AXIS_CONSUMED is supported by Compact GuardLogix 5580, CompactLogix 5380, and CompactLogix 5480 controllers only.	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_CONSUMED AXIS_SERVO AXIS_SERVO_DRIVE AXIS_GENERIC_DRIVE AXIS_GENERIC **Tip:** AXIS_GENERIC is supported by the ControlLogix 5570 and the GuardLogix 5570 controllers only.	Tag	Name of the axis that provides the position input to the Output Cam. Ellipsis launches Axis Properties dialog. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Execution Target	SINT, INT, or DINT	SINT, INT, or DINT	Immediate or Tag	The execution target defines the specific Output Cam from the set connected to the named axis. Behavior is determined by the following: 0...7 – Output Cams executed in the Logix controller. 8...31 – Reserved for future use.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

Output	DINT	DINT	Tag	A set of 32 output bits that are set or reset based on the specified Output Cam. It can be either a memory location or a physical output. If Pending is selected as the Execution Schedule, then Output is ignored.
Input	DINT	DINT	Tag	A set of 32 input bits that can be used as enable bits depending on the specified Output Cam. It can be either a memory location or a physical input. If Pending is selected as the Execution Schedule, then Input is ignored.
Output Cam	OUTPUT_CAM	OUTPUT_CAM	Array Tag	An array of OUTPUT_CAM elements. The elements do not need to be ordered and the array size is determined by the number of cam elements specified. The array size is limited by the available memory of the Logix controller.
Cam Start Position	SINT, INT, DINT, or REAL	SINT, INT, DINT, or REAL	Immediate or Tag	Cam Start Position with the Cam End Position define the left and right boundaries of the Output Cam range.
Cam End Position	SINT, INT, DINT, or REAL	SINT, INT, DINT, or REAL	Immediate or Tag	Cam End Position with the Cam Start Position define the left and right boundaries of the Output Cam range.
Output Compensation	OUTPUT_COMPENSATION	OUTPUT_COMPENSATION	Array Tag	Is an array of 1 to 32 OUTPUT_COMPENSATION elements. The array indices correspond to the output bit numbers. The minimum size of an array is determined by the highest compensated output bit.

Execution Mode	SINT, INT, or DINT	SINT, INT, or DINT	Immediate	There are three (3) possible execution modes. The behavior is determined by the mode selected. The options are: 0 = Once – Output Cam is disarmed and the Process Complete Bit of the Motion Instruction is set when the cam position moves beyond the cam start or the cam end position. 1 = Continuous – Output Cam continues on the opposite side of the Output Cam range when the cam position moves beyond the cam start or the cam end position. 2 = Persistent – Output Cam disarms when the cam position moves beyond the cam start or the cam end position. The Output Cam is rearmed when the cam position moves back into the Output Cam range.
Execution Schedule	SINT, INT, or DINT	SINT, INT, or DINT	Immediate	Selects when to arm the Output Cam. Options are: 0 = Immediate – Output Cam is armed at once. 1 = Pending – Output cam is armed when the cam position of a currently executing Output Cam moves beyond its cam start or cam end position. When Pending is selected the following parameters are ignored Output, Input, Axis Arm Position, and Reference. 2 = Forward only – Output Cam is armed when the axis approaches or passes through the specified axis arm position in the forward direction. 3 = Reverse only – Output Cam is armed when the axis approaches or passes through the specified axis arm position in the reverse direction. 4 = Bi-directional – Output Cam is armed when the axis approaches or passes through the specified axis arm position in either direction.

Axis Arm Position	SINT, INT, DINT, or REAL	SINT, INT, DINT, or REAL	Immediate or Tag	This defines the axis position where the Output Cam is armed when the Execution Schedule is set to Forward Only, Reverse Only, or Bi-Directional and the axis moves in the specified direction. If Pending is selected as the Execution Schedule, then Axis Arm Position is ignored.
Cam Arm Position	SINT, INT, DINT, or REAL	SINT, INT, DINT, or REAL	Immediate or Tag	This defines the cam position associated with the axis arm position when the Output Cam is armed.
ReferenceF	SINT, INT, or DINT	SINT, INT, or DINT	Immediate	Sets whether the Output Cam is connected to either Command position or Actual position of the axis. If Pending is selected as the Execution Schedule, then Reference is ignored. 0 = Actual – the current position of the axis as measured by its encoder or other feedback device. 1 = Command – the desired or commanded position of the master axis.

Structured Text

For the array operands, you do not have to include the array index. If you do not include the index, the instruction starts with the first element in the array ([0]). See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

For the operands that require you to select from available options, enter your selection as:

This Operand	Has These Options Which You	
	Enter as Text	Or Enter as a Number
ExecutionMode	once continuous persistent	0 1 2
ExecutionSchedule	immediate pending forwardonly reverseonly bidirectional	0 1 2 3 4
Reference	actual command	0 1

MAOC Instruction

A valid Cam Arm position is any position, between and including, the Cam Start and Cam End positions. If the Cam Arm position is set to a value equal to (or very close to) the Cam Start or Cam End position, compensation may put a cam position out of range of the Cam Start and Cam End position. Compensation is affected by Output Compensation values specified for Position Offset, Latch Delay, and Unlatch Delay, as well as internal compensation values applied based on the Reference and Output parameters of the MAOC instruction.

No side effects occur if the MAOC instruction is configured with an Execution mode of Continuous or Persistent, and a pending MAOC instruction does not exist when the Output Cam is armed and the axis moves.

These side effects may occur if the MAOC instruction is configured with an Execution Mode of Once Only, and a pending MAOC exists when the Output Cam is armed and the axis moves.

One or more outputs may never change state.

The MAOC instruction may complete immediately.

One possible side effect of a pending MAOC instruction existing when the Output Cam is armed and the axis moves is that one or more outputs could begin executing based on the configuration of the pending MAOC instruction.

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set to true when the rung makes a false-to-true transition and remains set until the rung goes false.
.DN (Done) Bit 29	It is set to true when Output Cam has been successfully initiated.
.ER (Error) Bit 28	It is set to true to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.IP (In Process) Bit 26	It is set to true when the Output Cam has been initiated successfully and cleared to false if either superseded by another Motion Arm Output Cam command, terminated by a Motion Disarm Output Cam command, or cam position moves beyond defined Output Cam range while execution mode is set to once.
.PC (Process Complete) Bit 27	It is cleared to false on positive rung transition and set in once Execution Mode when cam position moves beyond defined Output Cam range.
.SEGMENT	It is set to the array index associated with error 36 (Illegal Output Cam) or error 37 (Illegal Output Compensation). Only the first of multiple errors is stored.

Description

The Motion Arm Output Cam (MAOC) instruction executes an output cam profile set up manually, programmatically, or by the Logix Designer Output Cam Editor. Internally, Output Cam objects handle Motion arm Cam functionality. Each Output Cam object is responsible for one output, which consists of 32 output bits. Each single output bit can be programmed separately. Currently Output Cam functionality is executed in the Logix controller every coarse update period (currently configurable between 1 and 32 ms).

Axis

The axis provides the position input to the Output Cam. The axis can be a virtual, physical or consumed one.

Execution Target

The execution target defines a specific Output Cam from the set that is connected to the specified axis. Currently, only eight Output Cams can be specified.

Specifying the Output Cam

To execute a MAOC instruction, a calculated Output Cam data array tag must be specified. Output Cam array tags may be created by the Logix Designer tag editor or the MAOC instruction using the built-in Output Cam Editor. The data defines the specifics for each Output Cam element. The number of Output Cam elements is limited by the amount of available memory. Zero or more cams can be defined for each output bit. There is no constraint on how these elements are arranged within the Output Cam array.

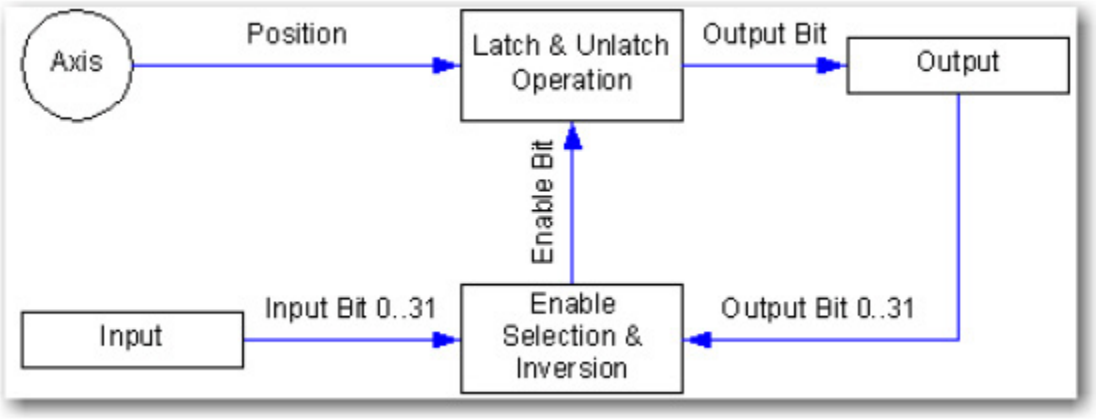
RefAn anomaly occurs when the output CAM ON window positions are redefined while the output controlled by the output CAM element is active. In some instances, the Motion Planner may not detect the off-crossing of the window and the output controlled by the output CAM element remains ON.er to the description of the OUTPUT_CAM structure for more information about data types and programming units.

IMPORTANT An anomaly occurs when the output CAM ON window positions are redefined while the output controlled by the output CAM element is active. In some instances, the Motion Planner may not detect the off-crossing of the window and the output controlled by the output CAM element remains ON.

This issue applies to any output point or virtual output controlled by an MAOC instruction.

Additionally, we recommend that you only change configuration when the CAM element is not active.

This diagram shows the relationships between the axis, input, and output that are defined by the Output Cam element.

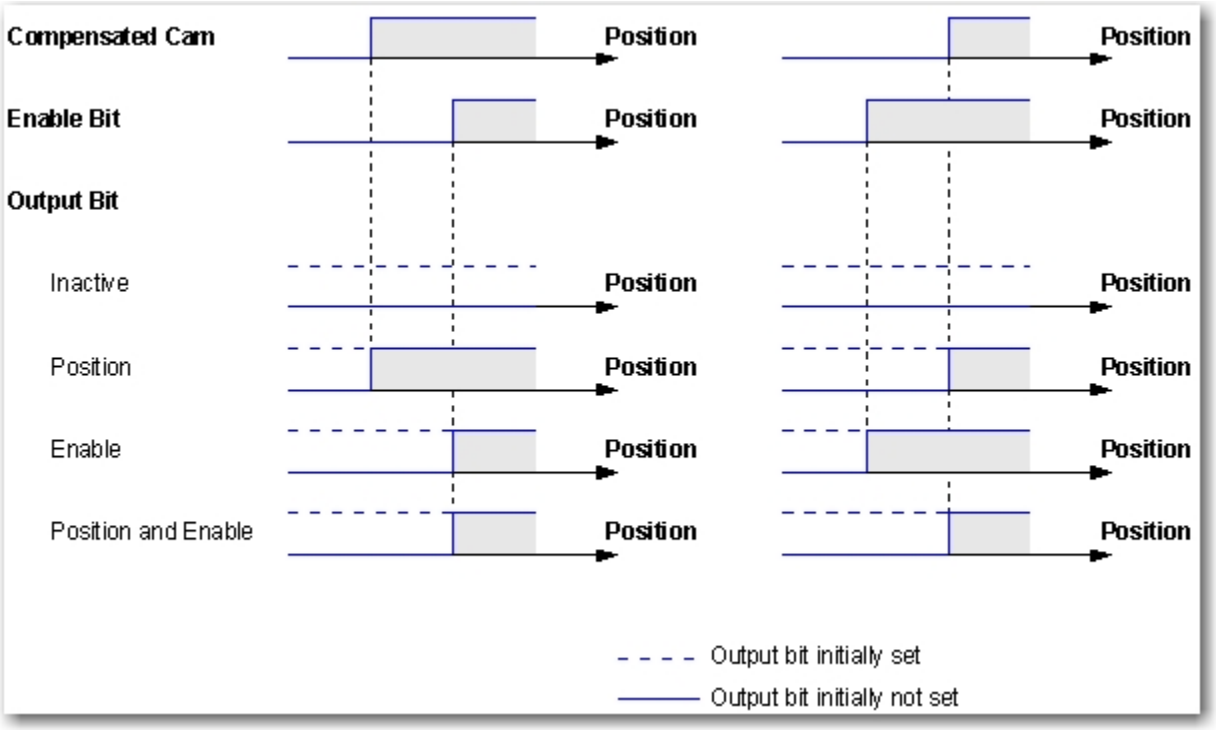


Latch Type

Depending on the selected LatchType, the corresponding output bit is set according to this table:

Latch Type	Behavior
Inactive	The output bit is not changed.
Position	The output bit is set when the axis enters the compensated cam range.
Enable	The output bit is set when the enable bit becomes active.
Position and Enable	The output bit is set when the axis enters the compensated cam range and the enable bit becomes active.

This diagram shows the effect of the selected latch type on the output bit for different compensated cam and enable bit combinations as function of position.

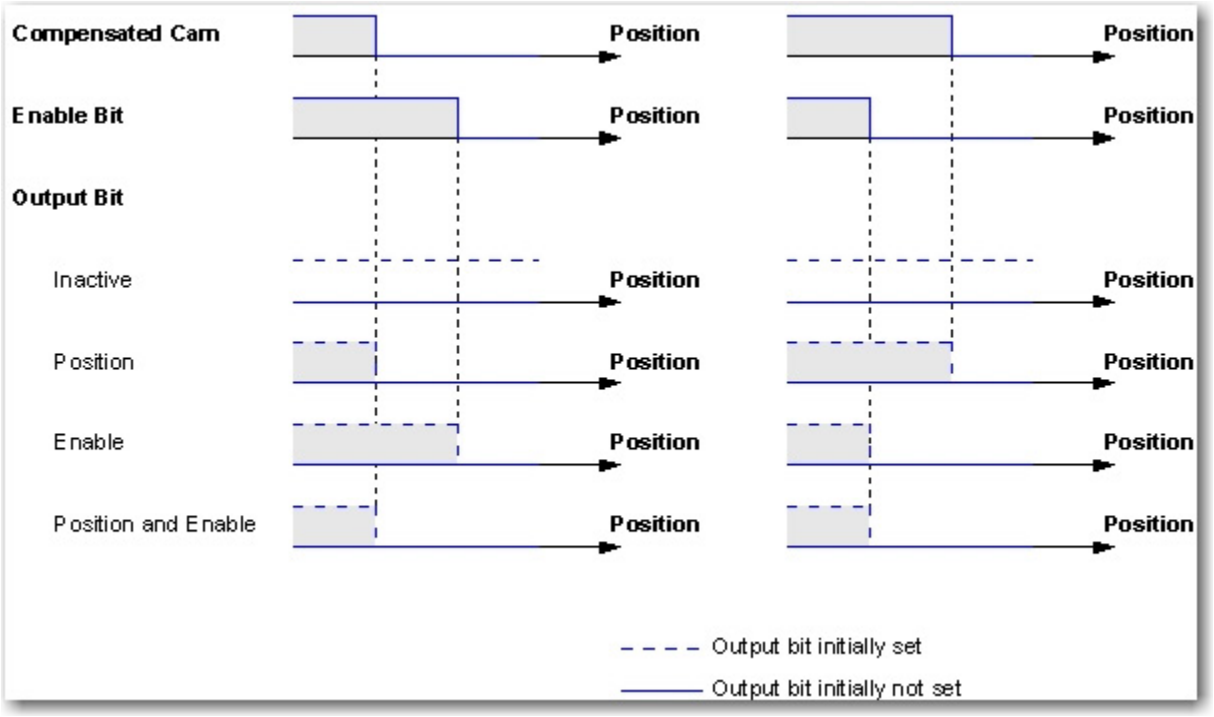


Unlatch Type

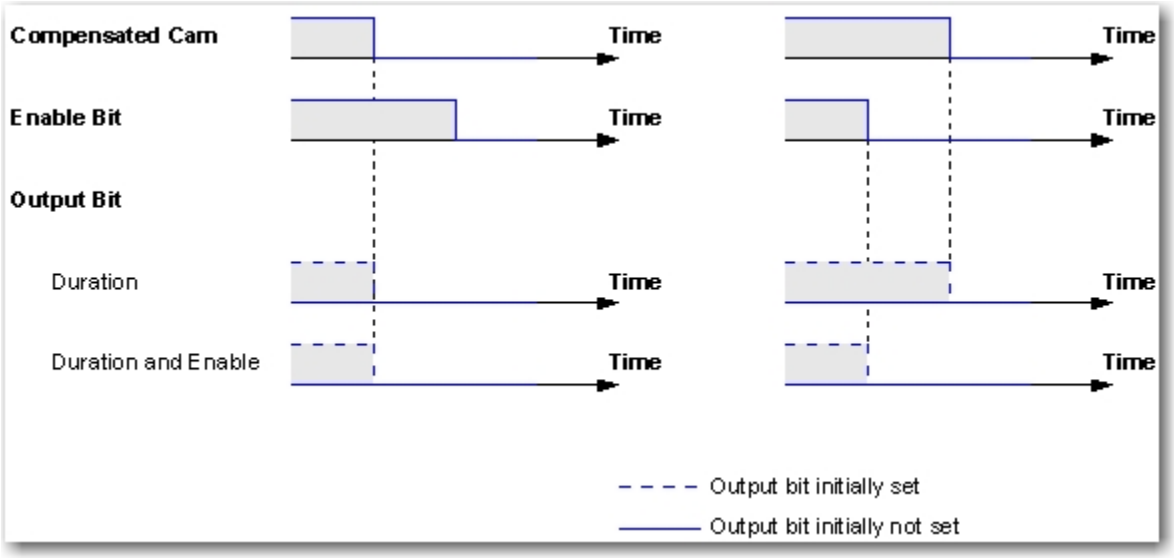
This diagram shows the effect of the selected latch type on the output bit for different compensated cam and enable bit combinations as function of position.

Latch Type	Behavior
Inactive	The output bit is not changed.
Position	The output bit is reset when the axis enters the compensated cam range.
Duration	The output bit is reset when the duration expires.
Enable	The output bit is reset when the enable bit becomes active.
Position and Enable	The output bit is reset when the axis leaves the compensated cam range or the enable bit becomes active.
Duration and Enable	The output is reset when the duration expires or the enable bit becomes inactive.

This diagram shows the effect of the selected unlatch type on the output bit for different compensated cam and enable bit combinations as function of position.



This diagram shows the effect of the selected unlatch type on the output bit for different compensated cam and enable bit combinations as function of time.



Left and Right Cam Positions

The Left and Right cam positions define the range of an Output Cam element. If the latch or unlatch type is set to "Position" or "Position and Enable" with the

enable bit active, the left and right cam positions specify the latch or unlatch position of the output bit.

Duration

If the unlatch type is set to "Duration" or "Duration and Enable" with the enable bit active, the cam duration specifies the time between the latching and the unlatching of the output bit.

Enable Type

Depending on the selected enable type, the enable bit is an element of either the input, inverted input, output, or inverted output.

Output Cam Array Checks

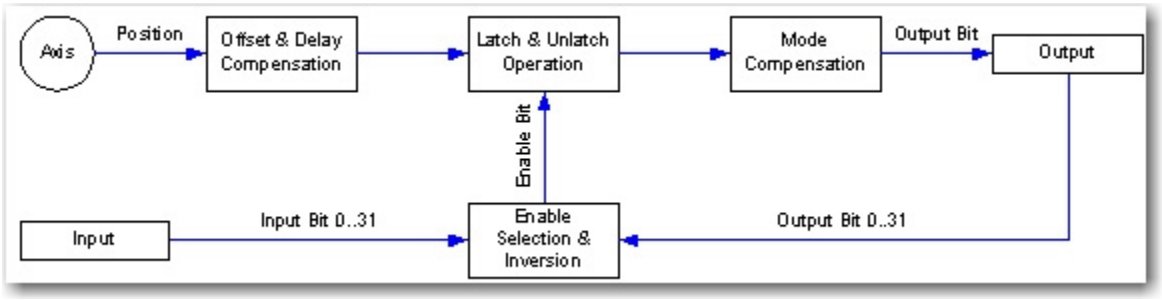
These output cam array checks are used with the MAOC instruction:

- If you select an output bit less than 0 or greater than 31, the Output Cam element is not considered and the user is warned with an instruction error Illegal Output Cam.
- If you select a latch type less than 0 or greater than 3, a value of "Inactive" is used and the user is warned with an instruction error Illegal Output Cam.
- If you select an unlatch type less than 0 or greater than 5, a value of "Inactive" is used and the user is warned with an instruction error Illegal Output Cam.
- If you select a left cam position greater than or equal to the right cam position and the latch or unlatch type is set to Position or Position and Enable, the Output Cam element is not considered and the user is warned with an instruction error Illegal Output Cam.
- If you select a left cam position less than the cam start position and the latch type is set to Position or Position and Enable, the cam start position is used and the user is warned with an instruction error Illegal Output Cam.
- If you select a right cam position greater than the cam end position and the unlatch type is set to Position or Position and Enable, the cam end position is used and the user is warned with an instruction error Illegal Output Cam.
- If you select a duration less than or equal to 0 and the unlatch type is set to Duration or Duration and Enable, the Output Cam element is not considered and the user is warned with an instruction error Illegal Output Cam.

- If you select an enable type less than 0 or greater than 3 and the latch or unlatch type is set to Enable, Position and Enable, or Duration and Enable, the Output Cam element is not considered and the user is warned with an instruction error Illegal Output Cam.
- If you select an enable bit less than 0 or greater than 31 and the latch or unlatch type is set to Enable, Position and Enable, or Duration and Enable, the Output Cam element is not considered and the user is warned with an instruction error Illegal Output Cam.
- Specifying Output Compensation

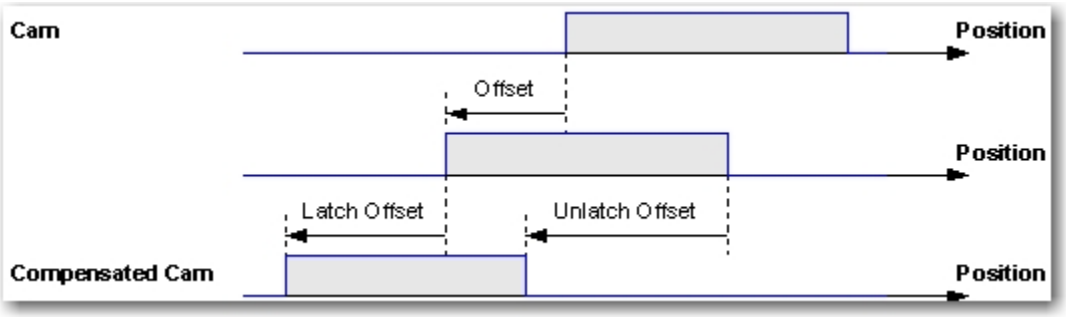
An Output Compensation data array tag may be specified via the Logix Designer tag editor. The data type defines the specifics for each output bit by specifying the characteristics of each actuator. The array indices correspond to the output bit numbers. The number of the highest compensated output bit defines the minimum size of this array. Changes to the output compensation take effect immediately.

This diagram shows the effect of the output compensation on the relationships between the axis, input, and output



Offset and Delay Compensation

The offset provides position compensation, while the latch and unlatch delay provides time delay compensation for the latch and unlatch operation. The following diagram shows the effect of the compensation values on an Output Cam element.



The cam range is defined by the left and right cam positions of the Output Cam element. The compensated cam range is defined by the cam range, offset, and latch and unlatch offsets. The latch and unlatch offsets are defined by the current speed v.

Latch Offset = v * Latch Delay

Unlatch Offset = v * Unlatch Delay

The resulting compensation offset can actually be larger than the difference between cam start and cam end position.

This equation illustrates the effect of the compensation values on the duration of an Output Cam element

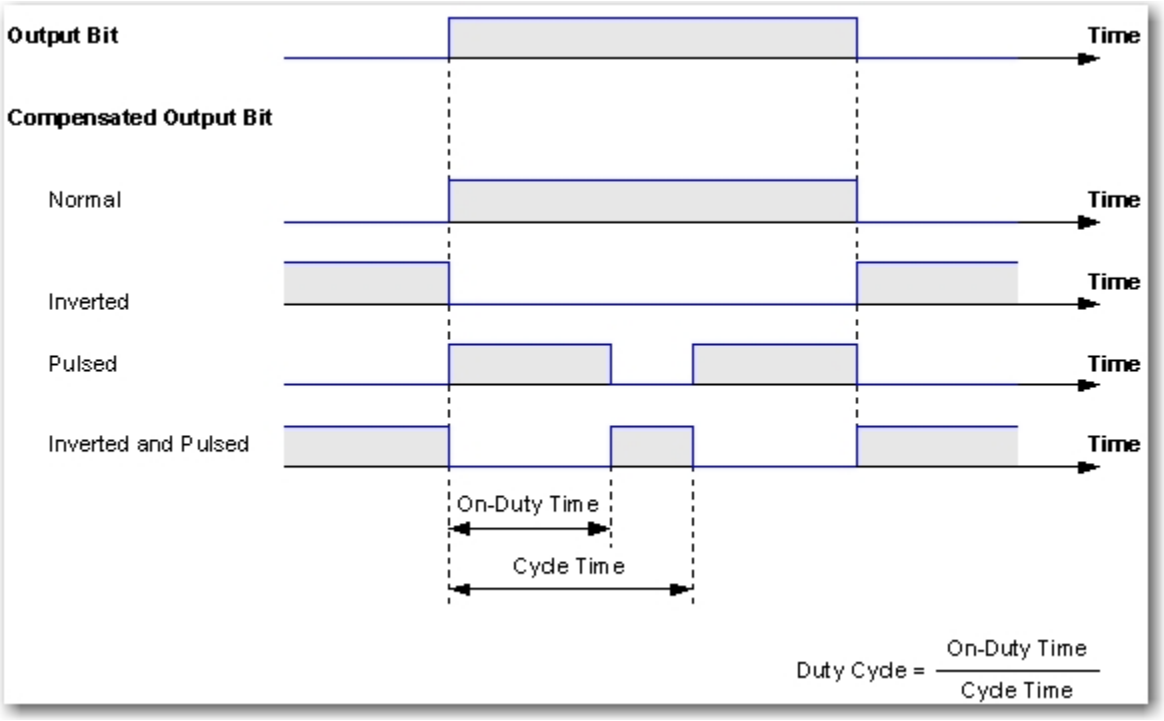
Compensated Duration = Duration + Latch Delay - Unlatch Delay

Mode Compensation

Depending on the selected mode, the compensated output bit is set according to the following table.

Mode	Behavior
Normal	The output bit is set, when the output of the latch and unlatch operation becomes active. The output bit is reset, when the output of the latch and unlatch operation becomes inactive.
Inverted	The output bit is set, when the output of the latch and unlatch operation becomes inactive. The output bit is reset, when the output of the latch and unlatch operation becomes active.
Pulsed	The output bit is pulsed, when the output of the latch and unlatch operation is active. The on-duty state of the pulse corresponds to the active state of the output bit. The output bit is reset, when the output of the latch and unlatch operation becomes inactive.
Inverted and Pulsed	The output bit is pulsed, when the output of the latch and unlatch operation is active. The on-duty state of the pulse corresponds to the inactive state of the output bit. The output bit is set, when the output of the latch and unlatch operation becomes inactive.

This diagram shows the effect of the mode, cycle time, and duty cycle on an output bit.



Output Compensation Array Checks

This output compensation array checks are used with the MAOC instruction.

If you select a latch and unlatch delay combination that results in a compensated cam of less than minimum width, the width of the compensated cam is set to the minimum.

If you select a mode less than 0 or greater than 3, a Normal mode is considered and you are warned with an instruction error Illegal Output Compensation.

If you select a duty cycle less than 0 or greater than 100 and the mode is set to Pulsed or Inverted and Pulsed, a 0 or 100 duty cycle is considered and you are warned with an instruction error Illegal Output Compensation.

If you select a cycle time less than or equal to 0 and the mode is set to Pulsed or Inverted and Pulsed, the output bit is not pulsed and you are warned with an instruction error Illegal Output Compensation.

Output

The output is the set of 32 output bits that can be set and reset depending on the specified Output Cam. The output can be either a memory location or a physical output (for example, Local.0.O.Data).

Input

The input is the set of 32 input bits that are can be used as enable bits depending on the specified Output Cam. The input can be either a memory location or a physical input (for example, Local.0.I.Data).

Cam Start and Cam End Positions

The cam start and cam end positions define the left and right boundary of the Output Cam range. When the cam position moves beyond the cam start or cam end position, the behavior of the Output Cam is defined by the execution mode and execution schedule. Changes to the cam start or cam end position don't take effect until the execution of a current MAOC instruction completes.

Execution Mode

Depending on the selected execution mode, the Output Cam behavior may differ, when the cam position moves beyond the cam start or cam end position.

Mode	Behavior
Once	When the cam position moves beyond the cam start or cam end position, the Output Cam is disarmed and the Process Complete bit of the Motion Instruction is set.
Persistent	When the cam position moves beyond the cam start or cam end position, the Output Cam is disarmed. However, when the cam position moves back into the Output Cam range the Output Cam is rearmed.
Continuous	When the cam position moves beyond the cam start or cam end position, the Output Cam continues on the opposite side of the Output Cam range.

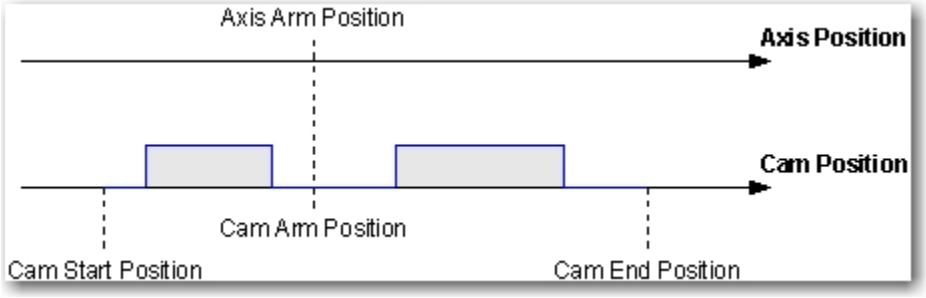
Execution Schedule

Depending on the selected execution schedule, the Output Cam is armed according to the following table.

Mode	Behavior
Immediate	The Output Cam is armed immediately.
Pending	The Output Cam is armed, when the cam position of an armed Output Cam moves beyond its cam start or cam end position.
Forward Only	The Output Cam is armed, when the axis approaches or passes through the specified axis arm position in the forward direction.
Reverse Only	The Output Cam is armed, when the axis approaches or passes through the specified axis arm position in the reverse direction.
Bidirectional	The Output Cam is armed, when the axis approaches or passes through the specified axis arm position in the forward or reverse direction.

Axis Arm and Cam Arm Positions

The axis arm position defines the axis position where the Output Cam is armed, if the execution schedule is set to either forward only, reverse only, or bi-directional and the axis moves in the specified direction. The cam arm position defines the cam position that is associated with the axis arm position, when the Output Cam is armed. Changes to the axis arm or cam arm position only take effect after the execution of an MAOC instruction.



Reference

Depending on the selected reference, the Output Cam is connected to either the actual or command position of the axis.

Important: The MAOC instruction execution completes in a single scan, thus the Done (.DN) bit and the In Process .IP bit are set immediately. The In Process .IP bit remains set until the cam position moves beyond the cam start or cam end position in Once execution mode, is superseded by another MAOC instruction or is disarmed by the MDOC instruction. The Process Complete bit is cleared immediately when the MAOC executes and set when the cam position moves beyond the cam start or cam end position in Once execution mode.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis Operand. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is set to true. Otherwise, the .EN bit is not affected. The .DN,.ER ,.IP and .PC bits are not affected.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See *Motion Error Codes (.ERR)* for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. Extended Error codes for the Parameter Out of Range (13) error code lists a number that refers to the number of the operand as they are listed in the faceplate from top to bottom with the first operand being counted as zero. Therefore for the MAOC instruction, an extended error code of 4 would refer to the Output operand's value. You would then have to check your value with the accepted range of values for the instruction. See *Motion Error Codes (.ERR)* for Motion Instructions.

Execution

If ERR is	And EXERR is	Then Cause	Corrective Action
36	Varies	The size of the Output Cam array is not supported or the value of at least one member is out of range: Output bit less than 0 or greater than 31. Latch type less than 0 or greater than 3. Unlatch type less than 0 or greater than 5. Left or right position is out of cam range and the latch or unlatch type is set to "Position" or "Position and Enable". Duration less than or equal to 0 and the unlatch type is set to "Duration" or "Duration and Enable". Enable type less than 0 or greater than 3 and the latch or unlatch type is set to "Enable", "Position and Enable", or "Duration and Enable". Enable bit less than 0 or greater than 31 and the latch or unlatch type is set to "Enable", "Position and Enable", or "Duration and Enable". Latch type is set to "Inactive" and unlatch type is set to either "Duration" or "Duration and Enable".	Illegal Output Cam
37	Varies	Either the size of the Output Compensation array is not supported or the value of one of its members is out of range.	Illegal Output Compensation
The array index associated with errors 36 and 37 are stored in .SEGMENT of the Motion Instruction data type. Only the first of multiple errors are stored. The specific error detected is stored in Extended Error Code. With the ability to dynamically modify the Output Cam table, the Illegal Output Cam error 36 may occur while the MAOC is in-process. In general, the cam elements in which an error was detected will be skipped. The following are exceptions, and will continue to be processed. Error 2, Latch Type Invalid. Latch Type defaults to Inactive. Error 3, Unlatch Type Invalid. Unlatch Type defaults to Inactive. Error 8, with Unlatch Type of Duration and Enable. Will behave as an Enable Unlatch type.			

Status Bits

MAOC Effects Status Bits

Status bits may be used to determine if an MAOC instruction can be initiated. The MAOC instruction affects the following status words in the Motion Axis Structure:

- OutputCamStatus
- OutputCamPendingStatus
- OutputCamLockStatus
- OutputCamTransitionStatus

Each above is a DINT with bits 0 to 7 corresponding to the 8 execution targets. Bit 0 is execution target 0; bit 1 is execution target 1, etc.

If the Execution Schedule is set to Forward Only, Reverse Only or Bi-Directional, an MAOC instruction can be initiated when either of the following two conditions exist:

OutputCamStatus bit corresponding to execution target = FALSE

OutputCamStatus bit corresponding to execution target = TRUE
OutputCamLockStatus bit corresponding to execution target = FALSE
OutputCamTransitionStatus bit corresponding to execution target = FALSE

If the Execution Schedule is Pending, the MAOC instruction is initiated if either of the following two conditions exist:

OutputCamStatus bit corresponding to execution target = FALSE

OutputCamStatus bit corresponding to execution target = TRUE
OutputCamTransitionStatus bit corresponding to execution target = FALSE

Axis and Module Fault Conditions Disarm Output Cams

When the controller detects one of the following faults, it disarms output cams:

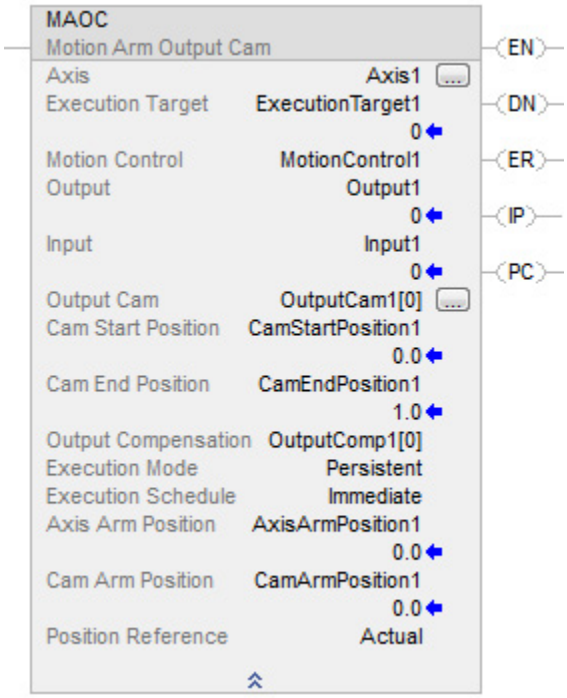
- For Axis_Servo and Axis_Servo_Drive, axis feedback loss fault
- For Axis_Servo and Axis_Servo_Drive, module fault
- For Axis_Consumed, physical axis fault

Those faults produce unreliable feedback data.

Also, if an axis fault exists when an MAOC instruction is initiated, the instruction errs.

Example

Ladder Diagram



Structured Text

MAOC(Axis1, ExecutionTarget1, MotionControl1, Output1, Input1, OutputCam1[0], CamStartPosition1, CamEndPosition1, OutputComp1[0], Persistent, Immediate, AxisArmPosition1, CamArmPosition1, actual);

Scheduled Output Module

The 1756-OB16IS, 1732E-OB8M8SR, and 1756-OB16IEFS Scheduled Output modules are designed to work in conjunction with the MAOC motion instruction to provide position based output control (also known as PLS). The MAOC instruction by itself allows position based output control by using the position of any motion axis in ControlLogix, and CompactLogix for the 1732E-OB8M8SR module, as the position reference and any output or boolean as the output. The MAOC updates the outputs based on motion axis position at the motion group coarse update rate (typically 2ms to 10ms). The output turn-on and turn-off times are determined by axis position and velocity.

To schedule the on/off times of the scheduled outputs of the module in resolutions down to that rated for the module, outputs are scheduled as follows by the available modules.

For the 1756-OB16IS, of 8 of its 16 outputs (outputs 0 to 7) in 100 µs increments. Outputs are scheduled by entering data into one or more of the 16 schedules provided by the output connection data store.

For the 1732E-OB8M8SR, of 8 of its 8 outputs (outputs 0 to 7) in 100 µs increments. Outputs are scheduled by entering data into one or more of the 16 schedules provided by the output connection data store.

For the 1756-OB16IEFS, of 16 of its 16 outputs (outputs 0 to 15) in 10 µs increments. Outputs are scheduled by entering data into one or more of the 32 schedules provided by the output connection data store.

IMPORTANT When using the 1756-OB16IS module with the MAOC instruction, make sure you use the default Communication Format for the module, that is, Schedule Output Data Per Point. If you change the Communication Format when the module is used with an MAOC instruction, an error may result.

Operation

Outputs are scheduled on a per point basis and each individual output point is controlled by its own timestamp.

Individual schedules are created in the controller, stored in the output image table for the module, and sent over the backplane or EtherNet/IP to the Scheduled Output module. The schedule specifies a sequence count, the output point to be associated with the schedule, the time at which an output value should be applied to the physical output point, and the value to be applied at the scheduled time. The I/O module receives and stores the schedule. The timestamp of each schedule is monitored by the module. When a schedule has expired, that is, the current time matches the scheduled timestamp, the output value is then applied to the corresponding output bit. Timer hardware in the ASIC is used to optimize the scheduling algorithm. This hardware also reduces the latency and jitter performance. Status of each schedule is reported in the output echo connection and reflected in the input image for the module.

The scheduled output functionality relies on CST (Coordinated System Time) timestamp in the 1756-OB16IS module. At least one controller in the chassis must be a CST time master.

The 1756-OB16IEFS and 1732E-OB8M8SR modules rely on UTC time with a CIP Sync Grandmaster clock synchronizing all the outputs.

For the 1756-OB16IS module, unused outputs may be used as normal outputs and are applied immediately rather than waiting for the CST timestamp to expire. A mask is sent to the module to indicate which outputs are to function as normal outputs.

Each scheduled output module can be individually scheduled. Refer to the following table for the scheduled support that is available for each module type.

Module	Total Outputs	Scheduled Outputs	Total Schedules	Minimum Scheduled Interval	Timebase
1756-OB16IS	16 [0 to 15]	8 [0 to 7]	16	100µs	CST
1732E-OB8M8SR	8 [0 to 7]	8 [0 to 7]	16	100µs	UTC
1756-OB16IEFS	16 [0 to 15]	16 [0 to 15]	32	5µs	UTC

The scheduled outputs must be between output points. Outputs that are not scheduled are used as normal output points. A mask is used to indicate which points are scheduled and which points are unscheduled. All of the scheduling configuration is done through the MAOC instruction.

If a new schedule as indicated by a change in the sequence count is received by the I/O module before the current schedule has expired, the current schedule is overwritten. This mechanism can be used to cancel currently active schedule. Status bits returned in the output echo connection may be used to determine the current state of each schedule and to trigger corresponding event tasks.

If a new schedule is sent by the controller to a 1756-OB16IS and the CST timestamp has already past, the output is asserted until the lower 24 bits of the CST time has completely wrapped around. The module does not check for an expired CST timestamp.

WARNING: If the time between two schedules is less than the minimum schedule interval, then jitter occurs. This means that even though two outputs are scheduled at different time, they both activate at the same time.

Remote Operation

Scheduled outputs using the 1756-OB16IS module do not work with a remote chassis.

Usage with MAOC Instruction

When used with motion and the MAOC instruction, values in the output image are controlled by the Motion Planner firmware in the controller. The Motion Planner triggers the data to be sent to the module. Although, the normal program or task scan also triggers data to be sent to the module. Data integrity is maintained by the firmware always setting the sequence count for a given schedule last.

The Output Cam instruction processes cam events for scheduled outputs one coarse update period sooner than unscheduled outputs. When a programmed on or off event is detected, a schedule is sent to the output module to turn the

output on or off at the appropriate time within the next coarse update period. The Output Cam instruction divides the coarse update period into sixteen time slots.

For example, a coarse update period of 2 ms will yield sixteen 125 µs time slots. Cam on/off events will be assigned to time slots based on their position within the coarse update period. If both latch and unlatch events for a cam element are assigned to the same time slot, they will cancel each other out. This implies that the minimum pulse width of a cam element is greater than one time slot.

The minimum pulse width of a cam element should be greater than the 100 µs OB16IS minimum pulse width, or the 1/16 coarse update minimum pulse width, whichever is larger.

The MAOC instruction detects latch and unlatch events one coarse update ahead and schedules the event to occur within the next coarse update. This is accomplished by applying a one coarse update internal delay to each scheduled output latch and unlatch position. When the latch or unlatch event is detected, the delta time from the start of the coarse update to the event is calculated, and the output is scheduled to occur at the CST or UTC corresponding to the next coarse update period. To facilitate this, Output Cam functionality has access to the CST or UTC captured when the current coarse update period occurred.

The MAOC instruction is able to process both scheduled and unscheduled output bits for the 1756-OB16IS.

Scheduled outputs that are not configured for use in an MAOC's Output Cam Table are not allocated in the resultant Schedule Mask. You can manipulate those outputs as general discrete outputs from separate logic. What this implies is that for the 1756-OB16IS module, you cannot directly affect output bits 0...7 or 0...15, but you do have the ability to modify output bits 8...15.

Tip: For the 1756-OB16IS module, outputs 0 to 7 can be forced by forcing the Data Bit to 0 or 1 and its corresponding bit in the ScheduleMask to 0. For outputs 8 to 15, only the Data Bit needs to be forced. For the 1732E-OB8M8SR and 1756-OB16IEFS modules, force outputs 0 to 7 and outputs 0 to 15 in the same manner that you use for the 1756-OB16IS module.

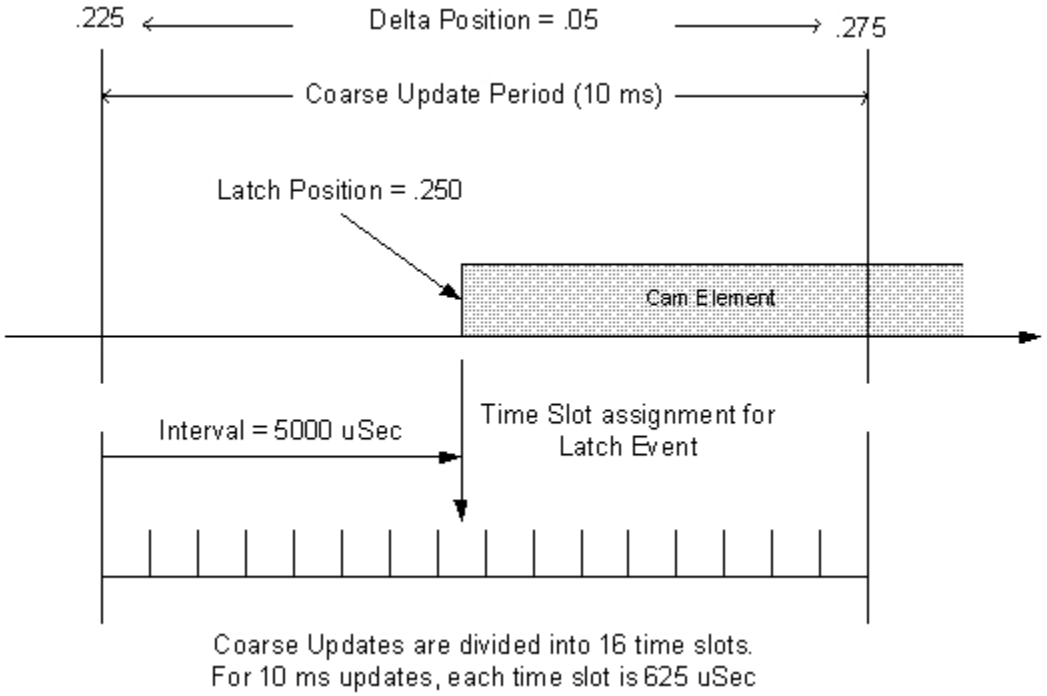
Due to the limit of 16 schedules supported by the 1756-OB16IS and 1732E-OB8M8SR modules and 32 for the 1756-OB16IEFS module, some constraints are applied to the number of events that can be processed every coarse update period.

Only 8 schedules are available each coarse update for the 1756-OB16IS and 1732E-OB8M8SR modules. This allows for two consecutive coarse updates in which each update contains 8 output events. As a group of 8 schedules are currently being processed by the scheduled output modules, a second group of 8 schedules can concurrently be set up for the next coarse update.

Similarly, only 16 schedules are available each coarse update for the 1756-OB16IEFS module. This allows for two consecutive coarse updates in which each update contains 16 output events. As a group of 16 schedules are

currently being processed by the scheduled output modules, a second group of 16 schedules can concurrently be set up for the next coarse update.

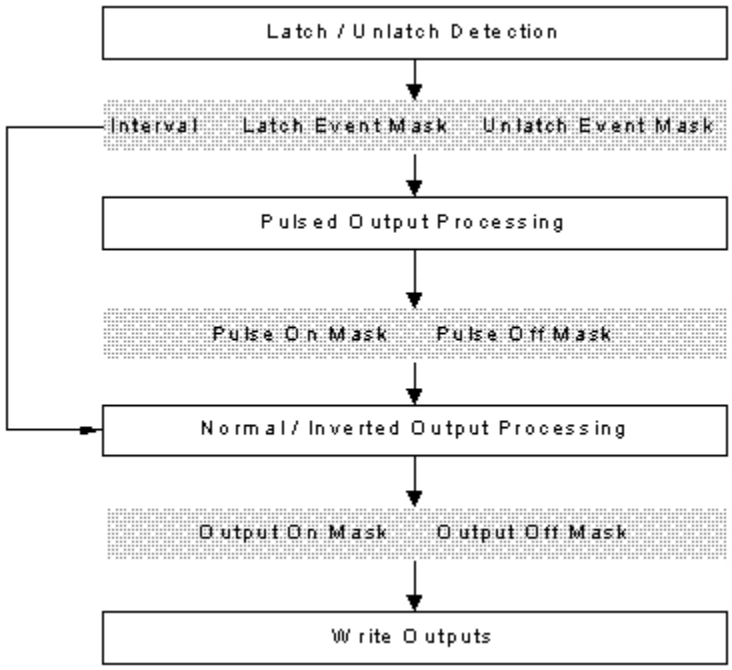
This diagram illustrates the relationship between the coarse update period, a cam latch event and the time slots for the 1756-OB16IS.



Each Time Slot stores the information described in this table.

Topic	Description
Latch Event Mask	When a latch event is detected, the time slot in which it belongs is calculated and the bit in the Latch Event Mask corresponding to the output bit of the latch is set.
Unlatch Event Mask	When an unlatch event is detected, the time slot in which it belongs is calculated and the bit in the Unlatch Event Mask corresponding to the output bit of the unlatch is set.
Interval	The time in micro-seconds from the start of the coarse update in which the Latch or Unlatch event occurs.
Pulse On Mask	For pulsed outputs, the time slot in which a pulse on event is calculated and the bit in the Pulse On Mask corresponding to the output bit of the pulse event is set.
Pulse Off Mask	For pulsed outputs, the time slot in which a pulse off event is calculated and the bit in the Pulse Off Mask corresponding to the output bit of the pulse event is set.
Output On Mask	For normal outputs, the bit corresponding to the output bit of the Latch or Pulse On event is set indicating that the output is to be turned on for these events. For inverted outputs, the bit corresponding to the output bit of the Unlatch or Pulse Off event is set indicating that the output is to be turned on for these events.
Output Off Mask	For normal outputs, the bit corresponding to the output bit of the Unlatch or Pulse Off event is set indicating that the output is to be turned off for these events. For inverted outputs, the bit corresponding to the output bit of the Latch or Pulse On event is set indicating that the output is to be turned off for these events.

This is a simplified overview of how Time Slot data is utilized.



Time slots are also used to process overlapping cam elements. A semaphore is maintained to indicate the currently active state of each output bit. In addition, if a programmed cam element Latch and Unlatch event occurs in the same time slot, they cancel each other out.

The minimum width of a cam element corresponds to the width of a time slot, or 1/16 the coarse update period.

I/O Subsystem

You can specify the Output parameter of an MAOC instruction as either a memory tag or an Output Module's data tag. A pointer to the tag is passed into the MAOC instruction. Also passed into the MAOC instruction is an internal parameter of type IO_MAP. If the Output parameter references controller memory, the IO_MAP parameter is NULL. If the Output parameter references an output module, the IO_MAP parameter points to the map structure for the module. The MAOC instruction can then determine if the Output parameter is associated with a scheduled output module by checking the module type stored in the driver table.

Output Data Structure

Field	Size	Description
Value	4 bytes	Data values for un-scheduled output bits. 0 = Off 1 = On

Mask	4 bytes	Selects which output bits are to be scheduled. Only the first eight bits (0 to 7) or 16 bits (0 to 15) can be scheduled. 0 = Not scheduled 1 = Scheduled
ScheduleTimestamp	8 bytes	1756-OB16IEFS and 1732E-OB8M8SR only. Course update timestamp in UTC time.

Array of 16 Schedule Structures

Field	Size	Description
Schedule ID	1 byte	Valid ID's are 1 to 16 for the 1756-OB16IS and 1732E-OB8M8SR modules, 1...32 for the 1756-OB16IEFS module. Any other value indicates that the schedule is not to be considered.
Sequence Number	1 byte	The scheduled output modules will maintain a copy of the schedule. A change in sequence number will tell the modules to process the data in this schedule.
Point ID	1 byte	Indicates the output bit associated with this schedule. Entered as a value 00 to 07 for the 1756-OB16IS and 1732E-OB8M8SR modules and 00 to 15 for the 1756-OB16IEFS module.
Point Value	1 byte	Next state of output bit specified in Point ID. 0 = Off 1 = On
Timestamp	4 bytes	1756-OB16IS - The lower 32 bits of CST. 1756-OB16IEFS, 1732E-OB8M8SR - Offset from coarse update timestamp (Scheduled time = ScheduleTimestamp + Time Stamp). Indicates when to change the state of the specified output bit.

Schedule Processing

The Value and Mask fields are processed and all unscheduled data bits are moved to the module output data store. This data is written to the output terminals after all schedules have been processed. Each schedule is processed. The schedule is not considered if the:

- Schedule ID is not in the range of 1 to 16 or 1 to 32.
- Point ID is not in the range of 0 to 7 or 0 to 15.
- Sequence Number has not changed.

If the schedule is to be considered, it is marked active. All active schedules are examined every several microseconds. The schedule Timestamp is compared to the current time. If the current time is greater than or equal to the scheduled Timestamp, the Point Value in the schedule is moved to the module output data store for the specified output bit.

See also

Motion Arm Output Cam (MAOC) on page 273

Specifying the Output Cam

To execute a MAOC instruction, a calculated Output Cam data array tag must be specified. Output Cam array tags may be created by the Logix Designer application tag editor or the MAOC instruction using the built-in Output Cam

Editor. The data defines the specifics for each Output Cam element. The number of Output Cam elements is limited by the amount of available memory. Zero or more cams can be defined for each output bit. There is no constraint on how these elements are arranged within the Output Cam array.

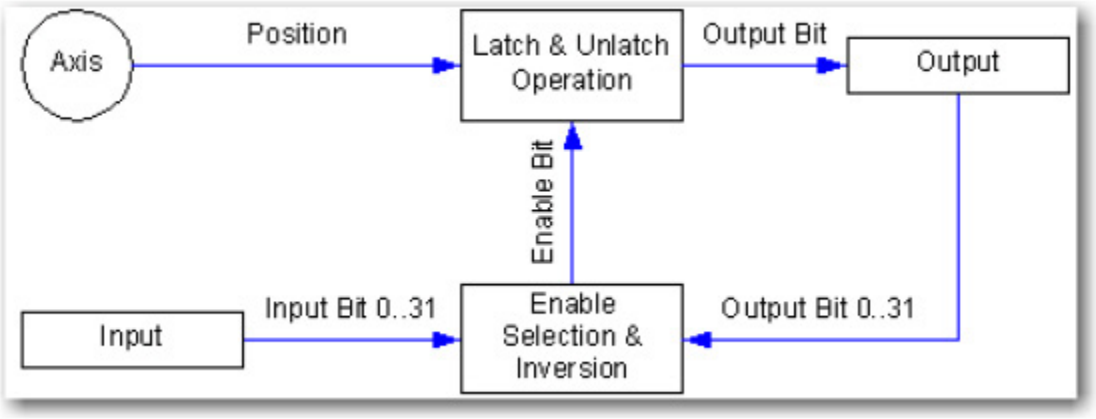
Refer to the description of the OUTPUT_CAM structure for more information about data types and programming units.

IMPORTANT An anomaly occurs when the output CAM ON window positions are redefined while the output controlled by the output CAM element is active. In some instances, the Motion Planner may not detect the off-crossing of the window and the output controlled by the output CAM element remains ON.

This issue applies to any output point or virtual output controlled by an MAOC instruction.

Additionally, we recommend that you only change configuration when the CAM element is not active.

This diagram shows the relationships between the axis, input, and output that are defined by the Output Cam element.

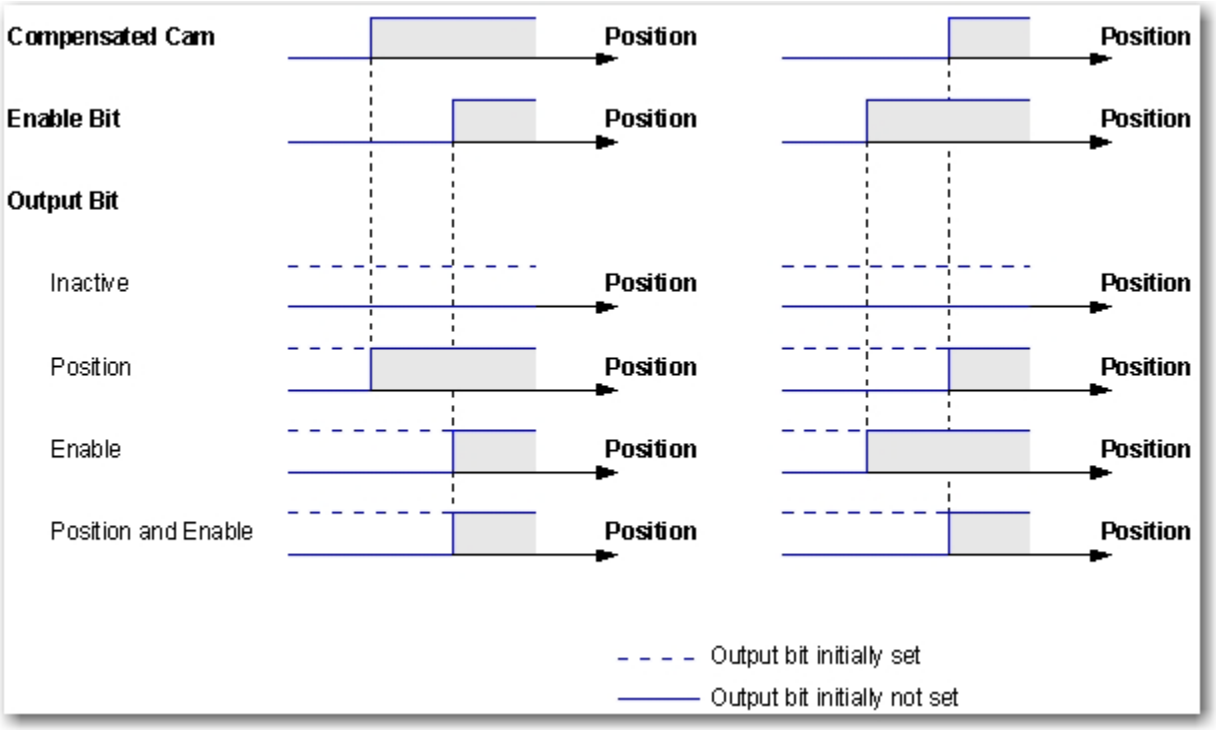


Latch Type

Depending on the selected LatchType, the corresponding output bit is set according to this table:

Latch Type	Behavior
Inactive	The output bit is not changed.
Position	The output bit is set when the axis enters the compensated cam range.
Enable	The output bit is set when the enable bit becomes active.
Position and Enable	The output bit is set when the axis enters the compensated cam range and the enable bit becomes active.

This diagram shows the effect of the selected latch type on the output bit for different compensated cam and enable bit combinations as function of position.

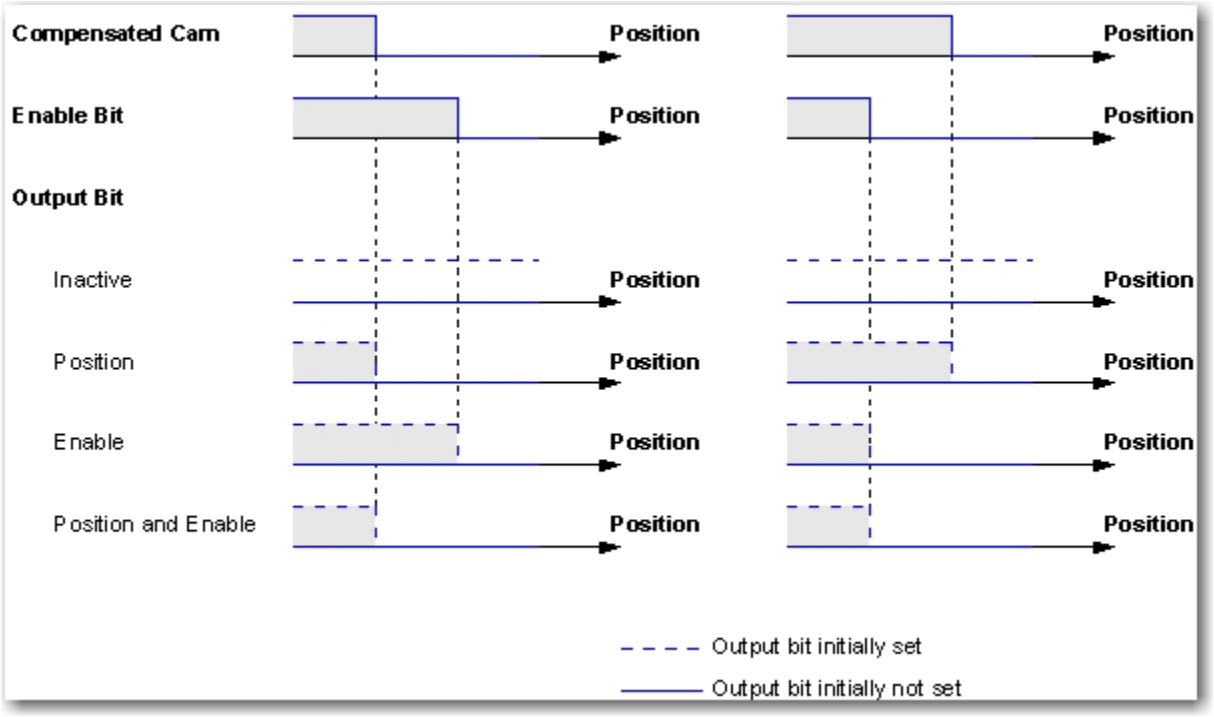


Unlatch Type

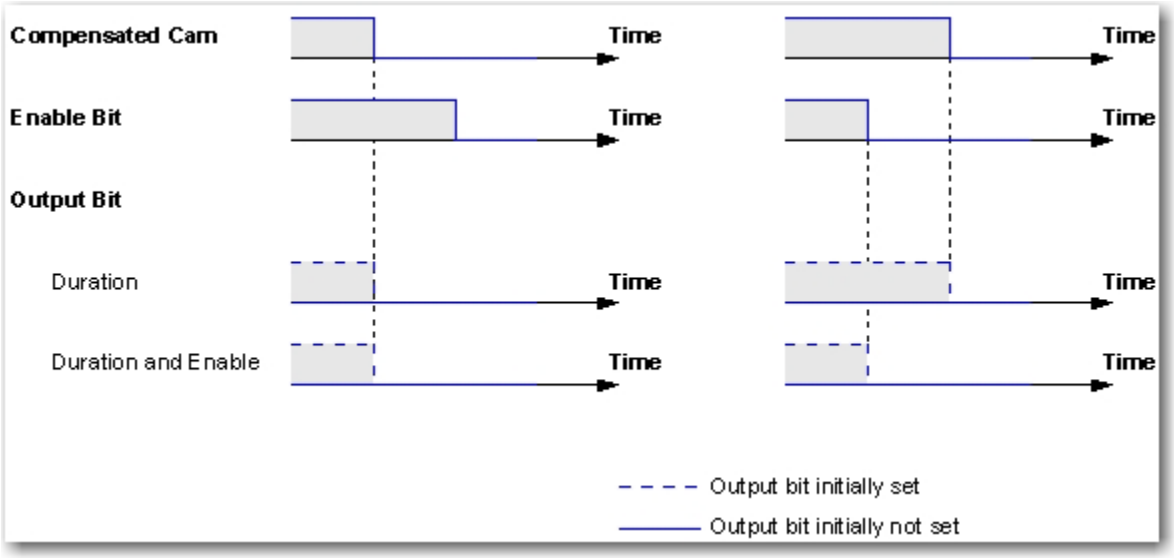
This diagram shows the effect of the selected latch type on the output bit for different compensated cam and enable bit combinations as function of position.

Latch Type	Behavior
Inactive	The output bit is not changed.
Position	The output bit is reset when the axis enters the compensated cam range.
Duration	The output bit is reset when the duration expires.
Enable	The output bit is reset when the enable bit becomes active.
Position and Enable	The output bit is reset when the axis leaves the compensated cam range or the enable bit becomes active.
Duration and Enable	The output is reset when the duration expires or the enable bit becomes inactive.

This diagram shows the effect of the selected unlatch type on the output bit for different compensated cam and enable bit combinations as function of position.



This diagram shows the effect of the selected unlatch type on the output bit for different compensated cam and enable bit combinations as function of time.



Left and Right Cam Positions

The Left and Right cam positions define the range of an Output Cam element. If the latch or unlatch type is set to Position or Position and Enable with the

enable bit active, the left and right cam positions specify the latch or unlatch position of the output bit.

Duration

If the unlatch type is set to Duration or Duration and Enable with the enable bit active, the cam duration specifies the time between the latching and the unlatching of the output bit.

Enable Type

Depending on the selected enable type, the enable bit is an element of either the input, inverted input, output, or inverted output.

Output Cam Array Checks

This output cam array checks are used with the MAOC instruction.

If you select	Then	Instruction error
an output bit less than 0 or greater than 31	the Output Cam element is not considered	Illegal Output Cam
a latch type less than 0 or greater than 3	a value of Inactive is used	
an unlatch type less than 0 or greater than 5	a value of Inactive is used	
a left cam position greater than or equal to the right cam position and the latch or unlatch type is set to Position or Position and Enable	the Output Cam element is not considered	
a left cam position less than the cam start position and the latch type is set to Position or Position and Enable	the cam start position is used	
a right cam position greater than the cam end position and the unlatch type is set to Position or Position and Enable	the cam end position is used	
a duration less than or equal to 0 and the unlatch type is set to Duration or Duration and Enable	the Output Cam element is not considered	
an enable type less than 0 or greater than 3 and the latch or unlatch type is set to Enable, Position and Enable, or Duration and Enable	the Output Cam element is not considered	
an enable bit less than 0 or greater than 31 and the latch or unlatch type is set to Enable, Position and Enable, or Duration and Enable	the Output Cam element is not considered	

See also

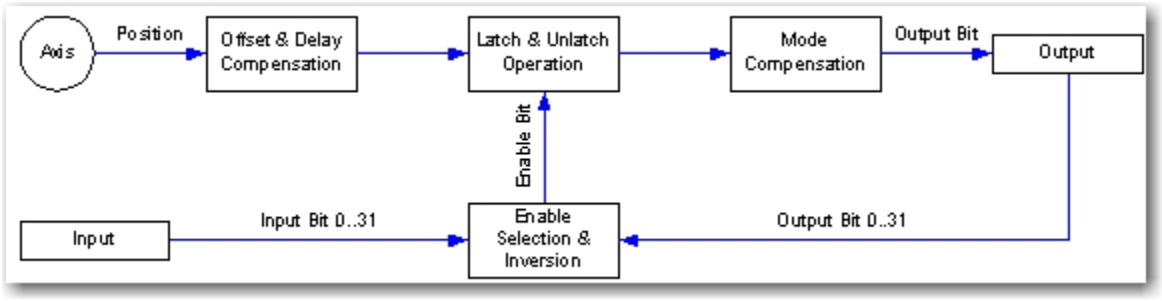
Motion Arm Output Cam (MAOC) on page 273

Specifying Output Compensation

An Output Compensation data array tag may be specified via the Logix Designer application tag editor. The data type defines the specifics for each output bit by specifying the characteristics of each actuator. The array indices correspond to the output bit numbers. The number of the highest

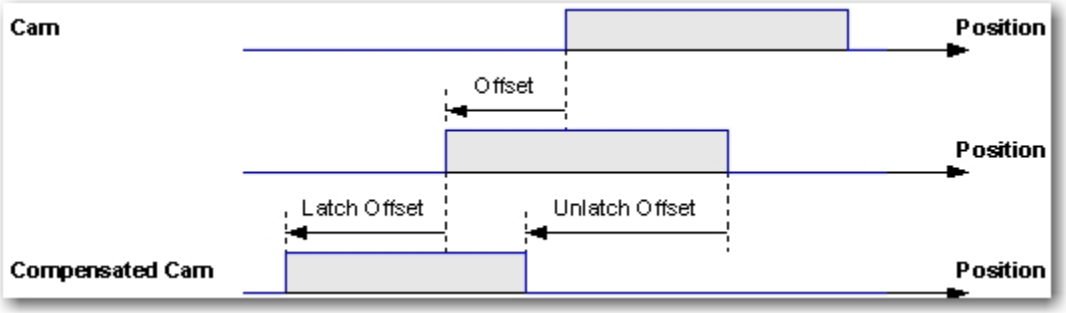
compensated output bit defines the minimum size of this array. Changes to the output compensation take effect immediately.

This diagram shows the effect of the output compensation on the relationships between the axis, input, and output



Offset and Delay Compensation

The offset provides position compensation, while the latch and unlatch delay provides time delay compensation for the latch and unlatch operation. This diagram shows the effect of the compensation values on an Output Cam element.



The cam range is defined by the left and right cam positions of the Output Cam element. The compensated cam range is defined by the cam range, offset, and latch and unlatch offsets. The latch and unlatch offsets are defined by the current speed v.

Latch Offset = v * Latch Delay

Unlatch Offset = v * Unlatch Delay

The resulting compensation offset can actually be larger than the difference between cam start and cam end position.

This equation illustrates the effect of the compensation values on the duration of an Output Cam element

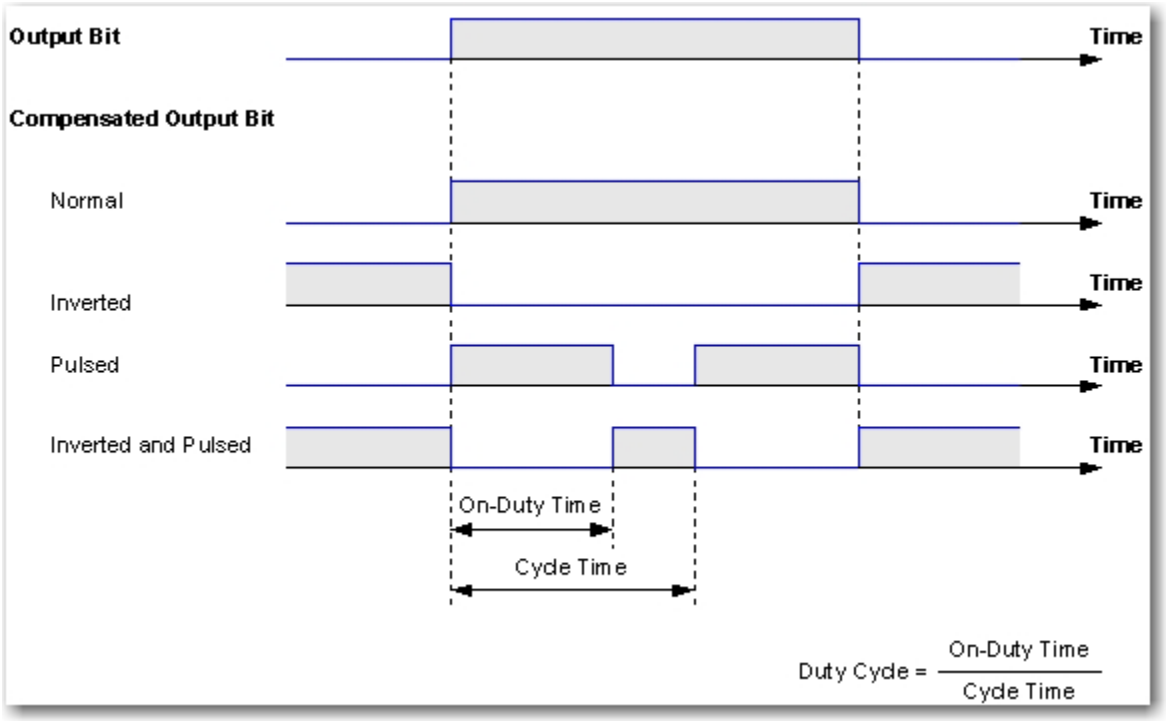
Compensated Duration = Duration + Latch Delay - Unlatch Delay

Mode Compensation

Depending on the selected mode, the compensated output bit is set according to this table.

Mode	Behavior	When
Normal	The output bit is set	the output of the latch and unlatch operation becomes active.
	The output bit is reset	the output of the latch and unlatch operation becomes inactive.
Inverted	The output bit is set	the output of the latch and unlatch operation becomes inactive.
	The output bit is reset	the output of the latch and unlatch operation becomes active.
Pulsed	The output bit is pulsed	when the output of the latch and unlatch operation is active. The on-duty state of the pulse corresponds to the active state of the output bit.
	The output bit is reset	the output of the latch and unlatch operation becomes inactive.
Inverted and Pulsed	The output bit is pulsed	the output of the latch and unlatch operation is active. The on-duty state of the pulse corresponds to the inactive state of the output bit.
	The output bit is set	the output of the latch and unlatch operation becomes inactive.

This diagram shows the effect of the mode, cycle time, and duty cycle on an output bit.



Output Compensation Array Checks

This output compensation array checks are used with the MAOC instruction.

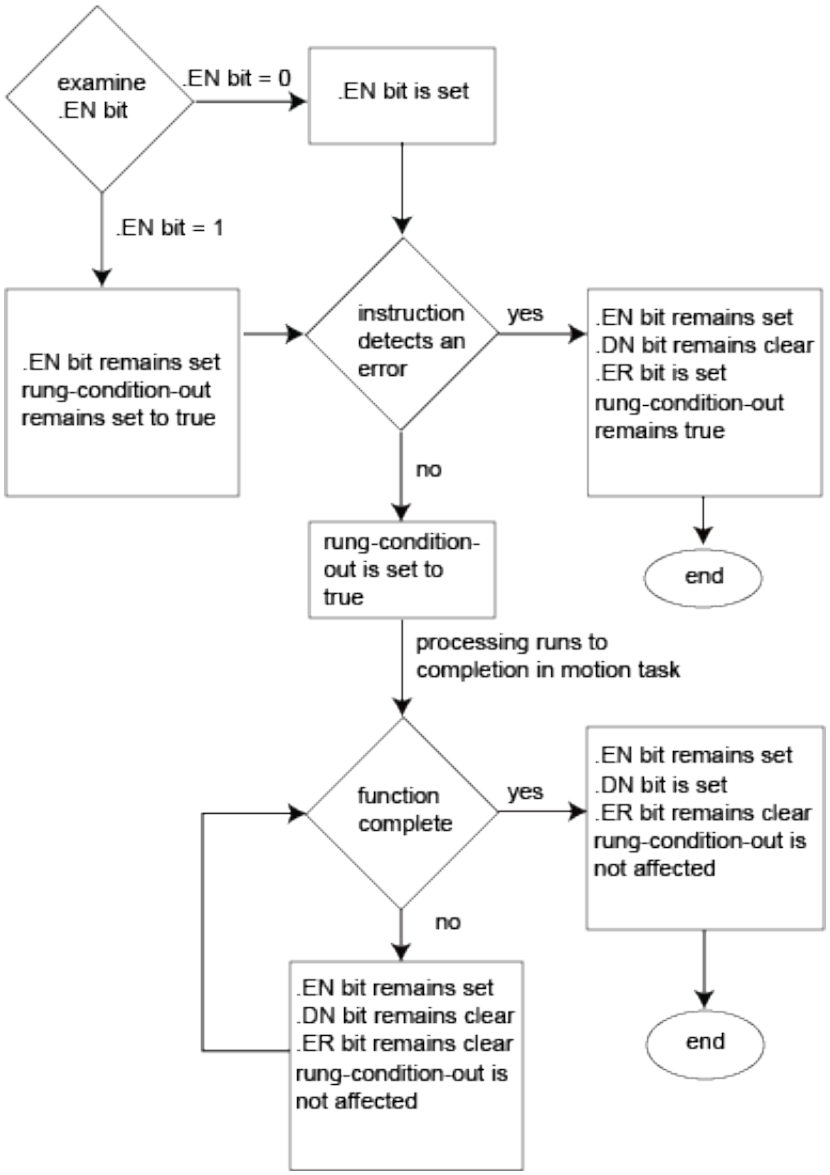
If you select	Then	Instruction error
a latch and unlatch delay combination that results in a compensated cam of less than minimum width	the width of the compensated cam is set to the minimum.	
a mode less than 0 or greater than 3	a Normal mode is considered	Illegal Output Compensation
a duty cycle less than 0 or greater than 100 and the mode is set to Pulsed or Inverted and Pulsed	a 0 or 100 duty cycle is considered	

a cycle time less than or equal to 0 and the mode is set to Pulsed or Inverted and Pulsed	the output bit is not pulsed	

See also

Motion Arm Output Cam (MAOC) on page 273

MAOC Flow Chart (True)



Motion Disarm Output Cam (MDOC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

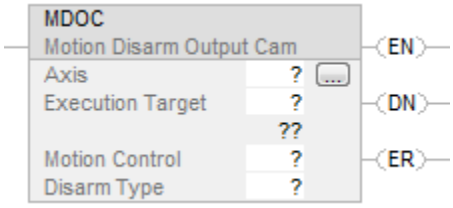
The Motion Disarm Output Cam (MDOC) instruction initiates the disarming of one or more Output Cams connected to the specified axis. Based on the disarm type, the MDOC instruction disarms either all Output Cams or only a

specific Output Cam. The corresponding outputs maintain the last state after the disarming.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MDOC(Axis, ExecutionTarget, MotionControl, DisarmType);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description

Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_CONSUMED **Tip:** AXIS_CONSUMED is supported by Compact GuardLogix 5580, CompactLogix 5380, and CompactLogix 5480 controllers only.	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS CONSUMED AXIS_SERVO AXIS_SERVO_DRIVE AXIS_GENERIC_DRIVE AXIS_GENERIC **Tip:** AXIS_GENERIC is supported by the ControlLogix 5570 and the GuardLogix 5570 controllers only.	Tag	Name of the axis that provides the position input to the Output Cam. Ellipsis launches Axis Properties dialog. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Execution Target	SINT, INT, or DINT	SINT, INT, or DINT	Immediate or Tag	The execution target defines the specific Output Cam from the set connected to the named axis. Behavior is determined by the following: 0...7 – Output Cams executed in the Logix controller. 8...31 – Reserved for future use.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Disarm Type	SINT, INT, or DINT	SINT, INT, or DINT	Immediate	Selects one or all Output Cams to be disarmed for a specified axis. Select either: 0 = All – Disarms all Output Cams connected to the specified axis. 1 = Specific – Disarms the Output Cam that is connected to the specified axis and defined by the Execution Target.

Structured Text

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

For the operands that require you to select from available options, enter your selection as:

This Operand	Has These Options Which You

	Enter as Text	Or Enter as a Number
Disarm Type	all specific	0 1

MOTION_INSTRUCTION Structure

Mnemonic	Description
.DN (Done) Bit 29	It is set to true when Output Cam(s) have been successfully disarmed.
.ER (Error) Bit 28	It is set to true to indicate that the instruction detected an error.

Description

The MDOC instruction disarms a specific or all output cams for a specified axis depending on the selected disarm type. The axis provides the position input to the Output Cam. The execution target defines a specific Output Cam from the set that is connected to the specified axis.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction.A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis Operand. See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is set to true. Otherwise, the .EN bit is not affected. The .DN,.ER ,.IP and .PC bits are not affected.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See *Motion Error Codes (.ERR)*.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. Extended Error codes for the Parameter Out of Range (13) error code lists a number that refers to the number of the operand as they are listed in the faceplate from top to bottom with the first operand being counted as zero. Therefore for the MDOC instruction, an extended error code of 4 would refer to the Disarm Type operand's value. You would then have to check your value with the accepted range of values for the instruction.

Execution

If ERR is	And EXERR is	Then Cause	Corrective Action
36	Varies	The size of the Output Cam array is not supported or the value of at least one member is out of range: Output bit less than 0 or greater than 31. Latch type less than 0 or greater than 3. Unlatch type less than 0 or greater than 5. Left or right position is out of cam range and the latch or unlatch type is set to "Position" or "Position and Enable". Duration less than or equal to 0 and the unlatch type is set to "Duration" or "Duration and Enable". Enable type less than 0 or greater than 3 and the latch or unlatch type is set to "Enable", "Position and Enable", or "Duration and Enable". Enable bit less than 0 or greater than 31 and the latch or unlatch type is set to "Enable", "Position and Enable", or "Duration and Enable". Latch type is set to "Inactive" and unlatch type is set to either "Duration" or "Duration and Enable".	Illegal Output Cam
37	Varies	Either the size of the Output Compensation array is not supported or the value of one of its members is out of range.	Illegal Output Compensation

The array index associated with errors 36 and 37 are stored in .SEGMENT of the Motion Instruction data type. Only the first of multiple errors are stored. The specific error detected is stored in Extended Error Code. With the ability to dynamically modify the Output Cam table, the Illegal Output Cam error 36 may occur while the MAOC is in-process. In general, the cam elements in which an error was detected will be skipped. The following are exceptions, and will continue to be processed. Error 2, Latch Type Invalid. Latch Type defaults to Inactive. Error 3, Unlatch Type Invalid. Unlatch Type defaults to Inactive. Error 8, with Unlatch Type of Duration and Enable. Will behave as an Enable Unlatch type.

Status Bits

MDOC Effects Status Bits

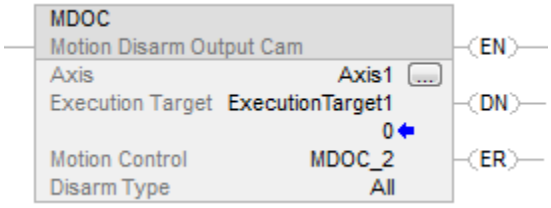
The MDOC instruction affects the following status words in the Motion Axis Structure:

- OutputCamStatus
- OutputCamPendingStatus
- OutputCamLockStatus
- OutputCamTransitionStatus

Each above is a DINT with bits 0 to 7 corresponding to the 8 execution targets. Bit 0 is execution target 0; bit 1 is execution target 1, etc.

Example

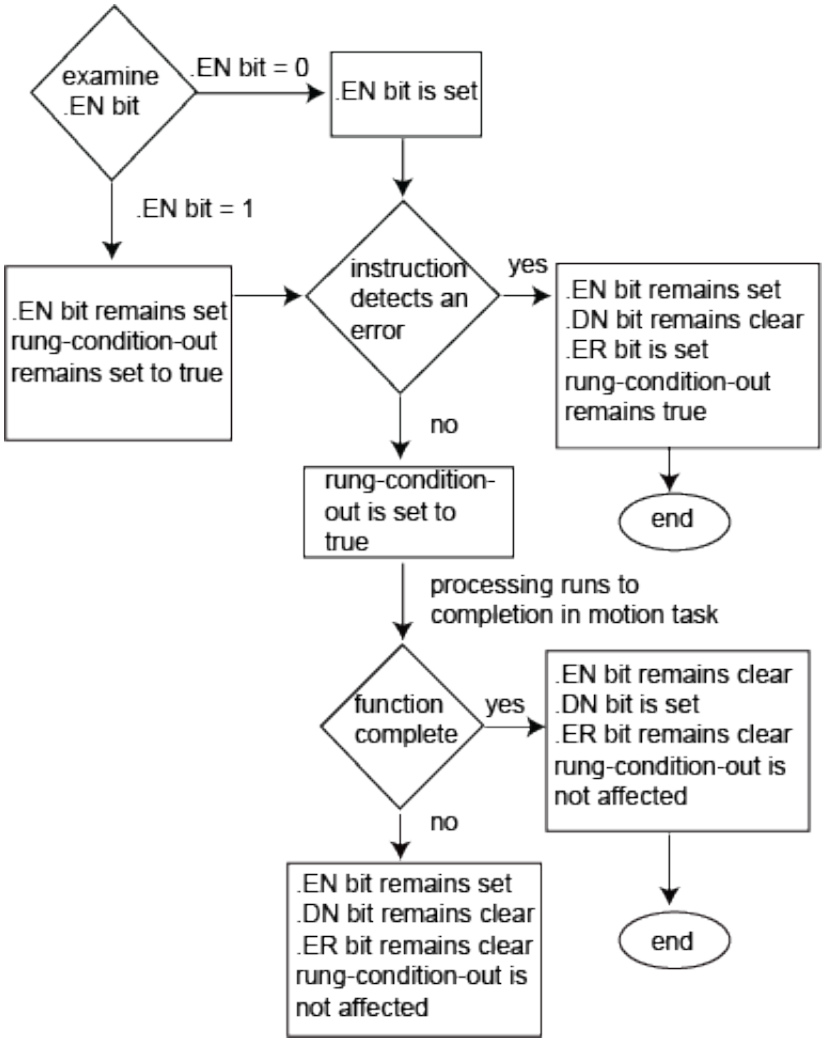
Ladder Diagram



Structured Text

MDOC(Axis1, ExecutionTarget1, MDOC_2, All);

MDOC Flow Chart (True)



Motion Configuration Instructions

Motion Configuration Instructions

Configuration instructions include all motion instructions that are used to establish and apply servo configuration parameters to an axis. This group of instructions includes hookup diagnostic instructions and tuning instructions.

Use the motion configuration instructions to tune an axis and run diagnostics tests for the servo system. The tests include:

- A motor encoder hookup test
- An encoder hookup test
- A marker test
- A commutation test (for PM motors only)
- Determine important motor parameters

Available Instructions

Ladder Diagram and Structured Text

MAAT	MRAT	MAHD	MRHD

Function Block

Not available

IMPORTANT Tags used for the motion control attribute of instructions should only be used once. Re-use of the motion control tag in other instructions can cause unintended operation. This may result in damage to equipment or personal injury.

Configuration instructions include all motion instructions that are used to establish and apply servo configuration parameters to an axis. This group of instructions includes hookup test diagnostic instructions and tuning instructions.

Use the motion configuration instructions to tune an axis and to run diagnostic tests for the servo system. These tests include:

- A motor encoder hookup test.
- An encoder hookup test.
- A marker test
- A commutation test (for PM motors only)

- Determine important motor parameters

The motion configuration instructions are:

If you want to:	Use this instruction:
Compute a complete set of servo gains and dynamic limits based on a previously executed MRAT instruction. The MAAT instruction also updates the servo module with the new gain parameters.	MAAT
Command the servo module to run a tuning motion profile for an axis.	MRAT
Apply the results of a previously executed MRHD instruction. The MAHD instruction generates a new set of encoder and servo polarities based on the observed direction of motion during the MRHD instruction.	MAHD
Command the servo module to run one of three diagnostic tests on an axis.	MRHD

See also

Motion Event Instructions on page 249

Motion Group Instructions on page 225

Motion Move Instructions on page 73

Motion State Instructions on page 23

Multi-Axis Coordinated Motion Instructions on page 359

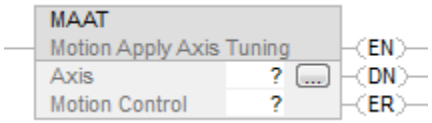
Motion Apply Axis Tuning (MAAT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, and ControlLogix 5580 controllers.

The Motion Apply Axis Tuning (MAAT) instruction is used to compute a complete set of servo gains and dynamic limits based on the results of a previously run Motion Run Axis Tuning (MRAT) instruction and update the motion module with these new gain parameters. While this instruction takes no explicit parameters, input is derived from the Axis Tuning Configuration parameters as described in Tune Status Parameter. After execution of the MAAT instruction, the corresponding axis should be ready for servo activation.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MAAT(Axis,MotionControl);

Operands

There are data conversion rules for mixed data types within an instruction. See *Data Conversions*.

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Axis	AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis on which to perform operation. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

See Structured Text Syntax for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the EnableIn goes false.
.DN (Done) Bit 29	The done bit indicates when the instruction completes an apply axis-tuning process.

.ER (Error) Bit 28	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

Description

The MAAT instruction is used to execute a series of computations resulting in values for gain and dynamic configuration parameters on the specified axis. As part of the work performed by MAAT, these resultant configuration parameters are applied so that the axis is ready for full servo operation. This instruction is designed to follow execution of the Motion Run Axis Tuning (MRAT) which generates axis input configuration values for the MAAT instruction. See the Motion Run Axis Tuning (MRAT) description for more information. MAAT requires no explicit input parameters; simply enter or select the desired physical axis.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

The MAAT instruction uses axis configuration parameters as input and output. The input configuration parameters that MAAT uses are shown in the table below. Refer to the Motion Axis Object specification for a detailed description of these parameters.

The axis configuration parameters that MAAT uses as input depends on the External Drive configuration. If the External Vel Servo Drive configuration bit parameter is TRUE, indicating interface to an external velocity servo drive, these input parameters are required.

Axis Parameter	Data Type	Units	Meaning
Tuning Velocity	Real	pos units/sec	Top Speed of Tuning Profile.
Tune Accel	Real	pos units/sec^2	Calculated Acceleration Time of Tuning Profile.
Tune Decel	Real	pos units/sec^2	Calculated Deceleration Time of Tuning Profile.
Tune Velocity Scaling	Real	mV/KCPS	Measured Velocity Scaling factor of axis Drive/Motor/Encoder system.
Tune Velocity Bandwidth	Real	Hertz	Bandwidth of External Velocity Servo Drive.

If the External Vel Servo Drive configuration bit parameter is FALSE, indicating interface to an external torque servo drive, these input parameters are required.

Axis Parameter	Data Type	Units	Meaning
Damping Factor	Real	-	Damping Factor used to calculate the gains.
Tuning Velocity	Real	pos units/sec	Top Speed of Tuning Profile.
Tune Accel	Real	pos units/sec^2	Calculated Acceleration Time of Tuning Profile.

Tune Decel	Real	pos units/sec^2	Calculated Deceleration Time of Tuning Profile.
Effective Inertia	Real	mV/KCPS^2	Computed Effective Inertia of Drive/Motor system.
Position Servo Bandwidth	Real	Hertz	Maximum Position Servo Loop Bandwidth.

The axis configuration parameters that MAAT generates as output depend on the External Drive configuration. If the External Vel Servo Drive configuration bit parameter is TRUE, indicating interface to an external velocity servo drive, these output parameters are generated.

Axis Parameter	Data Type	Units	Meaning
Pos Proportional Gain	Real	1/msec	Position Servo Loop Proportional Gain.
Pos Integral Gain	Real	1/msec^2	Position Servo Loop Integral Gain -- Set to Zero.
Velocity Feedforward	Real	-	Position Servo Loop Proportional Gain.
Acceleration Feedforward	Real	-	Velocity Command Feedforward -- Set to Zero.
Max Speed	Real	pos units/sec	Maximum Speed for Motion Profiles – Set to Tuning Velocity
Max Acceleration	Real	pos units/sec^2	Maximum Acceleration for Motion Profiles
Max Deceleration	Real	pos units/sec^2	Maximum Deceleration for Motion Profiles
Output Filter Bandwidth	Real	Hertz	Bandwidth of Low Pass Servo Output Filter
Output Scaling	Real	mV/ KCPS	Scale Factor applied to output of the Position Servo Loop to the DAC.
Position Error Tolerance	Real	pos units	Maximum Servo Loop Position Error allowed without Fault.

If the External Vel Servo Drive configuration bit parameter is FALSE, indicating interface to an external torque servo drive, these output parameters are generated.

Axis Parameter	Data Type	Units	Meaning
Pos Proportional Gain	Real	1/msec	Position Servo Loop Proportional Gain.
Pos Integral Gain	Real	1/msec^2	Position Servo Loop Integral Gain.
Vel Proportional Gain	Real	1/msec	Velocity Servo Loop Proportional Gain.
Vel Integral Gain	Real	1/msec^2	Velocity Servo Loop Integral Gain.
Velocity Feedforward	Real	-	Position Servo Loop Proportional Gain.
Acceleration Feedforward	Real	-	Velocity Command Feedforward.
Max Speed	Real	pos units/sec	Maximum Speed for Motion Profiles – Set to Tuning Velocity
Max Acceleration	Real	pos units/sec^2	Maximum Acceleration for Motion Profiles.

Maximum Deceleration	Real	pos units/sec^2	Maximum Deceleration for Motion Profiles.
Output Filter Bandwidth	Real	Hertz	Bandwidth of Low Pass Servo Output Filter.
Output Scaling	Real	mV/KCPS^2	Scale Factor applied to output of the Velocity Servo Loop to the DAC.
Position Error Tolerance	Real	pos units	Maximum Servo Loop Position Error allowed without Fault.

The output parameters generated by the MAAT instruction are immediately applied to the specified axis so that subsequent motion can be performed.

For more information about tuning configuration parameters see Tune Status Parameter.

To successfully execute a MAAT instruction, the targeted axis must be configured as a Servo axis and be in the Axis Ready state, with servo action off. If these conditions are not met, the instruction errors.

IMPORTANT The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is only set after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle EnableIn from false to true each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis operand. See *Common Attributes* for operand-related faults.

Condition/StateExecution Ladder Diagram	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is set to true. Otherwise, the .EN bit is not affected. The .DN,.ER ,.IP and .PC bits are not affected.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See *Motion Error Codes (ERR)* for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. See Motion Error Codes (ERR) for Motion Instructions. The following Extended Error codes help to pinpoint the problem when the MAAT instruction receives a Servo Message Failure (12) error message.

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	No resource (2)	Not enough memory resources to complete request. (SERCOS)
SERVO_MESSAGE_FAILURE (12)	Object Mode conflict (12)	Axis is in shutdown.
SERVO_MESSAGE_FAILURE (12)	Permission denied (15)	Enable input switch error. (SERCOS)
SERVO_MESSAGE_FAILURE (12)	Device in wrong state (16)	Redefine Position, Home, and Registration 2 are mutually exclusive (SERCOS), device state not correct for action. (SERCOS)

Status Bits

MAAT Changes to Status Bits

The MAAT instruction does not make any changes to the status bits.

Example 1

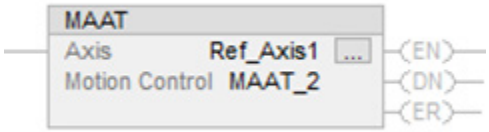
When the input conditions are true, the controller computes a complete set of servo gains and dynamic limits for axis1 based on the results of the previously executed Motion Run Axis Tuning (MRAT) instruction.

Ladder Diagram

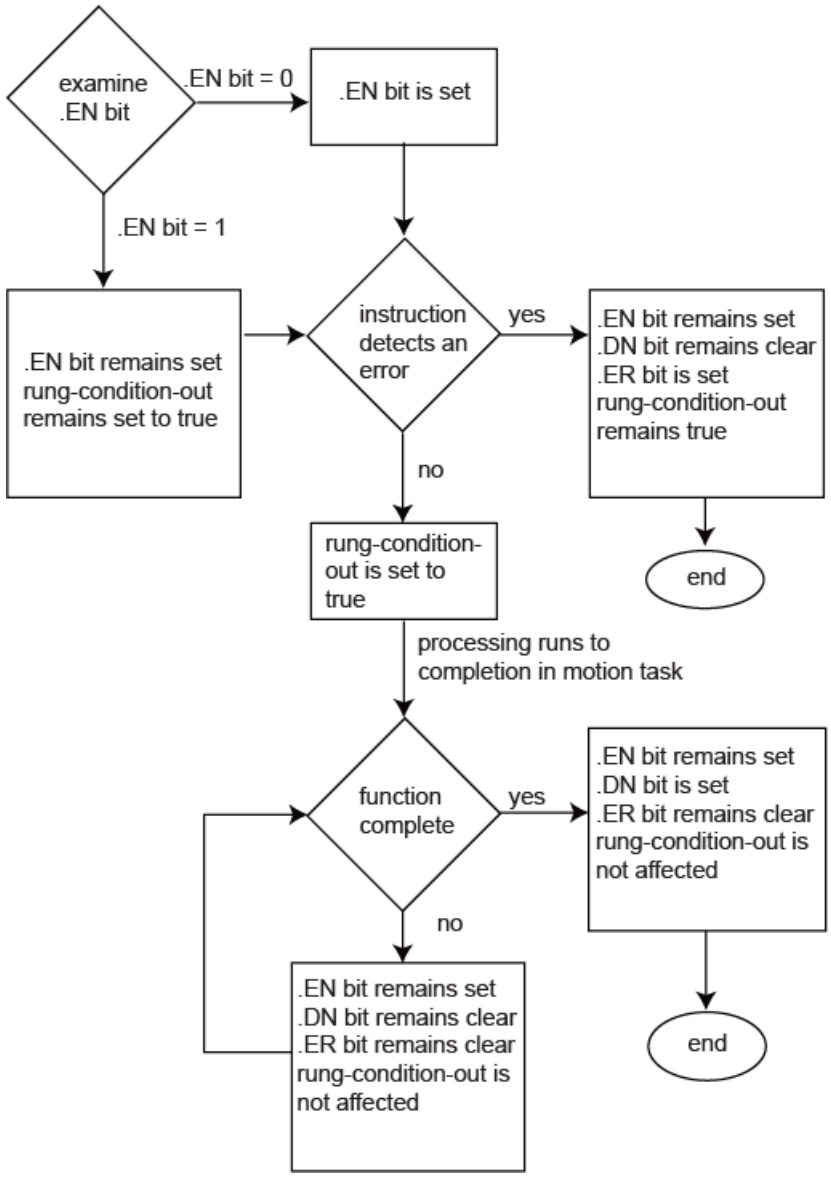
Example 2

When the input conditions are true, the controller computes a complete set of servo gains and dynamic limits for a reference to Axis1 based on the results of the previously executed Motion Run Axis Tuning (MRAT) instruction.

Ladder Diagram



MAAT Flow Chart (True)



Motion Run Axis Tuning (MRAT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

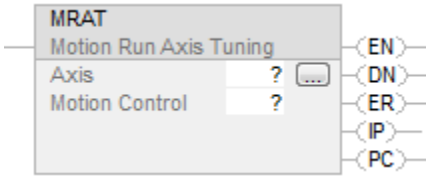
Use the Motion Run Axis Tuning (MRAT) to command the motion module to run a tuning motion profile for the specified axis. The tuning motion profile consists of one or more acceleration and deceleration ramps induced by applying fixed voltages to the servo's drive output. Note that this instruction does not at any time close the servo loop. While this instruction takes no explicit input parameters, it does derive input from the Axis Tuning Configuration parameters. The result of executing the MRAT instruction is a set of measurement data that is stored in the Axis Object for subsequent use with the Motion Apply Axis Tuning (MAAT) instruction.

This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MRAT(Axis,MotionControl);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description
Axis	AXIS_CIP_DRIVE	AXIS_CIP_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis on which to perform the operation. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.

Motion Control	MOTION_INSTRUCTION		Tag	Structure used to access instruction status parameters.

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Mnemonic	Description
.DN (Done) Bit 29	It is set after the tuning process has been successfully completed.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.IP (In Process) Bit 26	It is set on positive rung transition and cleared after the tuning process is complete, or terminated by a stop command, shutdown, or a servo fault
.PC (Process Complete) Bit 27	It is set after the tuning process has been successfully completed

Description - AXIS_SERVO, AXIS_SERVO_DRIVE

The MRAT instruction is used to execute a tuning motion profile on the specified axis. During this brief tuning motion profile, the motion module makes timing and velocity measurements that serve as input data for a subsequent Motion Apply Axis Tuning (MAAT) instruction. MRAT requires no explicit input parameters; simply enter or select the desired physical axis.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

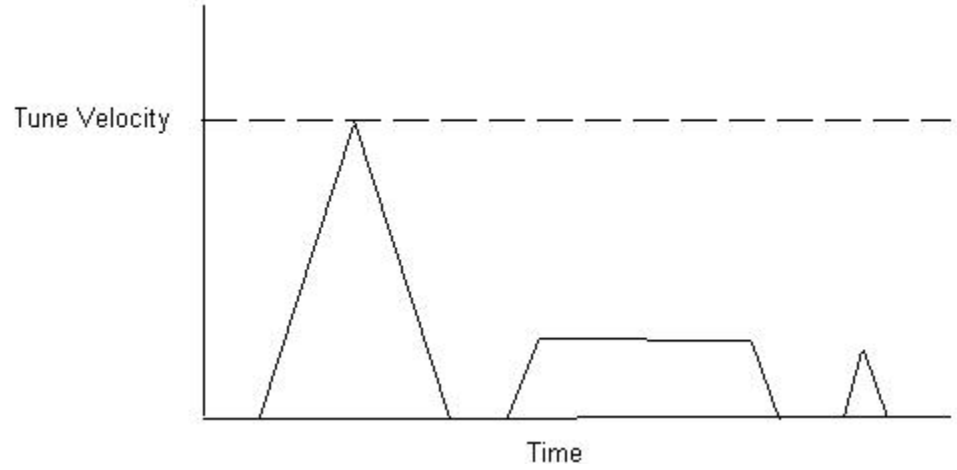
The MRAT instruction uses axis configuration parameters as input and output. The input configuration parameters that MRAT uses are shown in this table.

Axis Parameter	Data Type	Units	Meaning
Tuning Direction	Boolean	-	Direction of Tuning Motion (0-Fwd, 1-Rev).
Tuning Travel Limit	Real	pos units	Maximum allowed excursion of Axis.
Tuning Velocity	Real	pos units/sec	Top Speed of Tuning Profile.
Dumping Factor	Real	-	Damping Factor used to calculate the maximum Position Servo Bandwidth.

Based on the above configuration parameters, MRAT execution generates a motion event on the specified axis that consists of a single triangular velocity profile or a series of three such profiles. Tune Velocity must be within the maximum speed capability of the drive and motor. The configured value for Tune Velocity should be set to the desired maximum operating speed of the axis so that the resulting tuning parameters are based on the dynamics of the system at that speed.

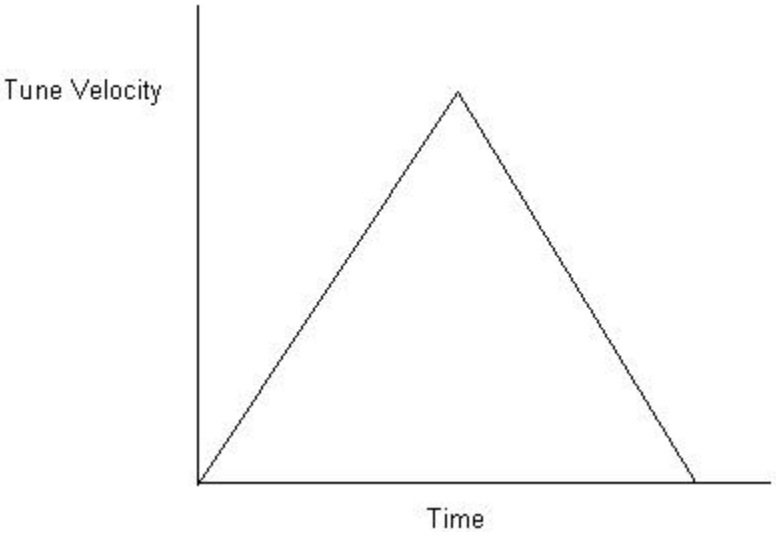
If the External Vel Servo Drive configuration bit parameter is TRUE, indicating interface to an external velocity servo drive, three pulses are applied to the axis. The tuning velocity profile for this case is shown in this diagram.

Tuning Velocity Profile when True



If the External Vel Servo Drive configuration bit parameter is FALSE, indicating interface to an external torque servo drive, only one pulse is applied to the axis. The tuning velocity profile is shown below.

Tuning Velocity Profile when False



The axis configuration parameters that MRAT generates as output depend on the External Drive configuration. If the External Vel Servo Drive configuration bit parameter is TRUE, indicating interface to an external velocity servo drive, these output parameters are generated.

Axis Parameter	Data Type	Units	Meaning
Tune Status	Real	-	Status Report of the Tuning Process.
Tune Accel Time	Real	seconds	Measured Acceleration Time of Tuning Profile.
Tune Decel Time	Real	seconds	Measured Deceleration Time of Tuning Profile.
Tune Accel	Real	pos units/sec2	Calculated Acceleration Time of Tuning Profile.
Tune Decel	Real	pos units/sec2	Calculated Deceleration Time of Tuning Profile.

Tune Velocity Scaling	Real	mV/KCPS	Measured Velocity Scaling factor of axis Drive/Motor/Encoder system.
Tune Rise Time	Real	mV/KCPS	Measured Rise Time of Tuning Step Response Profile.
Tune Velocity Bandwidth	Real	Hertz	Computed Bandwidth of External Velocity Servo Drive

If the External Vel Servo Drive configuration bit parameter is FALSE, indicating interface to an external torque servo drive, these output parameters are generated.

Axis Parameter	Data Type	Units	Meaning
Tune Status	Real	-	Status Report of the Tuning Process.
Tune Accel Time	Real	seconds	Measured Acceleration Time of Tuning Profile.
Tune Decel Time	Real	seconds	Measured Deceleration Time of Tuning Profile.
Tune Accel	Real	pos units/sec2	Calculated Acceleration Time of Tuning Profile.
Tune Decel	Real	pos units/sec2	Calculated Deceleration Time of Tuning Profile.
Effective Inertia	Real	mV/KCPS	Computed Effective Inertia of Drive/Motor system.
Position Servo Bandwidth	Real	Hertz	Calculated Maximum Position Servo Loop Bandwidth.

The above output parameters generated by the MRAT instruction serve as inputs to a subsequent MAAT instruction which performs further tuning calculations and applies the results to various axis' servo and dynamic configuration parameters.

Description - AXIS_CIP_DRIVE

The MRAT instruction is used to execute a tuning motion profile on the specified CIP axis. MRAT requires no explicit input parameters; simply enter or select the desired physical axis.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

The MRAT instruction uses the CIP Axis configuration parameters as input and output. The input configuration parameters that MRAT uses are shown in this table.

Axis Parameter	Data Type	Units	Meaning
Tuning Direction	Short Integer	-	It determines the direction of the motion profile initiated by the Inertia Test service associated with the MRAT instruction. 0 = Unidirectional Forward 1 = Unidirectional Reverse 2 = Bi-Directional Forward 3 = Bi-Directional Reverse.
Tuning Travel Limit	Real	Position Units	It is used by the Inertia Test service, associated with the MRAT instruction, to limit the excursion of the axis during the test.

Tune Speed	Real	Position Units/sec	The Tuning Speed attribute value determines the maximum speed used by the Inertia Test service initiated motion profile. This attribute should be set to the desired maximum operating speed of the motor prior to running the test.
Tuning Torque	Real	% Rated	It determines the maximum torque used by the Inertia Test service initiated motion profile. This attribute should be set to the desired maximum safe torque level prior to running the test. The default value is 100%, which yields the most accurate measure of the acceleration and deceleration capabilities of the system.
Damping Factor	Real	-	It is used in calculating the maximum Position and Velocity Servo Bandwidth values during execution of the MRAT instruction.

The input configuration parameters can also be set using the **Axis Properties - Autotune** dialog box.

The Loop Response selection is used by the software to determine the value for the Damping Factor.

Loop Response	Damping Factor
Low	1.5
Medium	1.0
High	0.8

Based on the above configuration parameters, MRAT execution generates a motion event on the specified axis that consists of a triangular velocity profile. The tuning procedure will measure maximum acceleration and deceleration rates based on ramps to and from the Tuning Speed. Thus, the accuracy of the measured acceleration and deceleration capability is reduced by tuning at a speed other than the desired operating speed of the system.

The axis configuration parameters that MRAT generates as output for CIP axis are shown in this table:

Axis Parameter	Data Type	Units	Meaning
Tuning Status	Integer	-	The Tune Status attribute returns status of the last run Inertia Test service that initiates a process on the targeted drive axis.
Tune Accel Time	Real	Seconds	Measured Acceleration time in seconds of the Tuning profile.
Tune Decel Time	Real	Seconds	Measured Acceleration time in seconds of the Tuning profile.
Tune Accel	Real	Position Units/sec2	Measured Acceleration of the Tuning profile.
Tune Decel	Real	Position Units/sec2	Measured Deceleration of the Tuning profile.
Tune Inertia Mass	Real	% Motor Rated / (Motor Units/Sec2)	The estimated inertia or mass for the axis as calculated from the measurements made during the tuning process.
Tune Friction	Real	% Rated	The amount of friction measured during Tuning profile. This value can be used to configure the Friction Compensation feature of the drive.

Tune Load Offset	Real	% Rated	This value represents the active load offset measured during the Tune profile. This value can be used to set the Torque Offset of the drive to cancel out the active load torque/force.
Position Servo Bandwidth	Real	Hertz	It represents the unity gain bandwidth of the position loop that is used to calculate the position loop gains.
Velocity Servo Bandwidth	Real	Hertz	It represents the unity gain bandwidth of the velocity loop that is used to calculate the velocity loop gains.

The above output parameters generated by the MRAT instruction serve as inputs to compute the Position and Velocity loop gains, Position and Velocity Error Tolerances, Feed Forward Gains, Load Ratio, Maximum Acceleration, Maximum Deceleration, System Inertia, System Acceleration and Friction Compensation.

If the Gain Tuning Config Bits parameter bit zero is the Run Inertia Test Bit. This bit determines whether or not the MRAT tuning instruction will send a Test Inertia service to the drive to perform an inertia measurement. If this bit is set, the Inertia Test shall be performed. If the bit is clear, the MRAT will immediately complete without an inertia measurement. It will only calculate the Pos and Velocity Servo Loop Bandwidths based on the Loop response or the Damping factor.

Tune Status Parameter

Conditions may occur that make it impossible for the controller to properly perform the tuning operation. When this is the case, the tuning process is automatically aborted and a tuning fault reported that is stored in the Tune Status output parameter (GSVable). It is also possible to manually abort a tuning process using a Motion Axis Stop (MAS) instruction which results in a tuning fault reported by the Tune Status parameter. This table shows possible values for Tuning Status.

Status Code	Code	Meaning
Tune Success	0	Tune process has been successful.
Tune In Process	1	Tuning is in progress.
Tune Aborted	2	Tuning Process was aborted.
Tune Time-out	3	Tuning Process has timed out.
Tune Servo Fault	4	Tuning Process Failed due to Servo Fault.
Tune Travel Fault	5	Axis reached Tuning Travel Limit.
Tune Polarity Fault	6	Axis motion heading in wrong direction due to incorrect motor/encoder polarity configuration.
Tune Speed Fault	7	Axis tuning speed too low to achieve minimum measurement accuracy.
Tune Configuration Fault	8	The specified axis tuning configuration is not allowed and a fault occurs.
Tune Not Allowed In Axis Test Mode	128	The axis that is in Axis Test Mode.

IMPORTANT The Tune Status Parameter is not to be mistaken for the .STATUS sub-tag of the MRAT instruction.

To successfully execute a MRAT instruction on an axis, the targeted axis must be configured as a Servo Axis Type, the axis must be in the Axis Ready state and Axis Test Mode must be disabled. If any of these conditions are not met then the instruction returns an error.

IMPORTANT When the MRAT instruction is initially executed the In Process (.IP) bit is set and the Process Complete (.PC) bit is cleared. The MRAT instruction execution can take multiple scans to execute because it requires transmission of multiple messages to the motion module. The Done (.DN) bit, is not set immediately, but only after these messages are successfully transmitted. The In Process (.IP) bit is cleared and the Process Complete (.PC) bit is set at the same time that the Done (.DN) bit is set.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction.A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis operand. See Common Attributes for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See Error Codes (ERR) for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. The following Extended Error Codes help to pinpoint the problem when the MRAT instruction receives an error message.

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	Process terminated on request (1)	Tune execution followed by an instruction to shutdown/disable drive, or a motion stop instruction or a Processor change requests a cancel of Tune.
SERVO_MESSAGE_FAILURE (12)	Object Mode conflict (12)	Axis is in shutdown.
SERVO_MESSAGE_FAILURE (12)	Device in wrong state (16)	Incorrect Tune Process order. (SERCOS)
TUNE_PROCESS_ERROR (14)	Tune Test prohibited (1)	Axis Test Mode enabled- Tune Test is not supported in this configuration.

Status Bits

MRAT Changes to Status Bits

Bit Name	State	Meaning
DriveEnableStatus	TRUE	Axis is in Drive Control state with the Drive Enable output active while the Tuning Profile is running.
TuneStatus	TRUE	The axis is running a tuning process.

Example 1

When the input conditions are true, the controller commands the servo module to run a tuning motion profile for axis1.

Ladder Diagram

Neutral text for ladder diagram normal scan

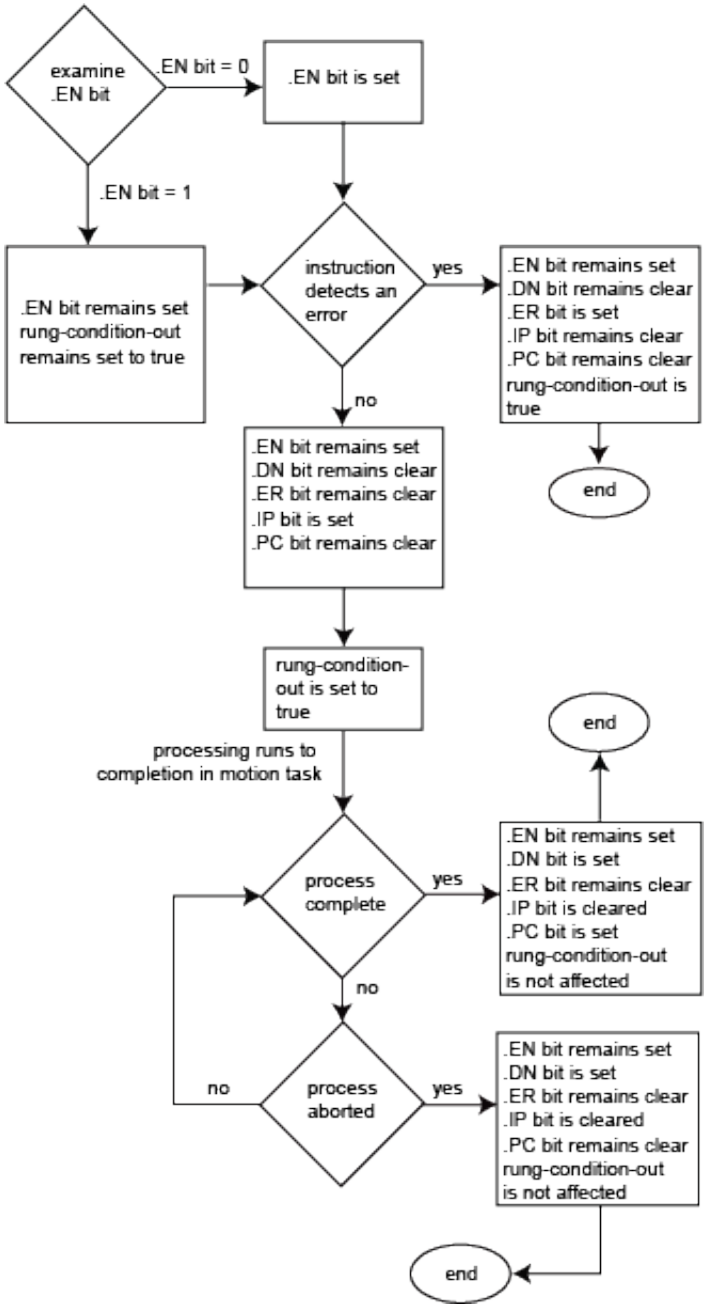
MRAT (Axis1, MRAT_1);

Example 2

When the input conditions are true, the controller commands the servo module to run a tuning motion profile for Axis1 through a reference to it.

Ladder Diagram

MRAT Flow Chart (True)



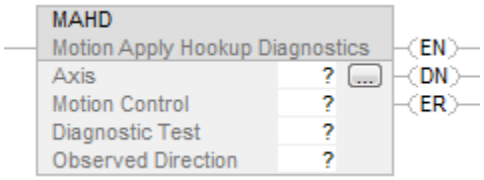
Motion Apply Hookup Diagnostics (MAHD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, and ControlLogix 5580 controllers.

The Motion Apply Hookup Diagnostics (MAHD) instruction is used to apply the results of a previously run the Motion Run Hookup Diagnostics (MRHD) instruction to generate a new set of encoder and servo polarities based on the Observed Direction of motion during the test. As part of the application process the instruction updates the motion module with these new polarity settings. After execution of the MAHD instruction, and assuming that a stable set of gains was established, the corresponding axis should be ready for servo activation.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MAHD(Axis,MotionControl,DiagnosticTest,ObservedDirection);

Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion.

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Axis	AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis on which to perform the operation. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Diagnostic Test	UDINT	Immediate	Selects the specific test for the motion module to run: 0 = motor/encoder hookup test 1 = encoder hookup test 2 = encoder marker test
Observed Direction	BOOLEAN	Immediate	Sets the direction of the test motion. Select either: 0 = forward 1 = reverse

See Structured Text Syntax for more information on the syntax of expressions within structured text.

For the operands that require you to select from available options, enter your selection as:

This Operand	Has These Options Which You	
	Enter as Text	Or Enter as a Number
DiagnosticTest	motor_encoder encoder marker	0 1 2
ObservedDirection	forward reverse	0 1

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set after the hookup test apply process has been successfully executed.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.

Description

The MAHD instruction is used to execute a series of computations resulting in values for the Encoder Polarity and Servo Polarity configuration bit parameters of the specified axis. As part of work performed by MAHD, these resultant configuration bit parameters are applied to the motion module so that the axis is ready for full servo operation. This instruction is designed to follow execution of the Motion Run Hookup Diagnostics (MRHD) instruction which generates axis input configuration values for the MAHD instruction. See the MRHD instruction description for more information. MAHD requires specification of the Diagnostic Test to apply and the Observed Direction of motion during the previous Motion Run Hookup Diagnostics (MRHD) instruction test process. Enter or select the Diagnostic Test and the Observed Direction and the desired physical axis.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

The MAHD instruction uses axis configuration parameters as input and output. The input configuration parameters that MAHD uses are shown in this table. The Test Direction Forward bit is automatically established as output from the Motion Run Hookup Diagnostics (MRHD) instruction.

Axis Parameter	Data Type	Units	Meaning

Test Direction Forward	Boolean	-	Direction of axis travel during hookup test as seen by the motion module.

Motor Encoder Hookup Test

If the Motor Encoder Test is selected, the controller computes the proper setting for both the Encoder Polarity and the Drive Polarity based on the Observed Direction instruction parameter and the state of Test Direction Forward bit which was established by the output of the Motion Run Hookup Diagnostics (MRHD) instruction. Once the Encoder Polarity and Drive Polarity settings are computed the MAHD applies these values to the corresponding axis configuration parameter bits as shown in this table:

Axis Parameter	Data Type	Units	Meaning
Encoder Polarity Negative	Boolean	-	Inverts the sense of the encoder feedback input to the motion module.
Drive Polarity Negative	Boolean	-	Inverts the sense of the DAC analog output from the motion module.

Encoder Hookup Test

If the Encoder Test is selected, the controller computes the proper setting for just the Encoder Polarity based on the Observed Direction instruction parameter and the state of Test Direction Forward bit which was established by the output of the Motion Run Hookup Diagnostics (MRHD) instruction. Once the Encoder Polarity and Drive Polarity settings are computed, the MAHD applies these values to the corresponding axis configuration parameter bits as shown in this table.

Axis Parameter	Data Type	Units	Meaning
Encoder Polarity Negative	Boolean	-	Inverts the sense of the encoder feedback input to the motion module.

To successfully execute a MAHD instruction running the Motor Encoder Test, the targeted axis must be configured as either a Servo or Feedback Only axis type. If any of these conditions are not met than the instruction errs.

IMPORTANT The instruction execution may take multiple scans to execute because it requires multiple coarse updates to complete the request. The Done (.DN) bit is not set immediately, but only after the request is completed.

This is a transitional instruction:

- In relay ladder, toggle EnableIn from false to true each time the instruction should execute.

- In structured text, condition the instruction so that it only executes on a transition.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis operand. See Common attributes for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See Motion Error Codes (ERR) for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. See Motion Error Codes (ERR) for Motion Instructions. The following Extended Error codes help to pinpoint the problem when the MAHD instruction receives a Servo Message Failure (12) error message.

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	No resources (2)	Not enough memory resources to complete request. (SERCOS)
SERVO_MESSAGE_FAILURE (12)	Object Mode conflict (12)	Axis is in shutdown.
SERVO_MESSAGE_FAILURE (12)	Permission denied (15)	Enable input switch error. (SERCOS)
SERVO_MESSAGE_FAILURE (12)	Device in wrong state (16)	Redefine Position, Home, and Registration 2 are mutually exclusive (SERCOS), device state not correct for action. (SERCOS)

Status Bits

MAHD Changes to Status Bits

None

Example 1

When the input conditions are true, the controller applies the results of a previously executed Motion Run Hookup Diagnostics (MRHD) instruction to axis1.

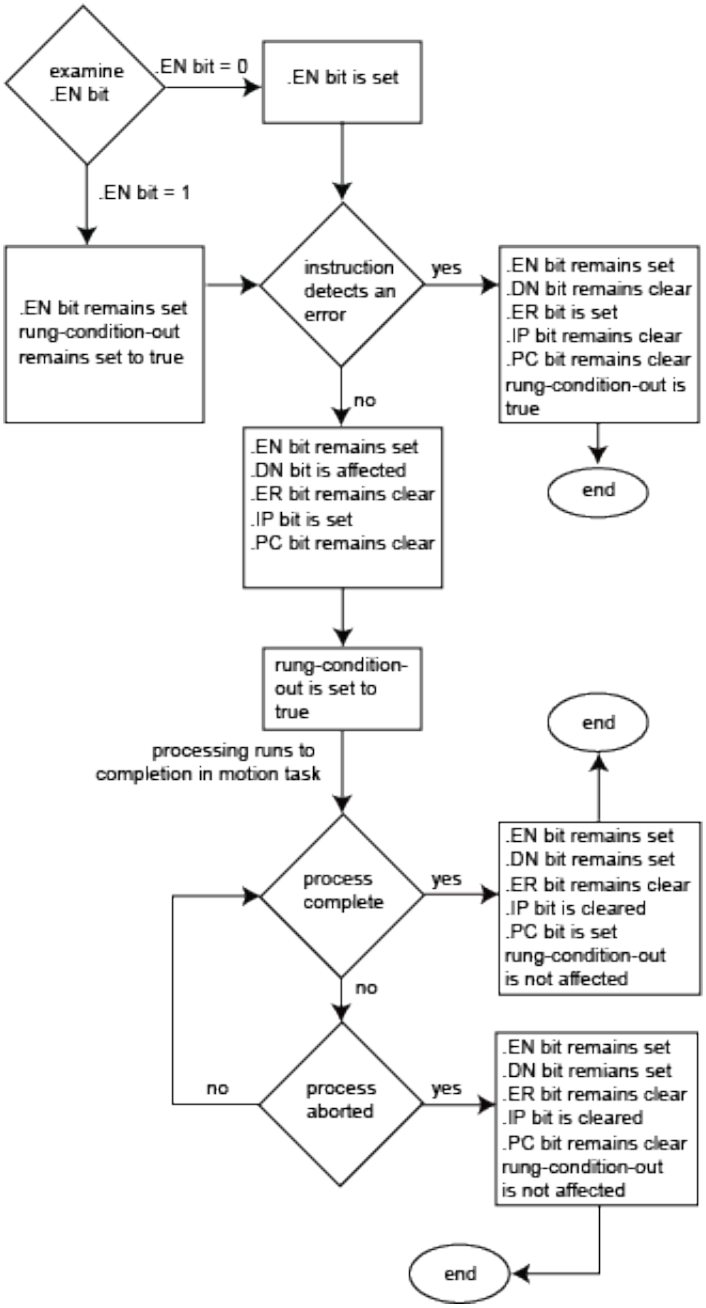
Ladder Diagram

Example 2

When the input conditions are true, the controller applies the results of a previously executed Motion Run Hookup Diagnostics (MRHD) instruction to Axis1 through a reference to it.

Ladder Diagram

MAHD Flow Chart (True)



Motion Run Hookup Diagnostics (MRHD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Use the Motion Run Hookup Diagnostics (MRHD) instruction to command the motion module to run any one of three different diagnostics on the specified axis as selected by the Diagnostic Test. Currently diagnostics are available to test the motor/encoder hookup for a servo axis, the encoder hookup, and the encoder marker hookup. Commutation Test is also available but only on an AXIS_CIP_DRIVE axis. Only the motor/encoder diagnostic initiates motion on the axis. This action consists of a short move of a user Motor Encoder Test Increment. The move is initiated by roughly 1 Volt per second ramping level of the servo's drive output. The result of executing the

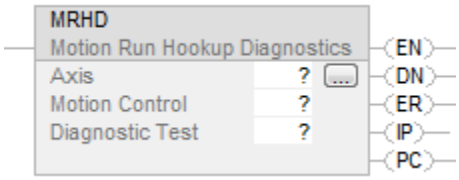
MRHD instruction is that the parameters, Test Status and Test Direction Forward are updated.

This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MRHD(Axis,MotionControl,DiagnosticTest);

Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion.

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description
Axis	AXIS_CIP_DRIVE	AXIS_CIP_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis on which to perform the operation. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Diagnostic test	DINT	DINT	Immediate	Selects the specific test for the motion module to run: 0 = motor/encoder hookup test 1 = encoder hookup test 2 = encoder marker test 3=commutation test

See Structured Text Syntax for more information on the syntax of expressions within structured text.

For the operands that require you to select from available options, enter your selection as:

This Operand	Has These Options Which You	
	Enter as Text	Or Enter as a Number
DiagnosticTest	motor_encoder encoder marker commutation	0 1 2 3

MOTION_INSTRUCTION Structure

Mnemonic	Description
.EN (Enable) Bit 31	It is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done) Bit 29	It is set after the hookup test apply process has been successfully executed.
.ER (Error) Bit 28	It is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.IP (In Process) Bit 26	It is set on positive rung transition and cleared after the diagnostic test process is complete, or terminated by a stop command, shutdown, or a servo fault.
.PC (Process Complete)Bit 27	It is set after the diagnostic test process has been successfully completed.

Description - AXIS_SERVO, AXIS_SERVO_DRIVE

The MRHD instruction is used to execute various test diagnostics on the specified axis to test the integrity and, in some cases, the polarity of servo field connections. There are currently test diagnostics supporting drive hookup, encoder hookup, marker hookup and motion module OK contact hookup. Test for Commutation also available when using Axis_CIP_Drive. During some of these test processes the motion module generates output to the external drive to produce a small amount of motion. Measurements made during some of these hookup diagnostic tests are saved as output configuration parameters that also serve as input data for a subsequent MAHD (Motion Apply Hookup Diagnostic) instruction. MRHD requires only one explicit input parameter, Diagnostic Test. Enter or select the Diagnostic Test to run and the axis to test.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

The MRHD instruction uses axis configuration parameters as input and output. The input configuration parameters that MRHD uses are shown in this table.

Axis Parameter	Data Type	Units	Meaning
Motor Encoder Test Increment	Real	-	Distance that the Axis must travel to satisfy the Hookup Diagnostic Test.

The axis configuration parameters that MRHD generates as output depend on the specified Hookup Diagnostic.

Motor Encoder Hookup Test

If the Motor Encoder Test is selected, the motion module enables the external drive and generates a 1 Volt per second output ramp to the drive while monitoring the encoder feedback. When the axis has moved a distance greater than or equal to the configured Motor Encoder Test Increment, the test voltage is set back to zero and the drive disabled. The motion module then reports the direction of travel which is stored as one of these output parameters:

Axis Parameter	Data Type	Units	Meaning
Test Status	Integer	-	Status Report of the Hookup Diagnostic Test Process.
Test Direction Forward	Boolean	-	Direction of axis travel during hookup test as seen by the motion module.

If due to improper hookup, or some other problem with the system, the axis feedback fails to detect that axis reaching the configured Motor Encoder Test Increment within 2 seconds, the servo sets the test voltage back to zero and disables the drive. The control reflects this condition through the Test Status axis output parameter. This usually indicates that either the cabling to the drive or the cabling to the encoder is incorrect. Running MRHD with the

Encoder Hookup Test selected is an effective method of isolating the problem to the encoder or drive.

Encoder Hookup Test

If the Encoder Test is selected, the motion module does not generate any axis motion, but simply monitors axis encoder feedback. The axis can then be moved by hand or by some other independent drive actuator to generate motion. When the motion module detects that the axis has moved a distance greater than or equal to the configured Motor Encoder Test Increment, the test is complete. The motion module then reports the direction of travel as one of the following MRHD output parameters.

Axis Parameter	Data Type	Units	Meaning
Test Status	Integer	-	Status Report of the Hookup Diagnostic Test Process.
Test Direction Forward	Boolean	-	Direction of axis travel during hookup test as seen by the motion module.

If due to improper hookup, or some other problem with the system, the axis feedback fails to detect the axis reaching the configured Motor Encoder Test Increment after moving the axis at least that distance, then abort the test using the MAS instruction and check the encoder wiring.

Marker Hookup Test

If the Marker Test is selected, the motion module does not generate any axis motion, but simply monitors axis encoder feedback. The axis can then be moved by hand or by some other independent drive actuator to generate motion. When the motion module detects a marker (Channel Z) pulse, the test is then complete. The motion module then reports success via the Test Status.

Axis Parameter	Data Type	Units	Meaning
Test Status	Integer	-	Status Report of the Hookup Diagnostic Test Process.
Test Direction Forward	Boolean	-	Direction of axis travel during hookup test as seen by the motion module.

If due to improper hookup, or some other problem with the system, the axis feedback fails to detect that axis reaching the configured Motor Encoder Test Increment after moving the axis at least that distance, then abort the test using the MAS instruction and check the encoder wiring.

Test Status

Conditions may occur that make it impossible for the control to properly perform the test operation. When this is the case, the test process is automatically aborted and a test fault is reported and stored in the Test Status output parameter. It is also possible to manually abort a test process using a MAS instruction which results in a test fault reported by the Test Status parameter. Possible values for Test Status are shown in the table below:

Error Message	Code	Definition
Test Success	0	Test Process has been successful.
Test In Process	1	Test is in progress.
Test Aborted	2	Test Process was aborted by user.
Test Time-out	3	Test Process has exceeded timed out (2 seconds).
Test Servo Fault	4	Test Process Failed due to Servo Fault.
Test Increment Fault	5	Test Process Failed due to insufficient test increment distance to make a reliable measure.

To successfully execute a MRHD instruction running the Motor Encoder Test, the targeted axis must be configured as a Servo Axis Type and the axis must be in the Axis Ready state. For other tests this instruction executes properly on either a Servo or Feedback Only axis type. If any of these conditions are not met than the instruction errs.

IMPORTANT When the MRHD instruction is initially executed the In process (.IP) bit is set and the Process Complete (.PC) bit is cleared. The MRHD instruction execution can take multiple scans to execute because it requires transmission of multiple messages to the motion module. The Done (.DN) bit, is not set immediately, but after these messages are successfully transmitted. The In process (.IP) bit is cleared and the Process Complete (.PC) bit is set at the same time that the Done (.DN) bit is set.

Description - AXIS_CIP_DRIVE

The MRHD instruction is used to execute various test diagnostics on the specified CIP axis to test the integrity and, in some cases, the polarity of servo field connections. There are currently test diagnostics supporting drive hookup, encoder hookup, marker hookup and motor commutation hookup. During some of these test processes, the motion module generates output to the external drive to produce a small amount of motion. Measurements made during some of these hookup diagnostic tests are saved as output configuration parameters. MRHD requires only one explicit input parameter, Diagnostic Test. Enter or select the Diagnostic Test to run and the axis to test.

If the targeted axis does not appear in the list of available axes, the axis has not been configured for operation. Use the Tag Editor to create and configure a new axis.

The MRHD instruction uses the CIP axis configuration parameters as input and output. The input configuration parameters that the MRHD uses are shown in this table.

Axis Parameter	Data Type	Units	Meaning
Hookup Test Distance	Real	Position Units	Distance that the axis must travel to satisfy the selected hookup test process.
Hookup Test Time	Real	Seconds	Time that the axis must continue moving to satisfy the selected hookup test
Hookup Test Feedback Channel	USINT	-	Used by the Hookup Test service when the encoder test is selected to determine which feedback channel to test 1 = Feedback 1 2 = Feedback 2.

The CIP axis configuration parameters that MRHD generates as output depend on the specified Hookup Diagnostic.

Motor Encoder Hookup Test

The Motor Encoder Hookup test is selected, the motion module enables the external drive and generates a 1 Volt per second output ramp to the drive while monitoring the encoder feedback. When the axis has moved a distance greater than or equal to the configured Motor Encoder Test Increment, the test voltage is set back to zero and the drive is disabled. The motion module then reports the direction of travel that is stored as one of these output parameters.

Axis Parameter	Data Type	Units	Meaning
Hookup Test Status	USINT	-	Returns the status of the last Run Hookup Test service on the targeted drive axis. The Hookup Test Status attribute can be used to determine when the hookup test service has successfully completed. Conditions may occur, however, that make it impossible for the drive to properly perform the operation. When this is the case, the test process is automatically terminated and a test error is reported that is stored in the Hookup Test Status output parameter. 0 = test process successful 1 = test in progress 2 = test process aborted 3 = test process timed-out 4 = test process faulted 5 = test failed - no feedback 1 counts 6 = test failed - no feedback 2 counts 7-127 = reserved 128 = test not allowed in Axis Test Mode 129-255 = reserved 2
Hookup Test Feedback Direction 1	USINT	-	Reports the direction of axis travel during the last hookup test, as detected by the drive's feedback 1 device. 0 = The drive's feedback 1 device detected a positive direction, that is, increasing counts. 1 = The drive's feedback 1 device detected a negative direction, that is, decreasing counts. 2-255 = reserved The value for Hookup Test Feedback 1 Direction, as determined by the hookup test, does not depend on the current feedback, motor, or motion polarity attribute configuration. This value, combined with the user's definition of forward direction, can be used to configure the various polarity attributes for the correct directional sense.
Hookup Test Feedback Direction 2	USINT	-	Reports the direction of axis travel during the last hookup test, as detected by the drive's feedback 2 device. 0 = The drive's feedback 2 device detected a positive direction, that is, increasing counts. 1 = The drive's feedback 2 device detected a negative direction, that is, decreasing counts. 2-255 = reserved The value for Hookup Test Feedback 2 Direction, as determined by the hookup test, does not depend on the current feedback, motor, or motion polarity attribute configuration. This value, combined with the user's definition of forward direction, can be used to configure the various polarity attributes for the correct directional sense.

If due to improper hookup, or some other problem with the system, the axis feedback fails to detect the axis reaching the configured Motor Encoder Test Increment within 2 seconds, the servo sets the test voltage back to zero and disables the drive. The control reflects this condition through the Test Status axis output parameter. This usually indicates that either the cabling to the drive or the cabling to the encoder is incorrect. Running MRHD with the Encoder Hookup Test selected is an effective method of isolating the problem to the encoder or drive.

Encoder Hookup Test

If the Encoder Test is selected, the motion module does not generate any axis motion, but simply monitors axis encoder feedback. The axis can then be moved by hand or by some other independent drive actuator to generate motion. When the motion module detects that the axis has moved a distance greater than or equal to the configured Motor Encoder Test Increment, the test is complete. The motion module then reports the direction of travel as one of these MRHD output parameters:

Axis Parameter	Data Type	Units	Meaning
Hookup Test Status	USINT	-	Returns the status of the last Run Hookup Test service on the targeted drive axis. The Hookup Test Status attribute can be used to determine when the hookup test service has successfully completed. Conditions may occur, however, that make it impossible for the drive to properly perform the operation. When this is the case, the test process is automatically terminated and a test error is reported that is stored in the Hookup Test Status output parameter. 0 = test process successful 1 = test in progress 2 = test process aborted 3 = test process timed-out 4 = test process faulted 5 = test failed - no feedback 1 counts 6 = test failed - no feedback 2 counts 7-127 = reserved 128 = test not allowed in Axis Test Mode 129-255 = reserved 2
Hookup Test Feedback Direction 1	USINT	-	Reports the direction of axis travel during the last hookup test, as detected by the drive's feedback 1 device. 0 = The drive's feedback 1 device detected a positive direction, that is, increasing counts. 1 = The drive's feedback 1 device detected a negative direction, that is, decreasing counts. 2-255 = reserved The value for Hookup Test Feedback 1 Direction, as determined by the hookup test, does not depend on the current feedback, motor, or motion polarity attribute configuration. This value, combined with the user's definition of forward direction, can be used to configure the various polarity attributes for the correct directional sense.
Hookup Test Feedback Direction 2	USINT	-	Reports the direction of axis travel during the last hookup test, as detected by the drive's feedback 2 device. 0 = The drive's feedback 2 device detected a positive direction, that is, increasing counts. 1 = The drive's feedback 2 device detected a negative direction, that is, decreasing counts. 2-255 = reserved The value for Hookup Test Feedback 2 Direction, as determined by the hookup test, does not depend on the current feedback, motor, or motion polarity attribute configuration. This value, combined with the user's definition of forward direction, can be used to configure the various polarity attributes for the correct directional sense.

If due to improper hookup, or some other problem with the system, the axis feedback fails to detect the axis reaching the configured Motor Encoder Test Increment after moving the axis at least that distance, then abort the test using the MAS instruction and check the encoder wiring.

Marker Hookup Test

If the Marker Test is selected, the motion module does not generate any axis motion, but simply monitors axis encoder feedback. The axis can then be moved by hand or by some other independent drive actuator to generate motion. When the motion module detects a marker (Channel Z) pulse, the test is then complete. The motion module then reports success via the Test Status.

Axis Parameter	Data Type	Units	Meaning
Hookup Test Status	USINT	-	Returns the status of the last Run Hookup Test service on the targeted drive axis. The Hookup Test Status attribute can be used to determine when the hookup test service has successfully completed. Conditions may occur, however, that make it impossible for the drive to properly perform the operation. When this is the case, the test process is automatically terminated and a test error is reported that is stored in the Hookup Test Status output parameter. 0 = test process successful 1 = test in progress 2 = test process aborted 3 = test process timed-out 4 = test process faulted 5 = test failed - no feedback 1 counts 6 = test failed - no feedback 2 counts 7-127 = reserved 128 = test not allowed in Axis Test Mode 129-255 = reserved 2
Hookup Test Feedback Direction 1	USINT	-	Reports the direction of axis travel during the last hookup test, as detected by the drive's feedback 1 device. 0 = The drive's feedback 1 device detected a positive direction, that is, increasing counts. 1 = The drive's feedback 1 device detected a negative direction, that is, decreasing counts. 2-255 = reserved The value for Hookup Test Feedback 1 Direction, as determined by the hookup test, does not depend on the current feedback, motor, or motion polarity attribute configuration. This value, combined with the user's definition of forward direction, can be used to configure the various polarity attributes for the correct directional sense.

Hookup Test Feedback Direction 2	USINT	-	Reports the direction of axis travel during the last hookup test, as detected by the drive's feedback 2 device. 0 = The drive's feedback 2 device detected a positive direction, that is, increasing counts. 1 = The drive's feedback 2 device detected a negative direction, that is, decreasing counts. 2-255 = reserved The value for Hookup Test Feedback 2 Direction, as determined by the hookup test, does not depend on the current feedback, motor, or motion polarity attribute configuration. This value, combined with the user's definition of forward direction, can be used to configure the various polarity attributes for the correct directional sense.

If due to improper hookup, or some other problem with the system, the axis feedback fails to detect that axis reaching the configured Motor Encoder Test Increment after moving the axis at least that distance, then abort the test using the MAS instruction and check the encoder wiring.

Commutation Test

The Commutation test only applies to PM motors. This test applies current to the motor to align the rotor and check for proper phasing of a UVW encoder or Hall sensor, if applicable. Finally, the test measures the commutation offset.

Tip: For linear stages, make sure there is enough travel. If there is not enough travel, the test produces a fault.

Axis Parameter	Data Type	Units	Definition
Hookup Test Status	USINT	-	Returns the status of the last Run Hookup Test service on the targeted drive axis. The Hookup Test Status attribute can be used to determine when the hookup test service has successfully completed. Conditions may occur, however, that make it impossible for the drive to properly perform the operation. When this is the case, the test process is automatically terminated and a test error is reported that is stored in the Hookup Test Status output parameter. 0 = test process successful 1 = test in progress 2 = test process aborted 3 = test process timed-out 4 = test process faulted 5 = test failed - no feedback 1 counts 6 = test failed - no feedback 2 counts 7-127 = reserved 128 = test not allowed in Axis Test Mode 129-255 = reserved 2

Hookup Test Commutation Polarity	USINT	-	Reports if the UVW phasing of the Encoder or Hall Sensor match the phasing of the Motor. If the motor and UVW commutation phasing do not match the Commutation Polarity is Normal. If it is determined that the phasing for the motor and commutation device do not match, this parameter reports that the Commutation direction is Inverted. This value can be used to configure the Commutation Polarity attribute. 0 = normal 1 = inverted 2-255 = reserved
Hookup Test Commutation Offset	Real	Electrical Degrees	The Hookup Test Commutation Offset reports the measured commutations offset of a PM motor during the Commutation Test. This represents the value that must be applied to the motor position accumulator in order to align the Electrical Angle signal with motor stator windings. This value can be used to configure the Commutation Offset attribute.

Test Status

This parameter returns the status of the last Run Hookup Test service on the targeted drive axis. Conditions may occur that make it impossible for the control to properly perform the test operation. When this is the case, the test process is automatically aborted and a test fault is reported and stored in the Hookup Test Status output parameter. Possible values for Test Status are shown in the table below:

Error Message	Code	Definition
Test Success	0	Test Process has been successful.
Test In Process	1	Test is in progress.
Test Aborted	2	Test Process was aborted by user.
Test Time-out	3	Test Process has exceeded timed out (2 seconds).
Test Servo Fault	4	Test Process Failed due to Servo Fault.
No Feedback 1	5	Test Process Failed - no feedback 1 counts.
No Feedback 2	6	Test Process Failed - no feedback 2counts.
Test not allowed in Axis Test Mode	128	Test Process Failed - Axis Test Mode enabled.

In this transitional instruction, the relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of incorrect type is passed to the Axis operand. See Common Attributes for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See Motion Error Codes (ERR) for Motion Instructions.

Extended Error Codes

Extended Error Codes provide additional instruction specific information for the Error Codes that are generic to many instructions. Motion Error Codes (ERR) for Motion Instructions. The following Extended Error codes help to pinpoint the problem when the MRHD instruction receives an error message.

Associated Error Code (decimal)	Extended Error Code (decimal)	Meaning
SERVO_MESSAGE_FAILURE (12)	Process terminated on request (1)	Test execution followed by an instruction to shutdown/disable drive, or a motion stop instruction or a Processor change requests a cancel of the Test.
SERVO_MESSAGE_FAILURE (12)	Object Mode conflict (12)	Axis is in shutdown.
SERVO_MESSAGE_FAILURE (12)	Device in wrong state (16)	Incorrect Tune Process order. (SERCOS)
TEST_PROCESS_ERROR (15)	Motor Feedback Test Prohibited (1)	Axis Test Mode enabled - Motor and Feedback Test is not supported in this configuration.

Status Bits

MRHD Changes to Status Bits

Bit Name	State	Meaning
DriveEnableStatus	TRUE	The axis is in Drive Control state. The Drive Enable output is active while the Tuning Profile is running.
TestStatus	TRUE	The axis is running a testing process.

Example 1

When the input conditions are true, the controller runs the encoder diagnostic test on axis1.

Ladder Diagram

Neutral Test Representation for RLL

MRHD(Axis1, MRHD_1, Motor_Encoder);

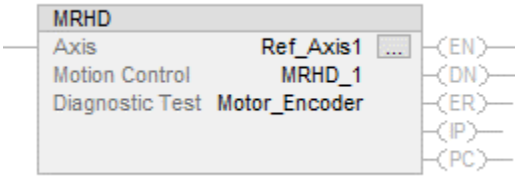
Structured Text

MRHD(Axis1,MRHD_1, Motor_Encoder);

Example 2

When the input conditions are true, the controller runs the encoder diagnostic test on Axis1 through a reference to it

Ladder Diagram



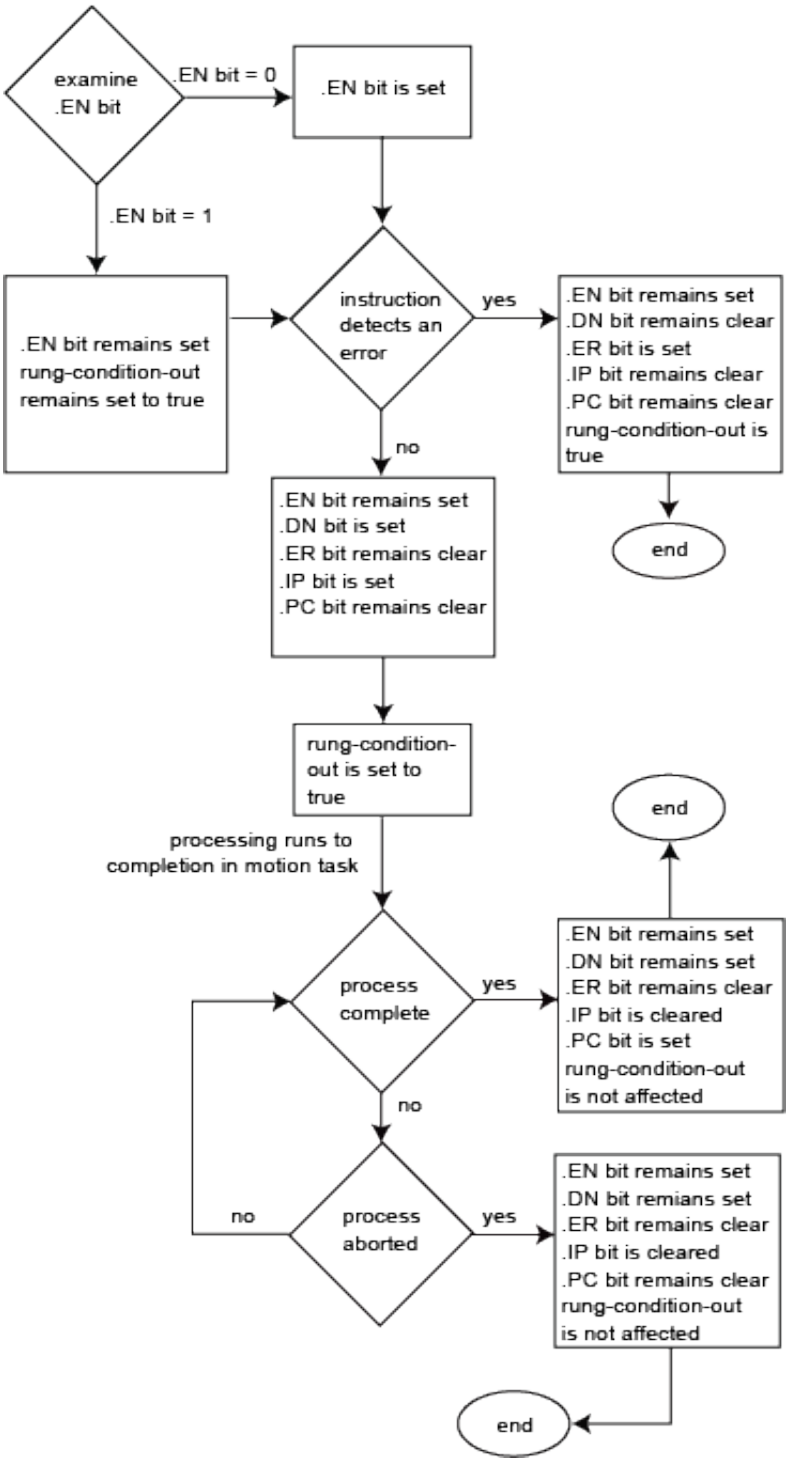
Neutral Text Representation for RL

MRHD(Ref_Axis1, MRHD_1, Motor_Encoder);

Structured Tex

MRHD(Ref_Axis1, MRHD_1, Motor_Encoder);

MRHD Flow Chart (True)



Modify Motion Configuration Parameters

In your ladder logic program, you can modify motion configuration parameters using the SSV instruction. For example, you can change position loop gain, velocity loop gain, and current limits within your program.

Multi-Axis Coordinated Motion Instructions

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

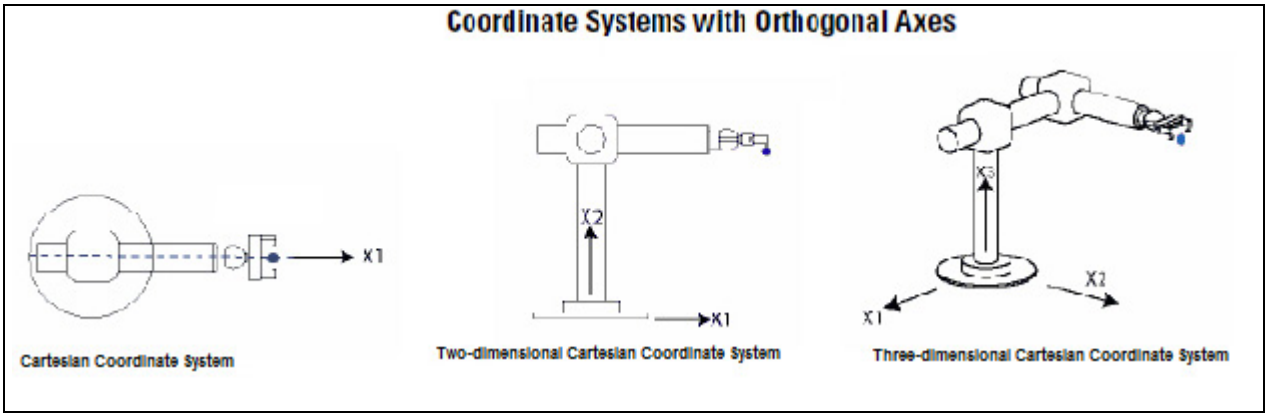
MCS	MCLM	MCCM	MCCD	MCSD	MCSR	MCT	MCTP

MDCC	MCTPO	MCPM	MCTO

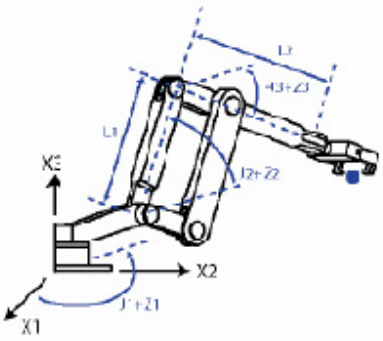
Function Block

Not available

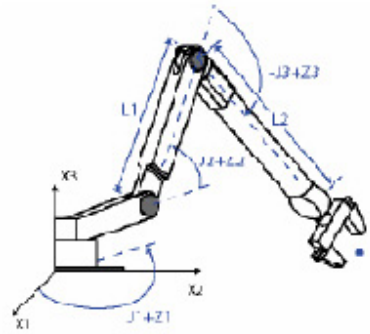
Coordinate system examples are:



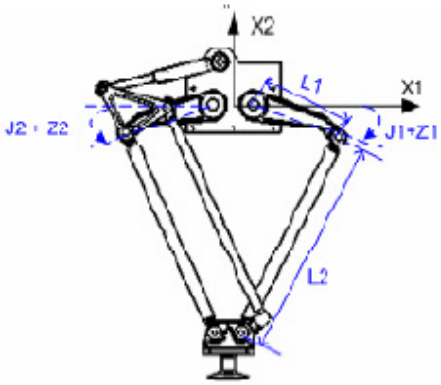
Coordinate Systems with Non-orthogonal Axes



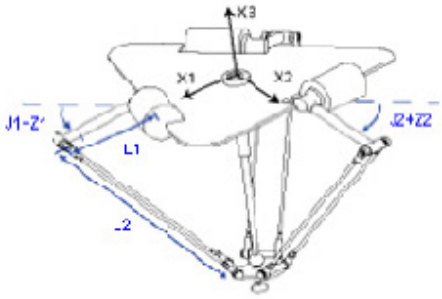
Articulated Dependent Coordinate System



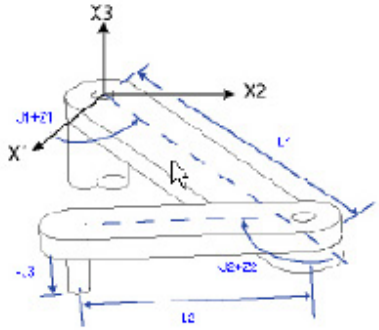
Articulated Independent Coordinate System



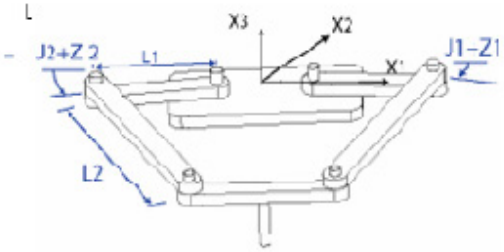
Delta Two dimensional Coordinate System



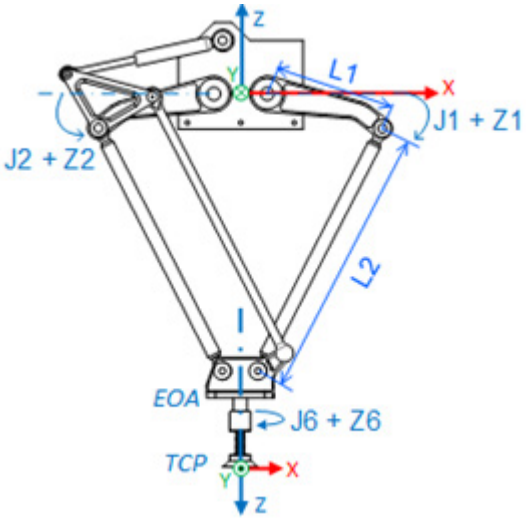
Delta Three dimensional Coordinate System



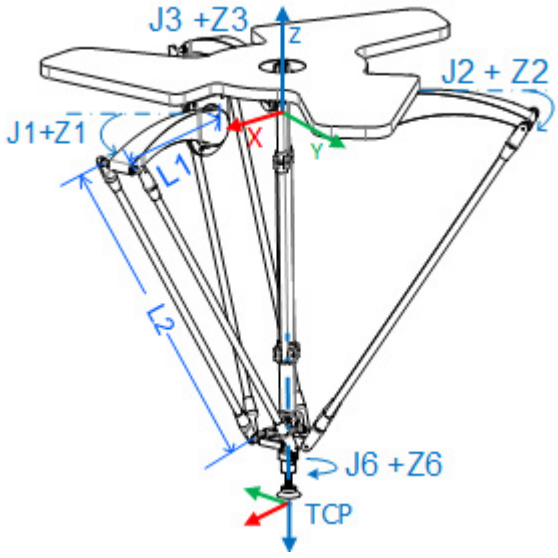
SCARA Independent Coordinate System



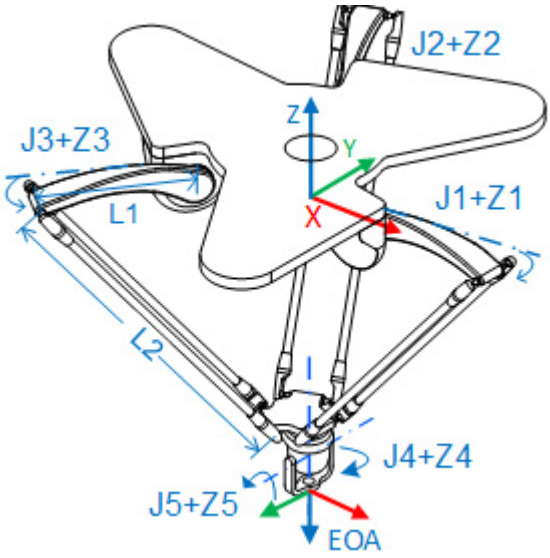
SCARA Delta Coordinate System



Delta J1J2J6 Coordinate System



Delta J1J2J3J6 Coordinate System



Delta J1J2J3J4J5 Coordinate System

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. This instruction is only available on supported controllers.	MCTP
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

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, this ladder diagram uses Coordinate System cs1 to blend Move1 into Move2.

Example Ladder Diagram for Blended Instructions

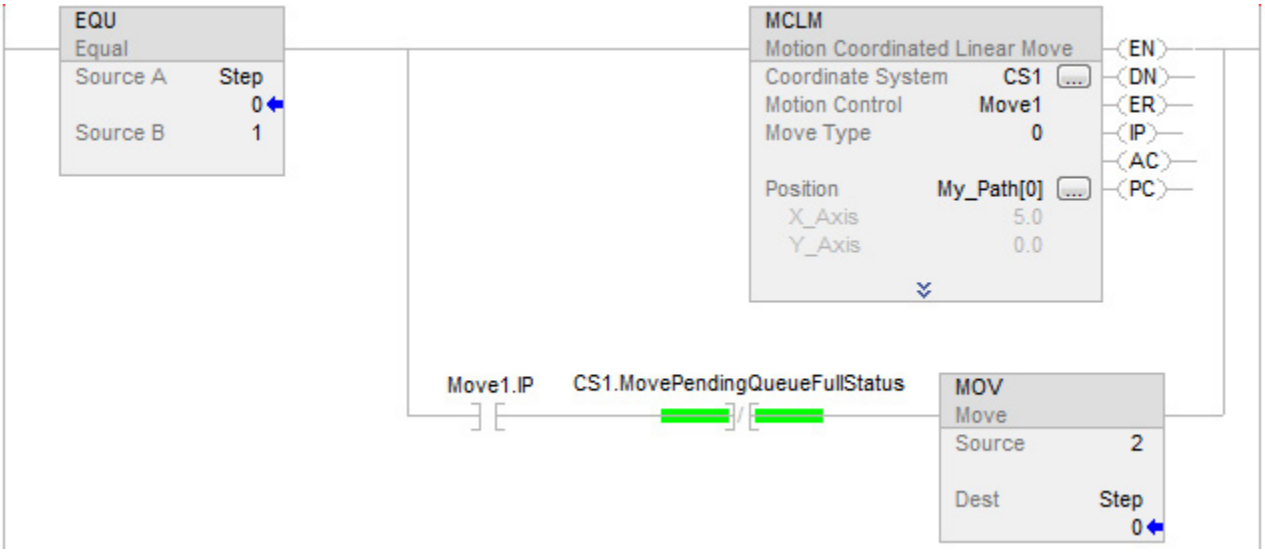
If Step=1 then

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

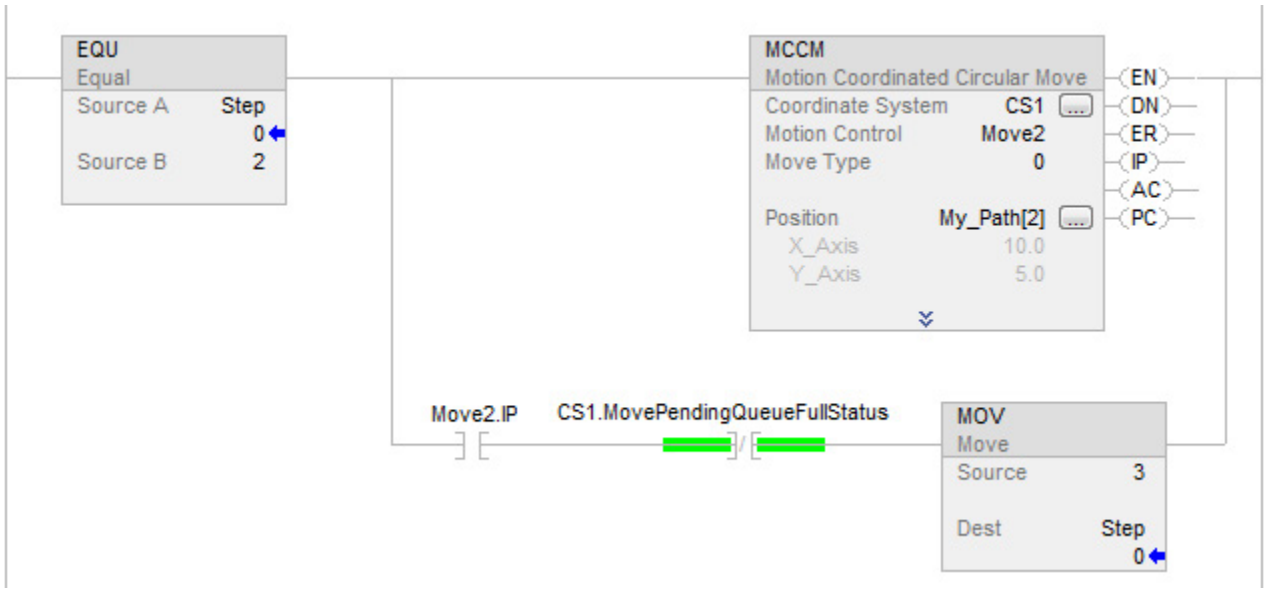
Move1 is already happening.

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.

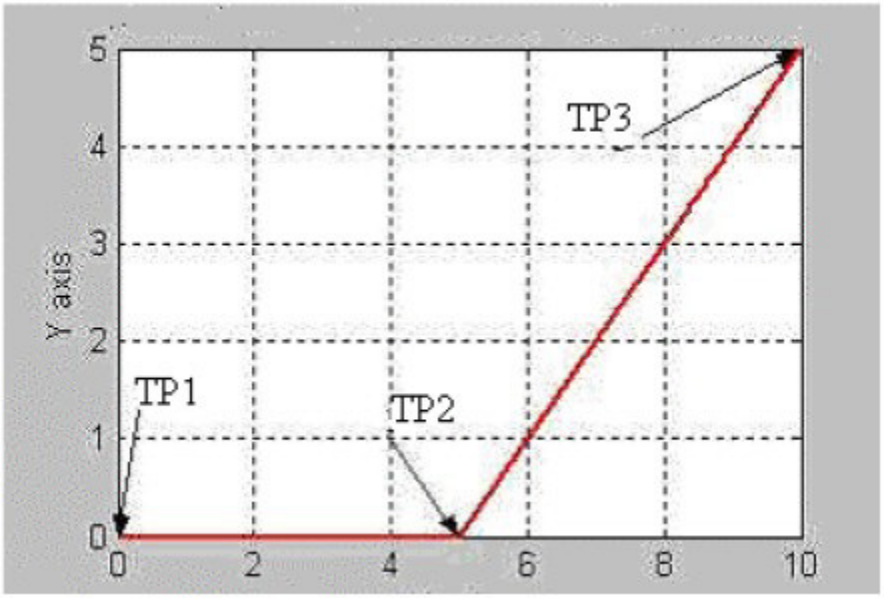
Bit Parameters

When	Then
One instruction is active and a second instruction is pending in the queue.	• MovePendingStatus bit = 1 • MovePendingQueueFullStatus bit = 1 • You can't queue another instruction
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. 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

linear → linear move



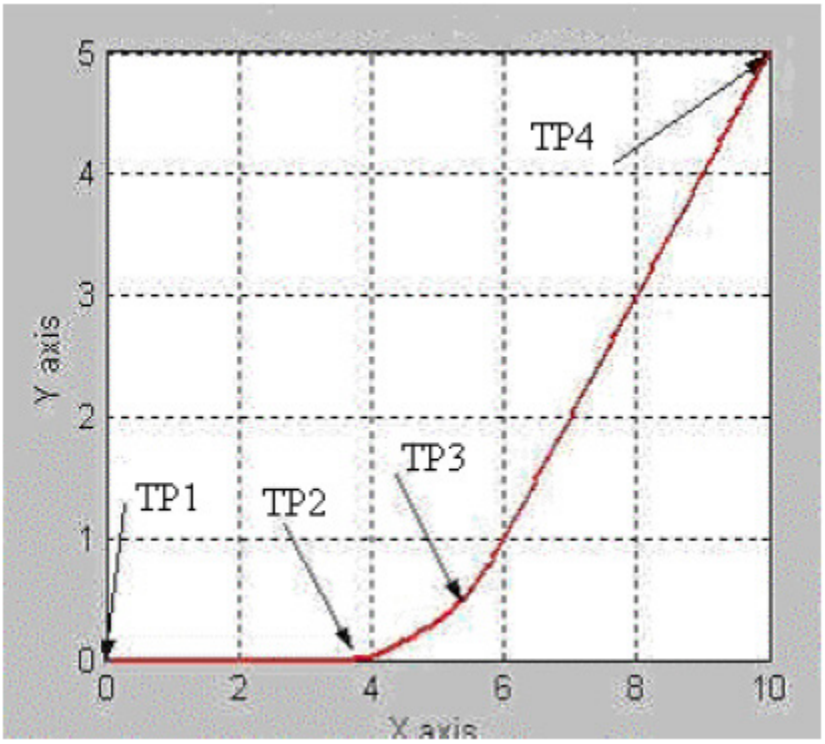
The following table shows the bit status at the various transition points shown in the preceding graph with termination type of either Actual Tolerance or No Settle.

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

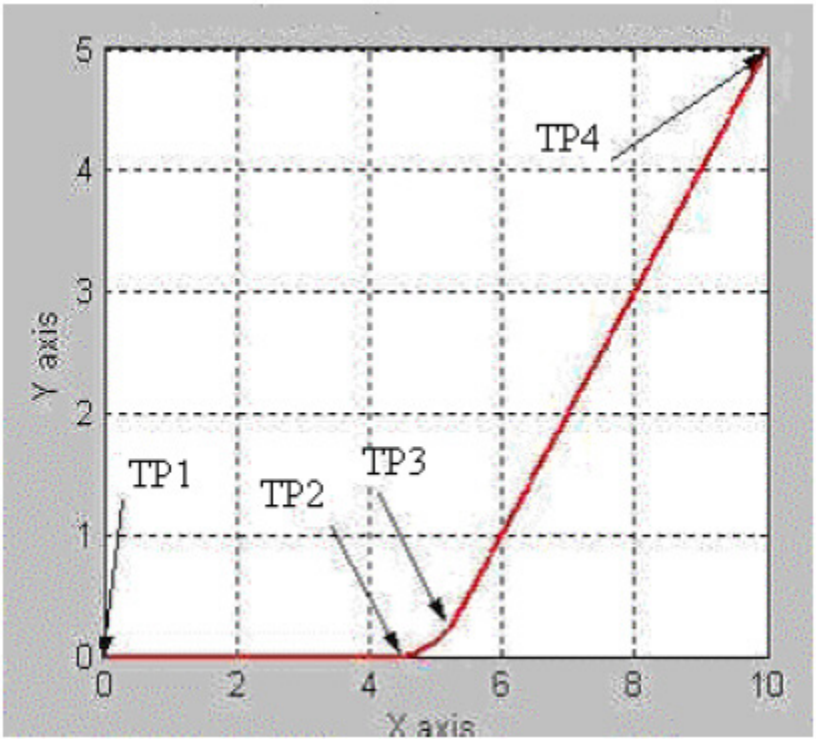


The following table shows the bit status at the various transition points shown in the preceding graph with termination type of No Decel. For No Decel termination type distance-to-go for transition point TP2 is equal to deceleration distance for the Move1 instruction. If Move 1 and Move 2 are collinear, 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
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

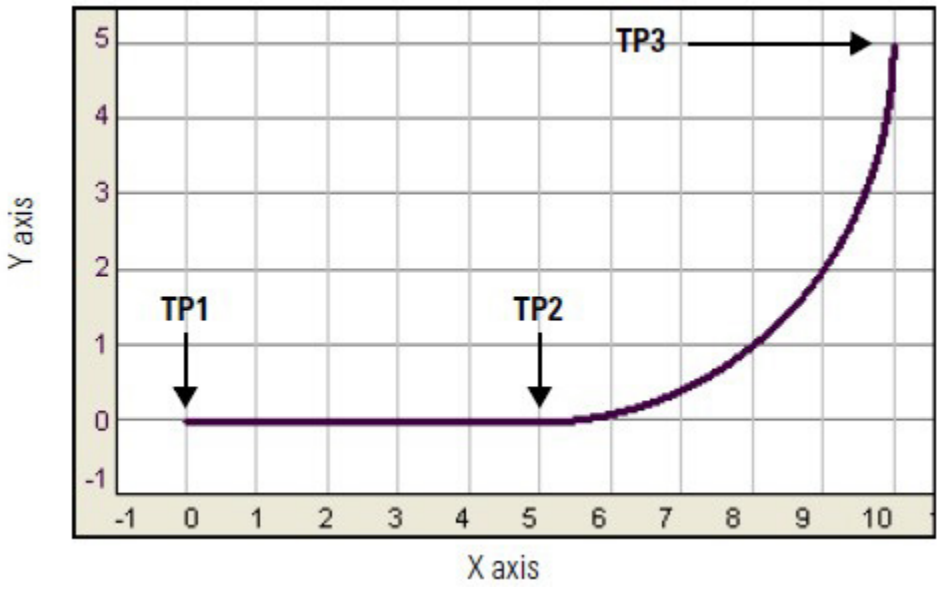


The following table shows the bit status at the various transition points shown in the preceding graph with termination type of Command Tolerance. For Command Tolerance termination type distance-to-go for transition point TP2 is equal to Command Tolerance for the Coordinate System cs1.

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 on page 517

Speed, Acceleration, Deceleration, and Jerk Enumerations for Coordinated Motion on page 508

Status Bits for Motion Instructions when (MCLM, MCCM) when MDCC is Active on page 514

Input and Output Parameters Structure for Coordinate System Motion Instructions on page 527

Changing Between Master Driven and Time Driven Modes for Coordinated Motion Instructions on page 516

Master Driven Coordinated Control (MDCC)

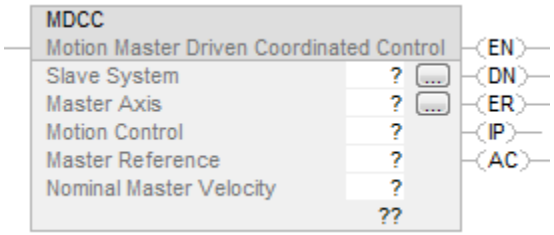
This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The Master Driven Coordinated Control (MDCC) instruction defines a Master:Slave relationship between a Master Axis and a Slave Coordinate System.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MDCC (SlaveCoordinateSystem, MasterAxis, MotionControl, MasterReference, NominalMasterVelocity);

Operands

Ladder Diagram and Structured Text

Operand	Type CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Type CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Format	Description
Slave System	COORDINATE_SYSTEM	COORDINATE_SYSTEM	tag	The Coordinate System that the Master Axis controls when the motion planner is in Master Driven mode. Ellipsis launches the Coordinate System properties dialog. Upon verification, you receive a verification error if the Slave is a non-Cartesian Coordinate System or if the Master Axis is in the Slave Coordinate System. The MDCC link is broken when the following instructions are executed: On any axis in the Slave Coordinate System or the Slave Coordinate System: MAS (All), MCS (All), MGS, MASD, MCSD, MGSD, a mode change. Note that MAS (anything other than All) and MCS do NOT break the MDCC link. The Shutdown instructions (MGSD, MASD, MCSD) never go IP. On the Master Axis: MASD, MCSD, and MGSD. Note that MAS and MCS for any Stop Type, including All, do NOT break the MDCC link. A mode change (Rem Run to Rem Prog or Rem Prog to Rem Run) or an axis fault also breaks the MDCC link. For controllers that support the REF_TO motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.

Master Axis	AXIS_CONSUMED AXIS_CIP_DRIVE AXIS_VIRTUAL REF_TO_AXIS_CONSUMED REF_TO_AXIS_CIP_DRIVE REF_TO_AXIS_VIRTUAL **Tip:** AXIS_CONSUMED is supported by Compact GuardLogix 5580, CompactLogix 5380, and CompactLogix 5480 controllers only.	AXIS_CONSUMED AXIS_SERVO AXIS_SERVO_DRIVE AXIS_GENERIC AXIS_GENERIC_DRIVE AXIS_CIP_DRIVE AXIS_VIRTUAL	tag	Any configured Single Axis that the Slave Coordinate System follows. The Master Axis can be any axis that was configured.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	tag	Control tag for the instruction.
Master Reference	UINT32	UINT32	immediate tag	Selects the Master position source as either Actual Position (0) or Command Position (1).
Nominal Master Velocity	REAL	REAL	immediate tag	Used as an additional conversion constant for master driven MCPM moves programmed in percentage of maximum.

Master Reference

The Master Reference for an MDCC instruction selects the Master Axis position source.

The enumerations for Master Reference Axis are:

Actual – Slave motion is generated from the actual (current) position of the Master Axis as measured by its encoder or other feedback device.

Command – Slave motion is generated from the command (desired) position of the Master Axis. Because there is no Command Position for a Feedback Only Axis, if you select either Actual or Command for Master Reference, the Actual Position of the Master Axis is used. The Actual and Command Position are always the same for this case. No error is generated. Because there is no Actual Position for a Virtual axis, if you select either Actual or Command for Master Reference, the Command Position is used. No error is generated.

An error is generated if a MDCC instruction is executed that changes the Master Reference of a Slave Coordinate System that is in motion. The new MDCC instruction errors and the original one remains active.

Nominal Master Velocity

If master driven coordinated move is programmed to run at a speed that equals 100% of maximum and master axis runs at its Nominal Master Velocity, then the coordinated move will run at the configured maximum speed of the coordinate system.

Calculate actual master axis velocity in percentage of the maximum as a percentage of actual master velocity with respect to nominal master velocity. For example, nominal master vel =50 ips therefore actual vel = 40 ips = (40/50) * 100 = 80%.

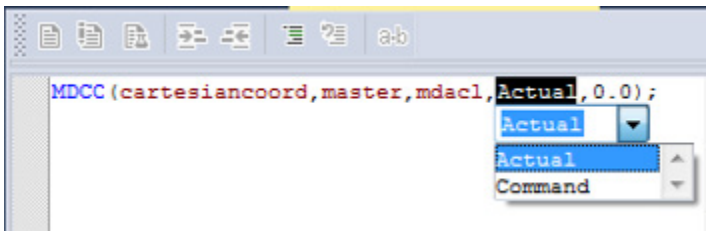
Calculate actual slave velocity in a percentage of the maximum as a product of programmed slave velocity in the percentage of the maximum and actual master axis velocity percentage of the maximum. For example, programmed slave velocity = 50% of the maximum, master velocity = 80% of the maximum, final slave velocity = 80 * 50 = 40% of the maximum.

Convert final slave velocity in a percentage of maximum to Units/sec as usual. For example, final velocity = 40%, coordinate system maximum is speed 200 Units/sec, final velocity limit is 200 * (40/100) = 80 Units/sec.

Structured Text

See Structured Text Syntax for more information on the syntax of expressions within structured text.

You have the option to browse for enumerations in the Structured Text Editor as shown.



Outputs

Mnemonic	Description
.EN (Enable) Bit 31	The enable bit is set when the rung transitions from false-to-true and stays set until the rung goes false.
.DN (Done) Bit 29	The done bit is set when the coordinate MDCC instruction is successfully initiated.
.ER (Error) Bit 28	The error bit is set when there is an invalid combination of parameters in the MDCC instruction.
.IP (In Process) Bit 26	The in process bit is set when the MDCC instruction is activated and reset by an instruction (for example, the MCSD instruction).
.AC (Active) Bit 23	The active bit is set when an MCLM or MCCM is activated (that is, when the AC bit of the MCLM or MCCM instruction is set) on a Coordinate System that is selected as the Slave Coordinate System of the MDCC instruction.

Description

MDCC is a transitional instruction:

In relay ladder, toggle the Rung-condition-in from false to true each time the instruction should execute.

In structured text, condition the instruction so that it only executes on a transition.

The Motion Master Driven Coordinate Control instruction (MDCC) is used by the Master Driven Speed Control (MDSC) to synchronize one or more motion axes or Coordinate System to a common Master Axis when the Slave System is a Coordinate System.

The MDCC instruction defines the relationship between the external Master Axis and the Slave Coordinate System for the MCLM and the MCCM Instructions. When an MDCC is executed (goes IP), the specified Slave Coordinate System in the MDCC instruction is logically geared with the designated Master Axis. After motion on the Master Axis is initiated, all the axes in the Coordinate System specified as the Slave Coordinate System follow the Master Axis motion at the programmed dynamics of the programmed instruction.

There are no changes in any active motion when a new MDCC instruction is activated. Activating an MDCC instruction just puts the parameters programmed in the MDCC instruction into a pending state. The parameters in the pending MDCC instruction are changed if you execute a succeeding MDCC instruction before a new MCLM or MCCM instruction is activated. The MDCC becomes active (AC bit is set) only when all queued motion is complete and the motion queue is empty.

All motion in the queue keeps using the same Master Axis even if there is a pending MDCC with a different master. The values in the pending MDCC instruction are only used when the next MCLM or MCCM instruction is activated on the Slave Coordinate System when the queue is empty, or a MCLM or MCCM is executed (goes IP) with a Merge type of All or Coordinate. (Note that this is because the merge empties the queue.)

Motion Direct Command and the MDCC Instruction

To obtain Motion Direct support for the MDCC instruction, you must first program an MDCC in one of the supporting program languages before you execute an MCLM or MCCM in Master driven Mode.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of the incorrect type is passed to the Coordinate System or Axis operands. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false and rung-condition-in is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See Motion Error Codes (ERR) for Motion Instructions.

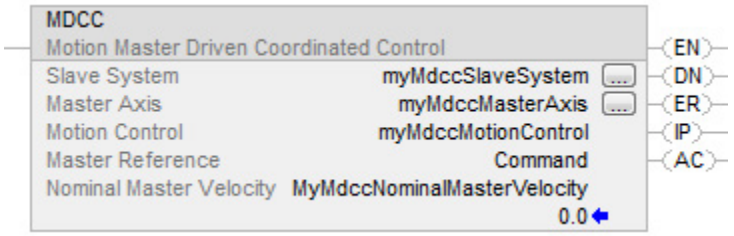
Verification Errors

An invalid or No Master Axis will cause new errors to be generated when verified by the programming software. The following conditions may cause this error:

- The Master Axis is a member of the Slave Coordinate System.
- The Master Axis or the Slave Coordinate System is not configured.
- The Master Axis or an axis in the Slave Coordinate System is inhibited.
- A redefine position is in progress.
- Home of the Master axis or an axis in the Slave Coordinate System is in progress.

Example

Ladder Diagram



Structured Text

MDCC(myMdccSlaveSystem,myMdccMasterAxis,myMdccMotionControl,Command,myMdccNominalMasterVelocity);

See also

Structured Text Syntax on page 665

Motion Error Codes (.ERR) on page 581

Index Through Arrays on page 691

Motion Calculate Transform Position (MCTP)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Important:	You can also use this instruction with the following controllers: 1756-L6 controllers 1756-L7S controllers 1769-L18ERM controllers 1769-L27ERM controllers 1769-L30ERM controllers 1769-L33ERM controllers 1769-L36ERM controllers

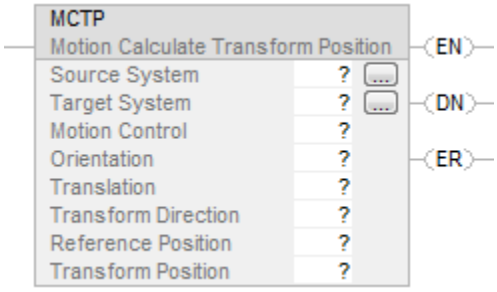
Use the MCTP instruction to calculate the position of a point in one coordinate system to the equivalent point in a second coordinate system.

Important:	Tags used for the motion control attribute of instructions should only be used once. Re-use of the motion control tag in other instructions can cause unintended operation. This may result in damage to equipment or personal injury.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MCTP(Source System, Target System, Motion Control, Orientation, Translation, Transform Direction, Reference Position, Transform Position);

Operands

Ladder Diagram and Structured Text

Operand	Type	Format	Description

Source System	COORDINATE_SYSTEM	Tag	Cartesian coordinate system for Cartesian positions of the robot. For controllers that support the REF_TO motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.
Target System	COORDINATE_SYSTEM	Tag	Reference to Non-Cartesian coordinate system that controls the actual equipment. For controllers that support the REF_TO motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.
Motion Control	MOTION_INSTRUCTION	Tag	Control tag for the instruction.
Orientation	REAL[3]	Array	Do you want to rotate the target position around the X1, X2, or X3 axis? **If** / **Then** No: Leave the array values at zero. Yes: Enter the degrees of rotation into the array. Put the degrees of rotation around X1 in the first element of the array, and so on. Use an array of three REALs even if a coordinate system has only one or two axes.
Translation	REAL[3]	Array	Do you want to offset the target position along the X1, X2, or X3 axis? **If** / **Then** No: Leave the array values at zero. Yes: Enter the offset distances into the array. Enter the offset distances in coordinate units. Put the offset distance for X1 in the first element of the array, and so on. Use an array of three REALs even if a coordinate system has only one or two axes.

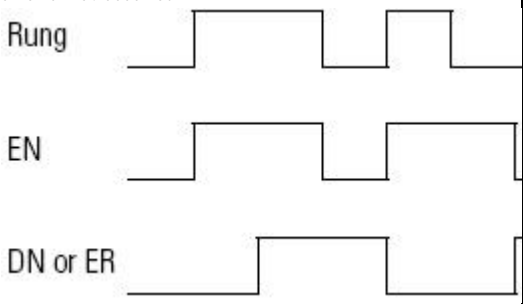
Operand	Type	Format	Description				
Transform Direction	DINT	Immediate	**For Robot Type**	**To calculate**	**With the base turned to the**	**And the robot is**	**Select**
			All	Cartesian Position	→	→	Forward
			Cartesian Delta 2D Delta 3D SCARA Delta	Joint Angles	→	→	Inverse
			Articulated Independent Articulated Dependent SCARA Independent	Joint Angles	Same Quadrant as the Point	Right Arm Configuration	Inverse Right Arm
						Left Arm Configuration	Inverse Left Arm

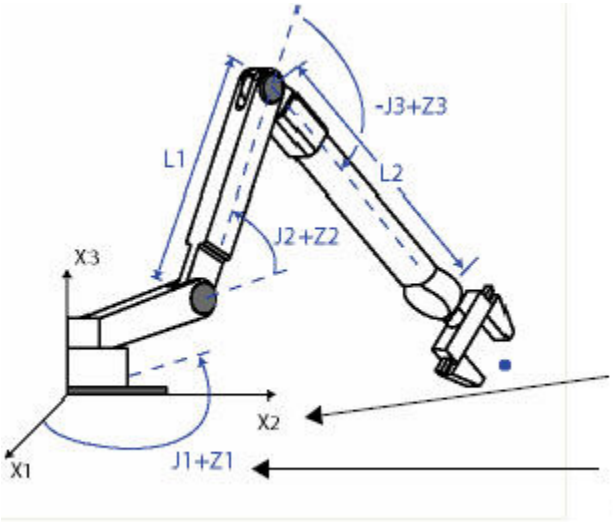
					Opposite Quadrant from the Point	Right Arm Configuration	Inverse Right Arm Mirror
						Left Arm Configuration	Inverse Left Arm Mirror
Reference Position	REAL[3]	Array	If the transform direction is:		Then enter an array that has the:		
			Forward		Joint Angles		
			Inverse		Cartesian Positions		
Transform Position	REAL[3]	Array	Array that stores the calculated position				

Enter the transform direction without spaces. For example, a transform direction of Inverse Left Arm is entered as InverseLeftArm.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Data Type

To see if	Check if this bit is on	Data Type	Notes
The rung is true	EN	BOOL	Sometimes the EN bit stays on even if the rung goes false. This happens if the rung goes false before the instruction is done or an error has occurred. 
The instruction is done.	DN	BOOL	
An error happened	ER	BOOL	Identify the error number listed in the error code field of the Motion control tag then, refer to Motion Error Codes.



The MCTP instruction is similar to the MCT instruction except the MCTP instruction doesn't start a transform. It calculates a position once each time you execute it.

Programming Guidelines

Follow these guidelines to use an MCTP instruction.

MCTP Instruction Guidelines

Guideline	Example and Notes
Toggle the rung from false to true to execute the instruction.	This is a transitional instruction. In a ladder diagram, toggle the Rung-condition-in from false to true each time you want to execute the instruction.
In structured text, condition the instruction so that it only executes on a transition.	In structured text, instructions execute each time they are scanned. Condition the instruction so that it only executes on a transition. Use either of these methods: Qualifier of an SFC action Structured text construct

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of the incorrect type is passed to the Coordinate System operands.See *Common Attributes* for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See *Motion Error Codes (.ERR)* for Motion Instructions.

Extended Error Codes

Use Extended Error Codes (EXERR) for more instruction about an error. See *Motion Error Codes (.ERR)* for Motion Instructions.

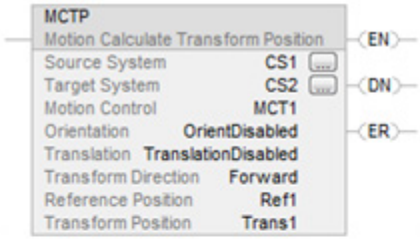
Changes to Status Bits

None

Data Flow of MCTP Instruction Between Two Coordinate Systems

The following illustrations show the flow of data when an MCTP Instruction is executed to perform a forward transformation and an inverse transformation. The CS1 indicator represents a Cartesian coordinate system containing X1, X2 and X3 axes as the source of the MCTP instruction. The CS2 indicator represents the joint coordinate system containing J1, J2 and J3 axes as the target of the MCTP instruction.

Data Flow When a Move is Executed with an MCTP Instruction - Forward Transform



Input Data

CS2: DATA	SOURCE
Link Lengths (L1, L2)	Coordinate System dialog
Base Offsets (X1b, X2b, X3b)	Coordinate System dialog
End Effector Offsets (X1e, X2e, X3e)	Coordinate System dialog
Zero Angle Orientations (Z1, Z2, Z3)	Coordinate System dialog
Orientation (Array [3])	Instruction Faceplate
Translations (Array [3])	Instruction Faceplate
Transform Direction	Instruction Faceplate
Reference Position Typically Joint - Target	Instruction Faceplate

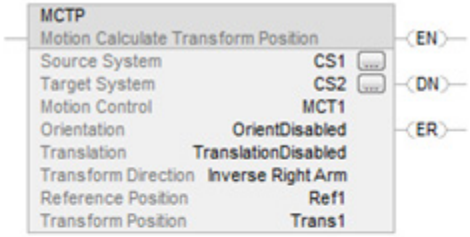
Executed Instruction

MCTP

Computed Output

CS1: Data	Destination
Cartesian Positions (X1, X2, X3)	Instruction Faceplate
Typically Cartesian	Transform Position

Data Flow When a Move is Executed with an MCTP Instruction - Inverse Transform



Input Data

CS1: DATA	SOURCE
Link Lengths (L1, L2)	Coordinate System dialog
Base Offsets (X1b, X2b, X3b)	Coordinate System dialog
End Effector Offsets (X1e, X2e, X3e)	Coordinate System dialog
Zero Angle Orientations (Z1, Z2, Z3)	Coordinate System dialog
Orientation (Array [3])	Instruction Faceplate
Translations (Array [3])	Instruction Faceplate
Transform Direction	Instruction Faceplate
Reference Position Typically Cartesian - Source	Instruction Faceplate

Executed Instruction

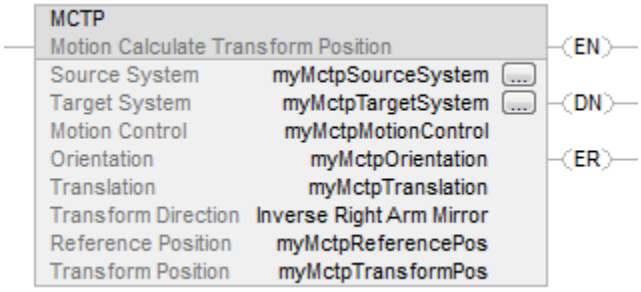
MCTP

Computed Output

CS2: Data	Destination
Joint Positions (J1, J2, J3)	Instruction Faceplate
Typically Joint	Transform Position

Examples

Ladder Diagram



Structured Text

MCTP(myMctpSourceSystem, myMctpTargetSystem, myMctpMotionControl, myMctpOrientation, myMctpTranslation, InverseRightArmMirror, myMctpReferencePos, myMctpTransformPos);

See also

Motion Error Codes (.ERR) on page 581

Multi-Axis Coordinated Motion Instructions on page 359

Common Attributes on page 691

Structured Text Syntax on page 665

Motion Coordinated Transform with Orientation (MCTO)

This information applies to the Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Use the MCTO instruction to establish a bidirectional transform that is set up between a Cartesian and a robot system with coordinates that are joint axes of a robot. The XYZ translation coordinates and the RxRyRz orientation coordinates in the fixed angle convention define the Cartesian coordinates. The geometrical configurations of the robots typically have joint axes that are not orthogonal. The coordinate system type, such as Delta, specifies the geometrical configurations.

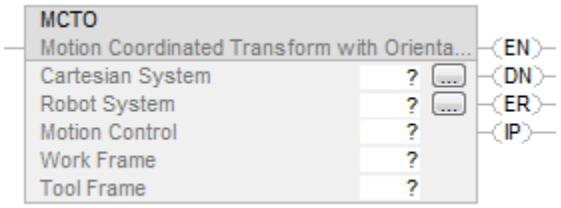
This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	No	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MCTO(CartesianSystem, RobotSystem, MotionControl, WorkFrame, ToolFrame);

Operands

IMPORTANT Do not use the same tag name for more than one instruction in the same program. Do not write to any instruction output tag under any circumstances.

ATTENTION: If instruction operands are changed while in Run mode, the pending edits must be accepted and the controller mode cycled from Program to Run for the changes to take effect.

Configuration

The following table provides the operands used to configure the instruction. These operands cannot be changed at runtime.

Operand	Data Type	Format	Description
Cartesian System	COORDINATE_SYSTEM	tag	Cartesian coordinate system used to program the moves. Reference to Cartesian coordinate system used to program the moves. For controllers that support the REF_TO motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.
Robot System	COORDINATE_SYSTEM	tag	Non-Cartesian coordinate system that controls the actual equipment. Reference to Non-Cartesian coordinate system that controls the actual equipment. For controllers that support the REF_TO motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.
Motion Control	MOTION_INSTRUCTION	tag	Control tag for the instruction.
Work Frame	POSITION_DATA	immediate tag	Work Frame offsets are the offsets used to locate the user Work frame of the Robot relative to the origin of the Robot base frame. These offsets consist of an XYZ and RxRyRz value. This allows the programs to be written in the user work space or world frame and transformed to the Robot base frame. To rotate the target position around the X, Y, or Z axis or offset the target position along the X, Y, or Z axis of the Robot base coordinate system, enter the degrees of rotation into the Rx, Ry, and Rz tag members in units of degrees of rotation, and enter the offset distances into the X, Y, and Z tag members in coordination units. Set the ID member to a value greater than or equal to zero. To maintain the work frame at the robot base frame, leave structure values at zero, or set the operand tag value to zero.

Operand	Data Type	Format	Description
Tool Frame	POSITION_DATA	immediate tag	Tool Center Point (TCP) offsets are the offsets used to locate the tool center relative to the center of the End of Arm. These offsets consist of an XYZ and RxRyRz value. To have the target position account for an attached tool having translation and/or orientation offsets, enter the tool offset distances into the X, Y, and Z tag members in coordination units. Enter the degrees of tool rotation into the Rx, Ry, and Rz tag members in units of degrees of rotation. Set the ID member to a value greater than or equal to zero. To have the target position reflect only the point at the end-of-arm, leave the structure values at zero, or set the operand tag value to zero.

For further information of configuring coordinate systems refer to the Motion Coordinate System User Manual publication MOTION-UM002.

IMPORTANT Do not write to any instruction output tag under any circumstances.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or reference of the incorrect type is passed to the Coordinate System operands. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Same as Rung-condition-in is false.
Rung-condition-in is false	The .EN, .DN, and .ER are cleared to false.
Rung-condition-in is true and .EN bit is false	The .EN bit is set to true and the instruction executes.
Rung-condition-in is true and .EN bit is true	N/A
Postscan	Same as Rung-condition-in is false.

Structured Text

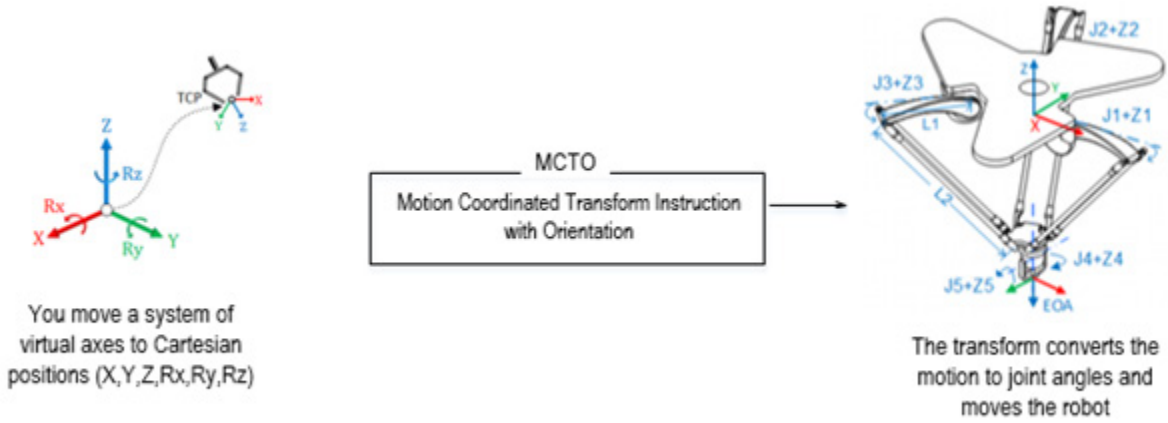
Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false followed by Rung-condition-in is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Data Flow of MCTO instruction between two coordinate systems

Establish the reference frame for the joint coordinate system to ensure the transformations work properly. Refer to the applicable geometry configuration topic for further information on establishing the reference frame for a robot coordinate system.

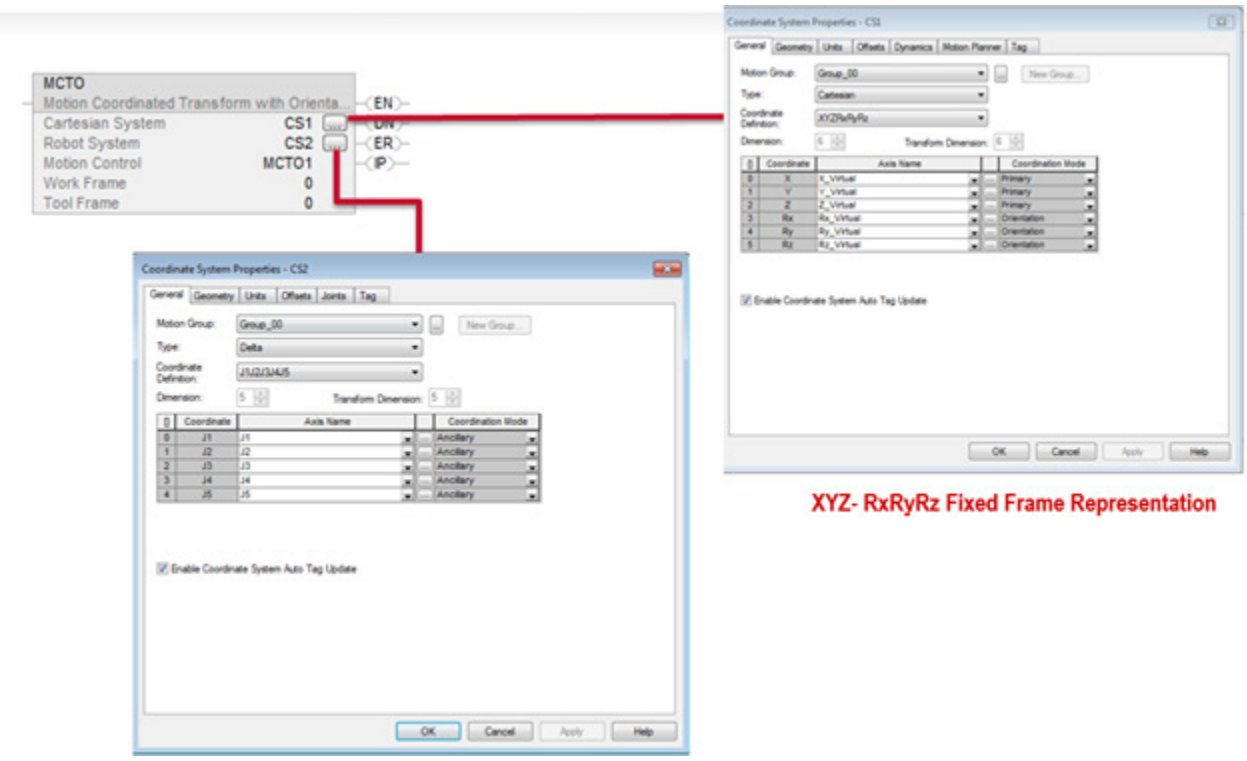
Once the robot reference frame is established, the robot can be moved to any desired position in joint space when transform is not enabled. When MCTO is initiated for the first time, a forward transform is executed first to set the corresponding Cartesian coordinate positions. Once the MCTO instruction is active, a bidirectional transform link is established, so that if the Cartesian coordinate is commanded to move to a target position in the Cartesian space along a linear path, the robot moves to the Cartesian target coordinates along a linear path. Similarly, if the Robot is commanded to move to a joint coordinate position, the Robot moves to the joint target position along a non-Cartesian path. While MCTO instruction is active, the system maintains the coordinate system related data for Cartesian and Robot coordinate systems.

This diagram illustrates the transformation from a Cartesian coordinate system(X,Y,Z,Rx,Ry,Rz) to a Delta robot geometry with 5 axes (J1J2J3J4J5). The Rx, Ry, and Rz represent orientations around X,Y,Z axes in a positive direction.



Configure an MCTO instruction

The following illustration shows how to configure an MCTO instruction, with the Cartesian coordinate system as the source and the Delta 5 axis geometry as the target. Configure the source and target coordinate systems in the coordinate system dialog box.



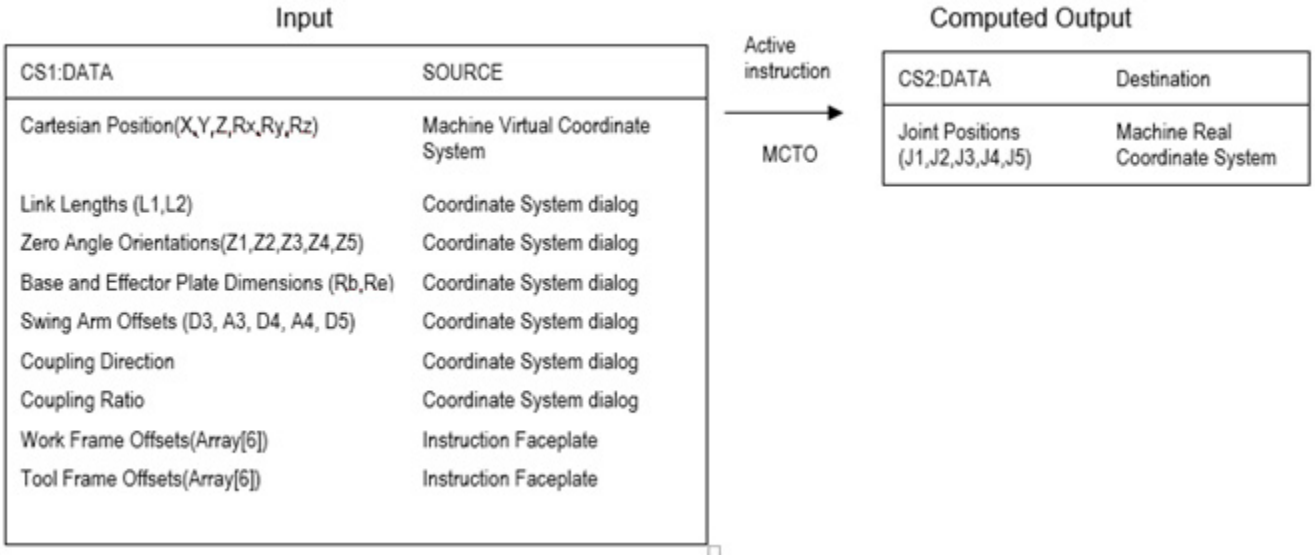
Data flow during forward transform when MCTO is active

A forward transform executes when a move executes on joint coordinates of a robot system with an enabled MCTO instruction. During forward transform, the current joint angle positions compute the corresponding Cartesian coordinate positions based on geometry.

Input Data			Computed Output	
CS2:DATA	SOURCE	Active instruction	CS1:DATA	Destination
Joint Positions (J1,J2,J3,J4,J5)	Machine Real Coordinate System	→ MCTO	Cartesian Positions (X,Y,Z,Rx,Ry,Rz)	Machine Virtual Coordinate System
Link Lengths (L1,L2)	Coordinate System dialog			
Zero Angle Orientations(Z1,Z2,Z3,Z4,Z5)	Coordinate System dialog			
Base and Effector Plate Dimensions (Rb,Re)	Coordinate System dialog			
Swing Arm Offsets (D3, A3, D4, A4, D5)	Coordinate System dialog			
Coupling Direction	Coordinate System dialog			
Coupling Ratio	Coordinate System dialog			
Work Frame Offsets(Array[6])	Instruction Faceplate			
Tool Frame Offsets(Array[6])	Instruction Faceplate			

Data flow during inverse transform when MCTO is active

When MCTO is active, during inverse transform, the instruction uses the current cartesian positions and geometry configuration to compute the corresponding joint angle positions. MCTO returns an Error 61 when moving both Cartesian and Joint coordinate systems simultaneously.



Fault codes

For the Motion Coordinated Transform with Orientation (MCTO) instruction, an error will occur at runtime if invalid operand values are provided.

Extended Error codes

Extended Error codes help to further define the error message given by the instruction. Their meaning is dependent upon the Error Code with which they are associated.

Error Code	EX_ERROR Code	Description
13	3	Value Out Of Range (base angle) Any orientation angle > 360 or < -360
13	3	Value Out Of Range (base ID) ID equals any negative value.
13	4	Value Out Of Range (tool angle) Any orientation angle > 360 or < -360
13	4	Value Out Of Range (tool ID) ID equals any negative value.
61	1	Connection Conflict: Transform Motion Group Error. Cartesian or Robot coordinate system is ungrouped, or associated to a different motion group from the other.

Error Code	EX_ERROR Code	Description
61	2	Connection Conflict: Transform Duplicate Systems Error. Operand 0 and Operand 1 are of the same coordinate system type.
61	3	Connection Conflict: Transform Source Dimension Error. Transform dimension of the Cartesian coordinate system is < 2.
61	4	Connection Conflict: Transform Target Dimension Error. Robot coordinate system Transform Dimension is equal to zero.
61	5	Attempting to use the CS as multiple sources - FAN-OUT error.
61	6	Attempting to use the CS as multiple targets - FAN-IN error.
61	9	Connection Conflict Transform Axes Overlap Error An axis is a member of both the Cartesian and Robot systems.
61	10	Axis or axes are in motion with the below exception: Gearing or camming motion is in progress on the Cartesian CS axes. The Cartesian CS slave axes cannot have a master axis from the Robot CS.
61	12	Connection Conflict Transform Invalid Link Length Link length values for any robot geometry must be > 0.0 units.
61	13	Axis is shut down
61	14	Axis is inhibited
61	15	Connection Conflict Transform Invalid Delta Configuration Link Length1 must not be equal to LinkLength2 End Effector Offset1 Re must not be negative For Delta J1J2J6 and Delta J1J2J3J6 End Effector Offset3(D3) must not be negative (Link length 1 + Rb - Re) must be less than link length 2 (Link length 1 + Rb – Re) must be positive or greater than zero
61	18	Connection Conflict Transform Invalid Articulated Configuration When the offset value is invalid for Articulated Independent J1J2J3J4J5J6 geometry, this error is generated. Base offset Yb must be equal to 0.0 End-Effector offset Ye must be equal to 0.0 End-Effector offset Ze must be equal to 0.0
67	1	Invalid Transform Position Invalid Rx Orientation Invalid Rx orientation value at EOA transformations. For 4 axis geometries only Rx = 180 deg position is allowed. For all other positions, it generates an error.

Error Code	EX_ERROR Code	Description
67	2	Invalid Transform Position Invalid Ry Orientation Invalid Ry orientation value at EOA transformations. For 4 axis geometries, after removing work frame and tool frame, there should not be any Ry orientation values at EOA
67	3	Invalid Transform Position Invalid Rz Orientation
147	3-5	Invalid Orientation Scaling Constant Extended error code 3 : Rx Extended error code 4 : Ry Extended error code 5 : Rz Orientation axis has scaling constant > MAX_K_CONSTANT_FOR_ORIENTATION_AXIS. or has scaling constant which is not Integer or has Conversion ratio between Coordination Units and Position Units other than 1:1.
148	3-5	MCPM Orientation Offset Not Rx orientation offset not valid. Extended error code 3 : Rx orientation offset not valid. Extended error code 4 : Ry orientation offset not valid. Extended error code 5 : Rz orientation offset not valid. If robot Geometry is (MO_CD_J1J2J6 or MO_CD_J1J2J3J6) and: • workframe offset Rx isn't 0 or • workframe offset Ry isn't 0 or • toolframe offset Rx isn't 0 or • toolframe offset Ry isn't 0 Or if robot Geometry is MO_CD_J1J2J3J4J5 and: • workframe offset Rx isn't 0 or • workframe offset Ry isn't 0 or • toolframe offset Rx isn't 0 or • toolframe offset Rz isn't 0
149	3-5	Orientation Axis Not Virtual Extended error code 3 : Rx axis must be virtual if transforms are enabled. Extended error code 4 : Ry axis must be virtual if transforms are enabled. Extended error code 5 : Rz axis must be virtual if transforms are enabled.
151	1	Joint Angle J1 Beyond Limits For details, refer to Geometry related Maximum Joint Limit Section.
151	2	Joint Angle J2 Beyond Limits For details, refer to Geometry related Maximum Joint Limit Section.

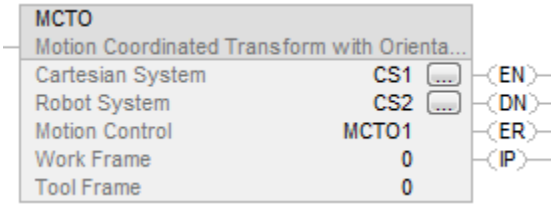
Error Code	EX_ERROR Code	Description
151	3	Joint Angle J3 Beyond Limits For details, refer to Geometry related Maximum Joint Limit Section.
151	4	Joint Angle Beyond Limits This indicates the error condition when the Joint 4 in a 5 axis Delta goes beyond turns counter range limit (45899.99 <J4<-45900) Ext Error 4 : Joint J4 Beyond Limit
151	5	Joint Angle Beyond Limits This indicates error condition when the Joint 5 in a 5 axis Delta goes beyond +/-179, +179 > J5 <-179 Ext Error 5 : Joint J5 Beyond Limit
151	6	Joint Angle Beyond Limits This indicates the error condition when the Joint 6 in a 4 axis Delta goes beyond turns counter range limit (45899.99<J6<-45900) Ext Error 6 : Joint J6 Beyond Limit
152	1	Maximum Orientation Speed Exceed for Rx When Orientation axis Rx is commanded to move by an angle greater than or equal to 180 degrees in one coarse update period, this error and extended error is returned.
152	2	Maximum Orientation Speed Exceed for Ry When Orientation axis Ry is commanded to move by an angle greater than or equal to 180 degrees in one coarse update period, this error and extended error is returned.
152	3	Maximum Orientation Speed Exceed for Rz When Orientation axis Rz is commanded to move by an angle greater than or equal to 180 degrees in one coarse update period, this error and extended error is returned.
153	1	Invalid Translation Position MOP Invalid X Translation Translation on X axis is Invalid
153	2	Invalid Translation Position MOP Invalid Y Translation Translation on Y axis is Invalid
153	3	Invalid Translation Position MOP Invalid Z Translation Translation on Z axis is Invalid
156	1	Singularity Condition Error Joint 1 axis is close to Arm Singularity condition in Articulated Independent J1J2J3J4J5J6 Geometry Ext Error 1: Arm Singularity Condition
156	2	Singularity Condition Error Joint 3 axis is close to Elbow Singularity condition in Articulated Independent J1J2J3J4J5J6 Geometry Ext Error 2: Elbow Singularity Condition

Error Code	EX_ERROR Code	Description
156	3	Singularity Condition Error Joint 5 axis is close to Wrist Singularity condition in Articulated Independent J1J2J3J4J5J6 Geometry Ext Error 1: Wrist Singularity Condition

Example

The following examples activate the transformations between Cartesian coordinate system CS1 and robot coordinate system CS2, which is a Delta 5 axis geometry. In the first example, neither work frame nor tool frame are specified, so Cartesian positions are computed at robot end-of-arm (EOA), with respect to the robot base frame.

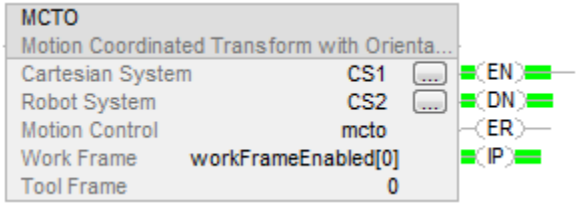
Ladder Diagram



When MCTO is active, with a move to Cartesian positions X = 7.361, Y = -4.25, Z = -928.18, Rx = 180, Ry = 30 and Rz = -30, the MCTO computes the corresponding Joint angle as shown in the following diagram.

J1.ActualPosition	Contro...	9.999875
J2.ActualPosition	Contro...	9.999875
J3.ActualPosition	Contro...	9.999875
J4.ActualPosition	Contro...	30.0
J5.ActualPosition	Contro...	-30.0

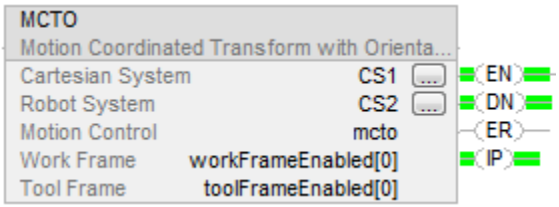
Consider the same MCTO, with work frame offsets enabled for X = 50, Y = 50 and Rz = 50. Now MCTO will compute the Cartesian positions with respect to the new work frame.



When a move is programmed to the same end position, the corresponding joint positions are as shown in this diagram.

J1.ActualPosition	Contro...	5.511625
J2.ActualPosition	Contro...	8.650625
J3.ActualPosition	Contro...	16.992374
J4.ActualPosition	Contro...	-20.0
J5.ActualPosition	Contro...	-30.0

Consider that the MCTO is programmed with tool frame offsets also enabled for Ry = 50 along with the work frame offsets. MCTO now computes the Cartesian positions with respect to the end of the new work frame and tool frame.



This diagram shows the corresponding Joint angle positions after programming a move to the same end position.

J1.ActualPosition	Contro...	8.266875
J2.ActualPosition	Contro...	13.584
J3.ActualPosition	Contro...	22.686874
J4.ActualPosition	Contro...	-20.0
J5.ActualPosition	Contro...	-80.0

For more information on configuring offsets, see Configure Coordinate System Offsets.

Structured Text

MCTO(CS1, CS2, MCTO1, 0, 0);

Tip: For further information on creating geometries with orientation support, see the Motion Coordinate System User Manual publication MOTION-UM002.

See also

Structured Text Syntax on page 665

Index Through Arrays on page 691

Motion Error Codes (.ERR) on page 581

Multi-Axis Coordinated Motion Instructions on page 359

Define coordinate system frames

Motion Coordinated Path Move (MCPM)

Use the MCPM instruction to start a multi-dimensional coordinated path move for the specified Primary axes (X, Y, Z) and orientation axes (Rx, Ry, Rz) of a Cartesian coordinate system. Use this instruction to program Cartesian moves on robots with orientation control.

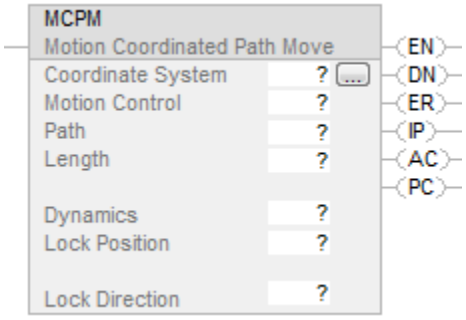
Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.
-

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MCPM (CoordinateSystem, MotionControl, Path, Length, Dynamics, LockPosition, LockDirection);

Operands

Important: Do not use the same tag name for more than one instruction in the same program. Do not write to any instruction output tag under any circumstances.

ATTENTION: If instruction operands are changed while in Run mode, the pending edits must be accepted and the controller mode cycled from Program to Run for the changes to take effect.

Configuration

This table provides the operands used to configure the instruction. These operands cannot be changed at runtime.

Operand	Data Type	Format	Description
Coordinate System	COORDINATE_SYSTEM	Tag	Cartesian coordinate system used to program the moves. Reference to Cartesian coordinate system used to program the moves. For controllers that support the REF_TO motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

For further information of configuring coordinate systems refer to the Motion Coordinate System User Manual publication MOTION-UM002.

Inputs

This table explains the instruction inputs. The inputs may be field device signals from input devices or derived from user logic.

Operand	Data Type	Format	Description
Path	PATH_DATA	Tag	See PATH_DATA Structure
Length	DINT	Immediate Tag	This input is immediate and indicates the length of the PATH_DATA input. **Tip:** Set the length to 1. Values greater than 1 are reserved for future use.
Dynamics	DYNAMICS_DATA	Tag	See DYNAMICS_DATA Structure
Lock Position	REAL	Tag	Position on the Master Axis where a Slave should start following the master after the move has been initiated on the slave axis coordinate system. **Tip:** Lock Position is only valid when the MCPM instruction is used in Master Driven Speed Control mode.
Lock Direction	UINT32	Tag	Specifies the conditions when the Lock should be activated. **Tip:** Lock Direction is only valid when the MCPM instruction is used in Master Driven Speed Control mode.

PATH_DATA Structure

Operand	Scroll, List, or Check Box	Data Type	Default	Notes
Interpolation Type	List of Point-to-Point (0) Continuous Path Linear (1)	DINT	0	Point-to-point move[2] Continuous Path Linear See Interpolation Type related topics below.

Operand	Scroll, List, or Check Box	Data Type	Default	Notes
Position [X, Y, Z, Rx, Ry, Rz]	List of constant or variable	REAL [9] 1	0	[X,Y.Z] in Coordination Units, [Rx, Ry, Rz] X-Y-Z fixed angle format in degrees Index 0: X 1: Y 2: Z 3: Rx 4: Ry 5: Rz 6: *2 7: *2 8: *2 See Position related topics in the following section.
Robot Configuration	List of Bit values : **Bit0** – Robot Configuration Change(1)/Same(0) **Bit1** – Lefty(1)/Righty(0) **Bit2** – Above(1)/Below(0) **Bit3** – Flip(1)/No flip(0)	DINT	0	**Bit 0 to 3** – Applies only to Articulated and SCARA geometries. Set to zero for Delta geometries. See Robot configuration.
Turns Counters	List of variables	INT 16 [4]	0	Index 0: J1 1: J4 2: J6 3 2 Joint axes turns counters. Each integer is a signed value (±127).
Move Type	List of Absolute (0), Incremental (1)	DINT	0	Select the move type. Refer to Motion Coordinated Linear Move (MCLM on page 456) for more information on this operand.
Termination Type	List of Actual Tolerance(0), No Settle(1) Command Tolerance(6)	DINT	0	Refer to Motion Coordinated Linear Move (MCLM) on page 456 for more information on this operand.
Command Tolerance Linear	List of constant or variable	REAL	0	Used for Cartesian Primary axes position only. Coordination Units Linear

DYNAMICS_DATA_STRUCTURE

Operand	Scroll, List, or Check Box	Data Type	Default	Notes
Units Mode	List of 0 = % of Maximum 1 = Coord Units (per)	DINT	0	See Units Mode section.
Time Units	List of 0 = Seconds 1 = Master Units	DINT	0	Not applicable if Percentage of Maximum is selected as unit mode. Applies to speed, acceleration, and deceleration only. See Time Units section.
Profile	List of 0= Trapezoidal 1= S-Curve	REAL	0	1 See Profile section.

Operand	Scroll, List, or Check Box	Data Type	Default	Notes
Speed	List of Constant or Value	REAL	0	[1] % of Maximum or Coordination Units/Time Units.
Acceleration	List of Constant or Value	REAL	0	[1] % of Maximum or Coordination Units/Time Units[2].
Deceleration	List of Constant or Value	REAL	0	[1] % of Maximum or Coordination Units/Time Units[2].
Acceleration Jerk	List of Constant or Value	REAL	0	% of Time accelerating For all axes always See Acceleration Jerk section below. Applies to acceleration and orientation acceleration.
Deceleration Jerk	List of Constant or Value	REAL	0	% of Time deceleration For all axes always. Applies to deceleration and orientation deceleration.
Orientation Speed	List of Constant or Value	REAL	0	% of Maximum of orientation speed in coordinate system configuration or Degrees/Time Unit.
Orientation Acceleration	List of Constant or Value	REAL	0	% of Maximum orientation acceleration in coordinate system configuration or Degrees/Time Unit2.
Orientation Deceleration	List of Constant or Value	REAL	0	% of Maximum orientation deceleration in coordinate system configuration or Degrees/Tim Unit2.

[1] The units mode specifies the units for dynamics parameters of the coordinated move and not the individual axes. The two selections give an option for the user to program directly in Coordination Units or % of the maximum configured in the coordinate system tag. All the primary axes are configured in different user units, such as mm, inches, and cm, while the orientation axes units are in degrees.

[2] Reserved for future use.

Path Data

Interpolation Type

- **Point to Point**: In this type of motion the end position of the Tool Center point (TCP) is designated but the path used to reach the end position is irrelevant. This is generally the quickest way to move the TCP to a destination position. This type of interpolation will be available only for the Articulated and SCARA type geometries in future releases.
- **Continuous Path Linear**: In this type of motion the TCP moves from a starting position to a commanded end point along a straight line.

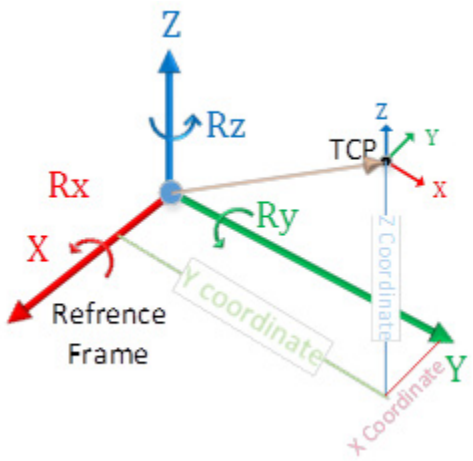
During the linear move the orientation of the TCP changes continuously from the start orientation position to the commanded orientation position.

Position

This is a one dimensional array, whose dimension is defined to be at least equivalent to the number of axes specified in the coordinate system, 6 in Cartesian coordinate system where X, Y, Z, Rx, Ry, Rz are values of the TCP with reference to a reference frame. The Position array defines either the new absolute or incremental position.

Below is an example of TCP point which has translation on X, Y, and Z axis followed with rotation on Z axis (Rz = 90°) resulting in TCP's X axis aligning with Y axis of reference frame.

See Configuring Cartesian XYZRxRyRz Coordinate System for more details on Cartesian coordinate system.



Robot configuration

A robot can reach an end destination position with different joint positions and, consequently different robot configurations within the confines of established joint limits. If the user wants the robot to move to a position with a specific configuration in continuous path (CP) mode, specify the end position along with the desired pose of the robot.

- RIGHTY versus LEFTY
- ABOVE versus BELOW
- FLIP versus NOFLIP

For more details refer to the Motion Coordinate System User Manual publication MOTION-UM002 for configuration sections for SCARA Independent J1J2J3J6 Robots, Articulated Dependent J1J2J3J6 Robots, and Articulated Independent J1J2J3J4J5J6 Robots.

Dynamics Data

Profile

- Trapezoidal (0): When the value is set to 0 the profile setting takes precedence over the acceleration Jerk and deceleration Jerk value (values are defaulted to 0%), as a result the velocity profile is always trapezoidal. Refer to the Profile information in Motion Coordinated Linear Move (MCLM) on page 456 for further details.
- S Curve (1): When the value is set to 1 the acceleration and deceleration jerk values are taken into account. The planner attempts to achieve the acceleration and deceleration value calculated from the dynamics settings. Refer to the Profile information in Motion Coordinated Linear Move (MCLM) on page 456 for further details.

Units Mode

- % of Maximum (0): When Units Mode is selected to be percentage of Maximum then the 6 dynamics parameters (as shown in 1st column in table below) are programmed to be percentage of those defined in dynamics tab of the coordinate system (as shown in 2nd column in table below).
  For Example, if Units Mode is set to percentage of Maximum, the Speed is set to 50, and the Vector. Maximum Speed is set to 100mm/sec: the maximum linear speed of the MCLM could be up to 50% of the Maximum speed, i.e. 50mm/sec. This option is available for only CP moves (Support for PTP in future).

Dynamics Data Parameter	Coordinate System Parameters Specification
Speed	Vector. Maximum Speed. Primary
Acceleration	Vector. Maximum Acceleration. Primary
Deceleration	Vector. Maximum Deceleration. Primary.
Orientation Speed	Vector Maximum Speed Orientation
Orientation Acceleration	Vector Maximum Acceleration Orientation
Orientation Deceleration	Vector Maximum Deceleration Orientation

- Coordination Units (1): The speed for 6 parameters (as shown in 1st Column in table above) are programming units of Coordination Units per Time Unit, as defined by the Time Units parameter.
  - Speed is programmed in Coordination Units per Second or Coordination Units per Master Unit.
  - Acceleration and Deceleration in Coordination Units per Seconds^2 or Coordination Units per second Master Units^2.
  - Orientation Speed is programmed in Degrees per Second or Degrees per Master Unit.

- Orientation Acceleration and Deceleration in Degrees per Seconds^2 or Degrees per Master Units^2

Time Units

- Seconds (0): Time units is in seconds
- Master Units (1): In a Master Slave configuration, the Speed/Acceleration/Deceleration of Master determines the final Primary and Orientation speed/Acceleration/Deceleration of the MCPM instruction.

For the example consider that a MDCC establishes a linkage between Master axis and slave Coordinate system. The Master axis is being moved by an MAM instruction and an MCPM instruction is executing: The resultant Speed/Acceleration/Deceleration value can be calculated as shown below.

- Formula for velocity conversion from [Units/Master Units] to [Units/Seconds]:
  V_S[SU/TU] = V_S[SU/MU] * VM[MU/TU]
  If the Slave is running at 3 Degrees/MU and master is running at 2 MU/sec then the slave speed is: 3*2 = 6 Degrees/Second.
- Formula for acceleration and deceleration conversion from [Units/Master Units2] to [Units/Seconds2]:
  aS[SU/TU2] = aS[SU/MU2] * VM^2 [MU/TU] + VS[SU/MU] * aM[MU/TU2]
  - If the Slave is accelerating at 5 Degrees/MU2 and master is running at constant speed 2 MU/sec (aM=0), the slave acceleration would be: 5*2*2 = 20 Degrees/Second2.
  - If the Slave is accelerating at 5 Degrees/MU2, its instantaneous speed is 3 Degrees/MU and the Master is running at speed 2 MU /sec and acceleration 4 MU /sec2 then the slave acceleration is: 5*2*2 + 3*4 = 32 Degrees/Second2.

Acceleration Jerk

This parameter specifies the percentage of Time, of the Acceleration that the MCPM path should use, to compute the acceleration jerk rate of the MCPM path move. This is always programmed in units of percentage of time.

See Jerk Unit topic below on how to convert percentage of Time to engineering units.

See Acceleration Jerk topic in MCLM document for more details

Deceleration Jerk

This parameter specifies the percentage of Time for the Deceleration that the MCPM path should use to compute the deceleration jerk rate of the MCPM path move. This is always programmed in units of percentage of time.

See Jerk Units topic below on how to convert percentage of Time to engineering units.

See Deceleration Jerk topic in MCLM document for more details.

Jerk Unit: percentage of time (% of time)

If you want to convert % of Time to engineering units, use these equations.

- For Acceleration jerk calculation in Engineering Units, use this calculation:

$$J_a\left[\frac{EU}{S^3}\right] = \frac{a_{max}^2\left[\frac{EU}{S^2}\right]}{V_{max}\left[\frac{EU}{S}\right]}\left(\frac{200}{J_a\ [\%\ of\ Time]} - 1\right)$$

- For deceleration jerk calculation in Engineering Units, use this calculation:

$$J_d\left[\frac{EU}{S^3}\right] = \frac{d_{max}^2\left[\frac{EU}{S^2}\right]}{V_{max}\left[\frac{EU}{S}\right]}\left(\frac{200}{J_d\ [\%\ of\ Time]} - 1\right)$$

Orientation Speed

Orientation speed is a vector speed, its value is applied as a single orientation vector composed of Rx Ry and Rz as orthogonal components. For example if the Orientation speed is 5 Deg/Sec.

$$\sqrt{V_{Rx}^2 + V_{Ry}^2 + V_{Rz}^2} = \sqrt{25} = 5 \text{ Degrees/Sec}$$

Orientation Acceleration and Deceleration

Orientation acceleration/deceleration is a vector value, its value is applied as a single orientation vector composed of Rx, Ry and Rz as orthogonal components. For example if the Orientation acceleration or deceleration is set to 5 Deg/Sec², then

$$\sqrt{A_{Rx}^2 + A_{Ry}^2 + A_{Rz}^2} = \sqrt{25} = 5 \text{ Deg/Sec}^2$$

Lock Direction

Specifies the direction that the master axis must be moving when it crosses the Lock Position for the lock to be activated.

Tip: When the Time Unit is set to Master units the value set in this field should be the direction of Master. Refer to the MCLM instruction for further information on the Lock Direction operand.

Lock Position

Specifies the master axis position where the slave will become locked to the master axis.

Outputs

N/A

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of the incorrect type is passed to the Coordinate System operands See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken

Condition/State	Action Taken
Prescan	Same as Rung-condition-in is false.
Rung-condition-in is false	The .EN, .DN, and .ER are cleared to false.
Rung-condition-in is true and .EN bit is false	The .EN bit is set to true and the instruction executes.
Rung-condition-in is true and .EN bit is true	N/A
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by Rung-condition-in is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See Motion Error Codes (.ERR) for motion instructions.

Extended Error Codes

Extended Error codes provide additional instruction-specific information for the Error Codes that are generic to other instructions. See Motion Error Codes (ERR) for Motion Instructions. Extended Error Codes meaning depends on the Error Codes they are associated with.

Error Code (.ERR)	Extended Error Code (.EXERR)	Description
7	0 thru 5	Shutdown State Error For a motion coordinated instruction, look at the extended error code (EXERR). It identifies which axis caused the error. Example: If EXERR is zero, check the axis for dimension zero.
11	0 thru 5	Axis Not Configured For single axis instructions: The Extended Error code for MAG, MDAC, MAPC, MAM, MAJ, MATC, and MCD is defined as: 1 = Slave axis 2 = Master Axis For the MAM, MCD, and MAJ instructions in time driven mode, the axis being moved is a slave axis. For multi-axes instructions: The Extended Error code for MCPM, MDCC, MCLM, MCCM, and MCCD is defined as: The axis number in the coordinate system where 0 = 1st axis 2 = Master Axis or 3rd Slave Axis
13	2, 3, 5	Parameter Out of Range An EXERR = 0 means the first operand of the instruction is outside its range.
16	0 thru 5	Homing is in process on an axis. Extended error code indicates which axis caused the error.
25	0 thru 5	You attempted to execute an instruction that is not correct. Extended error code indicates which axis caused the error **Tip:** MCPM returns this error if the axis is configured in torque mode.
53	0 thru 5	Axis is inhibited For single axis instructions, the Extended Error code for MAG, MDAC, MAPC, MAM, MAJ, MATC, and MCD is defined as: 1 = Slave axis 2 = Master Axis For the MAM, MCD, and MAJ instructions in time driven mode, the axis being moved is a slave axis. For multi-axes instructions, the Extended Error code for MCPM, MDCC, MCLM, MCCM, and MCCD is defined as: The axis number in the coordinate system where 0 = 1st axis 2 = Master Axis or 3rd Slave Axis

Error Code (.ERR)	Extended Error Code (.EXERR)	Description
65	0 thru 5	Axis Position overflow The range for position depends on the conversion constant of the axis. Maximum positive position = 2,147,483,647 / conversion constant of the axis. Maximum negative position = -2,147,483,648 / conversion constant of the axis. Select a conversion constant of 2,097,152 counts/inch. In this case: • Maximum positive position = 2,147,483,647 / 2,097,152 counts/inch = 1023 inches. • Maximum negative position = -2,147,483,648 / 2,097,152 counts/inch = -1023 inches. For a motion coordinated instruction, look at the extended error code (EXERR). It identifies which axis caused the error.
76	0 thru 5	Maximum deceleration jerk is set to zero You cannot start motion that uses an S-curve profile if the maximum deceleration jerk for the axis is zero. (EXERR). It identifies which axis caused the error.
138	0	MCPM Path Data Invalid Value MCPM Path Data Interpolation Type If the interpolation type is set to anything other than 0 or 1, the instruction will report this error.

138	1	MCPM Path Data has Invalid Value.
		The instruction will report error if either of the below conditions are true.
		• If the end position is either infinity or NAN
		• If kinematic transform is active
		  • If end position Rx is not between +/-180 deg for absolute moves
		  • If end position Rz is not between +/-180 deg for absolute moves
		  • If end position Ry is not between +/-90 deg for absolute moves
		• If kinematic transform is active AND robot Geometry is Delta J1J2J6 (Delta 3 axes) or Delta J1J2J3J6 (Delta 4 axes)
		  • If end position Rx is not 180° for absolute moves
		  • If end position Ry is not 0° for absolute moves
		• If kinematic transform is active AND robot Geometry is Delta J1J2J3J4J5 (Delta 5 axis)
		  • If end position Rx is not 0 or 180° for absolute moves.

138	2	MCPM Path Data Invalid Value in MCPM Path Data Robot Config If an invalid Robot Configuration is specified (any bits other than but 0,1,2,3 are set), the instruction will report this error.
138	3	MCPM Path Data Invalid Value in MCPM Path Data Turns Counter If turns counter value is not within +/-127, the instruction will report this error.

Error Code (.ERR)	Extended Error Code (.EXERR)	Description
138	4	MCPM Path Data Invalid Value in MCPM Path Data Move Type If the move type is not either absolute or incremental, the instruction will report this error.
138	5	MCPM Path Data Invalid Value in MCPM Path Data TT Type If any term types other than 0, 1 and 6 are specified, the instruction will report this error.
138	6	MCPM Path Data Invalid Value in MCPM Path Data CMD TOL If negative command tolerance is specified or INF or NAN is specified, the instruction will report this error.
139	0	MCPM Dyn Data Invalid Value in MCPM Dyn Data Units Mode If Units Mode is specified anything other than 0, 1, or 2, instruction will report this error.
139	1	MCPM Dyn Data Invalid Value in MCPM Dyn Data Time Units If Time Units is specified anything other than 0 or 1, the instruction will report this error.
139	2	MCPM Dyn Data Invalid Value in MCPM Dyn Data Profile If Profile is specified anything other than 0(trap) or 1(s-curve), the instruction will report this error.
139	3	MCPM Dyn Data Invalid Value in MCPM Dyn Data Speed If the speed is negative or INF or NAN, the instruction will report this error.
139	4	MCPM Dyn Data Invalid Value in MCPM Dyn Data Accel If the accel is zero or negative or INF or NAN, the instruction will report this error.
139	5	MCPM Dyn Data Invalid Value in MCPM Dyn Data Decel If the decel is zero or negative or INF or NAN, the instruction will report this error.
139	6	MCPM Dyn Data Invalid Value in MCPM Dyn Data Accel Jerk If Jerk Units are %time and if the accel jerk is INF, NAN, less than or equal to zero or value greater than 100%, then the instruction will error. **Tip:** These errors apply only for s-curve.
139	7	MCPM Dyn Data Invalid Value MCPM Dyn Data Decel jerk If Jerk Units are %time, and if the decel jerk is NAN and less than or equal to zero, then the instruction will report error. **Tip:** This error only applies for s-curve.
139	8	MCPM Dyn Data Invalid Value MCPM Dyn Data Orientation Speed If the orientation speed is negative or INF or NAN, the instruction will report this error.
139	9	MCPM Dyn Data Invalid Value MCPM Dyn Data Orientation Accel If the orientation accel is zero or negative or INF or NAN, the instruction will report this error.
139	10	MCPM Dyn Data Invalid Value MCPM Dyn Data Orientation Decel If the orientation decel is zero or negative or INF or NAN, the instruction will report this error.

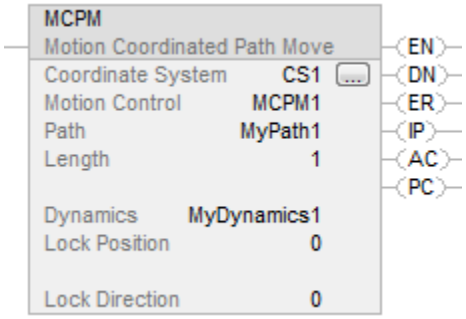
Error Code (.ERR)	Extended Error Code (.EXERR)	Description
155	none	MCPM Robot Geometry Not Supported If the user tries to run the MCPM instruction on an Articulated Independent J1J2J3J4J5J6 robot system, the instruction reports this error.
157	none	MCPM Joint Direction Sense Not Supported If the user tries to set the joint direction sense and run the MCPM instruction, the instruction reports this error.

Diagnostic Codes and Corrective Actions

N/A

Example

Ladder Diagram



Structured Text

MCPM (CS1, MCPM1, MyPath1, 1, MyDynamics1, 0, 0);

Tip: For further information on creating geometries with orientation support, see the Motion Coordinate System User Manual publication MOTION-UM002.

See also

Structured Text Syntax on page 665

Index Through Arrays on page 691

Motion Error Codes (.ERR) on page 581

Motion Coordinated Transform with Orientation (MCTO) on page 381

Define coordinate system frames

Motion Coordinated Change Dynamics (MCCD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The Motion Coordinated Change Dynamics (MCCD) instruction starts a change in the path dynamics of the specified coordinate system. Based upon the Motion Type, the MCCD changes the coordinated motion profile that is currently active on the system.

IMPORTANT Tags used for the motion control attribute of instructions should only be used once. Re-use of the motion control tag in other instructions can cause unintended operation. This may result in damage to equipment or personal injury.

IMPORTANT Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is by changing move dynamics (MCD or MCCD) or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-curve velocity profile can overshoot if:

maximum deceleration is decreased while the move is decelerating or close to the deceleration point; or

maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time

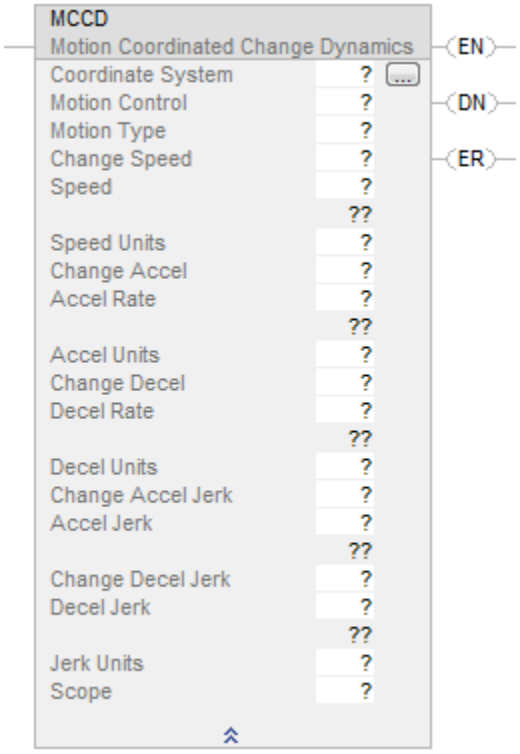
Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MCCD(CoordinateSystem, MotionControl, MotionType, ChangeSpeed, Speed, SpeedUnits, ChangeAccel, AccelRate, AccelUnits, ChangeDecel, DecelRate, DecelUnits, ChangeAccelJerk, AccelJerk, ChangeDecelJerk, DecelJerk, JerkUnits, Scope);

Operands

There are data conversion rules for mixed data types within an instruction. See *Data Conversion*.

Inputs

This table explains the instruction inputs.

Operand	Type	Format	Description
Coordinate System	COORDINATE_SYSTEM	tag	Coordinated group of axes. For controllers that support the REF_TO motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.
Motion Control	MOTION_INSTRUCTION	tag	Structure used to access instruction status parameters.
Motion Type	DINT	immediate	1 = Coordinated Move
Change Speed	DINT	immediate	0 = No 1 =Yes
Speed	SINT, INT, DINT, or REAL	immediate tag	[coordinate units]
Speed Units	DINT	immediate	0 = Units per Sec 1 = % of Maximum 4 = Units per MasterUnit
Change Accel	DINT	immediate	0 = No 1 = Yes
Accel Rate	SINT, INT, DINT, or REAL	immediate tag	[coordinate units]
Accel Units	DINT	immediate	0 = Units per Sec^2 1 = % of Maximum 4 = Units per MasterUnit^2
Change Decel	DINT	immediate	0 = No 1 =Yes
Decel Rate	SINT, INT, DINT, or REAL	immediate tag	[coordinate units]
Decel Units	DINT	immediate	0 = Units per Sec^2 1 = % of Maximum 4 = Units per MasterUnit^2
Change Accel Jerk	DINT	immediate	0 = No 1 = Yes
Accel Jerk	SINT, INT, DINT, or REAL	immediate tag	You must always enter a value for the Accel Jerk operand. This instruction only uses the value if the Profile is configured as S-curve. Accel Jerk is the acceleration jerk rate for the coordinate system. Use these values to get started. Accel Jerk = 100 (% of Time) Jerk Units = 2
Change Decel Jerk	DINT	immediate	0 = No 1 = Yes

Decel Jerk	SINT, INT, DINT, or REAL	immediate tag	You must always enter a value for the Decel Jerk operand. This instruction only uses the value if the Profile is configured as S-curve. Decel Jerk is the deceleration jerk rate for the coordinate system. Use these values to get started. Decel Jerk = 100 (% of Time) Jerk Units = 2
Jerk Units	DINT	immediate	0 = Units per sec^3 1 = % of Maximum 2 = % of Time (use this value to get started) 4 = Units per MasterUnit^3 6 = % of Time Master Driven
Scope	DINT	immediate	0 = Active Motion 1 = Active and Pending Motion

Structured Text

When entering enumerations for the operand value in structured text, multiple word enumerations must be entered without spaces. For example: when entering Decel Units the value should be entered as unitspersec^2 rather than Units per Sec^2 as displayed in the ladder logic.

See *Structured Text Syntax* for more information on the syntax of expressions within structured text.

For the operands that have enumerated values, enter your selection.

This Operand	**Has These Options Which You**	
	Enter as Text	**Or as**
Coordinate System	No enumeration	tag
Motion Control	No enumeration	tag
Motion Type	coordinatedmove	1 = Coordinated Move
Change Speed	no yes	0 = No 1 = Yes
Speed	No enumeration	immediate or tag
Speed Units	unitspersec %ofmaximum unitspermasterunit	0 1 4
Change Accel	no yes	0 = No 1 = Yes
Accel Rate	No enumeration	immediate or tag
Accel Units	unitspersec2 %ofmaximum unitspermasterunit2	0 1 4
Change Decel	no yes	0 = No 1 = Yes
Decel Rate	No enumeration	immediate or tag

Decel Units	unitspersec2 %ofmaximum unitspermasterunit2	0 1 4
Change Accel Jerk	no yes	0 = No 1 = Yes
Accel Jerk	No enumeration	immediate or tag You must always enter a value for the Accel operand. This instruction only uses the value if the Profile is configured as S-curve. Use this value to get started. Accel Jerk = 100 (% of Time)
Change Decel Jerk	no yes	0 = No 1 = Yes
Decel Jerk	No enumeration	immediate or tag You must always enter a value Decel Jerk operand. This instruction only uses the value if the Profile is configured as S-curve. Use this value to get started. Decel Jerk = 100 (% of Time) Jerk Units = 2
Jerk Units	unitspersec3 %ofmaximum %oftime unitspermasterunit3 %oftimemasterdriven	0 1 2 (use this value to get started) 3 6
Scope	activemotion activeandpendingmotion	0 = Active Motion 1 = Active and Pending Motion

Outputs

The following table explains the instruction outputs.

Mnemonic	Description
.EN (Enable) Bit 31	The Enable bit is set when the rung transitions from false to true. It resets when the rung transitions from true to false.
.DN (Done) Bit 29	The Done bit resets when the rung transitions from false to true. It sets when target position is calculated successfully.
.ER (Error) Bit 28	The Error bit resets when the rung transitions from false to true. It sets when target position fails to calculate successfully.

Description

MCCD is a transitional instruction:

- In relay ladder, toggle the Rung-condition-in from false to true each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition.

Coordinate System

The Coordinate System operand specifies the set of motion axes that define the dimensions of a coordinate system. The coordinate system supports up to three (3) primary axes.

Motion Type

The motion type operand determines which motion profile to change. Currently Coordinated Move is the only available option.

Coordinated Move - when selected, the Coordinated Move option changes the motion of the currently active move in the coordinate system.

Change Speed

The Change Speed operand determines whether or not to change the speed of the coordinated motion profile.

- No - no change is made to the speed of the coordinated motion.
- Yes - the speed of the coordinated motion is changed by the value defined in the Speed and Speed Units operands.

Speed

The Speed operand defines the maximum speed along the path of the coordinated move.

Speed Units

The Speed Units operand defines the units applied to the Speed operand either directly in coordinate units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system.

Change Accel

The Change Accel operand determines whether or not to change the acceleration of the coordinated motion profile.

- No - no change is made to the acceleration of the coordinated motion.
- Yes - the acceleration of the coordinated motion is changed by the value defined in the Accel Rate and Accel Units operands.

Accel Rate

The Accel Rate operand defines the maximum acceleration along the path of the coordinated move.

Accel Units

The Accel Units operand defines the units applied to the Accel Rate operand either directly in coordinate units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system.

Change Decel

The Change Decel operand determines whether or not to change the deceleration of the coordinated motion profile.

- No - no change is made to the deceleration of the coordinated motion.
- Yes - the deceleration of the coordinated motion is changed by the value defined in the Decel Rate and Decel Units operands.

Decel Rate

The Decel Rate operand defines the maximum deceleration along the path of the coordinated move.

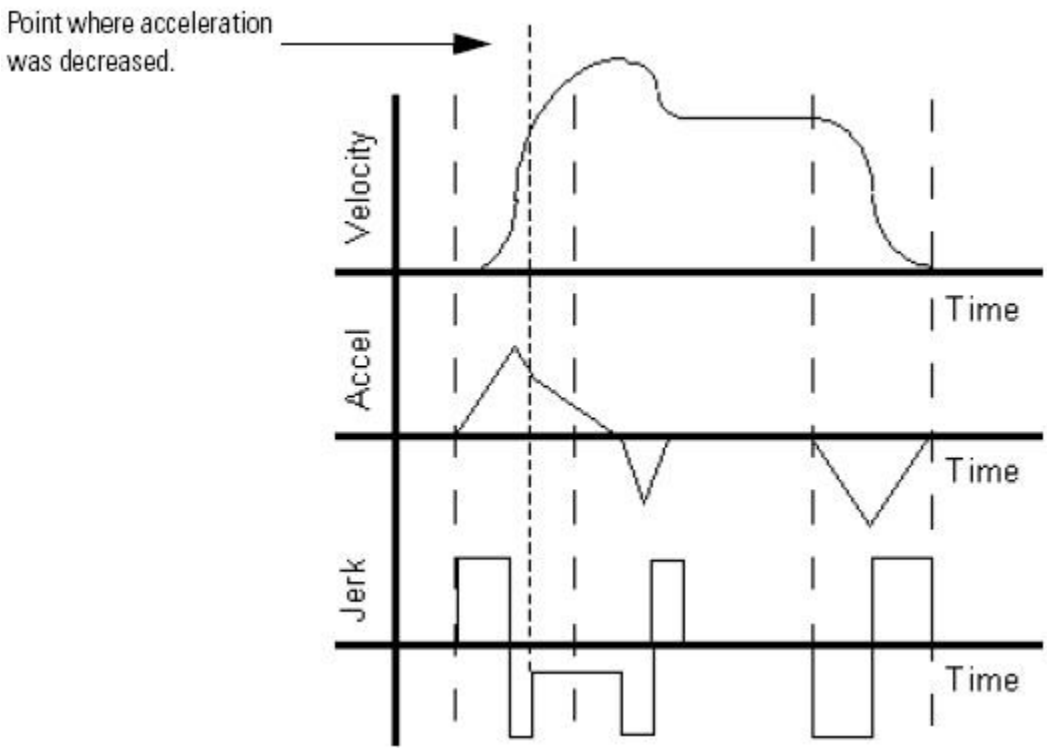
Decel Units

The Decel Units operand defines the units applied to the Decel Rate operand either directly in coordinate units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system.

Impact of Changes to Acceleration and Deceleration Values on Motion Profile

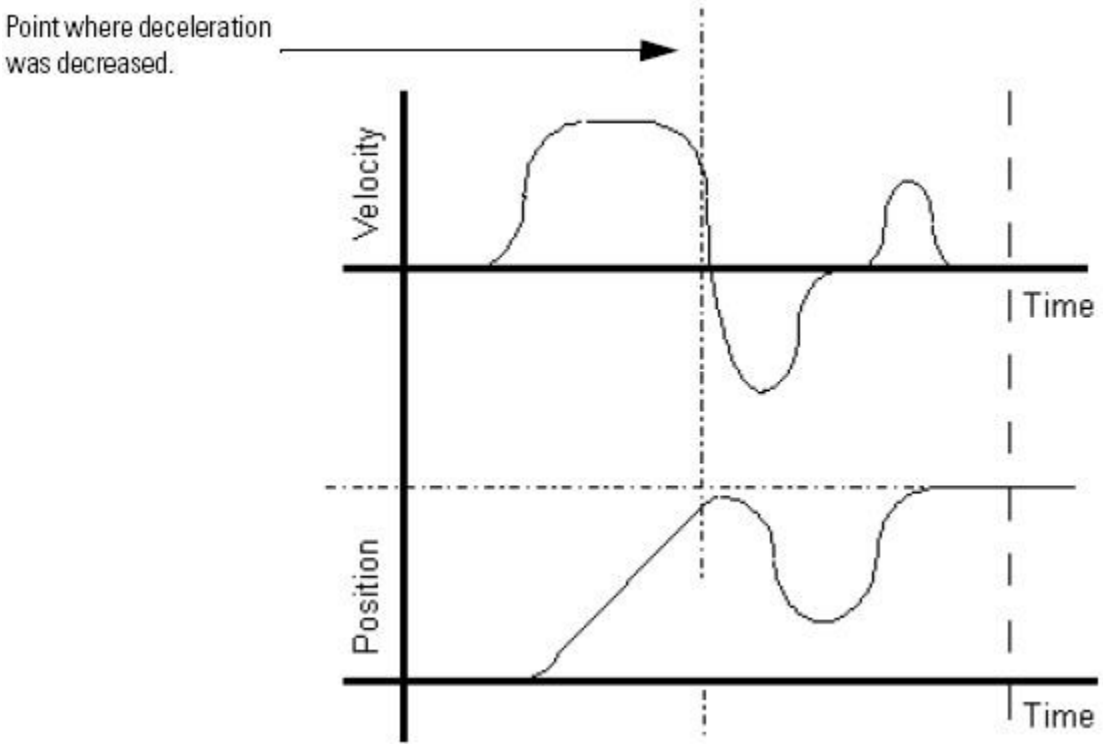
The following graph illustrates what could happen when a MCCD instruction is used to reduce the acceleration as velocity approaches maximum. The new acceleration Jerk Rate becomes smaller, further limiting the maximum change in acceleration. Velocity overshoot occurs due to the additional time required for acceleration to reach zero. Another profile is generated to bring velocity back to the programmed maximum.

Effect of Change to Acceleration



This graph illustrates what could happen when an MCCD instruction is used to reduce the deceleration as velocity and position approach their target endpoints. The new deceleration Jerk Rate becomes smaller. The time required to decelerate to zero causes velocity to undershoot, passing through zero and becoming negative. Axis motion also reverses direction until velocity returns to zero. An additional profile is generated to bring position back to the programmed target.

Effect of Change to Deceleration



Change Accel Jerk

The Change Accel Jerk operand determines whether or not to change the acceleration jerk of the coordinated motion profile.

- No - no change is made to the acceleration jerk of the coordinated motion.
- Yes - the acceleration of the coordinated motion is changed by the value defined in the Accel Jerk Rate and Jerk Units operands.

Accel Jerk

Accel Jerk defines the maximum acceleration jerk for the programmed move. For more information on calculating Accel Jerk, refer to the Jerk Units below.

Change Decel Jerk

The Change Decel Jerk operand determines whether or not to change the deceleration jerk of the coordinated motion profile.

- No - no change is made to the deceleration jerk of the coordinated motion.

- Yes - the deceleration of the coordinated motion is changed by the value defined in the Decel Jerk Rate and Jerk Units operands

Decel Jerk

Decel Jerk defines the maximum deceleration jerk for the programmed move. For more information on calculating Decel Jerk, refer to the Jerk Units.

Jerk Units

The jerk units define the units that are applied to the values entered in the Accel Jerk and Decel Jerk operands. The values are entered directly in the position units of the specified coordinate system or as a percentage. When configured using % of Maximum, the jerk is applied as a percentage of the Maximum Acceleration Jerk and Maximum Deceleration Jerk operands specified in the coordinate system attributes. When configured using % of Time, the value is a percentage based on the Speed, Accel Rate, and Decel Rate specified in the instruction.

If you want to convert engineering units to % of Time, use these equations.

For Accel Jerk:

$$j_a\,[\mathrm{EU/s^3}] = \frac{a_{max}{}^2\,[\mathrm{EU/s^2}]}{v_{max}\,[\mathrm{EU/s}]}\left(\frac{200}{j_a\,[\%\ \mathrm{of\ time}]} - 1\right)$$

For Decel Jerk:

$$j_a\,[\mathrm{EU/s^3}] = \frac{a_{max}{}^2\,[\mathrm{EU/s^2}]}{v_{max}\,[\mathrm{EU/s}]}\left(\frac{200}{j_a\,[\%\ \mathrm{of\ time}]} - 1\right)$$

If you want to convert % of Time to engineering units, use these equations.

For Accel Jerk:

$$j_a\,[\%\ \mathrm{of\ time}] = \frac{2}{1 + \dfrac{j_a\,[\mathrm{EU/s^3}]\ v_{max}\,[\mathrm{EU/s}]}{a_{max}^2\,[\mathrm{EU/s^2}]}}\,100$$

For Decel Jerk:

$$j_d\,[\%\ \mathrm{of\ time}] = \frac{2}{1 + \dfrac{j_d\,[\mathrm{EU/s^3}]\ v_{max}\,[\mathrm{EU/s}]}{d_{max}^2\,[\mathrm{EU/s^2}]}}\,100$$

Scope

Choosing Active Motion for the Scope operand specifies that the changes affect only the motion dynamics of the active coordinated motion instruction. Choosing Active and Pending Motion specifies that the changes affect the motion dynamics of the active coordinated motion instruction and any pending coordinated motion instruction in the queue. Currently the queue size is limited to one instruction after the active instruction.

Changing Between MDSC and Time Driven Modes in Master Driven Speed Control (MDSC)

You cannot change from Master Driven Mode to Time Driven Mode or vice versa with an MCCD instruction. You receive a runtime error if you attempt to change modes.

Fault Codes

For the Master Driven Speed Control (MDSC) function, an error will occur at runtime if you attempt to change the mode of the system from Master Driven to Time Driven or from Time Driven to Master Driven.

Extended Error Codes

Extended Error codes help to further define the error message given for this particular instruction. Their behavior is dependent upon the Error Code with which they are associated.

The Extended Error Codes for Servo Off State (5), Shutdown State (7), Axis Type Not Servo (8), Axis Not Configured (11), Homing In Process Error (16), and Illegal Axis Data type (38) errors all function in the same fashion. A number between 0...n is displayed for the Extended Error Code. This number is the index to the Coordinate System indicating the axis that is in the error condition. See Motion Error Codes (ERR) for Motion Instructions.

For the MCCD instruction, Error Code 13 - Parameter Out of Range, Extended Errors return a number that indicates the offending parameter as listed on the faceplate in numerical order from top to bottom beginning with zero. For example, 2 indicates the parameter value for Move Type is in error.

Referenced Error Code and Number	Extended Error Numeric Indicator	Instruction Parameter	Description
Parameter Out Of Range (13)	2	Move Type	Move Type is either less than 0 or greater than 1.
Parameter Out Of Range (13)	4	Speed	Speed is less than 0.
Parameter Out of Range (13)	7	Accel Rate	Accel Rate is less than or equal to 0.

Parameter Out of Range (13)	10	Decal Rate	Decel Rate is less than or equal to 0.

For the Error Code 54 – Maximum Deceleration Value is Zero, if the Extended Error returns a positive number (0-n) it's referring to the offending axis in the coordinate system. Go to the Coordinate System Properties General Tab and look under the Brackets ([ ])column of the Axis Grid to determine which axis has a Maximum Deceleration value of 0. Click on the ellipsis button next to the offending axis to access the Axis Properties screen. Go to the Dynamics tab and make the appropriate change to the Maximum Deceleration Value. If the Extended Error number is -1, this means the Coordinate System has a Maximum Deceleration Value of 0. Go to the Coordinate System Properties Dynamics Tab to correct the Maximum Deceleration value.

MCCD Changes to Status Bits:

For the Master Driven Speed Control (MDSC) function, when the MCCD is executed (goes IP), the CalculatedDataAvailable (CDA) status bit is cleared (as specified by the Scope variable of the MCCD instruction), in each MCLM and MCCM instruction tag, which indicates that the Event Distances has been computed. (The Scope variable specifies either the Active Motion instruction or Active Motion and Pending instruction (that is, all instructions, in the queue)).

After the MCCD is complete and the Event Distances have been recomputed, the CalculatedDataAvailable status bit is set again. Therefore, look at the CalculatedDataAvailable status bit after the MCCD instruction has been completed to determine when to use the recomputed Event Distances.

If a MCCD is executed (goes IP), the CDA bit is cleared. The Calculated Data for the move is recomputed using the new dynamics parameters. The CDA bit is set again when computations are complete. The Calculated Data that is recomputed is measured from the original Motion Start Point (MSP) to the Event Distance point using the new dynamics parameters as changed by the MCCD instruction - not from the point of the MCCD.

Note that if the MCCD changes the speed to 0, the Event Distance is not recomputed; the CDA bit is not set. The Event Distance is however recomputed if a second MCCD is issued to restart the motion. The recomputed Calculated Data includes the duration of the stopped motion.

If the Event Distance is set to 0, the Calculated Data is set to equal the position that equals the length of the move. This may be one or two coarse update periods before the PC bit is set because of an internal delay. The end position is typically achieved in the middle of a coarse update period, which adds up to one additional coarse update period to the delay. Therefore, if the master is moved a distance equal to the Calculated Data, you must wait up to 2 iterations more for the PC bit of the slave move to be set.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of the incorrect type is passed to the Coordinate System operands. See *Index Through Arrays* for array-indexing faults.

Execution

Ladder Diagram

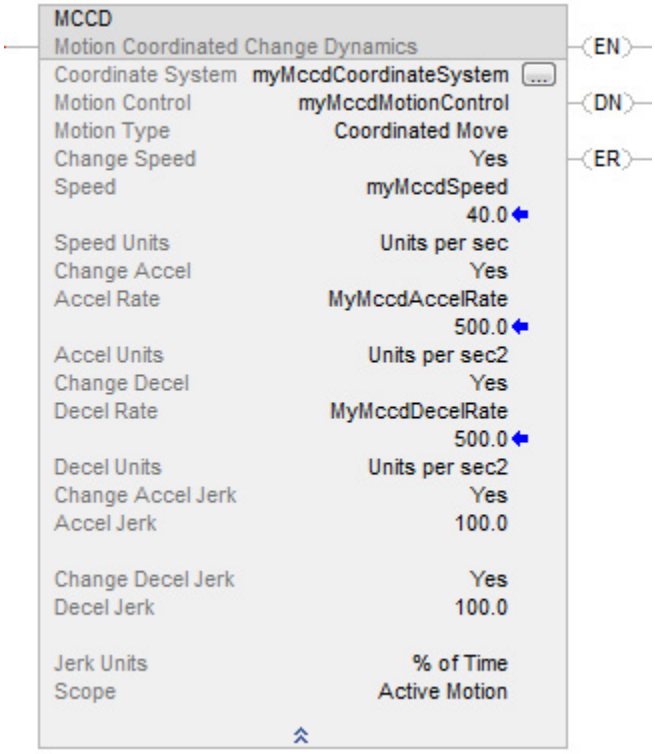
Condition/State	Action Taken
Prescan	The .EN, .DN, and .ER bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false followed by Rung-condition-in is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Example

Ladder Diagram



Structured Text

MCCD(myMccdCoordinateSystem, myMccdMotionControl, CoordinatedMove, Yes, MyMccdSpeed, Unitspersec, Yes, MyMsccdAccelRate, Unitspersec2, Yes, MyMccdDecelRate, Unitspersec2, Yes, 100.0, Yes, 100.0, %ofTime, ActiveMotion);

Motion Calculate Transform Position with Orientation (MCTPO)

This information applies to the Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Use the MCTPO instruction to:

- Compute the Cartesian positions, given the joint positions, when the Transform Direction is configured to Forward.
- Compute the Joint positions, given the Cartesian positions, when the Transform Direction is configured to Inverse.

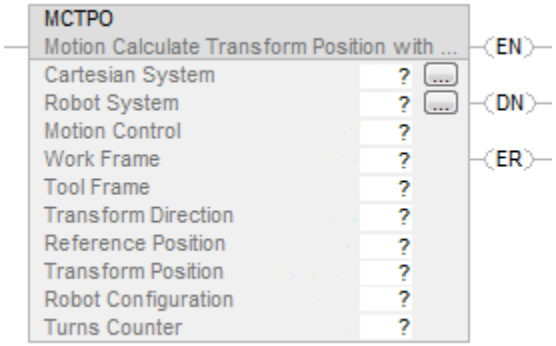
This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	No	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MCTPO(CartesianSystem, RobotSystem, MotionControl, WorkFrame, ToolFrame, Direction, ReferencePosition, TransformPosition, RobotConfiguration, TurnsCounter);

Operands

IMPORTANT Do not use the same tag name for more than one instruction in the same program. Do not write to any instruction output tag under any circumstances.

IMPORTANT If instruction operands are changed while in Run mode, the pending edits must be accepted and the controller mode cycled from Program to Run for the changes to take effect.

Configuration

This table provides the operands used to configure the instruction. These operands cannot be changed at runtime.

Operand	Data Type	Format	Description
Cartesian System	COORDINATE_SYSTEM	Tag	Cartesian coordinate system used to program the moves. For controllers that support the REF_TO motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.
Robot System	COORDINATE_SYSTEM	Tag	Non-Cartesian coordinate system that controls the actual equipment. Reference to Non-Cartesian coordinate system that controls the actual equipment. For controllers that support the REF_TO motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.
Motion Control	MOTION_INSTRUCTION	Tag	The control tag for the instruction
Work Frame	POSITION_DATA	Immediate Tag	Work Frame offsets are the offsets used to locate the user Work frame of the Robot relative to the origin of the Robot base frame. These offsets consist of an XYZ and RxRyRz value. This allows the programs to be written in the user work space or world frame and transformed to the Robot base frame. To rotate the target position around the X, Y, or Z axis or offset the target position along the X, Y, or Z axis of the Robot base coordinate system, enter the degrees of rotation into the Rx, Ry, and Rz tag members in units of degrees of rotation. Enter the offset distances into X, Y, and Z tag members in coordination units. Set the ID member to a value greater than or equal to zero. To maintain the work frame at the robot base frame, leave structured values at zero, or set the operand tag value to zero.

Operand	Data Type	Format	Description
Tool Frame	POSITION_DATA	Immediate Tag	Tool Center Point (TCP) offsets are the offsets used to locate the tool center relative to the center of the End of Arm. These offsets consist of an XYZ and RxRyRz value. To have the target position account for an attached tool having translation and/or orientation offsets, enter the tool offset distances into X, Y, and Z tag members in coordination units. Enter the degrees of tool rotation into the Rx, Ry, and Rz tag members in units of degrees of rotation. Set the ID member to a value greater than or equal to zero. To have the target position reflect only the point at the end-of-arm, leave the structure values at zero, or set the operand tag value to zero.

For further information of configuring coordinate systems refer to the Motion Coordinate System User Manual, publication MOTION-UM002.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of the incorrect type is passed to the Coordinate System operands. See Index Through Arrays for array-indexing faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	Same as Rung-condition-in is false.
Rung-condition-in is false	The .EN, .DN, .ER are cleared to false.
Rung-condition-in is true and .EN bit is false	The .EN bit is set to true and the instruction executes.
Rung-condition-in is true and .EN bit is true	N/A
Postscan	Same as Rung-condition-in is false.

Structured Text

Condition/State	Action Taken

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by Rung-condition-in is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Inputs

Operand	Data Type	Format	Description
Transform Direction	DINT	Tag	To calculate Cartesian position, select Forward(0). To calculate Robot joint positions, select Inverse(1).
Reference Position	REAL[6]	Tag	If the transform direction is forward, enter an array that has joint angles. If the transform direction is inverse, enter an array that has Cartesian positions.
Robot Configuration	DINT	Tag	Bit 0 of the Robot Configuration is a don't care, and is ignored for TD = Inverse and TD = Forward. This input operand is valid only for the Inverse Transform Direction. List of bit values: **Bit0** – Robot Configuration Change(1)/Same(0) **Bit1** – Lefty(1)/Righty(0) **Bit2** – Above(1)/Below(0) **Bit3** – Flip(1)/No flip(0) Robot Configuration must be zero (0) for Delta robot geometries.
Robot Turns Counters	INT16[4]	Tag	A counter indicating the number of times the 180 degree point is crossed in the +/- direction that the respective axis has turned. If Transform Direction is Forward, the value is computed by the system and returned via this tag. If Transform Direction is Inverse, this value determines final joint positions. The following are array index assignment for the joint axes turns count: Index 0 - J1 axis Index 1 - J4 axis Index 3 - J6 axis Index 4 - Reserved

Outputs

Operand	Data Type	Format	Description
Transform Position	REAL[6]	Tag	Array that stores the calculated position. If the transform direction is: • Inverse then an array has the Joint Angles. • Forward then an array has the Cartesian Positions.

Operand	Data Type	Format	Description
Robot Turns Counters	INT16[4]	Tag	If Transform Direction is Forward, this value is computed by the system and returned via this tag.
Robot Configuration	DINT	Tag	This output operand is valid only for the Forward Transform Direction. In Forward Transform, the value is computed by the system and returned through this tag. For Delta Robot geometries, robot configuration output is always zero.

IMPORTANT For 4-Axis Articulated Dependent Robot geometries, if you enter a cartesian position that maps to the Joint value of -179.00 for J2, the MCTPO inverse transform returns the error JOINT_ANGLE_EXCEEDED due to a rounding error in Kinematics Calculations.
For J3 limits, refer to the user documentation supplied by your robot manufacturer.

Fault Codes

An error occurs at runtime if invalid operand values are provided.

Extended Error Codes

Extended Error codes provide additional instruction-specific information for the Error Codes that are generic to other instructions. See Motion Error Codes (ERR) for Motion Instructions. Extended Error Codes meaning depends on the corresponding Error Codes.

Error Code	EX_ERROR Code	Description
13	3	Value Out Of Range (base angle) Any orientation angle > 360 or < -360.
13	3	Value Out Of Range (base ID) ID equals any negative value.
13	4	Value Out Of Range (tool angle) Any orientation angle > 360 or < -360.
13	4	Value Out Of Range (tool ID) ID equals any negative value.
13	5	Parameter Out Of Range Transform Direction.
13	6	Parameter Out Of Range Reference pos array end < 6
13	6	Parameter Out Of Range Input position Rx, Ry, or Rz exists and value >± 180 degrees

Error Code	EX_ERROR Code	Description
13	6	Parameter Out Of Range Operand 6 If robot geometry is Delta J1J2J6 or Delta J1J2J3J6 and if input position Rx exists and is not equal to 180°. If robot geometry is Delta J1J2J6 or Delta J1J2J3J6 and if input position Ry is not equal to 0°. If robot geometry is Delta J1J2J3J4J5 and if input position Rx exists and value is not equal to 0° or 180°.
13	7	Parameter Out Of Range Transform pos array end < 6
13	9	Parameter Out Of Range Any turns counter provided for the inverse transform exceeds 127 or -127 turns.
61	1	Connection Conflict Transform Motion Group Error Cartesian or Robot coordinate system is ungrouped or associated to a different motion group from the other.
61	2	Connection Conflict Transform Duplicate Systems Error Operand 0 and Operand 1 are the same coordinate system type
61	3	Connection Conflict Transform Source Dimension Error Transform dimension of the Cartesian coordinate system is < 2.
61	4	Connection Conflict Transform Target Dimension Error Robot coordinate system Transform Dimension is equal to zero.
61	9	Connection Conflict Transform Axes Overlap Error An axis is a member of both the Cartesian and Robot systems.
61	12	Connection Conflict Transform Invalid Link Length Link length values for any robot geometry must be > 0.0 units.
61	15	Connection Conflict Transform Invalid Delta Configuration Link Length1 must not be equal to LinkLength2 End Effector Offset1 Re must not be negative For Delta J1J2J6 and Delta J1J2J3J6 End Effector Offset3(D3) must not be negative (Link length 1 + Rb - Re) must be less than link length 2 (Link length 1 + Rb – Re) must be positive or greater than zero
61	18	Connection Conflict Transform Invalid Articulated Configuration When the offset value is invalid for Articulated Independent J1J2J3J4J5J6 geometry, this error is generated. Base offset Yb must be equal to 0.0 End-Effector offset Ye must be equal to 0.0 End-Effector offset Ze must be equal to 0.0

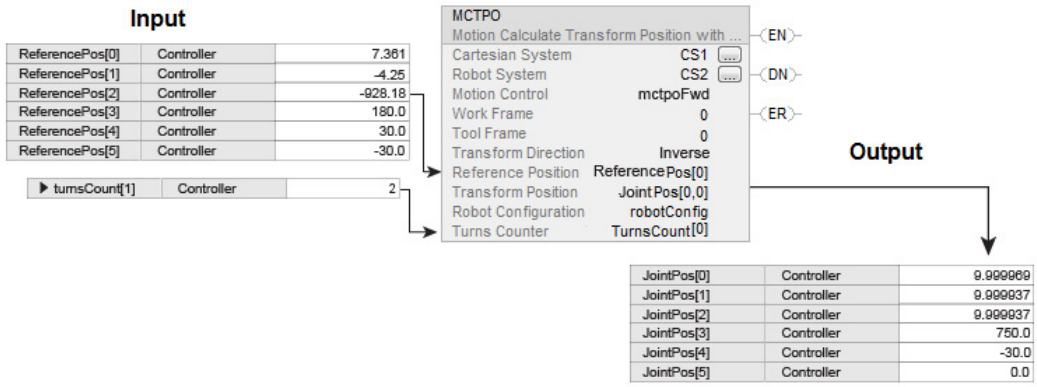
Error Code	EX_ERROR Code	Description
67	1	Invalid Transform Position MOP Invalid Rx orientation Invalid Rx orientation value at EOA transformations. For 4 axis geometries (Delta, SCARA, Articulated Dependent) only Rx = 180 deg position is allowed. For all other positions, it will generate error.
67	2	Invalid Transform Position MOP Invalid Ry orientation Invalid Ry orientation value at EOA transformations. For 4 axis geometries, after removing work frame and tool frame, there should not be any Ry orientation values at EOA.
67	3	Invalid Transform Position MOP Invalid Ry orientation
147	3 - 5	Invalid Orientation Scaling Constant Extended error code 3 : Rx Extended error code 4 : Ry Extended error code 5 : Rz Orientation axis has scaling constant > MAX_K_CONSTANT_FOR_ORIENTATION_AXIS. or has scaling constant which is not Integer or has Conversion ratio between Coordination Units and Position Units other than 1:1.
148	3 -5	MCPM Orientation Offset Not Zero Rx orientation offset not valid. Extended error code 3 : Rx orientation offset not valid. Extended error code 4 : Ry orientation offset not valid. Extended error code 5 : Rz orientation offset not valid. If robot Geometry is (MO_CD_J1J2J6 or MO_CD_J1J2J3J6) and: • workframe offset Rx isn't 0 or • workframe offset Ry isn't 0 or • toolframe offset Rx isn't 0 or • toolframe offset Ry isn't 0 Or if robot Geometry is MO_CD_J1J2J3J4J5 and: • workframe offset Rx isn't 0 or • workframe offset Ry isn't 0 or • toolframe offset Rx isn't 0 or • toolframe offset Rz isn't 0
151	1	Joint Angle J1 Beyond Limits For details, refer to the geometry specific configuration sections for the joint range limitation.
151	2	Joint Angle J2 Beyond Limits For details, refer to the geometry specific configuration sections for the joint range limitation.
151	3	Joint Angle J3 Beyond Limits For details, refer to the geometry specific configuration sections for the joint range limitation.

Error Code	EX_ERROR Code	Description
151	4	Joint Angle J4 Beyond Limits For details, refer to the geometry specific configuration sections for the joint range limitation.
151	5	Joint Angle J5 Beyond Limits For details, refer to the geometry specific configuration sections for the joint range limitation.
151	6	Joint Angle J6 Beyond Limits For details, refer to the geometry specific configuration sections for the joint range limitation.
153	1	Invalid Translation Position MOP Invalid X Translation Translation on X axis is Invalid
153	2	Invalid Translation Position MOP Invalid Y Translation Translation on Y axis is Invalid
153	3	Invalid Translation Position MOP Invalid Z Translation Translation on Z axis is Invalid
156	1	Singularity Condition Error Joint 1 axis is close to Arm Singularity condition in 6-axis Articulated Independent Geometry Ext Error 1: Arm Singularity Condition
156	2	Singularity Condition Error Joint 3 axis is close to Elbow Singularity condition in 6-axis Articulated Independent Geometry Ext Error 2: Elbow Singularity Condition
156	3	Singularity Condition Error Joint 5 axis is close to Wrist Singularity condition in 6-axis Articulated Independent Geometry Ext Error 1: Wrist Singularity Condition

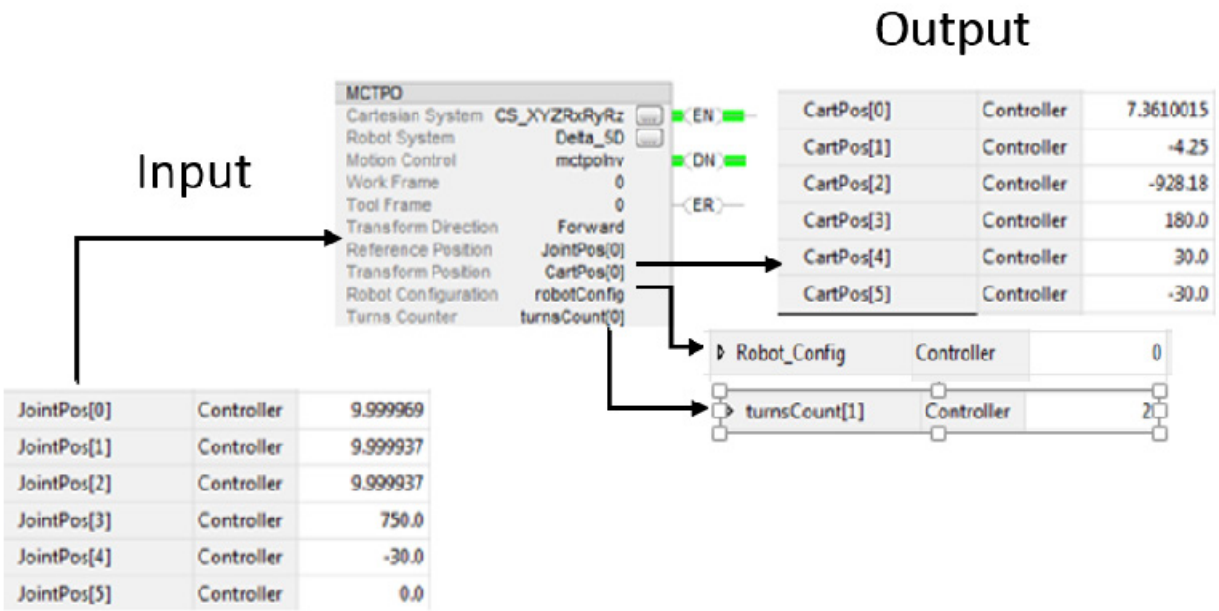
Example

Ladder Diagram

This example illustrates an MCTPO instruction with Transform Direction as Inverse, where the user feeds Cartesian positions and turns counter as input. The instruction computes the corresponding target joint angle positions and is written to the Transform Position parameter as the output.



This example illustrates the MCTPO instruction with Transform Direction as Forward. The target positions are guided into the Reference position operand as input. The instruction computes the corresponding Cartesian positions and turns counter as the output.



Structured Text

MCTPO(CS1, CS2, MCTPo1[0], BaseFrame, ToolFrame, Forward, refPos[0], transPos[0], robotConfig[0], TurnsCounter[0]);

Tip: For further information on creating geometries with orientation support, see the Motion Coordinate System User Manual publication MOTION-UM002.

See also

Structured Text Syntax on page 665

Index Through Arrays on page 691

Motion Error Codes (.ERR) on page 581

Multi-Axis Coordinated Motion Instructions on page 359

Define coordinate system frames

Motion Coordinated Circular Move (MCCM)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Use the MCCM instruction to initiate a two or three-dimensional circular coordinated move for the specified axes within a Cartesian coordinate system. New position is defined as either an absolute or incremental position and done at the desired speed. The actual speed of the MCCM is a function of the mode of the move (commanded speed or percent of maximum speed). The speed of the move is based on the time it takes to complete the circular move using the programmed axes. Each axis is commanded to move at a speed that allows for all axes to reach the endpoint (target position) at the same time.

The dimension of the circle is defined by the number of axes contained within the coordinate system. For example, if you have a coordinate system that contained three axes with an MCCM instruction that has motion in only two dimensions, the resultant move is still considered a three-dimensional arc or circle.

IMPORTANT Tags used for the motion control attribute of instructions should only be used once. Re-use of the motion control tag in other instructions can cause unintended operation. This may result in damage to equipment or personal injury.

IMPORTANT Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is by changing move dynamics (MCD or MCCD) or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-curve velocity profile can overshoot if:

maximum deceleration is decreased while the move is decelerating or close to the deceleration point; or

maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

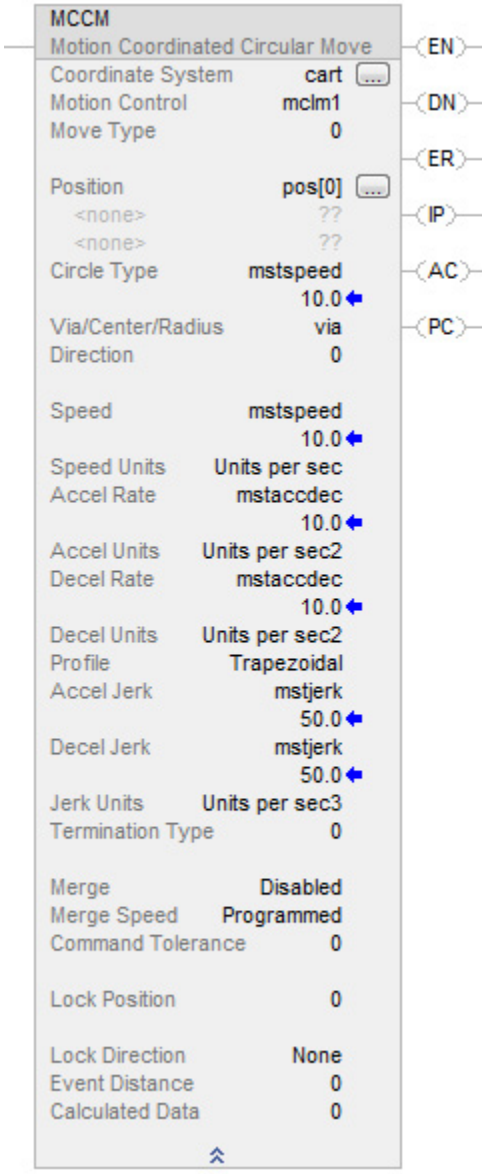
This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MCCM (CoordinateSystem, MotionControl, MoveType, Position, CircleType, Via/Center/Radius, Direction, Speed, SpeedUnits, AccelRate, AccelUnits, DecelRate, DecelUnits, Profile, AccelJerk, DecelJerk,

JerkUnits,Terminationtype, Merge, Mergespeed, CommandTolerance, LockPosition, LockDirection, EventDistance, CalculatedData);

Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion.

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Coordinate System	COORDINATE_SYSTEM	Tag	Coordinated group of axes. Reference to coordinated group of axes. For controllers that support the REF_TO motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Move Type	SINT, INT, or DINT	Immediate or Tag	Select the Move Type: 0 = Absolute 1 = Incremental
Position	REAL	Array tag [ ]	[coordinate units]
Circle Type	SINT, INT, or DINT	Immediate or Tag	0 = Via 1 = Center 2 = Radius 3 = Center Incremental
Via/Center/Radius	REAL	array tag[ ] (via/center) Immediate or tag (radius)	[coordinate units]
Direction	SINT, INT, or DINT	Immediate or Tag	**2D** / **3D** 0 = CW / Shortest 1 = CW / Longest 2 = CW Full / Shortest Full 3 = CCW Full / Longest Full
Speed	SINT, INT, DINT, or REAL	Immediate or Tag	[coordinate units]
Speed Units	SINT, INT, or DINT	Immediate	0 = Units per Sec 1 = % of Maximum 4= Units per MasterUnit
Accel Rate	SINT, INT, DINT, or REAL	Immediate or Tag	[coordinate units]
Accel Units	SINT, INT, or DINT	Immediate	0 = Units per Sec^2 1 = % of Maximum 4= Units per MasterUnit^2
Decel Rate	SINT, INT, DINT, or REAL	Immediate or Tag	[coordinate units]

Decel Units	SINT, INT, or DINT	Immediate	0 = Units per Sec^2 1 = % of Maximum 4= Units per MasterUnit^2
Profile	SINT, INT, or DINT	Immediate	0 = Trapezoidal 1 = S-curve
Accel Jerk	SINT, INT, DINT, or REAL	Immediate or Tag	You must always enter values for the Accel and Decel Jerk operands. This instruction only uses the values if the Profile is configured as S-curve. Accel Jerk is the acceleration jerk rate for the coordinate system. Decel Jerk is the deceleration jerk rate for the coordinate system. Enter the jerk rates in these Jerk Units. 0 = Units per sec^3 1 = % of Maximum 2 = % of Time 4 = Units per MasterUnit^3 6 = % of Time-Master Driven Use these values to get started. Accel Jerk = 100 (% of Time) Decel Jerk = 100 (% of Time) Jerk Units = 2
Decel Jerk	SINT, INT, DINT, or REAL	Immediate or Tag	
Jerk Units	SINT, INT, or DINT	Immediate or Tag	
Termination Type	SINT, INT, or DINT	Immediate or Tag	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 Choose a Termination Type in the Related Topics section below.
Merge	SINT, INT, or DINT	Immediate	0 = Disabled 1 = Coordinated Motion 2 = All Motion
Merge Speed	SINT, INT, or DINT	Immediate	0 = Programmed 1 = Current
Command Tolerance	REAL	Immediate Real or Tag	The position on a coordinated move where blending should start. This parameter is used in place of Command Tolerance in the Coordinate System if Termination Type 6 is used. **Tip**: Termination type 2 is identical to Termination Type 6 except the Command Tolerance value from the coordinate system is used and this parameter is ignored.

Lock Position	REAL	Immediate or Tag	Position on the Master Axis where a Slave should start following the master after the move has been initiated on the Slave Axis. See the Structure section below for more information.
Lock Direction	UINT32	Immediate	Specifies the conditions when the Lock Position should be used. See the Structure section below for more information.
Event Distance	REAL ARRAY or 0	Array	The position(s) on a move measured from the end of the move. See the Structure section below for more information.
Calculated Data	REAL ARRAY or 0	Array	Master Distance(s) (or time) needed from the beginning of the move to the Event Distance point. See the Structure section below for more information.

Structured Text

See Structured Text Syntax for more information on the syntax of expressions within structured text.

When entering enumerations for the operand value in structured text, multiple word enumerations must be entered without spaces. For example: when entering Decel Units the value is entered as unitspersec² rather than Units per Sec² as displayed in the ladder logic.

Use the entries in this table as a guide when entering structured text operands.

This Operand	**Has These Options Which You**	
	Enter as Text	**Or as**
Coordinate System	No enumeration	Tag
Motion Control	No enumeration	Tag
Move Type	No enumeration	0 (Absolute) 1 (Incremental)
Position	No enumeration	Array Tag
Circle Type	No enumeration	Tag 0 = Via 1 = Center 2 = Radius 3 = Center Incremental
Via/Center/Radius	No enumeration	Array Tag (via/center) Immediate or tag (radius)

Direction	No enumeration	**2D** 0=CW 1=CW 2=CW Full **3D** Shortest Longest Shortest Full
Speed	No enumeration	Immediate or Tag
Speed Units	unitspersec %ofmaximum unitspermasterunit	0 1 4
Accel Rate	No enumeration	Immediate or Tag
Accel Units	unitspersec^2 %ofmaximum unitspermasterunit^2	0 1 4
Decel Rate	No enumeration	Immediate or Tag
Decel Units	unitspersec^2 %ofmaximum unitspermasterunit^2	0 1 4
Profile	trapezoidal s-curve	0 1
Accel Jerk	No enumeration	Immediate or tag You must always enter a value for the Accel and Decel Jerk operands. This instruction only uses the values if the Profile is configured as S-curve. Use these values to get started. Accel Jerk = 100 (% of Time) Decel Jerk = 100 (% of Time) Jerk Units = 2
Decel Jerk	No enumeration	
Jerk Units	Unitspersec^3 %ofmaximum %oftime unitspermasterunit3 %oftimemasterdriven	0 1 2 (use this value to get started) 4 6
Termination Type	No enumeration	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 Choose a Termination Type in the Related Topics section below.
Merge	disabled coordinatedmotion allmotion	0 1 2
Merge Speed	programmed current	0 1
Command Tolerance	No enumeration	Immediate or Tag
Lock Position	No enumeration	Immediate, Real, or Tag

Lock Direction	None immediateforwardonly Immediatereverseonly positionforward positionreverse	0 1 2 3 4
Event Distance	No enumeration	Array
Calculated Data	No enumeration	Array

Coordinate System

The Coordinate System operand specifies the system of motion axes that define the dimensions of a Cartesian coordinate system. For this release, the coordinate system supports up to three (3) primary axes. Only the axes configured as primary axes (up to 3) are included in speed calculations. Only primary axes participate in the actual circular move.

Motion Control

The following control bits are affected by the MCCM instruction.

Mnemonic	Description
.EN (Enable) Bit 31	The Enable bit is set when the rung transitions from false to true and resets when the rung goes from true to false.
.DN (Done) Bit 29	The Done bit sets when the coordinated instruction has been verified and queued successfully. Because it's set at the time it's queued it may appear as set when a runtime error is encountered during the verify operation after it comes out of the queue. It resets when the rung transitions from false to true.
.ER (Error) Bit 28	The Error bit resets when the rung transitions from false to true. It sets when the coordinated move fails to initiate successfully. It can also be set with the Done bit when a queued instruction encounters a runtime error.
.IP (In Process) Bit 26	The In Process bit sets when the coordinated move is successfully initiated. It resets when there is a succeeding move and the coordinated move reaches the new position, or when there is no succeeding move and the coordinated move reaches the termination type specifications, or when the coordinated move is superseded by another MCCM or MCLM instruction with a Merge Type of Coordinated Move or when terminated by an MCS or an MCSD instruction.
.AC (Active) Bit 23	When you have a coordinated move instruction queued, the Active bit lets you know which instruction is controlling the motion. It sets when the coordinated move becomes active. It is reset when the Process Complete bit is set or when the instruction is stopped.

.PC (Process Complete) Bit 27	The Process Complete bit is reset when the rung transitions from false to true. It is set when there is no succeeding move and the coordinated move reaches the new position, or when there is a succeeding move and the coordinated move reaches the specified termination type.
.ACCEL (Acceleration Bit) Bit 01	The Acceleration bit sets while the coordinated move is in the acceleration phase. It resets while the coordinated move is in the constant velocity or deceleration phase, or when coordinated motion concludes.
.DECEL (Deceleration Bit) Bit 02	The Deceleration bit sets while the coordinated move is in the deceleration phase. It resets while the coordinated move is in the constant velocity or acceleration phase, or when coordinated motion concludes.

Move Type

The Move Type operand determines the method used by the position array to indicate the path of the coordinated move and the method the via/center/radius parameter uses to indicate the via and center circle positions. The options are: Absolute or Incremental.

- Absolute - the coordinate system moves to the specified Position at the defined Speed, using the Accel and Decel Rates as designated by their respective operands, along a circular path.
  When an axis is configured for rotary operation, absolute moves are handled in the same manner as with linear axes. When the axis position exceeds the Unwind parameter, an error is generated.
- The sign of the specified position array is interpreted by the controller as the direction for the move. Negative position values instruct the interpolator to move the rotary axis in a negative direction to obtain the desired absolute position. A positive value indicates that positive motion is desired to reach the target position. To move to the unwind position in the negative direction a negative unwind position value must be used as 0 and -0 are treated as 0. When the position is greater than the unwind value, an error is generated. The axis can move through the unwind position but never incrementally more than one unwind value.
- Incremental - the coordinate system moves the distance as defined by the position array at the specified Speed, using the Accel and Decel rates determined by the respective operands, along a circular path.

The specified distance is interpreted by the interpolator and can be positive or negative. Negative position values instruct the interpolator to move the rotary axis in a negative direction, while positive values indicate positive motion is desired to reach the target position.

Position

The Position operand is a one dimensional array whose dimension is at least equivalent to the number of axes specified in the coordinate system. It is the position array that defines the new absolute or incremental position.

Circle Type

The Circle Type operand specifies how the array labeled via/center/radius is interpreted. The options are: Via, Circle, Radius, Center Incremental.

- Via indicates that the via/center/radius array members specify a via point between the start and end points.
- Center indicates that the via/center/radius array members contain the circle center.
- Radius indicates that the first via/center/radius array member contains the radius. Other members are ignored. Radius is valid only in two-dimensional coordinate systems.
- Center Incremental indicates that the via/center/radius array members define a position that always incrementally defines the center of the circle regardless of Move Type operand. Sign of the incremental value is measured from the start point to the center.

Via/Center/Radius

Depending on the selected Move Type and Circle Type, the via/center/radius position parameter defines the absolute or incremental value of a position along the circle, the center of the circle, or the radius of the circle as defined in the following table. If the Circle Type is via or center, the via/center/radius position parameter is a one-dimensional array, whose dimension is defined to be at least the equivalent of the number of axes specified in the coordinate system. If the Circle type is radius, the via/center/radius position parameter is a single value.

Move Type	Cycle Type	Behavior
Absolute	Via	The via/center/radius position array defines a position along the circle. For a non-full circle case, the Position parameter array defines the endpoint of the arc. For a full circle case, the Position parameter array defines any second point along the circle except the endpoint.
Incremental	Via	The sum of the via/center/radius position array and the old position defines the position along the circle. For a non-full circle case, the sum of the Position parameter array and the old position defines the endpoint of the arc. For a full circle case, the sum of the Position parameter array and the old position defines any second point along the circle except the endpoint.

Absolute	Center	The via/center/radius position array defines the center of the circle. For a non-full circle case, the Position parameter array defines the endpoint of the arc. For a full circle case, the Position parameter array defines any second point along the circle except the endpoint.
Incremental	Center	The sum of the via/center/radius position array and the old position defines the center of the circle. For a non-full circle case, the sum of the Position parameter array and the old position defines the endpoint of the arc. For a full circle case, the sum of the Position parameter array and the old position defines any second point along the circle except the endpoint.
Absolute or Incremental	Radius	The via/center/radius position single value defines the arc radius. The sign of the value is used to determine the center point to distinguish between the two possible arcs. A positive value indicates a center point that generates an arc less than 180 degrees. A negative value indicates a center point that generates an arc greater than 180 degrees. This Circle Type is only valid for two-dimensional circles. The position parameter array follows the Move Type to define the endpoint of the arc.
Absolute	Center Incremental	The sum of the via/center/radius position array and the old position defines the center position of the circle. For a non-full circle case, the Position parameter array defines the endpoint of the arc. For a full circle case, the Position parameter array defines any second point along the circle except the endpoint.
Incremental	Center Incremental	The sum of the via/center/radius position array and the old position defines the center position of the circle. For a non-full circle case, the sum of the Position parameter array and the old position defines the endpoint of the arc. For a full circle case, the sum of the Position parameter array and the old position defines any second point along the circle except the endpoint.

Direction

The Direction operand defines the rotational direction of a 2D circular move as either clockwise or counterclockwise according to the right-hand screw rule. For a 3D circular move the direction is either Shortest or Longest. In both 2D and 3D it can also indicate if the circular move is to be a full circle.

Speed

The Speed operand defines the maximum vector speed along the path of the coordinated move.

Speed Units

The Speed Units operand defines the units applied to the Speed operand either directly in coordinate units or as a percentage of the maximum values defined in the coordinate system.

Accel Rate

The Accel Rate operand defines the maximum acceleration along the path of the coordinated move.

Accel Units

The Accel Units operand defines the units applied to the Accel Rate operand either directly in coordinate units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system.

Decel Rate

The Decel Rate operand defines the maximum deceleration along the path of the coordinated move.

Decel Units

The Decel Units operand defines the units applied to the Decel Rate operand either directly in coordinate units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system.

Profile

The Profile operand determines whether the coordinated move uses a trapezoidal or an S-curve velocity profile.

Accel Jerk

Accel Jerk defines the maximum acceleration jerk for the programmed move. For more information on calculating Accel Jerk, refer to the Jerk Units section below.

Decel Jerk

Decel Jerk defines the maximum deceleration jerk for the programmed move. For more information on calculating Decel Jerk, refer to the Jerk Units section below.

Jerk Units

The jerk units define the units that are applied to the values entered in the Accel Jerk and Decel Jerk operands. The values are entered directly in the position units of the specified coordinate system or as a percentage. When configured using % of Maximum, the jerk is applied as a percentage of the Maximum Acceleration Jerk and Maximum Deceleration Jerk operands specified in the coordinate system attributes. When configured using % of Time, the value is a percentage based on the Speed, Accel Rate, and Decel Rate specified in the instruction.

If you want to convert engineering units to % of Time, use these equations.

For Accel Jerk:

$$j_a\,[\mathrm{EU/s^3}] = \frac{a_{max}{}^2\,[\mathrm{EU/s^2}]}{v_{max}\,[\mathrm{EU/s}]}\left(\frac{200}{j_a\,[\%\ \mathrm{of\ time}]} - 1\right)$$

For Decel Jerk:

$$j_a\,[\mathrm{EU/s^3}] = \frac{a_{max}{}^2\,[\mathrm{EU/s^2}]}{v_{max}\,[\mathrm{EU/s}]}\left(\frac{200}{j_a\,[\%\ \mathrm{of\ time}]} - 1\right)$$

If you want to convert % of Time to engineering units, use these equations.

For Accel Jerk:

$$j_a\,[\%\ \mathrm{of\ time}] = \frac{2}{1 + \dfrac{j_a\,[\mathrm{EU/s^3}]\ v_{max}\,[\mathrm{EU/s}]}{a_{max}^2\,[\mathrm{EU/s^2}]}}\,100$$

For Decel Jerk:

$$j_d\,[\%\ \mathrm{of\ time}] = \frac{2}{1 + \dfrac{j_d\,[\mathrm{EU/s^3}]\ v_{max}\,[\mathrm{EU/s}]}{d_{max}^2\,[\mathrm{EU/s^2}]}}\,100$$

Important Consideration

If you program tangent circles with different Jerk rates (Decel Jerk of first circle and Accel Jerk of the second circle), then you might get a slight velocity discontinuity at the intersection of the two circles. The size of the discontinuity depends on magnitude of the Jerk difference. In other words, the smaller the Jerk difference, the smaller the velocity glitch. Therefore, we recommend that you do not program Jerk rates on tangent circles.

Termination Type

For Master Driven Speed Control (MDSC), when all sequential instructions run in the same mode (Master Driven Mode or Time Driven Mode), then all termination types are supported. If the termination type switches in the coordinated motion queue, errors may generate depending on the sequence of motion types.

The following is only applicable if a move on a slave Coordinate System uses a Blending Termination Type (Termination Types 2, 3, or 6) and is programmed in MDSC mode.

If you use the Calculated Data returned in the last MCCM instruction of a motion sequence to program the length that the master axis has to move for the motion sequence to go PC, then there is the possibility that you will have to add a small safety margin to the Calculated Data. If you do not add this margin, there is a chance that the motion of the master axes completes before the entire motion sequence programmed on the Slave Coordinate System finishes. If this occurs, the last motion instruction on the Slave Coordinate System remains active and does not go PC. The value of the small safety margin is dependent on the Command Tolerance used for the first and last move in the motion sequence as follows:

CUP = Coarse Update Period

MAS = Master Axis Speed

- If a Command Tolerance value of 100% is used for the first move in the sequence then:

SafetyMargin1 = CUP * MAS

else

SafetyMargin1 = CUP * MAS * .02

- For all other moves in the blending sequence between first and last:

SafetyMargin2 = CUP * MAS * .02 * number of blending moves between 1st and last

- If a Command Tolerance value of 100% is used for the last in the sequence then:

SafetyMargin3 = CUP * MAS

else

SafetyMargin3 = CUP * MAS * .02

- Final SafetyMargin = SafetyMargin1 + SafetyMargin2 + SafetyMargin3

Once a sequence is programmed and verified, it will repeat.

Click the link Choose a Termination Type in the Related Topics section below for information.

Merge

The merge defines whether or not to turn the motion of all specified axes into a pure coordinated move. The options are: Merge Disabled, Coordinated Motion, or All Motion.

- Merge Disabled - Any currently executing single axis motion instructions involving any axes defined in the specified coordinate system are not affected by the activation of this instruction, and result in superimposed motion on the affected axes. An error is flagged if a second instruction is initiated in the same coordinate system or in another coordinate system containing any axes in common with the coordinate system that is active.
- Coordinated Motion - Any currently executing coordinated motion instructions involving the same specified coordinate system are terminated, and the active motion is blended into the current move at the speed defined in the merge speed parameter. Any pending coordinated motion instructions in the specified coordinate system are canceled. Any currently executing system single axis motion instructions involving any axes defined in the specified coordinate system are not affected by the activation of this instruction, and result in superimposed motion on the affected axes.
- All Motion - Any currently executing single axis motion instructions involving any axes defined in the specified coordinate system and any currently executing coordinated motion instructions are terminated. The prior motion is merged into the current move at the speed defined in Merge Speed parameter. Any pending coordinated move instructions are canceled.

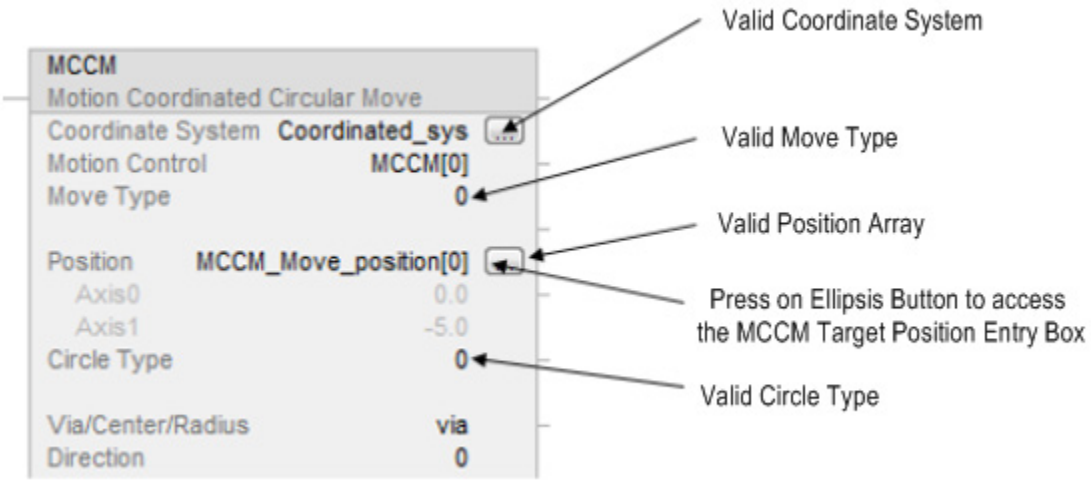
Merge Speed

The Merge Speed operand defines whether the current speed or the programmed speed is used as the maximum speed along the path of the coordinated move when Merge is enabled. Current speed is the vector sum of all motion (for example, jogs, MAM's, and geared motion) for all axes defined in the current coordinate system.

MCCM Target Position Entry Dialog Box

The MCCM Target Position Entry Dialog box is accessed by pressing the ellipsis button to the right of the position operand of the ladder instruction faceplate. The Target Position Entry box can only be accessed if the coordinate system for the instruction has been named, has a valid tag name for the Position operand that contains enough elements to accommodate the number of axes, selected a valid Move Type and a valid Circle Type. If these criteria have not been satisfied, an error message is displayed on the status bar.

MCCM Ladder Valid Values for Accessing Target Position Entry Box



Press the ellipsis and the **Target Position Entry Position** tab dialog box displays.

Tip: This action is not possible for a reference.

Target Position Entry Dialog Box Fields

Feature	Description
Axis Name	This column has the names of each axis in the coordinate system named in the ladder faceplate. These names are not editable.

Target Position/Target Increment	The values in this column are numeric. They show the endpoint or incremental departure of the move depending on the active Move Type. The column heading indicates which is displayed.
Actual Position	This column contains the current actual position of the axes in the coordinate system. These values update dynamically when on-line and the Coordinate System Auto Tag Update is enabled.
Via Position/Via Increment Center Position/Center Increment Radius	Depending on the Circle Type selected, this column contains the Via point position or increment, the Center Position or increment.
Set Targets = Actuals	This button is enabled when the Move Type is Absolute and is used to copy the value from the Actual Position fields to the Target Position fields.
Set Vias = Actuals	This button is only active if the Move Type is Absolute. It is used to copy the values from the Actual Position fields to the Vias Fields.

The Move Type and Circle Type selected govern the appearance of this dialog box. This table illustrates how the screen is affected by the combinations of Move Type and Circle Type selected.

Target Position Entry Dialog Box Changes

Move Type	Circle Type	Behavior
Absolute	Via	Target column is entitled Target Position. Via column is entitled Via Position. Set Targets = Actuals button is active. Set Vias = Actuals button is active.
Incremental	Via	Target column is entitled Target Increment. Via Column is entitled Via Increment. Set Targets = Actuals button is inactive (appears dimmed). Set Vias = Actuals button is inactive (appears dimmed).
Absolute	Center	Target column is entitled Target Position. Center column is entitled Center Position. Set Targets = Actuals button is active. Set Vias = Actuals button is active.
Incremental	Center	Target column is entitled Target Increment. Center Column is entitled Center Increment. Set Targets = Actuals button is inactive (appears dimmed). Set Vias = Actuals button is inactive (appears dimmed).
Absolute	Radius	Target column is entitled Target Position. Radius column is entitled Radius. Set Targets = Actuals button is active. Set Vias = Actuals button is inactive (appears dimmed).
Incremental	Radius	Target column is entitled Target Increment. Radius Column is entitled Radius. Set Targets = Actuals button is inactive (appears dimmed). Set Vias = Actuals button is inactive (appears dimmed).

Absolute	Center Incremental	Target column is entitled Target Position. Center Incremental column is entitled Center Incremental. Set Targets = Actuals button is active. Set Vias = Actuals button is inactive (appears dimmed).
Incremental	Center Incremental	Target column is entitled Target Increment. Center Incremental column is entitled Center Incremental. Set Targets = Actuals button is inactive (appears dimmed). Set Vias = Actuals button is inactive (appears dimmed).

MCCM is a transitional instruction:

- In relay ladder, toggle the Rung-condition-in from false to true each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition. See Structured Text Syntax.

Structure

See Input and Output Parameters Structure for Single Axis Motion Instructions for the input and output parameters that are available for the MCCM instruction via the Master Driven Speed Control (MDSC) function. Before any of these parameters is active, you must execute an MDCC instruction and it must be active (IP bit is set).

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of the incorrect type is passed to the Coordinate System operand. See Common Attributes for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Runtime Error Conditions

You cannot switch from Time Driven Mode to Master Driven Mode if the master speed is zero unless the slave speed is zero too.

Extended Error Codes

Extended Error codes help to further define the error message given for this particular instruction. Their behavior is dependent upon the Error Code with which they are associated.

The Extended Error Codes for Servo Off State (5), Shutdown State (7), Axis Type Not Servo (8), Axis Not Configured (11), Homing In Process Error (16), and Illegal Axis Data type (38) errors all function in the same fashion. A number between 0...n is displayed for the Extended Error Code. This number is the index to the Coordinate System indicating the axis that is in the error condition.

For Error Code Axis Not Configured (11) there is an additional value of -1 which indicates that Coordinate System was unable to setup the axis for coordinate motion.

For the MCCM instruction, Error Code 13 - Parameter Out of Range, Extended Errors return a number that indicates the offending parameter as listed on the faceplate in numerical order from top to bottom beginning with zero. For example, 2 indicates the parameter value for Move Type is in error.

Referenced Error Code and Number	Extended Error Numeric Indicator	Instruction Parameter	Description
Parameter Out Of Range (13)	0	Coordinate System	Number of primary axes is not 2 or 3.
Parameter Out Of Range (13)	2	Move Type	Move Type is either less than 0 or greater than 1.
Parameter Out Of Range (13)	3	Position	The position array is not large enough to provide positions for all the axes in the coordinate system.
Parameter Out Of Range (13)	4	Circle Type	Circle Type is either less than 0 or greater than 4.
Parameter Out Of Range (13)	5	Via/Center/Radius	The size of the Via/Center array is not large enough to provide positions for all of the axes in the defining via/center point.
Parameter Out Of Range (13)	6	Direction	Direction is either less than 0 or greater than 3.
Parameter Out of Range (13)	7	Speed	Speed is less than 0.
Parameter Out of Range (13)	9	Accel Rate	Accel Rate is less than or equal to 0.
Parameter Out of Range (13)	11	Decal Rate	Decel Rate is less than or equal to 0.

Parameter Out of Range (13)	14	Termination Type	Termination Type is less than 0 or greater than 3.

For the Error Code 54 – Maximum Deceleration Value is Zero, if the Extended Error returns a positive number (0-n) it's referring to the offending axis in the coordinate system. Go to the Coordinate System Properties General Tab and look under the Brackets ([ ])column of the Axis Grid to determine which axis has a Maximum Deceleration value of 0. Click on the ellipsis button next to the offending axis to access the Axis Properties screen. Go to the Dynamics tab and make the appropriate change to the Maximum Deceleration Value. If the Extended Error number is -1, this means the Coordinate System has a Maximum Deceleration Value of 0. Go to the Coordinate System Properties Dynamics Tab to correct the Maximum Deceleration value.

MCCM Changes to Status Bits:

Status bits provide a means for monitoring the progress of the motion instruction. There are three types of Status bits that provide pertinent information.

- Axis Status bits
- Coordinate System
- Coordinate Motion

When the MCCM instruction initiates, the status bits undergo these changes.

Axis Status Bits

Bit Name	Meaning
CoordinatedMotionStatus	Sets when the MCCM instruction executes and is cleared when the instruction completes.

Coordinate System Status Bits

Bit Name	Meaning
MotionStatus	Sets when the MCCM instruction is active and the Coordinate System is connected to its associated axes.

Coordinated Motion Status Bits

Bit Name	Meaning
AccelStatus	Sets when vector is accelerating. Clears when a blend is in process or when vector move is at speed or decelerating.
DecelStatus	Sets when vector is decelerating. Clears when a blend is in process or when vector move is accelerating or when move completes.

ActualPosToleranceStatus	Sets for Actual Tolerance termination type only. The bit is set after the following two conditions have been met. 1) Interpolation is complete. 2) The actual distance to the programmed endpoint is less than the configured coordinate system's Actual Tolerance value. It remains set after the instruction completes. It is reset when a new instruction is started.
CommandPosToleranceStatus	Sets for all termination types whenever the distance to the programmed endpoint is less than the configured coordinate system's Command Tolerance value and remains set after the instruction completes. It is reset when a new instruction is started. The CommandPosToleranceStatus (CS_CMD_POS_TOL_STS) status bit in the Coordinate System is set as follows: TT0, TT1, TT4, TT5 - Bit is set when the distance to the endpoint is less than the Command Tolerance value. The bit is cleared when the first move is complete. TT2, TT6 - Bit is set when the distance to the endpoint is less than the Command Tolerance value. The bit is cleared when the blend is started (that is, when the second move is started). Thus, you may not see the bit if the blend is started at the Command Tolerance (CT) point. The blend may have been deferred slightly beyond the CT point if the next move is a short move or for time matching of the acceleration and deceleration of the two adjacent moves. TT3 - Bit is set when the distance to the endpoint is less than the Command Tolerance value (like TT2 and TT6). The bit is cleared when the blend is started. Thus, you may not see the bit if the blend is started at the deceleration point. The blend may have been deferred slightly beyond the deceleration point if the next move is a short move or for time matching of the acceleration and deceleration of the two adjacent moves.
StoppingStatus	The Stopping Status bit is cleared when the MCCM instruction executes.
MoveStatus	Sets when MCCM begins axis motion. Clears on the .PC bit of the last motion instruction or a motion instruction executes which causes a stop.
MoveTransitionStatus	Sets when No Decel or Command Tolerance termination type is satisfied. When blending collinear moves the bit is not set because the machine is always on path. It clears when a blend completes, the motion of a pending instruction starts, or a motion instruction executes which causes a stop. Indicates not on path.
MovePendingQueueFullStatus	Sets when the instruction queue is full. It clears when the queue has room to hold another new coordinated move instruction.
CoorMotionLockStatus	Set when an axis lock is requested for an MCLM or MCCM instruction and the axis has crossed the Lock Position. Cleared when an MCLM or MCCM is initiated. For the enumerations Immediate Forward Only and Immediate Reverse Only, the bit is set immediately when the MCLM or MCCM is initiated. When the enumeration is Position Forward Only or Position Reverse Only, the bit is set when the Master Axis crosses the Lock Position in the specified direction. The bit is never set if the enumeration is NONE. The CoordMotionLockStatus bit is cleared when the Master Axis reverses direction and the Slave Axis stops following the Master Axis. The CoordMotionLockStatus bit is set again when the Slave Coordinate System resumes following the Master Axis. The CoordMotionLockStatus bit is also cleared when an MCCS is initiated.

Currently, Coordinated Motion supports the queueing of one coordinated motion instruction. Therefore the MovePendingStatus bit and the MovePendingQueueFullStatus bit are always the same.

Circular Programming Reference Guide

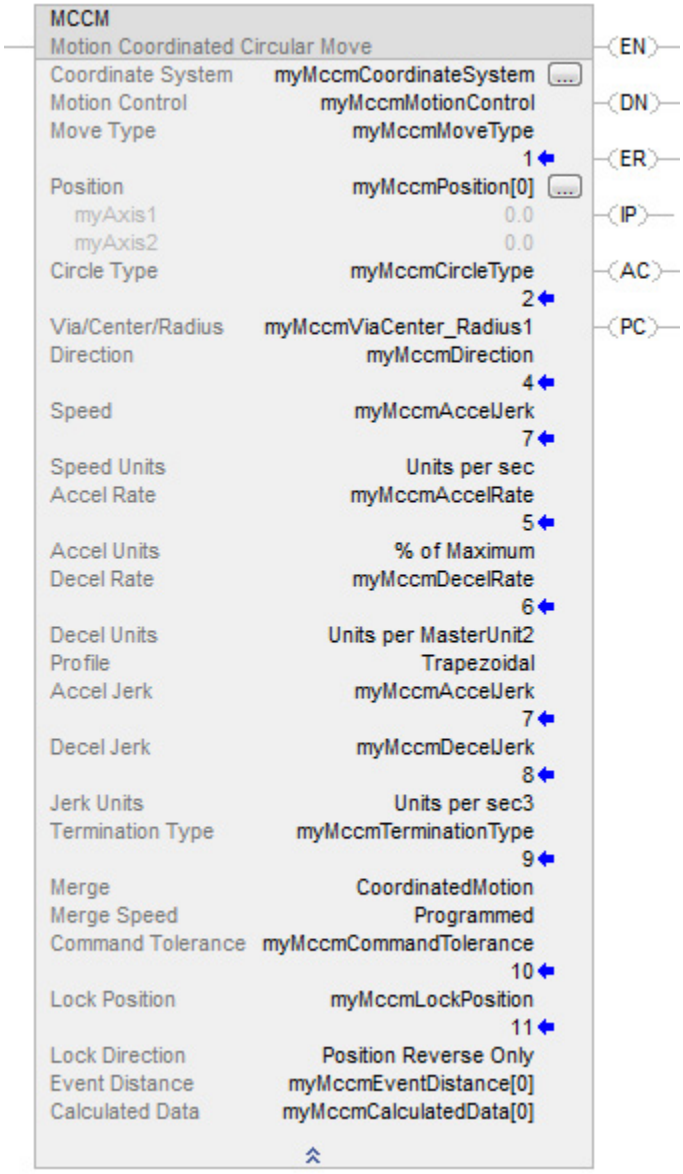
Circle Type	Used in 2D/3D/Both	Validation Errors	Direction - 2D	Direction - 3D	Comment
Radius	2D	Error 25; Illegal Instruction Error 45 Endpoint = Startpoint Error 49; R too small (|R| < .001) or R too short to span programmed points.	CW/CCW as viewed from the + perpendicular to the circular plane.	N/A	A + radius forces arc length to be <= 180° (Shortest arc). A - radius forces arc length to be => 180° (Longest arc). Full Circles can be programmed. For full circles: set Position to be any point on circle except Startpoint and use one of the Full direction types.
Center Point	Both	Error 44; Collinearity (3D only) Error 45; Endpoint = Startpoint (3D only) Error 46; Start/End radius mismatch (|R1 - R2| > .15 * R1).	CW/CCW as viewed from the + perpendicular to the circular plane.	Shortest/Longest arc. In Full circles, placement of endpoint defines shortest/longest paths referred to by direction parameter.	Full Circles can be programmed. In 2D only, Endpoint = Startpoint is legal. Therefore, full circles may be generated: By setting Endpoint = Startpoint, in which case, all direction types produce full circles. By setting Endpoint not = Startpoint and using Full direction type. For 3D Full Circles: set Position to be any point on the circle except Startpoint, and use one of the Full direction types. Position defines both arc and Shortest direction types.
Via Point	Both	Error 44; Collinearity Error 45; Endpoint = Startpoint	Via point always determines direction.	Via point always determines direction. Direction operand is only used to determine if circle is partial or full.	Full Circles can be programmed. For full circles: set Position to be any point on circle except Startpoint and use one of the Full direction types.

Master Driven Speed Control (MDSC) and Motion Direct Command Support

The Motion Direct commands are not available in the instruction tree for the MCCM instruction. You must program an MCCM in one of the supported programming languages before you execute an MAM or MAJ in Time Driven Mode. A runtime error will occur if an MCCM is not previously executed in an MAM and MAJ in Master Driven Mode.

Example 1

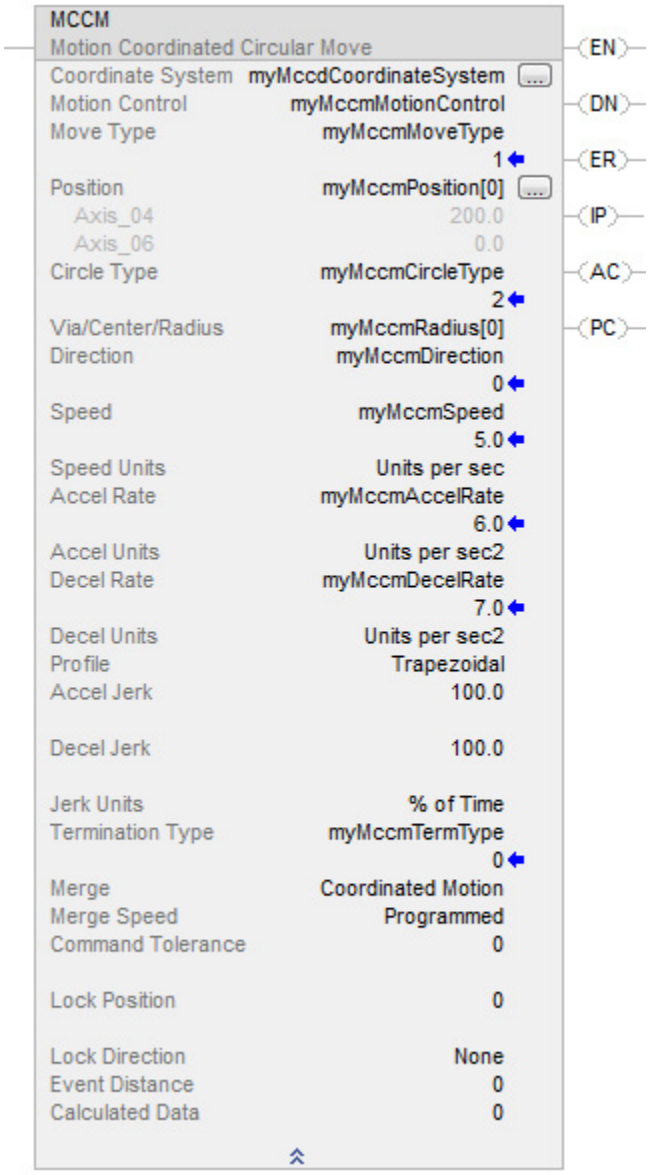
Ladder Diagram



Example 2

Via/Center/Radius parameter as an array type.

Ladder Diagram



Structured Text

MCCM(myMccmCoordinateSystem, myMccmMotionControl, myMccmMoveType, myMccmPosition[0], myMccmCircleType, myMccmRadius[0], myMccmDirection, myMccmSpeed, Unitspersec, myMccmAccelRate, Unitspersec2, myMccmDecelRate, Unitspersec2, Trapezoidal, 100.0, 100.0, %ofTime, myMccmTermType, CoordinatedMotion, Programmed, 0, 0, None, 0, 0);

Motion Coordinated Linear Move (MCLM)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Use the MCLM instruction to start a single or multi-dimensional linear coordinated move for the specified axes within a Cartesian coordinate system. You can define the new position as absolute or incremental.

IMPORTANT Tags used for the motion control attribute of instructions should only be used once. Re-use of the motion control tag in other instructions can cause unintended operation. This may result in damage to equipment or personal injury.

IMPORTANT Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is by changing move dynamics (MCD or MCCD) or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-curve velocity profile can overshoot if:

maximum deceleration is decreased while the move is decelerating or close to the deceleration point; or

maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

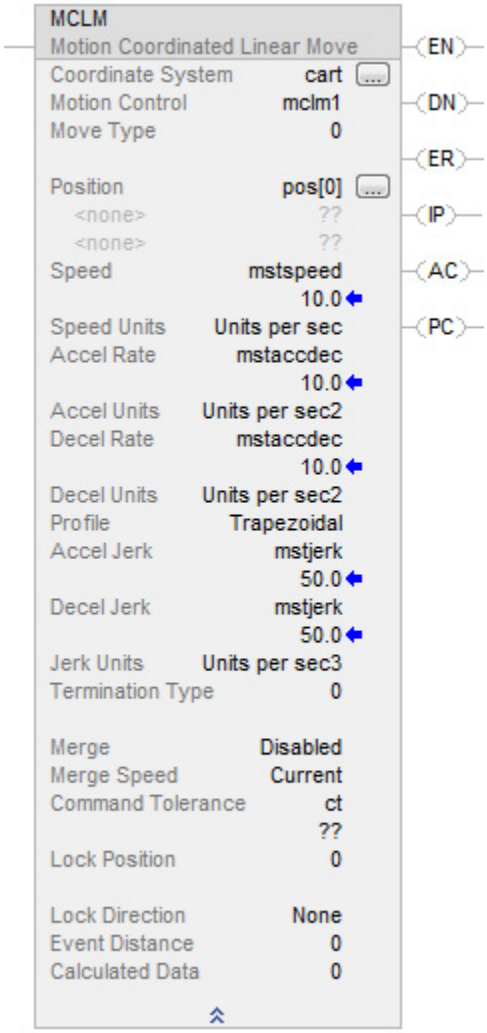
This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MCLM (CoordinateSystem, MotionControl, MoveType, Position, Speed, Speedunits, Accelrate, Accelunits, Decelrate, Decelunits, Profile, Acceljerk, Deceljerk, Jerkunits, TerminationType, Merge, Mergespeed, CommandTolerance,LockPosition, LockDirection, EventDistance, CalculatedData);

Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion.

Ladder Diagram and Structured Text

The Motion Coordinated Linear Move (MCLM) instruction performs a linear move using up to three (3) axes statically coupled as primary axes in a Cartesian coordinate system. You specify whether to use an absolute or incremental target position, the desired speed, maximum acceleration, maximum deceleration, acceleration jerk, deceleration jerk, and the units of each. The actual speed is a function of the programmed units of the speed (Units per sec, or % of Maximum, as configured for the coordinate system), and the combination of primary axes that are commanded to move. Each axis is commanded to move at a speed that allows all axes to reach the programmed endpoint (target position) at the same time.

Operand	Type	Format	Description
Coordinate System	COORDINATE_SYSTEM	Tag	Coordinated group of axes. Reference to coordinated group of axes. For controllers that support the REF_TO motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.
Move Type	SINT, INT, or DINT	Immediate or Tag	Select the Move Type: 0 = Absolute 1 = Incremental
Position	REAL	Array tag [ ]	[coordinate units]
Speed	SINT, INT, DINT, or REAL	Immediate or Tag	[coordinate units]
Speed Units	SINT, INT, or DINT	Immediate	0 = Units per Sec 1 = % of Maximum 4 = Units per MasterUnit
Accel Rate	SINT, INT, DINT, or REAL	Immediate or Tag	[coordinate units]
Accel Units	SINT, INT, or DINT	Immediate	0 = Units per Sec^2 1 = % of Maximum 4 = Units per $MasterUnit^2$
Decel Rate	SINT, INT, DINT, or REAL	Immediate or Tag	[coordinate units]
Decel Units	SINT, INT, or DINT	Immediate	0 = Units per Sec^2 1 = % of Maximum 4 = Units per $MasterUnit^2$
Profile	SINT, INT, or DINT	Immediate	0 = Trapezoidal 1 = S-curve
Accel Jerk	SINT, INT, DINT, or REAL	Immediate or Tag	You must always enter values for the Accel and Decel

Decel Jerk	SINT, INT, DINT, or REAL	Immediate or Tag	Jerk operands. This instruction only uses the values if the Profile operand is configured as S-curve.
Jerk Units	SINT, INT, or DINT	Immediate	Enter the jerk rates in these Jerk Units. 0 = Units per sec^3 1 = % of Maximum 2 = % of Time 4 = Units per MasterUnit^3 6 = % of Time-Master Driven Use these values to get started. Accel Jerk = 100 (% of Time) Decel Jerk = 100 (% of Time) Jerk Units = 2
Termination Type	SINT, INT, or DINT	Immediate or Tag	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 Choose a Termination Type in the Related Topics section below.
Merge	SINT, INT, or DINT	Immediate	0 = Disabled 1 = Coordinated Motion 2 = All Motion
Merge Speed	SINT, INT, or DINT	Immediate	0 = Programmed 1 = Current
Command Tolerance	REAL	Immediate Real or Tag	The position on a coordinated move where blending should start. This parameter is used in place of Command Tolerance in the Coordinate System if Termination Type 6 is used. **Tip**: Termination type 2 is identical to Termination Type 6 except the Command Tolerance value from the coordinate system is used and this parameter is ignored.
Lock Position	REAL	Tag	Position on the Master Axis where a Slave should start following the master after the move has been initiated on the Slave Axis. See the Structure section below for more information.
Lock Direction	UINT32	Immediate	Specifies the conditions when the Lock Position should be used. See the Structure section below for more information.
Event Distance	REAL ARRAY or 0	Array Tag	The position(s) on a move measured from the end of the move. See the Structure section below for more information.
Calculated Data	REAL ARRAY or 0	Array Tag	Master Distance(s) (or time) needed from the beginning of the move to the Event Distance point. See the Structure section below for more information.

When you enter enumerations for the operand value in structured text, multiple word enumerations must be entered without spaces. For example: when entering Decel Units the value should be entered as unitspersec^2 rather than Units per Sec^2 as displayed in the ladder logic.

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Use the entries in this table as a guide when entering structured text operands.

This Operand	Has These Options Which You	
	Enter as Text	**Or as**
Coordinate System	No enumeration	Tag
Motion Control	No enumeration	Tag
Move Type	No enumeration	0 (Absolute) 1 (Incremental)
Position	No enumeration	Array Tag
Speed	No enumeration	Immediate or Tag
Speed Units	unitspersec %ofmaximum unitspermasterunits	0 1 4
Accel Rate	No enumeration	Immediate or Tag
Accel Units	unitspersec^2 %ofmaximum unitspermasterunit^2	0 1 4
Decel Rate	No enumeration	Immediate or Tag
Decel Unit	unitspersec^2 %ofmaximum unitspermasterunit^2	0 1 4
Profile	trapezoidal s-curve	0 1
Accel Jerk	No enumeration	Immediate or tag You must always enter a value for the Accel and Decel Jerk operands. This instruction only uses the values if the Profile is configured as S-curve. Use these values to get started. Accel Jerk = 100 (% of Time) Decel Jerk = 100 (% of Time) Jerk Units = 2
Decel Jerk	No enumeration	
Jerk Units	unitspersec^3 %ofmaximum %oftime unitspermasternit3 %oftimemasterdriven	0 1 2 (use this value to get started) 4 6
Termination Type	No enumeration	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 Choose a Termination Type in the Related Topics section below.
Merge	disabled coordinatedmotion allmotion	0 1 2

Merge Speed	programmed current	0 1
Command Tolerance	No enumeration	Immediate or Tag
Lock Position	No enumeration	Immediate, Real, or Tag
Lock Direction	None immediateforwardonly immediatereverseonly positionforward positionreverse	0 1 2 3 4
Event Distance	No enumeration	Array
Calculated Data	No enumeration	Array

Coordinate System

The Coordinate System operand specifies the set of motion axes that define the dimensions of a Cartesian coordinate system. For this release the coordinate system supports up to three (3) primary axes. Only those axes configured as primary axes are included in the coordinate velocity calculations.

Motion Control

The following control bits are affected by the MCLM instruction.

Mnemonic	Description
.EN (Enable) Bit 31	The Enable bit is set when the rung transitions from false to true and resets when the rung goes from true to false.
.DN (Done) Bit 29	The Done bit sets when the coordinated instruction has been verified and queued successfully. Because it's set at the time it's queued it may appear as set when a runtime error is encountered during the verify operation after it comes out of the queue. It resets when the rung transitions from false to true.
.ER (Error) Bit 28	The Error bit is reset when the rung transitions from false to true. It is set when the coordinated move has not successfully initiated. It is also set with the Done Bit when a queued instruction encounters a runtime error.
.IP (In Process) Bit 26	The In Process bit is set when the coordinated move is successfully initiated. It is reset when there is no succeeding move and the coordinated move reaches the new position, or when there is a succeeding move and the coordinated move reaches the specifications of the termination type, or when the coordinated move is superseded by another MCLM or MCCM instruction with a merge type of Coordinated Move, or when terminated by an MCS instruction.
.AC (Active) Bit 23	When you have a coordinated move instruction queued, the Active bit lets you know which instruction is controlling the motion. It sets when the coordinated move becomes active. It is reset when the Process Complete bit is set or when the instruction is stopped.
.PC (Process Complete) Bit 27	The Process Complete bit is reset when the rung transitions from false to true. It is set when there is no succeeding move and the coordinated move reaches the new position, or when there is a succeeding move and the coordinated move reaches the specified termination type.

.ACCEL (Acceleration Bit) Bit 00	The Acceleration bit sets while the coordinated move is in the acceleration phase. It resets while the coordinated move is in the constant velocity or deceleration phase, or when coordinated motion concludes.
.DECEL (Deceleration Bit) Bit 01	The Deceleration bit sets while the coordinated move is in the deceleration phase. It resets while the coordinated move is in the constant velocity or acceleration phase, or when coordinated motion concludes.
.CalculatedDataAvailable Bit 02	The CalculatedDataAvailable bit is set in response to an Event Distance request in an instruction faceplate parameter for these instructions.
.TrackingMaster Bit 03	The TrackingMaster bit is set when the acceleration is complete in MDSC Mode. This means that the Slave axis is synchronized to the Master Axis. It is reset when any of the following occurs on the slave axis: When the slave axis starts either to accelerate or decelerate for any reason, such as an MCD or a MAS being issued etc. The Slave Axis is linked to another Master axis. For this case the bit is set again in the new instruction status word when the Slave Axis starts tracking the new Master axis again. The slave axis is stopped. The Tracking Master is cleared as soon as the stop is initiated on the slave axis. This bit is never set when Lock Direction== NONE. Note that the Tracking Master bit on the slave is not affected by any operation (i.e. stop, MCD etc on the) master axis. The Tracking Master bit is always reset in Time Driven Mode.

Move Type

The Move Type operand specifies the method used to indicate the coordinated move path. The Move Type can be either Absolute or Incremental.

- Absolute - the axes move via a linear path to the position defined by the position array at the Speed, Accel Rate and Decel Rate as specified by the operands.

When the axis is configured for rotary operation, an Absolute Move type behaves in the same manner as for a linear axis. When the axis position exceeds the Unwind parameter, it is unwound. In this way, axis position is never greater than the Unwind value nor less than zero.

The sign of the specified position is interpreted by the interpolator and can be either positive or negative. Negative position values instruct the interpolator to move the rotary axis in a negative direction to obtain the desired absolute position, while positive values indicate that positive motion is desired to reach the target position. When the position value is greater than the unwind value, an error is generated. The axis never moves through more than one unwind cycle before stopping at an absolute position.

- Incremental - the coordinate system moves the distance as defined by the position array at the specified Speed, using the Accel and Decel rates determined by the respective operands, via a linear path.

The specified distance is interpreted by the interpolator and can be positive or negative. Negative position values instruct the interpolator to move the axis in

a negative direction, while positive values indicate positive motion is desired to reach the target position. Motion greater than one unwind cycle is allowed in Incremental mode.

Position

A one dimensional array, whose dimension is defined to be at least the equivalent of the number of axes specified in the coordinate system. The Position array defines either the new absolute or incremental position.

Speed

The Speed operand defines the maximum vector speed along the path of the coordinated move.

Speed Units

The Speed Units operand defines the units applied to the Speed operand either directly in coordinate units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system.

Accel Rate

The Accel Rate operand defines the maximum acceleration along the path of the coordinated move.

Accel Units

The Accel Units operand defines the units applied to the Accel Rate operand either directly in coordinate units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system.

Decel Rate

The Decel Rate operand defines the maximum deceleration along the path of the coordinated move.

Decel Units

The Decel Units operand defines the units applied to the Decel Rate operand either directly in coordinate units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system.

Profile

The Profile operand determines whether the coordinated move uses a trapezoidal or S-curve velocity profile.

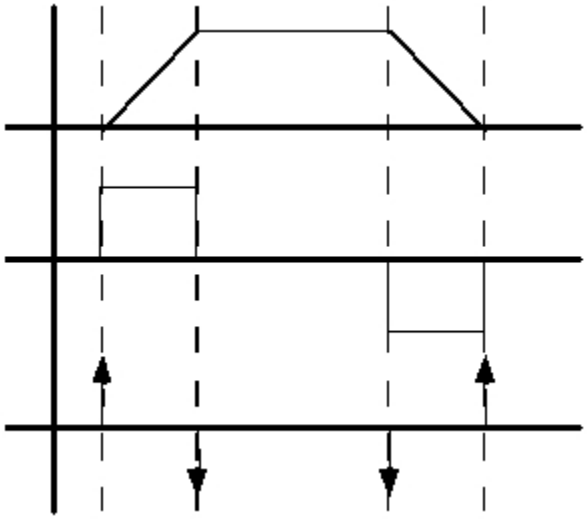
The ControlLogix motion controller provides trapezoidal (linear acceleration and deceleration), and S-curve (controlled jerk) velocity profiles. A guide to the effects of these motion profiles on various application requirements is shown.

Velocity Profile Effects

Profile	**ACC/DEC**	**Motor**	**Priority of control**			
			Highest to Lowest			
Trapezoidal	Fastest	Worse	Acc/Dec	Velocity	Position	
S-Curve	2X Slower	Best	Jerk	Acc/Dec	Velocity	Position

Trapezoidal

The trapezoidal velocity profile is the most commonly used profile since it provides the most flexibility in programming subsequent motion and the fastest acceleration and deceleration times. The maximum change in velocity is specified by acceleration and deceleration. Since jerk is not a factor for trapezoidal profiles, it is considered infinite and is shown as series of vertical lines in this graph.



S-curve

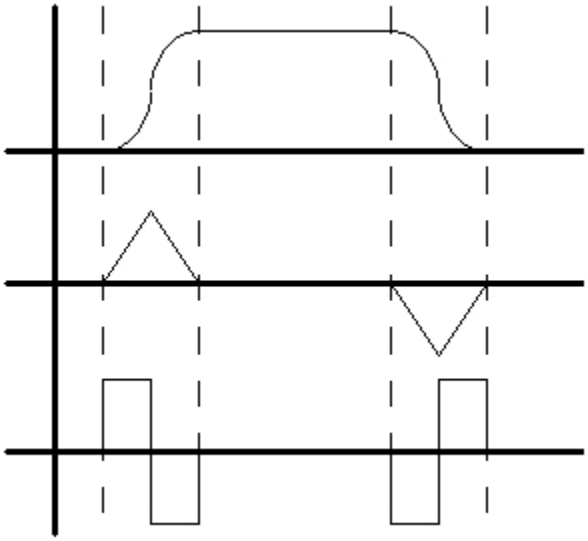
S-curve velocity profiles are most often used when the stress on the mechanical system and load needs to be minimized. The S-curve profile, however, sacrifices acceleration and deceleration time compared to the trapezoidal. The maximum rate at which velocity can accelerate or decelerate is further limited by jerk.

Coordinate motion acceleration and deceleration jerk rate calculations are performed when these instructions are started.

- MAJ
- MAM
- MAS
- MCD
- MCS
- MCCD
- MCCM
- MCLM

The calculated Jerk Rate produces triangular acceleration and deceleration profiles, as shown in this diagram.

S-curve Accel/Decel Time



For an S-curve move, the Jerk rate is determined based on the programmed velocity, acceleration, and deceleration values, not on the length of the move. Logix Designer software attempts to keep the Jerk rate constant when blending moves that have the same acceleration and deceleration values, even though the move may not be long enough to reach the programmed velocity (velocity-limited move).

If an S-curve move is configured as	Then increasing the velocity
Not velocity-limited	Decreases the execution time of the move
Velocity-limited	Increases the execution time of the move

For S-curve moves that are programmed with a zero speed, the Jerk Rate is determined by the rate of speed programmed for the previous instruction with a non-zero speed.

See the MCCD instruction for more details about the impact changes made by an MCCD instruction.

Accel Jerk

Accel Jerk defines the maximum acceleration jerk for the programmed move. For more information on calculating Accel Jerk, refer to the Jerk Units section below.

Decel Jerk

Decel Jerk defines the maximum deceleration jerk for the programmed move. For more information on calculating Decel Jerk, refer to the Jerk Units section below.

Jerk Units

The jerk units define the units that are applied to the values entered in the Accel Jerk and Decel Jerk operands. The values are entered directly in the position units of the specified coordinate system or as a percentage. When configured using % of Maximum, the jerk is applied as a percentage of the Maximum Acceleration Jerk and Maximum Deceleration Jerk operands specified in the coordinate system attributes. When configured using % of Time, the value is a percentage based on the Speed, Accel Rate, and Decel Rate specified in the instruction.

Termination Type

For Master Driven Speed Control (MDSC), when all sequential instructions run in the same mode (Master Driven Mode or Time Driven Mode), then all termination types are supported. If the termination type switches in the coordinated motion queue, errors may generate depending on the sequence of motion types.

The following is only applicable if a move on a slave Coordinate System uses a Blending Termination Type (Termination Types 2, 3, or 6) and is programmed in MDSC mode.

If you use the Calculated Data returned in the last MCLM instruction of a motion sequence to program the length that the master axis has to move for the motion sequence to go PC, then there is the possibility that you will have to add a small safety margin to the Calculated Data. If you do not add this margin, there is a chance that the motion of the master axes completes before the entire motion sequence programmed on the Slave Coordinate System finishes. If this occurs, the last motion instruction on the Slave Coordinate System remains active and does not go PC. The value of the small safety margin is dependent on the Command Tolerance used for the first and last move in the motion sequence as shown:

Tip: Safety margin calculations use the following abbreviations:
CUP = Coarse Update Period
MAS = Master Axis Speed

1. If a Command Tolerance value of 100% is used for the first move in the sequence then:

   SafetyMargin1 = CUP * MAS

else
SafetyMargin1 = CUP * MAS * .02

2. For all other moves in the blending sequence between first and last:

   SafetyMargin2 = CUP * MAS * .02 * number of blending moves between 1st and last

3. If a Command Tolerance value of 100% is used for the last in the sequence then:

   SafetyMargin3 = CUP * MAS
   else
   SafetyMargin3 = CUP * MAS * .02

4. Final SafetyMargin = SafetyMargin1 + SafetyMargin2 + SafetyMargin3

   Once a sequence is programmed and verified, it repeats.

Refer to the See also section for more information.

Merge

The Merge operand determines whether or not to turn the motion of all specified axes into a pure coordinated move. The Merge options include: Merge Disabled, Coordinated Motion, or All Motion.

- Merge Disabled

  Any currently executing single axis motion instructions involving any axes defined in the specified coordinate system are not affected by the activation of this instruction, and results in superimposed motion on the affected axes. Also, any coordinated motion instructions involving the same specified coordinate system runs to completion based on its termination type.

- Coordinated Motion

  Any currently executing coordinated motion instructions involving the same specified coordinate system are terminated. The active motion is blended into the current move at the speed defined in the merge speed parameter. Any pending coordinated motion instructions are cancelled. Any currently executing system single axis motion instructions involving any axes defined in the specified coordinate system will not be affected by the activation of this instruction, and will result in superimposed motion on the affected axes.

- All Motion

  Any currently executing single axis motion instructions involving any axes defined in the specified coordinate system and any currently executing coordinated motion instructions are terminated. The prior motion is merged into the current move at the speed defined in Merge

Speed parameter. Any pending coordinated move instructions are cancelled.

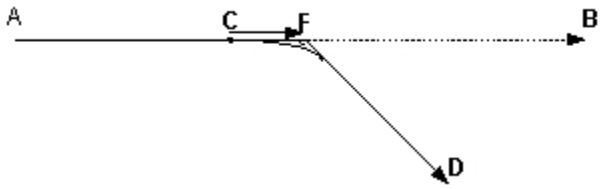
Merge Speed

The Merge Speed operand defines whether the current speed or the programmed speed is used as the maximum speed along the path of the coordinated move when Merge is enabled.

Currently, Coordinated Motion only supports the queueing of one coordinated motion instruction. Therefore the MovePendingStatus bit and the MovePendingQueueFullStatus bit are always the same.

Additional Information On Merging Instructions

A move from point A to point B is initiated as shown in the figure below. When the axis is at point C, a incremental merge to point D is initiated. As a result the current instruction is terminated at point C. The control computes the deceleration distance needed at point C along the vector AB from the current velocity to zero velocity. This distance is shown as vector CF. The imaginary point F is then calculated by adding the vector CF to point C. The resultant merged motion from C to D is shown in the illustration below. The motion follows the curved line starting from C then joins the straight line from F to D. Point D is calculated from the original point of the merge (point C) using the incremental data in the merge instruction. This path is identical as if the original programmed move was made from point A to F then from F to D with a termination type of No Decel.



This example applies to linear merges.

Attempting to merge a circular move can result in programming errors if the resultant path does not define a circle. The circle center in incremental mode is calculated from point C (the point of the merge). However, a circle must exist from point F (the calculated end of the deceleration) to the end of the merged move.

Merging in Incremental Mode

The Merge for coordinated motion operates differently from a merge on an MAM. For the MCLM, any uncompleted motion at the point of the merge is discarded. For example, assume that you have a single axis MCLM programmed in incremental mode from a starting absolute position = 0 and with the programmed incremental distance = 4 units. If a merge occurs at an absolute position of 1 and the merge is another incremental move of 4 units, the move completes at a position = 5.

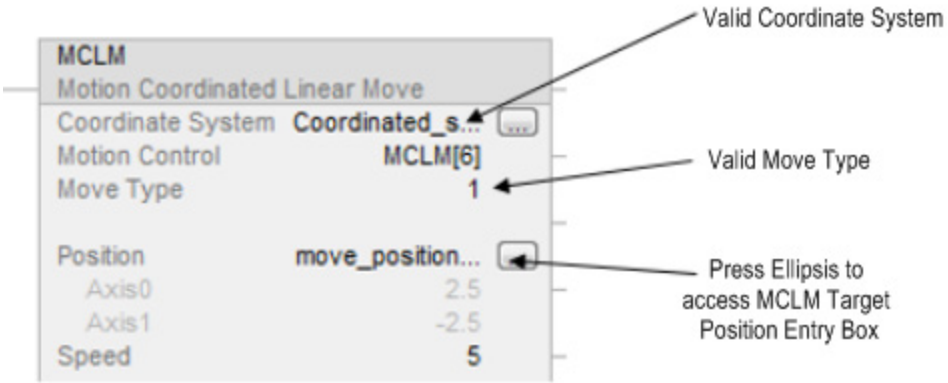
If this example occurs on a MAM programmed in incremental mode, the final position = 8.

MCLM Target Position Entry Dialog

The Target Position Entry Dialog for the MCLM instruction provides an easy format for editing Position. To gain access to the Target Position Entry dialog box you must have inserted the name of the coordinated system into the instruction, you must have a valid tag name entered in the position field with sufficient elements to handle the number of axes, and you must have selected a valid Move Type.

To access the MCLM Instruction Target Position Entry Dialog box, press the ellipsis after the Position line on the instruction faceplate.

MCLM Ladder Valid Values for Accessing Target Position Entry Box



Pressing the ellipsis button at the Position line of the ladder instruction faceplate calls the **Target Position** tab for editing the position values.

Tip: This action is not possible for a reference.

The dialog title indicates the Coordinate System and Tag Names for the instruction.

Feature	Description
Axis Name	These fields list the names of each axis contained in the Coordinate System. You cannot alter the axis names in this dialog.
Target Position/Target Increment	This field contains the endpoint or increment of the coordinated move as specified in the instruction faceplate. It is numeric.
Actual Position	These are the current actual positions of the axes in the coordinate system. These positions are updated dynamically when on-line and Coordinate System Auto Tag Update is enabled.
Set Targets = Actuals Button	When in Absolute mode, this button automatically copies the actual position values to the Target Position Column. When in Incremental mode, this button automatically copies the actual position values to the Target Increment Column.

The selected Move type governs the appearance and the availability of the Set Targets = Actuals button.

When the Move Type is Absolute, the target column is entitled Target Position and when the Move Type is Incremental, the target column is entitled Target Increment and the Set Targets = Actuals button appears dimmed.

MCLM is a transitional instruction:

- In relay ladder, toggle the Rung-condition-in from false to true each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition. See Structured Text Syntax.

Structure

See *Input and Output Parameters Structure for Single Axis Motion Instructions* for the input and output parameters that are available for the MCLM instruction via the Master Driven Speed Control (MDSC) function. Before any of these parameters is active, you must execute an MDCC instruction and it must be active (IP bit is set).

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction.LA major fault can occur if an uninitialized reference or a reference of the incorrect type is passed to the Coordinate System operands. See Common Attributes for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Runtime Error Conditions

You cannot switch from Time Driven Mode to Master Driven Mode if the master speed is zero unless the slave speed is also zero.

Extended Error Codes

Extended Error codes help to further define the error message given for this particular instruction. Their behavior is dependent upon the Error Code with which they are associated.

The Extended Error Codes for Servo Off State (5), Shutdown State (7), Axis Type Not Servo (8), Axis Not Configured (11), Homing In Process Error (16), and Illegal Axis Data type (38) errors all function in the same fashion. A number between 0...n is displayed for the Extended Error Code. This number is the index to the Coordinate System indicating the axis that is in the error condition.

For Error Code Axis Not Configured (11) there is an additional value of -1 which indicates that Coordinate System was unable to setup the axis for coordinate motion. See Motion Error Codes (ERR) for Motion Instructions.

For the MCLM instruction, Error Code 13 - Parameter Out of Range, Extended Errors return a number that indicates the offending parameter as listed on the faceplate in numerical order from top to bottom beginning with zero. For example, 2 indicates the parameter value for Move Type is in error.

Referenced Error Code and Number	Extended Error Numeric Indicator	Instruction Parameter	Description
Parameter Out Of Range (13)	2	Move Type	Move Type is either less than 0 or greater than 1.
Parameter Out Of Range (13)	3	Position	The position array is not large enough to provide positions for all the axes in the coordinate system.
Parameter Out Of Range (13)	4	Speed	Speed is less than 0.
Parameter Out Of Range (13)	6	Accel Rate	Accel Rate is less than or equal to 0.
Parameter Out Of Range (13)	8	Decal Rate	Decel Rate is less than or equal to 0.
Parameter Out Of Range (13)	11	Termination Type	Termination Type is less than 0 or greater than 3.

For the Error Code 54 – Maximum Deceleration Value is Zero, if the Extended Error returns a positive number (0-n) it's referring to the offending axis in the coordinate system. Go to the Coordinate System Properties General Tab and look under the Brackets ([ ])column of the Axis Grid to determine which axis has a Maximum Deceleration value of 0. Click on the ellipsis button next to the offending axis to access the Axis Properties screen. Go to the Dynamics tab and make the appropriate change to the Maximum Deceleration Value. If the Extended Error number is -1, this means the Coordinate System has a Maximum Deceleration Value of 0. Go to the Coordinate System Properties Dynamics Tab to correct the Maximum Deceleration value.

MCLM Changes to Status Bits

Status bits provide a means for monitoring the progress of the motion instruction. There are three types of Status bits that provide pertinent information.

- Axis Status bits
- Coordinate System Status bits
- Coordinate Motion Status bits

When the MCLM instruction initiates, the status bits undergo the following changes.

Motion Instruction Predefined Data Type Status Bits

See *Status Bits for Motion Instructions* (MCLM, MCCM) when MDCC Is Active

Axis Status Bits

Bit Name	Meaning
CoordinatedMotionStatus	Sets when the instruction starts. Clears when the instruction ends.

Coordinate System Status Bits

Bit Name	Meaning
MotionStatus	Sets when the MCLM instruction is active and the Coordinate System is connected to its associated axes.

Coordinated Motion Status Bits

Bit Name	Meaning
AccelStatus	Sets when vector is accelerating. Clears when a blend is in process or when vector move is decelerating.
DecelStatus	Sets when vector is decelerating. Clears when a blend is in process or when vector move is accelerating.
ActualPosToleranceStatus	Sets for Actual Tolerance termination type only. It sets after the following two conditions are met. 1) Interpolation is complete. 2) The actual distance to the programmed endpoint is less than the configured coordinate system Actual Tolerance value. The bit remains set after an instruction completes. The bit is reset when a new instruction is started.
CommandPosToleranceStatus	Sets for all termination types whenever the distance to the programmed endpoint is less than the configured coordinate system Command Tolerance value. The bit remains set after an instruction completes. It resets when a new instruction is started. The CommandPosToleranceStatus (CS_CMD_POS_TOL_STS) status bit in the Coordinate System is set as follows: TT0, TT1, TT4, TT5 - Bit is set when the distance to the endpoint is less than the Command Tolerance value. The bit is cleared when the first move is complete. TT2, TT6 - Bit is set when the distance to the endpoint is less than the Command Tolerance value. The bit is cleared when the blend is started (that is, when the second move is started). Thus, you may not see the bit if the blend is started at the Command Tolerance (CT) point. The blend may have been deferred slightly beyond the CT point if the next move is a short move or for time matching of the acceleration and deceleration of the two adjacent moves. TT3 - Bit is set when the distance to the endpoint is less than the Command Tolerance value (like TT2 and TT6). The bit is cleared when the blend is started. Thus, you may not see the bit if the blend is started at the deceleration point. The blend may have been deferred slightly beyond the deceleration point if the next move is a short move or for time matching of the acceleration and deceleration of the two adjacent moves.
StoppingStatus	The Stopping Status bit is cleared when the MCLM instruction initiates.
MoveStatus	Sets when MCLM begins axis motion. Clears on .PC bit of the last motion instruction or when a motion instruction executes which causes a stop.
MoveTransitionStatus	Sets when No Decel or Command Tolerance termination type is satisfied. When blending collinear moves the bit is not set because the machine is always on path. It clears when a blend completes, the motion of a pending instruction starts, or a motion instruction executes which causes a stop. Indicates not on path.
MovePendingQueueFullStatus	Sets when the instruction queue is full. It clears when the queue has room for a new coordinated motion instruction.

CoordinateSystemLockStatus	Set when an axis lock is requested for an MCLM or MCCM instruction and the axis has crossed the Lock Position. Cleared when an MCLM or MCCM is initiated. For the enumerations Immediate Forward Only and Immediate Reverse Only, the bit is set immediately when the MCLM or MCCM is initiated. When the enumeration is Position Forward Only or Position Reverse Only, the bit is set when the Master Axis crosses the Lock Position in the specified direction. The bit is never set if the enumeration is NONE. The CoordMotionLockStatus bit is cleared when the Master Axis reverses direction and the Slave Axis stops following the Master Axis. The CoordMotionLockStatus bit is set again when the Slave Coordinate System resumes following the Master Axis. The CoordMotionLockStatus bit is also cleared when an MCCS is initiated.

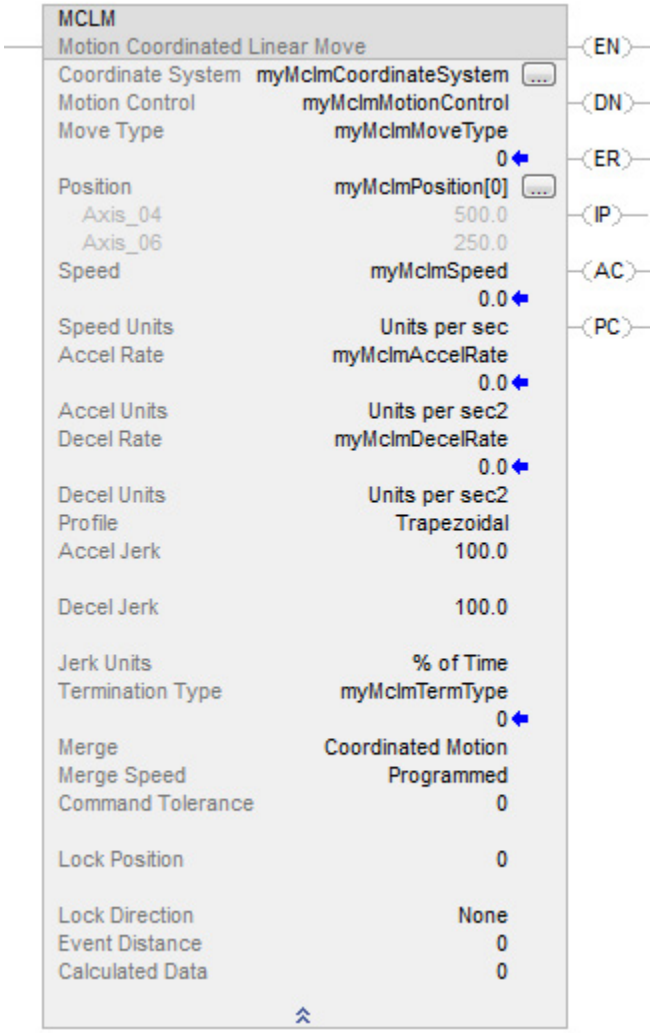
Currently, Coordinated Motion only supports the queueing of one coordinated motion instruction. Therefore the MovePendingStatus bit and the MovePendingQueueFullStatus bit are always the same.

Master Driven Speed Control (MDSC) and Motion Direct Command Support

The Motion Direct commands are not available in the instruction tree for the MCLM instruction. You must program an MCLM in one of the supported programming languages before you execute an MAM or MAJ in Time Driven Mode. A runtime error will occur if an MCLM is not previously executed in an MAM and MAJ in Master Driven Mode.

Examples

Ladder Diagram



Structured Text

MCLM(myMclmCoordinateSystem, myMclmMotionControl, myMclmMoveType, myMclmPosition[0], myMclmSpeed, Unitspersec, myMclmAccelRate, Unitspersec2, myMclmDecelRate, Unitspersec2, Trapezoidal, 100.0, 100.0, %ofTime, myMclmTermType, CoordinatedMotion, Programmed, 0, 0, None, 0, 0);

See also

Structured Text Syntax on page 665

Motion Error Codes (.ERR) on page 581

Multi-Axis Coordinated Motion Instructions on page 359

Common Attributes on page 691

Data Conversions on page 697

Motion Coordinated Shutdown (MCSD)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Use the Motion Coordinated Shutdown (MCSD) instruction to perform a controlled shutdown of all the axes in the named coordinate system.

IMPORTANT Tags used for the motion control attribute of instructions should only be used once. Re-use of the motion control tag in other instructions can cause unintended operation. This may result in damage to equipment or personal injury.

IMPORTANT Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is by changing move dynamics (MCD or MCCD) or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-curve velocity profile can overshoot if:

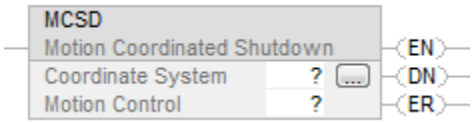
maximum deceleration is decreased while the move is decelerating or close to the deceleration point; or

maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MCSD(CoordinateSystem, MotionControl);

Operands

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Coordinate System	COORDINATE_SYSTEM	Tag	Coordinated group of axes. Reference to the coordinated group of axes. For controllers that support the REF_TO Motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Coordinate System

The Coordinate System operand specifies the set of motion axes that define the dimensions of a Cartesian coordinate system. For this release the coordinate system supports up to three (3) primary axes. Only the axes configured as primary axes (up to 3) are included in the coordinate velocity calculations.

Motion Control

The following control bits are affected by the MCSD instruction.

Mnemonic	Description
.EN (Enable) Bit 31	The Enable bit sets when the rung transitions from false to true. It resets when the rung goes from true to false.
.DN (Done) Bit 29	The Done bit sets when the coordinated shutdown is successfully initiated. It resets when the rung transitions from false to true.
.ER (Error) Bit 28	The Error bit sets when the coordinated shutdown fails to initiate successfully. It resets when the rung transitions from false to true.

MCSD is a transitional instruction:

- In relay ladder, toggle the Rung-condition-in from cleared to set each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition. See Structured Text Syntax.

Master Driven Speed Control (MDSC) and the MCSD Instruction

When the coordinate system is shut down:

- The IP bit of the Master Driven Coordinate Control (MDCC) instruction is reset on an axis that is shutdown.
- The AC bit of the MDCC instruction resets when the axis is stopped as it is shutdown.
- The MCSD instruction clears the pending Master Axis for all future coordinate system motion instructions.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of the incorrect type is passed to the Coordinate System operand. See *Common Attributes* operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.

Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

MCSD Changes to Status Bits

Status bits provide a means for monitoring the progress of the motion instruction. There are three types of Status bits that provide pertinent information. They are: Axis Status bits, Coordinate System Status bits, and Coordinate Motion Status bits. When the MCS instruction initiates, the status bits undergo these changes.

Axis Status Bits

Bit Name	Meaning
CoordinatedMoveStatus	Cleared

Coordinate System Status Bits

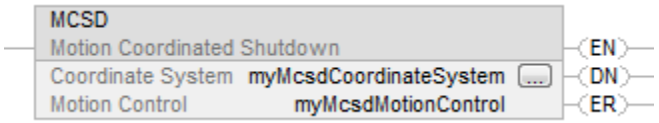
Bit Name	Meaning
ShutdownStatus	Sets when MCSD is executed and all associated axes are shutdown.
ReadyStatus	Cleared after MCSD executes.

Coordinated Motion Status Bits

Bit Name	Meaning
AccelStatus	Cleared after MCSD executes.
DecelStatus	Cleared after MCSD executes.
ActualPosToleranceStatus	Cleared after MCSD executes.
CommandPosToleranceStatus	Cleared after MCSD executes.
StoppingStatus	Cleared after MCSD executes.
MoveStatus	Cleared after MCSD executes.
MoveTransitionStatus	Cleared after MCSD executes.
MovePendingStatus	Cleared after MCSD executes.
MovePendingQueueFullStatus	Cleared after MCSD executes.

Examples

Ladder Diagram



Structured Text

MCSD(myMcsdCoordinateSystem,myMcsdMotionControl);

See also

Common Action Table for Slave and Master Axis on page 552

Structured Text Syntax on page 665

Motion Error Codes (.ERR) on page 581

Multi-Axis Coordinated Motion Instructions on page 359

Common Attributes on page 691

Motion Coordinated Shutdown Reset (MCSR)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

Use the Motion Coordinated Shutdown Reset (MCSR) instruction to reset all axes in a coordinate system. The MCSR instruction resets the axes from a shutdown state to an axis ready state. This instruction also clears any axis faults.

IMPORTANT Tags used for the motion control attribute of instructions should only be used once. Re-use of the motion control tag in other instructions can cause unintended operation. This may result in damage to equipment or personal injury.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram

Function Block

This instruction is not available in function block.

Structured Text

MCSR(CoordinateSystem, MotionControl);

Operands

Ladder Diagram and Structured Text

Operand	Type	Format	Description
Coordinate System	COORDINATE_SYSTEM	Tag	Coordinate System on which to perform the Shutdown Reset. Reference to Coordinate System on which to perform the Shutdown Reset. For controllers that support the REF_TO Motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Coordinate System

The Coordinate System operand specifies the set of motion axes that define the dimensions of a Cartesian coordinate system. For this release the coordinate system supports up to three (3) primary axes. Only the axes configured as primary axes (up to 3) are included in the coordinate velocity calculations.

Motion Control

These control bits are affected by the MCSR instruction.

Mnemonic	Description
.EN (Enable) Bit 31	The Enable bit is set when the rung transitions from false to true. It resets when the rung transitions from true to false.
.DN (Done) Bit 29	The Done bit sets when the coordinated shutdown reset is successfully initiated. It resets when the rung transitions from true to false.
.ER (Error) Bit 28	The Error bit sets when the reset of the coordinated shutdown fails to initiate. It resets when the rung transitions from false to true.

This is a transitional instruction:

- In relay ladder, toggle the Rung-condition-in from false to true each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction.A major fault can occur if an uninitialized reference or a reference of the incorrect type is passed to the Coordinate System operands. See Common Attributes in operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.

Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

MCSR Changes to Status Bits:

Status Bits provide a means for monitoring the progress of the motion instruction. There are three types of Status bits that provide pertinent information. They are: Axis Status bits, Coordinate System Status bits, and Coordinate Motion Status bits. When the MCS instruction initiates, the status bits undergo these changes.

Axis Status Bits

Bit Name	Meaning
CoordinatedMotionStatus	No effect

Coordinate System Status Bits

Bit Name	Meaning
ShutdownStatus	Clears the Shutdown status bit.

Coordinated Motion Status Bits

Bit Name	Meaning
MovePendingStatus	Flushes instruction queue and clears status bit.
MovePendingQueueFullStatus	Flushes instruction queue an clears status bit.

Examples

Ladder Diagram

Structured Text

MCSR(myMcsrCoordinateSystem,myMcsrMotionControl);

See also

Structured Text Syntax on page 665

Motion Error Codes (.ERR) on page 581

Multi-Axis Coordinated Motion Instructions on page 359

Common Attributes on page 691

Motion Coordinated Stop (MCS)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

The Motion Coordinated Stop (MCS) instruction initiates a controlled stop of coordinated motion. Any pending motion profiles are canceled.

IMPORTANT Tags used for the motion control attribute of instructions should only be used once. Re-use of the motion control tag in other instructions can cause unintended operation. This may result in damage to equipment or personal injury.

IMPORTANT Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is by changing move dynamics (MCD or MCCD) or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

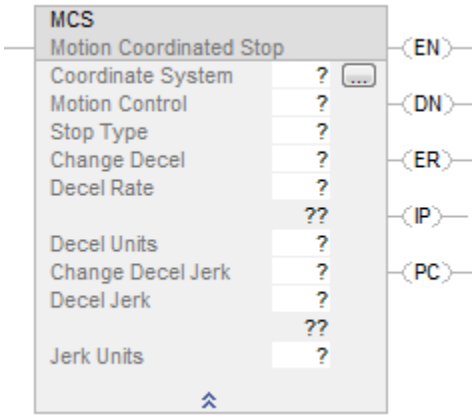
An S-curve velocity profile can overshoot if:

- maximum deceleration is decreased while the move is decelerating or close to the deceleration point; or
- maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MCS(CoordinateSystem, MotionControl,StopType, ChangeDecel, DecelRate,DecelUnits, ChangeDecelJerk,DecelJerk, JerkUnits);

Operands

There are data conversion rules for mixed data types within an instruction. See Data Conversion.

Ladder Diagram and Structured Text

Operand	Type	Format	Description	
Coordinate System	COORDINATE_SYSTEM	Tag	Name of the coordinate system. Reference to the coordinate system. For controllers that support the REF_TO Motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.	
Motion Control	MOTION_INSTRUCTION	Tag	Control tag for the instruction.	
Stop Type	DINT	Immediate	**If you want to**	**Choose this Stop Type**

			Stop all motion for the axes of the coordinate system and stop any transform that the coordinate system is a part of.	All (0) - For each axis, all motion generators, including the coordinated motion, are taken into account when computing the initial dynamics (i.e., acceleration rate and velocity) to be used in the Decel. Every axis in the coordinated system is stopped independently using the computed initial dynamics.
			Stop only coordinated moves.	Coordinated Move (2)
			Cancel any transform that the coordinate system is a part of.	Coordinated Transform (3)
Change Decel(1)	DINT	Immediate	If you want to	Then Choose
			Use the maximum deceleration rate of the coordinate system.	No (0)
			Specify the deceleration rate.	Yes (1)
Decel Rate	REAL	Immediate or Tag	**Important:** An axis could overshoot its target position if you reduce the deceleration while a move is in process. Deceleration along the path of the coordinated move. The instruction uses this value: • Only if Change Decel is Yes. • Only for coordinated moves. Enter a value greater than 0.	
Decel Units	DINT	Immediate	0 = Units per Sec^2 1 = % of Maximum Only "% of Maximum" is allowed on Cartesian geometries with coordinate definition = XYZRxRyRz. % of Cartesian max is used for XYZ and "% of orientation max" is used for Rx, Ry and Rz.	
Change Decel Jerk	SINT, INT, or DINT	Immediate	0 = No 1 = Yes	
Decel Jerk	SINT, INT, DINT, or REAL	Immediate or Tag	You must always enter a value for the Decel Jerk operand. This instruction only uses the value if the Profile is configured as S-curve. Decel Jerk is the deceleration jerk rate for the coordinate system. Use these values to get started. • Decel Jerk = 100 (% of Time) • Jerk Units = 2	

Jerk Units	SINT, INT, or DINT	Immediate	0 = Units per sec^3 1 = % of Maximum 2 = % of Time (use this value to get started) Only "% of Time" is allowed on Cartesian geometries with coordinate definition = XYZRxRyRz.

(1) Overshoot may occur if MCS is executed close to or beyond the deceleration point and the deceleration limit is decreased. Keep in mind that deceleration may be decreased indirectly by setting ChageDecel to NO if configured maximum deceleration rate is less than the active deceleration rate.

Structured Text

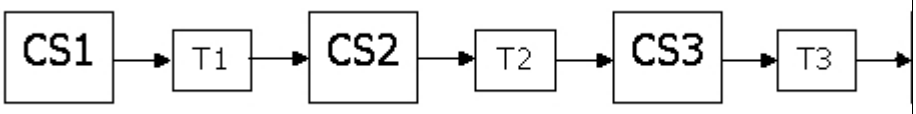
Enter the stop type and decel units without spaces.

See Structured Text Syntax for more information on the syntax of expressions within structured text.

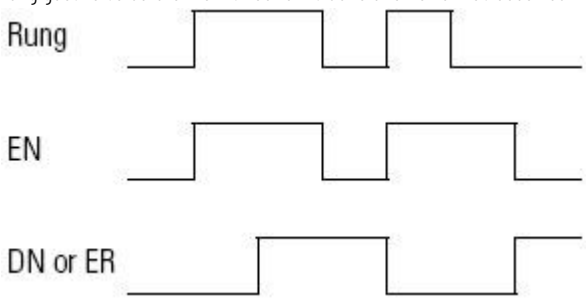
For example, enter the Coordinate System operand as CoordinateSystem.

How Stop Types Affect Transforms

This table describes how the stop types affect coordinate systems that are a part of a transform.

This Operand	Description
All	This stop type: • stops the axes in the specified coordinate system. It also stops the axes of any coordinate system that shares axes with this coordinate system. • cancels any transforms that the coordinate system is a part of.
Coordinate Move	This stop type stops only the coordinated moves. Any transforms stay active.
Coordinated Transform	This stop type cancels the transforms associated with the specified coordinate system. All transform-related motion stops on all associated target coordinate systems. However, source coordinate axes will continue to move as instructed. Example If four coordinate systems are linked via three transforms. And the first coordinate system (CS1) is the source and is processing commanded motion.  Executing an MCS instruction on CS2 and using a stop type of coordinated transform results in: • Transforms T1 and T2 are canceled. • Transform T3 stays active. • the axes in CS1 stay in motion. • the axes in Coordinate Systems CS2 and CS3 stop via the deceleration rate selected in the MCS instruction or the maximum coordinate deceleration rate. • the axes in CS4 follow the respective CS3 axes. In an Motion Axis Stop (MAS) instruction, a stop type of all also cancels transforms.

MOTION_INSTRUCTION Data Type

To see if	Check if this bit is on	Data Type	Notes
The rung is true	EN	BOOL	Sometimes the EN bit stays on even if the rung goes false. This happens if the rung goes false before the instruction is done or an error has occurred. 
The stop was successfully initiated	DN	BOOL	
An error happened	ER	BOOL	
The axis is stopping	IP	BOOL	Any of these actions ends the MCS instruction and turns off the IP bit: • The coordinate system is stopped. • Another MCS instruction supersedes this MCS instruction. • Shutdown instruction. • Fault Action.
The axis is stopped	PC	BOOL	The PC bit stays on until the rung makes a false-to-true transition.

Master Driven Speed Control (MDSC) and the MCS Instruction

If an MCS is issued when in Master Driven Mode, a switch is made to Time Driven Mode and the axes are stopped in Time Driven Mode. MCS All resets the IP bit of the Master Driven Coordinate Control (MDCC) instruction. Other stop types do not reset the IP bit.

The MCS All instruction clears the pending Master Axis for all future coordinated motion instructions. However, MCS ALL on the Master axis does not break the MDSC link.

The AC bit of the MDCC instruction is reset when the axis is stopped.

The instruction queue is cleared when an MCS All or MCS Coordinated is executed (goes IP).

The status bit CalculatedDataAvailable in an active motion instruction status word for an MCLM or MCCM instruction clears when an MCS is executed (goes IP). The CalculatedData is not recomputed.

Note that if a stop is issued very close to the programmed endpoint, the actual stop may be beyond the programmed endpoint, especially if run in Master Driven Mode.

MCS is a transitional instruction:

- In relay ladder, toggle the Rung-condition-in from false to true each time the instruction should execute.
- In structured text, condition the instruction so that it only executes on a transition.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction.A major fault can occur if an uninitialized reference or a reference of the incorrect type is passed to the Coordinate System operands. See Common Attributes for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes:

See Motion Error Codes (.ERR) for motion instructions.

Extended Error Codes

See Extended Error Codes for Motion Instructions. It has information about how to use the extended error codes. See Motion Error Codes (.ERR) for motion instructions.

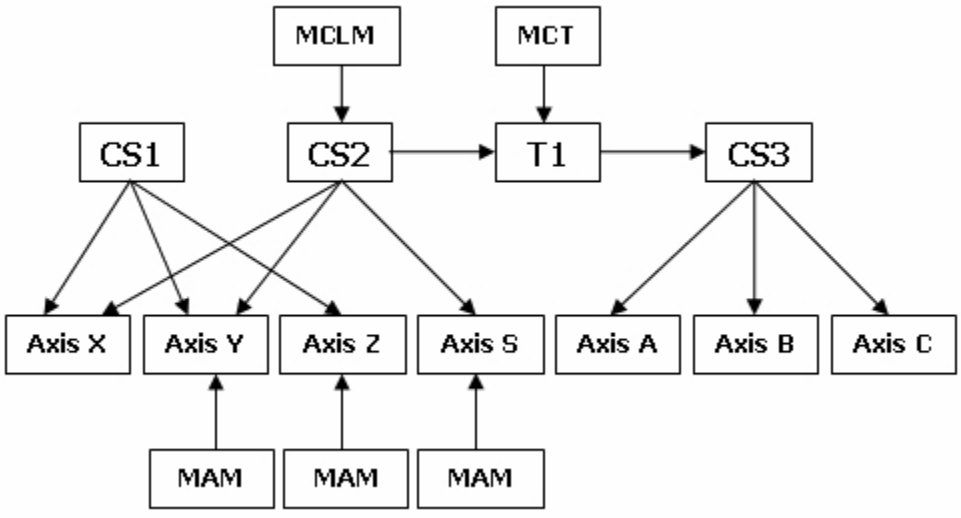
Changes to Status Bits

The instruction changes these status bits when it executes.

In the tag for the	This bit	When the stop type is	Turns
Axis	CoordinatedMotionStatus	→	Off when the coordinated move stops
	TransformStatus	Coordinated Move	Unchanged
		• All • Coordinated Transform	Off
	ControlledByTransformStatus	Coordinated Move	Off when the axes stop and the PC bit of the MCS instruction turns on
		• All • Coordinated Transform	Off
Coordinate System	MotionStatus	→	Off when the coordinated move stops
	AccelStatus	→	Off
	DecelStatus	→	On during the stop and then off when the stop completes
	StoppingStatus	→	On during the stop and then off when the PC bit turns on
	MoveStatus	→	Off
	MoveTransitionStatus	→	Off
	TransformSourceStatus	Coordinated Move	Unchanged
		• All • Coordinated Transform	Off
	TransformTargetStatus	Coordinated Move	Unchanged
		• All • Coordinated Transform	Off

How Stop Types Affect Transforms an Axis Motion Example

Suppose you have this situation.



Where:

- Coordinate system 1 (CS1) contains the X, Y, and Z axes.
- Coordinate system 2 (CS2) contains the Y, Z, and S axes.
- Coordinate system 3 (CS3) contains the A, B, and C axes.
- Transform (T1) links source coordinate CS2 to target CS3.
- CS2 (XYS) axes are mapped to CS3 (ABC) axes.
- MAM instructions executed on the Y, Z, and S axes.
- MCLM instruction executed on CS2.
- MCT instruction executed with CS2 as the source and CS3 as the target.
- No coordinate instructions were executed on CS2 or CS3.

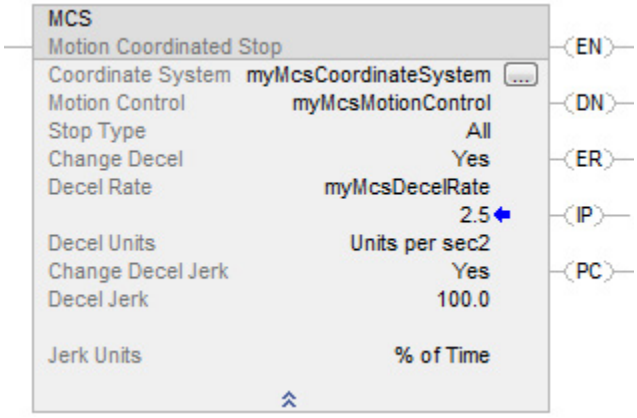
This table shows the results of executing various MCS and MAS instructions with different stop types.

Instruction	Stop Type	Result
MCS on CS1	All	The MCLM instruction on CS2 will stop.
		The MAM on Y will stop.
		The MAM on Z will stop.
		The MAM on S will continue.
		T1 is canceled.
		Axes ABC will stop due to canceling the transform.
MCS on CS2	All	The MCLM instruction on CS2 will stop.
		The MAM on Y will stop.
		The MAM on S will stop.
		The MAM on Z will continue.
		T1 is canceled.
		Axes ABC will stop due to canceling the transform.
MCS on CS3	All	The MCLM instruction on CS2 will continue.
		The MAM on Y will continue.
		The MAM on S will continue.
		The MAM on Z will continue.
		T1 is canceled.
		Axes ABC will stop due to canceling the transform.
MCS on CS1	Coordinated Move	The MCLM instruction on CS2 will continue.
		The MAM on Y will continue.
		The MAM on S will continue.
		The MAM on Z will continue.
		T1 stays active.
		Axes ABC will follow the respective CS2 axes.
MCS on CS2	Coordinated Move	The MCLM instruction on CS2 will stop.
		The MAM on Y will continue.
		The MAM on S will continue.
		The MAM on Z will continue.
		T1 stays active.
		Axes ABC will follow the respective CS2 axes.
MCS on CS3	Coordinated Move	The MCLM instruction on CS3 will stop.
		The MAM on Y will continue.
		The MAM on S will continue.

		The MAM on Z will continue.
		T1 stays active.
		Axes ABC will follow the respective CS2 axes.
MAS on Y	All	The MCLM instruction on CS2 will stop.
		The MAM on Y will stop.
		The MAM on S will continue.
		The MAM on Z will continue.
		T1 is canceled.
		Axes ABC will stop due to canceling the transform.
MAS on Y	Move	The MCLM instruction on CS2 will continue.
		The MAM on Y will stop.
		The MAM on S will continue.
		The MAM on Z will continue.
		T1 stays active.
		Axes ABC will follow the respective CS2 axes.
MAS on Z	All	The MCLM instruction on CS2 will continue.
		The MAM on Y will continue.
		The MAM on S will continue.
		The MAM on Z will stop.
		T1 stays active.
		Axes ABC will follow the respective CS2 axes.
MAS on Z	Move	The MCLM instruction on CS2 will continue.
		The MAM on Y will continue.
		The MAM on S will continue.
		The MAM on Z will stop.
		T1 stays active.
		Axes ABC will follow the respective CS2 axes.
MCS on CS1	Coordinated Transform	The MCLM instruction on CS2 will continue
		The MAM on Y will continue.
		The MAM on S will continue.
		The MAM on Z will continue.
		T1 stays active.
		Axes ABC will follow the respective CS2 axes.
MCS on CS2	Coordinated Transform	T1 is canceled.
		The MCLM instruction on CS2 will continue.
		The MAM on Y will continue.
		The MAM on S will continue.
		The MAM on Z will continue.
		Axes ABC will stop due to canceling the transform.
MCS on CS3	Coordinated Transform	T1 is canceled.
		The MCLM instruction on CS2 will continue.
		The MAM on Y will continue.
		The MAM on S will continue.
		The MAM on Z will continue.
		Axes ABC will stop due to canceling the transform.

Example

Ladder Diagram



Structured Text

MCS(myMcsCoordinateSystem,myMcsMotionControl,All,Yes,myMcsDecelRate,Unitspersec2,Yes,100.0,%ofTime);

See also

Motion Error Codes (.ERR) on page 581

Multi-Axis Coordinated Motion Instructions on page 359

Common Attributes on page 691

Data Conversions on page 697

Structured Text Syntax on page 665

Motion Coordinated Transform (MCT)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers.

IMPORTANT Use this instruction with these controllers:
- 1756-L6 controllers
- 1756-L7 controllers
- 1756-L7S controllers
- 1756-L8 controllers
- 1769-L18ERM controllers
- 1769-L27ERM controllers
- 1769-L30ERM controllers
- 1769-L33ERM controllers
- 1769-L36ERM controllers

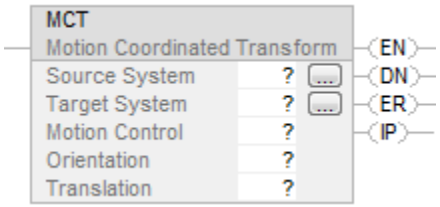
Use the MCT instruction to start a transform that links two coordinate systems together. This is like bi-directional gearing. One way to use the transform is to move a non-Cartesian robot to Cartesian positions.

IMPORTANT Tags used for the motion control attribute of instructions should only be used once. Re-use of the motion control tag in other instructions can cause unintended operation. This may result in damage to equipment or personal injury.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MCT(Source System, Target System, Motion Control, Orientation, Translation);

Operands

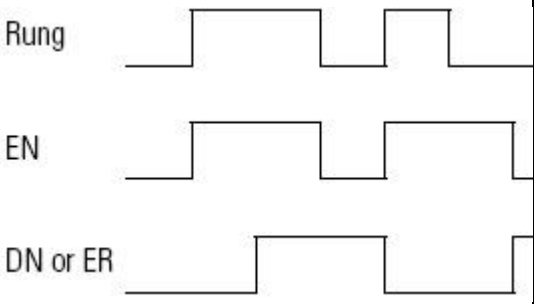
Ladder Diagram and Structured Text

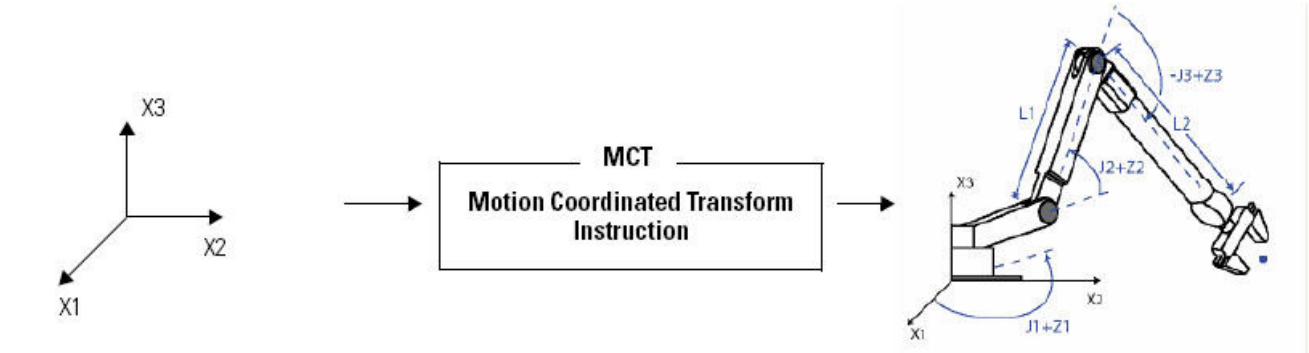
Operand	Type	Format	Description	
Source System	COORDINATE_SYSTEM	Tag	Coordinate system that you use to program the moves. Typically this is the Cartesian coordinate system. Reference to Coordinate system that you use to program the moves. Typically this is the Cartesian coordinate system. For controllers that support the REF_TO motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.	
Target System	COORDINATE_SYSTEM	Tag	Non-Cartesian coordinate system that controls the actual equipment. Reference to Non-Cartesian coordinate system that controls the actual equipment. For controllers that support the REF_TO motion data types, the coordinate system operand type can be replaced by the REF_TO_COORDINATE_SYSTEM type.	
Motion Control	MOTION_INSTRUCTION	Tag	Control tag for the instruction.	
Orientation	REAL[3] (units = coordinate units)	Array	Do you want to rotate the target position around the X1, X2, or X3 axis?	
			If	**Then**
			No	Leave the array vales at zero.
			Yes	Enter the degrees of rotation into the array. Put the degrees of rotation around X1 in the first element of the array, and then add the other elements.
			Use an array of three REALs even if a coordinate system has only one or two axes.	
Translation	REAL[3] (units = coordinate units)	Array	Do you want to offset the target position along the X1, X2, or X3 axis?	
			If	**Then**
			No	Leave the array values at zero.

			Yes	Enter the offset distances into the array. Enter the offset distances in coordinate units. Put the offset distance for X1 in the first element of the array, and then add the other elements.
			Use an array of three REALs even if a coordinate system has only one or two axes.	

See Structured Text Syntax for more information on the syntax of expressions within structured text.

MOTION_INSTRUCTION Data Type

To see if	Check if this bit is on	Data Type	Notes
The rung is true	EN	BOOL	Sometimes the EN bit stays on even if the rung goes false. This happens if the rung goes false before the instruction is done or an error has occurred. 
The instruction is done.	DN	BOOL	The transform keeps running after the instruction is done.
An error happened	ER	BOOL	Identify the error number listed in the error code field of the Motion control tag then, refer to Motion Error Codes.
The transform process is running.	IP	BOOL	Any of these actions cancels the transform and turns off the IP bit: Applicable stop instruction Shutdown instruction Fault action



You move a system of virtual axes to Cartesian positions (X1, X2, X3).

The transform converts the motion to joint angles and moves the robot.

The transform controls up to three joints of the robot: J1, J2, and J3.

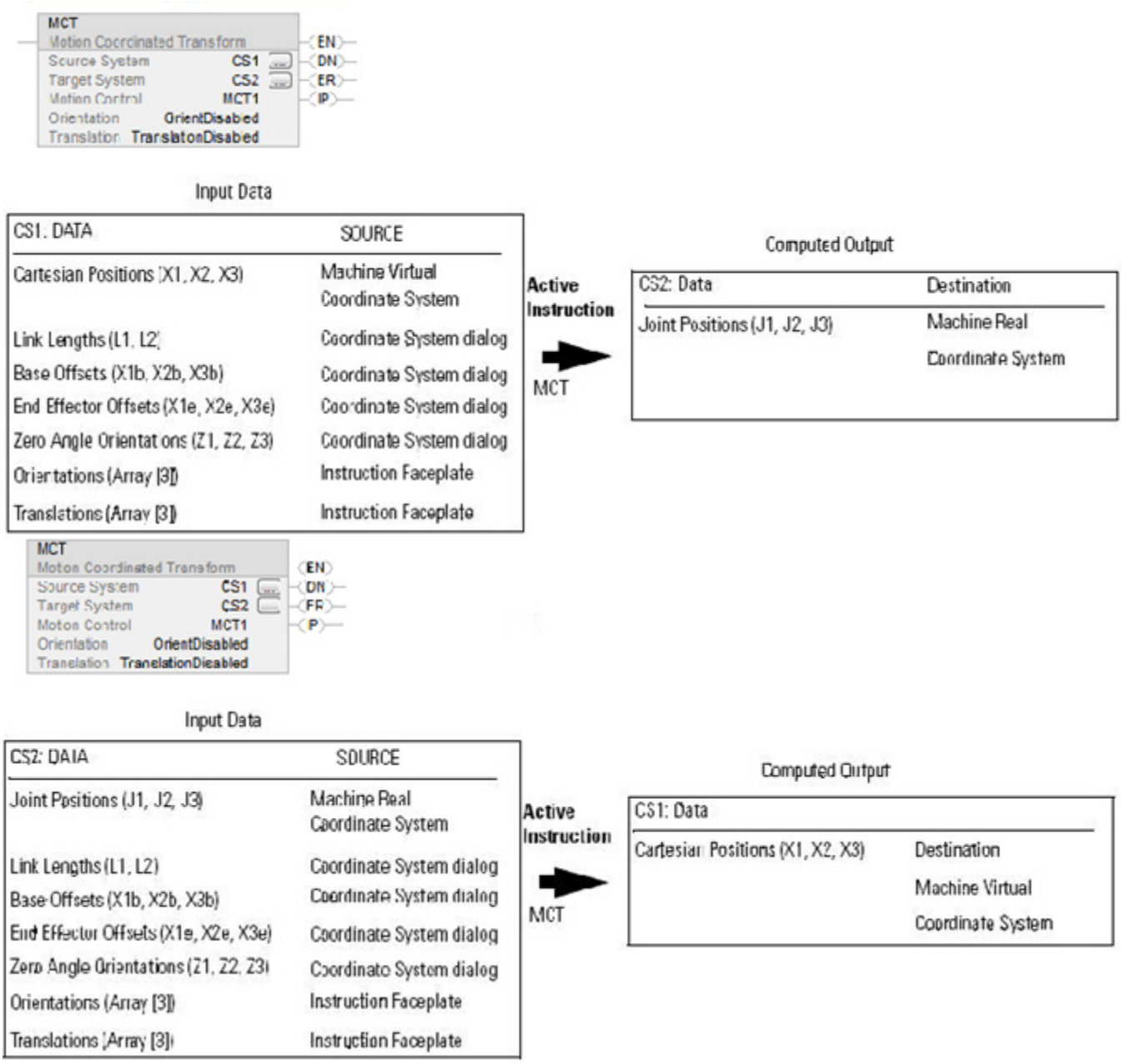
Data Flow of MCT Instruction between Two Coordinate Systems

The following illustrations show the flow of data when an MCT Instruction is active. CS1 is a Cartesian coordinate system containing X1, X2 and X3 axes as the source of the MCT instruction. CS2 is the joint coordinate system containing J1, J2, and J3 axes as the target of the MCT instruction.

All axes units are in Coordinate Units

Follow these guidelines to use an MCTP instruction.

Data Flow When a Move is executed with an MCT Instruction - Forward Transform

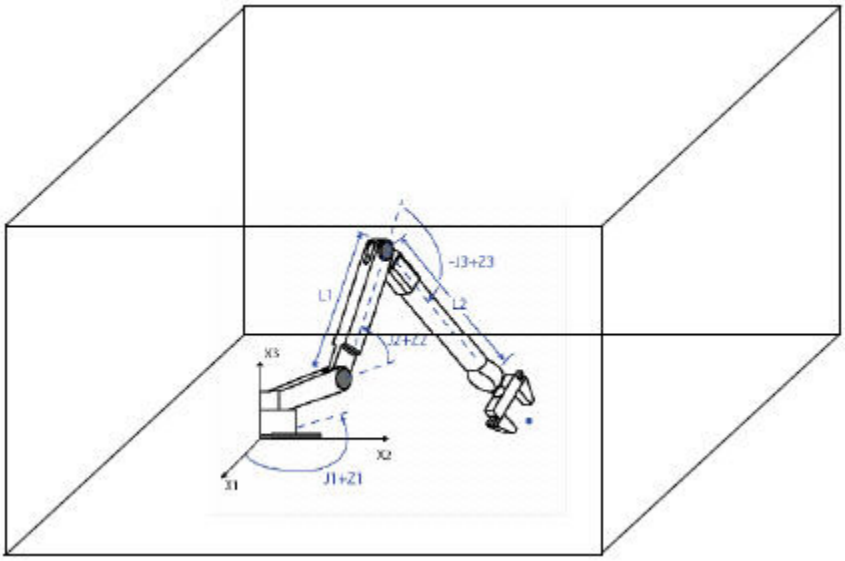


Data Flow When a Move is Executed with an MCT Instruction - Inverse Transform

Programming Guidelines

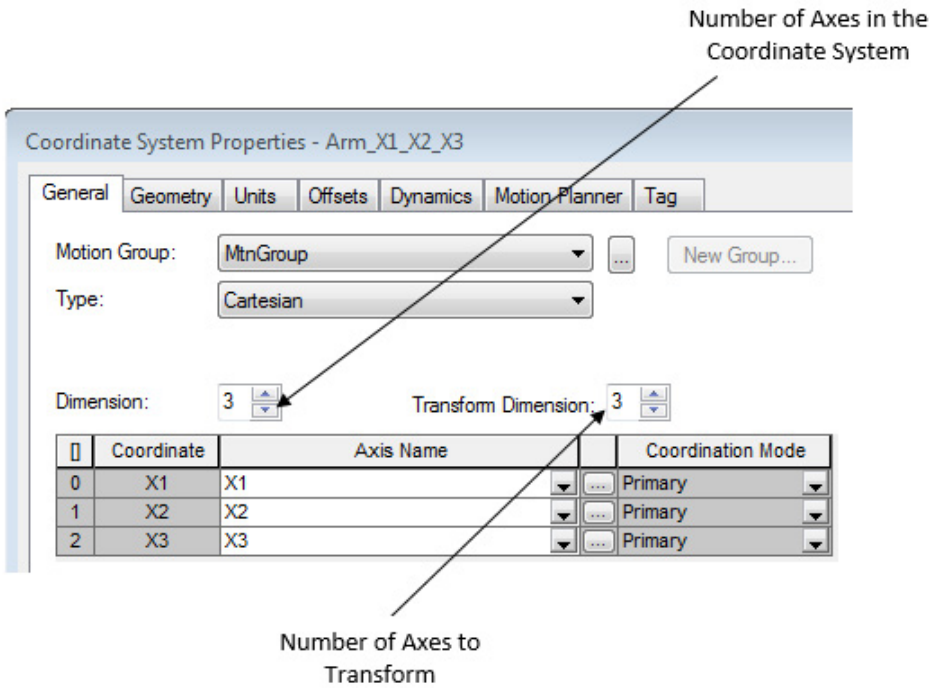
Follow these guidelines to use an MCT instruction.

Important:	Do not let the robot get fully stretched or fold back on itself. Otherwise it can start to move at a very high speed. In those positions, it loses its configuration as a left or right arm. When that happens, it can start to move at a very high speed. Determine the working limits of the robot and keep it within those limits.



Set up a coordinate system of axes for the Cartesian positions of the robot

These axes are typically virtual.



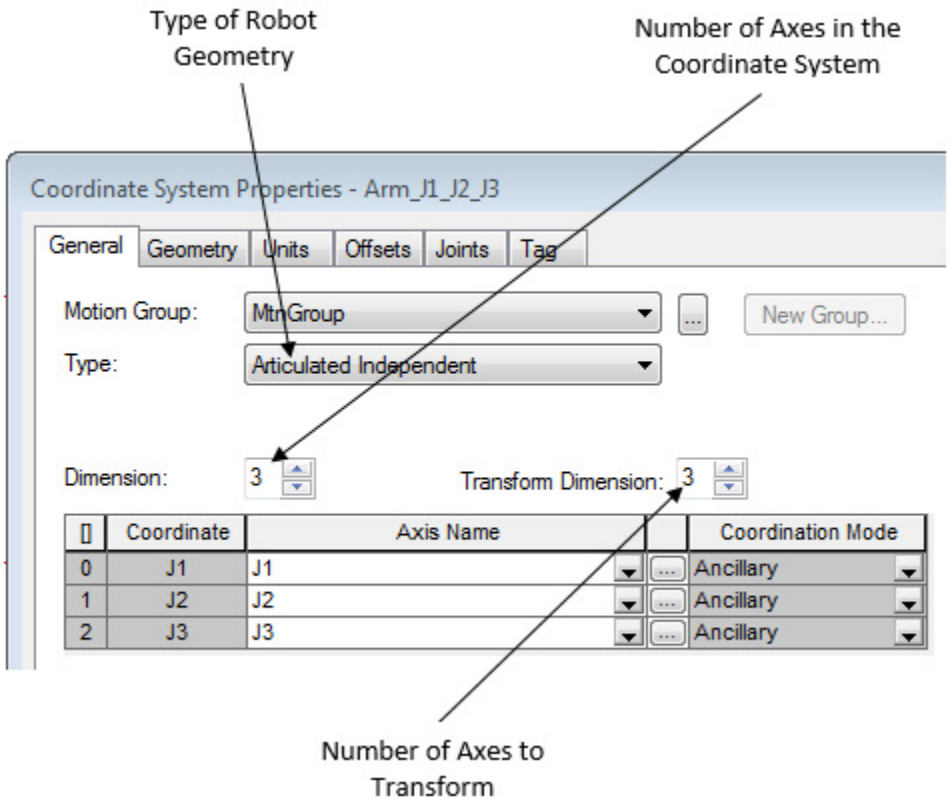
IMPORTANT You may see truncation error in the precision of computations. This happens when both of these conditions are true:

- The conversion constants of the virtual Cartesian axes in a transformation are small, such as 8000 counts/position unit.
- The link lengths of the non-Cartesian coordinate system are small, such as 0.5 inches.

It is best to give large conversion constants to the virtual Cartesian axes in a transform, such as 100,000 or 1,000,000 counts/position unit. The maximum travel limit of the robot is:

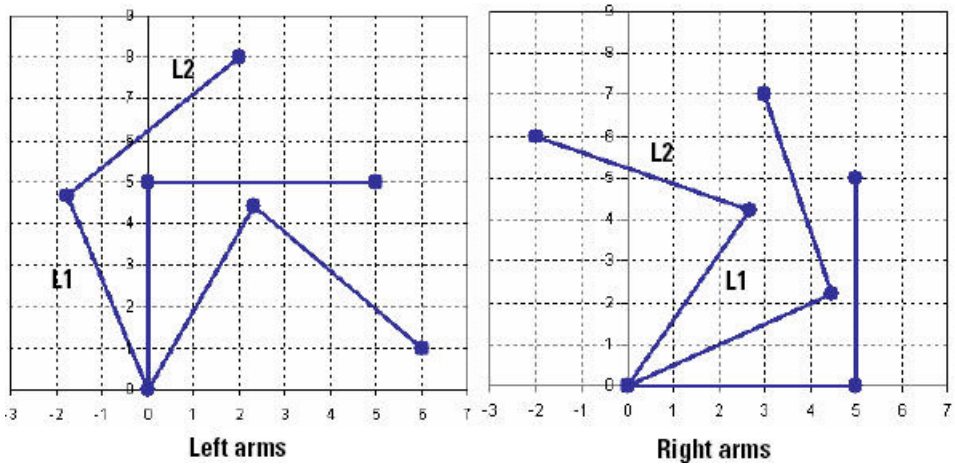
$$\frac{\pm 2^{31}}{\text{Conversion Constant}} \text{ Coordination Units}$$

Set up another coordinate system for the actual joints of the robot



Move the robot to a left- or right-arm starting position

Do you want the robot to move like a left arm or a right arm?



Before you start the transform, move the robot to a resting position that gives it the arm side that you want (left or right).

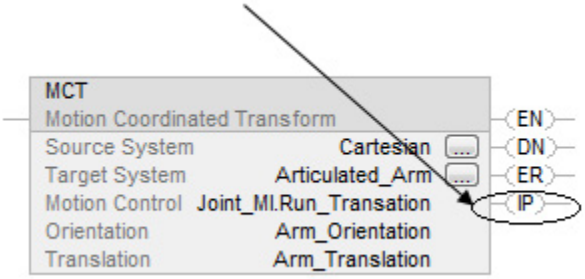
Once you start the transform and initiate a Cartesian move in the Source coordinate system, the robot stays as a left arm or a right arm. If it starts as a left arm, it moves as a left arm. If it starts as a right arm, it moves as a right

arm. You can always flip it from a left arm to a right arm or vice versa. To do that, move the joints directly.

Toggle the rung from false to true to execute the instruction

This is a transitional instruction. In a ladder diagram, toggle the Rung-condition-in from false to true each time you want to execute the instruction.

When you execute the instruction, the transform starts and the IP bit turns on.



You can let the rung go false once you execute the instruction. The transform stays active.

In structured text, condition the instruction so that it only executes on a transition

Start the transform before you start any motion.

In structured text, instructions execute each time they are scanned. Condition the instruction so that it only executes on a transition. Use one of these methods:

- Qualifier of an SFC action
- Structured text construct

You cannot start a transform if any motion process is controlling an axis of the source or target coordinate systems.

Example: Start the transform before you start gearing or camming.

Expect bi-directional motion between the source and target coordinate systems

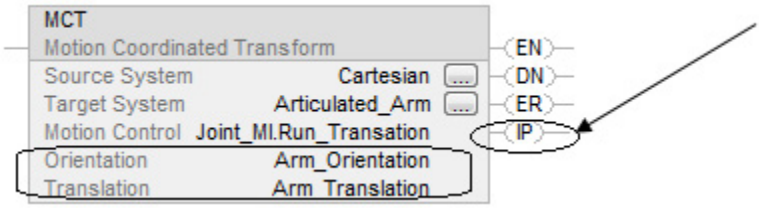
Use an MCS instruction to cancel the transform.

A transform is bi-directional.

When you start the transform, the position of the source coordinate system changes to match the corresponding position of the target coordinate system. After that, if you move either system, the other system moves in response.

The controller continues to control the axes even if you stop scanning the MCT instruction or its rung goes false. Use a Motion Coordinated Stop (MCS) instruction to stop the motion in the coordinate system, cancel the transform, or both.

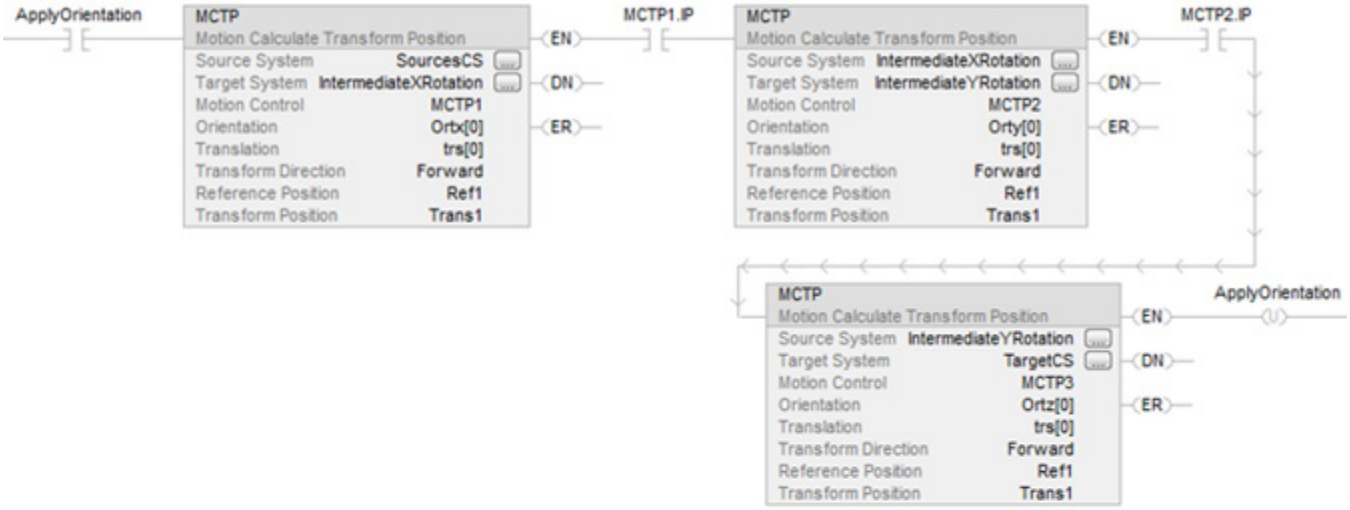
If you change the orientation or translation, execute the MCT instruction again



Then, execute the instruction again. To execute the instruction, toggle the rung-condition-in from false to true.

If you change the geometry of the equipment, execute the instruction again.

Specify and execute more than one orientation angle in the Orientation operand of the MCTP instruction



Ortx	Controller	{...}
Ortx[0]	Controller	30.0
Ortx[1]	Controller	0.0
Ortx[2]	Controller	0.0

Orty	Controller	{...}
Orty[0]	Controller	0.0
Orty[1]	Controller	40.0
Orty[2]	Controller	0.0

Ortz	Controller	{...}
Ortz[0]	Controller	0.0
Ortz[1]	Controller	0.0
Ortz[2]	Controller	60.0

The rotations around X, Y, Z are entered in the following order:

The first element from the first MCTP orientation operand array is used to specify X rotation.

The second element from the second MCTP orientation operand is used to specify Y rotation.

The third element from the third MCTP orientation operand is used to specify Z rotation.

Example

The following table shows before and after rotation orders. Note that the order of rotations in n-Dimensions is not commutative.

	Rotation: V20 or greater **Matrix Multiply Order: (Z*(YX)** **Rotate Around X then Y then Z:**
Rotation Cartesian => Cartesian	Logix Designer V20 or greater
MCT Orientation [x,y,z]	MCT Starting Position Resulting Oriented Position
1 Dimension Rotation	
Starting Position = [1, 2, 3]	
MCT Orient=[90, 0, 0]	Rotation: 90 (cw) around X. = [1, 3, -2]
MCT Orient=[0, 90, 0]	Rotation: 90 (cw) around Y. = [-3, 2, 1]
MCT Orient=[0, 0, 90]	Rotation: 90 (cw) around Z. = [2, -1, 3]
2 Dimension Rotation	
Starting Position = [1, 2, 3]	
MCT Orient=[90, 90, 0]	Rotation: 90 (cw) around X then 90 (cw) around Y. = [-3, 1, -2]
MCT Orient=[90, 0, 90]	Rotation: 90 (cw) around X then 90 (cw) around Z. = [2, 3, 1]
MCT Orient=[0, 90, 90]	Rotation: 90 (cw) around Y then 90 (cw) around Z. = [-3, -1, 2]
3 Dimension Rotation	
Starting Position = [1, 2, 3]	
MCT Orient=[90, 90, 90]	Rotation: 90 (cw) around X then 90 (cw) around Y then 90 (cw) around Z. = [-3, 2, 1]
MCT Orient=[-90, -90, -90]	Rotation: 90 (ccw) around X then 90 (ccw) around Y then 90 (ccw) around Z. = [3, -2, 1]

MCT Instruction Guidelines

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. A major fault can occur if an uninitialized reference or a reference of the incorrect type is passed to the Coordinate System operands See Common Attributes for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false, followed by rung is true in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See Motion Error Codes (ERR) for Motion Instructions.

Extended Error Codes

See Motion Error Codes (ERR) for Motion Instructions. Use Extended Error Codes (EXERR) for more instruction about an error.

ERR	EXERR	Corrective Action	Notes
61	1	Assign both coordinate systems to the motion group.	
	2	Check that you are using the correct source and target systems.	You cannot use the same coordinate system as source and target.
	3	Set the transform dimension of the source system to the number of axes in the system, up to three.	
	4	Set the transform dimension of the target system to the number of axes in the system to be transformed, up to three.	

	5	Use a different source system.	You can only use one coordinate system as the source for one active transform.
	6	Use a different target system.	You can only use one coordinate system as the target for one active transform.
	7	Look for source or target axes that you are already using in another transform. Use different axes in the coordinate system.	You can only use an axis in one source system and one target system.
	8	Use a target system that isn't't the source for this chain of transforms.	You cannot create a circular chain of transforms that leads back to the original source.
	9	Check that you have assigned the correct axes to each coordinate system.	You cannot use the same axes in the source and target systems.
	10	Stop all motion processes for all the axes in both systems (for example, jog, move, and gear).	You cannot start the transform if any motion process is controlling a source or target axis.
	11	Insufficient resources available to initiate the transform connection.	
	12	Set the link lengths.	You cannot use a link length of zero.
	13	Look for source or target axes that are in the shutdown state. Use a Motion Axis Shutdown Reset (MASR) instruction or direct command to reset the axes.	
	14	Uninhibit all the source or target axes.	
	15	Check the configured values for the base offsets and end effector offsets for the Delta or SCARA Delta robot.	(X1b-X1e) cannot be less than 0.0 for both the Delta and SCARA Delta robots. For Delta robots, this error can also occur if the value of L1 + (X1b-X1e) is greater than L2.
	16	Check the SCARA independent and SCARA Delta robot configurations to be sure that: The transform dimension for the source coordinate system is configured as 2. The configured third axes for the source coordinate system and the target coordinate system are the same.	
	17	Check the source and target coordinate systems to verify that the transform dimension of the source coordinate system equals the transform dimension of the target coordinate system.	

Changes to Status Bits

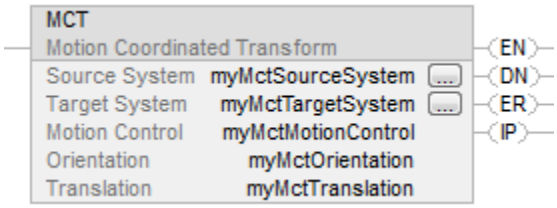
The instruction changes these status bits when it executes.

To see if	Check the tag for the	And this bit	For
A coordinate system is the source of an active transform.	Coordinate System	TransformSourceStatus	On

A coordinate system is the target of an active transform.	Coordinate System	TransformTargetStatus	On
An axis is part of an active transform.	Axis	TransformStateStatus	On
An axis is moving because of a transform.	Axis	ControlledByTransformStatus	On

Examples

Ladder Diagram



Structured Text

MCT(myMctSourceSystem,myMctTargetSystem,myMctMotionControl,myMctOrientation,myMctTranslation);

See also

Choose a Termination Type on page 517

Structured Text Syntax on page 665

Input and Output Parameters Structure for Coordinate System Motion Instructions on page 527

Motion Error Codes (.ERR) on page 581

Common Attributes on page 691

Speed, acceleration, deceleration, and jerk enumerations

Speed enumerations

Common enumerations are used for the speed parameter of all motion instructions. Some instructions accept only limited subset of the speed enumerations. Checks for valid unit combinations are done at instruction execution time. Some enumerations that are in the following table are not used now but are reserved for future enhancements. Additional tables are given below that further clarify which combinations are accepted in MDSC mode and which are accepted in Time Driven Mode.

Enumeration	Definition	Mode	Compatibility	Note

0 1 2 3	Units per sec % Maximum Reserved Reserved	Time Driven	Existing Enumeration Existing Enumeration New Enumeration	Reserved for Time based programming
4 5 6 7	Units per MasterUnit Reserved Reserved Master Units	MDSC	New Enumeration New Enumeration	Reserved for Time based programming

Follow these rules for Speed to determine allowable Time and MDSC Driven Mode:

- When Speed is in either units/sec, %max, or seconds, then the instruction is considered to be in Time Driven Mode, regardless of the selection of units for acceleration, deceleration, or jerk.
- When Speed is in either Master Units or in Units/MasterUnit, then the instruction is considered to be in Master Driven Mode, regardless of the selection of units for acceleration, deceleration, or jerk.
- Speed, Acceleration, Deceleration, and Jerk must always be programmed in the same mode (Time Driven or Master Driven) or you get a runtime error.
- When speed is specified in time unit seconds, the specified time is the total time of the move, including acceleration and deceleration time.
- When speed is specified in Master distance units, the specified distance is the total master distance of the move, including acceleration and deceleration distance of the Master Axis.

Acceleration and deceleration enumerations

These enumerations are defined for Acceleration and Deceleration Unit parameters for motion instructions.

Enumeration	Description	Mode	Compatibility	Notes
0 1 2 3	Units per sec2 % Maximum Reserved Reserved	Time	Existing Enumeration Existing Enumeration	Reserved for Time based programming
4 5 6 7	Units per MasterUnit2 Reserved Reserved Reserved	MDSC	New Enumeration	Reserved for Time based programming

This table shows acceptable combinations of Speed, Acceleration, and Deceleration units.

		Acceleration and Deceleration Units				
		Units per sec² (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit² (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**
Speed Units	**Units per sec (Time Driven Mode Units)**	Existing Enumeration	Existing Enumeration	Not Implemented	Not allowed - Time and Master Driven Units may not be combined.	
	% Maximum (Time Driven Mode Units)	Existing Enumeration	Existing Enumeration	Not Implemented		
	Seconds (Time Driven Mode Units)	Not Implemented	Not Implemented	New Enumeration		
	Units per MasterUnits (Master Driven Mode Units)	Not allowed - Time and Master Driven Units may not be combined.			New Enumeration	Not Implemented
	Master Units (Master Driven Mode Units)				Not Implemented	New Enumeration

These rules for Acceleration and Deceleration must be followed to determine allowable Time and Master Driven Mode:

- Speed, Acceleration, Deceleration, and Jerk must always be programmed in the same mode or you get an error.
- If Speed units are seconds, then acceleration, deceleration, and jerk units must be seconds too.
- If Speed units are Master units, then acceleration, deceleration, and jerk units must be Master units too.
- All unsupported unit combinations result in an error at runtime when the instruction is executed.

Jerk enumerations

The following enumerations are defined for time driven and MDSC driven Jerk units.

Enumeration	Description	Mode	Compatibility	Notes
0 1 2 3	Units per sec^3 % Maximum % of Time Reserved	Time	Existing Enumeration Existing Enumeration Existing Enumeration	Reserved for Time based programming
4 5 6 7	Units per MasterUnit^3 Reserved % of Time-Master Driven Reserved	MDSC	New Enumeration New Enumeration	Reserved for Time based programming

Acceptable combinations of Accel and Decel Units are based on the programmed Speed Units in the instruction as is shown in the table below. Use this table to clarify the differences in the following four tables.

Speed Units	Accel Units vs Jerk Units Defined in Table:
Units per Sec	Table 1
Units per Master Units	Table 2
Seconds	Table 3
Master Units	Table 4

This table shows acceptable combinations of Acceleration Units and Jerk Units when Speed Units are Units per Sec.

	Acceleration Units (Speed in Units per Second)				
	Units per sec^2 (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit^2 (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**

This table shows acceptable combinations of Acceleration Units and Jerk Units when Speed Units are Units per Master Unit.

		Acceleration (Speed in Units per Master Unit)				
		Units per sec^2 (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit^2 (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**
Jerk Units	**Units per sec^3 (Time Driven Mode Units)**	Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.			Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.	
	% Maximum (Time Driven Mode Units)					
	% of Time (Time Driven Mode Units)					
	Seconds (Time Driven Mode Units)					

	Units per MasterUnits³ (Master Driven Mode Units)	Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.	New Enumeration.	Not Implemented
	% of Time-Master Driven (Master Driven Mode Units)		New Enumeration.	Not Implemented
	Master Units (Master Driven Mode Units)		Not Implemented	Not Implemented

The following table shows acceptable combinations of Acceleration Units and Jerk Units when Speed Units are in Seconds.

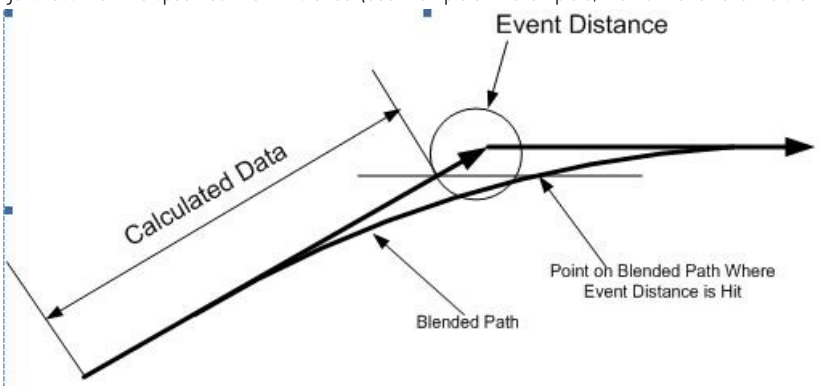
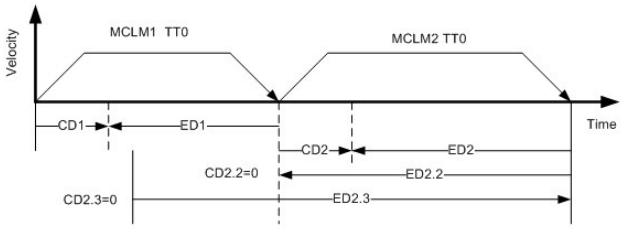
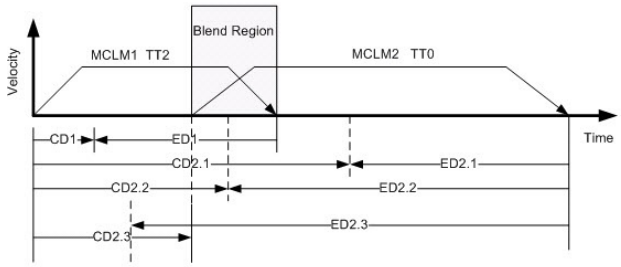
	Acceleration (Speed in Seconds)				
	Units per sec² (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit2 (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**

This table shows acceptable combinations of Acceleration Units and Jerk Units when Speed is in Master Units.

		Acceleration (Speed in MasterUnits)				
		Units per sec² (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit² (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**
Jerk Units	**Units per sec³ (Time Driven Mode Units)**	Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.			Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.	
	% Maximum (Time Driven Mode Units)					
	% of Time (Time Driven Mode Units)					
	Seconds (Time Driven Mode Units)					
	Units per MasterUnits³ (Master Driven Mode Units)	Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.			Not Implemented	Not Implemented
	% of Time-Master Driven (Master Driven Mode Units)				Not Implemented	New Enumeration.
	Master Units (Master Driven Mode Units)				Not Implemented	New Enumeration.

Returned Calculated Data Parameter

The returned Calculated Data value for Coordinated System Motion instructions is outlined in the table.

Mode	Returned Calculated Data Parameter
Master Driven	The returned Calculated Data parameter is the incremental delta Master position that is needed to make the Slave Coordinate System move from the point at which Slave Coordinate System is locked to the Master and starts moving along the programmed path to the point where distance to go is less than the specified Event Distance. (See Example 3. In example 3, the MSP for all event distances is point P0.)  • For Blended moves (that is, Termination Type =Command Tolerance or No Decel) The incremental Master Axis distance needed for the programmed move, in the Slave Coordinate System, to travel from the beginning of the move to the Blend Point. Note that this is where the PC bit of the instruction is set. • For all other termination types (that is, non-blended moves) The incremental Master Axis distance needed for programmed move, in the Slave Coordinate System, to travel from the beginning of the move to the programmed endpoint. Note that this is where the PC bit of instruction is set on the instruction moving the slave Another way to represent the Event Distance and the corresponding Calculated Data is on a Velocity versus Time plot as is shown in the following figure: Note that the first plot below is for non-blended moves (TT0/1), the second is for blended (TT2, 3, 6).   CD = Calculated Data CD2.1 = Calculated Data 1 for Move 2 ED = Event Distance ED2.3 = Event Distance 3 for Move 2 TT = Termination Type CD2.1 is how far the master has to move when the slave reaches position ED2.1 **Note**: As shown above, the Event Distance is > Move length and is therefore internally set equal to the move length. The Calculated Data for the move length is therefore returned. No error is forced.
Time Driven	The returned data in the Calculated Data parameter is the total time in seconds that is needed to make the Slave Coordinate System move from the move's start point to a point where distance to go is less than the specified Event Distance. If the specified data in the Event Distance is array element is 0.0, then the time it takes the entire move to complete is returned.

See also

Input and Output Parameters Structure for Coordinate System Motion Instructions on page 527

Status Bits for MCLM, MCCM when MDCC Is Active

This table describes the predefined data type status bits for motion instruction MCLM and MCCM.

Bit Name	Meaning
FLAGS	
EN	
DN	
ER	
PC	
IP	
AC	
ACCEL	Set as expected during motion. It is independent of Master acceleration. The ACCEL bit on the instruction driving the Slave Coordinate System (for example, MCLM) is set as the Slave Coordinate System is accelerating to its commanded speed. This bit is insensitive to acceleration occurring on the Master Axis. However, the AccelStatus bit, which is in the MotionStatus word of the Slave Coordinate System (not the instruction driving the Slave Coordinate System), is set or cleared based on changes in velocity of the Slave Coordinate System.
DECEL	Set as expected during motion. It is independent of Master deceleration. The DECEL bit on the instruction driving the Slave Coordinate System is set as the Slave Coordinate System is decelerating to its commanded speed. This bit is insensitive to deceleration occurring on the Master Axis. However, the DecelStatus bit, which is in the MotionStatus word of the Slave Coordinate System (not the instruction driving the slave axis), is set or cleared based on changes in velocity of the Slave Coordinate System.
TrackingMaster	Indicates that the Slave Coordinate System is tracking the Master Axis (only used in Master Driven Mode). When an instruction is initiated in Master Driven Mode, the Slave Coordinate System accelerates to the speed that is programmed for MDSC mode. The Tracking Master is set when the acceleration is complete in MDSC Mode. This means that the Slave Coordinate System is synchronized to the Master Axis. The Tracking Master bit is cleared when any of the following occurs on the Slave Coordinate System: • When the Slave Coordinate System starts to either accelerate or decelerate for any reason, for example, for an MCCD or an MCS being issued. • When the Slave Coordinate System is relinked to another Master Axis. In this situation, the TrackingMaster bit is first cleared and then it is set again in the new instruction status word when the Slave Coordinate System starts tracking the new Master Axis again. • The Slave Coordinate System is stopped. The Tracking Master is cleared as soon as the stop is initiated on the Slave Coordinate System. This bit is never set when LockDir = NONE. Note that the Tracking Master bit on the Slave Coordinate System is not affected by any operation (for example, MCS, MCCD) on the Master Axis. The Tracking Master bit is always cleared in Time Driven Mode.

CalculatedDataAvailable	Indicates that the requested data has been returned in the Calculated Data array element and that the Logix Designer application has updated the output data in the Calculated Data parameter. Only one status bit is used to indicate all Calculated Data is available.
	For the CalculatedDataAvailabe status bit, the moves in the motion queue are processed in batches. The first batch in the motion queue includes all moves in the queue up to and including the first move with a term type TT0 or TT1, or a move with a speed of 0.
	For moves in either Time Driven mode or Mater Driven mode, the CalculatedDataAvailable bit is set when:
	• MCLM or MCCM is enqueued and belongs to the first batch in the queue. There are two exceptions:
	  • Moves with a speed of 0, although belonging to the first batch, do not have their CalculatedDataAvailable bit set. Their CalculatedDataAvailable bit is set after their Speed is changed to nonzero with a MCCD.
	  • Moves with a term type TT2 through TT6 do not have their CalculatedDataAvailable bit set if they are the last move in the queue.
	CalculatedDataAvailable bit is cleared by:
	• MAS (all) or MASD - This clears the CalculatedDataAvailable bit of the active MAMs and all enqueued MCLM or MCCMs that contain the specified axis.
	• MCS (coordinated) - This only clears the CalculatedDataAvailable bit for all enqueued MCLM or MCCMs in the coordinate system being stopped.
	• MCS (all) or MCSD - This clears the CalculatedDataAvailable bit of all active MAMs that contain any axes in the referenced coordinate system and all enqueued MCLM or MCCMs of the coordinate system being stopped.
	• MGS or MGSD is executed (goes IP) - This clears the CalculatedDataAvailable bit of all active MAMs and all enqueued MCLM or MCCMs of the group being stopped or shutdown.
	• MCD or MCCD is executed (goes IP) - The CalculatedDataAvailable bit is reset and is immediately set again.
	• A MCLM or MCCM is executed (goes IP) with a merge enabled (either Coordinated or Merge All) - The CalculatedDataAvailable bit of all enqueued MCLM or MCCMs are cleared.
	MCLMs and MCCMs that are blending with the next coordinated motion instruction are still considered to be enqueued even if their PC flag was set when the blending was started.
	The CalculatedDataAvailable bit is not set for any move that Event Distance is not specified (that is, for any move where the Event Distance parameter in the instruction is zero).
	MSF and MDF do not alter the state of the CalculatedDataAvailable bit.

Coordinated Motion Status Bits

Bit Name	Meaning
CoordinateMotionStatus	Set when an axis lock is requested for an MCLM or MCCM instruction and the axis has crossed the Lock Position. Cleared when an MCLM or MCCM is initiated.
AccelStatus	Sets when vector is accelerating. Clears when a blend is in process or when vector move is at speed or decelerating.
DecelStatus	Sets when vector is decelerating. Clears when a blend is in process or when vector move is accelerating or when move completes.
ActualPosToleranceStatus	Sets for Actual Tolerance termination type only. The bit is set after the following two conditions have been met. 1) Interpolation is complete. 2) The actual distance to the programmed endpoint is less than the configured coordinate system's Actual Tolerance value. It remains set after the instruction completes. It is reset when a new instruction is started.
CommandPosToleranceStatus	Sets for all termination types whenever the distance to the programmed endpoint is less than the configured coordinate system's Command Tolerance value and remains set after the instruction completes. It is reset when a new instruction is started.
StoppingStatus	The Stopping Status bit is cleared when the MCCM instruction executes.
MoveStatus	Sets when MCCM begins axis motion. Clears on the .PC bit of the last motion instruction or a motion instruction executes which causes a stop.

MoveTransitionStatus	Sets when No Decel or Command Tolerance termination type is satisfied. When blending collinear moves the bit is not set because the machine is always on path. It clears when a blend completes, the motion of a pending instruction starts, or a motion instruction executes which causes a stop. Indicates not on path.
MovePendingQueueFullStatus	Sets when the instruction queue is full. It clears when the queue has room to hold another new coordinated move instruction.
TransformSourceStatus	The coordinate system is the source of an active transform.
TransformTargetStatus	The coordinate system is the target of an active transform.
CoorMotionLockStatus	Set when an axis lock is requested for an MCLM or MCCM instruction and the axis has crossed the Lock Position. Cleared when an MCLM or MCCM is initiated. For the enumerations Immediate Forward Only and Immediate Reverse Only, the bit is set immediately when the MCLM or MCCM is initiated. When the enumeration is Position Forward Only or Position Reverse Only, the bit is set when the Master Axis crosses the Lock Position in the specified direction. The bit is never set if the enumeration is NONE. The CoordMotionLockStatus bit is cleared when the Master Axis reverses direction and the Slave Coordinate System stops following the Master Axis. The CoordMotionLockStatus bit is set again when the Slave Coordinate System resumes following the Master Axis. The CoordMotionLockStatus bit is also cleared when an MCS is initiated.

Change between master and time driven modes

Changing the motion mode between Master Driven and Time Driven Mode and vice versa is automatically performed when another motion instruction (such as, MCLM and MCCM) is activated if the new instruction has been programmed in a different mode than the active motion instruction.

When the new motion instruction is activated, the system will assume that the desired mode for the new instruction is the mode (Master Driven or Time Driven) as specified in the programmed units of the speed parameter contained in the new instruction. At all times, including when changing modes, the Accel, Decel, and Jerk must be programmed in the same units as the Speed parameter or the instruction will get a MDSC_UNITS_CONFLICT_ERROR error.

A runtime MDSC_INVALID_MODE_OR_MASTER_CHANGE error will occur if you attempt to change from Master Driven Mode to Time Driven Mode or vice versa with an MCCD instruction.

If both the master and slave axes are idle (for example, paused), the MCLM or MCCM can make a change on the slave. However, the error MDSC IDLE_MASTER_AND_SLAVE_MOVING is generated if MDSC mode is started while the Slave Coordinate System is moving when the master is idle.

Different Time Driven and Master Driven Modes may be used for different motion types for superimposed motion. For example, the MAM may be in time drive mode for an axis in the Coordinate System and the MCLM may be in Master Driven Mode for the Coordinate System.

Change the Master Axis

Follow this sequence of events to transfer a Slave Coordinate System from one Master Axis to a second Master Axis.

- First, you must execute an MDCC instruction to reassign the Slave Coordinate System from the first Master Axis to the second Master Axis. This makes the reassignment pending. The IP bit of the MDCC instruction is set as an indication of the pending reassignment.
- Second, you must execute a new motion command (for example, an MCLM or MCCM). The Slave Coordinate System becomes unlocked from the first Master Axis and reassigned to the second Master Axis when this motion instruction is executed (goes IP).

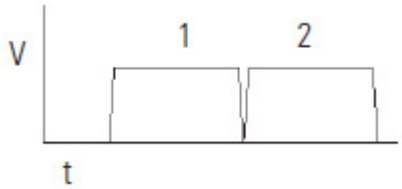
The final effective slave speed is computed as the product of the Master Axis' speed and the slave's programmed speed. If the new final effective Slave Coordinate System speed is less than 10%, depending on the move of the original Slave Coordinate System speed, then the change will not be allowed and the MDSC_INVALID_SLAVE_SPEED_REDUCTION error will occur. If the second Master Axis is idle (velocity=0), the motion instruction making this request receives an MDSC_IDLE_MASTER_AND_SLAVE_MOVING error.

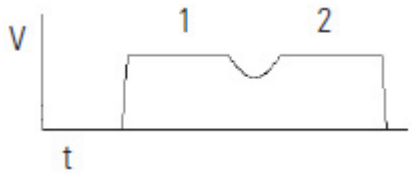
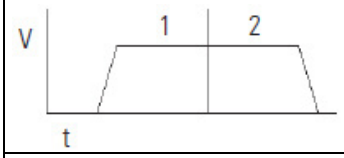
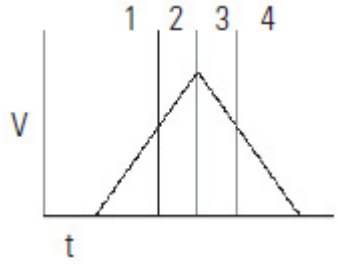
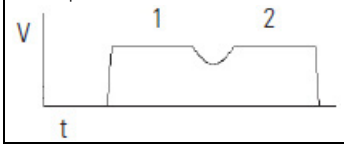
If the second Master Axis is moving while the transfer is being made, then you may look at the TrackingMaster instruction status bit of the motion instruction that is performing the transfer to determine when the transfer is finished. This bit is set when the acceleration or deceleration on the Slave Coordinate System is complete. At which time the Slave Coordinate System will be synchronized to the second Master Axis.

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:

If you want the axes to (vector speeds)	And you want the instruction to complete when	Then use this Termination Type
stop between moves.  	The following occurs: • 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.	0 - Actual Tolerance
	The command position equals the target position.	1 - No Settle
keep the speed constant **except** between moves.	The command position gets within the Command Position Tolerance of the coordinate system.	2 - Command Tolerance

	The axes get to the point at which they must decelerate at the deceleration rate.	3 - No Decel
transition into or out of a circle without stopping. 		4 - Follow Contour Velocity Constrained
accelerate or decelerate across multiple moves. 		5 - Follow Contour Velocity Unconstrained
use a specified Command Tolerance 	The command position gets within the Command Position Tolerance of the coordinate system.	6 - Command Tolerance Programmed

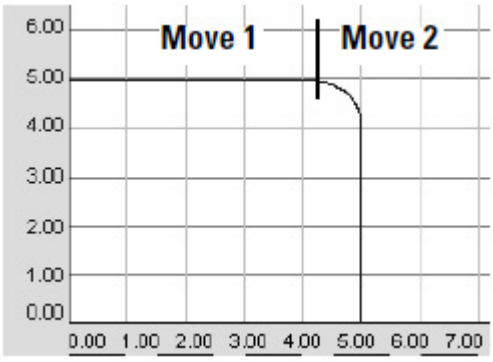
To make sure that this is the right choice for you:

- Review these tables.

Termination Type	Example Path	Description
0 - Actual Tolerance		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		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		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	To the	And uses the	For the
100% of the configured length of the first instruction using a Command Tolerance termination type	configured Command Tolerance for the Coordinate System	shorter of the two lengths	command Tolerance length used for the **first** instruction
100% of the configured length of the last move instruction using a Command Tolerance termination type	configured Command Tolerance for the Coordinate System	shorter of the two lengths	command Tolerance length used for the **next to last** instruction
50% of each of the lengths of all other move instructions	configured Command Tolerance for the Coordinate System	shorter of the two lengths	command Tolerance length used for **each individual** instruction

Termination Type	Example Path	Description
3 - No Decel		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. • The deceleration point depends on whether you use a trapezoidal or S-curve profile. • If you don't have a queued MCLM or MCCM instruction, the axes stop at the target position.

4 - Follow Contour Velocity Constrained		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. • 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. • The axes follow the path. • 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 Unconstrained		This termination type is similar to the contour velocity constrained. It has these differences: • Use this termination type to get a triangular velocity profile across several moves. This reduces jerk. • 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. • 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

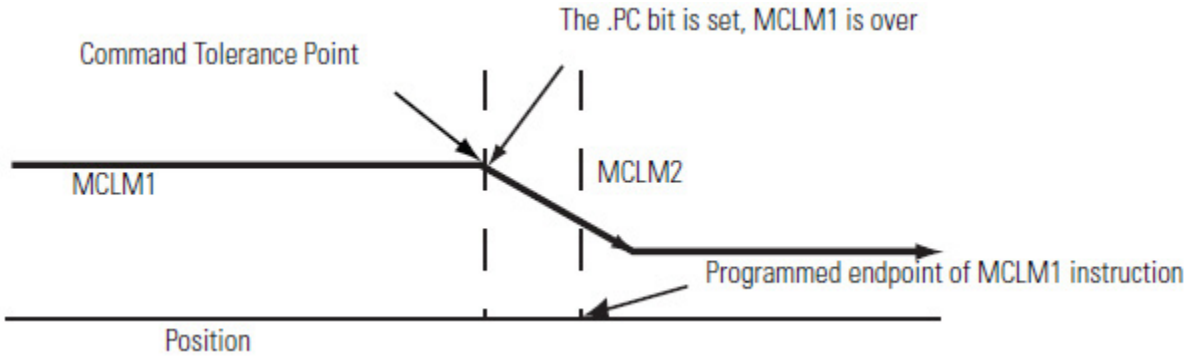
Collinear moves are those that lie on the same line in space. Their direction can be the same or opposite. The velocity profiles for collinear moves can be complex. This section provides you with examples and illustrations to help you understand the velocity profiles for collinear moves programmed with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

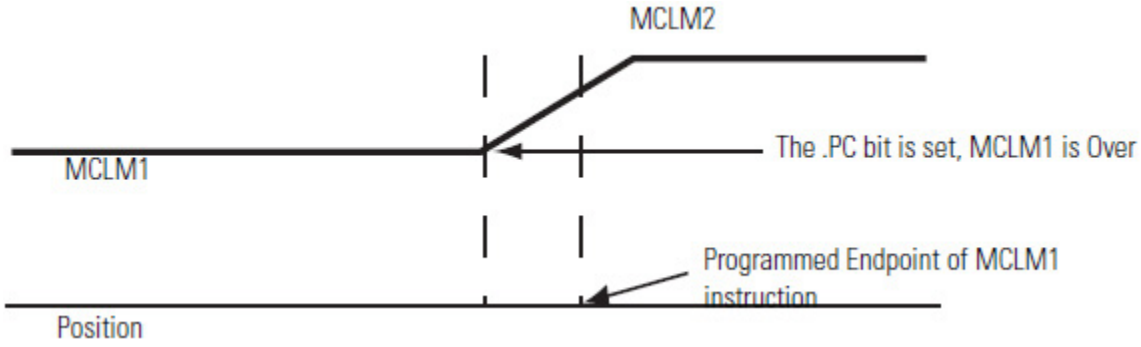
instruction reaches its Command Tolerance point, the move is over and the .PC bit is set.

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



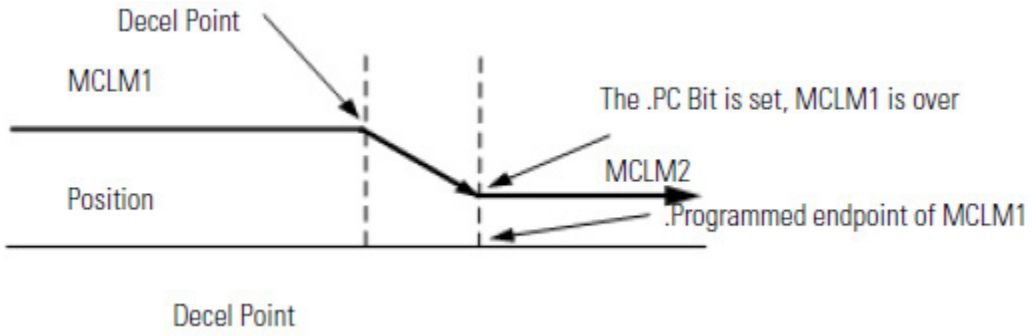
Velocity Profiles for Collinear Moves with Termination Types 3, 4, or 5

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

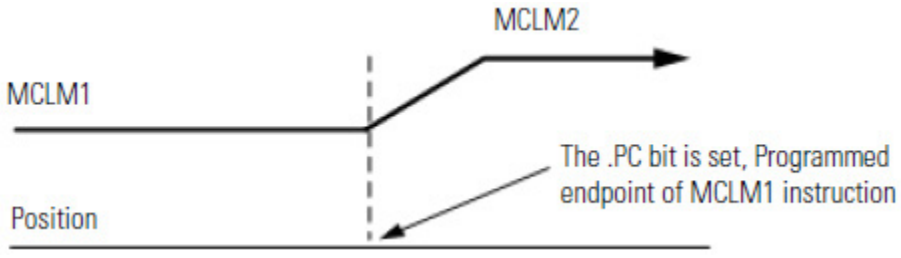


This illustration shows a velocity profile of two collinear moves. The second MCLM instruction has a **higher** velocity than the first MCLM instruction and one of these termination types are used:

- 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



Symmetric Profiles

Profile paths are symmetric for all motion profiles.

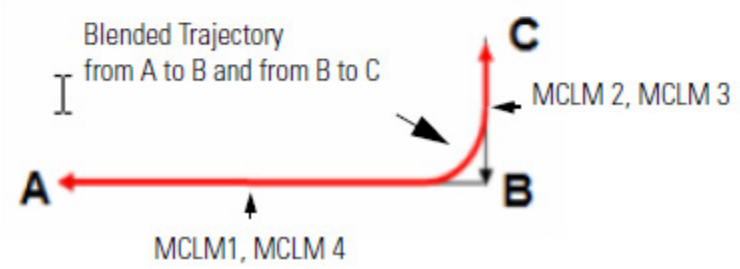
Programming the velocity, acceleration, and deceleration values symmetrically in the forward and reverse directions generates the same path from point A to point C in the forward direction, as from point C to point A in the reverse direction.

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.

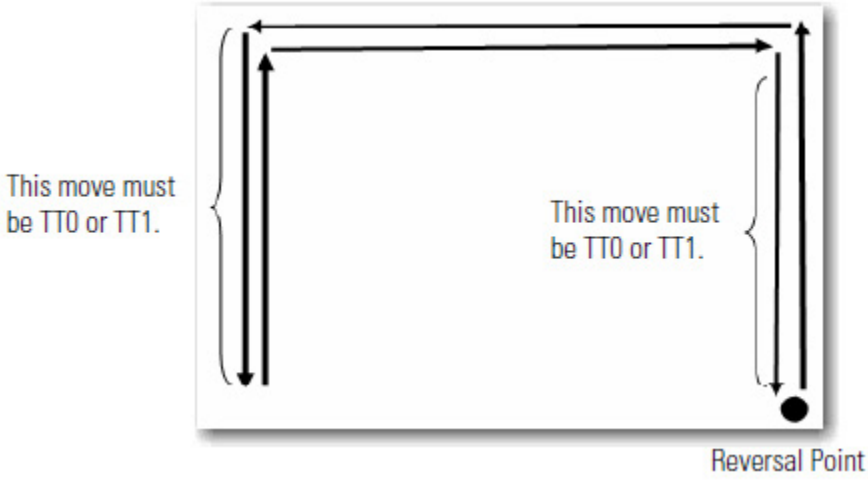
- MCLM 1 (point A to point B) is followed by MCLM 2 (point B to point C).
- MCLM 3 (point C to point B) is followed by MCLM 4 (point B to point A).
- The acceleration of MCLM 1 must be equal to the deceleration of MCLM 4.
- The deceleration of MCLM 1 must be equal to the acceleration a MCLM 4.
- The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

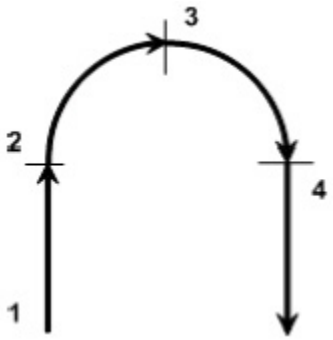
To guarantee that your trajectory is symmetric, you must terminate any sequence of moves by either Termination Types 0 or 1. You should also use a Termination Type of 0 or 1 at the Reversal Point of a profile that moves back on itself.



Using a TT2, TT3, TT4, TT5, ot 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.

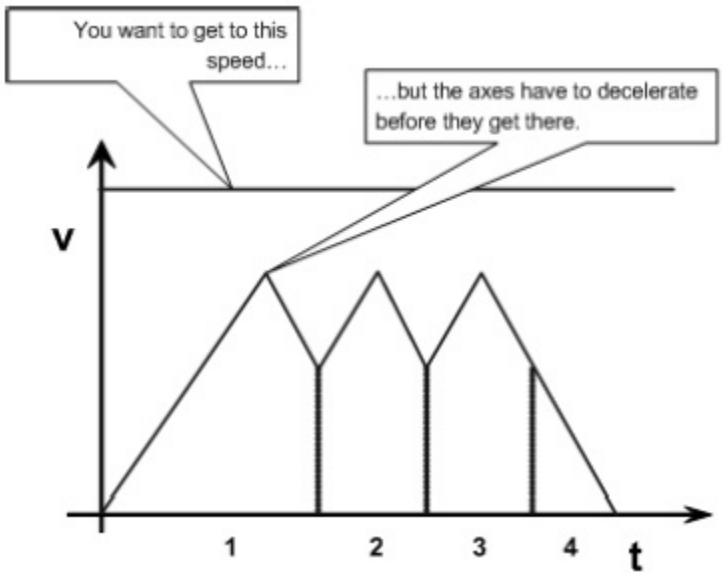
How To Get a Triangular Velocity Profile

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



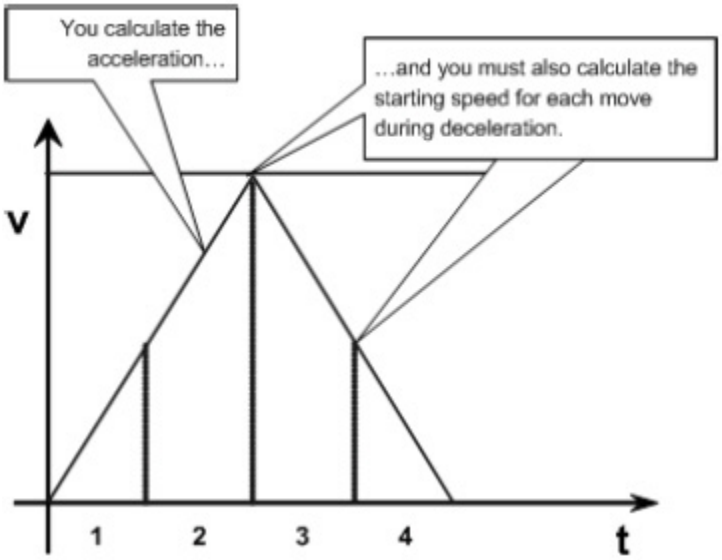
Then, use termination type 5. The other termination types may not let you get to the speed you want.

Termination Types 2, 3, 4, or 6



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.

Termination Type 5

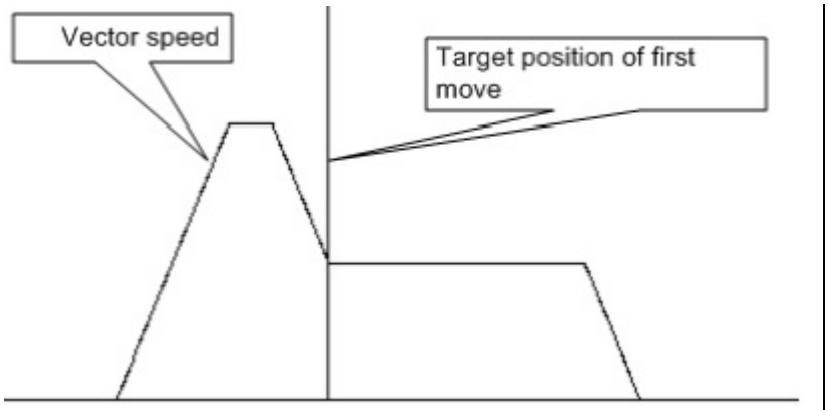
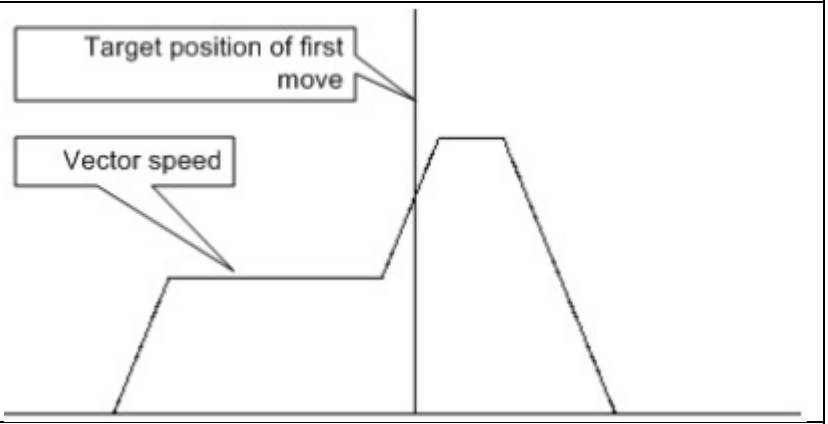
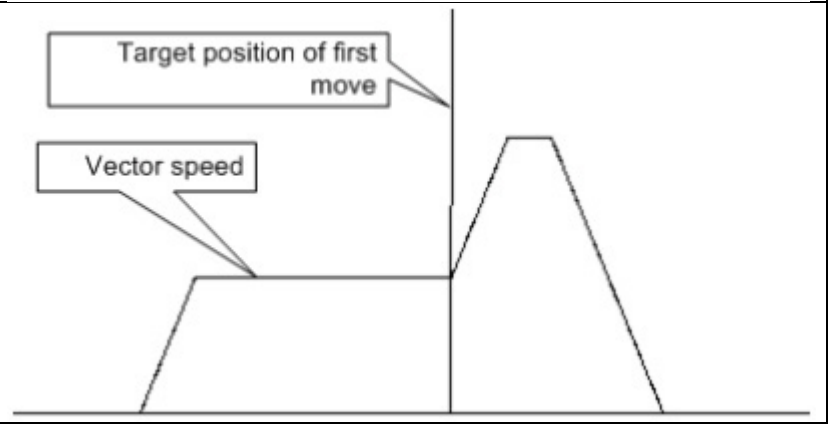


The axes accelerate to the speed that you want. You must calculate the starting speed for each move in the deceleration-half of the profile.

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.

If the next move is	And the Termination Type of the first move is	Then

Slower	2 - Command Tolerance 3 - No Decel 4 - Contour Velocity Constrained 5 - Contour Velocity Unconstrained 6 - Command Tolerance Programmed	
Faster	2 - Command Tolerance 3 - No Decel 6 - Command Tolerance Programmed	
	4 - Contour Velocity Constrained 5 - Contour Velocity Unconstrained	

Common Action Table for Slave Coordinate System and Master Axis

All commands in this table are for the Slave Coordinate System.

Instruction	Parameters	MDCC IP Bit
MGS		Reset
MGSD		Reset
MCS	Stop Type = Coordinated Motion	Not Changed
	Stop Type = Transform	Not Changed
	Stop Type = All	Reset
MCSD		Reset
MAS	Stop Type = Jog	Not Changed
	Stop Type = Move	Not Changed
	Stop Type = Time CAM	Not Changed
	Stop Type = All	Reset
MASD		Reset

MSF		Not Changed
MDF		Not Changed
Fault Action	Status Only	Not Changed
	Stop Motion	Reset
	Disable DRV	Reset
	Shutdown	Reset

When the same Slave Coordinate System is controlled by multiple Master Axes, if one MDCC relationship that contains the Slave Coordinate System is broken, then all MDCC relationships that contain the Slave Coordinate System are broken.

Common Action Table for Master Axis

All commands in this table are for the Master Axis.

Instruction	Parameters	MDCC IP Bit
MGS		Reset
MGSD		Reset
MCS	Stop Type = Coordinated Motion	Not Changed
	Stop Type = Transform	Not Changed
	Stop Type = All	Not Changed
MCSD		Reset
MAS	Any Stop Type (Jog, Move, Time CAM, All)	Not Changed
MASD		Reset
MSF		Not Changed
MDF		Not Changed
Fault Action	Status Only	Not Changed
	Stop Motion	Not Changed
	Disable DRV	Not Changed
	Shutdown	Reset

If the same Master Axis is controlling multiple Slave Coordinate System, then all MDCC relationships that contain the Master Axis are broken.

Input and Output Parameters Structure

The middle column of the table below identifies which parameter is applicable to each coordinate system motion instruction, that is, to MCLM and MCCM. Before any of the parameters identified in the first column below may be used in the MCLM or MCCM instruction, you must execute an MDCC instruction and it must be active (IP bit is set).

Parameter	Instruction	Mode
Input Parameters		
Lock Direction	MCLM, MCCM	Master Driven Only
Lock Position	MCLM, MCCM	Master Driven Only
Command Tolerance	MCLM, MCCM	Master Driven and Time Driven

Event Distance	MCLM, MCCM	All modes (Master Driven or Time Driven)

Output Parameter		
Calculated Data	MCLM, MCCM	All modes (Master Driven, Time Driven, and Timed Based)

Input Parameters

This table describes the input parameters.

Lock Direction

Data Type	Description	Valid Default Values
Immediate	This parameter is used for both Time Driven and Master Driven Mode. The controlling axis is the Master Axis (axis is programmed in the MDCC command) for Master Driven Mode and the axis that is programmed in the motion instruction (for example, MCLM) for Time Driven Mode. The first word of the Lock Direction enumeration definition (see enumeration table below) identifies the lock type as either: • Immediate (lock is performed immediately), or • Position (lock is performed when the Master Axis crosses the specified Lock Position). The second word of the enumeration specifies the direction in which the Master Axis has to be moving when it crosses the Lock Position for the lock to take effect. For an MCLM and MCCM instruction, the Slave Coordinate System always moves in one direction - its programmed direction - while it follows the Master Axis, regardless of the direction of the Master Axis. If the Master reverses, the Slave Coordinate System stops.	Valid = 0-4 Default = None (Enumeration 1-4 are currently not allowed in Time Driven mode.)

For Master Driven Mode, the enumerations are as follows:

(Forward is positive velocity, reverse is negative velocity.)

The enumerations table is shown.

Enumeration	Definition	Description
0	None	Indicates that the Lock Position is not active. If Lock Direction is set to None and the Master Driven mode is selected by the speed parameter of the motion instruction, the system will error. Conversely, if Lock direction is not set to a value other than None and the speed parameter units indicate Time Driven mode, an error is also generated.
1	Immediate Forward Only	Motion starts immediately when the Master is moving in the Forward Direction. The Master Axis is only followed while it is moving in the Forward Direction.
2	Immediate Reverse Only	Motion starts immediately when the Master Axis is moving in the Reverse Direction. The Master Axis is only followed while it is moving in the Reverse Direction.
3	Position Forward Only	Motion starts (that is, the Slave Coordinate System locks to the Master Axis) when the Master Axis crosses the Lock Position while it is moving in the Forward Direction. The Master Axis is only followed while it is moving in the Forward Direction. Note that if the start position equals the Lock Position and this enumeration is selected, then motion will not start because the Lock Position will not be crossed.

4	Position Reverse Only	Motion starts when the Master Axis crosses the Lock Position while it is moving in the Reverse Direction. The Master Axis is only followed while it is moving in the Reverse Direction. Note that if the start position equals the Lock Position and this enumeration is selected, then motion will not start because the Lock Position will not be crossed.

For Time Driven Mode, the enumerations are:

Enumeration	Definition	Description
0	None	Indicates that the Lock Position is not active.
1	Immediate Forward Only	The instruction will error with MDSC_LOCKDIR_CONFLICT (95).
2	Immediate Reverse Only	The instruction will error with MDSC_LOCKDIR_CONFLICT (95).
3	Position Forward Only	The instruction will error with MDSC_LOCKDIR_CONFLICT (95).
4	Position Reverse Only	The instruction will error with MDSC_LOCKDIR_CONFLICT (95).

Lock Position

Data Type	Description	Valid Default Values

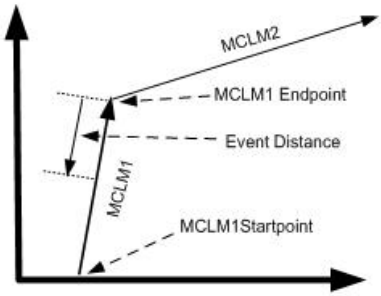
IMMEDIATE REAL or TAG	**Lock Position in Master Driven Mode** The position on the Master Axis where a Slave Coordinate System should start after the move has been initiated on the Slave Coordinate System when executing in Master Driven Mode. This is an absolute position (plus or minus) on the Master Axis in Master Axis units. You can specify a Lock Position to delay the start of motion of a Slave Coordinate System after the motion instruction has been initiated on the Slave Coordinate System. If an axis in the Slave Coordinate System is already moving and a coordinated move instruction (MCLM, or MCCM) with a Lock Position is activated on the Coordinate system, then you will receive an MDSC_LOCK_WHILE_MOVING error for the MCLM or MCCM instruction. Because Merge is always performed immediately when an instruction is enabled, a merge instruction that starts at a nonzero velocity with both a Lock Position and a Merge enabled will receive an MDSC_LOCK_WHILE_MOVING error. The Lock Direction determines the direction in which the Master Axis must be moving when it crosses the Lock Position before the Slave Coordinate System locks to the Master Axis. Note that if there is an unwind value specified on the Master Axis, then the Lock Position must be between 0 and the unwind value (that is, the Lock Position cannot be more than one unwind.) This parameter is only used in Master Driven Mode. **Lock Position in Time Driven Mode** There is no Lock Position in Time Driven Mode for a coordinate system. An error will be generated if the Lock Direction is not NONE and the system is in Time Driven Mode for an MCLM or MCCM. This parameter is only used in Master Driven Mode. **Axis Lock Behavior** When the Master axis crosses the Master Lock position in the direction as specified by the motion instruction, the Slave Coordinate System becomes locked to the Master axis. The LockStatus bit is set at this time. The MCLM and MCCM instructions on the Slave Coordinate System in MDSC mode go IP as soon as they reach the head of the motion queue. The head of the queue is defined as the move right before the active move. For the Immediate Forward Only or Immediate Reverse Only Lock Directions, the Slave Coordinate System gets locked to the Master Axis immediately when the MCLM or MCCM instruction is executed (goes IP). For the Position Forward Only or Position Reverse Only Lock Directions, the slave gets locked to the Master Axis when the Master Axis crosses the Master Lock Position in the direction as specified by the motion instruction. In either case, the LockStatus bit is set when the lock occurs. Because there is no bi-directional behavior defined, once locked, the Slave Coordinate System follows the Master only in the specified direction. If the Master reverses direction, then the Slave stops following the Master. Note that the LockStatus bit remains set until the Master decelerates to zero. It is cleared at the point of reversal of the Master axis. The Slave does not follow the Master while the Master travels in the reverse direction. If the Master axis changes directions again, then the axis LockStatus bit is set again when the Slave Coordinate System crosses the original reversal point, at which time the slave resumes following the Master Axis. On the Slave Coordinate System the following restrictions apply: • If a new instruction succeeds the active motion instruction but	Default = 0.0

Command Tolerance

Data Type	Description	Valid Default Values
IMMEDIATE REAL or TAG	The position on a coordinated move where blending should start. When Termination Type 6 is used, the Command Tolerance on the instruction faceplate is used instead of the value for the Command Tolerance that is configured in the Coordinate System.	Valid = 0.0

Event Distance

Data Type	Description	Valid Default Values

ARRAY or 0 (The array must be a minimum size of 4. If the array is greater than 4, only the first four locations specified are used.)	The position(s) on a move measured from the end of the move. This is an array of input values that specifies the incremental distances along the move on the Slave Coordinate System. Each member of the array is measured as follows: • Distances are measured starting from the end of the move towards the beginning of the move as shown in the following Figure.  • For a linear coordinated move instructions (MCLM), the parameter value in the Event Distance can be represented as a vector starting at the move's end point and pointing towards the beginning of the move. • For a circular coordinated move (MCCM), the parameter value in the Event Distance is an incremental distance measured along the circular arc (that is, arc length) starting at the move's endpoint and moving towards the beginning of the circular arc. If the value in the Event Distance array is 0.0, then it is the time or distance for the whole move. The values entered in the Event Distance array are the same for both Time Driven and Master Driven Mode. Only the returned values in the Calculated Data array are different depending on the programmed mode of the Slave Coordinated System. When Event Distance is specified as a negative number, then the Event Distance calculation is skipped and a -1 is returned in the Calculated Data array for the specified Event Distance parameter. There is no limit on the dimension of either the Event Distance or Calculated Data arrays. However, only a maximum of 4 elements (the specified value and the next 3) of the Event Distance array will be processed. Note that special consideration for the rare case of an overshoot when a MCD or MCCD is done close to the moves endpoint. For this case, the Calculated Data will include the overshoot when the Event Distance is 0, since the master will have to traverse this amount for the move to finish. For other Event Distances, the overshoot will not be included.	Default = 0 (no Event Distance array)

Output Parameters

This describes the output parameters.

Calculated Data

Data Type	Description	Valid and Default Values

	REAL ARRAY or 0	This is the Master Distance(s) (or time) needed for the Slave Coordinated System to travel from the beginning of the move to the Event Distance point. The returned Calculated Data value is dependent on: • the instruction type, (that is, MCLM or MCCM for coordinated motion) • the mode of the Slave Coordinate System (that is, Time Driven or Master Driven). • if superimposed motion is active, the Calculated Data does not include any of the superimposed motion. To understand the Calculated Data concept, it's important to understand that the Motion Start Point (MSP) for a coordinated move is defined as the last time that: • a TT0/TT1 was programmed, or completed or • the queue was empty, or • a merge occurred. If there was a dwell programmed in the queue, then the calculated data includes the time of the dwell. Note that the MSP could have occurred several moves prior to the move in which the Event Distance was specified. The Logix Designer application Motion Planner processes and calculates output data and places the result in the Calculated Data array as supplied in the instruction. The number of calculated array elements stored in the Calculated Data array is based on the follow conditions: • The number of elements in the Event Distance array. • For each of the first 4 elements Event Data array, one element will be computed and placed in the Calculated Data array. • The fifth element and beyond of the Event Distance array are ignored. Existing values in the Calculated Data array are overlaid when the Event Distance array is processed. A -1 will be returned in the Calculated Data array for each negative value in the Event Distance array. No Event Distance calculation is made for these array elements. You can change the Event Distance array elements dynamically in the program. However, if the Event Distance is changed after the instruction has been initiated (that is, the IP bit has been set), then the change is ignored. An error is generated if the size of the Calculated Data array is smaller than the Event Distance array. If the Event Distance is greater than the move length internally (vector length for MCLM, arc length for MCCM), it will be forced to equal the move length. If a MCD or MCCD is executed (indicated by status bit going IP), the CalculatedDataAvailable (CDA) bit will be cleared. The Calculated Data for the move will be recomputed using the new dynamics parameters. Only those items of the Calculated Data array whose Event Distance hasn't been reached yet are recomputed; other items are left as they are. Consequently, all Calculated Data array items contain valid information after the move is completed. The CDA bit will be set again when computations are complete, The Calculated Data that is recomputed will be measured from the original MSP to the Event Distance point using the new dynamics parameters as changed by the MCD or MCCD instruction, not from the point of the MCD or MCCD. Note that if the MCD changes the speed to 0, the Event Distance will not be recomputed; the CDA bit will be cleared and stay cleared. The Event Distance will however be recomputed if a second MCD or MCCD is issued to restart the motion. The recomputed Calculated Data will include the duration of the stopped motion.	Default = 0 (no Calculated Data array) or a REAL array tag

Calculated Data Example 1

Event Distance array = [11, 22, -5, 23, 44]

Calculated Data array = [f(11), f(22), -1 ,f(23)]

Where f is the calculated data function.

- Tip: The 44 is ignored because it is the fifth element in the Event Distance array. Nothing is returned in the corresponding 5th array element of Calculated Data array.
- A -1 is returned in the third element of the Calculated Data array because the corresponding Event Data Array element is negative.

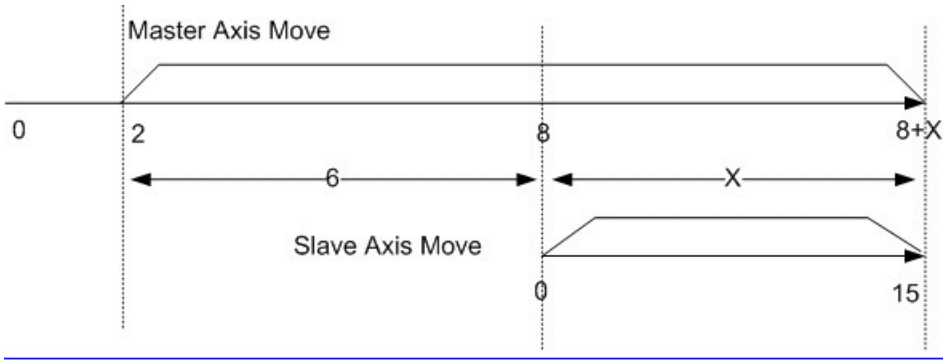
Calculated Data Example 2

Assume that the master axis is at a position of 2.0. The slave is programmed to an incremental value of 15.0 with a

Master Lock Position at 8.0. The Event Distance is set to 0.0, which means that we want the total

Master Distance (X in the diagram) needed for the slave to move 15.0 units starting when the Master is locked at a

position at 8.0. The incremental value of X is returned in the Calculated Data parameter.

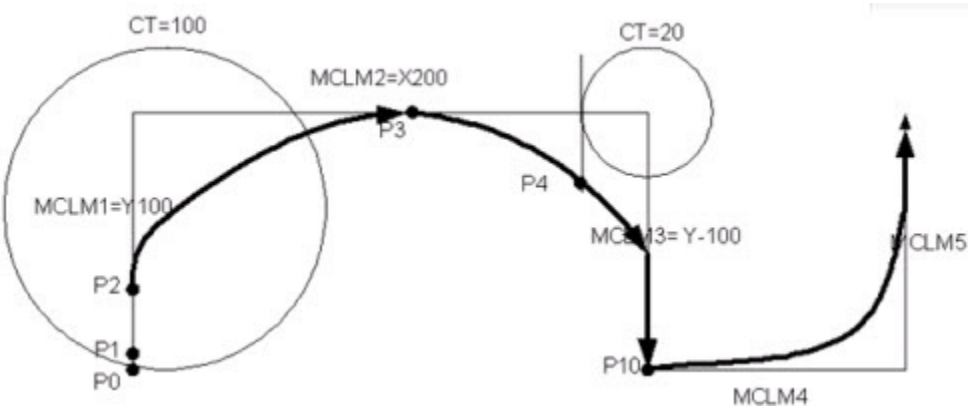


Calculated Data Example 3

This example illustrates using Event Distance and Calculated Data.

Note that the MSP for all event distances is point P0. The MSP is where the Slave is locked to the

Master and starts moving along the programmed path.



5 Move Segments are specified

Event Distance = ED

Command Tolerance = CT

- MCLM1 Y100; TT2 ED=50 CT=100
- MCLM2 X200; TT2 ED=100 CT=20
- MCLM3 Y-100; TT1 ED=100 CT=20
- MCLM4 X200; TT2 ED=100 CT=20
- MCLM5 Y100; TT2 ED=100 CT=20

The calculated data for MCLM1 is returned when MCLM2 is added to the queue and planned.

This will be at point P1 above. Master Distance is Measured from point P0.

The calculated data for MCLM2 is returned when MCLM3 is added to the queue and planned.

This will be at Point P2 above. Master Distance is Measured from point P0.

The calculated data for MCLM3 is returned when MCLM3 is added to the queue and planned.

This will be at point P3 above. Master Distance is Measured from point P0.

The calculated data for MCLM4 is returned when MCLM5 is added to the queue and planned.

This will be at point P10 above. Master Distance is Measured from point P10.

The calculated data for MCLM5 is never returned because MCLM5 is terminated with TT2 and it is

the last move in the queue. You should use TT0 or TT1 instead.

All Calculated Data is the master distance or time from the last MSP point.

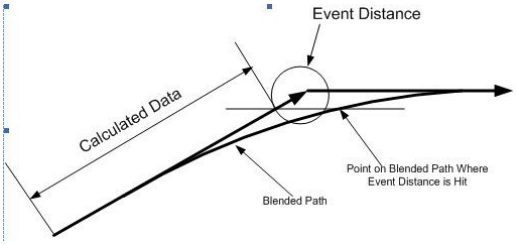
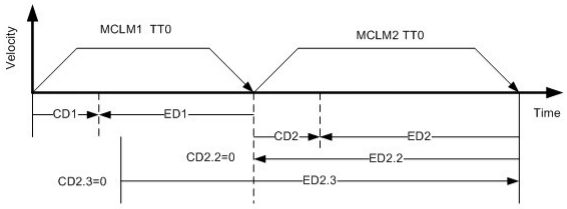
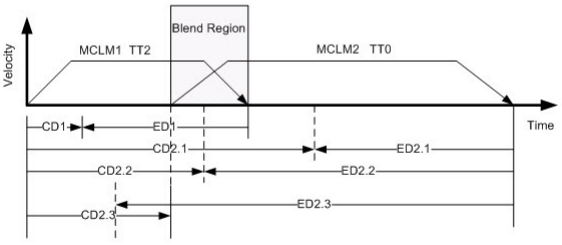
That is, it is where the slave is at rest, which is point P0 and point P10 above.

See also

Returned Calculated Data Parameter for Coordinated System Motion Instruction on page 512

Returned Calculated Data Parameter

The returned Calculated Data value for Coordinated System Motion instructions is outlined in the table.

Mode	Returned Calculated Data Parameter
Master Driven	The returned Calculated Data parameter is the incremental delta Master position that is needed to make the Slave Coordinate System move from the point at which Slave Coordinate System is locked to the Master and starts moving along the programmed path to the point where distance to go is less than the specified Event Distance. (See Example 3. In example 3, the MSP for all event distances is point P0.)  • For Blended moves (that is, Termination Type =Command Tolerance or No Decel) The incremental Master Axis distance needed for the programmed move, in the Slave Coordinate System, to travel from the beginning of the move to the Blend Point. Note that this is where the PC bit of the instruction is set. • For all other termination types (that is, non-blended moves) The incremental Master Axis distance needed for programmed move, in the Slave Coordinate System, to travel from the beginning of the move to the programmed endpoint. Note that this is where the PC bit of instruction is set on the instruction moving the slave Another way to represent the Event Distance and the corresponding Calculated Data is on a Velocity versus Time plot as is shown in the following figure: Note that the first plot below is for non-blended moves (TT0/1), the second is for blended (TT2, 3, 6).   CD = Calculated Data ED = Event Distance TT = Termination Type CD2.1 = Calculated Data 1 for Move 2 ED2.3 = Event Distance 3 for Move 2 CD2.1 is how far the master has to move when the slave reaches position ED2.1 **Note**: As shown above, the Event Distance is > Move length and is therefore internally set equal to the move length. The Calculated Data for the move length is therefore returned. No error is forced.
Time Driven	The returned data in the Calculated Data parameter is the total time in seconds that is needed to make the Slave Coordinate System move from the move's start point to a point where distance to go is less than the specified Event Distance. If the specified data in the Event Distance is array element is 0.0, then the time it takes the entire move to complete is returned.

See also

Input and Output Parameters Structure for Coordinate System Motion Instructions on page 527

Master Driven Speed Control Functionality

The Master Driven Speed Control (MDSC) function provides the ability to synchronize one or more motion axes to a common Master Axis. The MDSC uses the Motion Master Driven Axis Control (MDAC) instruction, which assigns a Master:Slave relationship for single axes.

The MDSC function provides synchronization of axes without the use of CAM profiles. The MDSC functions can also be used with Time Driven CAM profiles

(MATC) to make them act like position driven profiles synchronized with a Master Axis.

Master Driven Axis Control (MDAC)

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

The Master Driven Speed Control (MDSC) function provides the ability to synchronize one or more motion axes to a common Master Axis. The MDSC uses the Master Driven Axis Control (MDAC) instruction, which assigns a Master:Slave relationship for single axes.

Using the MDSC function provides synchronization of axes without the use of CAM profiles. The MDSC functions may also be used with Time Driven CAM profiles (MATC) to make them act like position driven profiles synchronized with a Master Axis.

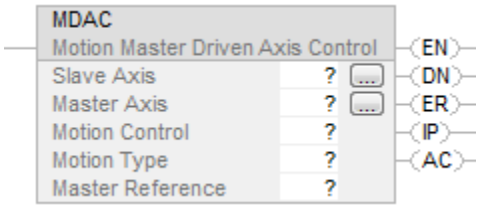
This is a transitional instruction. Follow these steps when using it:

- In ladder logic, insert an instruction to toggle the rung-condition-in from false to true each time the instruction should execute.
- In a Structured Text routine, insert a condition for the instruction to cause it to execute only on a transition.

Architecture	Standard	Safety
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Yes	No
Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers	Yes	No

Available Languages

Ladder Diagram



Function Block

This instruction is not available in function block.

Structured Text

MDAC (SlaveAxis, MasterAxis, MotionControl, MotionType, MasterReference);

Operands

Ladder Diagram

Operand	Type CompactLogix 5370, Compact GuardLogix 5370, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480	Type ControlLogix 5570, GuardLogix 5570, ControlLogix 5580, and GuardLogix 5580 controllers	Format	Description
Slave Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL	AXIS_SERVO AXIS_SERVO_DRIVE AXIS_GENERIC_DRIVE AXIS_CIP_DRIVE AXIS_VIRTUAL	Tag	The Single Axis being controlled by the Master Axis when the motion planner is in Master Driven mode. Ellipsis launches Axis properties dialog. The MDAC link is broken when the following instructions are executed: On the Slave Axis: MAS (All), MCS (All), MGS, MASD, MCSD, MGSD, a mode change. Note that MAS (anything other than All) and MCS (coordinated) do NOT break the MDAC link. The Shutdown instructions (MGSD, MASD, MCSD) never go IP. On the Master Axis: MASD, MCSD, and MGSD. Note that MAS and MCS for any Stop Type, including All, do NOT break the MDAC link. A mode change (Rem Run to Rem Prog or Rem Prog to Rem Run) or an axis fault also breaks the MDAC link. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.

Master Axis	AXIS_CONSUMED AXIS_CIP_DRIVE AXIS_VIRTUAL **Tip:** AXIS_CONSUMED is supported by Compact GuardLogix 5580, CompactLogix 5380, and CompactLogix 5480 controllers only.	AXIS_CONSUMED AXIS_SERVO AXIS_SERVO_DRIVE AXIS_GENERIC_DRIVE AXIS_CIP_DRIVE AXIS_VIRTUAL	Tag	Any configured Single Axis that the Slave Axis follows. The Master Axis can be any axis that has been configured. Ellipsis launches Axis properties dialog. For controllers that support the REF_TO motion data types, the supported axis operand type can be replaced by an equivalent REF_TO type.
Motion Control	MOTION_INSTRUCTION	MOTION_INSTRUCTION	Tag	Control tag for the instruction.
Motion Type	UNIT32	UNIT32	Immediate or Tag	Specifies the move type (MAM, MAJ, MATC, or MAM - Master Offset Move for a Position Cam) executing on the Slave Axis that will be controlled by the Master Axis when a single axis motion instruction is programmed in Master Driven Mode. The enumerations for Motion Type are: All = 0 Move = 1 Jog = 2 Time Cam = 3 Master Offset Move = 4
Master Reference	UNIT32	UNIT32	Immediate or Tag	Selects the Master Axis position source as either Actual Position (0) or Command Position (1).

The All for the motion type lets the system know that Master Driven Mode applies to all single axis instructions (that is, MAM, MAJ, MATC, and a MAM (Master Offset Move)). If the Motion Type of the active MDAC is set to anything other than All on a Slave Axis, then executing a MDAC All on the Slave Axis causes a runtime error. For example:

MDAC Move=MAM Master=X Slave=Y - Assigns master and slave axes

MDAC Move=ALL Master=Z Slave=Y - Causes an error because the previous MDAC assigned the MAM to the Y axis. Note that assigning the Slave Y axis to any other Master axis causes the same error because Y is already assigned to a MAM.

To be able to reassign the Master once the All option is used, it is sufficient to execute one of the following for either the Slave Axis or the Master Axis,

although you can also choose to execute one of the following for both the Slave Axis and the Master Axis:

- On the Slave Axis: MAS (All), MCS (All), MGS, MASD, MCSD, or MGSD
- On the Master Axis: MASD, MCSD, or MGSD

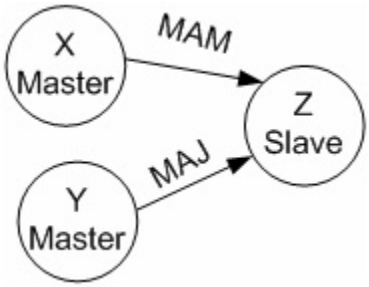
If there is an active MDAC (All) on a Slave, then setting the MDAC instruction to a Motion Type other than All causes a runtime error.

To be able to reassign the Master once anything other than the All option is used, you must execute any of the following:

- on the Slave Axis: MAS (All), MCS (All), MGS, MASD, MCSD, or MGSD
- On the Master Axis: MASD, MCSD, or MGSD

If you assign the same axis to be both a Master and a Slave Axis, an RSLogix 5000 software verification error is generated.

The same Slave axis can be assigned to 2 different Master axes. For example, if X, Y are the Master axes and Z is a Slave axis, then X can be assigned to be the Master axis for MAM instructions for Z and Y is the Master axis for MAJ instructions for Slave Z. Refer to the following illustration.



The Master Axis for an active Slave may be changed by executing a MDAC for the slave with a different Master. The MDAC puts the pending Master in a queue for the Slave. When the next single axis motion instruction is executed (goes IP) in either Replacement or Merge mode, the queued Master Axis becomes the Master for the Slave the next time a single axis motion is started. For example: In this example, the masters are already running. MDAC Move=MAM Master=X Slave=Z	Assigns master and slave axis
MAM X	Start MAM motion on the master 1
MAM Z	Start MAM motion on the slave
MAM Y	Start MAM motion on the Y master 2
MDAC Move=MAM Master=Y Slave=Z	Y is pending to be the new master
MAM Z	Start new MAM motion on the slave using new master 2

Use different control words on all MDAC instructions. If the same control word is used for the active and pending MDAC, the IP bit of the pending

MDAC will not work properly. When separate axis control words are used for the active and the pending MDAC instructions, the active and the pending MDAC will both have their IP bits set. However, only the active MDAC will have its AC bit set. This is standard operation for all instructions. No detection of the same control word used on multiple instructions is made.

Master Reference

The Master Reference for an MDAC instruction selects the Master Axis position source.

The enumerations for Master Reference Axis are:

Actual – Slave motion is generated from the actual (current) position of the Master Axis as measured by its encoder or other feedback device.

Command – Slave motion is generated from the command (desired) position of the Master Axis.
Because there is no Command Position for a Feedback Only Axis, if you select either Actual or Command for Master Reference, the Actual Position of the Master Axis is used. The Actual and Command Position are always the same for this case. No error is generated.
Because there is no Actual Position for a Virtual axis, if you select either Actual or Command for Master Reference, the Command Position is used. No error will be generated.

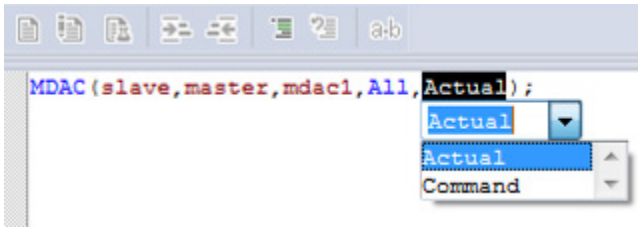
An error is generated if a MDAC instruction is executed that changes the Master Reference of a slave axis that is in motion. The new MDAC instruction will error and the original one will remain active.

Structured Text

Operand	Type	Format	Description
Source	SINT INT DINT REAL STRING structure	tag	Initial element to copy Important: the Source and Destination operands should be the same data type, or unexpected results may occur
Destination	SINT INT DINT REAL STRING LINT structure	tag	Initial element to be overwritten by the Source Important: the Source and Destination operands should be the same data type, or unexpected results may occur
Length	DINT	immediate or tag	Number of Destination elements to copy

See Structured Text Syntax for more information on the syntax of expressions within structured text.

Note that you have the option to browse for enumerations in the Structured Text Editor as shown in the following.



MOTION_INSTRUCTION Bit Leg Definitions for MDAC

Mnemonic	Description
.EN (Enable) Bit 31	The enable bit is set when the rung transitions from false-to-true and stays set until the rung goes false.
.DN (Done) Bit 29	The done bit is set when the Master Driven Axis Control instruction is successfully initiated.
.ER (Error) Bit 28	The error bit is set when there is an invalid combination of parameters in the MDAC instruction.
.IP (In Process) Bit 26	The in process bit is set when the MDAC instruction is activated and reset by an instruction (for example, the MASD instruction).
.AC (Active) Bit 23	The active bit is set when a move (MAJ, MAM or MATC) goes IP in Master Driven mode on the axis that is selected as the Slave Axis of the MDAC instruction. If the Slave Axis is started in Time Driven mode, then the AC bit of the MDAC does not go active. The IP bit of the MDAC instruction does not change at this time.

Description

The MDAC instruction is used to select a single axis as a Master Axis and a single axis as a Slave Axis. The MDAC instruction connects a Master and Slave Axis for a MAM, MAJ, and MATC instruction. When an MDAC is executed (goes IP), the specified Slave Axis in the MDAC instruction is logically geared with the designated Master Axis. After motion on both the Master and Slave Axes has been initiated, the Slave follows the Master Axis motion using the programmed dynamics of the motion instruction.

There are no changes in any active motion when a new MDAC instruction is activated. Activating a new MDAC instruction puts the parameters programmed in the MDAC into a pending state. The parameters in the pending MDAC instruction are overridden if you execute a succeeding MDAC before a new single axis motion instruction (MAM, MAJ, and MATC) is activated on the slave axis. The most recent values in the pending MDAC instruction are used when a new single axis motion instruction is activated on the Slave Axis.

A MATC with an Execution Schedule of Pending always uses the Instruction mode of the active MATC; Lock Position and Lock Direction are ignored. This is explained further in the following example.

Assume that there are four instructions executed in the following order:

- MDAC1: Master=X, Slave=Y
- MATC1: Slave Y, Immediate
- MDAC2: Master=Z Slave=Y MDAC2 is executed (goes IP) while MATC1 is moving (Master Axis is changed)
- MATC2: Slave=Y, Pending
- MATC2: will use the parameters from MDAC1 (Master and Slave) because MATC1 was moving (active) when MDAC2 was executed. If MATC2 and MDAC2 were executed in the reverse order, you get the same results. This follows the existing paradigm of MAPC instruction where the pending CAM uses the same master as the running CAM.

Motion Direct Command and the MDAC Instruction

To obtain Motion Direct support for Master Driven Speed Control mode, you must first program an MDAC in one of the supporting program languages before you execute an MAM or MAJ in Time driven Mode.

Affects Math Status Flags

No

Major/Minor Faults

None specific to this instruction. See the Common Attributes for operand-related faults.

Execution

Ladder Diagram

Condition/State	Action Taken
Prescan	The .EN, .DN, .ER, and .IP bits are cleared to false.
Rung-condition-in is false	The .EN bit is cleared to false if either the .DN or .ER bit is true.
Rung-condition-in is true	The .EN bit is set to true and the instruction executes.
Postscan	N/A

Structured Text

Condition/State	Action Taken
Prescan	See Prescan in the Ladder Diagram table.
Normal execution	See Rung-condition-in is false in the Ladder Diagram table.
Postscan	See Postscan in the Ladder Diagram table.

Error Codes

See Motion Error Codes (.ERR) for run time errors for motion instructions when MDAC is active.

Verification Errors

An invalid or No Master Axis will cause new errors to be generated when verified by the programming software. The following conditions may cause this error:

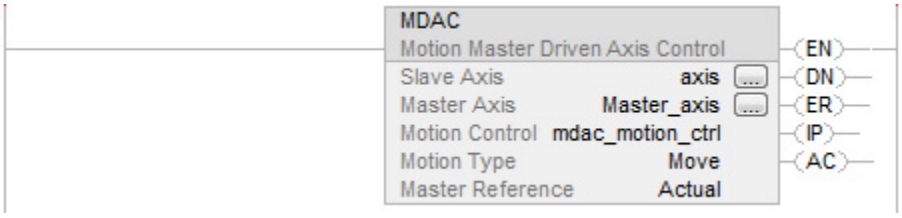
- The Master and Slave Axis are the same.
- Master or Slave Axis is not configured.
- Master or Slave Axis is inhibited.
- Redefine position is in progress.
- Home of an axis is in progress.

MDSC ALL Conflict. The ALL parameter is not allowed while a Slave axis motion generator (for example, the jog motion generator) is already assigned.

Examples

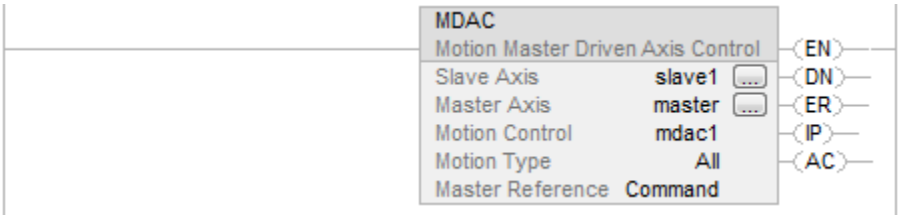
Example 1

Ladder Diagram



Example 2

Ladder Diagram



See also

Common Action Table for Slave and Master Axis on page 552

Motion Error Codes (.ERR) on page 581

Motion Move Instructions on page 73

Structured Text Syntax on page 665

Common Attributes on page 691

Change between Master Driven and Time Driven Modes for Single Axis Motion instructions

Changing the motion mode between Master Driven and Time Driven mode and vice versa is automatically performed when a new motion instruction, such as, MAM, MAJ, or MATC, is activated and the new instruction is programmed in a different mode than the active motion instruction.

When the new motion instruction is activated, the system assumes that the desired mode for the new instruction is the mode (Master Driven or Time Driven) as specified in the programmed units of the speed parameter contained in the new instruction. At all times, including when changing modes, the Accel, Decel, and Jerk must all be programmed in the same units as the Speed parameter or the instruction receives a MDSC_UNITS_CONFLICT_ERROR error.

A runtime MDSC_INVALID_MODE_OR_MASTER_CHANGE error occurs only if you attempt to change from Time Driven mode to MDSC mode, or vice versa with an MCD instruction.

If changing from Time Driven mode to Master Driven mode while an axis is moving and Lock Direction is not Immediate Forward or Reverse your receive error 95, MDSC Lock Direction Conflict.

If the master and slave axes are idle or paused, the MAM or MAJ can make a change on the slave. However, the error MDSC IDLE_MASTER_AND_SLAVE_MOVING is generated if MDSC mode is started while the slave is moving when the master is idle.

Different Time Driven and Master Driven modes can be used for different motion types for superimposed motion. For example, the MAM can be in Time Drive mode and the MAJ can be in Master Driven mode for the same Slave Axis.

Changing the Master Axis

The following sequence of events must be followed to transfer a Slave Axis from one Master Axis to a second Master Axis:

- First, you must execute an MDAC instruction to reassign the Slave Axis from the first Master Axis to the second Master Axis. This makes the reassignment pending. The IP bit of the pending MDAC instruction is set as an indication of the pending reassignment.
- Second, you must execute a new motion command (for example, an MAM or MAJ). The axis becomes unlocked from the first Master Axis and reassigned to the second Master Axis when this new motion instruction is executed (goes IP).

  Example: The MDAC mode becomes enabled once the MDAC instruction is executed.
  The MDAC mode becomes active after being enabled and a motion instruction with the MDAC speed unit selection is executed.
  If there is an active MDAC (All) on a Slave, then setting the MDAC instruction to a Motion Type other than All causes a runtime error.
  To reassign the Master once anything other than the All option is used, you must execute any of the following:
  - On the Slave Axis: MAS (All), MCS (All), MGS, MASD, MCSD, or MGSD
  - On the Master Axis: MASD, MCSD, or MGSD

  If you assign the same axis to be both a Master and a Slave Axis, an RSLogix 5000 software verification error is generated.

If the slave is not moving when the master axis is changed, no problem ever occurs. However, if the slave is in motion when the change in master axes occurs then the final effective slave speed is computed as the product of the Master Axis speed and the slave programmed speed. If the new final effective Slave Axis speed is less than 5% of its original speed after a change in the Master Axis, then the change is not be allowed and the MDSC_INVALID_SLAVE_SPEED_REDUCTION error occurs. This is always the case if the second master axis is idle (velocity=0). In this case, the motion instruction making this request receives an MDSC_IDLE_MASTER_AND_SLAVE_MOVING error.

If the second Master Axis is moving while the change in the master axis is being made, look at the TrackingMaster instruction status bit of the Motion Control words. The Motion Control words of the motion instruction that is performing the change displays when the change in masters is finished.

There are two reasons why the slave might accelerate. One, when the master driving the slave is accelerating and two, when the slave is accelerating due to the programmed acceleration in a new instruction.

When the acceleration of the Tracking Master bit in the instruction Control Word is cleared, it is not sensitive to the acceleration due to the master being accelerated. The master is only sensitive to the acceleration of the slave caused by a programmed change in the speed of two successive instructions.

Actions Taken When Stopping/Shutdown Instructions are Executed on the Slave Axis.

Action Table for Slave and Master Axis

All commands in the following table are for the Slave Axis system. The following table identifies the change in state of the MDAC link between the master and slave axis as each instruction in column 1 is executed on the slave axis.

Instruction	Parameters	MDAC IP Bit
MGS		Reset
MGSD		Reset
MCS	Stop Type = Coordinated Motion	Not Changed
	Stop Type = Transform	Not Changed
	Stop Type = All	Reset
MCSD		Reset
MAS	Stop Type = Jog	Not Changed AC bit for the MDAC is set because a MAJ is active, then the AC bit will be reset.
	Stop Type = Move	Not Changed AC bit for the MDAC is set because a MAM is active, then the AC bit will be reset.
	Stop Type = Time CAM	Not Changed AC bit for the MDAC is set because a MATC is active, then the AC bit will be reset.
	Stop Type = All	Reset AC bit for the MDAC is set because any single axis motion command is active then the AC bit will be reset.
MASD		Reset
MSF		Not Changed
MDF		Not Changed
Fault Action	Status Only	Not Changed
	Stop Motion	Reset
	Disable DRV	Reset
	Shutdown	Reset

If the same Slave Axis or a Slave Coordinate System is controlled by multiple Master Axes. If one MDAC or MCCD relationship that contains the Slave or Slave Coordinate System is broken, then all MDAC or MDCC relationships that contain the Slave or Slave Coordinate System are broken.

The MDAC link is broken when the following instructions are executed on the slave axis:

- MAS (All), MCS (All), MGS, MASD, MCSD, MGSD, a mode change.

- The MAS (anything other than All) and MCS (coordinated) do not break the MDAC link.

A mode change (Rem Run to Rem Prog or Rem Prog to Rem Run) or an axis fault also breaks the MDAC link.

For information on the MDCC instruction and the other coordinate instructions, see the Coordinate System User Manual, publication MOTION-UM002.

Actions Taken When Stopping/Shutdown Instructions are Executed on the Master Axis

The following table identifies the change in state of the MDAC link between the master and slave axis as each instruction in column 1 is executed on the master axis. All commands in the following table are for the master axis.

Actions for the Master Axis

Instruction	Parameters	MDAC IP Bit
MGS		Reset
MGSD		Reset
MCS	Stop Type = Coordinated Motion	Not Changed
	Stop Type = Transform	Not Changed
	Stop Type = All	Not Changed
MCSD		Reset
MAS	Any Stop Type Jog Move Time CAM All	Not Changed
MASD		Reset
MSF		Not Changed
MDF		Not Changed
Fault Action	Status Only	Not Changed
	Stop Motion	Not Changed
	Disable DRV	Not Changed
	Shutdown	Reset

If the same Master Axis is controlling multiple Slaves or a Slave Coordinate System, then all MDAC or MDCC relationships that contain the Master Axis are broken.

The MDAC link is broken on the Master Axis when the following instructions are executed on the Master Axis:

- MASD, MCSD, and MGSD. The MAS and MCS instructions for any
- Stop Type, including All, do not break the MDAC link.
- A mode change or an axis fault also breaks the MDAC link.

Input and Output Parameters Structure

Before any of the MDSC parameters, identified in table below, are used by the instructions MAM, MAJ or MATC, execute an MDAC instruction containing the axis as a slave and make sure it is active. This table identifies which parameter is applicable to the following single axis motion instructions, that is, to MAM, MAJ or MATC and the mode it is applicable in.

MDSC (Master Driven Speed Control) Parameter Structure

Parameter	Instruction	Mode
Input Parameters		
Lock Direction	MAM, MAJ, MATC	Master Driven Only
Lock Position	MAM, MAJ, MATC	Master Driven Only
Event Distance	MAM	All modes (Master Driven or Time Driven)
Instruction Mode	MATC	Identifies the mode of the instruction to be either Time Driven (0) or Master Driven mode (1)
Output Parameter		
Calculated Data	MAM	All modes (Master Driven, Time Driven, and Timed

MDSC Lock Direction Parameter Description

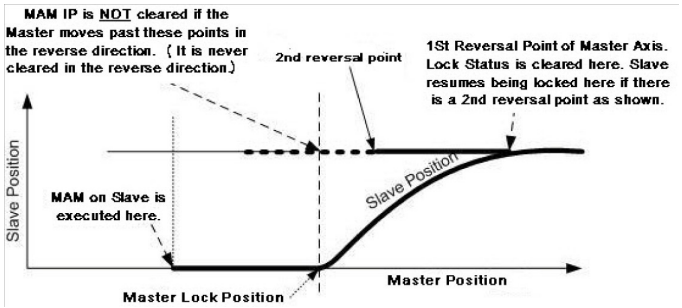
Input Parameter	Data Type	Description	Value
Lock Direction	Immediate	This parameter is used for both Time Driven and Master Driven mode. In master driven mode the Lock Direction is used by the axis specified as the Master Axis in the MDAC instruction. It is not used in Time Driven mode. The first word of the Lock Direction (Immediate or Position) enumeration definition identifies the lock type as either: • Immediate (lock is per formed immediately) • Position (lock is per formed when the Master Axis crosses the specified Lock Position) The second word of the Lock Direction (Forward or Reverse) specifies the direction in which the Master Axis has to be moving when it crosses the Lock Position for the lock to take effect. Forward is positive velocity, reverse is negative velocity. The instruction will error with MDSC_LOCKDIR_CONFLICT (95). The enumeration NONE must be used in time driven mode or the instruction errors. On of the other 4 enumerations must be used in master driven mode. For an MAM instruction, the Slave always moves in one direction -- the programmed direction. Once it starts moving it follows the Master Axis in the programmed direction, regardless of the direction of the Master Axis. If the Master reverses, the slave stops. These are the Master Driven mode enumerations.	Default = 0 0 = None 1 = Immediate For ward Only 2 = Immediate Reverse Only 3 = Position For ward Only 4 = Position Reverse Only

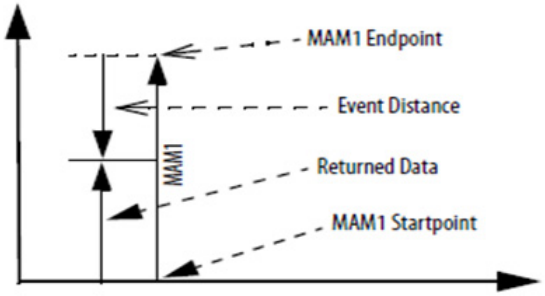
Enumeration Descriptions for the MDAC Input Parameter

Enumeration	Name	Description
0	None	Indicates that the Lock Position is not active. If Lock Direction is set to None and the Master Driven mode is selected by the speed parameter of the motion instruction, a run time error is generated. If Lock direction is set to a value other than None and the speed parameter units indicate Time Driven mode, an error is also generated.
1	Immediate Forward Only	Motion starts immediately when the Master is moving in the Forward Direction. The Master Axis is followed only while it is moving in the Forward Direction.
2	Immediate Reverse Only	Motion starts immediately when the Master Axis is moving in the Reverse Direction. The Master Axis is followed only while it is moving in the Reverse Direction.
3	Position For ward Only	Motion starts, for example, the Slave locks to the Master Axis when the Master Axis crosses the Lock Position while it is moving in the Forward Direction. The Master Axis is followed only while it is moving in the Forward Direction. If the start position of the Master Axis equals the Lock Position and this enumeration is selected, then motion does not start because the Lock Position is not crossed.

4	Position Reverse Only	Motion starts when the Master Axis crosses the Lock Position while it is moving in the Reverse Direction. The Master Axis is followed only while it is moving in the Reverse Direction. If the start position of the Master Axis equals the Lock Position and this enumeration is selected, then motion does not start because the Lock Position is not crossed.

Input Parameter	Data Type	Description	Value

Lock Position	IMMEDIATE REAL or TAG	Lock Position in Master Driven Mode After the slave axis motion has been initiated by a MAM or MATC, it goes IP but does not start moving until the master axis crosses the Master Lock Position. This is an absolute position (plus or minus) on the Master. Axis in Master Axis units. Specify a Lock Position to delay the start of motion of a Slave Axis after the motion instruction has been initiated on the Slave Axis. If a Slave Axis is already moving and a second move instruction of the same type (MAM, MAJ, MATC) is activated on the same Slave with a Lock Position, receive an MDSC LOCK WHILE MOVING error for the second instruction. Because Merge is always per formed immediately when an instruction is enabled, a merge instruction with both a Lock Position and a Merge enabled that is executed on a Slave Axis when it is at a nonzero velocity receives an MDSC LOCK WHILE MOVING error. The Lock Direction determines the direction in which the Master Axis must be moving when it crosses the Lock Position before the Slave locks to the Master Axis. If there is an unwind value specified on the Master Axis, then the Master Lock Position must be between 0 and the unwind value. That is, the Lock Position cannot be more than one unwind. This parameter is used only in Master Driven mode. **Lock Position in Time Driven Mode** In Time Driven mode, the Lock Direction must be set to None or an error is generated. **Axis Lock Behavior** When the Master axis crosses the Master Lock position in the direction as specified by the motion instruction, the slave becomes locked to the Master axis. The LockStatus bit of the Slave axis status word is set at this time. The MAM and MAJ instructions on the slave axis in MDSC mode go IP as soon as they are processed by the motion planner. For the Immediate For ward Only or Immediate Reverse Only Lock Directions, the slave gets locked to the Master Axis immediately when the instruction is executed (goes IP). For the Position For ward Only or Position Reverse Only Lock Directions, the slave gets locked to the master axis when the master axis crosses the Master Lock Position in the direction as specified by the motion instruction. In either case, the LockStatus bit of the Slave position status word is set when the lock occurs. Because there is no bidirectional behavior defined, once locked, the Slave follows the Master only in the specified direction. If the Master reverses direction, then the Slave stops following the Master. The LockStatus bit remains set until the Master decelerates to zero prior to reversal. It is cleared at the point of reversal of the Master axis. The Slave does not follow the Master while the Master travels in the reverse direction. If the Master axis changes directions again, then the axis LockStatus bit is set again when the Slave Axis crosses the original reversal point at which time the slave resumes following the Master Axis. **Axis Lock Behavior**  MDSC_LOCK_DIR_MASTER_DIR_MISSMATCH is generated if a new instruction with the Lock Direction parameter in the opposite direction of the current master direction is merged or replaces an active motion instruction (on the Slave Axis). For example, if the programmed Lock Direction is Immediate For ward Only and the master axis is moving in reverse direction. A new instruction used to merge an active instruction on the Slave Axis must use the Immediate For ward Only or Immediate Reverse Only Lock Direction. If the new instruction, which is merged, uses the Position For ward Only or Position Reverse Only Lock Direction, the error MDSC_LOCKDIR_CONFLICT is generated when the instruction	Default = 0

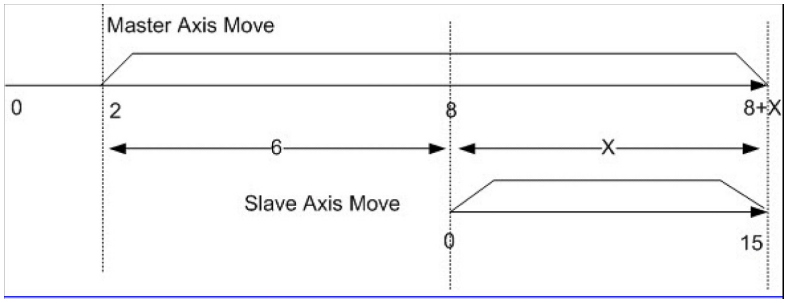
Event Distance ARRAY		The position(s) on a move measured from the end of the move. This is an array of input values that specifies the incremental distances along the move on the Slave. Each member of the array is measured as follows: • Distances are measured star ting from the end of the move towards the beginning of the move as shown in this figure. **Event Distance Measured**  If the value in the Event Distance array is 0.0, then it is the time or distance for the whole move. If the value is greater than or equal to the move length then a 0 is returned. The values entered in the Event Distance array are the same for both Time Driven and Master Driven Mode, only the returns values in the Calculated Data array are different depending on the programmed mode of the slave axis. When Event Distance is specified as a negative number, then the Event Distance calculation is skipped and a -1 is returned in the Calculated Data array for the specified Event Distance parameter. There is no limit on the dimension of either the Event Distance or Calculated Data arrays. However, a maximum of 4 elements (the specified value and the next 3) of the Event Distance array is processed. Special consideration for the rare case of an overshoot when a MCD or MCCD is done close to the moves endpoint. For this case, the Calculated Data will include the overshoot when the Event Distance is 0, because the master has to traverse this amount for the move to finish. For other Event Distances, the overshoot is not included.	Default = 0 No Event Distance array or a REAL array tag. The array must be a minimum size of 4. If the array is greater than 4, only the first four locations specified are used.
Instruction Mode	INIT32	Specifies if an MATC should be executed in Time Driven Mode (0) or Master Driven Mode.	Valid = 0 or 1 Default value = 0

Output Parameter Description

Output Parameter	Data Type	Description	Value

Calculated Data	REAL ARRAY or 0	The calculated output for the Event Distance input parameter, that is the Master Distance(s) (or time) measured from the beginning of the move to the Event Distance point. The returned Calculated Data value is dependent on the following: The instruction type, (that is, MAM for single axis and MCLM/MCCM for coordinated motion) The mode of the Slave Axis (that is, Time Driven or Master Driven). If superimposed motion is active, the Calculated Data does not include any of the superimposed motion. The returned Calculated Date value depends on the following modes. **Master Driven** The returned Calculated Data parameter is the incremental delta Master position that is needed to make the Slave Axis move from the point at which Slave is locked to the Master and star ts moving along the programmed path, to the point where distance to go is less than the specified Event Distance. If the specified data in the Event Distance is array element is 0.0, the master needed for the entire move to complete is returned. **Time Driven** The returned data in the Calculated Data parameter is the total time in seconds that is needed to make the axis move from the move's star t point to a point where distance to go is less than the specified Event Distance. If the specified data, in the Event Distance is array element is 0.0, then the time it takes the entire move to complete is returned. If the value is greater than or equal to the move length then a 0 is returned. The Logix Designer application Motion Planner processes and calculates output data and places the result in the Calculated Data array as supplied in the instruction. The number of calculated array elements stored in the Calculated Data array is based on the follow conditions: The number of elements in the Event Distance array. • For each of the first 4 elements Event Data array, one element will be computed and placed in the Calculated Data array. • The fifth element and beyond of the Event Distance array are ignored. Existing values in the Calculated Data array are overlaid when the Event Distance array is processed. A -1 is returned in the Calculated Data array for each negative value in the Event Distance array. No Event Distance calculation is made for these array elements. You can change the Event Distance array elements dynamically in the program. However, if the Event Distance is changed after the instruction has been initiated (that is, the IP bit has been set), then the change is ignored. An error is generated if size of the Calculated Data array is smaller than the Event Distance array. The default value for versions when bringing old systems for ward (earlier than v20) is 0, signifying that there is no Event Distance array. If the Event Distance is greater than the move length, it is internally forced to equal the move length. **Example 1** Event Distance array = [11, 22, -5, 23, 44] Calculated Data array = [f(11), f(22), -1 ,f(23)] Where f is the calculated data function. The 44 is ignored because it is the fifth element in the Event Distance array. A -1 is returned in the third element of the Calculated Data array because the corresponding Event data Array element is negative. A status bit (CalculatedDataAvailable) in the existing motion	Default = 0 No calculated Data array or a REAL array tag

Master Distance X



Speed, Acceleration, Deceleration, and Jerk Enumerations

Common enumerations are used for the speed parameter of all motion instructions. Some instructions accept only a limited subset of the speed enumerations. Checks for valid unit combinations are done at instruction execution time. Some enumerations that are in the following table are not used now but are reserved for future enhancements.

Additional tables are given below that further clarify which combinations are accepted in MDSC mode and which are accepted in Time Driven mode.

Speed Unit Parameter Descriptions

Mode	Enumerations	Compatibility
Time Driven	0 = Units per sec2	Existing Enumeration
	1 =% Maximum	Existing Enumeration
	2 = Reser ved	
	3 = Seconds Time based programming	New Enumeration
MDSC	4 = Units per MasterUnit	New Enumeration
	5 = Reser ved	
	6 = Reser ved	
	7 = Master Units Analogous to seconds in time based programming	New Enumeration

These rules must be followed to program the dynamics units (Speed, Accel/Decel, and Jerk) of all motion instructions.

- When Speed is in either units/sec, %max, or seconds, then the instruction is considered to be in Time Driven mode, regardless of the selection of units for acceleration, deceleration, or jerk.
- When Speed is in either Master Units or in Units/MasterUnit, then the instruction is considered to be in Master Driven mode, regardless of the selection of units for acceleration, deceleration, or jerk.
- Speed, Acceleration, Deceleration, and Jerk must always be programmed in the same mode (Time Driven or Master Driven) or you get a runtime error.
- When speed is specified in time unit (seconds), the specified time is the total time of the move, including acceleration and deceleration time.

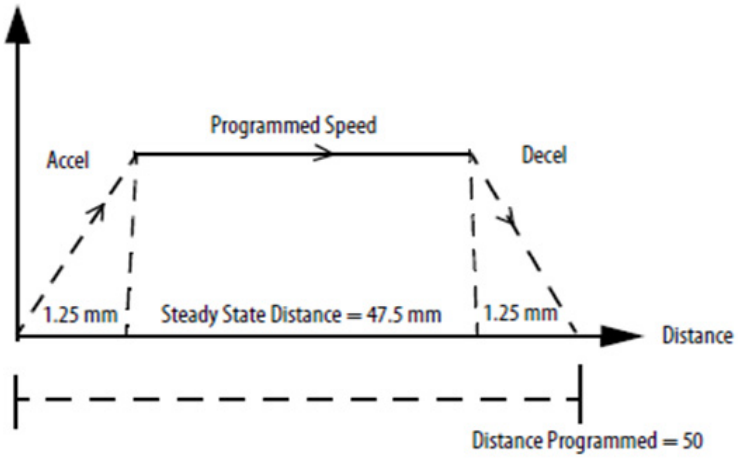
- When speed is specified in Master distance units, the specified distance is the total master distance of the move, including acceleration and deceleration distance of the Master Axis.

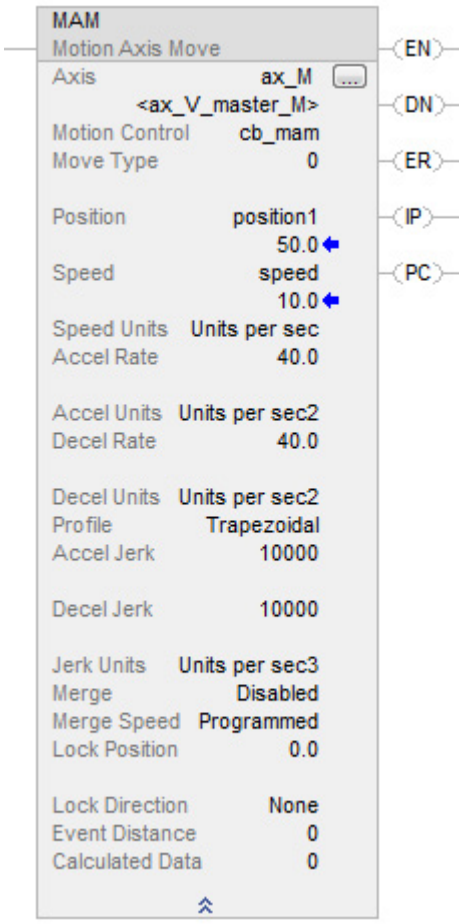
This figure is an example of speed as programmed in Logix Designer application, version 19 and earlier. You only had one option to program, speed directly as a rate in units of distance/time.

Programming Rate in Logix Designer Application Version 19 and Earlier

In Logix Designer application, version 19 and earlier, you could only program speed as units.

Internally, the controller would calculate the total time of the move, accel, and decel times.





Logix Designer application version 19 and earlier

MAM instruction programmed as rate.

Position 50.0 mm (start 0.0) Speed 10.0 mm/sec

Accel 40.0 mm/sec2

Decel 40.0 mm/sec2

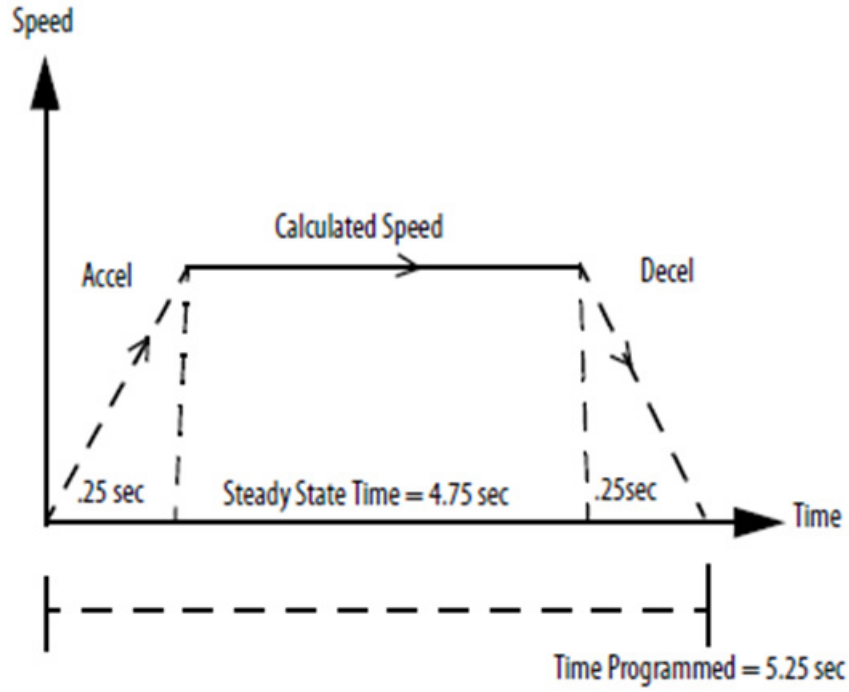
So Travel_Distance = area under the curve [accel + at_speed + decel] Travel_Distance = 50 mm

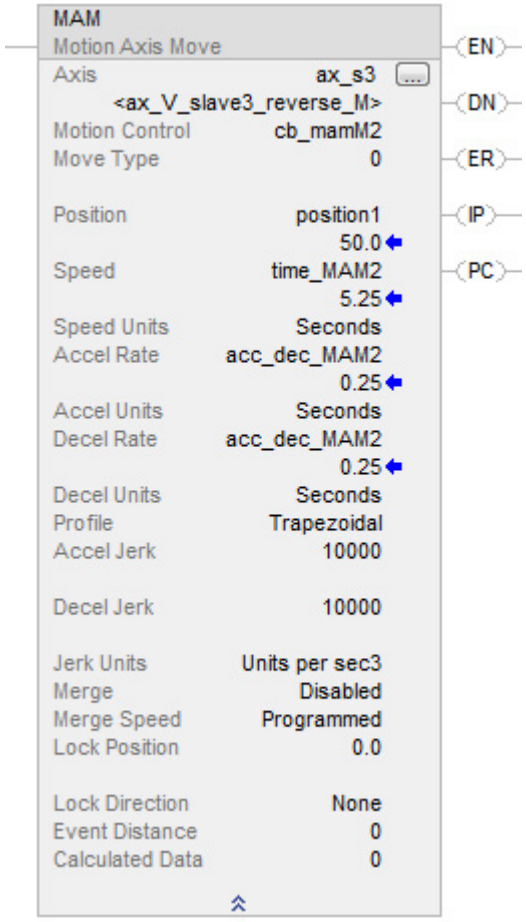
Travel_Distance = 50 mm [1.25 mm + 47.5 mm + 1.25 mm

In this figure, we are programming time. In Logix Designer application, version V20, the controller calculates the speed of the move: Speed & Accel/Decel as time [seconds].

Programming Time in Logix Designer Application Version 20 and Later

In Logix Designer application, version V20 and later you can program the accel and decel and the total time of the move directly.





Logix Designer application version 20 and later

MAM instruction programmed as time.

Position 50.0 mm (start 0.0)

Speed 5.25 sec

Accel 0.25 sec

Decel 0.25 sec

So Travel_Distance = area under the curve [accel + at_speed + decel]

Travel_Distance = 50 mm

Travel_Time = 5.25 sec [0.25 + 4.75 + 0.25 sec]

Acceleration and Deceleration Enumerations

The following enumerations are defined for Acceleration and Deceleration Unit parameters for motion instructions.

Acceleration and Deceleration Unit Parameter Descriptions

Mode	Enumerations	Compatibility
Time	0 = Units per sec2	Existing Enumeration
	1 = % Maximum	Existing Enumeration
	2 = Reser ved	
	3 = Seconds Time based programming	New Enumeration
MDSC	4 = Units per MasterUnit2	New Enumeration
	5 = Reser ved	
	6 = Reser ved	
	7 = Master Units Analogous to seconds in time based programming	New Enumeration

The following table shows acceptable combinations of Speed, Acceleration, and Deceleration units.

Combinations of Speed, Acceleration, and Deceleration Units

Speed Units	Acceleration and Deceleration Units				
	Units per sec2 (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit2 (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**
Units per sec (Time Driven Mode Units)	Existing Enumeration	Existing Enumeration	Not Implemented	Not allowed - Time and Master Driven Units can not be combined.	
% Maximum (Time Driven Mode Units)	Existing Enumeration	Existing Enumeration	Not Implemented		
Seconds (Time Driven Mode Units)	Not Implemented	Not Implemented	New Enumeration		
Units per MasterUnits (Master Driven Mode Units)	Not allowed - Time and Master Driven Units cannot be combined.			New Enumeration	Not Implemented
Master Units (Master Drive Mode Units)				Not Implemented	New Enumeration

These rules must be followed to determine allowable Time and Master Driven mode when programming Acceleration and Deceleration units:

- Speed, Acceleration, Deceleration, and Jerk must always be programmed in the same mode or you get an error.
- If Speed units are Seconds, then Acceleration, Deceleration, and Jerk units must be seconds too.
- If Speed units are Master units, then Acceleration, Deceleration, and Jerk units must be Master units too.
- All unsupported unit combinations result in an err at runtime when the instruction is executed.

Jerk Enumerations

The following enumerations are defined for time driven and MDSC driven Jerk Units.

Time Driven and MDSC Driven Jerk Units Descriptions

Mode	Compatibility	Enumerations
Time	Existing Enumeration	0 = Units per sec3
	Existing Enumeration	1 =% Maximum
	Existing Enumeration	2 = % of Time
	New Enumeration	3 = Seconds Time based programming
MDSC	New Enumeration	4 = Units per MasterUnit2

		5 = Reser ved
	New Enumeration	6 = % of Time-master Driven
	New Enumeration	7 = Master Units Analogous to seconds in time based programming

Acceptable combinations of Accel and Decel Units are based on the programmed Speed Units in the instruction as is shown in the table below. This table is used to clarify the differences in the following four tables.

Speed Units	Accel Units versus Jerk Units
Units per Sec	Table 145
Units per Master Units	Table 146
Seconds	Table 147
Master Units	Table 148

The following table shows acceptable combinations of Acceleration Units and Jerk Units when Speed Units are Units per Second.

Acceleration Units and Jerk Units when Speed Units are Units per Second

Jerk Units	Acceleration and Deceleration Units				
	Units per sec2 (Time Driven Mode Units)	% Maximum (Time Driven Mode Units)	Seconds (Time Driven Mode Units)	Units per MasterUnit2 (Master Driven Mode Units)	Master Units (Master Driven Mode Units)
Units per sec3 (Time Driven Mode Units)	Existing Enumeration	Existing Enumeration	Not Implemented	Incompatible combinations of Time and Master Driven mode. A runtime error occurs.	
% of Time (Time Driven Mode Units)	Existing Enumeration	Existing Enumeration	Not Implemented		
Seconds (Time Driven Mode Units)	Not Implemented	Not Implemented	Not Implemented		
Units per MasterUnits3 (Master Driven Mode Units)	Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.			Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.	
% of Time -Master Driven (Master Driven Mode Units)					
Master Units (Master Driven Mode Units)					

The following table shows acceptable combinations of Acceleration Units and Jerk Units when Speed Units are Units per Master Unit.

Acceleration Units and Jerk Units when Speed Units are Units per Master Unit

Jerk Units	Acceleration and Deceleration Units

	Units per sec2 (Time Driven Mode Units)	% Maximum (Time Driven Mode Units)	Seconds (Time Driven Mode Units)	Units per MasterUnit2 (Master Driven Mode Units)	Master Units (Master Driven Mode Units)
Units per sec3 (Time Driven Mode Units)	Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.			Incompatible combinations of Time and Master Driven mode. A runtime error occurs.	
% Maximum (Time Driven Mode Units)					
% of Time (Time Driven Mode Units)					
Seconds (Time Driven Mode Units)					
Units per MasterUnits3 (Master Driven Mode Units)	Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.			New Enumeration	Not Implemented
% of Time -Master Driven (Master Driven Mode Units)				New Enumeration	Not Implemented
Master Units (Master Driven Mode Units)				Not Implemented	Not Implemented

The following table shows acceptable combinations of Acceleration Units and Jerk Units when Speed Units are in Seconds.

Acceleration Units and Jerk Units when Speed Units are in Seconds.

Speed Units in Seconds		Acceleration (Speed in Seconds)				
		Units per sec2 (Time Driven Mode Units)	% Maximum (Time Driven Mode Units)	Seconds (Time Driven Mode Units)	Units per MasterUnit2 (Master Driven Mode Units)	Master Units (Master Driven Mode Units)
	Units per sec3 (Time Driven Mode Units)	Not Implemented	Not Implemented	Not Implemented	Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.	
	% Maximum (Time Driven Mode Units)	Not Implemented	Not Implemented	Not Implemented		
	% of Time (Time Driven Mode Units)	Not Implemented	Not Implemented	New Enumeration.		
	Seconds (Time Driven Mode Units)	Not Implemented	Not Implemented	New Enumeration.		

Jerk Units	Units per MasterUnits3 (Master Driven Mode Units)	Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.	Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.
	% of Time-Master Driven (Master Driven Mode Units)		
	Master Units (Master Driven Mode Units)		

The following table shows acceptable combinations of Acceleration Units and Jerk Units when Speed is in Master Units.

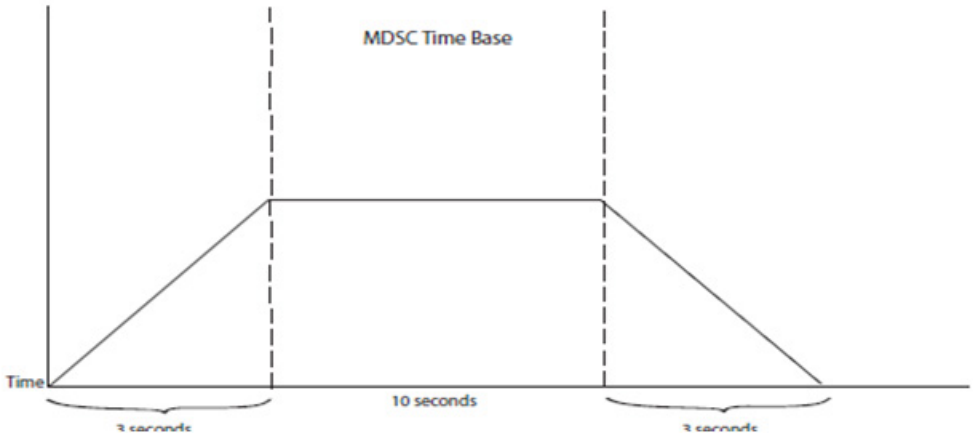
Acceleration Units and Jerk Units when Speed is in Master Units

Speed Units in Master Units		Acceleration (Speed in MasterUnits)				
		Units per sec2 (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit2 (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**
Jerk Units	Units per sec3 (Time Driven Mode Units)	Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.			Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.	
	% Maximum (Time Driven Mode Units)					
	% of Time (Time Driven Mode Units)					
	Seconds (Time Driven Mode Units)					
	Units per MasterUnits3 (Master Driven Mode Units)	Incompatible combinations of Time and Master Driven mode. An error occurs when you verify the routine.			Not Implemented	Not Implemented
	% of Time -Master Driven (Master Driven Mode Units)				Not Implemented	New Enumeration
	Master Units (Master Driven Mode Units)				Not Implemented	New Enumeration

Time Based Planning

With time based planning, the dynamics of the move, that is, the Speed, Acceleration, Deceleration, or Jerk can be directly programmed in units of seconds.

MDSC Time Base Example



If the sum of the Acceleration and Deceleration time is greater than the total time as specified by the speed, the Acceleration and Deceleration time is reduced proportionately so the total specified time in the Speed Parameter is met.

The Speed, when programmed in seconds, takes priority over Acceleration and Deceleration, which in turn takes priority over Jerk. The move will always complete in the specified total time. If the time is too short for the axes physical limits, then a servo drive fault results.

If programmed, move parameters are inconsistent, then an attempt to fix the inconsistency is done by giving priority to parameters in the following order (the lower number has the higher priority):

1. Total move length. This parameter is always satisfied as programmed, it is never changed.
2. Total move duration
3. Acceleration and Deceleration time
4. Acceleration and deceleration Jerk
   - If the Acceleration Jerk time is greater than 50% of the Acceleration Time, then the time is reduced so that the Acceleration completes in the specified Acceleration Time. Similar calculations are performed for the Deceleration Jerk times.
   - If speed units are programmed in units of seconds, then acceleration and deceleration must also be programmed in seconds. For this case, the jerk can be programmed in either units of seconds or percentage of time.

     Tip: Time based planning cannot be used for jogged moves (MAJ).

Time Based Planning is only functional for moves starting and ending at zero velocity. An error generates if a move is started with a non-zero velocity or acceleration. You have the option to wake-up a paused or dwelling move with a time based move. See Dwells for more information.

The values of zero for acceleration, deceleration, or jerk times are permitted and will generate infinite acceleration, deceleration, or jerk. A value of zero for speed generates a runtime error. For an S-curve profile, an infinite jerk will change to a trapezoidal profile.

All the existing functionality for the Time Based programming mode is supported when you operate in Master Driven mode. Time becomes master distance in Master Driven mode.

Important: Time based planning is not implemented for coordinated moves in Logix Designer application, version 20. Dynamics in seconds are incompatible with Merge Speed = Current. This will result in error 94, MDSC_UNITS_CONFLICT.

Dwells

You have the option to program a dwell by using Time Based Programming in either Time Driven mode or Master Driven mode. When a zero length move is programmed, the duration of the dwell is programmed in the Speed parameter and the Acceleration, Deceleration, and Jerk parameters are ignored.

When in time driven mode, the duration of the dwell is programmed in seconds. When in MDSC mode, the duration of the dwell is programmed in units of Master Distance. If speed is specified in Master Units, the move remains active until the specified Master distance is traversed.

Similarly, when in Time Driven mode, program the move time directly in seconds and with a zero departure. This results in a programmed delay of the specified time.

A zero length move that is programmed with a Speed of 0 Seconds or zero Units per Master Units complete in the minimum time possible, which is 1 coarse update period.

In MDSC mode, the dwell starts either at the Master Lock Position or immediately, depending on the programmed Lock Direction parameter, and continues for a duration as specified in the Speed parameter.

Time Based Programming Errors

There are two time based programming errors:

- AXIS_NOT_AT_REST, Error 100
- MDSC_UNITS_CONFLICT, Error 94

An AXIS_NOT_AT_REST, (Error 100) occurs if a move is programmed by using Time Based Planning and is started when the active move is at a nonzero velocity. This means that a move by using Time Based Planning with the Merge Enabled in an instruction will cause an error for most cases because merge is typically used when the axes are moving.

MDSC_UNITS_CONFLICT, (Error 94) occurs if speed is programmed in units of seconds and acceleration, deceleration, or jerk is not programmed in seconds (or % of Time for jerk).

Path Fidelity

The path fidelity (remaining on path) is maintained through the full range of Master dynamics of Speed, Acceleration, Deceleration, and Jerk in Master Driven mode. This means that the move path does not change as the Master speed is changed with an MCD.

There is no end position overshoot as the Master velocity is changed; however, if the Master velocity is changed extremely rapidly, there can be a Slave velocity or acceleration or deceleration overshoot.

Status Bits for Motion Instructions (MAM, MATC, MAJ) When MDAC Is Active

The following table describes the predefined data type status bits for motion instruction MAM, MATC, and MAJ.

Status Bits for Motion Instructions when an MDAC Instruction is Active

Bit	Description
.EN (Enable)	The Enable bit is set when the rung makes a false-to-true transition and remains set until the servo message transaction is completed and the rung goes false.
.DN (Done)	Timing done output. Indicates when the accumulated time is greater than or equal to the preset value.
.ER (Error)	The Error bit is set to indicate that the instruction detected an error, such as if you specified an unconfigured axis.
.PC (Process Complete)	The Process Complete bit is set after the diagnostic test process has been successfully completed.
.IP (In Process)	The In Process bit is set on positive rung transition and cleared after the diagnostic test process is complete, or terminated by a stop command, shutdown, or a servo fault.
.AC (Active)	The Active bit is set when a move (MAJ, MAM or MATC) goes IP in Master Driven mode on the axis that is selected as the Slave Axis of the MDAC instruction. The AC bit will be reset when all single axis motion that is being controlled by the MDAC is completed. If the Slave Axis is star ted in Time Driven mode, then the AC bit of the MDAC does not go active. The IP bit of the MDAC instruction does not change at this time.
ACCEL	The ACCEL bit is set as expected during motion. It is independent of Master Axis acceleration. The ACCEL bit on the instruction driving the Slave Axis, for example, MAM on the Master Axis is set as the Slave Axis accelerating to its commanded speed as a result of the master axis acceleration. In addition, The ACCEL bit of the slave axis is set when it is accelerating due to not being at the programmed acceleration. This bit is insensitive to acceleration occurring on the Master Axis. However, the AccelStatus bit, which is in the MotionStatus word of the Slave Axis (not the instruction driving the slave axis), is set or cleared based on changes in the programmed velocity of the Slave Axis.
DECEL	The DECEL bit is set as expected during motion. It is independent of Master Axis deceleration. The DECEL bit on the instruction driving the Slave Axis is set as the Slave Axis is decelerating to its commanded speed. as a result of the master axis deceleration. In addition, The DECEL bit of the slave axis is set when it is decelerating due to not being at its programmed deceleration. This bit is insensitive to deceleration occurring on the Master Axis. However, the DecelStatus bit, which is in the MotionStatus word of the Slave Axis (not the instruction driving the slave axis), is set or cleared based on changes in the programmed velocity of the Slave Axis.
TrackingMaster	Indicates that the Slave Axis is tracking the Master Axis. Only used in Master Driven mode. When an instruction is initiated in Master Driven mode, the Slave Axis accelerates to the speed that its programmed for MDSC mode. This bit is insensitive to slave acceleration/deceleration of the master axis. The Tracking Master is set when the acceleration is complete in MDSC mode. This means that the Slave Axis is synchronized to the Master Axis. This bit is insensitive to the accel/decel of the master axis. The Tracking Master bit is cleared when any of the following occurs on the Slave Axis: • When the Slave Axis star ts to either accelerate or decelerate for any reason, for example, for an MCD or an MAS being issued. • When the Slave Axis is relinked to another Master Axis. In this situation, the TrackingMaster bit is first cleared and then it is set again in the new instruction status word when the Slave Axis starts tracking the new Master Axis again. • The Slave Axis is stopped. The Tracking Master is cleared as soon as the stop is initiated on the Slave Axis. This bit is never set when LockDir = NONE. The Tracking Master bit on the Slave Axis is not affected by any operation (for example, MAS, MCD) on the Master Axis. The Tracking Master bit is always cleared in Time Driven mode.
CalculatedDataAvailable	Indicates when the output data in the Calculated Data parameter has been updated and is available. The CalculatedDataAvailable bit is not set for any move that Event Distance is not specified, that is, for any move where the Event Distance parameter in the instruction is zero, this is not the value in the parameter array.

Program a velocity profile and jerk rate and tune an S-Curve Profile

Use this chapter to program a velocity profile and jerk rate, and tune an S-Curve Profile.

Definition of Jerk

Jerk is the rate of change of acceleration or deceleration.

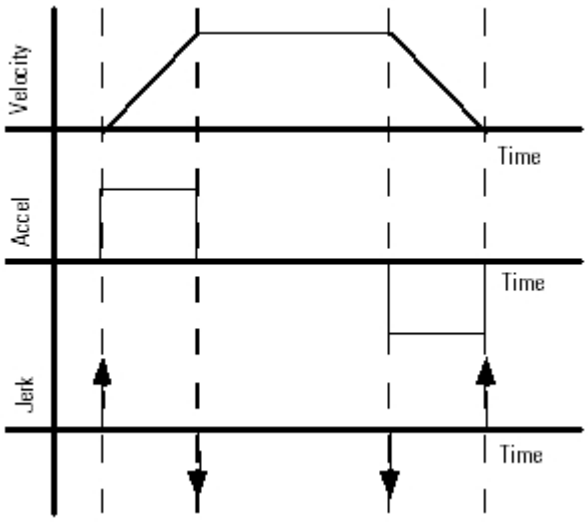
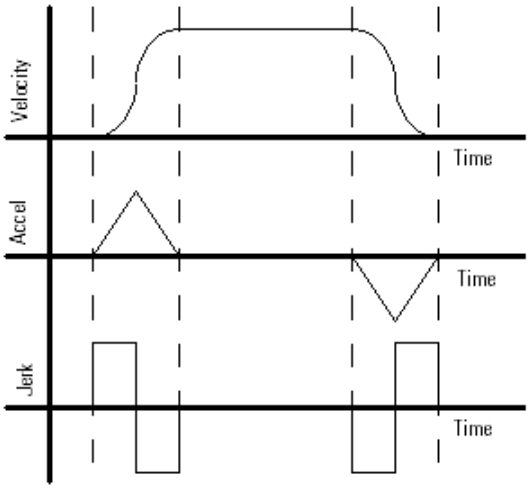
Example:

If acceleration changes from 0 to 40 mm/s^2 in 0.2 seconds, the jerk is as follows.

(40 mm/s^2 - 0 mm/s^2) / 0.2 s = 200 mm/s^3.

Choose a Profile

Consider cycle time and smoothness when you choose a profile.

If you want	Choose This Profile	Considerations
• Fastest acceleration and deceleration times • More flexibility in programming subsequent motion	Trapezoidal  {bmct Trapezoidal.bmp}	Jerk **does not** limit the acceleration and deceleration time. • The Acceleration and Deceleration rates control the maximum change in Velocity. • Your equipment and load get more stress than with an S-curve profile. • Jerk is considered infinite and is shown as a vertical line.
Smoother acceleration and deceleration that reduces the stress on the equipment **and load**	S-curve  	Jerk limits the acceleration and deceleration time. • It takes longer to accelerate and decelerate than a trapezoidal profile. • If the instruction uses an S-curve profile, the controller calculates acceleration, deceleration, and jerk when you start the instruction. • The controller calculates triangular acceleration and deceleration profiles.

Velocity Profile Effects

The table summarizes the differences between profiles.

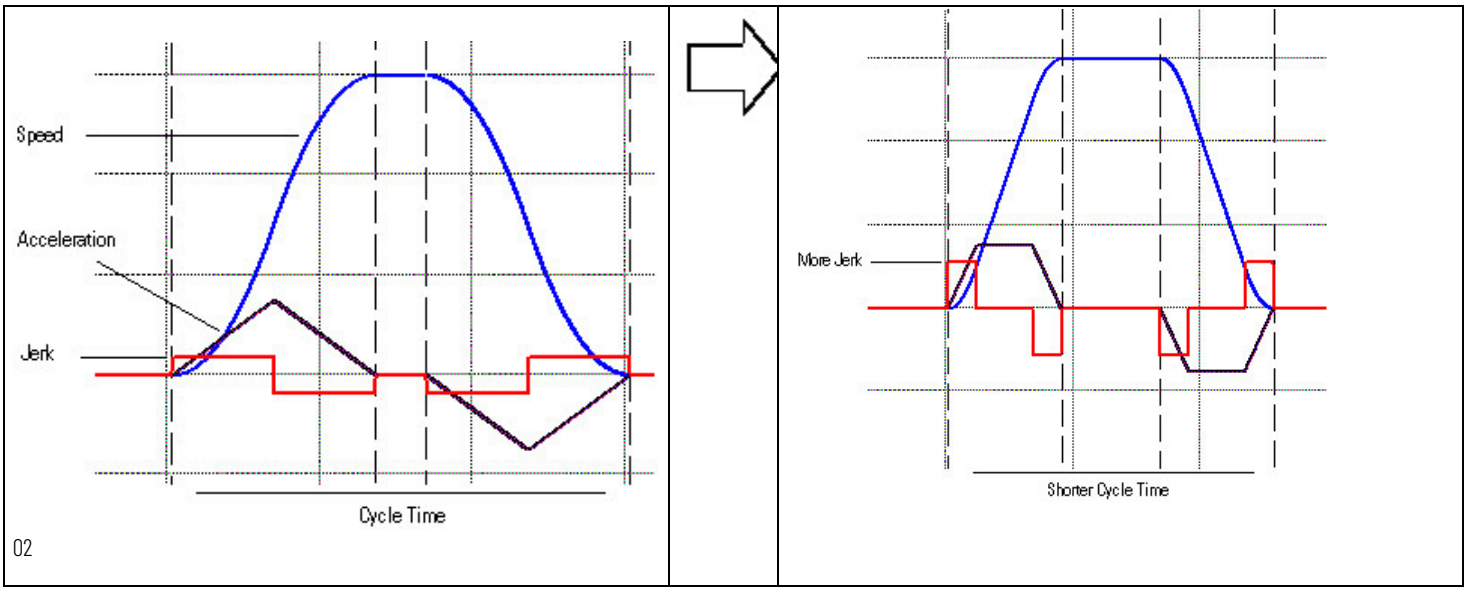
Profile	ACC/DEC	Motor	Priority of Control		
Type	Time	Stress	Highest To Lowest		
Trapezoidal	Fastest	Worst	Acc/Dec	Velocity	Position
S-Curve	2X Slower	Best	Jerk	Acc/Dec	Velocity

Tune an S-Curve Profile

Use this procedure to balance the smoothness and cycle time of motion that uses an S-curve profile.

When to Do This Procedure

Do this procedure when you want to decrease the cycle time of an S-curve motion profile, but keep some of the profile's smoothness.



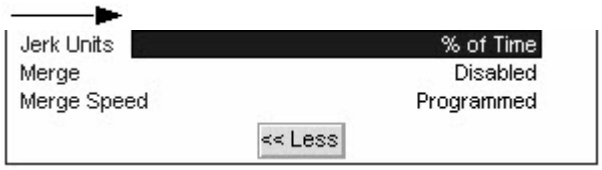
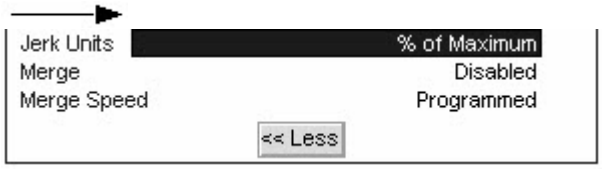
To use this procedure, your application must meet these requirements.

- The controller is at revision 16 or later.
- One of these instructions produces the motion.
  - Motion Axis Move (MAM)
  - Motion Axis Jog (MAJ)
  - Motion Axis Stop (MAS)
- The instruction uses an S-curve profile.

To tune an s-curve profile

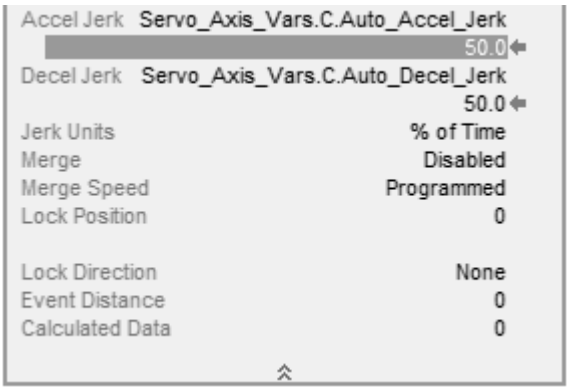
Important: In this procedure, you increase the jerk. This increases the stress on the equipment and load. Make sure you can identify when the equipment or load has reached its jerk limit.

1. Are the Jerk Units set to % of Time?

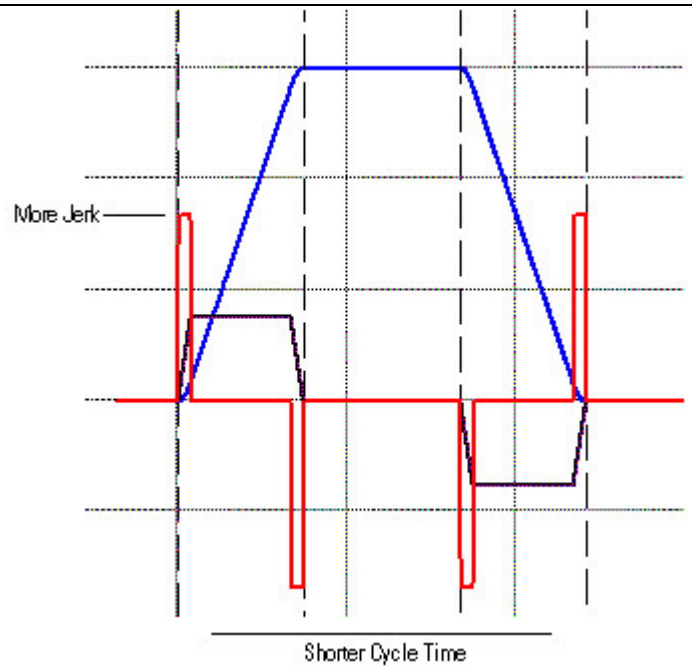
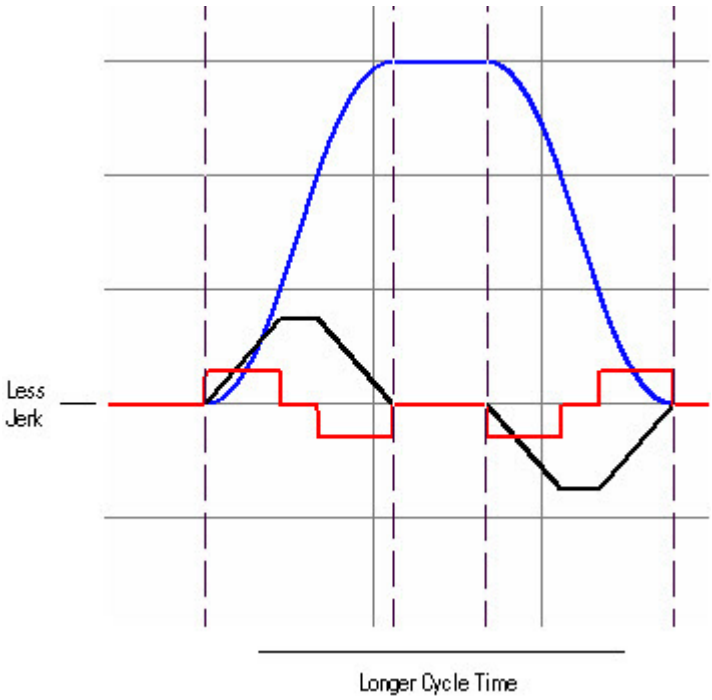
If the Jerk Units Are	Then
% of Time 	Continue with step 2.
% of Maximum 	A. Change the Jerk Units to % of Time
Units per sec3	B. Continue with step 2.

2. Set the Jerk values to 50% of Time.

Example:



3. Test your equipment and observe its jerk.
4. Adjust the jerk values.

If There Is	Then	Which Results In
NOT too much jerk	Reduce the % of Time	
Too much jerk	Increase the % of Time	

5. Repeat steps 3 and 4 until you have the desired balance between smoothness and cycle time.

See also

Motion Move Instructions on page 73

Motion Configuration Instructions on page 315

Motion Error Codes, faults, and attributes

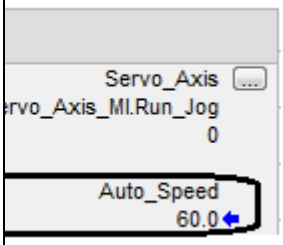
This chapter provides an overview of the error codes, faults and attributes for motion instructions.

Motion Error Codes (.ERR)

This table lists the error codes for Logix Designer software motion instructions.

Motion Instruction Error Codes Descriptions

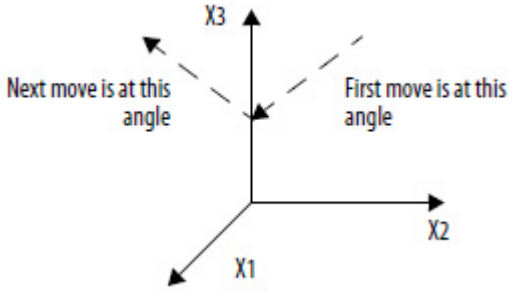
Error	Corrective Action or Cause	Notes
1	Reserved Error Code 1	Reserved for future use.
2	Reserved Error Code 2	Reserved for future use.
3	Look for another instance of this type of instruction. See if its EN bit is on but its DN and ER bits are off (enabled but not done or erred). Wait until its DN or ER bit turns on.	Execution Collision You cannot execute an instruction by using the same control word as another instruction if the other instruction is not done or has an error. Regardless, it is recommended that each instruction has a unique control word.
4	Open the servo loop before you execute this instruction.	Servo On State Error
5	Close the servo loop before you execute this instruction.	Servo Off State Error For a motion coordinated instruction, refer to the online help for the instruction for extended error code definitions. They identify which axis caused the error. Example: If the extended error code is zero, check the axis for index zero of the coordinate system.
6	Disable the axis drive.	Drive On State Error
7	Execute a Motion Axis Shutdown Reset (MASR) instruction or direct command to reset the axis.	Shutdown State Error For a motion coordinated instruction, refer to the online help for the instruction for extended error code definitions. They identify which axis caused the error. Example: If the extended error code is zero, check the axis for index zero of the coordinate system.
8	The configured axis type is not correct.	Wrong Axis Type For a motion coordinated instruction, refer to the online help for the instruction for extended error code definitions. They identify which axis caused the error. Example: If the extended error code is zero, check the axis for index zero of the coordinate system.
9	The instruction tried to execute in a direction that aggravates the current overtravel condition.	Overtravel Condition
10	The master axis reference is the same as the slave axis reference or the Master Axis is also an axis in the Save Coordinate System.	Master Axis Conflict

11	At least one axis is not configured to a physical motion module or has not been assigned to a Motion Group.	Axis Not Configured For single axis instructions: the Extended Error code for MAG, MDAC, MAPC, MAM, MAJ, MATC, and MCD is defined as: 1 = Slave axis 2 = Master Axis For the MAM, MCD, and MAJ instructions in time driven mode, the axis being moved is a slave axis. For multi-axes instructions: the Extended Error code for MDCC, MCLM, MCCM, MCCD, and MCPM is defined as: The axis number in the coordinate system where 0 = 1st axis 2 = Master Axis or 3rd Slave Axis
12	Messaging to the servo module failed.	Servo Message Failure
13	Refer to the online help for the instruction for extended error code definitions. Example: An MAJ instruction has an ERR = 13 and an EXERR = 3. In this case, change the speed so that it's in range. 	Parameter Out Of Range An EXERR = 0 means the first operand of the instruction is outside its range.
14	The instruction cannot apply the tuning parameters because of an error in the run tuning instruction.	Tune Process Error
15	The instruction cannot apply the diagnostic parameters because of an error in the run diagnostic test instruction.	Test Process Error
16	Wait until the homing process is done. For coordinated move instructions, it identifies which axis caused the error.	Home In Process Error
17	The instruction tried to execute a rotary move on an axis that is not configured for rotary operation.	Axis Mode Not Rotary
18	The axis type is configured as unused.	Axis Type Unused
19	The motion group is not in the synchronized state. This could be caused by a missing or misconfigured servo module.	Group Not Synchronized Group sync status is only cleared on a group overlap or CST loss fault.
20	The axis is in the faulted state.	Axis In Faulted State
21	The group is in the faulted state.	Group In Faulted State
22	Stop the axis before you execute this instruction.	Axis In Motion
23	An instruction attempted an illegal change of dynamics.	Illegal Dynamic Change
24	Take the controller out of test mode.	Illegal Controller Mode
25	Either one of these could be the reason for the error: 1. If the number of axis in the coordinate system is not equal to 2 and MCCM is programmed for Circle Type 'Radius'. 2. If the axis used in the instruction is configured in Torque control mode.	Illegal Instruction
26	The cam array is of an illegal length.	Illegal Cam Length
27	The cam profile array is of an illegal length.	Illegal Cam Profile Length

28	You have an illegal segment type in the cam element.	Illegal Cam Type The .SEGMENT field of Motion Instruction identifies which cam profile element contains the invalid segment type. A .SEGMENT = 3 means the 4th element (or [3]) of the cam profile array contains the invalid segment type.
29	You have an illegal order of cam elements.	Illegal Cam Order The .SEGMENT field of Motion Instruction identifies which cam profile element contains the invalid (non-ascending) master value. A .SEGMENT = 3 means the 4th element (or [3]) of the cam profile array contains the invalid (non-ascending) master value.
30	You tried to execute a cam profile while it is being calculated.	Cam Profile Being Calculated
31	The cam profile array you tried to execute is in use.	Cam Profile Being Used
32	The cam profile array you tried to execute has not been calculated.	Cam Profile Not Calculated
33	A MAM - Master Offset move was attempted without a Position CAM in process.	Position Cam Not Enabled
34	A MAH instruction is trying to start while a registration is already running.	Registration in Progress
35	The specified execution target exceeds the number of Output Cam targets configured for the axis.	Illegal Execution Target
36	Either the size of the Output Cam array is not supported or the value of one of its members is out of range.	Illegal Output Cam ExErr#1: Output bit less than 0 or greater than 31. ExErr#2: Latch type less than 0 or greater than 3. ExErr#3: Unlatch type less than 0 or greater than 5. ExErr#4: Left or right position is out of cam range and the latch or unlatch type is set to `Position' or `Position and Enable'. ExErr#5: Duration less than or equal to 0 and the unlatch type is set to `Duration' or `Duration and Enable'. ExErr#6: Enable type less than 0 or greater than 3 and the latch or unlatch type is set to `Enable', `Position and Enable', or `Duration and Enable'. ExErr#7: Enable bit less than 0 or greater than 31 and the latch or unlatch type is set to `Enable', `Position and Enable', or `Duration and Enable'. ExErr#8: Latch type is set to `Inactive' and unlatch type is set to either `Duration' or `Duration and Enable'.
37	Either the size of the Output Compensation array is not supported or the value of one of its members is out of range. The array index associated with errors 36 and 37 are stored in .SEGMENT of the Motion Instruction data type. Only the first of multiple errors are stored. The specific error detected is stored in Extended Error Code. With the ability to dynamically modify the Output Cam table, the Illegal Output Cam error 36 can occur while the MAOC is in-process. In general, the cam elements where an error was detected will be skipped. The following are exceptions and will continue to be processed. Error 2, Latch Type Invalid. Latch Type defaults to Inactive. Error 3, Unlatch Type Invalid. Unlatch Type defaults to Inactive. Error 8, with Unlatch Type of Duration and Enable. Will behave as an Enable Unlatch type.	Illegal Output Compensation ExErr#1: Mode less than 0 or greater than 3. ExErr#2: Cycle time less than or equal to 0 and the mode is set to `Pulsed' or `Inverted and Pulsed'. ExErr#3: Duty cycle less than 0 or greater than 100 and the mode is set to `Pulsed' or `Inverted and Pulsed'.

38	The axis data type is illegal. It is incorrect for the operation.	Illegal Axis Data Type
39	You have a conflict in your process. Test and Tune cannot be run at the same time.	Process Conflict
40	You are trying to run a MSO or MAH instruction when the drive is locally disabled.	Drive Locally Disabled
41	The Homing configuration is illegal. You have an absolute homing instruction when the Homing sequence is not immediate.	Illegal Homing Configuration
42	The MASD or MGSD instruction has timed out because it did not receive the shutdown status bit. Usually a programmatic problem caused when either MASD or MGSD is followed by a reset instruction that is initiated before the shutdown bit has been received by the shutdown instruction.	Shutdown Status Timeout
43	You have tried to activate more motion instructions than the instruction queue can hold.	Coordinate System Queue Full
44	You have drawn a line with three 3 points and no centerpoint viapoint or plane centerpoint can be determined.	Circular Collinearity Error
45	You have specified one 1 point radius or "drawn a line" centerpoint, viapoint and no centerpoint radius or plane centerpoint, viapoint can be determined.	Circular Start End Error
46	The programmed centerpoint is not equidistant from start and end point.	Circular R1 R2 Mismatch Error
47	Contact Rockwell Automation Support.	Circular Infinite Solution Error
48	Contact Rockwell Automation Support.	Circular No Solutions Error
49	|R| < 0.01. R is basically too small to be used in computations.	Circular Small R Error
50	The coordinate system tag is not associated with a motion group.	Coordinate System Not in Group
51	You have set your Termination Type to Actual Position with a value of 0. This value is not supported.	Invalid Actual Tolerance
52	At least one axis is currently undergoing coordinated motion in another coordinate system.	Coordination Motion In Process Error
53	Uninhibit the axis.	Axis Is Inhibited For single axis instructions, the Extended Error code for MAG, MDAC, MAPC, MAM, MAJ, MATC, and MCD is defined as: 1 = Slave axis 2 = Master Axis For the MAM, MCD, and MAJ instructions in time driven mode, the axis being moved is a slave axis. For multi-axes instructions, the Extended Error code for MDCC, MCLM, MCCM, MCCD, MCTO, and MCPM is defined as: The axis number in the coordinate system where 0 = 1st axis 2 = Master Axis or 3rd Slave Axis

54	You cannot start motion if the maximum deceleration for the axis is zero. 1. Open the properties for the axis. 2. On the **Planner** tab, enter a value for the **Maximum Deceleration**. For coordinated move instructions, it identifies which axis caused the error.	Zero Max Decel
61	Refer to the online help for the instruction for extended error code definitions.	Connection Conflict
62	Cancel the transform that controls this axis or don't use this instruction while the transform is active.	Transform In Progress You cannot execute this instruction if the axis is part of an active transform.
63	Cancel the transform that controls this axis or wait until the transform is done moving the axis.	Axis In Transform Motion You cannot execute this instruction if a transform is moving the axis.
64	Use a Cartesian coordinate system.	Ancillary Not Supported You cannot use a non-Cartesian coordinate system with this instruction.
65	Once the error occurs, position the axis (or master axis) within bounds to execute instructions generating motion on the axis. This error occurs with MAM, MAPC, MCLM, MCCM, MCPM instructions and axes that are part of Kinematics transforms. This error occurs when instruction executes and the absolute position is outside the overtravel limits.	Absolute position outside the overtravel limits
66	Be sure to keep the robot in the arm solution that you configured it in. You can configure the robot in either a left arm or right arm solution.	You are attempting to fold back an articulated independent or dependent two axis robot on itself at the quadrant boundaries.
67	Either one of these could be the reason for the error: 1. You're trying to move to a place the robot cannot reach. 2. MCT, MCTO, MCTP or MCTPO attempted while at origin. To avoid having the robot fold back on itself or extend beyond its reach, joint limits are calculated internally by the firmware for Delta2D, Delta3D and SCARA Delta robots. If you try and configure a move that violates these limits, this error occurs. Refer to the online help for the instruction for extended error code definitions. It identifies which orientation axis caused the error.	Invalid Transform Position
68	Move the joints so that the end of the robot is not at the origin of the coordinate system.	Transform At Origin You cannot start the transform if the joint angles result in X1 = 0 and X2 = 0.

69	Check the maximum speed configuration of the joints. Use target positions that keep the robot from getting fully stretched or folding back on itself at the origin of the coordinate system. Move in a relatively straight line through positions where X1 = 0 and X2 = 0.	Max Joint Velocity Exceeded The calculated speed is very high. This happens when the robot either: gets fully stretched. folds back on itself. moves away from X1 = 0 and X2 = 0 in a different angle than it approached that position. is configured with the wrong velocity limit. Example: These moves produce this error. 
70	Look for source or target axes that are configured as rotary positioning mode. Change them to linear positioning mode.	Axes In Transform Must Be Linear A transform works only with linear axes.
71	Wait until the transform that you are canceling is completely canceled.	Transform Is Canceling
72	Check the target positions. A calculated joint angle is beyond +/- 360·.	Max Joint Angle Exceeded
73	Check that each MCT instruction in this chain is producing valid positions.	Coordinate System Chaining Error This MCT instruction is part of a chain of MCT instructions. There is a problem with one of the instructions in the chain.
74	Change the orientation to angles that are within +/- 360·.	Invalid Orientation Angle
75	Use this instruction only with a 1756-L6x controller.	Instruction Not Supported You can use an MCT or MCTP instruction only with a 1756-L6x controller.
76	You cannot start motion that uses an S-curve profile if the maximum deceleration jerk for the axis is zero. 1. Open the properties for the axis. 2. On the **Planner** tab, enter a value for the **Maximum Deceleration Jerk**.	Zero Max Decel Jerk
77	How many axes are in your coordinate system? 2 — Use a non-mirror transform direction. 3 — Use a non-inverse transform direction.	Transform Direction Not Supported You're trying to use the mirror directions with a 3-axis coordinate system and a non-zero base offset (X2b) or effector offset (X2e). Mirror directions are not supported for 2-axis Coordinate Systems. You are attempting to use either a 2 or 3-axis Cartesian, Delta2D, Delta3D or SCARA Delta target coordinate system with transform directions other than forward and inverse. You can use inverse mirror directions only when both these conditions are true: You have a 3-axis coordinate system. The base offset (X2b) and end effector offset (X2e) of the X2 dimension are zero.
78	New check for a secondary Instruction overlap on top of an active Stop instruction.	Not Allowed While Stopping You cannot overlap certain Motion instructions while stopping. Wait for the first instruction to complete before starting the second instruction.

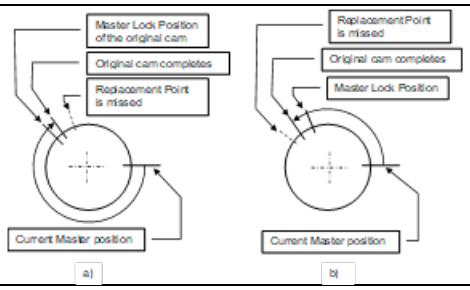
79	Home your axis again. Error of Home instruction occurs, if any other motion on the axis is encountered during the homing sequence.	Internal Homing Sequence Error Invalid Planner State If you see this error, rehome your axis in your application program. Make sure the axis is stopped before home is attempted. If the error persists, contact Rockwell Automation Support.
80	When referencing a Scheduled Output Module, for example, the OB16IS, make sure that the Output operand of the MAOC references O:Data, and that the Scheduled Output Module's communication format is set to 'Scheduled Output Data per point'.	MAOC Invalid Output Operand If the MAOC output operand references an OB16IS Scheduled Output module, two additional checks occur when the MAOC is initiated. The Output operand must be referencing the beginning of the module's output data tag, 'O.Data'. The communications format of the OB16IS module must be the default "Scheduled Output Data per Point". If either of these checks fail you see this error. ExErr#1: Invalid Data Tag Reference - The Output operand is not pointing to the O.Data element of the module's output data tag. This is applicable to 5069-OB16F, 1756-OB16IEFS, 1732E-OB8M8SR, and OB16IS modules only. ExErr#2: Invalid Module Communication Format for one of these modules: 5069-OB16F, 1756-OB16IEFS, 1756-OB16IS, 1732E-OB8M8SR modules. ExErr#3: CIP Sync not synchronized - Scheduled output module reporting not synchronized to a CIP Sync master. Applicable to the 5069-OB16F, 1756-OB16IEFS, 1732E-OB8M8SR modules only. ExErr#4: Grandmaster Clock mismatch - Scheduled output module has different Grandmaster clock than the controller. Applicable to the 5069-OB16F, 1756-OB16IEFS, 1732E-OB8M8SR modules only.
81	Error on MGSR, if a MASD or MGS (programmed) is executed while the MGSR is still in process. Do not overlap the MASD instruction or MGS stop instruction with Stop Mode = Programmed on an active MGSR instruction.	Partial Group Shutdown Reset. If your application program is actively executing an MGSR instruction and you try to execute an MASD instruction or MGS stop instruction with Stop Mode = Programmed on one of the axes affected by the active MGSR instruction, you will see this error on the MGSR instruction.
82	The axis was found to be in the incorrect operational axis state.	CIP axis in incorrect state.
83	This instruction cannot be performed due to control mode or feedback selection.	Illegal Control Mode or Method. The MDS instruction is not valid in Position Loop or Feedback Only modes. The error also triggers when the CIP motion device does not have a valid feedback to perform the instruction.
84	The CIP drive device digital input is not assigned.	Drive Digital Input Not Assigned
85	Instruction not allowed when redefine position is in process. Performing MAH while MRP is in process results in this instruction error.	Homing, redefine position in progress An Active Redefine Position instruction is in process. You would get this error if any of the motion planner instructions are executed while MRP is in progress. Motion Instructions included are: MAM, MAJ, MCLM , MCCM, MATC, MAPC , MDAC, MDCC, and MCPM.
86	Current use of the MDS instruction requires an optional attribute that is not supported.	Optional attribute not supported by the integrated motion drive being used. Executing a MDS instruction on a CIP- Velocity Loop with Feedback axis associated with a Kinetix 6500 drive errors. The instruction requires an optional attribute that is not currently supported. The MDS instruction is not supported by the drive type.
87	The instruction is invalid while running direct controlled motion.	Not Allowed While In Direct Motion

88	The instruction is invalid while running planned motion.	Not Allowed While Planner Active
93	A move was programmed in MDSC mode before the MDSC link has been established by the execution of a MDAC or MDCC.	MDSC Not Activated
94	Some dynamics units belong to Master Driven Mode and some to Time Driven Mode. Some units are time based whereas others are velocity based, for example, Speed in Seconds and Acceleration in units/sec2. Incompatibility of units. Dynamics in Seconds are incompatible with Merge Speed = Current.	MDSC Units Conflict
95	All instructions in the queue must use a compatible Lock Direction, for example, Position Forward Only and Immediate Forward Only. Lock Direction = None and speed units belong to Master Driven Mode.	MDSC Lock Direction Conflict If you change from Time Driven mode to Master Driven mode while an axis is moving and Lock Direction is not Immediate Forward or Reverse you will get error 95 MDSC Lock Direction Conflict.
96	MDAC (All) and MDAC (something other than All) on the same slave.	MDSC MDAC All Conflict
97	Trying to replace a running Master with a new Master whose speed is zero, or replacing a Slave that is moving via an MAM with another MAM with the same or a different Master that is not moving.	MDSC Idle Master and Slave Moving
98	The actual direction of master axis' motion does not match the direction programmed by Lock Direction parameter (IMMEDIATE FORWARD ONLY or IMMEDIATE REVERSE ONLY) when the slave is already moving.	MDSC Lock Direction Master Direction Mismatch
99	Either of these could be the reason for the error: • Performing MDCC on non-Cartesian coordinate system • Using Lock Position for MATC in Time Driven Mode • If the master axis in the MDCC instruction is not from the Base Update Group.	MDSC Feature Not Supported
100	If speed is in seconds or Master units, move must start from rest.	Axis Not At Rest.
101	Return data array is either nonexistent or not big enough to store all the requested data.	MDSC Calculated Data Size Error.
102	Attempt to activate a second MDSC instruction with a Lock Position or a Merge with a Lock Position while an axis is moving.	MDSC Lock While Moving.
103	If the Master Axis is changed and the new slave speed is less than 5% of the original slave speed for Single Axis instructions, or 10%, depending on the move of the original Slave Coordinate System speed, then this error will occur and the change will not be allowed. The same applies when changing from Time Driven mode to MDSC mode.	MDSC Invalid Slave Speed Reduction.

104	IF: a motion instruction performs either: A change in the Master Axis A change in speed units AND: if in the same update period, the instruction is either forced to pause with a speed of zero, or stopped with a MAS or MCS THEN: the velocity profile is changed to trapezoidal and this error code is reported.	MDSC 2 Instructions were started in 1 Update Period therefore Jerk was Maximized.
105	An instruction in the coordinated motion queue is either trying to change the Master Axis or changing the mode from MDSC mode to Time Driven mode or from Time Driven mode to MDSC mode.	MDSC Invalid Mode Or Master Change.
106	Cannot use Merge to Current when programming in time driven mode using seconds or master driven mode using master units. Change merge speed parameter.	Merge to Current using seconds is illegal.
107	Target device does not support the requested operation, service, or both.	There is not any corrective action that can be taken.
108	Coordinated System contains a multiplexed axis.	Motion coordinated instructions cannot contain multiplexed axes.
109	You attempted to use an axis that is defined as a converter or track section.	Invalid Axis Configuration A converter or track section cannot be used for this instruction.
110	You attempted to put an axis configured as a converter into a coordinate system definition.	A converter cannot be used for this instruction.
111	You cannot start motion if the maximum acceleration for the axis is zero. 1. Open the properties for the axis. 2. On the **Planner** tab, enter a value for the **Maximum Acceleration.**	Zero Max Accel
112	Operand Nominal Master Velocity in MDCC instruction must be equal to zero in CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers.	Operand not supported.
121	The Coordinate Definition cannot be set to <none>.	Invalid coordinate definition for the instruction.
125	Operand 0 must be a Cartesian coordinate system type.	Source coordinate system is not Cartesian.
126	Operand 1 must NOT be a Cartesian coordinate system type.	Target coordinate system is Cartesian.
128	Turns counter functionality is not supported. Turns counter is not supported on J1 and J4 on 4 axis geometries and J1 and J6 on 5 axis geometries. Turns counter functionality is not supported when transform is disabled.	Turns Counter Not Supported.
130	Cannot use the Coordinate system that supports orientation with MCLM, MCCM and MCCD instructions. Corrective action: 1. Modify the coordinate definition of the Cartesian coordinate system to <none> OR 2. Use the configured Cartesian Coordinate system XYZRxRyRz with the MCPM instruction which supports orientation axes control.	Not allowed on the orientation supported geometries.

132	Operands of MCS should be set as follows. Otherwise the instruction will error. 1. If **Change Decel** is set to **Yes**, then **Decel Units** must be **% of Maximum**. Units per Sec2 is not acceptable because different decel value is needed for Cartesian and orientation axes. 2. **Change Decel Jerk** must be set to **Yes**. Setting it to **No** is not acceptable because different jerk value is needed for Cartesian and orientation axes. 3. **Decel Jerk Units** must be programmed in **% of time**. Units per Sec3 and % of Maximum, are not acceptable because different value is needed for Cartesian and orientation axes.	MCS Units conflict.
135	If MCPM speed, accel or decel units are programmed in % of max and the associated master driven MDCC instruction nominal master velocity is set to zero, the instruction will error.	MCPM zero nominal master velocity.
136	There are two possible situations that can result in this conflict: • MCPM continuous path (CP) move is programmed with robot configuration parameter different from the current robot configuration. • If there is already some CP move in the queue and the new programmed move robot configuration is different from the move in queue.	MCPM robot configuration conflict.
137	The robot configuration parameter for the MCTPO instruction is not valid for this Robot geometry.	Invalid robot configuration.
138	Refer to the extended error codes in the online help for the MCPM instruction for details related to this error code.	MCPM path data invalid value.
139	Refer to the extended error codes in the online help for the MCPM instruction for details related to this error code.	MCPM Dynamics Data invalid value.
140	Wait until the Servo On operation is complete.	Servo On in progress.
141	Wait until the Servo Off operation is complete.	Servo Off in progress.
142	Wait for the Shutdown Reset operation to complete.	Shutdown Reset in progress.
143	Wait for Axis Homing to complete.	Home in progress.
144	Wait for the Motion redefine position operation to complete.	Redefine in progress.
145	Wait for the Shutdown operation to complete.	Shutdown in progress.
146	Cannot start motion if the maximum orientation deceleration for the coordinate system is zero.	Maximum orientation deceleration is zero.
147	A orientation axis (Rx, Ry, or Rz) in the coordinate system has one of the following: • The conversion constant (planner feedback counts per position unit) is greater than 20,000 value. • The conversion constant is not an integer • A conversion ratio between Coordination Units and Position Units is not 1:1 Refer to the online help for the instruction for extended error code definitions. It identifies the orientation axis.	Invalid orientation scaling constant.

148	The MCTO or MCTPO instruction reports this error when the Orientation offset is not valid when: • Delta Robot Geometry is J1J2J6 or J1J2J3J6 and Workframe offset for Rx or Ry is not 0 or the Toolframe offset for Rx or Ry is not 0 • Delta Robot Geometry is J1J2J3J4J5 and Workframe offset for Rx or Ry is not 0 or the Toolframe offset for Rx or Rz is not 0 Refer to the online help for the instruction for extended error code definitions. It identifies the orientation axis.	MCTO or MCTPO orientation offset is not zero.
149	Orientation axes (Rx, Ry, and Rz) must be virtual if the coordinate system is involved in the MCTO or MCTPO transforms. Refer to the online help for the instruction for extended error code definitions. It identifies the orientation axis.	Orientation axis is not virtual.
150	The instruction reports this error when an orientation axis (Rx, Ry, or Rz) is used as a master axis of a gearing, camming, or master driven instructions	Master axis is orientation axis.
151	The Joint angle in a Delta J1JJ2J6, DeltaJ1J2J3J6, or Delta J1J2J3J4J5 geometry goes beyond the joint angle limit. Refer to the online help for the instruction for extended error code definitions. It identifies the Joint that is outside its range.	Joint angle beyond its limits.
152	Error occurs when the orientation axis is commanded to move by and angle greater than or equal to 180 degrees in one coarse update period. Refer to the online help for the instruction for extended error code definitions. It identifies the orientation axis.	Maximum orientation speed exceeded for an orientation axis.
153	The error occurs if the programmed Cartesian position (X, Y or Z) cannot be attained by the robot. For example: A Delta J1J2J6 robot can operate only in a 2D X-Z plane without a tool. If you program a Cartesian point with Y, a non-zero value, this error occurs with extended error 2 that represents invalid translation position Y. Refer to the online help for the instruction for extended error code definitions. It identifies the Joint that is outside its range.	Invalid translation position.
155	MCPM currently does not support any action on 6-axis robot geometry, so if attempted this error occurs.	Robot geometry is not supported in MCPM.
170	The instruction reports this error when a cam replacement is attempted when there is no active cam.	
171	The error occurs when attempting to replace a cam and replacement point not in range of cam profile.	
172	Attempting to replace a cam when Execution Schedule is “Pending”.	The cam type is “Replace and Restart” or “Replace and continue” and the Execution Schedule is “Pending”. Execution Schedule of “Pending” is only associated with cam type “New Cam”.

173	Cam completed before crossing the replacement position.	
174	The start point for cam replacement is beyond existing cam boundary.	 Example: An Active cam is locked, and Master Lock Position of a replacement cam is set to a point beyond the right cam boundary.
176	Motion Calculate Slave Values (MCSV) cannot find a Master value for the supplied Slave value.	
178	Selected "Cam Type" is not supported.	"Replace and Restart" and "Replace and Continue" are not supported on the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers.
179	The instruction reports this error when the cam profile array operand contains an invalid value (such as overflow or not a number). Use the Motion Calculate Cam Profile (MCCP) instruction or Cam Profile Editor to recalculate the cam profile and confirm that the master and slave values do not contain an invalid value.	For MATC, MAPC, and MCSV instructions, the .SEGMENT field of Motion Instruction identifies which cam profile element contains the invalid value. A .SEGMENT = 3 means the fourth element (or [3]) of the Cam Profile array contains the invalid number. Check the Master, Slave, C0, C1, C2, and C3 values of the Cam Profile array element.

You will get an error if certain Motion Instructions overlap while Motion Stop Instructions are active. In this case, an instruction is actively stopping and a second instruction is initiated that overlaps the active instruction. The table below lists some of the overlap instances that will generate errors.

In this case:

Error # 7 = Shutdown State Error.

Error #61, ExErr #10 = Connection Conflict, Transform Axes Moving or Locked By Other Operation.

Error #78 = Not Allowed While Stopping.

Generated Errors in Overlap Instances

The following table lists additional overlap instances that will generate errors.

	Active Stopping Instruction								
	MGS			**MGSD**	**MCS**			**MAS**	
Initiated Second Instruction	**Stop Mode = Fast Stop**	**Stop Mode = Fast Disable**	**Stop Mode = Programmed**		**Stop Type = Coordinated Move**	**Stop Type = Coordinated Transform**	**Stop Type = All**	**All Stop Types Except StopType = All**	**Stop Type = All**
MAAT	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MRAT	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MAHD	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78

MRHD	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MAH	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MAJ	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MAM	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MAG	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MCD	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MAPC	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MATC	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MDO	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MCT	Error #78	Error #78	Error #78	Error # 7	Error #61 ExErr #10	Error #61 ExErr #10	Error #61 ExErr #10	Error #61 ExErr #10	Error #61 ExErr #10
MCTO	Error #78	Error #78	Error #78	Error #7	Error #61 ExErr #10	Error #61 ExErr #10	Error #61 ExErr #10	Error #61 ExErr #10	Error #61 ExErr #10
MCCD	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MCLM/MCCM (Merge = Disabled)	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78		Error #78
MCLM/MCCM (Merge=Enabled)	Error #78	Error #78	Error #78	Error # 7		Error #78	Error #78		Error #78
MCPM	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78		Error #78

Additional Generated Errors in Overlap Instances

		Active Stopping Instruction						
		MGS			**MGSD**	**MCS**	**MAS**	**MASD**
Initiated Second Instruction	**Stop Type**	**Stop Mode = Fast Stop**	**Stop Mode = Fast Disable**	**Stop Mode = Programmed**	**None**	**Stop Type = All**	**Stop Type = All**	**None**
MGS	Stop Mode = Fast Stop	Error #78	Error #78	Error #78	Error #7			
	Stop Mode = Fast Disable	Error #78	Error #78	Error #78	Error #7			
	Stop Mode = Programmed	Error #78	Error #78	Error #78	Error #7			
MGSR	None	Error #78	Error #78	Error #78	Error #7			Error #7
MCS	Stop Type = Coordinated Move	Error #78	Error #78	Error #78	Error #7	Error #78	Error #78	
	Stop Type = Coordinated Transform	Error #78	Error #78	Error #78	Error #7	Error #78	Error #78	
	All Stop Types Except StopType = All	Error #78	Error #78	Error #78	Error #7			
MAS	Stop Type!= All	Error #78	Error #78	Error #78	Error #7	Error #78	Error #78	Error #7
	Stop Type = All	Error #78	Error #78	Error #78	Error #7			Error #7
MASR	None	Error #78	Error #78	Error #78	Error #7			Error #7

See also

Handle Motion Faults on page 594

Motion Attributes on page 595

Understand Motion Status and Configuration Parameters on page 639

Troubleshoot Axis Motion on page 639

Handle Motion Faults

Two types of motion faults exist.

Type	Description	Example
Errors	Do not impact controller operation Should be correct to optimize execution time and ensure program accuracy	A Motion Axis Move (MAM) instruction with a parameter out of range
Minor/Major	Caused by a problem with the servo loop Can shutdown the controller if you do not correct the fault condition	The application exceeded the servo loop position error tolerance

Understanding Errors

Executing a motion instruction within an application program can generate errors. The MOTION_INSTRUCTION tag has a field that contains the error code (any number from 1 to 23 depending on the error).

Understanding Minor/Major Faults

Several faults can occur that are not caused by motion instructions. For example, a loss of encoder feedback or an actual position exceeding an over-travel limit will cause faults. The motion faults are considered Type II faults with error codes from 1 to 32. For more information about motion error codes and handling faults, see Logix 5000 Controllers Major, Minor, and I/O Faults Programming Manual publication 1756-PM014.

Tip: You can configure a fault as either minor (non-major) or major by using the Axis Wizard-Group window.

Motion Attributes

Motion attributes are parameters that the Logix Designer application uses to define the operational characteristics of a motion module or axis, how the module or axis interacts with the system, and how the conversion of motion commands into actual motion of a device is performed.

Motion Attribute Folders

The motion attributes are divided into three main folders which indicate how the attributes are used:

- Interface
- Status
- Configuration

Data Type Groups

The motion attributes are divided into groups, which are designated by a G and a number. These groups depend on the specific data type, and an axis may be associated with many groups.

The following table lists the Group number and the axis data type it affects.

Group	AXIS_SERVO_DRIVE	AXIS_SERVO	Generic	AXIS_VIRTUAL	AXIS_CONSUMED	Feedback
G1	X	X	X	X	X	X
G2	X	X	X			X
G3					X	
G4	X	X	X	X	X	
G5	X	X	X		X	
G6	X	X	X	X		X
G7	X	X	X			
G8	X	X	X	X		
G9		X				
G10	X					
G11	X	X				
G12	X	X				X

Motion Attributes

The following table lists all of the attributes associated with motion (in alphabetical order).

Variable:	Data Type	Instruction	Description

AccelerationCommand	REAL	GSV	G11/Status The current acceleration reference to the output summing junction for the specified axis. Represents the output of the inner velocity control loop.
AccelerationDataScaling	INT	GSV	G10/Configuration This advanced scaling attribute maps directly to the SERCOS IDNs. It is automatically configured to the appropriate default, and is read-only.
AccelerationDataScalingExp	INT	GSV	G10/Configuration This advanced scaling attribute maps directly to the SERCOS IDNs. It is automatically configured to the appropriate default, and is read-only.
AccelerationDataScalingFactor	DINT	GSV	G10/Configuration This advanced scaling attribute maps directly to the SERCOS IDNs. It is automatically configured to the appropriate default, and is read-only.
AccelerationFeedback	REAL	GSV	G11/Status The actual velocity of the axis, as estimated by the servo module. It is calculated by taking the difference in the estimated velocity over the servo update interval. Acceleration Feedback is a signed value. The sign (+ or -) depends on the direction in which the axis is currently moving.
AccelerationFeedforwardGain	REAL	GSV/SSV	G11/Configuration The value used to provide the torque command output to generate the command acceleration. The optimal value is 100%.
AccelerationLimitBipolar	REAL	GSV/SSV	G10.Configuration This maps directly to the SERCOS IDNs and is automatically configured to reasonable default values. Manipulation is not needed unless motivated by a specific application requirement.
AccelerationLimitNegative	REAL	GSV/SSV	G10.Configuration This maps directly to the SERCOS IDNs and is automatically configured to reasonable default values. Manipulation is not needed unless motivated by a specific application requirement.
AccelerationLimitPositive	REAL	GSV/SSV	G10.Configuration This maps directly to the SERCOS IDNs and is automatically configured to reasonable default values. Manipulation is not needed unless motivated by a specific application requirement.
ActualAcceleration	REAL	GSV	G1/Status The current instantaneously measured acceleration of an axis. It is calculated as the current increment to the actual velocity per coarse update interval. This is a signed value, with the sign depending on the direction of the axis.
ActualPosition	REAL	GSV	G1/Status The actual position of your axis as of one coarse update period ago.
ActualVelocity	REAL	GSV	G1/Status The current instantaneously measured speed of an axis, calculated as the current increment to the actual position per coarse update interval. This is a signed value, with the sign depending on the direction of the axis. The internal resolution limit of the actual velocity is 1 encoder count per coarse update.
AssignedGroupInstance	DINT	GSV	G1/Interface The instance number of the motion group that contains your axis.
ATConfigurationList	DINT ARRAY		Maps directly to the SERCOS IDN, and is automatically set based on the current ServoLoopConfiguration. This value is read-only.
AttributeErrorCode	INT	GSV	G11/Status Returns the appropriate CIP error code when an Axis Configuration fault occurs. It is not reset to zero until the motion module is reconfigured.

<table>
<tr><td>AttributeErrorID</td><td>INT</td><td>GSV</td><td colspan="3">G11/Status
Retains the identification associated with the error code returned when an Axis Configuration fault occurred. The module must be reconfigured to reset this attribute to zero.</td></tr>
<tr><td>AuxFeedbackConfiguration</td><td>INT</td><td>GSV</td><td colspan="3">G10/Configuration
Maps directly to the SERCOS IDN, and is automatically set based on the current Drive Polarity Settings. All command bits are set according to the current Feedback Type value.</td></tr>
<tr><td>AuxFeedbackRatio</td><td>REAL</td><td>GSV</td><td colspan="3">G10/Configuration
Represents the quantitative relationship between the auxiliary feedback device and the motor. It is used in range limit and default calculations during configuration, based on the selected motor's specifications. It is also used by the drive when running the dual feedback loop configuration.</td></tr>
<tr><td>AuxFeedbackResolution</td><td>DINT</td><td>GSV</td><td colspan="3">G10/Configuration
Provides the A-B drive with the resolution of the associated feedback device in cycles. It also provides the Logix controller and drive with critical information to compute conversion constants used in converting drive units to feedback counts.</td></tr>
<tr><td>AuxFeedbackType</td><td>INT</td><td>GSV</td><td colspan="3">G10/Configuration
Provides the A-B drive with specific device configuration information for the auxiliary feedback device.</td></tr>
<tr><td>AuxPositionFeedback</td><td>REAL</td><td>GSV</td><td colspan="3">G11/Status
The current value of the position feedback coming from the auxiliary feedback input.</td></tr>
<tr><td>AverageVelocity</td><td>REAL</td><td>GSV</td><td colspan="3">G1/Status
The current speed of an axis, as calculated by averaging the actual velocity of the axis over the configured Average Velocity Timebase for the axis. It is a signed value to indicate the direction in which the axis is moving.</td></tr>
<tr><td>AverageVelocityTimebase</td><td>REAL</td><td>GSV</td><td colspan="3">G1/Configuration
Specifies the desired time in seconds for use in calculating the Average Velocity of the axis. The values should be large enough to filter out the small changes in velocity which would otherwise result in a noisy velocity value, yet small enough to track significant changes in axis velocity. A value between 0.25 and 0.50 seconds works for most applications.</td></tr>
<tr><td>AxisConfigurationState</td><td>SINT</td><td>GSV</td><td colspan="3">G1/Interface
Used for debugging purposes to indicate the present location of the axis in the axis configuration state-machine. Consumed and virtual axes use this attribute also.</td></tr>
<tr><td rowspan="5">AxisControlBits</td><td rowspan="5">DINT</td><td rowspan="5">GSV</td><td colspan="3">G11/Status</td></tr>
<tr><td>Bit</td><td>Bit Name</td><td>Meaning</td></tr>
<tr><td>0</td><td>Abort Process Request</td><td>When set, the servo module/drive disables any active tuning or testing process.</td></tr>
<tr><td>1</td><td>Shutdown Request</td><td>When set, the servo module forces the axis/drive into the shutdown state.</td></tr>
<tr><td>2</td><td>Zero DAC Request (Servo)
Reserved (Servo Drive)</td><td>When set, the servo module forces the DAC output for the axis to zero volts.</td></tr>
<tr><td></td><td>3</td><td>Abort Home Request</td><td>When set, any homing procedures are canceled.</td></tr>
</table>

			4	Abort Event Request	When set, any active watch or registration procedures are canceled.
			5-14	Reserved	
			15	Change Cmd Reference	This bit is set when the Logix controller switches to a new position coordinate system for command position.
			16-31	Reserved	
AxisDataType	SINT	GSV	G3/Interface Determines which data template, memory format, and set of attributes are created and applied for this axis instance. It can only be set as part of an axis create service. **Value: Meaning** 0 feedback 1 consumed 2 virtual 3 generic 4 servo 5 servo drive		
AxisEventBits	DINT	GSV	G1/Status The axis event bits for your axis.		
			Bit	**Bit Name**	**Meaning**
			0	WatchEventArmedStatus	Set when a watch event has been armed through execution of the MAW instruction.
			1	WatchEventStatus	Set when a watch event has occurred.
			2	Registration1EventArmedStatus	Set when registration checking has been armed for registration input 1 because a MAR instruction has been executed.
			3	Registration1EventStatus	Set when a registration event has occurred on registration input 1.
			4	Registration2EventArmedStatus	Set when registration checking has been armed for registration input 2 because a MAR instruction has been executed.
			5	Registration2EventStatus	Set when a registration event has occurred on registration input 2.
			6	HomeEventArmedStatus	Set when a home event has been armed through the execution of an MAH instruction.

			7	HomeEventStatus	Set when a home event has occurred.
			8-31	Reserved	
AxisFaultBits	DINT	GSV	G1/Status The axis fault bits for your axis.		
			Bit	**Bit Name**	**Meaning**
			0	PhysicalAxisFault	Set if there is one or more fault conditions reported by the physical axis.
			1	ModuleFault	Set when a serious fault has occurred with the motion module associated with the selected axis. 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.
			2	ConfigurationFault	Set when an update operation targeting an axis configuration attribute of an associated motion module has failed.
			3	GroupFault	Set when one or more faults have occurred related to the motion group associated with the selected axis. Usually a group fault affects all axes associated with the motion group. A group fault generally results in the shutdown of all associated axes. Reconfiguration of the entire motion subsystem is required to recover from a group fault condition.
			4	MotionFault	When set indicates one or more fault conditions have occurred related to the Motion Planner function.

			5	GuardFault	When set indicates that one or more fault conditions have occurred related to the embedded Guard Motion safety function of a drive. Guard Faults are only applicable if the drive device is equipped with "Hardwired" Guard Safety functionality.
			6	InitializationFault	Set when initialization of a CIP Motion device fails for any reason.
			7	APRFault	Set when during axis configuration the system is not able to recover the absolute position of the axis.
			8	SafetyFault	Indicates one or more fault conditions have occurred related to the axis safety function. Safety Faults are only applicable if the motion device supports "Networked" Safety functionality through a CIP Safety connection.
			9-31	Reserved	

AxisInfoSelect1	DINT	GSV/SSV	G1/Configuration Enables periodic data updates for selected servo/drive status attributes. It is designed to reduce the flow of unnecessary data for the Servo/SERCOS module. **Value: Meaning** 0 none 1 Position Command 2 Position Feedback 3 Aux. Position Feedback 4 Position Error 5 Position Int. Error 6 Velocity Command 7 Velocity Feedback 8 Velocity Error 9 Velocity Int. Error 10 Accel. Command 11 Accel. Feedback 12 Servo Output Level (Servo) 13 Marker Distance 14 Torque Command (Servo Drive) 15 Torque Feedback (Servo Drive) 16 Pos Dynamic Torque Limit (Servo Drive) 17 Neg Dynamic Torque Limit (Servo Drive) 18 Motor Capacity (Servo Drive) 19 Drive Capacity (Servo Drive) 20 Power Capacity (Servo Drive) 21 Bus Regulator Capacity (Servo Drive) 22 Motor Electrical Angle (Servo Drive) 23 Torque Limit Source (Servo Drive) 24 DC Bus Voltage (Servo Drive)
AxisInfoSelect1	DINT	GSV/SSV	G1/Configuration Enables periodic data updates for selected servo status attributes. It is designed to reduce the flow of unnecessary data for the Servo module. **Value: Meaning** 0 none 1 Position Command 2 Position Feedback 3 Aux. Position Feedback 4 Position Error 5 Position Int. Error 6 Velocity Command 7 Velocity Feedback 8 Velocity Error 9 Velocity Int. Error 10 Accel. Command 11 Accel. Feedback 12 Servo Output Level 13 Marker Distance
AxisInstance	DINT	GSV	G1/Interface Returns the instance number of an axis. Major fault records only contain the instance of the offending axis; this attribute is used to determine if the instance number matches that of the offending axis.
AxisResponseBits	DINT	GSV	G1/Status The axis fault bits for your axis. **Bit** / **Bit Name** / **Meaning**

			0	Abort Process Acknowledge	When set, the servo module accepts that the tuning or test process has been canceled.
			1	Shutdown Acknowledge	When set, the servo module accepts that the axis has been forced into the shutdown state.
			2	Zero DAC Acknowledge (Servo) Reserved (Servo Drive)	When set, the servo module accepts that the DAC output for the axis has been set to zero volts.
			3	Abort Home Acknowledge	When set, the servo module accepts that the active Homing procedure is aborted.
			4	Abort Event Acknowledge	When set, the servo module accepts that the active watch or registration procedure is canceled.
			5-14	Reserved	
			15	Change Pos Reference	Set when the Servo loop switches to a new position coordinate system. Logix controllers use this bit to account for the offset implied by the shift in reference point.
			16-31	Reserved	
AxisState	SINT	GSV	G1/Interface Indicates the operating state of the axis. **Value: Meaning** 0 Axis Ready 1 Direct Drive Control 2 Servo Control 3 Axis Faulted 4 Axis Shutdown		
AxisStatusBits	DINT	GSV	G1/Status The axis status bits for your axis.		
			Bit	**Bit Name**	**Meaning**
			0	ServoActionStatus	Set when the associated axis motor control function is tracking command reference from the controller.

			1	DriveEnableStatus	Set when the power structure associated with the axis is currently enabled. If the bit is not set then the power structure associated with the axis is currently disabled.
			2	AxisShutdownStatus	Set when the associated axis is currently in the Shutdown state. As soon as the axis is transitioned from the Shutdown state to another state, the Shutdown Status bit is cleared.
			3	ConfigurationUpdateInProcess	Provides a method of monitoring the progress of one or more specific module configuration attribute updates initiated by either a Set Attribute List service or an SSV in the user program. As soon as such an update is initiated, the Logix processor sets the ConfigurationUpdateIn Process bit. The bit will remain set until the Set Attribute List reply comes back from the servo module indicating that the data update process was successful. Thus the Configuration Update Status Bits attribute provides a method of waiting until the servo configuration data update to the connected motion module is complete before starting a dependent operation.

			4	InhibitStatus	Set when the axis is in the inhibited state. This bit can also be used to determine when an inhibit/uninhibit operation has been completed (for example, when the connection has been shutdown, reconnected and then the reconfiguration process completed). During the inhibit/uninhibit process this bit remains in the previous state and then after completion it is updated to the new state.

			5	DirectControlStatus	Set when axis motion is driven by the Direct Velocity Control and Direct Torque Control functions. In this mode, the Motion Planner functionality is disabled so any attempt to move the axis with a Motion Planner instruction, such as, MAM, MAJ, MAG, and so on results in an instruction error. Furthermore, in Direct Control there is no need to establish or maintain absolute reference position so attempted execution of MAH and MRP instructions also results in an instruction error. When the Direct Control Status bit is clear axis motion is controlled by the Motion Planner. Any attempt to move the axis in this mode with a Direct control instruction, such as MDS, results in an instruction error. This bit only applies to CIP Drive axis types. The Direct Control Status bit is set by the Motion Drive Start instruction (MDS) and once set, can only be cleared by executing an MSO instruction from the Stopped or Stopping State. Similarly, once the Direct Control Status bit is cleared by the Motion Servo On instruction (MSO), the bit can only be set again by executing an MDS instruction from the Stopped or Stopping State.

			6	AxisUpdateStatus	This bit indicates whether or not this axis instance was updated in last execution of Motion Task. In general, axis instances are updated in Motion Task according to their Axis Update Schedule. Thus, a given axis instance may or may not be updated during Motion Task execution. When inspected as part of an Event Task triggered by Motion Group Execution, the Axis Update bit can be used to qualify program instructions based on whether or not the axis was updated by the preceding Motion Task.
			7-31	Reserved	
AxisStructureAddress	DINT		G1/Interface Used to return the actual physical address in memory where the axis instance is located.		
AxisType	INT	GSV/SSV	G11/Configuration Establishes the intended use for the axis. This value also controls the behavior of the servo module's Axis Status LEDs and the Logix Designer application uses the current configured type to control the look of its tab dialogs. Value: Meaning 0 unused 1 feedback only 2 servo		
BrakeEngageDelayTime	REAL	GSV/SSV	G10/Configuration Controls the amount of time the drive continues to apply torque to the motor after the brake output is changed to engage the brake.		
BrakeReleaseDelayTime	REAL	GSV/SSV	G10/Configuration Controls the amount of time the drive holds off from tracking command reference changes after the brake output is changed to release the brake.		
BusRegulatorCapacity	REAL	GSV/SSV	G10/Status Displays the present utilization of the axis bus regulator as a percent of rated capacity.		
BusRegulatorID	INT	GSV	G10/Configuration Contains the enumeration of the specific A-B Bus Regulator or System Shunt catalog numbers associated with the axis. If it does not match, an error is generated.		
C2CConnectionInstance	DINT	GSV	G1/Interface Used with the C2CMapInstance attribute to associate the axis to the consumed data when the AxisDataType is set to Consumed. This attribute is the connection instance under the C2C map instance, which provides the axis data sent to it from another axis via the C2C connection.		

C2CMapInstance	DINT	GSV	G1/Interface Used with the C2CConnectionInstance attribute to associate the axis to the consumed data when the AxisDataType is set to Consumed. For all other axis data types, if the axis is to be produced, then the attribute is set to 1 to indicate the connection is off the local controller's map instance.
CommandAcceleration	REAL	GSV	G4/Status The commanded speed of an axis as generated by any previous motion instruction. It is calculated as the current increment to the command velocity per coarse update interval. This is a signed value with the sign, with the sign depending on the direction of the axis.
CommandPosition	REAL	GSV	G4/Status The command position of the axis that is to be acted upon one coarse update period from now.
CommandVelocity	REAL	GSV	G4/Status The commanded speed of an axis as generated by any previous motion instructions. It is a signed value, with the sign indicating the direction of the axis
ConversionConstant	REAL	GSV/SSV	G1/Configuration Must be established for each axis to let axis position be displayed and motion to be programmed in the position units specified by the PositionUnits attribute. It is also known as the K Constant, where K represents the number of feedback counts per position unit.
DampingFactor	REAL	GSV/SSV	G11/Configuration The value used in calculating the maximum position servo bandwidth during the execution of the MRAT instruction. It controls the dynamic response of the servo axis. The default value of 0.8 should work fine for most applications
DCBusVoltage	DINT	GSV	G10/Status The present voltage on the DC Bus of the drive.
DriveAxisID	INT	GSV	G10/Configuration Contains the CIP Product Code of the drive amplifier associated with the axis. If the codes don't match, an error is returned during the configuration process.
DriveCapacity	REAL	GSV	G10/Status Represents the present utilization of drive capacity as a percent of rated capacity.
DriveFaultAction	SINT	GSV/SSV	G9/Configuration Sets the type of operation to be performed when a drive fault occurs. **Value** — **Meaning** 0 — shutdown - most severe action to a fault; reserved for faults that could endanger machinery or personnel 1 — disabled drive - when the fault occurs, the drive is immediately disabled, the servo output zeroed, and the drive enable output deactivated 2 — stop command - when the fault occurs, the axis immediately begins decelerating the axis command position to a stop at the configured Maximum Deceleration Rate, without disabling servo action or the servo module's Drive Enable Output. This is the gentlest stopping action for a fault 3 — status only - when set to status only, motion faults must be handled by the application program.
DriveFaultBits	DINT	GSV	G10/Status **Bit** **Bit Name** **Meaning**

			0	PosSoftOvertravelFault	Sets when the axis travels or attempts to travel beyond the configured value for Maximum Positive Travel.
			1	NegSoftOvertravelFault	Sets when the axis travels or attempts to travel beyond the configured value for Maximum Negative Travel.
			2	PosHardOvertravelFault	Sets when the axis travels or attempts to travel beyond the maximum position limit set by the hardware limit switch mounted to the machine.
			3	NegHardOvertravelFault	Sets when the axis travels or attempts to travel beyond the minimum position limit set by the hardware limit switch mounted to the machine.
			4	FeedbackFault	Sets when the differential electrical signal for one or more of the feedback channels are at same level or loss of feedback power or common electrical connection between servo module or drive and the feedback device.
			5	FeedbackNoiseFault	Sets when simultaneous transitions of feedback of A and B are detected by the servo module.
			6	AuxFeedbackFault	Sets when the differential electrical signal for one or more of the feedback channels are at same level or loss of feedback power or common electrical connection between servo module or drive and the auxiliary feedback device.

			7	AuxFeedbackNoiseFault	Sets when simultaneous transitions of feedback of A and B are detected by the servo module.
			8-12	Reserved	
			13	GroundShortFault	Sets when the drive detects an imbalance in the DC bus supply current – indicating that current is flowing through an improper ground connection.
			14	DriveHardFault	Set when the drive detects a serious hardware fault.
			15	OverspeedFault	Sets when the speed of the axis exceeds the overspeed limit which is typically set to 150% of configured velocity limit for the motor.
			16	OverloadFault	Sets if the load limit for the motor/drive persists after the Overload warning bit has been set.
			17	DriveOvertempFault	Sets when the drive's temperature exceeds the drive shutdown temperature.
			18	MotorOvertempFault	Sets when the motor's temperature exceeds the motor shutdown temperature.
			19	DriveCoolingFault	Sets when the ambient temperature surrounding the drive's control circuitry exceeds the ambient shutdown temperature.
			20	DriveControlVoltageFault	Sets when the power supply voltages fall outside of acceptable limits.
			21	FeedbackFault	Sets when one of the feedback sources has a problem that prevents the drive from receiving accurate or reliable position information from the feedback device.

			22	CommutationFault	Sets when the commutation feedback source has a problem that prevents the drive from receiving accurate or reliable motor shaft information to perform commutation.
			23	DriveOvercurrentFault	Sets when the drive output current exceeds the drive's predefined operating limits.
			24	DriveOvervoltageFault	Sets when drive DC bus voltage exceeds the bus's predefined operating limits.
			25	DriveUndervoltageFault	Sets when drive DC bus voltage is below the bus's predefined operating limits.
			26	PowerPhaseLostFault	Sets when the drive detects that one or more of the three power line phases is lost from the 3 phase power inputs.
			27	PositionErrorFault	Sets when the axis position error exceeds the value set for Position Error Tolerance.
			28	SERCOSFault	Sets either when a requested SERCOS procedure fails to execute properly or the drive node detects a SERCOS communication fault.
			29	OvertravelFault	
			30-31	Reserved	
DriveModeTimeConstant	REAL	GSV/SSV	G11/Configuration The sum of the drive's current loop time constant, feedback sample period, and the time constant associated with the velocity feedback filter. It is used by the MRAT instruction to calculate the Maximum Velocity and Position Servo bandwidth values when the axis is configured for an External Torque Servo Drive.		
DrivePolarity	DINT	GSV/SSV	G10/Configuration		
			Bit	**Bit Name**	**Meaning**

<table>
<tr><td></td><td></td><td></td><td>0</td><td>CustomPolarity</td><td>Used to let you tailor the polarity configuration, using the various polarity parameters defined by the SERCOS interface standard.</td></tr>
<tr><td></td><td></td><td></td><td>1</td><td>PositivePolarity</td><td>Set automatically, using the MRHD and MAHD instructions.</td></tr>
<tr><td></td><td></td><td></td><td>2</td><td>NegativePolarity</td><td>Set automatically, using the MRHD and MAHD instructions.</td></tr>
<tr><td>DriveScalingBits</td><td>DINT</td><td>GSV</td><td colspan="3">G10/Configuration
The custom scaling bit is used to enable custom scaling by the various scaling parameters defined by the SERCOS interface standard. When the bit is clear (default), the scaling parameters are all set based on the preferred Rockwell automation SERCOS drive scaling factors.
Value: Meaning
0 Standard/CustomScaling
1-31 Reserved</td></tr>
<tr><td>DriveStatusBits</td><td>DINT</td><td>GSV</td><td colspan="3">G10/Status
These are the status bits for the drive.</td></tr>
<tr><td></td><td></td><td></td><td>Bit</td><td>Bit Name</td><td>Meaning</td></tr>
<tr><td></td><td></td><td></td><td>0</td><td>ServoActionStatus</td><td>Set when servo loops on the associated axis is enabled and able to follow commands.</td></tr>
<tr><td></td><td></td><td></td><td>1</td><td>DriveEnableStatus</td><td>Set when drive's power structure associated with the axis has been activated.</td></tr>
<tr><td></td><td></td><td></td><td>2</td><td>AxisShutdownStatus</td><td>Set when the associated axis is in shutdown state.</td></tr>
<tr><td></td><td></td><td></td><td>3</td><td>ProcessStatus</td><td>Set when axis tuning or hookup diagnostic operation is in progress on the associated physical axis.</td></tr>
<tr><td></td><td></td><td></td><td>4</td><td>Reserved</td><td></td></tr>
<tr><td></td><td></td><td></td><td>5</td><td>Reserved</td><td></td></tr>
<tr><td></td><td></td><td></td><td>6</td><td>HomeInputStatus</td><td>Represents the current state of the dedicated homing input – it is set when Home input is active.</td></tr>
<tr><td></td><td></td><td></td><td>7</td><td>Registration1InputStatus</td><td>Represents the current state of the dedicated registration 1 input – it is set when registration 1 input is active.</td></tr>
</table>

			8	Registration2InputStatus	Represents the current state of the dedicated registration 2 input – it is set when registration 2 input is active.
			9	PositiveOvertravelInputStatus	Represents the current state of the dedicated Positive Overtravel input – set when Positive Overtravel input is active.
			10	NegativeOvertravelInputStatus	Represents the current state of the dedicated Negative Overtravel input – set when Negative Overtravel input is active.
			11	EnableInputStatus	The current state of the dedicated enable input – set when it is active.
			12	AccelerationLimitStatus	Set when the magnitude of the commanded acceleration to the velocity servo loop is greater than the configured Velocity Limit.
			13	AbsoluteReferenceStatus	Sets after an absolute homing procedure – it remains set until the drive resets its configuration parameters to default values or an active/passive home is performed.
			14	Reserved	
			15	Reserved	
			16	VelocityLockStatus	Set when the magnitude of the physical axis Velocity Feedback is within the configured Velocity Window of the current velocity command.
			17	VelocityStandstillStatus	Set when the magnitude of the physical axis Velocity Feedback is within the configured Velocity Standstill Window of zero speed.

			18	VelocityThresholdStatus	Set when the magnitude of the physical axis Velocity Feedback is less than the configured Velocity Threshold.
			19	TorqueThresholdStatus	Not available.
			20	TorqueLimitStatus	Set when the magnitude of the axis torque command is greater than the configured Torque Limit.
			21	VelocityLimitStatus	Set when the magnitude of the commanded velocity to the velocity servo loop input is greater than the configured Velocity Limit.
			22	PositionLockStatus	Set when the magnitude of the axis position error becomes less than or equal to the configured Position Lock Tolerance value for the associated physical axis.
			23	PowerLimitStatus	Not available.
			24	Reserved	
			25	LowerVelocityThresholdStatus	Not available.
			26	HighVelocityThresholdStatus	Not available.
			27-31	Reserved	
DriveThermalFaultAction	SINT	GSV/SSV	G10/Configuration Sets the type of action performed when a drive thermal fault occurs. **Value/Meaning** 0 = shutdown - most severe action to a fault; reserved for faults that could endanger machinery or personnel 1 = disabled drive - when the fault occurs, the drive is immediately disabled, the servo output zeroed, and the drive enable output deactivated 2 = stop command - when the fault occurs, the axis immediately begins decelerating the axis command position to a stop at the configured Maximum Deceleration Rate, without disabliing servo action or the servo module's Drive Enable Output. This is the gentlest stopping action for a fault 3 = status only - when set to status only, motion faults must be handled by the application program.		
DriveWarningBits	DINT	GSV	G10/Status These are the status bits for the drive.		
			Bit	**Bit Name**	**Meaning**

			0	DriveOverloadWarning	Set when motor load limit is exceeded. It gives the control program a chance to reduce motor loading to avoid a shutdown situation.
			1	DriveOvertempWarning	Set when drive's temperature limit is exceeded. It gives the program a chance to increase drive cooling before causing a shutdown situation.
			2	MotorOvertempWarning	Set when the motor's temperature limit is exceeded. It gives the program a chance to increase motor cooling before entering a shutdown situation.
			3	CoolingErrorWarning	Set when ambient temperature limit inside the drive enclosure is exceeded. It gives the program a chance to increase drive cooling to avoid a shutdown situation.
			4-31	Reserved	
ExternalDriveType	DINT	GSV	G9/Configuration Set the attribute to velocity servo drive when the servo module axis is interfacing with an external velocity servo drive to disable the servo modules internal digital velocity loop. When configuring as a torque servo drive, select torque servo drive to enable the servo module axis' internal digital velocity loop. **Value: Meaning** 0 torque servo drive 1 velocity servo drive		
FaultConfigurationBits	DINT	GSV/SSV	G10/Status These are the status bits for the drive.		
			Bit	**Bit Name**	**Meaning**
			0	SoftOvertravelChecking	When set, it enables a periodic test that monitors the axis' current position and issues a Positive/Negative Overtravel Fault if it travels beyond the set limits

			1	DriveFaultChecking (Servo)	Set this bit if you are using the servo module's drive fault input and specify the drive fault contact configuration of the amplifier's drive fault output.
			1	HardOvertravelChecking (Servo Drive)	When set, it enables a periodic test to monitor the current state of the positive and negative overtravel limit switch inputs and issues a Positive/Negative Hard Overtravel Fault if the axis position travels activate the limit switches.
			2	DriveFaultNormallyClosed (Servo)	This attribute controls sense of the Drive Fault input to the servo module.
			2-31	Reserved (Servo Drive)	
			3-31	Reserved (Servo)	
FeedbackFaultAction	SINT	GSV/SSV	G11/Configuration Sets the type of action to be taken when a Feedback Fault is encountered. **Value/Meaning** 0 = shutdown - most severe action to a fault; reserved for faults that could endanger machinery or personnel 1 = disabled drive - when the fault occurs, the drive is immediately disabled, the servo output zeroed, and the drive enable output deactivated 2 = stop command - when the fault occurs, the axis immediately begins decelerating the axis command position to a stop at the configured Maximum Deceleration Rate, without disabliing servo action or the servo module's Drive Enable Output. This is the gentlest stopping action for a fault 3 = status only - when set to status only, motion faults must be handled by the application program.		
FeedbackNoiseFaultAction	SINT	GSV/SSV	G11/Configuration Sets the type of action to be taken when a Feedback Noise Fault is encountered. **Value/Meaning** 0 = shutdown - most severe action to a fault; reserved for faults that could endanger machinery or personnel 1 = disabled drive - when the fault occurs, the drive is immediately disabled, the servo output zeroed, and the drive enable output deactivated 2 = stop command - when the fault occurs, the axis immediately begins decelerating the axis command position to a stop at the configured Maximum Deceleration Rate, without disabliing servo action or the servo module's Drive Enable Output. This is the gentlest stopping action for a fault 3 = status only - when set to status only, motion faults must be handled by the application program.		

<table>
<tr><td>FrictionCompensation</td><td>REAL</td><td>GSV/SSV</td><td>G11/Configuration
The fixed output level added to or subtracted from the Servo Output, based on its current sign, to break static friction, or sticktion. This value should be just under the value needed to break the sticktion because a larger value can cause the axis to dither.</td></tr>
<tr><td>HardOvertravelFaultAction</td><td>SINT</td><td>GSV/SSV</td><td>G11/Configuration
Sets the type of action performed when a hard overtravel fault occurs.
Value/Meaning
0 = shutdown - most severe action to a fault; reserved for faults that could endanger machinery or personnel
1 = disabled drive - when the fault occurs, the drive is immediately disabled, the servo output zeroed, and the drive enable output deactivated
2 = stop command - when the fault occurs, the axis immediately begins decelerating the axis command position to a stop at the configured Maximum Deceleration Rate, without disabliing servo action or the servo module's Drive Enable Output. This is the gentlest stopping action for a fault
3 =status only - when set to status only, motion faults must be handled by the application program.</td></tr>
<tr><td>HomeConfigurationBits</td><td>DINT</td><td>GSV/SSV</td><td>G6/Configuration
<table><tr><th>Bit</th><th>Bit Name</th><th>Meaning</th></tr><tr><td>0</td><td>Reserved</td><td></td></tr><tr><td>1</td><td>HomeSwitchNormallyClosed</td><td>This determines the normal state of the home limit switch used by the homing sequence. The normal state of the switch is its state prior to being engaged by the axis during the homing sequence.</td></tr><tr><td>2</td><td>Reserved</td><td></td></tr><tr><td>3-31</td><td>Reserved</td><td></td></tr></table></td></tr>
<tr><td>HomeDirection</td><td>SINT</td><td>GSV/SSV</td><td>G6/Configuration
The homing direction mode for your axis.
Value: Meaning
0 unidirectional forward
1 bidirectional forward
2 unidirectional reverse
3 bidirectional reverse</td></tr>
<tr><td>HomeMode</td><td>SINT</td><td>GSV/SSV</td><td>G6/Configuration
The homing mode for your axis.
Value/Meaning
0 = passive homing - redefines the current absolute position of the axis on the occurrence of a home switch or encoder marker event. This is used to calibrate uncontrolled axes.
1 = active homing (default) - when you choose this option, the desired homing sequence is selected by specifying whether or not a home limit switch and/or encoder marker is used for this axis.
2 = absolute - establishes the true absolute position of the axis by applying the configured Home Position to the reported position of the absolute feedback device.</td></tr>
</table>

HomeOffset	REAL	GSV/SSV	G6/Configuration When applied to active or passive homing, using a non-immediate sequence, the HomeOffset is the desired position offset of the axis HomePosition from the position at which the home event occurred. Home Offset is applied at the end of the specified homing sequence before the axis moves to the HomePosition.
HomePosition	REAL	GSV/SSV	G6/Configuration The desired absolute position for the axis after the specified homing sequence has been completed.
HomeReturnSpeed	REAL	GSV/SSV	G7/Configuration Controls the speed of the jog profile used after the first leg of an active bidirectional homing sequence.
HomeSequence	SINT	GSV/SSV	G6/Configuration The homing sequence type for your axis. **Value: Meaning:** 0 immediate (default) 1 switch 2 marker 3 switch then marker
HomeSpeed	REAL	GSV/SSV	G7/Configuration Controls the speed of the jog profile used in the first leg of an active homing sequence.
IntegratorHoldEnable	SINT	GSV/SSV	G11/Configuration When configured as TRUE, the servo loop temporarily disables any enabled integrators while command position is changing. It is used by point-to-point moves to minimize the integrator wind-up during motion. When FALSE, all active integrators are always on.
InterpolatedActualPosition	REAL	GSV	G1/Status It is the interpolation of the actual position, based on past axis trajectory history, at the time specified by the InterpolatedTime attribute.
InterpolatedCommandPosition	REAL	GSV	G4/Status It is the interpolation of the commanded position, based on past axis trajectory history, at the time specified by the InterpolatedTime attribute.
InterpolationTime	DINT	GSV/SSV	G1/Status It is the 32-bit CST time used to calculate the interpolated positions. When this attribute is updated with a valid CST value, the InterpolatedActual and InterpolatedCommand positions are automatically updated.
MapInstance	DINT	GSV	G2/Interface The I/O map instance of the motion compatible module. This attribute can only be set if you did not assign the axis to a group or if you assigned it to a group in the group inhibit mode.
MarkerDistance	REAL	GSV	G11/Status The distance between the axis position where the home switch input was detected and the axis position where the marker event was detected. It is useful in aligning a home limit switch relative to a feedback marker pulse to provide a repeatable homing operation.
MasterOffset	REAL	GSV	G4/Status The position offset that is applied to the master side of the position cam. It shows the same unwind characteristic as the position of a linear axis.
MaximumAcceleration	REAL	GSV/SSV	G8/Configuration Used by motion instructions to determine the acceleration rates to apply to the axis. The value is automatically set to ~ 85% of the measured Tuning Acceleration by the MAAT instruction

<table>
<tr><td>MaximumDeceleration</td><td>REAL</td><td>GSV/SSV</td><td>G8/Configuration
Used by motion instructions to determine the deceleration rates to apply to the axis. The value is automatically set to ~ 85% of the measured Tuning Deceleration by the MAAT instruction</td></tr>
<tr><td>MaximumNegativeTravel</td><td>REAL</td><td>GSV/SSV</td><td>G11/Configuration
Sets the software travel limit for the negative value. If the axis passes beyond the set limit a Software Overtravel fault is issued. This value must be less than the MaximumPositiveTravel value</td></tr>
<tr><td>MaximumPositiveTravel</td><td>REAL</td><td>GSV/SSV</td><td>G11/Configuration
Sets the software travel limit for the positive value. If the axis passes beyond the set limit a Software Overtravel fault is issued. This value must be greater than the MaximumNegativeTravel value.</td></tr>
<tr><td>MaximumSpeed</td><td>REAL</td><td>GSV/SSV</td><td>G8/Configuration
This is used by various motion instructions to determine the steady-state speed of the axis. This value for the axis is automatically set to the tuning speed by the MAAT instruction. It is usually set to ~90% of the maximum speed rating for the motor.</td></tr>
<tr><td>MDTConfigurationList</td><td>DINT ARRAY</td><td></td><td>Maps directly to the SERCOS IDN. It is automatically set based on the current ServoLoopConfiguration and is Read-only.</td></tr>
<tr><td>MemoryUsage</td><td>DINT</td><td>GSV</td><td>G3/Interface
Determines the amount of memory the created instance consumes in bytes.</td></tr>
<tr><td>MemoryUse</td><td>INT</td><td>GSV</td><td>G1/Interface
The Logix Designer application uses this attribute to create axis instances in I/O memory for axes that are either produced or consumed. It can only be set as part of an axis create service and it controls in which controller memory the object instance is created.</td></tr>
<tr><td>ModuleChannel</td><td></td><td></td><td>G2/Interface
Used to associate an axis to a specific channel on a motion compatible module by specifying the ModuleChannel attribute.</td></tr>
<tr><td>ModuleClassCode</td><td>DINT</td><td>GSV</td><td>G12/Interface
The CIP class code of the object in the motion module which is supporting motion.</td></tr>
<tr><td>ModuleFaultBits</td><td>DINT</td><td>GSV</td><td>G11/Status
Module Fault information is passed from a physical module or device to the controller via an 8-bit value contained in the header of the Synchronous Input connection assembly. These fault bits are updated every coarse update period by the Motion Task
<table>
<tr><th>Bit</th><th>Bit Name</th><th>Meaning</th></tr>
<tr><td>0</td><td>ControlSyncFault</td><td>Sets when the Logix controller detects that several position update messages in a row have been missed due to failure of the synchronous communications connection.</td></tr>
</table></td></tr>
</table>

			1	ModuleSyncFault	Sets when the motion module detects that several position update messages in a row have been missed due to failure of the synchronous communications connection.
			1-31	Reserved (Motion Axis Object)	
			2	Timer Event Fault	Sets when the servo module detects a problem with the module's timer event functionality.
			3	ModuleHardwareFault	Sets when the associated servo module detects a hardware problem that is going to need replacement of the module to correct.
			4-31	Reserved (Servo)	
			4	SERCOSRingFault (Servo Drive)	Set when SERCOS module detects that a problem has occurred on the SERCOS ring, such as a light has been broken or drive is powered down.
			5	Reserved (Servo Drive)	
MotionStatusBits	DINT	GSV	G1/Status The motion status bits for your axis.		
			Bit	**Bit Name**	**Meaning**
			0	AccelStatus	Used to determine if the axis is currently being commanded to accelerate.
			1	DecelStatus	Used to determine if the axis is currently being commanded to decelerate.
			2	MoveStatus	Set if a move motion profile is in progress.
			3	JogStatus	Set if a jog motion profile is in progress.
			4	GearingStatus	Set if an axis is gearing to another axis.
			5	HomingStatus	Set if a home motion profile is currently in progress.

			6	StoppingStatus	Set if there is a stopping process currently in progress. This bit is used to stop an axis initiated by an MAS, MGS, MGSP, Stop Motion fault action, or mode change.
			7	AxisHomedStatus	Set to 1 by the MAH instruction upon successful completion of the configured homing sequence. This bit indicates that an absolute machine reference position has been established. When this bit is set, operations that require a machine reference, such as Software Overtravel checking can be enabled. This bit is cleared under the following conditions: • Download, Control power cycle, or Reconnection with Incremental Feedback device. • Absolute Position Recovery (APR) fails with Absolute Feedback device. • Feedback Integrity bit is cleared by CIP Motion drive. The AxisHomedStatus bit is directly used by the control system to qualify the Software Overtravel checking function. If the AxisHomedStatus bit is clear, Soft Overtravel checking will not occur even if the Soft Overtravel Checking bit is set.
			8	PositionCamStatus	Set when a Pcam profile is currently in progress.
			9	TimeCamStatus	Set when a Tcam profile is currently in progress.

			10	PositionCamPendingStatus	Set if a Pcam motion profile is waiting for another to end. This is initiated by executing an MAPC instruction with Pending execution selected.
			11	TimeCamPendingStatus	Set if a Tcam motion profile is waiting for another to end. This is initiated by executing an MATC instruction with Pending execution selected.
			12	GearingLockStatus	Set whenever the slave axis is locked to the master axis in a gearing relationship.
			13	PositionCamLockStatus	Sets when the master axis meets the starting condition of an active Pcam motion profile.
			14	TimeCamLockStatus	Set when the master axis meets the starting condition of an active Tcam motion profile.
			15	MasterOffsetMoveStatus	Set if a Master Offset Move motion profile is in progress.
			16	CoordinatedMotionStatus	Set if a Coordinate Motion profile is in progress.
			17	TransformStateStatus	Set if a Transform State profile is in progress.
			18	ControlledByTransformStatus	Set if a Control by Transform profile is in progress.
			19	DirectVelocityControlStatus	Set when the axis speed is directly controlled by the Direct Command Velocity value. This bit is set by the Motion Drive Start instruction (MDS) and only applies to CIP Drive axis types.

			20	DirectTorqueControlStatus	Set when the axis torque is directly controlled by the Command Torque value. This bit is set by the Motion Drive Start instruction (MDS) and only applies to CIP Drive axis types.
			21	MovePendingStatus	
			22	MoveLockStatus	Set when the master axis satisfies the Lock Direction request of a Motion Axis Move (MAM) Instruction. If the Lock Direction is Immediate Forward Only or Immediate Reverse Only the MoveLockStatus bit will be set immediately when the MAM is initiated. If the Lock Direction is Position Forward Only or Position Reverse Only the bit will be set when the Master Axis crosses the Master Lock Position in the specified direction. This bit is cleared when the Master Axis reverses direction and the Slave Axis stops following the Master Axis. The MoveLockStatus bit is set again when the Slave Axis resumes following the Master Axis.
			23	JogPendingStatus	

			24	JogLockStatus	Set when the master axis satisfies the Lock Direction request of a Motion Axis Jog (MAJ) Instruction. If the Lock Direction is Immediate Forward Only or Immediate Reverse Only the JogLockStatus bit will be set immediately when the MAJ is initiated. If the Lock Direction is Position Forward Only or Position Reverse Only the bit will be set when the Master Axis crosses the Master Lock Position in the specified direction. This bit is cleared when the Master Axis reverses direction and the Slave Axis stops following the Master Axis. The JogLockStatus bit is set again when the Slave Axis resumes following the Master Axis.
			25	MasterOffsetMovePendingStatus	

			26	MasterOffsetMoveLockStatus	Set when the master axis satisfies the Lock Direction request of a Master Offset Move executed using MAM instruction. If the Lock Direction is Immediate Forward Only or Immediate Reverse Only the MasterOffsetMoveLock Status bit will be set immediately when the MAM is initiated. If the Lock Direction is Position Forward Only or Position Reverse Only the bit will be set when the Master Axis crosses the Master Lock Position in the specified direction. This bit is cleared when the Master Axis reverses direction and the Slave Axis stops following the Master Axis. The MasterOffsetMoveLock Status bit is set again when the Slave Axis resumes following the Master Axis.
			27	MaximumSpeedExceeded	Set when the axis command velocity at any time exceeds the maximum speed configured for an axis. This bit is cleared when the axis velocity is reduced below the maximum speed.
			28-31	Reserved	
MotorCapacity	REAL	GSV	G10/Status Displays the present utilization of motor capacity as a percent of rated capacity.		
MotorData	SINT ARRAY		A structure with a length element and an array of bytes that contain motor configuration information needed by the drive to operate the motor.		
MotorElectricalAngle	REAL	GSV	Shows the present electrical angle of the motor shaft.		
MotorFeedbackConfiguration	INT	GSV	Maps directly to the SERCOS IDN. It is automatically set based on the current Drive Polarity Settings. All command bits are set according to the current Feedback Type value.		

MotorFeedbackResolution	DINT	GSV	Provides the A-B drive with the resolution of the feedback device in cycles. It also gives the Logix controller and drive with critical information to compute conversion constants used in converting drive units to feedback counts
MotorFeedbackType	INT	GSV	Provides the A-B drive with specific device configuration information for the mounted motor.
MotorID	DINT	GSV	Contains the specific A-B motor catalog number associated with the axis. If the Motor ID does not match that of the actual motor, an error is generated during the drive configuration process.
MotorThermalFaultAction	SINT	GSV	Sets the type of action performed when a motor thermal fault occurs. **Value/Meaning:** 0 = shutdown – most severe action to a fault and is reserved for faults that could endanger machinery or personnel. 1 = disabled drive – when the fault occurs the drive is immediately disabled, the servo output zeroed, and drive enable output is deactivated 2 = stop command – when fault occurs the axis immediately begins decelerating the axis command position to a stop at the configured Maximum Deceleration Rate without disabling servo action or the servo module's Drive Enable Output. This is the gentlest stopping action for a fault. 3 = status only – when set to status only, motion faults must be handled by the application program
NegativeDynamicTorqueLimit	REAL	GSV	G10/Status Represents the maximum negative torque/current limit magnitude. It should be the lowest value of all torque/current limits in the drive at a given time. It includes the amplifier peak limit, motor peak limit, user current limit, amplifier thermal unit, and the motor thermal limit.
OutputCamExecutionTargets	DINT	GSV	G1/Interface Used to specify the number of output cam nodes attached to the axis. It can only be set as part of an axis create service and dictates the number of output cam nodes created and attached to that axis. The ability to configure the number of Output Execution Targets for a specific axis reduces the memory required per axis for users who do not need Output Cam functionality.
OutputCamLockStatus	DINT	GSV	G1/Status Set when an Output Cam has been armed. It is initiated by an MAOC instruction with Immediate execution selected, when a pending output cam changes to armed, or when the axis approaches or passes through the specified arm position.
OutputCamPendingStatus	DINT	GSV	G1/Status Set if an OutputCam is currently pending completion of another Output Cam initiated by executing an MAOC instruction with Pending execution selected.
OutputCamStatus	DINT	GSV	G1/Status Set when an Output Cam has been initiated. The bit is reset when the cam position moves beyond the cam start or cam end position in Once execution mode with no Output Cam pending or when the Output Cam is ended by an MDOC instruction.
OutputCamTransitionStatus	DINT	GSV	G1/Status Set when a transition between the currently armed and pending Output Cam is in process.
OutputLimit	REAL	GSV/SSV	G9/Configuration Provides a method of limiting the maximum servo output voltage of a physical axis to a specified level.

OutputPFilterBandwidth	REAL	GSV/SSV	G11/Configuration Controls the bandwidth of the servo's/ drive's low-pass digital output filter. If set to zero (the default), the programmable low-pass output filter is bypassed. The filter can be used to filter out or reduce high frequency variation of the servo module output to the drive. The lo-pass filter adds lag to the servo loop pushing the system toward instability. Decreasing the OutputLPFilterBandwidth requires lowering the Position or Velocity ProportionalGain to maintain stability.
OutputNotchFilterFrequency	REAL	GSV/SSV	G10/Configuration Controls the center frequency of the drive's digital notch filter. It is bypassed if it is configured to the default of zero. It is particularly useful on attenuating mechanical resonance phenomena.
OutputOffset	REAL	GSV/SSV	G9/Configuration The fixed value used to offset the drift caused by the cumulative drive offsets of the servo module DAC output and the servo drive input. This value added is to the Servo Output.
PositionCommand	REAL	GSV	G11/Status Within the Servo Loop, this attribute is used to control the position of the axis.
PositionDataScaling	INT	GSV	G10/Configuration This advanced scaling attribute maps directly to the SERCOS IDNs. It is automatically configured to the appropriate default. It is read-only
PositionDataScalingExp	INT	GSV	G10/Configuration This advanced scaling attribute maps directly to the SERCOS IDNs. It is automatically configured to the appropriate default. It is read-only.
PositionDataScalingFactor	DINT	GSV	G10/Configuration This advanced scaling attribute maps directly to the SERCOS IDNs. It is automatically configured to the appropriate default. It is read-only.
PositionError	REAL	GSV	G11/Status The difference between the actual and command position of a servo axis. You can use this in conjunction with other error terms, to drive the motor to where the actual position equals the command position in an active servo loop.
PositionErrorFaultAction	SINT	GSV/SSV	G11/Configuration Sets the type of operation performed when a position error fault occurs. **Value: Meaning:** 0 = shutdown – most severe action to a fault and is reserved for faults that could endanger machinery or personnel. 1 = disabled drive – when the fault occurs the drive is immediately disabled, the servo output zeroed, and drive enable is output is deactivated 2 = stop command – when fault occurs the axis immediately begins decelerating the axis command position to a stop at the configured Maximum Deceleration Rate without disabling servo action or the servo module's Drive Enable Output. This is the gentlest stopping action for a fault. 3 = status only – when set to status only, motion faults must be handled by the application program
PositionErrorTolerance	REAL	GSV/SSV	G11/Configuration Specifies the amount of position error that the servo/drive tolerates before issuing a position error fault. It is interpreted as a ± quantity. A value of 0.75 position units means that an error is generated when the position error is greater than 0.75 or less than -0.75.
PositionFeedback	REAL	GSV	G11/Status Represents the current position of the axis within the servo loop.

PositionIntegralGain	REAL	GSV/SSV	G11/Configuration Improves the steady-state performance of the system. It achieves accurate axis positioning despite disturbances such as static friction and gravity. Increasing the integral gain increases the ultimate positioning accuracy of the system. Increase it too much though and you get system instability.
PositionIntegratorError	REAL	GSV	G11/Status The running sum of the Position Error for the specified axis. You can use this in conjunction with other error terms, to drive the motor to where the actual position equals the command position in an active servo loop.
PositionLockTolerance	REAL	GSV/SSV	G11/Configuration Specifies how much position error that the servo/SERCOS module tolerates when giving a true position locked status indication. Used in conjunction with the PositionLockedStatus bit, it is useful in controlling position accuracy. It is interpreted as a ± quantity. A value of 0.01 inches means that an error is generated when the position error is greater than 0.01 or less than -0.01
PositionPolarity	INT	GSV	G10/Configuration Maps directly to the SERCOS IDN. It is automatically set based on the current Drive Polarity Settings. All command bits are set according to the Command polarity bit value and all feedback bits are set according to the Feedback Polarity bit setting.
PositionProportionalGain	REAL	GSV/SSV	G11/Configuration This value is multiplied by the position error to create a component to the VelocityCommand that attempts to correct for the position error. Increasing the gain value increases the bandwidth of the position servo loop and results in greater static stiffness of the axis which is a measure of the corrective force applied to an axis for a given position error.
PositionServoBandwidth	REAL	GSV/SSV	G11/Configuration This value represents the unity gain bandwidth that is used to calculate the gains for a subsequent MAAT instruction. Within the constraints of a stable servo system, the higher the PositionServoBandwidth the better the dynamic performance of the system. A maximum value for this is generated by the MRAT instruction. Computing the gains based on this maximum value by way of the MAAT instruction results in dynamic response in keeping with the current value of the DampingFactor.
PositionUnits	STRING		The Axis Object lets you choose user-defined engineering units rather than feedback counts for measuring and programming all motion related values. It supports an ASCII text string of up to 32 characters. The string is used by the Logix Designer application in the axis configuration dialogs.
PositionUnwind	DINT	GSV/SSV	G1/Configuration Used to perform the automatic unwind of the rotary axis. Electronic unwind allows infinite position range for rotary axes by subtracting the unwind value from both the actual and command position every time the axis makes a complete revolution. The unwind value is in units feedback counts per axis revolution and must be an integer.
PositiveDynamicTorqueLimit	REAL	GSV	G10/Status Represents the maximum positive torque/current limit magnitude. It should be the lowest value of all torque/current limits in the drive at a given time. It includes the amplifier peak limit, motor peak limit, user current limit, amplifier thermal unit, and the motor thermal limit.
PowerCapacity	REAL	GSV	G10/Status Displays the present utilization of the axis power supply as a percent of rated capacity.

PowerSupplyID	INT	GSV	G10/Configuration Contains the enumeration of the specific A-B Power Supply or System Module catalog numbers associated with the axis. If it does not match, an error is generated.
PrimaryOperationMode	INT	GSV	G10/Configuration Maps directly to the SERCOS IDN. It is automatically set based on the current ServoLoopConfiguration and is Read-only.
ProgrammedStopMode	SINT	GSV/SSV	G8/Configuration Determines how a specific axis is to stop when the ControlLogix processor has a critical processor mode change or when an explicit MGS instruction is executed with its stop mode set to programmed. **Value: Meaning:** 0 = **Fast Stop** (default) – the axis decelerates to a stop using the configured value for Maximum Deceleration. Servo action is maintained after the axis motion has stopped. 1 = **Fast Disable** – the axis decelerates to a stop using the configured value for Maximum Deceleration. Servo action is maintained after the axis motion has stopped at which time the axis is disabled, that is, DriveEnable disabled and Servo Action disabled. 2 = **Hard Disable** – the axis is immediately disabled, that is, Drive Enable disabled, Servo Action disabled, but the OK contact is left closed. Unless the drive is configured to provide some form of dynamic braking, the axis coasts to a stop. 3 = **Fast Shutdown** – the axis decelerates to a stop using the configured value for Maximum Deceleration. Once the axis motion is stopped the axis is placed in a shutdown state, such as, Drive Enable disabled, servo action disabled, and the OK contact open. To recover from the shutdown state you must execute either an MASR or MGSR instruction. 4 = **Hard Shutdown** – The axis is immediately placed in a shutdown state such as, Drive Enable disabled, servo action disabled, and the OK contact open. Unless the drive is configured to provide some form of dynamic braking, the axis coasts to a stop. To recover from the shutdown state you must execute either an MASR or MGSR instruction.
Registration1Position	REAL	GSV	G1/Status The absolute position of a physical or virtual axis at the occurrence of the most recent registration event for that axis. You can use the following equation to determine the maximum registration position error based on your axis speed: MaximumSpeed(PositionUnits/Seconds) = (Accuracy(PositionUnits))/Delay
Registration1Time	DINT	GSV	G1/Status Contains the lower 32 bits of CST time at which the registration event occurred.
Registration2Position	REAL	GSV	G1/Status The absolute position of a physical or virtual axis at the occurrence of the most recent registration event for that axis. You can use the following equation to determine the maximum registration position error based on your axis speed: MaximumSpeed(PositionUnits/Seconds) = (Accuracy(PositionUnits))/Delay
Registration2Time	DINT	GSV	G1/Status Contains the lower 32 bits of CST time at which the registration event occurred.

<table>
<tr><td>RotaryAxis</td><td>SINT</td><td>GSV/SSV</td><td colspan="3">G1/Configuration
Sets when the axis is Linear or Rotary. When set to true (1), it enables the rotary unwind capability of the axis providing limitless position range. When set to false (0), it indicates linear operation with a maximum total linear position range is 1 billion feedback counts before it rolls over to zero.
Value: Meaning:
0 Linear
1 Rotary</td></tr>
<tr><td>RotationalPosResolution</td><td>DINT</td><td>GSV</td><td colspan="3">G10/Configuration
This advanced scaling attribute maps directly to the SERCOS IDNs. It is automatically configured to the appropriate default. It is read-only.</td></tr>
<tr><td>SercosErrorCode</td><td>INT</td><td>GSV</td><td colspan="3">G10/Status
Used to identify the source of the drive parameter update failure that results in the AxisConfigurationFault. The error codes are derived from the IEC-1394 SERCOS Interface standard.</td></tr>
<tr><td rowspan="5">ServoFaultBits</td><td rowspan="5">DINT</td><td rowspan="5">GSV</td><td colspan="3">G11/Status</td></tr>
<tr><td>Bit</td><td>Bit Name</td><td>Meaning</td></tr>
<tr><td>0</td><td>PosSoftOvertravelFault</td><td>Sets when the axis travels or attempts to travel beyond the configured value for Maximum Positive Travel.</td></tr>
<tr><td>1</td><td>NegSoftOvertravelFault</td><td>Sets when the axis travels or attempts to travel beyond the configured value for Maximum Negative Travel.</td></tr>
<tr><td>2</td><td>PosHardOvertravelFault</td><td>Not Available. Sets when the axis travels or attempts to travel beyond the maximum position limit set by the hardware limit switch mounted to the machine.</td></tr>
<tr><td>3</td><td>NegHardOvertravelFault</td><td>Not Available. Sets when the axis travels or attempts to travel beyond the minimum position limit set by the hardware limit switch mounted to the machine.</td></tr>
</table>

			4	FeedbackFault	Sets when the differential electrical signal for one or more of the feedback channels are at same level or loss of feedback power or common electrical connection between servo module or drive and the feedback device.
			5	FeedbackNoiseFault	Sets when simultaneous transitions of feedback of A and B are detected by the servo module.
			6	AuxFeedbackFault	Not Available. Sets when the differential electrical signal for one or more of the feedback channels are at same level or loss of feedback power or common electrical connection between servo module or drive and the auxiliary feedback device.
			7	AuxFeedbackNoiseFault	Not Available. Sets when simultaneous transitions of feedback of A and B are detected by the servo module.
			8	PositionErrorFault	Sets when the axis position error exceeds the value set for Position Error Tolerance.
			9	DriveFault	Sets when the external servo drive detects a fault and reports it to the servo module via the Drive Fault input,
			10-31	Reserved	

<table>
<tr><td>ServoLoopConfiguration</td><td>INT</td><td>GSV/SSV</td><td colspan="3">G11/Configuration
Determines the specific configuration of the servo loop topology when the AxisType is set to servo. When AxisType is set to feedback only, it is used to select the feedback port. It also established several advanced drive configuration attributes that are part of the SERCOS interface standard.
Value: Meaning:
0 custom
1 feedback only
2 aux. feedback only
3 position servo
4 aux. position servo
5 dual position servo
6 dual command servo
7 aux. dual command servo
8 velocity servo
9 torque servo</td></tr>
<tr><td>ServoOutputLevel</td><td>REAL</td><td>GSV</td><td colspan="3">G9/Status
The current voltage level of the servo output of the specified axis. This attribute can be used in a drilling application to detect when the drill bit has engaged the surface of the work piece.</td></tr>
<tr><td rowspan="4">ServoPolarityBits</td><td rowspan="4">DINT</td><td rowspan="4">GSV/SSV</td><td colspan="3">G9/Configuration</td></tr>
<tr><td>Bit</td><td>Bit Name</td><td>Meaning</td></tr>
<tr><td>0</td><td>FeedbackPolarityNegative</td><td>This controls the polarity of the encoder feedback and ensures that when the axis moves in the user defined positive direction that the Actual Position value increases. This bit is configured automatically using the MRHD and MAHD instructions.</td></tr>
<tr><td>1</td><td>ServoPolarityNegative</td><td>This controls the polarity servo output to the drive. It ensures that the axis servo loop is closed as a negative feedback system and not an unstable positive feedback system. It is configured automatically using the MRHD and MAHD instructions</td></tr>
<tr><td></td><td></td><td></td><td>2-31</td><td>Reserved</td><td></td></tr>
<tr><td rowspan="4">ServoStatusBits</td><td rowspan="4">DINT</td><td rowspan="4">GSV</td><td colspan="3">G9/Status
The status bits for your servo loop</td></tr>
<tr><td>Bit</td><td>Bit Name</td><td>Meaning</td></tr>
<tr><td>0</td><td>ServoActionStatus</td><td>Set when servo action is enabled.</td></tr>
<tr><td>1</td><td>DriveEnableStatus</td><td>Set when drive is enabled.</td></tr>
</table>

			2	AxisShutdownStatus	Set when axis is in shutdown state.
			3	ProcessStatus	Set when axis tuning or hookup diagnostic operation is in progress.
			4	OutputLimitStatus	Set when magnitude of the output has reached or exceeded configured output limit value.
			5	PositionLockStatus	Set when magnitude of axis position error has become less than or equal to the Position Lock Tolerance value of the axis.
			6	HomeInputStatus	Set when Home Input is active.
			7	RegistrationInputStatus	Set when registration 1 input is active.
			8	Registration2InputStatus	Not available.
			9	PositiveOvertravelInputStatus	Not available.
			10	NegativeOvertravelInputStatus	Not available.
			11-15	Reserved	
			16-31	Reserved	
SoftOvertravelFaultAction	SINT	GSV/SSV	G11/Configuration Sets the type of action performed when a soft overtravel fault occurs. **Value/Meaning:** 0 = shutdown – most severe action to a fault and is reserved for faults that could endanger machinery or personnel. 1 = disabled drive – when the fault occurs the drive is immediately disabled, the servo output zeroed, and drive enable output is deactivated 2 = stop command – when fault occurs the axis immediately begins decelerating the axis command position to a stop at the configured Maximum Deceleration Rate without disabling servo action or the servo module's Drive Enable Output. This is the gentlest stopping action for a fault. 3 = status only – when set to status only, motion faults must be handled by the application program.		
StartActualPosition	REAL	GSV	G1/Status Stores the actual position of your axis at precisely the instant motion begins when a new motion planner instruction starts for the axis. You can use this value to correct for any motion occurring between the detection of an event and the action initiated by the event.		
StartCommandPosition	REAL	GSV	G4/Status Stores the command position of your axis at precisely the instant motion begins when a new motion planner instruction starts for the axis. You can use this value to correct for any motion occurring between the detection of an event and the action initiated by the event.		
StartMasterOffset	REAL	GSV	G4/Status the position offset that is applied to the master side of the position cam when the last MAM instruction with the move type set to either Absolute Master Offset or Incremental Master Offset was executed. It shows the same unwind characteristic as the position of a linear axis.		

StoppingTimeLimit	REAL	GSV/SSV	G10/Configuration Maps directly to the SERCOS IDNs and is automatically configured based on the current Drive Configuration.
StoppingTorque	REAL	GSV/SSV	G10/Configuration Maps directly to the SERCOS IDNs and is automatically configured based on the current Drive Configuration.
StrobeActualPosition	REAL	GSV	G1/Status Stores a snapshot of the actual position of an axis when the MGSP instruction executes. Because it stores the strobe positions for all axes in a specified group the values for different axes can be used to perform real time calculations.
StrobeCommandPosition	REAL	GSV	G4/Status Stores a snapshot of the commanded position of an axis when the MGSP instruction executes. Because it stores the strobe positions for all axes in a specified group the values for different axes can be used to perform real time calculations.
StrobeMasterOffset	REAL	GSV	G4/Status The position offset that is applied to the master side of the position cam when the last MGSP instruction was executed. It shows the same unwind characteristic as the position of a linear axis.
TelegramType	INT	GSV	G10/Configuration Maps directly to the SERCOS IDN. It is automatically set based on the current ServoLoopConfiguration and is Read-only.
TestDirectionForward	SINT	GSV	G11/Status Reports the direction of axis travel during hookup test as seen by the servo module/SERCOS drive when the MRHD instruction is run. **Value: Meaning:** 0 negative (reverse) direction 1 True - positive (forward) direction
TestIncrement	REAL	GSV/SSV	G11/Configuration A commissioning configuration attribute used with the MRHD instruction to determine the amount of motion needed to satisfy the test process. Because this value is used by the MRHD instruction it must be set before the instruction is run.
TestStatus	INT	GSV	G11/Status Provides information on the last run MRHD instruction. It reports the status of the instruction using the following values: **Value: Meaning:** 0 test process successful 1 test in progress 2 test process aborted by the user 3 test exceeded 2-second time-out 4 test failed – servo fault 5 test failed – insufficient test increment (Servo Drive) 6 test failed – wrong polarity (Servo Drive) 7 test failed – missing signal (Servo Drive) 8 test failed – device comm error (Servo Drive) 9 test failed – feedback config error (Servo Drive) 10 test failed – motor wiring error (Servo Drive)
TorqueCommand	REAL	GSV	G10/Status The command value when operating in torque mode.
TorqueDataScalling	INT	GSV	G10/Configuration This advanced scaling attribute maps directly to the SERCOS IDNs. It is automatically configured to the appropriate default. It is read-only.

TorqueDataScalingFactor	INT	GSV	G10/Configuration This advanced scaling attribute maps directly to the SERCOS IDNs. It is automatically configured to the appropriate default. It is read-only.
TorqueFeedback	DINT	GSV	G10/Status The torque feedback value when operating in torque mode.
TorqueLimitBipolar	REAL	GSV	G10/Configuration Maps directly to the SERCOS IDNs and is automatically configured to reasonable default values. Manipulation is not needed unless motivated by a specific application requirement.
TorqueLimitNegative	REAL	GSV/SSV	G10/Configuration Maps directly to the SERCOS IDNs and is automatically configured to reasonable default values. Manipulation is not needed unless motivated by a specific application requirement.
TorqueLimitPositive	REAL	GSV/SSV	G10/Configuration Maps directly to the SERCOS IDNs and is automatically configured to reasonable default values. Manipulation is not needed unless motivated by a specific application requirement.
TorqueLimitSource	DINT	GSV	G10/Status It displays the present source (if any) of any torque limiting the axis. The values are: **Value: Meaning:** 0 Not Limited 1 Neg. Torque Limit 2 Pos. Torque Limit 3 Amp Peak Limit 4 Amp I(t) Limit 5 Bus Regulator Limit 6 Bipolar Torque Limit 7 Motor Peak Limit 8 Motor I(t) Limit 9 Voltage Limit
TorqueOffset	REAL	GSV/SSV	G11/Configuration It is used to provide a dynamic torque command correction to the output of the velocity servo loop. It can be tied into custom outer control loop algorithms using Function Block programming.
TorquePolarity	INT	GSV	G10/Configuration It maps directly to the SERCOS IDN. It is automatically set based on the current Drive Polarity Settings. All command bits are set according to the Command polarity bit value and all feedback bits are set according to the Feedback Polarity bit setting.
TorqueScaling	REAL	GSV/SSV	G11/Configuration Used to convert the acceleration of the servo loop into equivalent % rated torque to the motor. It has the effect of normalizing the units of the servo loop gain parameters so their values are not affected by variations in feedback resolution, drive scaling, motor and load inertia, and mechanical gear ratios. The TorqueScaling value represents the inertia of the system and is related to the TuneInertia attribute value by a factor of the Conversion Constant. It is normally established by the MAAT instruction as part of the controller's automatic tuning procedure, but it can be manually calculated: TorqueScaling = 100 % Rated Torque / (Acceleration @ 100 % Rated Torque)
Torque Threshold	REAL	GSV/SSV	G10/Configuration Maps directly to the SERCOS IDNs and is automatically configured to reasonable default values. Manipulation is not needed unless motivated by a specific application requirement.

<table>
<tr><td>TuneAcceleration</td><td>REAL</td><td>GSV</td><td>G11/Status
Returns the measured acceleration value from the last run MRAT instruction. This value is used to calculate the Tune Inertia value of the axis and are used by a subsequent MAAT instruction for the tune value for the MaximumAcceleration attribute.</td></tr>
<tr><td>TuneAccelerationTime</td><td>REAL</td><td>GSV</td><td>G11/Status
The acceleration time in seconds measured during the last run MRAT instruction. This value is used in calculating the TuneAcceleration attribute.</td></tr>
<tr><td>TuneDeceleration</td><td>REAL</td><td>GSV</td><td>G11/Status
Returns the measured deceleration value from the last run MRAT instruction. This value is used to calculate the Tune Inertia value of the axis and are used by a subsequent MAAT instruction for the tune value for the MaximumDeceleration attribute.</td></tr>
<tr><td>TuneDecelerationTime</td><td>REAL</td><td>GSV</td><td>G11/Status
The deceleration time in seconds measured during the last run MRAT instruction. This value is used in calculating the TuneDeceleration attribute.</td></tr>
<tr><td>TuneInertia</td><td>REAL</td><td>GSV</td><td>G11/Status
The TuneInertia value represents the total inertia of the axis as calculated from the measurements made during the last MRAT instruction. The value is shown in terms of percent of full-scale servo output per MegaCounts/Sec2 of feedback input.</td></tr>
<tr><td>TuneRiseTime</td><td>REAL</td><td>GSV</td><td>G9/Status
The axis rise time in seconds measured during the last run MRAT instruction. This value only applies to axes that you configure to work with an external velocity servo drive. This value is used to calculate the TuneVelocityBandwidth.</td></tr>
<tr><td>TuneSpeedScaling</td><td>REAL</td><td>GSV</td><td>G9/Status
The axis drive scaling factor measured during the last executed MRAT instruction. It applies only to axes configured to interface with an external velocity servo drive. This value is applied to the VelocityScaling attribute by a subsequent MAAT instruction.</td></tr>
<tr><td>TuneStatus</td><td>INT</td><td>GSV</td><td>G11/Status
Returns the status of the last run MRAT instruction that initiates a tuning process on the targeted servo/SERCOS module axis. The following values apply to the status of the MRAT instruction:
Value: Meaning:
0 tune process successful
1 tuning in progress
2 tune process aborted by user
3 tune exceeded 2-second time-out
4 tune process failed due to servo/drive fault
5 axis reached Tuning Travel Limit
6 axis polarity set incorrectly
7 tune measurement fault (ServoDrive)
7 tune configuration fault (ServoDrive)</td></tr>
<tr><td>TuningConfigurationBits</td><td>DINT</td><td>GSV/SSV</td><td>G11/Configuration
The tuning configuration bits for your axis
<table>
<tr><th>Bit</th><th>Bit Name</th><th>Meaning</th></tr>
<tr><td>0</td><td>TuningDirectionReverse</td><td>It determines the direction of the tuning motion profile initiated by the MRAT instruction. If set (true), goes in reverse or negative direction.</td></tr>
</table></td></tr>
</table>

			1	TunePositionErrorIntegrator	Determines whether or not the MAAT instruction calculates a Position Integral Gain value. If set to clear (false), Position Integral Gain is set to zero.
			2	TuneVelocityErrorIntegrator	Determines whether or not the MAAT instruction calculates a Velocity Integral Gain value. If set to clear (false), Velocity Integral Gain Gain is set to zero.
			3	TuneVelocityFeedforward	Determines whether or not the MAAT instruction calculates a Velocity Feedforward Gain value. If set to clear (false), Velocity Feedforward Gain is set to zero.
			4	AccelerationFeedforward	Determines whether or not the MAAT instruction calculates an Acceleration Feedforward Gain value. If set to clear (false), Acceleration Feedforward Gain is set to zero.
			5	TuneVelocityLow-PassFilter	Determines whether or not the MAAT instruction calculates an Output Filter Bandwidth value. If set to clear (false), Output Filter Bandwidth is set to zero disabling the filter.
			6	Reserved	
TuningSpeed			G11/Configuration Determines the maximum speed of the MRAT instruction initiated tuning motion profile. It should be set to the desired maximum operating speed of the motor prior to running the MRAT instruction.		
TuningTorque			G11/Configuration Determines the maximum torque of the MRAT initiated tuning motion profile. Set it to the desired maximum safe torque level prior to running the MRAT instruction to ensure the most accurate measure of the acceleration and deceleration capabilities of the system.		
TuningTravelLimit			G11/Configuration Used by the MRAT instruction to limit the travel of the axis during tuning.		

VelocityCommand			G11/Status The current velocity reference to the velocity servo loop for a specified axis. It represents the output of the outer position control loop.
VelocityDataScaling			G10/Configuration This advanced scaling attribute maps directly to the SERCOS IDNs. It is automatically configured to the appropriate default. It is read-only.
VelocityDataScalingExp			G10/Configuration This advanced scaling attribute maps directly to the SERCOS IDNs. It is automatically configured to the appropriate default. It is read-only.
VelocityDataScalingFactor			G10/Configuration his advanced scaling attribute maps directly to the SERCOS IDNs. It is automatically configured to the appropriate default. It is read-only.
VelocityDroopr			G10/Configuration Maps directly to the SERCOS IDNs and is automatically configured to reasonable default values. Manipulation is not needed unless motivated by a specific application requirement.
VelocityError			G11/Status The difference between the commanded and actual velocity of a servo axis. You can use this value to drive the motor to where the velocity feedback equals the velocity command for an axis with an active velocity loop.
VelocityFeedback			G11/Status The actual velocity of your axis as estimated by the servo/SERCOS module. It is computed by applying a 1 KHz low-pass filter to the change in actual position over the servo update interval. VelocityFeedback is a signed value. The sign (+ or -) depends on the direction the axis is currently moving.
VelocityFeedforwardGain			G11/Configuration The value used to provide the velocity command output to generate the command velocity. The optimal value is 100%.
VelocityIntegralGain			G11/Configuration When configured for a torque loop servo drive, at every servo update the current Velocity Error is accumulated in a variable called Velocity integral Error. This value is multiplied by the VelocityIntegralGain to produce a component to the Servo Output or Torque Command that tries to correct the velocity error. The characteristic of the VelocityIntegralGain is that any non-zero Velocity Error accumulates in time to generate enough force to make the correction. VelocityIntegralGain makes it invaluable in applications where velocity accuracy is critical. The higher the value the faster the axis is driven to the zero Velocity Error condition.
VelocityIntegratorError			G11/Status The running sum of the Velocity Error for a specified axis. You can use this value to drive the motor to where the velocity feedback equals the velocity command for an axis with an active velocity loop.
VelocityLimitBipolar			G10/Configuration Maps directly to the SERCOS IDNs and is automatically configured to reasonable default values. Manipulation is not needed unless motivated by a specific application requirement.
VelocityLimitNegative			G10/Configuration Maps directly to the SERCOS IDNs and is automatically configured to reasonable default values. Manipulation is not needed unless motivated by a specific application requirement.
VelocityLimitPositive			G10/Configuration Maps directly to the SERCOS IDNs and is automatically configured to reasonable default values. Manipulation is not needed unless motivated by a specific application requirement.

VelocityOffset			G11/Configuration Used to provide a dynamic velocity correction to the output of the position servo loop. It can be tied into custom outer control loop algorithms using Function Block programming.
VelocityPolarity			G10/Configuration Maps directly to the SERCOS IDN. It is automatically set based on the current Drive Polarity Settings. All command bits are set according to the Command polarity bit value and all feedback bits are set according to the Feedback Polarity bit setting.
VelocityProportionalGain			G11/Configuration This value is multiplied by the velocity error to create a component to the Servo Output or Torque Command that attempts to correct for the velocity error, creating the damping effect. Increasing the gain value results in smoother motion, enhanced acceleration, reduced overshoot, and greater system stability. Too great of an increase can cause frequency instability and resonance effects.
VelocityScaling			G9/Configuration Used to convert the output of the servo loop into equivalent voltage to an external velocity servo drive. It has the effect of normalizing the units of the servo loop gain parameters so their values are not affected by variations in feedback resolution, drive scaling, or mechanical gear ratios. The VelocityScaling value is normally established by the servo's automatic tuning procedure but it can be manually calculated: VelocityScaling = 100 % / (Speed @ 100 %)
VelocityServoBandwidth			G11/Configuration Represents the unity gain bandwidth that is used to calculate the gains for subsequent MAAT instruction. Within the constraints of a stable servo system, the higher the VelocityServoBandwidth the better the dynamic performance of the system. A maximum value for this is generated by the MRAT instruction. Computing the gains based on this maximum value by way of the MAAT instruction results in dynamic response in keeping with the current value of the DampingFactor.
VelocityStandstillWindow			G10/Configuration Maps directly to the SERCOS IDNs and is automatically configured to reasonable default values. Manipulation is not needed unless motivated by a specific application requirement.
VelocityThreshold			G10/Configuration Maps directly to the SERCOS IDNs and is automatically configured to reasonable default values. Manipulation is not needed unless motivated by a specific application requirement.
VelocityWindow			G10/Configuration Maps directly to the SERCOS IDNs and is automatically configured to reasonable default values. Manipulation is not needed unless motivated by a specific application requirement.
WatchPosition			G1/Status The current set-point of an axis as set up in the last, most recently executed, MAW instruction for that axis.

Motion Status and Configuration Parameters

You can read motion status and configuration parameters in your ladder logic program using two methods.

Method	Example
Directly accessing the MOTION_GROUP and AXIS structures	Axis faults Motion status Servo status
Using the GSV instruction	Actual position Command position Actual velocity

Troubleshoot Axis Motion

Use this information to troubleshoot some situations that could happen while you are running an axis.

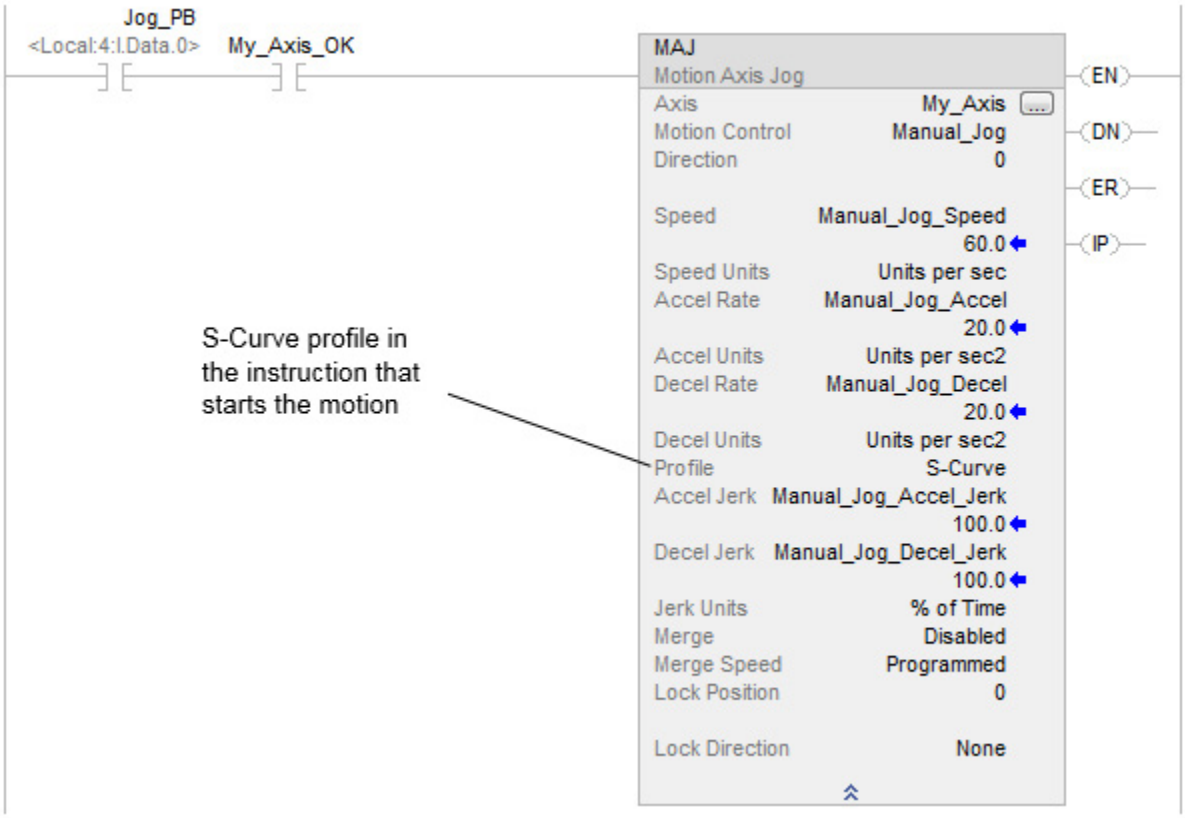
Why does my axis accelerate when I stop it?

In some instances, while an axis is accelerating, you try to stop it. The axis keeps accelerating for a short time before it starts to decelerate.

Example

You start a Motion Axis Jog (MAJ) instruction. Before the axis gets to its target speed, you start a Motion Axis Stop (MAS) instruction. The axis continues to speed up and then eventually slows to a stop.

Look For



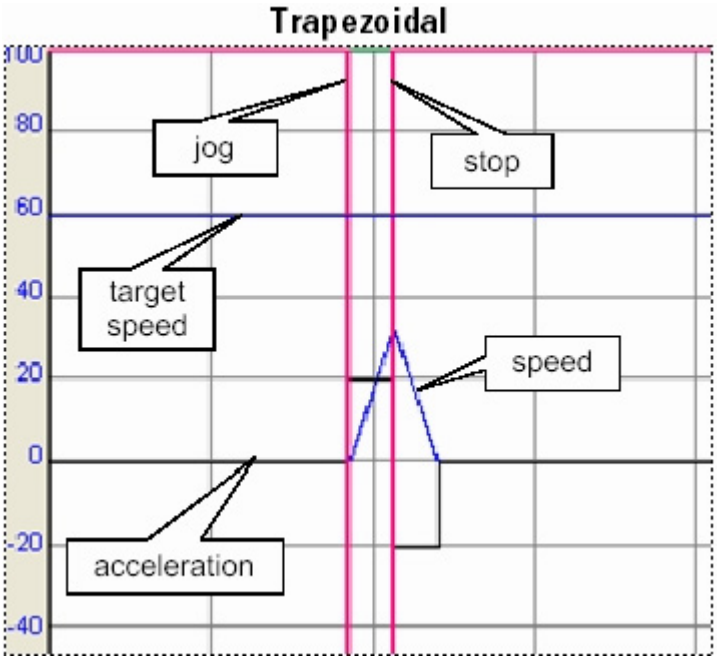
Cause

When you use an S-curve profile, jerk determines how fast an axis can change its acceleration and deceleration.

- An S-curve profile has to get acceleration to zero before the axis can slow down.
- The time it takes depends on the jerk, acceleration, and speed.
- In the meantime, the axis continues to speed up.

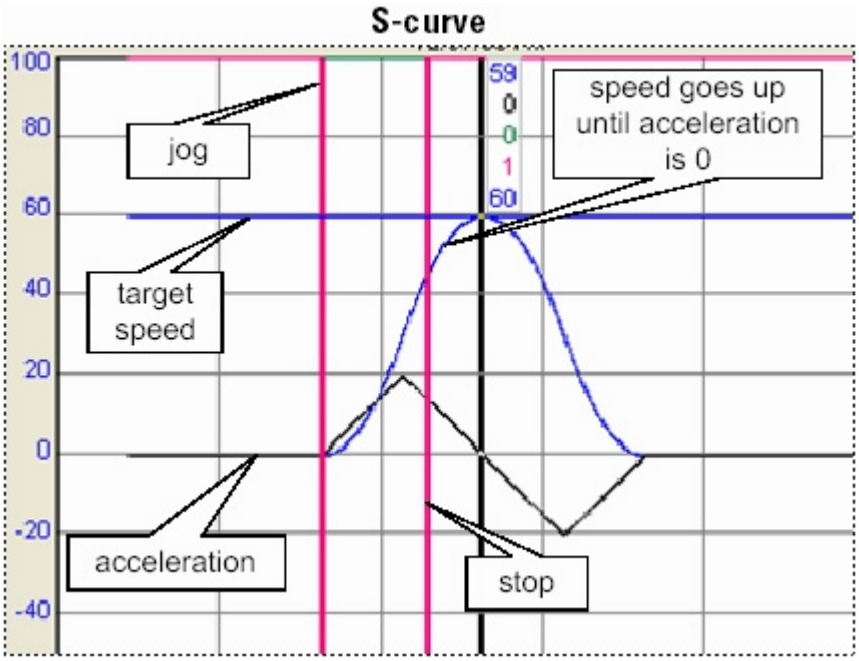
The following trends show how the axis stops with a trapezoidal profile and an S-curve profile.

Trapezoidal



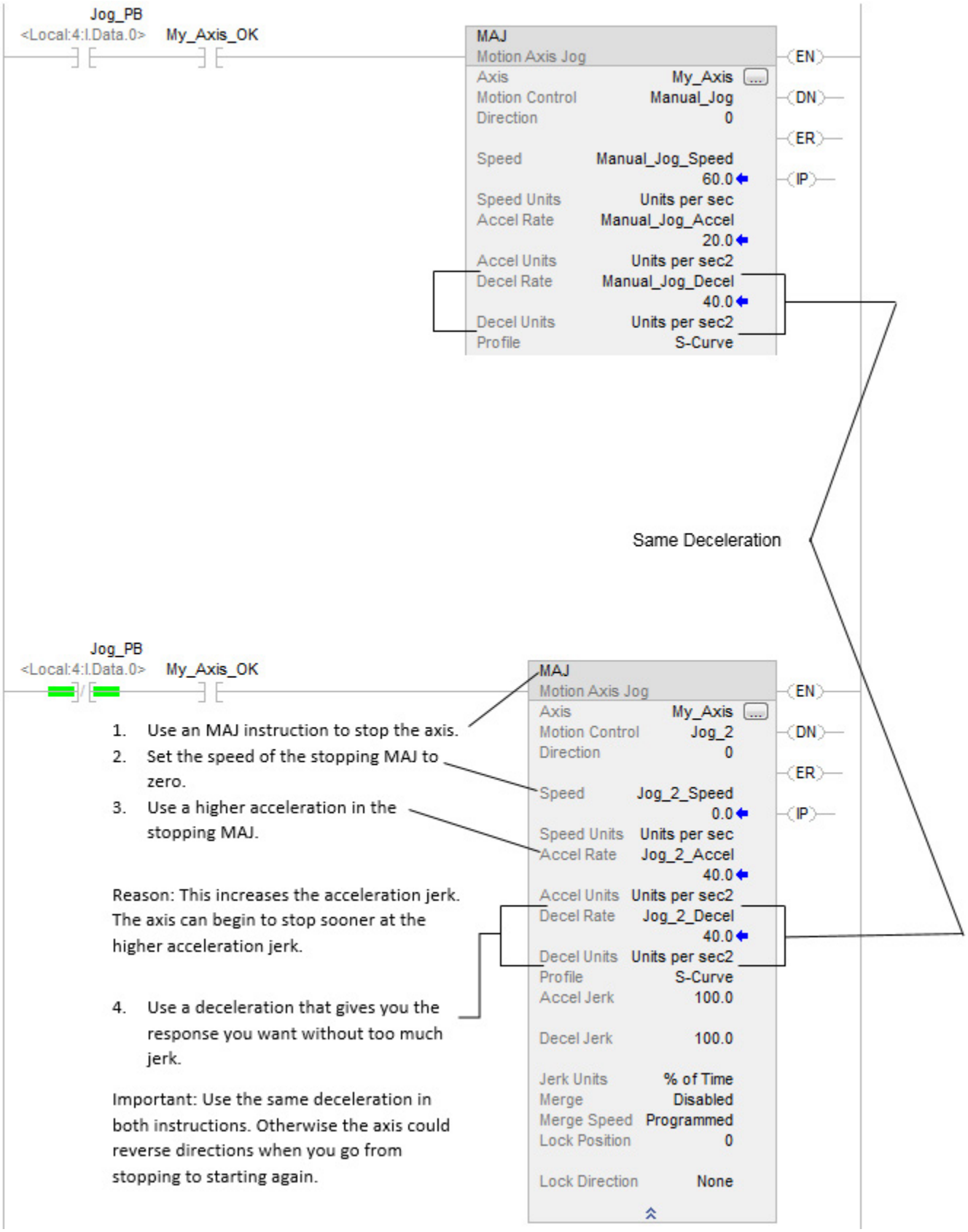
The axis slows down as soon as you start the stopping instruction.

S-curve



Corrective Action

Revision 15 or earlier



Revision 16 or later

Leave bit 0 of the DynamicsConfigurationBits attribute for the axis turned ON. Otherwise, this corrective action will not work.

Revision 16 and later lets you increase the deceleration jerk of an Motion Action Stop (MAS) instruction to get a quicker stop.

If the Jerk Units Are	Then Make This Change to the Decel Jerk
% of Time	Reduce the % of Time on the Decel Jerk
% of Maximum	Increase the % of Maximum on the Decel Jerk
Units per sec^3	Increase Units per sec^3 on the Decel Jerk

Why does my axis overshoot its target speed?

While an axis is accelerating, you try to stop the axis or change its speed. The axis keeps accelerating and goes past its initial target speed.

Eventually, it starts to decelerate.

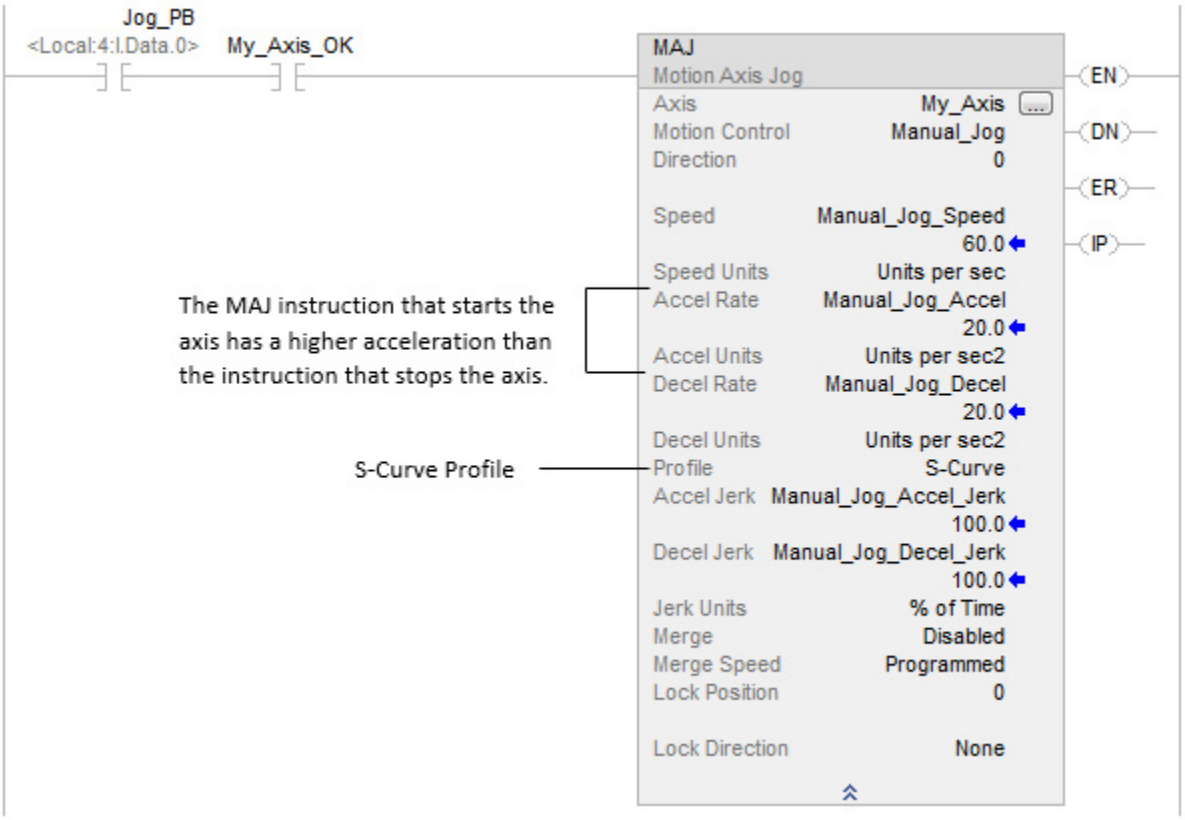
Important: Revision 16 improved how the controller handles changes to an S-curve profile. For more information.

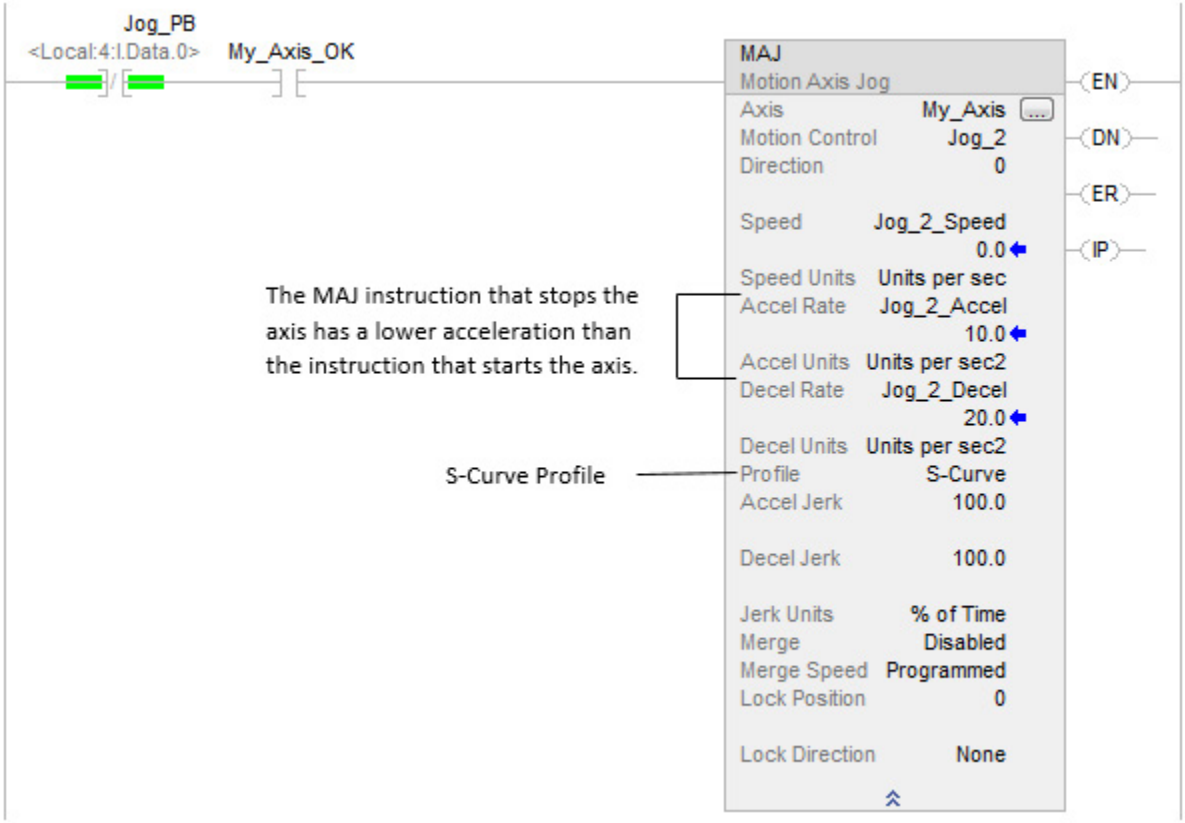
Example

You start a Motion Axis Jog (MAJ))instruction. Before the axis gets to its target speed, you try to stop it with another Motion Axis Jog (MAJ) instruction.

The speed of the second instruction is set to zero. The axis continues to speed up and overshoots its initial target speed. Eventually, it slows to a stop.

Look For


Jog_PB
<Local:4:I.Data.0>
My_Axis_OK
MAJ
Motion Axis Jog
Axis My_Axis
Motion Control Manual_Jog
Direction 0
Speed Manual_Jog_Speed
60.0
Speed Units Units per sec
Accel Rate Manual_Jog_Accel
20.0
Accel Units Units per sec2
Decel Rate Manual_Jog_Decel
20.0
Decel Units Units per sec2
Profile S-Curve
Accel Jerk Manual_Jog_Accel_Jerk
100.0
Decel Jerk Manual_Jog_Decel_Jerk
100.0
Jerk Units % of Time
Merge Disabled
Merge Speed Programmed
Lock Position 0
Lock Direction None
EN
DN
ER
IP
The MAJ instruction that starts the axis has a higher acceleration than the instruction that stops the axis.
S-Curve Profile

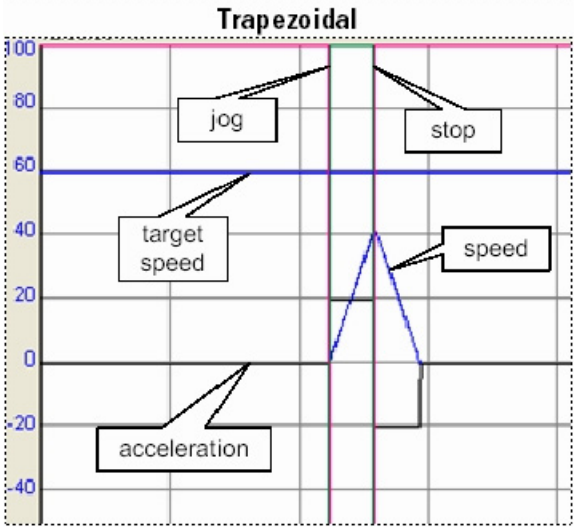

Jog_PB
<Local:4:I.Data.0>
My_Axis_OK
MAJ
Motion Axis Jog
Axis My_Axis
Motion Control Jog_2
Direction 0
Speed Jog_2_Speed
0.0
Speed Units Units per sec
Accel Rate Jog_2_Accel
10.0
Accel Units Units per sec2
Decel Rate Jog_2_Decel
20.0
Decel Units Units per sec2
Profile S-Curve
Accel Jerk 100.0
Decel Jerk 100.0
Jerk Units % of Time
Merge Disabled
Merge Speed Programmed
Lock Position 0
Lock Direction None
EN
DN
ER
IP
The MAJ instruction that stops the axis has a lower acceleration than the instruction that starts the axis.
S-Curve Profile

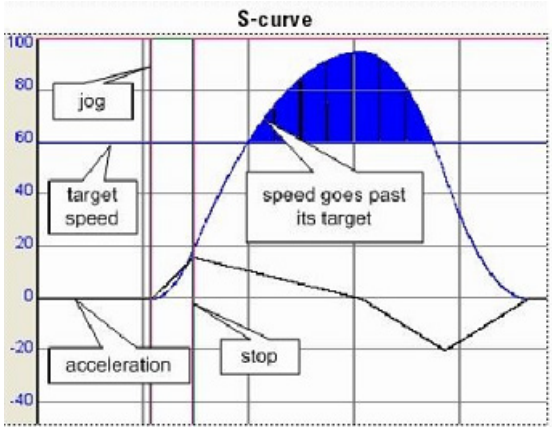
Cause

When you use an S-curve profile, jerk determines how fast an axis can change its acceleration and deceleration.

- When the stopping instruction starts, the controller recalculates jerk and builds a new S-curve profile.
- If the stopping instruction uses a lower acceleration, the jerk is lower. It takes longer at the lower jerk to get acceleration to zero.
- In the meantime, the axis continues past its initial target speed.

The following trends show how the axis stops with a trapezoidal profile and an S-curve profile.

<table>
<tr><th colspan="2">Stop while accelerating and reduce the acceleration rate</th></tr>
<tr><td>Trapezoidal
 

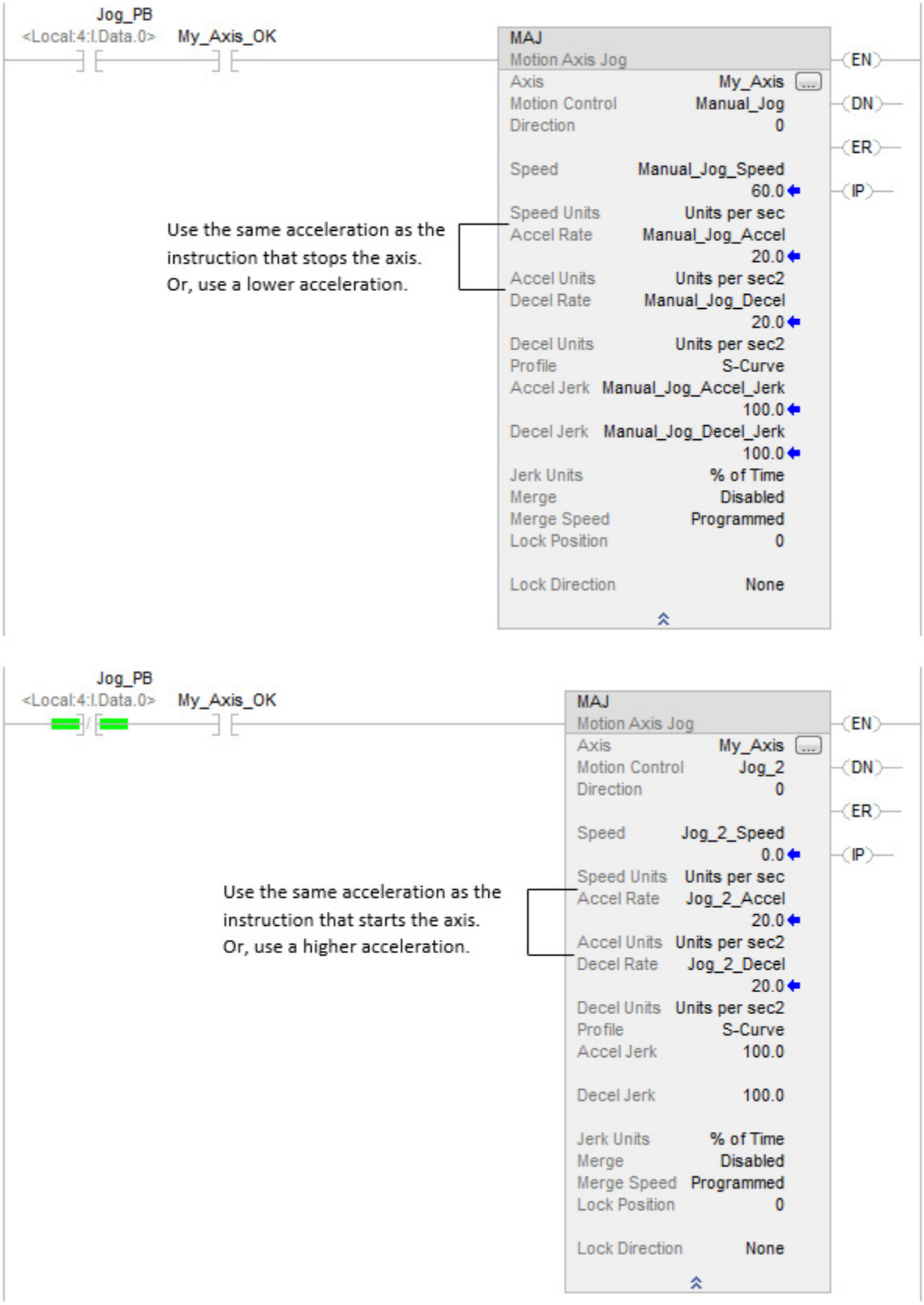
The axis slows down as soon as you start the stopping instruction. The lower acceleration does not change the response of the axis.</td><td>S-curve
 

The stopping instruction reduces the acceleration of the axis. It now takes longer to bring the acceleration to zero. The axis continues past its target speed until acceleration equals zero.</td></tr>
<tr><td>S-curve</td><td></td></tr>
</table>

Corrective Action

Use a Motion Axis Stop (MAS) instruction to stop the axis.

Or, set up your instructions like the following.



Why is there a delay when I stop and then restart a jog?

While an axis is jogging at its target speed, you stop the axis. Before the axis stops completely, you restart the jog. The axis continues to slow down before it speeds up.

Example

You use a Motion Axis Stop (MAS) instruction to stop a jog. While the axis is slowing down, you use a Motion Axis Jog (MAJ) instruction to start the axis again. The axis does not respond right away. It continues to slow down. Eventually, it speeds back up to the target speed.

Look For

For Stop Type, the instruction that stops the axis keeps the S-curve profile. Suppose you use an Motion Axis Stop (MAS) instruction with the Stop Type set to Jog. In that case, the axis keeps the profile of the Motion Axis Jog (MAJ) instruction that started the axis.

Cause

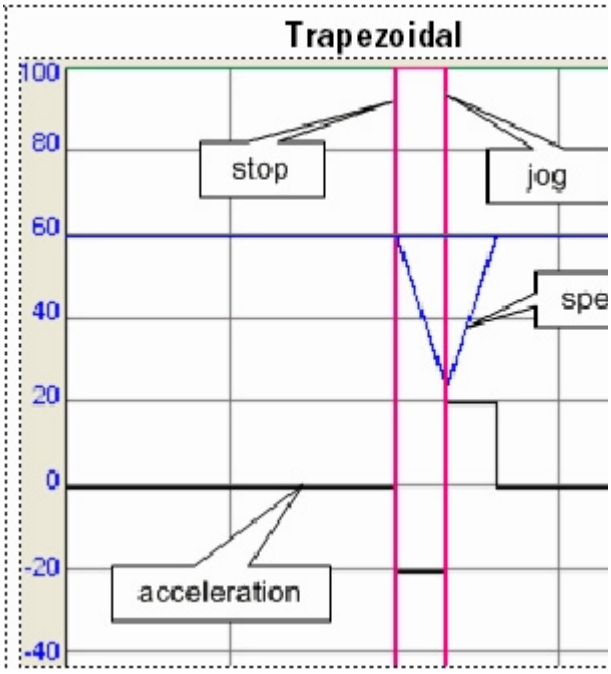
When you use an S-curve profile, jerk determines how fast an axis can change its acceleration and deceleration.

- An S-curve profile has to get acceleration to zero before the axis can speed up again.
- The axis continues to slow down until the S-curve profile brings the acceleration to zero.

The following trends show how the axis stops and starts with a trapezoidal profile and an S-curve profile.

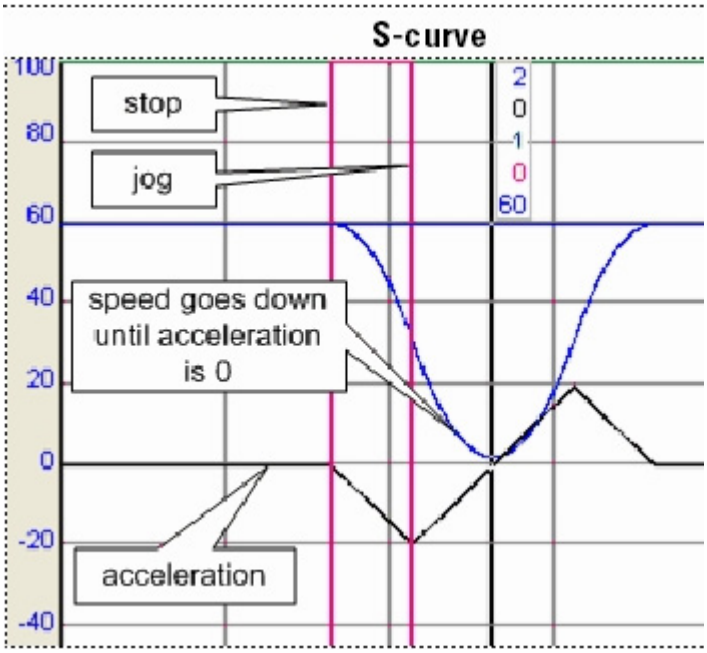
Stop while decelerating

Trapezoidal



The axis speeds back up as soon as you start the jog again.

S-curve



The axis continues to slow down until the S-curve profile brings the acceleration rate to zero.

Corrective Action

If your controller is this revision	Then	Result

15 or earlier	Increase the deceleration rate of the Motion Axis Jog (MAJ) instruction that starts the jog.	This increases the deceleration jerk. The axis stops the deceleration sooner at the higher deceleration jerk.
16 or later	Increase the deceleration jerk of the Motion Axis Jog (MAJ) instruction that starts the jog.	The axis stops the deceleration sooner at the higher deceleration jerk.

Why does my axis reverse direction when I stop and start it?

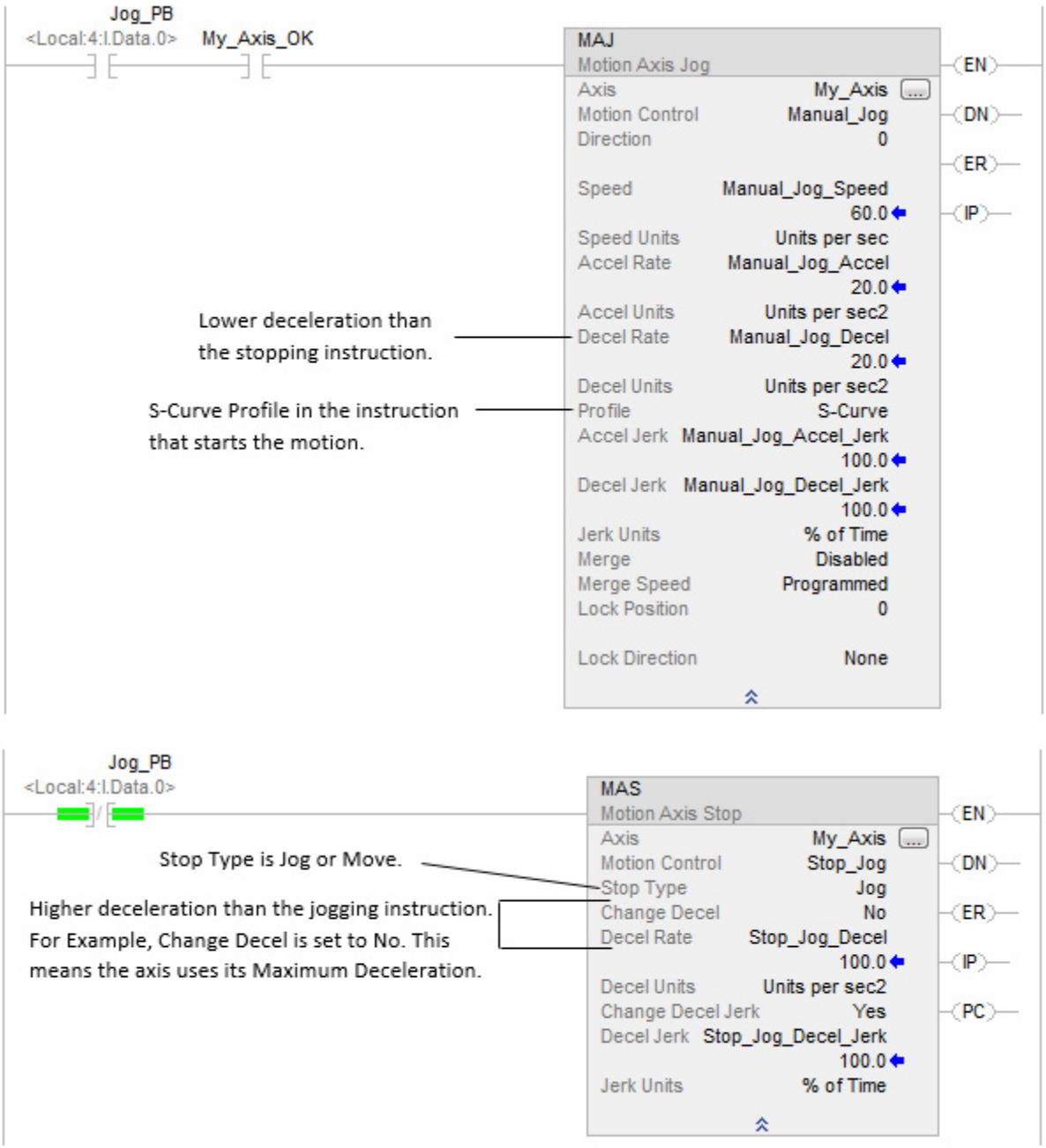
While an axis is jogging at its target speed, you stop the axis. Before the axis stops completely, you restart the jog. The axis continues to slow down and then reverse direction. Eventually, the axis changes direction again and moves in the programmed direction.

Important: You should not see this situation in revision 16 and later. See Corrective action for Revision 16 or later in this section.

Example

You use a Motion Axis Stop (MAS) instruction to stop a jog. While the axis is slowing down, you use a Motion Axis Jog (MAJ) instruction to start the axis again. The axis continues to slow down and then moves in the opposite direction. Eventually, the axis goes back to its programmed direction.

Look For



Cause

When you use an S-curve profile, jerk determines how fast an axis can change its acceleration and deceleration.

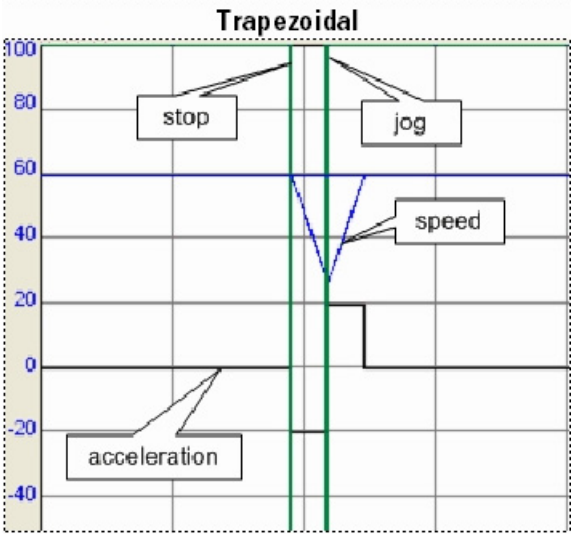
- When the Motion Axis Jog (MAJ) instruction starts again, the controller recalculates jerk and builds a new S-curve profile.
- If the Motion Axis Jog (MAJ) instruction uses a lower deceleration, the jerk is lower. It takes longer at the lower jerk to get deceleration to zero.

- In the meantime, the axis continues past zero speed and moves in the opposite direction.

The following trends show how the axis stops and starts with a trapezoidal profile and an S-curve profile.

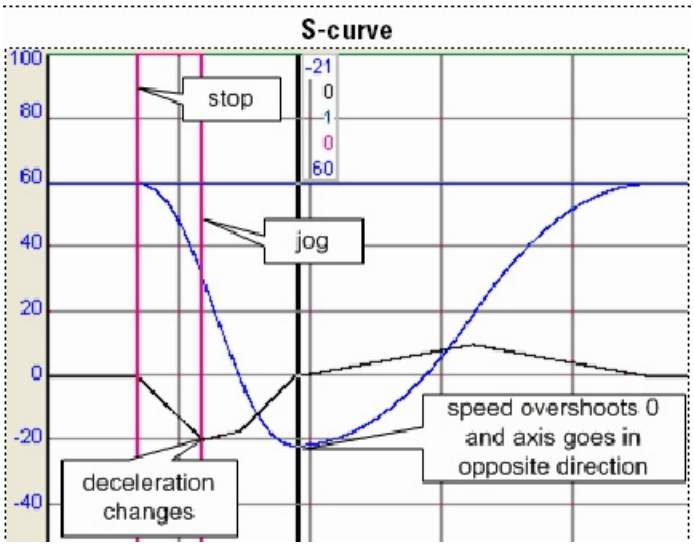
Stop while decelerating and reduce the deceleration rate

Trapezoidal



The axis speeds back up as soon as you start the jog again. The lower deceleration does not change the response of the axis.

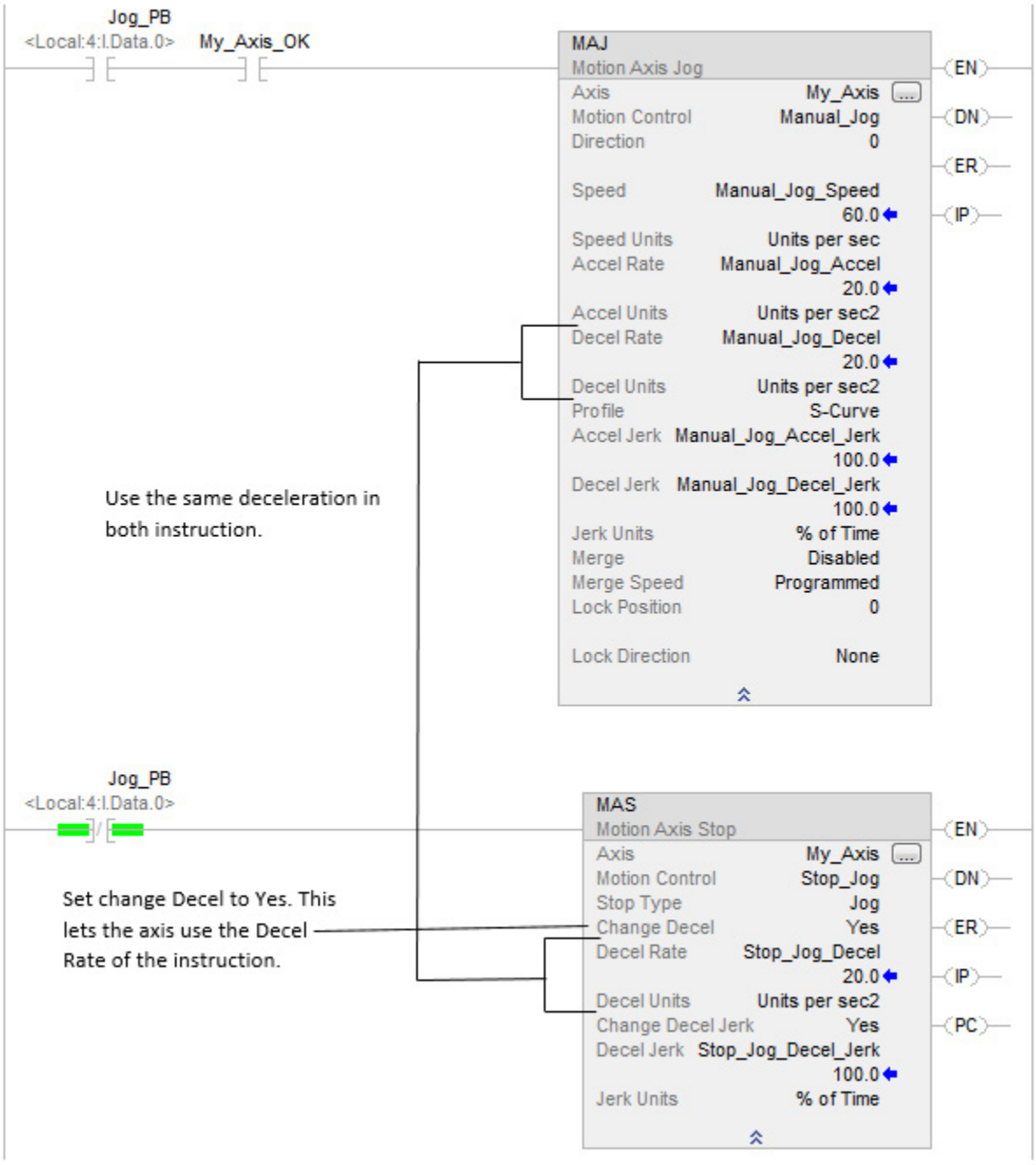
S-curve



The jog instruction reduces the deceleration of the axis. It now takes longer to bring the deceleration to zero. The speed overshoots zero and the axis moves in the opposite direction.

Corrective Action

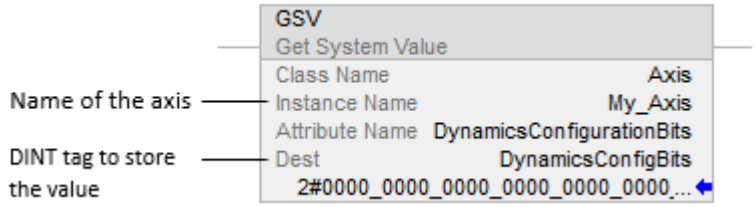
Revision 15 or earlier



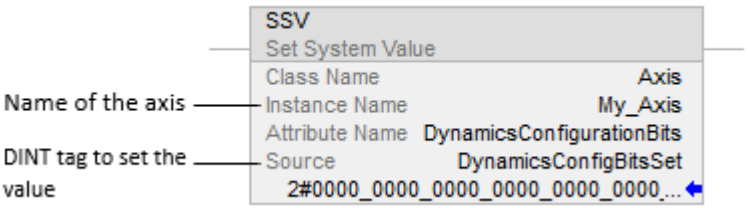
Revision 16 or later

Revision 16 improved how the controller handles changes to an S-curve profile. If you're still seeing an axis reversal, make sure bit 1 of the DynamicsConfigurationBits for the axis is on.

1. Use a Get System Value (GSV) instruction to see if the algorithm is on.



2. If bit 1 is off, turn it on.



For more information, use the following path: Help > Index > GSV/SSV Objects > Axis > Dynamics Configuration Bits

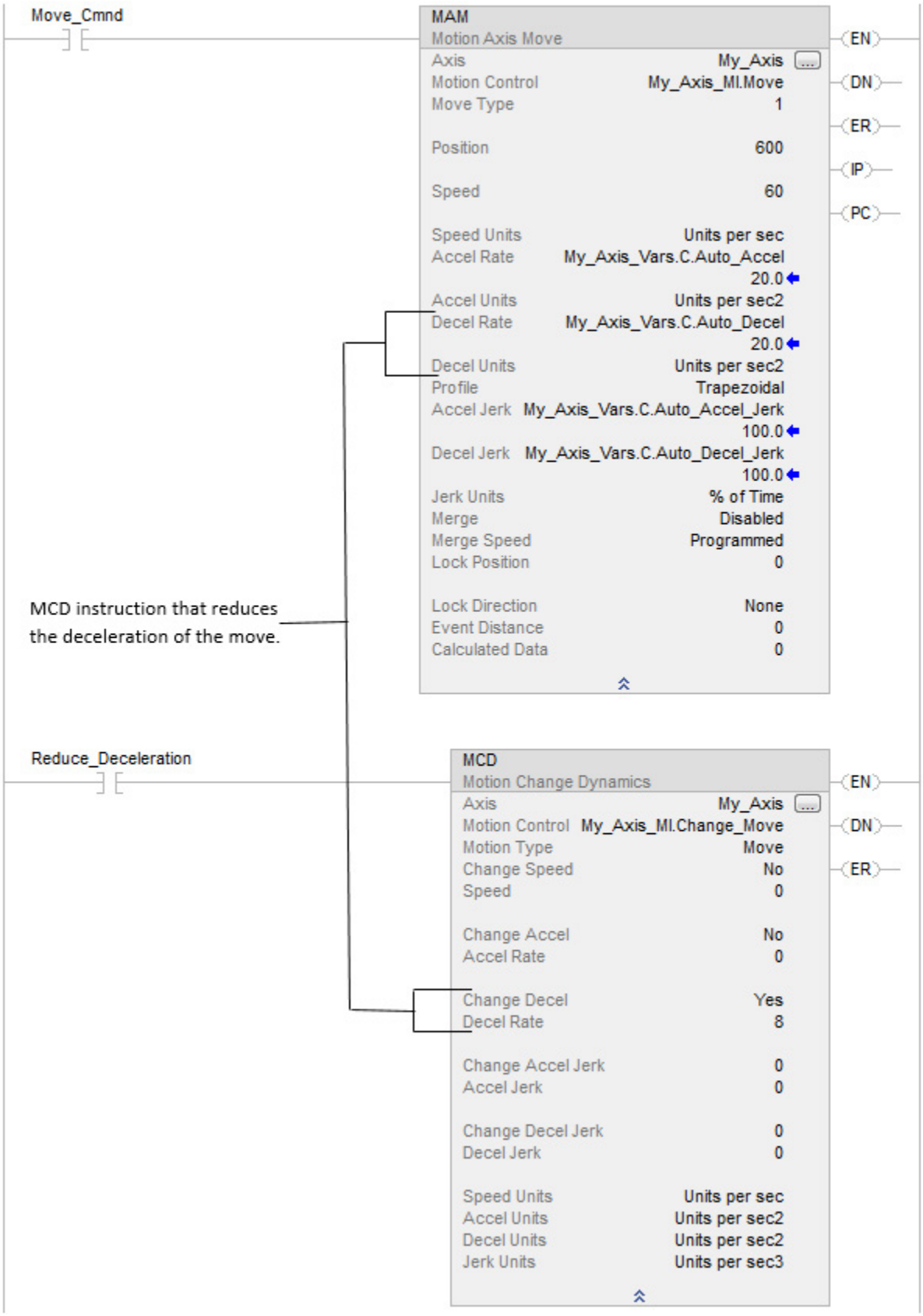
Why does my axis overshoot its position and reverse direction?

While an axis is moving to a target position, you change a parameter of the move. The axis overshoots its target position. Eventually the axis stops and moves back to its target position.

Example

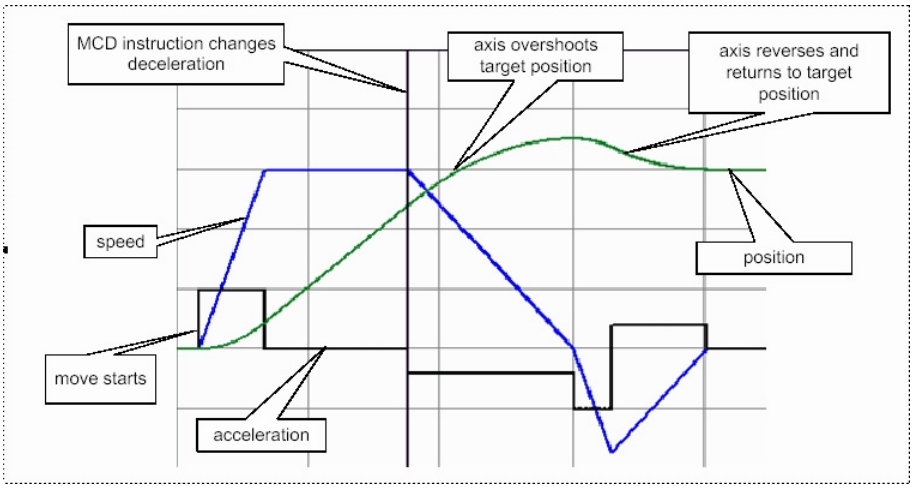
You use a Motion Change Dynamics (MCD) instruction to reduce the deceleration while a Motion Axis Move (MAM) instruction is in process. The axis continues past the target position of the move, stops, and returns to the target position.

Look For


Move_Cmnd
MAM
Motion Axis Move
Axis My_Axis
Motion Control My_Axis_MI.Move
Move Type 1
Position 600
Speed 60
Speed Units Units per sec
Accel Rate My_Axis_Vars.C.Auto_Accel
20.0
Accel Units Units per sec2
Decel Rate My_Axis_Vars.C.Auto_Decel
20.0
Decel Units Units per sec2
Profile Trapezoidal
Accel Jerk My_Axis_Vars.C.Auto_Accel_Jerk
100.0
Decel Jerk My_Axis_Vars.C.Auto_Decel_Jerk
100.0
Jerk Units % of Time
Merge Disabled
Merge Speed Programmed
Lock Position 0
Lock Direction None
Event Distance 0
Calculated Data 0
EN
DN
ER
IP
PC
MCD instruction that reduces the deceleration of the move.
Reduce_Deceleration
MCD
Motion Change Dynamics
Axis My_Axis
Motion Control My_Axis_MI.Change_Move
Motion Type Move
Change Speed No
Speed 0
Change Accel No
Accel Rate 0
Change Decel Yes
Decel Rate 8
Change Accel Jerk 0
Accel Jerk 0
Change Decel Jerk 0
Decel Jerk 0
Speed Units Units per sec
Accel Units Units per sec2
Decel Units Units per sec2
Jerk Units Units per sec3
EN
DN
ER

Cause

The axis does not have enough time at the new lower deceleration to stop at the target position. It stops past the target position. Then it corrects to get back to the target position.



Corrective Action

To avoid overshooting position, do one of the following.

- Avoid decreasing the deceleration or deceleration jerk while an axis is decelerating along an S-curve profile.
- Avoid increasing the programmed speed while an axis is decelerating along an S-curve profile. This has the same effect as decreasing the deceleration jerk.

Test any changes in small increments to make sure a change does not cause an overshoot during normal operation.

See also

Motion Axis Stop (MAS) on page 74

Motion Axis Jog (MAJ) on page 91

Motion Change Dynamics (MCD) on page 128

Motion Axis Move (MAM) on page 101

Overview of motion-related data types

Follow this chapter to gain information about motion-related data types.

CAM Structure

The CAM data type consists of slave and master point pairs, as well as an interpolation type. Because there is no association with a specific axis position or time, the point values are unitless. Specify the interpolation type for each segment as either linear or cubic. This table describes the format of the cam array element:

Mnemonic	Data Type	Description
MASTER	REAL	The x value of the point.
SLAVE	REAL	The y value of the point.
Segment Type	DINT	The type of interpolation 0 = linear 1 = cubic

Tip: You cannot convert the CAM data type to CAM_EXTENDED, and you cannot convert the CAM_EXTENDED data type to CAM.

CAM_PROFILE Structure

The CAM_PROFILE data type is an array of coefficients representing a calculated cam profile that can be used as input to a time cam or position cam instruction. The only element available is Status:

Mnemonic	Data Type	Description
Status	DINT	The status parameter indicates that the Cam Profile array element has been calculated. If you attempt the execution of a camming instruction using an uncalculated element in a cam profile, the instruction produces an error. 0 = cam profile element was not calculated 1 = cam profile element is being calculated 2 = cam profile element was calculated n = cam profile element was calculated and is being used by (n-2) MAPC and MATC instructions.

IMPORTANT Do not attempt to copy data between CAM_PROFILE and CAM_PROFILE_EXTENDED data types. It will result in data corruption.

Tip: You cannot convert the CAM_PROFILE data type to CAM_PROFILE_EXTENDED, and you cannot convert the CAM_EXTENDED data type to CAM.

MOTION_GROUP Structure

There is one MOTION_GROUP structure per controller. This structure contains status information about the group. Every motion instruction references the same MOTION_GROUP structure.

Mnemonic	Data Type	Description			
.GroupStatus	DINT	The status bits for the group.			
		Bit	**Number**	**Data Type**	**Description**
		.InhibStatus	00	DINT	inhibit status
		.GroupSynced	01	DINT	synchronization status
		.AxisInhibitStatus	02	DINT	
		-no-tag	02	DINT	Timer Event started
		Reserved	03-31		
.MotionFault	DINT	The motion fault bits for the group.			
		Bit	**Number**	**Data Type**	**Description**
		.ACAsyncConnFault	00	DINT	asynchronous connection fault
		.ACSyncConnFault	01	DINT	synchronous connection fault (controller declared)
		Reserved	02-31		
.ServoFault	DINT	The servo-module fault bits for the group.			
		Bit	**Number**	**Data Type**	**Description**
		.POtrvlFault	00	DINT	positive overtravel fault
		.NOtrvlFault	01	DINT	negative overtravel fault
		.PosErrorFault	02	DINT	position error fault
		.EncCHALossFault	03	DINT	encoder channel A loss fault
		.EncCHBLossFault	04	DINT	encoder channel B loss fault
		.EncCHZLossFault	05	DINT	encoder channel Z loss fault
		.EncNsFault	06	DINT	encoder noise fault
		.DriveFault	07	DINT	drive fault
		.SynConnFault	00	DINT	synchronous connection fault (servo declared)
		.HardFault	01	DINT	servo hardware fault
		Reserved	02-31		
.GroupFault	DINT	The fault bits for the group.			
		Bit	**Number**	**Data Type**	**Description**
		.GroupOverlapFault	00	DINT	group overlap fault
		.CSTLossFault	01	DINT	the controller has lost synchronization with the CST master
		.GroupTaskLoadingFault	02	DINT	the group coarse update period is too low; user application tasks are not getting enough time to execute
		Reserved	03-31		
AxisFault	DINT	The faults bit for the axis.			
		Bit	**Number**	**Data Type**	**Description**

		.PhysicalAxisFault	00	BOOL	A Servo or Drive fault has occurred.
		.ModuleFault	01	BOOL	A serious fault has occurred with the motion module associated with the selected axis. Usually affects all axes associated with the motion module.
		.ConfigFault	02	BOOL	One or more axis attributes associated with a motion module or drive has not been successfully updated to match the value of the corresponding attribute of the local controller.
		Reserved	03-31		

MOTION_INSTRUCTION Data Type

You must define a motion control tag for each motion instruction that you use. The tag uses the MOTION_INSTRUCTION data type and stores status information about the instruction.

Mnemonic	Data Type	Description	
FLAGS	DINT	Use this DINT to access all the status bits for the instruction in one 32-bit value.	
		For this status bit	**Use this bit number**
		EN	31
		DN	29
		ER	28
		PC	27
		IP	26
		AC	23
		DECEL	1
		ACCEL	0
EN	BOOL	The enable bit indicates that the instruction is enabled (the rung-in and rung-out condition is true).	
DN	BOOL	The done bit indicates that all calculations and messaging (if any) are complete.	
ER	BOOL	The error bit indicates when the instruction is used illegally.	
PC	BOOL	The process complete bit indicates that the operation is complete. The .DN bit sets after an instruction has completed execution. The .PC bit sets when the initiated process has completed.	
IP	BOOL	The in process bit indicates that a process is being executed.	
AC	BOOL	The Active Bit lets you know which instruction is controlling the motion when you have instructions queued. It sets when the instruction becomes active. It is reset when the Process Complete bit is set or when the instruction is stopped.	
ACCEL	BOOL	The .ACCEL bit indicates that the velocity has increased for the individual instruction that it is tied to, that is, jog, move, gearing.	
TrackingMaster	BOOL	Indicates that the Slave Coordinate System is tracking the Master Axis (only used in Master Driven Mode).	
CalculatedDataAvailable	BOOL	Indicates that the requested data has been returned in the Calculated Data array element and that the Logix Designer application has updated the output data in the Calculated Data parameter. Only one status bit is used to indicate all Calculated Data is available.	
DECEL	BOOL	The .DECEL bit indicates that the velocity has decreased for the individual instruction that it is tied to, that is, jog, move, gearing.	
ERR	INT	The error value contains the error code associated with a motion function. See Error Codes (ERR) for Motion Instructions.	
STATUS	SINT	The status of any message associated with the motion function.	
		Message Status	**Description**
		0x0	The message was successful.
		0x1	The module is processing another message.
		0x2	The module is waiting for a response to a previous message.
		0x3	The response to a message failed.
		0x4	The module is not ready for messaging.

STATE	SINT	The execution status value keeps track of the execution state of a function. Many motion functions have several steps and this value tracks these steps. The execution status is always set to 0 when the controller sets the EN bit for a motion instruction. Other execution states depend on the motion instruction.
SEGMENT	DINT	A segment is the distance from one point up to but, not including the next point. A SEGMENT gives the relative position by segment number as the Cam is executing.
EXERR	SINT	Extended error code - use it for more information about an error.

See also

Error Codes (ERR) on page 581

OUTPUT_CAM Structure

The OUTPUT_CAM data type is an array that defines the specifics for each Output Cam element. The OUTPUT_CAM contains the following members:

Mnemonic	Data Type	Description
OutputBit	DINT	You must select an output bit within the range of 0 to 31. A selection of less than 0 or greater than 31 results in an Illegal Output Cam error and the cam element is not considered.
LatchType	DINT	The Latch Type determines how the corresponding output bit is set. A value of less than 0 or greater than 3 results in an Illegal Output Cam error and a latch type of Inactive is used. 0 = Inactive - The output bit is not changed. 1 = Position - The output bit is set when the axis enters the compensated cam range. 2 = Enable - The output bit is set when the enable bit becomes active. 3 = Position and Enable - The output bit is set when the axis enters the compensated cam range and the enable bit becomes active.
UnlatchType	DINT	The Unlatch Type determines how the output bit is reset. Selecting a value less than 0 or greater than 5 results in an Illegal Output Cam error and an unlatch type of Inactive is used. 0 = Inactive- -The output bit is not changed. 1 = Position - The output bit is reset when the axis leaves the compensated cam range. 2 = Duration - The output bit is reset when the duration expires. 3 = Enable - The output bit is reset when the enable bit becomes inactive. 4 = Position and Enable - The output bit is reset when the axis leaves the compensated cam range or the enable bit becomes inactive. 5 = Duration and Enable - The output bit is reset when the duration expires or the enable bit becomes inactive.
Left	REAL	The left cam position along with the right cam position define the cam range of the Output Cam element. The left and right cam positions specify the latch or unlatch positions of the output bit when the latch or unlatch type is set to **Position** or **Position and Enable** with the enable bit active. If the left position is less than the Cam Start position or greater than the Cam End position, an Illegal Output Cam error is returned and the cam element is not considered.
Right	REAL	The right cam position along with the left cam position define the cam range of the Output Cam element. The right and left cam positions specify the latch or unlatch positions of the output bit when the latch or unlatch type is set to **Position** or **Position and Enable** with the enable bit active. If the right position is less than the Cam Start position or greater than the Cam End position, an Illegal Output Cam error is returned and the cam element is not considered.
Duration	REAL	Duration specifies the time in seconds between latching and unlatching when the Unlatch Type is **Duration** or **Duration and Enable** with the enable bit active. A value less than or equal to 0 results in an Illegal Output Cam error and the cam element is not considered.
EnableType	DINT	This defines the source and polarity of the specified EnableBit when LatchType or UnlatchType is **Enable**, **Position and Enable** or **Duration and Enable**. A value of less than 0 or greater than 31 results in an Illegal Output Cam error and the cam element is not considered. 0 = Input - The enable bit is in the Input parameter. 1 = Inverted Input - The enable bit is in the input parameter and is active low. 2 = Output - The enable bit is in the Output parameter. 3 = Inverted Output - The enable bit is in the Output parameter and is active low.
EnableBit	DINT	The value of the Enable Bit selected must be between 0 and 31 when LatchType or UnlatchType is **Enable**, **Position and Enable** or **Duration and Enable**. A value of less than 0 or greater than 31 results in an Illegal Output Cam error and the cam element is not considered.

OUTPUT_COMPENSATION Structure

The OUTPUT_COMPENSATION data type defines the details for each output bit by setting the characteristics of each actuator. OUTPUT_COMPENSATION contains the following members.

Mnemonic	Data Type	Description
Offset	REAL	Offset provides position compensation for both the latch and unlatch operations.
LatchDelay	REAL	Latch delay, programmed in seconds, provides time compensation for the latch operation.
UnlatchDelay	REAL	Unlatch delay, programmed in seconds, provides time compensation for the unlatch operation.
Mode	DINT	The mode determines the behavior of the output bit. The available mode options are: 0 = Normal - The output bit is set for the latch operation and is reset for the unlatch operation. 1 = Inverted - The output bit is reset for the latch operation and set for the unlatch operation. 2 = Pulsed - The output bit is set for the latch operation and for the on-duty state of the pulse and is reset for the unlatch operation and for the off-duty state of the pulse. 3 = Inverted and Pulsed - The output bit is reset for the latch operation and for the on-duty state of the pulse and is set for the unlatch operation and for the off-duty state of the pulse. A value of less than 0 or greater than 3 results in an Illegal Output Compensation error and a mode of Normal is used.
CycleTime	REAL	Pulse time in seconds. If the mode is Pulsed or Inverted and Pulsed, and CycleTime is less than or equal to 0, an Illegal Output Compensation error results and a mode of Normal is used.
DutyCycle	REAL	The percent of CycleTime in which the pulse is to be turned on (i.e., on-duty). A value of 50 represents 50 percent on-duty. A value of less than 0 or greater than 100 returns an Illegal Output Compensation error and DutyCycle is limited to either 0 or 100.

Overview of Structured Text Programming

This chapter describes issues that are unique with structured text programming. Review the information in this chapter to gain a better understanding about how your structure text programming will execute.

Structured Text Syntax

Structured text is a textual programming language that uses statements to define what to execute.

- Structured text is not case sensitive.
- Use tabs and carriage returns (separate lines) to make your structured text easier to read. They have no effect on the execution of the structured text.

Structured text is not case sensitive. Structured text can contain these components.

Term	Definition	Examples
Assignment	Use an assignment statement to assign values to tags. The := operator is the assignment operator. Terminate the assignment with a semi colon ';.'	tag := expression;
Expression	An expression is part of a complete assignment or construct statement. An expression evaluates to a number (numerical expression), a String (string expression), or to a true or false state (BOOL expression)	
Tag Expression	A named area of the memory where data is stored (BOOL, SINT, INT, DINT, REAL, String).	value1
Immediate Expression	A constant value	4
Operators Expression	A symbol or mnemonic that specifies an operation within an expression.	tag1 + tag2 tag1 >= value1
Function Expression	When executed, a function yields one value. Use parentheses to contain the operand of a function. Even though their syntax is similar, functions differ from instructions in that functions can be used only in expressions. Instructions cannot be used in expressions.	function(tag1)

Instruction	An instruction is a standalone statement.
	An instruction uses parentheses to contain its operands.
	Depending on the instruction, there can be zero, one, or multiple operands.
	When executed, an instruction yields one or more values that are part of a data structure. Terminate the instruction with a semi colon(;).
	Even though their syntax is similar, instructions differ from functions in that instructions cannot be used in expressions. Functions can be used only in expressions.
		instruction();
		
		instruction(operand);
		
		instruction(operand1, operand2,operand3);

Construct	A conditional statement used to trigger structured text code (that is, other statements). Terminate the construct with a semi colon (;).	IF...THEN CASE FOR...DO WHILE...DO REPEAT...UNTIL EXIT

Comment	Text that explains or clarifies what a section of structured text does. Use comments to make it easier to interpret the structured text. Comments do not affect the execution of the structured text. Comments can appear anywhere in structured text.	//comment (*start of comment . . . end of comment*) /*start of comment . . . end of comment*/

See also

Structured Text Components: Assignments on page 666

Structured Text Components: Expressions on page 668

Structured Text Components: Instructions on page 673

Structured Text Components: Constructs on page 674

Structured Text Components: Comments on page 689

ST Components: Assignments

Use an assignment to change the value stored within a tag. An assignment has this syntax:

tag := expression;

where:

Component	Description	
Tag	Represents the tag that is getting the new value; the tag must be a BOOL, SINT, INT, DINT, STRING, or REAL. **Tip:** The STRING tag is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.	
:=	Is the assignment symbol	
Expression	Represents the new value to assign to the tag	
	If tag is this data type	**Use this type of expression**
	BOOL	BOOL
	SINT INT DINT REAL	Numeric
	STRING (CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only).	String type, including string tag and string literal (CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only).
;	Ends the assignment	

The tag retains the assigned value until another assignment changes the value.

The expression can be simple, such as an immediate value or another tag name, or the expression can be complex and include several operators and functions, or both. Refer to Expressions for more information.

Tip: I/O module data updates asynchronously to the execution of logic. If you reference an input multiple times in your logic, the input could change state between separate references. If you need the input to have the same state for each reference, buffer the input value and reference that buffer tag. For more information, see Logix 5000 Controllers Common Procedures, publication 1756-PM001. You can also use Input and Output program parameters which automatically buffer the data during the Logix Designer application execution. See LOGIX 5000 Controllers Program Parameters Programming Manual, publication 1756-PM021.

See also

Assign an ASCII character to a string data member on page 668

Specify a non-retentive assignment on page 667

Structured Text Components: Expressions on page 668

Character string literals on page 675

Specify a non-retentive assignment

The non-retentive assignment is different from the regular assignment described above in that the tag in a non-retentive assignment is reset to zero each time the controller:

- Enters the Run mode
- Leaves the step of an SFC if you configure the SFC for Automatic reset. This applies only if you embed the assignment in the action of the step or use the action to call a structured text routine by using a JSR instruction.

A non-retentive assignment has this syntax:

tag [:=] expression ;

where:

Component	Description	
tag	Represents the tag that is getting the new value; the tag must be a BOOL, SINT, INT, DINT, STRING, or REAL. **Tip:** The STRING tag is applicable to CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only.	
[:=]	Is the non-retentive assignment symbol.	
expression	Represents the new value to assign to the tag.	
	If tag is this data type	**Use this type of expression**
	BOOL	BOOL
	SINT	Numeric
	INT	
	DINT	
	REAL	

	STRING (CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only).	String type, including string tag and string literal CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers(only)

See also

Assign an ASCII character to a string data member on page 668

Structured Text Components: Assignments on page 666

ASCII character to string member

Assign an ASCII character to a string data member

Use the assignment operator to assign an ASCII character to an element of the DATA member of a string tag. To assign a character, specify the value of the character or specify the tag name, DATA member, and element of the character. For example:

This is OK	This is not OK
string1.DATA[0] := 65;	string1.DATA[0] := A;
string1.DATA[0]:= string2.DATA[0];	string1 := string2; **Tip:** This assigns all content of string2 to string1 instead of just one character.

To add or insert a string of characters to a string tag, use either of these ASCII string instructions:

To	Use this instruction
Add characters to the end of a string	CONCAT
Insert characters into a string	INSERT

See also

Structured Text Components: Expressions on page 668

Character string literals on page 675

ST Components: Expressions

An expression is a tag name, equation, or comparison. To write an expression, use any of the following:

- Tag name that stores the value (variable)
- Number that you enter directly into the expression (immediate value)
- String literal that you enter directly into the expression (CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers only)
- Functions, such as: ABS, TRUNC
- Operators, such as: +, -, <, >, And, Or

Follow these guidelines for writing expressions:

- Use any combination of upper-case and lower-case letter. For example, these variations of "AND" are acceptable: AND, And, and.
- For more complex requirements, use parentheses to group expressions within expressions. This makes the whole expression easier to read, and ensures that the expression executes in the desired sequence.

Use these expressions for structured text:

BOOL expression: An expression that produces the BOOL value of 1 (true) or 0 (false).

- A bool expression uses bool tags, relational operators, and logical operators to compare values or check if conditions are true or false. For example, tag1>65.
- A simple bool expression can be a single BOOL tag.
- Typically, use bool expressions to condition the execution of other logic.

Numeric expression: An expression that calculates an integer or floating-point value.

- A numeric expression uses arithmetic operators, arithmetic functions, and bitwise operators. For example, tag1+5.
- Nest a numeric expression within a BOOL expression. For example, (tag1+5)>65.

String expression: An expression that represents a string

- A simple expression can be a string literal or a string tag

Use this table to select the operators for expressions.

If	Use
Calculating an arithmetic value	Arithmetic operators and functions
Comparing two values or strings	Relational operators
Verifying if conditions are true or false	Logical operators
Comparing the bits within values	Bitwise operators

See also

Use arithmetic operators and functions on page 669

Use relational operators on page 670

Use logical operators on page 672

Use bitwise operators on page 672

Use operators and functions

Combine multiple operators and functions in arithmetic expressions.

Operators calculate new values.

To	Use this operator

Add	+
Subtract/negate	-
Multiply	*
Exponent (x to the power of y)	**
Divide	/
Modulo-divide	MOD

Functions perform math operations. Specify a constant, a non-Boolean tag, or an expression for the function.

For	Use this function
Absolute value	ABS (numeric_expression)
Arc cosine	ACOS (numeric_expression)
Arc sine	ASIN (numeric_expression)
Arc tangent	ATAN (numeric_expression)
Two-argument arctangent	ATAN2 (numeric_expression)
Cosine	COS (numeric_expression)
Radians to degrees	DEG (numeric_expression)
Natural log	LN (numeric_expression)
Log base 10	LOG (numeric_expression)
Degrees to radians	RAD (numeric_expression)
Sine	SIN (numeric_expression)
Square root	SQRT (numeric_expression)
Tangent	TAN (numeric_expression)
Truncate	TRUNC (numeric_expression)

The table provides examples for using arithmetic operators and functions.

Use this format	Example	
	For this situation	Write
value1 operator value2	If gain_4 and gain_4_adj are DINT tags and your specification says: 'Add 15 to gain_4 and store the result in gain_4_adj''	gain_4_adj := gain_4+15;
operator value1	If alarm and high_alarm are DINT tags and your specification says: 'Negate high_alarm and store the result in alarm.'	alarm:= -high_alarm;
function(numeric_expression)	If overtravel and overtravel_POS are DINT tags and your specification says: 'Calculate the absolute value of overtravel and store the result in overtravel_POS.'	overtravel_POS := ABS(overtravel);
value1 operator (function((value2+value3)/2)	If adjustment and position are DINT tags and sensor1 and sensor2 are REAL tags and your specification says: 'Find the absolute value of the average of sensor1 and sensor2, add the adjustment, and store the result in position.'	position := adjustment + ABS((sensor1 + sensor2)/2);

See also

Structured Text Components: Expressions on page 668

Use relational operators

Relational operators compare two values or strings to provide a true or false result. The result of a relational operation is a BOOL value.

If the comparison is	The result is
True	1
False	0

Use these relational operators.

For this comparison	Use this operator	Optimal data type
Equal	=	DINT, REAL, String type
Less than	<	DINT, REAL, String type
Less than or equal	<=	DINT, REAL, String type
Greater than	>	DINT, REAL, String type
Greater than or equal	>=	DINT, REAL, String type
Not equal	<>	DINT, REAL, String type

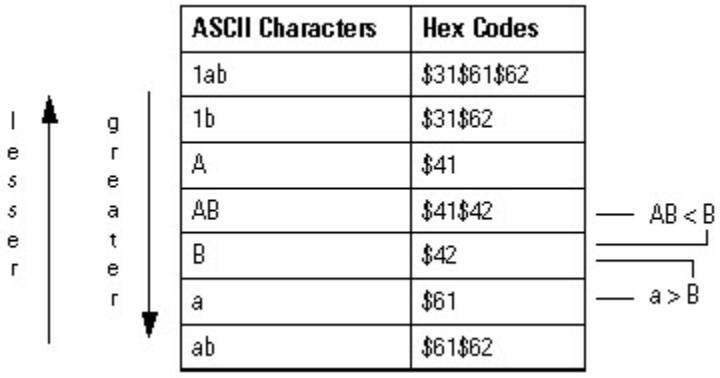
The table provides examples of using relational operators

Use this format	Example	
	For this situation	Write
value1 operator value2	If temp is a DINT tag and your specification says: 'If temp is less than 100· then…'	IF temp<100 THEN...
stringtag1 operator stringtag2	If bar_code and dest are string tags and your specification says: 'If bar_code equals dest then...'	IF bar_code=dest THEN...
stringtag1 operator 'character string literal'	If bar_code is a string tag and your specification says: 'If bar_code equals 'Test PASSED' then...'	IF bar_code='Test PASSED' THEN...
char1 operator char2 To enter an ASCII character directly into the expression, enter the decimal value of the character.	If bar_code is a string tag and your specification says: 'If bar_code.DATA[0] equals 'A' then...'	IF bar_code.DATA[0]=65 THEN...
bool_tag := bool_expressions	If count and length are DINT tags, done is a BOOL tag, and your specification says: 'If count is greater than or equal to length, you are done counting.'	Done := (count >= length);

How strings are evaluated

The hexadecimal values of the ASCII characters determine if one string is less than or greater than another string.

- When the two strings are sorted as in a telephone directory, the order of the strings determines which one is greater.



- Strings are equal if their characters match.

- Characters are case sensitive. Upper case "A" ($41) is not equal to lower case "a" ($61).

See also

Structured Text Components: Expressions on page 668

Use logical operators

Use logical operators to verify if multiple conditions are true or false. The result of a logical operation is a BOOL value.

If the comparison is	The result is
true	1
false	0

Use these logical operators.

For this comparison	Use this operator	Optimal data type
logical AND	&, AND	BOOL
logical OR	OR	BOOL
logical exclusive OR	XOR	BOOL
logical complement	NOT	BOOL

The table provides examples of using logical operators.

Use this format	Example	
	For this situation	Use
BOOLtag	If photoeye is a BOOL tag and your specification says: "If photoeye_1 is on then..."	IF photoeye THEN...
NOT BOOLtag	If photoeye is a BOOL tag and your specification says: "If photoeye is off then..."	IF NOT photoeye THEN...
expression1 & expression2	If photoeye is a BOOL tag, temp is a DINT tag, and your specification says: "If photoeye is on and temp is less than 100 then..."	IF photoeye & (temp<100) THEN...
expression1 OR expression2	If photoeye is a BOOL tag, temp is a DINT tag, and your specification says: "If photoeye is on or temp is less than 100 then...".	IF photoeye OR (temp<100) THEN...
expression1 XOR expression2	If photoeye1 and photoeye2 are BOOL tags and your specification says: "If: photoeye1 is on while photoeye2 is off or photoeye1 is off while photoeye2 is on then..."	IF photoeye1 XOR photoeye2 THEN...
BOOLtag := expression1 & expression2	If photoeye1 and photoeye2 are BOOL tags, open is a BOOL tag, and your specification says: "If photoeye1 and photoeye2 are both on, set open to true"	open := photoeye1 & photoeye2;

See also

Structured Text Components: Expressions on page 668

Use bitwise operators

Bitwise operators manipulate the bits within a value based on two values.

The following provides an overview of the bitwise operators.

For	Use this operator	Optimal data type
bitwise AND	&, AND	DINT

bitwise OR	OR	DINT
bitwise exclusive OR	XOR	DINT
bitwise complement	NOT	DINT

This is an example.

Use this format	Example	
	For this situation	Use
value1 operator value2	If input1, input2, and result1 are DINT tags and your specification says: "Calculate the bitwise result of input1 and input2. Store the result in result1."	result1 := input1 AND input2;

See also

Structured Text Components: Expressions on page 668

Determine the order of execution

The operations written into an expression perform in a prescribed order.

- Operations of equal order perform from left to right.
- If an expression contains multiple operators or functions, group the conditions in parenthesis "( )". This ensures the correct order of execution, and makes it easier to read the expression.

Order	Operation
1	()
2	function (...)
3	**
4	- (negate)
5	NOT
6	*,/,MOD
7	+,- (subtract)
8	<,<=,>,>=
9	=,<>
10	&,AND
11	XOR
12	OR

See also

Structured Text Components: Expressions on page 668

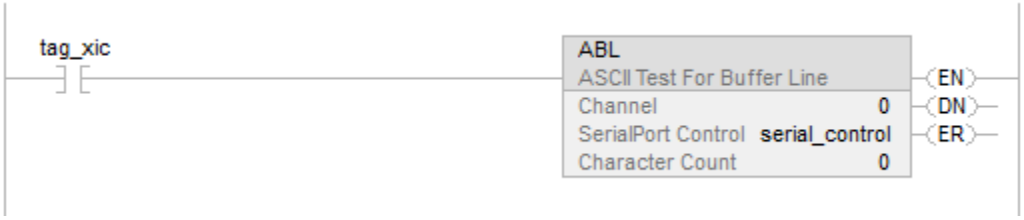
ST Components: Instructions

Structured text statements can also be instructions. A structured text instruction executes each time it is scanned. A structured text instruction within a construct executes every time the conditions of the construct are true. If the conditions of the construct are false, the statements within the construct are not scanned. There is no rung-condition or state transition that triggers execution.

This differs from function block instructions that use EnableIn to trigger execution. Structured text instructions execute as if EnableIn is always set.

This also differs from ladder diagram instructions that use rung-condition-in to trigger execution. Some ladder diagram instructions only execute when rung- condition-in toggles from false to true. These are transitional ladder diagram instructions. In structured text, instructions execute when they are scanned unless pre-conditioning the execution of the structured text instruction.

For example, the ABL instruction is a transitional instruction in ladder diagram. In this example, the ABL instruction only executes on a scan when tag_xic transitions from cleared to set. The ABL instruction does not execute when tag_xic stays set or when tag_xic clears.



In structured text, if writting this example as:

IF tag_xic THEN ABL(0,serial_control);

END_IF;

The ABL instruction will execute every scan that tag_xic is set, not just when tag_xic transitions from cleared to set.

If you want the ABL instruction to execute only when tag_xic transitions from cleared to set, you have to condition the structured text instruction. Use a one-shot to trigger execution.

osri_1.InputBit := tag_xic;

OSRI(osri_1);

IF (osri_1.OutputBit) THEN

ABL(0,serial_control);

END_IF;

ST Components: Constructs

Program constructs alone or nest within other constructs.

If	Use this construct
Doing something if or when specific conditions occur	IF... THEN
Selecting what to do based on a numerical value	CASE... OF
Doing something a specific number of times before doing anything else	FOR... DO
Continuing doing something when certain conditions are true	WHILE... DO
Continuing doing something until a condition is true	REPEAT... UNTIL

Some Key Words are Reserved

These constructs are not available:

- GOTO
- REPEAT

Logix Designer application will not let you use them as tag names or constructs.

See also

IF_THEN on page 676

CASE_OF on page 679

FOR_DO on page 681

WHILE_DO on page 684

REPEAT_UNTIL on page 686

Character string literals

Character string literals include single byte or double byte encoded characters. A single-byte string literal is a sequence of zero or more characters that are prefixed and terminated by the single quote character ('). In single byte character strings, the three-character combination of the dollar sign ($) followed by two hexadecimal digits is interpreted as the hexadecimal representation of the eight-bit character code as shown in the following table.

Tip: Character string literals are only applicable to the CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers.
Studio 5000 only supports single byte characters.

Character string literals

No.	Description	Example
1a	Empty string (length zero)	''
1b	String of length one or character CHAR containing a single character	'A'
1c	String of length one or character CHAR containing the “space” character	' '
1d	String of length one or character CHAR containing the “single quote” character	'$''
1e	String of length one or character CHAR containing the “double quote” character	'"'
1f	Support of two character combinations	'RL'
1g	Support of a character representation with ‘$’ and two hexadecimal characters	'$0A'

Two-character combinations in character strings

No.	Description	Example
1	Dollar sign	$$
2	Single quote	$'
3	Line feed	$L or $l

4	Newline	$N or $n
5	Form feed (page)	$P or $p
6	Carriage return	$R or $r
7	Tabulator	$T or $t

Tip: The newline character provides an implementation-independent means of defining the end of a line of data for both physical and file I/O; for printing, the effect is that of ending a line of data and resuming printing at the beginning of the next line.
The $' combination is only valid inside single quoted string literals.

See also

Structured Text Components: Assignments on page 666

String Types

Store ASCII characters in tags that use a string type data type to:

- Use the default STRING data type, which stores up to 82 characters
- Create a new string type that stores less or more characters

To create a new string type, refer to the Logix 5000 Controllers ASCII Strings Programming Manual publication 1756-PM013.

Each string type contains the following members:

Name	Data Type	Description	Notes
LEN	DINT	number of characters in the string	The LEN automatically updates to the new count of characters whenever using: • The String Browser to enter characters • Instructions that read, convert, or manipulate a string The LEN shows the length of the current string. The DATA member may contain additional, old characters, which are not included in the LEN count.
DATA	SINT array	ASCII characters of the string	To access the characters of the string, address the name of the tag. For example, to access the characters of the string_1 tag, enter string_1. Each element of the DATA array contains one character. Create new string types that store less or more characters.

See also

Character string literals on page 675

IF_THEN

Use IF_THEN to complete an action when specific conditions occur.

Operands

IF bool_expression THEN

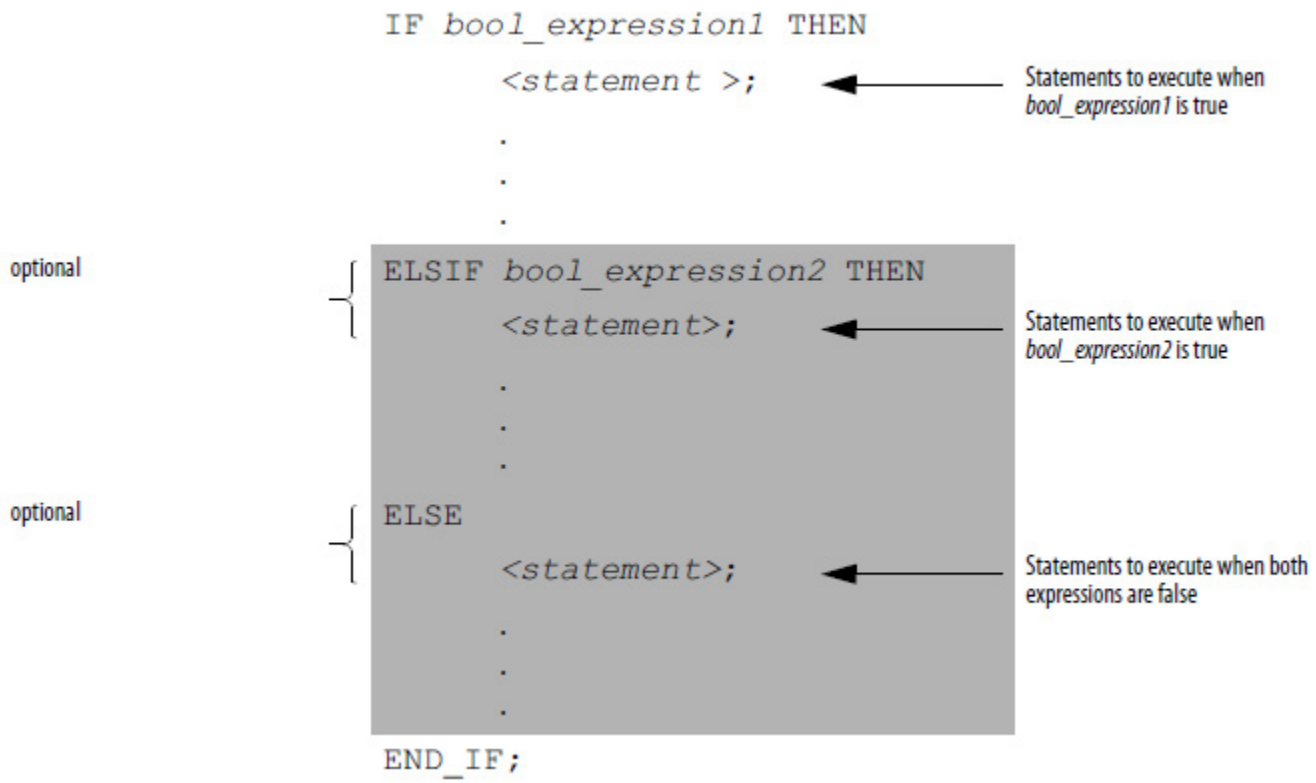
<statement>;

Operand	Type	Format	Enter

Bool_ expression	BOOL	Tag expression	BOOL tag or expression that evaluates to a BOOL value (BOOL expression)

Description

The syntax is described in the table.



To use ELSIF or ELSE, follow these guidelines.

To select from several possible groups of statements, add one or more ELSIF statements.

Each ELSIF represents an alternative path.

Specify as many ELSIF paths as you need.

The controller executes the first true IF or ELSIF and skips the rest of the ELSIFs and the ELSE.

To do something when all of the IF or ELSIF conditions are false, add an ELSE statement.

The table summarizes different combinations of IF, THEN, ELSIF, and ELSE.

If	And	Use this construct
Doing something if or when conditions are true	Do nothing if conditions are false	IF_THEN
	Do something else if conditions are false	IF_THEN_ELSE
Selecting alternative statements or groups of	Do nothing if conditions are false	IF_THEN_ELSIF

statements based on input conditions	Assign default statements if all conditions are false	IF_THEN_ELSIF_ELSE

Affects Math Status Flags

No

Major/Minor Faults

None.

Examples

Example 1

IF...THEN

If performing this	Enter this structured text
IF rejects > 3 then	IF rejects > 3 THEN
conveyor = off (0)	conveyor := 0;
alarm = on (1)	alarm := 1;
	END_IF;

Example 2

IF_THEN_ELSE

If performiing this	Enter this structured text
If conveyor direction contact = forward (1) then	IF conveyor_direction THEN
light = off	light := 0;
Otherwise light = on	ELSE
	light [:=] 1;
	END_IF;

The [:=] tells the controller to clear light whenever the controller does the following :

Enters the RUN mode.

Leaves the step of an SFC if you configure the SFC for Automatic reset. (This applies only if you embed the assignment in the action of the step or use the action to call a structured text routine via a JSR instruction.)

Example 3

IF...THEN...ELSIF

If performing this	Enter this structured text
If sugar low limit switch = low (on) and sugar high limit switch = not high (on) then	IF Sugar.Low & Sugar.High THEN
inlet valve = open (on)	Sugar.Inlet [:=] 1;
Until sugar high limit switch = high (off)	ELSIF NOT(Sugar.High) THEN
	Sugar.Inlet := 0;
	END_IF;

The [:=] tells the controller to clear Sugar.Inlet whenever the controller does the following :

Enters the RUN mode.

Leaves the step of an SFC if you configure the SFC for Automatic reset. (This applies only if you embed the assignment in the action of the step or use the action to call a structured text routine via a JSR instruction.)

Example 4

IF...THEN...ELSIF...ELSE

If performing this	Enter this structured text
If tank temperature > 100	IF tank.temp > 200 THEN
then pump = slow	pump.fast :=1; pump.slow :=0; pump.off :=0;
If tank temperature > 200	ELSIF tank.temp > 100 THEN
then pump = fast	pump.fast :=0; pump.slow :=1; pump.off :=0;
Otherwise pump = off	ELSE
	pump.fast :=0; pump.slow :=0; pump.off :=1;
	END_IF;

CASE_OF

Use CASE_OF to select what to do based on a numerical value.

Operands

CASE numeric_expression OF

selector1: statement;

selectorN: statement; ELSE

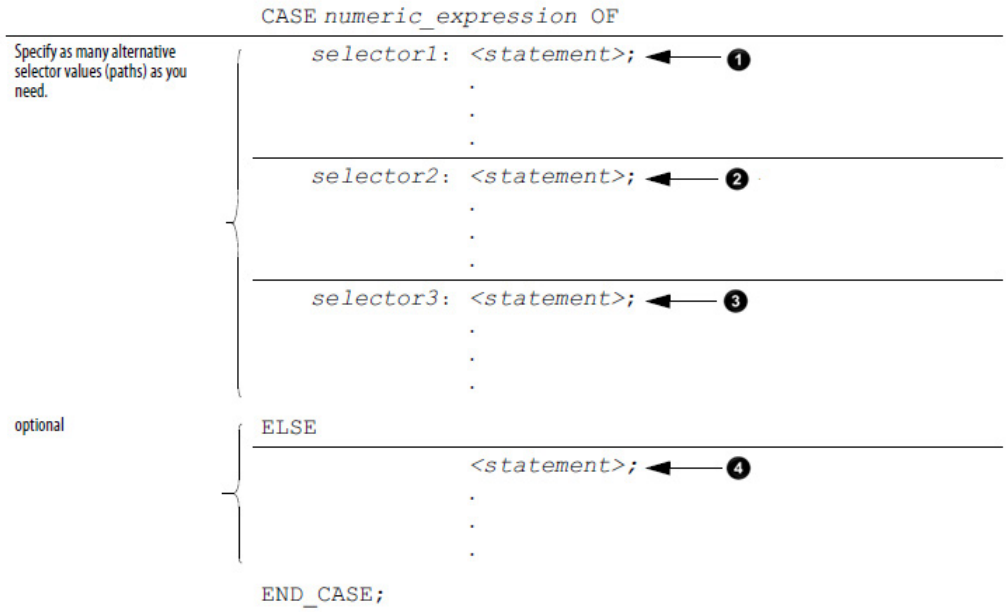
Structured Text

Operand	Type	Format	Enter
Numeric_ expression	SINT INT DINT REAL	Tag expression	Tag or expression that evaluates to a number (numeric expression)
Selector	SINT INT DINT REAL	Immediate	Same type as numeric_expression

IMPORTANT If using REAL values, use a range of values for a selector because a REAL value is more likely to be within a range of values than an exact match of one, specific value.

Description

The syntax is described in the table.



These are the syntax for entering the selector values.

When selector is	Enter
One value	value: statement
Multiple, distinct values	value1, value2, valueN : <statement> Use a comma (,) to separate each value.
A range of values	value1..valueN : <statement> Use two periods (..) to identify the range.
Distinct values plus a range of values	valuea, valueb, value1..valueN : <statement>

The CASE construct is similar to a switch statement in the C or C++ programming languages. With the CASE construct, the controller executes only the statements that associated with the first matching selector value.

Execution always breaks after the statements of that selector and goes to the END_CASE statement.

Affects Math Status Flags

No

Major/Minor Faults

None

Example

If you want this	Enter this structured text
If recipe number = 1 then Ingredient A outlet 1 = open (1) Ingredient B outlet 4 = open (1)	CASE recipe_number OF
	1: Ingredient_A.Outlet_1 :=1; Ingredient_B.Outlet_4 :=1;
If recipe number = 2 or 3 then Ingredient A outlet 4 = open (1) Ingredient B outlet 2 = open (1)	2,3: Ingredient_A.Outlet_4 :=1; Ingredient_B.Outlet_2 :=1;
If recipe number = 4, 5, 6, or 7 then Ingredient A outlet 4 = open (1) Ingredient B outlet 2 = open (1)	4 to 7: Ingredient_A.Outlet_4 :=1; Ingredient_B.Outlet_2 :=1;
If recipe number = 8, 11, 12, or 13 then Ingredient A outlet 1 = open (1) Ingredient B outlet 4 = open (1)	8,11...13 Ingredient_A.Outlet_1 :=1; Ingredient_B.Outlet_4 :=1;
Otherwise all outlets = closed (0)	ELSE
	Ingredient_A.Outlet_1 [:=]0; Ingredient_A.Outlet_4 [:=]0; Ingredient_B.Outlet_2 [:=]0; Ingredient_B.Outlet_4 [:=]0;
	END_CASE;

The [:=] tells the controller to also clear the outlet tags whenever the controller does the following:

Enters the RUN mode.

Leaves the step of an SFC if configuring the SFC for Automatic reset. This applies only embedding the assignment in the action of the step or using the action to call a structured text routine via a JSR instruction.

FOR_DO

Use the FOR_DO loop to perform an action a number of times before doing anything else.

When enabled, the FOR instruction repeatedly executes the Routine until the Index value exceeds the Terminal value. The step value can be positive or negative. If it is negative, the loop ends when the index is less than the terminal value.. If it is positive, the loop ends when the index is greater than the terminal value.

Each time the FOR instruction executes the routine, it adds the Step size to the Index.

Do not loop too many times in a single scan. An excessive number of repetitions causes the controller watchdog to timeout and causes a major fault.

Operands

FOR count:= initial_value TO

final_value BY increment DO

<statement>;

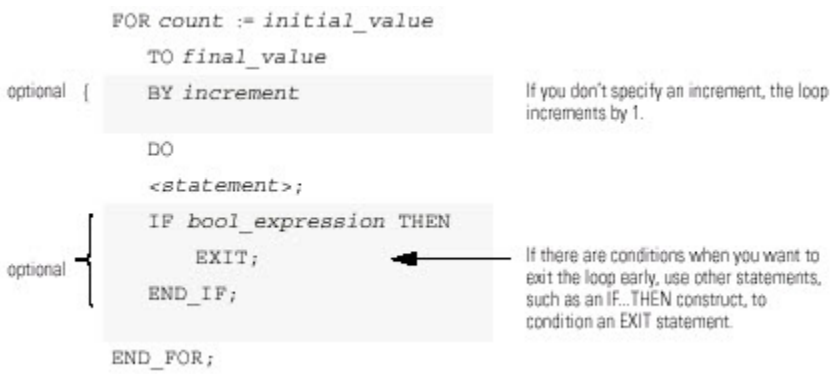
END_FOR;

Operand	Type	Format	Description
count	SINT INT DINT	Tag	Tag to store count position as the FOR_DO executes
initial_ value	SINT INT DINT	Tag expression Immediate	Must evaluate to a number Specifies initial value for count
final_ value	SINT INT DINT	Tag expression Immediate	Specifies final value for count, which determines when to exit the loop
increment	SINT INT DINT	Tag expression Immediate	(Optional) amount to increment count each time through the loop If you don't specify an increment, the count increments by 1.

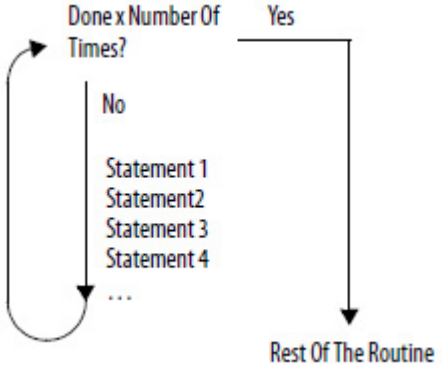
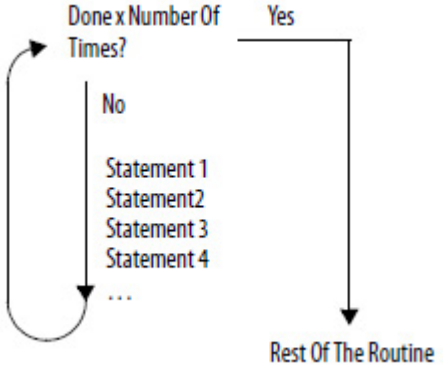
IMPORTANT Do not iterate within the loop too many times in a single scan.
The controller does not execute other statements in the routine until it completes the loop.
A major fault occurs when completing the loop takes longer than the watchdog timer for the task.
Consider using a different construct, such as IF_THEN.

Description

The syntax is described in the table.



This diagrams illustrates how a FOR_DO loop executes, and how an EXIT statement leaves the loop early.

	
The FOR_DO loop executes a specific number of times.	To stop the loop before the count reaches the last value, use an EXIT statement.

Affects Math Status Flags

No

Major/Minor Faults

A major fault will occur if	Fault type	Fault code
The construct loops too long.	6	1

Example 1

If performing the following,	Enter this structured text

Clear bits 0...31 in an array of BOOLs:
Initialize the subscript tag to 0.
Clear i . For example, when subscript = 5, clear array[5].
Add 1 to subscript.
If subscript is ≤ to 31, repeat 2 and 3.
Otherwise, stop.
	For subscript:=0 to 31 by 1 do
	array[subscript] := 0;
	End_for;

Example 2

If performing the following,	Enter this structured text

A user-defined data type (structure) stores the following information about an item in your inventory:
• Barcode ID of the item (String data type)
• Quantity in stock of the item (DINT data type)
An array of the above structure contains an element for each different item in your inventory. You want to search the array for a specific product (use
	SIZE(Inventory,0,Inventory_Items);
	For position:=0 to Inventory_Items - 1 do
	If Barcode = Inventory[position].ID then
	Quantity := Inventory[position].Qty;
	Exit;
	End_if;

its bar code) and determine the quantity that is in stock. 1. Get the size (number of items) of the Inventory array and store the result in 2. Inventory_Items (DINT tag). Initialize the position tag to 0. 3. If Barcode matches the ID of an item in the array, then: Set the Quantity tag = Inventory[position].Qty. This produces the quantity in stock of the item. Stop. Barcode is a string tag that stores the bar code of the item for which you are searching. For example, when position = 5, compare Barcode to Inventory[5].ID. 4. Add 1 to position. 5. If position is ≤ to (Inventory_Items -1), repeat 3 and 4. Since element numbers start at 0, the last element is 1 less than the number of elements in the array. Otherwise, stop.	End_for;

WHILE_DO

Use the WHILE_DO loop to continue performing an action while certain conditions are true.

Operands

WHILE bool_expression DO

<statement>;

Structured Text

Operand	Type	Format	Description
bool_expression	BOOL	tag expression	BOOL tag or expression that evaluates to a BOOL value

IMPORTANT Do not iterate within the loop too many times in a single scan.
The controller does not execute any other statements in the routine until it completes the loop.
A major fault occurs when completing the loop takes longer than the watchdog timer for the task.
Consider using a different construct, such as IF_THEN.

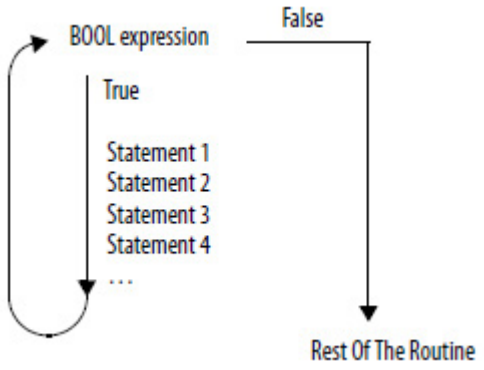
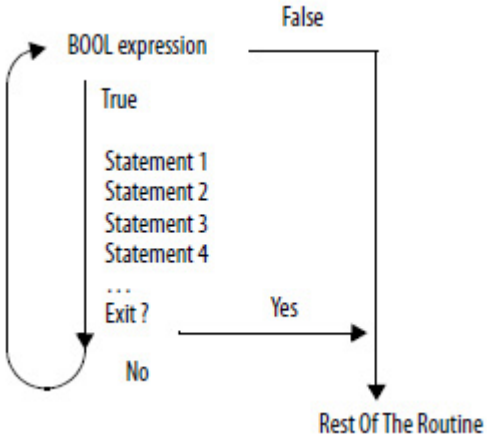
Description

The syntax is:

```
WHILE bool_expression1 DO
   <statement>;          <-- statements to execute while bool_expression1 is true
   IF bool_expression2 THEN
     EXIT;               <-- If there are conditions when you want to exit the loop early, use other statements, such as an IF...THEN construct, to condition an EXIT statement.
   END_IF;
END_WHILE;
```

(IF ... END_IF is optional)

The following diagrams illustrate how a WHILE_DO loop executes, and how an EXIT statement leaves the loop early.

	
While the bool_expression is true, the controller executes only the statements within the WHILE_DO loop.	To stop the loop before the conditions are true, use an EXIT statement.

Affects Math Status Flags

No

Fault Conditions

A major fault will occur if	Fault type	Fault code
the construct loops too long	6	1

Example 1

If performing the following,	Enter this structured text	
The WHILE_DO loop evaluates its conditions first. If the conditions are true, the controller then executes the statements within the loop. This differs from the REPEAT_UNTIL loop because the REPEAT_UNTIL loop executes the statements in the construct and then determines if the	pos := 0;	
	While ((pos <= 100) & structarray[pos].value <> targetvalue)) do	
		pos := pos + 2;
		String_tag.DATA[pos] := SINT_array[pos];

conditions are true before executing the statements again. The statements in a REPEAT_UNTIL loop are always executed at least once. The statements in a WHILE_DO loop might never be executed.	end_while;

Example 2

If performing the following,	Enter this structured text

Move ASCII characters from a SINT array into a string tag. (In a SINT array, each element holds one character.) Stop when you reach the carriage return.
Initialize Element_number to 0.
Count the number of elements in SINT_array (array that contains the ASCII characters) and store the result in SINT_array_size (DINT tag).
If the character at SINT_array[element_number] = 13 (decimal value of the carriage return), then stop.
Set String_tag[element_number] = the character at SINT_array[element_number].
Add 1 to element_number. This lets the controller check the next character in SINT_array.
Set the Length member of String_tag = element_number. (This records the number of characters in String_tag so far.)
If element_number = SINT_array_size, then stop. (You are at the end of the array and it does not contain a carriage return.)
	element_number := 0;
	SIZE(SINT_array, 0, SINT_array_size);
	While SINT_array[element_number] <> 13 do
	String_tag.DATA[element_number] := SINT_array[element_number];
	element_number := element_number + 1;
	String_tag.LEN := element_number;
	If element_number = SINT_array_size then
	exit;
	end_if;
	end_while;

REPEAT_UNTIL

Use the REPEAT_UNTIL loop to continue performing an action until conditions are true.

Operands

REPEAT

<statement>;

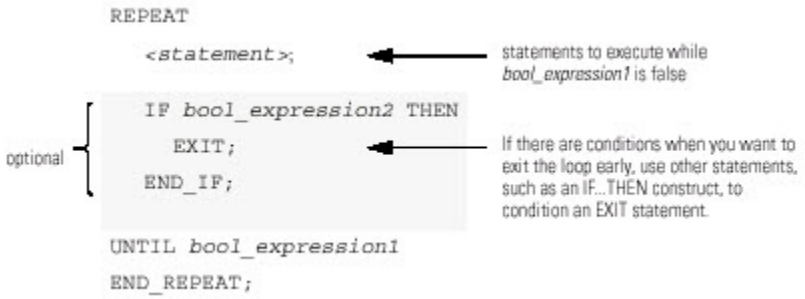
Structured Text

Operand	Type	Format	Enter
bool_ expression	BOOL	Tag expression	BOOL tag or expression that evaluates to a BOOL value (BOOL expression)

IMPORTANT Do not iterate within the loop too many times in a single scan.
The controller does not execute other statements in the routine until it completes the loop.
A major fault occurs when completing the loop takes longer than the watchdog timer for the task.
Consider using a different construct, such as IF_THEN.

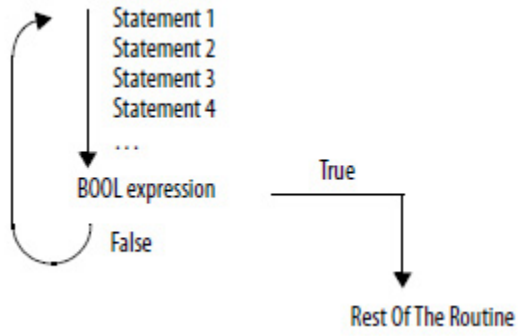
Description

The syntax is:

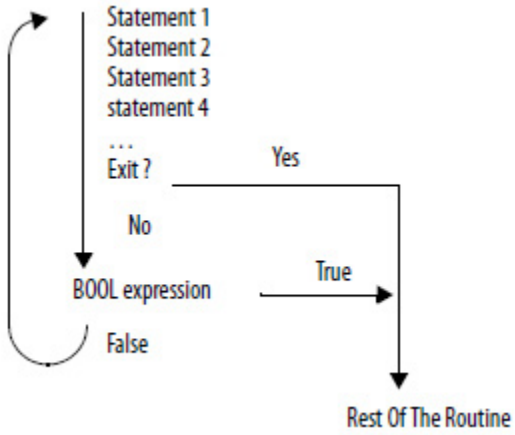


The following diagrams show how a REPEAT_UNTIL loop executes and how an EXIT statement leaves the loop early.

While the bool_expression is false, the controller executes only the statements within the REPEAT_UNTIL loop.



To stop the loop before the conditions are false, use an EXIT statement.



Affects Math Status Flags

No

Fault Conditions

A major fault will occur if	Fault type	Fault code
The construct loops too long	6	1

Example 1

If performing the following,	Enter this structured text

The REPEAT_UNTIL loop executes the statements in the construct and then determines if the conditions are true before executing the statements again. This differs from the WHILE_DO loop because the WHILE_DO The WHILE_DO loop evaluates its conditions first.
If the conditions are true, the controller then executes the statements within the loop. The statements in a REPEAT_UNTIL loop are always executed at least once. The statements in a WHILE_DO loop might never be executed.
	pos := -1;
	REPEAT
	pos := pos + 2;
	UNTIL ((pos = 101) OR (structarray[pos].value = targetvalue))
	end_repeat;

Example 2

If performing the following,	Enter this structured text

Move ASCII characters from a SINT array into a string tag. (In a SINT array, each element holds one character.) Stop when you reach the carriage return.
Initialize Element_number to 0.
Count the number of elements in SINT_array (array that contains the ASCII characters) and store the result in SINT_array_size (DINT tag).
Set String_tag[element_number] = the character at SINT_array[element_number].
	element_number := 0;
	SIZE(SINT_array, 0, SINT_array_size);
	Repeat
	String_tag.DATA[element_number] := SINT_array[element_number];
	element_number := element_number + 1;
	String_tag.LEN := element_number;
	If element_number = SINT_array_size then
	exit;

Add 1 to element_number. This lets the controller check the next character in SINT_array. Set the Length member of String_tag = element_number. (This records the number of characters in String_tag so far.) If element_number = SINT_array_size, then stop. (You are at the end of the array and it does not contain a carriage return.) If the character at SINT_array[element_number] = 13 (decimal value of the carriage return), then stop.	end_if; Until SINT_array[element_number] = 13 end_repeat;

ST Components: Comments

To make your structured text easier to interpret, add comments to it.

- Comments let you use plain language to describe how your structured text works.
- Comments do not affect the execution of the structured text.

To add comments to your structured text:

To add a comment	Use one of these formats
on a single line	//comment (*comment*) /*comment*/
at the end of a line of structured text	
within a line of structured text	(*comment*) /*comment*/
that spans more than one line	(*start of comment. . .end of comment*) /*start of comment. . .end of comment*/

For example:

Format	Example

//comment	**At the beginning of a line**
	//Check conveyor belt direction
	IF conveyor_direction THEN...
	At the end of a line
	ELSE //If conveyor isn't moving, set alarm light
	light := 1;
	END_IF;

(*comment*)	Sugar.Inlet[:=]1;(*open the inlet*) IF Sugar.Low (*low level LS*)& Sugar.High (*high level LS*)THEN... (*Controls the speed of the recirculation pump. The speed depends on the temperature in the tank.*) IF tank.temp > 200 THEN...
/*comment*/	Sugar.Inlet:=0;/*close the inlet*/ IF bar_code=65 /*A*/ THEN... /*Gets the number of elements in the Inventory array and stores the value in the Inventory_Items tag*/ SIZE(Inventory,0,Inventory_Items);

Common attributes for Motion instructions

Follow the guidelines in this chapter for the common attributes for the Motion instructions.

Common Attributes

For more information on attributes that are common to the Logix 5000™ instructions, click any of the topics below.

Math Status Flags on page 700

Immediate Values on page 692

Data Conversions on page 697

Index Through Arrays on page 691

Bit Addressing on page 702

Index Through Arrays

To dynamically change the array element that your logic references, use tag or expression as the subscript to point to the element. This is similar to indirect addressing in PLC-5 logic. Use these operators in an expression to specify an array subscript:

Tip:

- Logix Designer allows subscripts that are extended data type tags only, and does not support subscript expressions that have extended data types.
- All available integer elementary data types can be used as a subscript index. Only use SINT, INT, and DINT tags with operators to create a subscript expression.

Operator	Description
+	add
-	subtract/negate
*	multiply
/	divide
AND	AND
BCD_TO	BCD to integer
NOT	complement
OR	OR
TO_BCD	integer to BCD
SQRT	square root
XOR	exclusive OR

For example:

Definitions	Example	Description
my_list defined as DINT[10]	my_list[5]	This example references element 5 in the array. The reference is static because the subscript value remains constant.
my_list defined as DINT[10] position defined as DINT	MOV the value 5 into position my_list[position]	This example references element 5 in the array. The reference is dynamic because the logic can change the subscript by changing the value of position.
my_list defined as DINT[10] position defined as DINT offset defined as DINT	MOV the value 2 into position MOV the value 5 into offset my_list[position+offset]	This example references element 7 (2+5) in the array. The reference is dynamic because the logic can change the subscript by changing the value of position or offset.

Tip: When entering an array subscript, make sure it is within the boundaries of the specified array. Instructions that view arrays as a collection of elements generate a major fault (type 4, code 20) if a subscript exceeds its corresponding dimension.

Immediate values

When you enter an immediate value (constant) in decimal format (for example, -2, 3) the controller stores the value by using 32 bits. If you enter a value in a radix other than decimal, such as binary or hexadecimal, and do not specify all 32 bits, the controller places a zero in the bits that you do not specify (zero-fill).

IMPORTANT Zero-fill of immediate binary, octal or hexadecimal values less than 32 bits.

If you enter	The controller stores
-1	16#ffff ffff (-1)
16#ffff (-1)	16#0000 ffff (65535)
8#1234 (668)	16#0000 029c (668)
2#1010 (10)	16#0000 000a (10)

Integer Immediate Values

If you enter	The controller stores
Without any suffix	DINT
"U" or "u"	UDINT
"L" or "l"	LINT
"UL", "ul", "Ul", or "uL"	ULINT

Floating Point Immediate Values

If you enter	The controller stores
Without any suffix	REAL
"L" or "l"	LREAL

Floating Point Values

This information applies to the CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, GuardLogix 5570, Compact GuardLogix 5380, CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. Controller differences are noted where applicable.

Logix controllers handle floating point values according to the IEEE 754 standard for floating-point arithmetic. This standard defines how floating point numbers are stored and calculated. The IEEE 754 standard for floating point math was designed to provide speed and the ability to handle very large numbers in a reasonable amount of storage space.

A REAL tag stores a single-precision, normalized floating-point number.

An LREAL tag stores a double-precision, normalized floating-point number.

The controllers support these elementary data types:

Controllers	Data Type
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	REAL, LREAL
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	REAL

Denormalized numbers and -0.0 are treated as 0.0

If a computation results in a NAN value, the sign bit could be positive or negative. In this situation, the software displays 1#.NAN with no sign.

Not all decimal values can be exactly represented in this standard format, which results in a loss of precision. For example, if you subtract 10 from 10.1, you expect the result to be 0.1. In a Logix controller, the result could very well be 0.10000038. In this example, the difference between 0.1 and 0.10000038 is .000038%, or practically zero. For most operations, this small inaccuracy is insignificant. To put things in perspective, if you were sending a floating point value to an analog output module, there would be no difference in the output voltage for a value being sent to the module that differs by .000038%.

Guidelines for Floating-point Math Operations

Follow these guidelines:

When performing certain floating-point math operations, there may be a loss of precision due to rounding error. Floating-point processors have their own internal precision that can impact resultant values.

Do not use floating point math for money values or for totalizer functions. Use INT or DINT values, scale the values up, and keep track of the decimal place (or use one INT or DINT value for dollars, and a second INT or DINT value for cents).

Do not compare floating-point numbers. Instead, check for values within a range. The LIM instruction is provided specifically for this purpose.

Totalizer Examples

The precision of the REAL data type affects totalization applications such that errors occur when adding very small numbers to very large numbers.

For example, add 1 to a number over a period of time. At some point the add will no longer affect the result because the running sum is much greater than 1, and there are not enough bits to store the entire result. The add stores as many upper bits as possible and discards the remaining lower bits.

To work around this, do math on small numbers until the results get large. Then, transfer them to another location for additional large-number math. For example:

- x is the small incremented variable.
- y is the large incremented variable.
- z is the total current count that can be used anywhere.
- x = x+1;
- if x = 100,000;
- {
- y = y + 100,000;
- x = 0;
- }
- z = y + x;

Or another example:

- x = x + some_tiny_number;
- if (x >= 100)
- {
- z = z + 100;
- x = x - 100; // there might be a tiny remainder
- }

Elementary data types

The controller supports the elementary data types defined in IEC 1131-3 defined data types. The elementary data types are:

Data type	Description	Range
BOOL	1-bit boolean	0 = cleared 1 = set
SINT	1-byte integer	-128 to 127
INT	2-byte integer	-32,768 to 32,767
DINT	4-byte integer	-2,147,483,648 to 2,147,483,647

LINT	8-byte integer **Note:** The LINT data type has limited use on CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers. They can be used only with copy (COP, CPS) instructions, the CST/WallClock Time attribute, time synchronization, and Add-On Instructions.	-9223372036854775808 to 9223372036854775807
DT	Date and time. 64-bit storage; units are in microseconds.	DT#1970-01-01-00:00:00.000_000(UTC+00:00) to DT#2250-12-31-23:59:59.999_999(UTC+00:00) **Tip:** In some time zones, such as Singapore, this is the range: DT#1970-01-01-08:00:00.000_000(UTC+08:00) to DT#2251-01-01-07:59:59.999_999(UTC+08:00)
LDT	Long date and time. 64-bit storage; units are in nanoseconds.	LDT#1970-01-01-00:00:00.000_000(UTC+00:00) to LDT#2250-12-31-23:59:59.999_999_999(UTC+00:00) **Tip:** In some time zones, such as Singapore, this is the range: LDT#1970-01-01-08:00:00.000_000_000(UTC+08:00) to LDT#2251-01-01-07:59:59.999_999_999(UTC+08:00)
TIME32	Duration of time. 32-bit storage; units are in microseconds.	T32#-35m_47s_483ms_647us to T32#35m_47s_483ms_647us
TIME	Duration of time. 64-bit storage; units are in microseconds.	T#-106751990d_12h_59m_59s_999ms_999us to T#106751990d_12h_59m_59s_999ms_999us
LTIME	Long duration of time. 64-bit storage; units are in nanoseconds.	LT#-106750d_12h_59m_59s_999ms_999us_999ns to LT#106750d_12h_59m_59s_999ms_999us_999ns
USINT	1-byte unsigned integer	0 to 255
UINT	2-byte unsigned integer	0 to 65,535
UDINT	4-byte unsigned integer	0 to 4,294,967,295
ULINT	8-byte unsigned integer	0 to 18,446,744,073,709,551,615
REAL	4-byte floating-point number	-3.4028235E38 to -1.1754944E-38 (negative values) and 0.0 and 1.1754944E-38 to 3.4028235E38 (positive values)
LREAL	8-byte floating-point number	-1.7976931348623157E308 to -2.2250738585072014E-308 (negative values) and 0.0 and 2.2250738585072014E-308 to 1.7976931348623157E308 (positive values)

These controllers support the following elementary data types:

Controllers	Data type
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	SINT, INT, DINT, LINT, REAL USINT, UINT, UDINT, ULINT, LREAL
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	SINT, INT, DINT, LINT, REAL.

The controller handles all immediate values as DINT data types.

The REAL data type also stores ± infinity and ± NAN, but the software display differs based on the display format.

Data type conversions

When data types are mixed for operands within an instruction, some instructions automatically convert data to an optimal data type for that instruction. In some cases, the controller converts data to fit a new data type; in some cases, the controller just fits the data as best it can.

Conversion	**Result**		
larger integer to smaller integer	The controller truncates the upper portion of the larger integer and generates an overflow. For example:		
	Decimal		**Binary**
	DINT	65,665	0000_0000_0000_0001_0000_0000_1000_0001
	INT	129	0000_0000_1000_0001
	SINT	-127	1000_0001
SINT or INT to REAL	No data precision is lost		
DINT to REAL	Data precision could be lost. Both data types store data in 32 bits, but the REAL type uses some of its 32 bits to store the exponent value. If precision is lost, the controller takes it from the least-significant portion of the DINT.		
LREAL to LREAL	No data precision is lost.		
LREAL TO REAL	Data precision could be lost.		
LREAL/REAL to unsigned integer	Data precision could be lost. If the source value is too big to fit into destination the controller stores what it can and may produce an overflow.		
Signed Integer/Unsigned Integer to LREAL/REAL	If the integer value has more significant bits than can be stored in the destination, the lower bits will be truncated.		
Signed integer to unsigned integer	If the source value is too big to fit into destination, the controller stores what it can and may produce an overflow.		
Unsigned integer to signed integer	If the source value is too big to fit into destination, the controller stores what it can and may produce an overflow.		
REAL to integer	The controller rounds the fractional part and truncates the upper portion of the non-fractional part. If data is lost, the controller sets the overflow status flag. Rounding is to the nearest whole number: less than 0.5, round down; equal to 0.5, round to nearest even integer; greater than 0.5, round up For example:		
	REAL (source)	**DINT (result)**	
	1.6	2	
	-1.6	-2	
	1.5	2	
	-1.5	-2	
	1.4	1	
	-1.4	-1	
	2.5	2	

	-2.5	-2

Do not convert data to or from the BOOL data type.

IMPORTANT The math status flags are set based on the value being stored. Instructions that normally do not affect math status keywords might appear to do so if type conversion occurs because of mixed data types for the instruction parameters. The type conversion process sets the math status keywords.

Safety Data Types

The Logix Designer application prevents the modification of a User Defined or Add-On Defined type that would cause an invalid data type for User Defined or Add-On Defined types that are referenced directly or indirectly by a Safety tag. (This includes nested structures.)

Safety tags can be composed of the following data types:

- All elementary data types.
- Predefined types that are used for safety application instructions.
- User-defined data types or arrays that are composed of the previous two types.

Online edits of user-defined data type member names in safety tags

Online editing is allowed for member names of user-defined data types on CompactLogix 5380, Compact GuardLogix 5380, CompactLogix 5480, ControlLogix 5580, and GuardLogix 5580 controllers. However, online editing is disabled when a user-defined data type is used on a safety tag and the controller is in the Safety Secured state.

See also

Math Status Flags on page 700

Data Conversions

Data conversions occur when mixing data types in programming.

When programming:	Conversions can occur when you:
Ladder Diagram Structured Text	Mix data types for the parameters within one Instruction or expression.
Function Block	Wire two parameters that have different data types

Instructions execute faster and require less memory if all the operands of the instruction use:

- The same data type.
- An intermediate data type:
  - All function block instructions support one data type operand only.

- If mixing data types or use tags that are not the optimal data type, the controller converts the data according to these rules:
  - Operands are converted according to the ranking of data types from SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, and LREAL with ranking from 1 (the lowest) to 10 (the highest).

Tip: To reduce the time and memory for converting data, use the same data type for all the operands of an instruction.

Convert SINT or INT to DINT or DINT to LINT

A SINT or INT input source tag gets promoted to a DINT value by a sign-extension for Source Tag. Instructions that convert SINT or INT values to DINT values use one of the following conversion methods.

This conversion method	Converts data by placing
Sign-extension	The value of the leftmost bit (the sign of the value) into each bit position to the left of the existing bits until there are 32 or 64 bits.
Zero-fill	Zeroes to the left of the existing bits until there are 32 or 64 bits.

Logical instructions use zero fill. All other instructions use sign-extension

The following example shows the results of converting a value using sign-extension and zero-fill.

This value	2#1111_1111_1111_1111	(-1)
Converts to this value by sign-extension	2#1111_1111_1111_1111_1111_1111_1111_1111	(-1)
Converts to this value by zero-fill	2#0000_0000_0000_0000_1111_1111_1111_1111	(65535)

If you use a SINT or INT tag and an immediate value in an instruction that converts data by sign-extension, use one of these methods to handle immediate values.

Specify any immediate value in the decimal radix.

If you enter the value in a radix other than decimal, specify all 32 bits of the immediate value. To do so, enter the value of the leftmost bit into each bit position to its left until there are 32 bits.

Create a tag for each operand and use the same data type throughout the instruction. To assign a constant value, either:

Enter it into one of the tags.

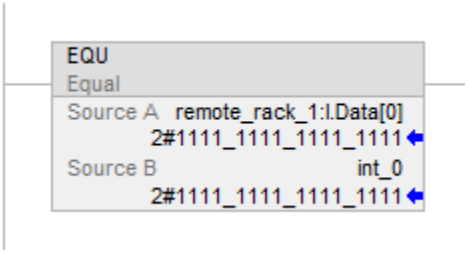
Add a MOV instruction that moves the value into one of the tags.

Use a MEQ instruction to check only the required bits.

The following examples show two ways to mix an immediate value with an INT tag. Both examples check the bits of a 1771 I/O module to determine if all the bits are on. Since the input data word of a 1771 I/O module is an INT tag, it is easiest to use a 16-bit constant value.

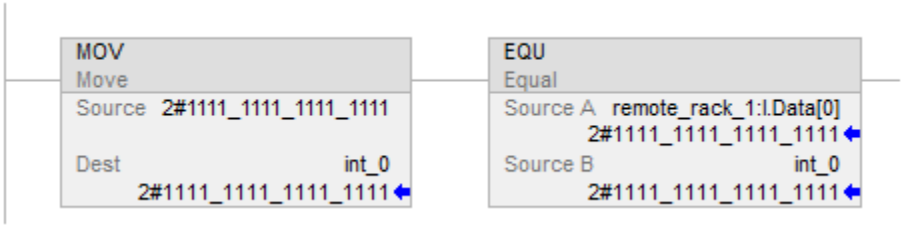
IMPORTANT Mixing an INT tag with an immediate value

Since remote_rack_1:I.Data[0] is an INT tag, the value to check it against is also entered as an INT tag.



IMPORTANT Mixing an INT tag with an immediate value

Since remote_rack_1:I.Data[0] is an INT tag, the value to check it against first moves into int_0, also an INT tag. The EQU instruction then compares both tags.



Convert Integer to REAL

The controller stores REAL values in IEEE single-precision, floating-point number format. It uses one bit for the sign of the value, 23 bits for the base value, and eight bits for the exponent (32 bits total). If you mix an integer tag (SINT, INT, or DINT) and a REAL tag as inputs in the same instruction, the controller converts the integer value to a REAL value before the instruction executes.

- A SINT or INT value always converts to the same REAL value.
- A DINT value may not convert to the same REAL value:
- A REAL value uses up to 24 bits for the base value (23 stored bits plus a 'hidden' bit).
- A DINT value uses up to 32 bits for the value (one for the sign and 31 for the value).

If the DINT value requires more than 24 significant bits, it might not convert to the same REAL value. If it will not, the controller stores the uppermost 24 bits rounded to the nearest even value.

Convert DINT to SINT or INT

To convert a DINT value to a SINT or INT value, the controller truncates the upper portion of the DINT and stores the lower bits that fit in the data type. If the value is too large the conversion generates an overflow.

	Convert a DINT to an INT and a SINT	
This DINT value	Converts to this smaller value	
16#0001_0081 (65,665)	INT:	16#0081 (129)
	SINT:	16#81 (-127)

Convert REAL to SINT, INT, or DINT

To convert a REAL value to an integer value, the controller rounds any fractional part and stores the bits that fit in the result data type. If the value is too large the conversion generates an overflow.

Numbers round as in the following examples.

Fractions < 0.5 round down to the nearest whole number.

Fractions > 0.5 round up to the nearest whole number.

Fractions = 0.5 round up or down to the nearest even number.

Important:	Conversion of REAL values to DINT values
This REAL value	**Converts to this DINT value**
-2.5 -3.5	-2 -4
-1.6	-2
-1.5	-2
-1.4	-1
1.4	1
1.5	2
1.6	2
2.5 3.5	2 4

Math Status Flags

Follow the guidelines in this topic for Math Status Flags.

Description

Controllers	**Description**
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	A set of Math Status Flags for accessing directly with instructions. These flags are only updated in ladder diagram routines, and are not tags, and flag aliases are not applicable.

Controllers	Description
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	A set of Math Status Flags for accessing directly with instructions. These flags are updated in all routine types, but are not tags, and flag aliases are not applicable.

Status Flags

Status Flag	Description (For CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers)	Description (For CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers)
S:FS First scan flag	The first scan flag is set by the controller: • The first time a program is scanned after the controller goes to Run mode • The first time a program is scanned after the program is uninhibited • When a routine is called from an SFC Action and the step that owns that Action is first scanned. Use the first scan flag to initialize data for use in later scans. It is also referred to as the first pass bit.	The first scan flag is set by the controller: • The first time a program is scanned after the controller goes to Run mode • The first time a program is scanned after the program is uninhibited • When a routine is called from an SFC Action and the Step that owns that Action is first scanned. Use this flag to initialize data for use in later scans. It is also referred to as the first pass bit.
S:N Negative flag	The controller sets the negative flag when the result of a math or logical operation is a negative value. Use this flag as a quick test for a negative value.	The controller sets the negative flag when the result of a math or logical operation is a negative value. Use this flag as a quick test for a negative value. Using S:N is more efficient than using the CMP instruction.
S:Z Zero flag	The zero flag is set by the controller when the result of a math or logical operation is zero. Use this flag as a quick test for a zero value. The zero flag clears at the start of executing an instruction capable of setting this flag.	The controller sets the zero flag when the result of a math or logical operation is zero. Use this flag as a quick test for a zero value.
S:V Overflow flag	The controller sets the overflow flag when: • The result of a math operation results in an overflow. For example, adding 1 to a SINT generates an overflow when the value goes from 127 through -128. • The destination tag is too small to hold the value. For example, if you try to store the value 123456 to a SINT or INT tag. Use the overflow flag to verify the result of an operation is still in range. If the data being stored is a string type, S:V is set if the string is too large to fit into the destination tag. **Tip:** If applicable, set S:V with an OTE or OTL instruction. Click **Controller Properties > Advanced tab > Report Overflow Faults** to enable or disable reporting overflow faults. If an overflow occurs while evaluating an array subscript, a minor fault is generated and a major fault is generated to indicate the index is out of range.	The controller sets the overflow flag when: • The result of a math operation results in an overflow. For example, adding 1 to a SINT generates an overflow when the value goes from 127...-128. • The destination tag is too small to hold the value. For example, if you try to store the value 123456 to a SINT or INT tag. Use the overflow flag to check that the result of an operation is still in range. A minor fault is generated anytime an overflow flag is set. **Tip:** If applicable, set S:V with an OTE or OTL instruction.

Status Flag	Description (For CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers)	Description (For CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers)
S:C Carry flag	The controller sets the carry flag when the result of a math operation resulted in the generation of a carry out of the most significant bit. Only the ADD and SUB instructions, and not the + and – operators, with integer values affect this flag.	The controller sets the carry flag when the result of a math operation resulted in the generation of a carry out of the most significant bit.
S:MINOR Minor fault flag	The controller sets the minor fault flag when there is at least one minor program fault. Use the minor fault tag to test if a minor fault occurred. This bit only triggers by programming faults, such as overflow. It is not triggered by a battery fault. The bit clears at the beginning of every scan. **Tip:** If applicable, explicitly set S:MINOR with an OTE or OTL instruction.	The controller sets the minor fault flag when there is at least one minor program fault. Use the minor fault flag to test if a minor fault occurred and take appropriate action. This bit is triggered only by programming faults, such as overflow. It is not triggered by a battery fault. The bit clears at the beginning of every scan. **Tip:** If applicable, explicitly set S:MINOR with an OTE or OTL instruction.
S:R Redundancy status flag	The S:R flag is set by the controller when the controller redundancy state is Synchronized Secondary.	Not supported on CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers.
IMPORTANT	The math status flags are set based on the stored value. Instructions that normally do not affect math status flags might appear to affect math status flags if type conversion occurs from mixed data types for the instruction parameters. The type conversion process sets the math status flags.	

Expressions in Array Subscripts

Controllers	Description
CompactLogix 5380, CompactLogix 5480, ControlLogix 5580, Compact GuardLogix 5380, and GuardLogix 5580 controllers	Expressions do not set status flags based on the results of math operations. If expressions overflow: • A minor fault generates if the controller is configured to generate minor faults. • A major fault (type 4, code 20) generates because the resulting value is out of range.
CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers	Expressions set status flags based on the results of math operations. If an array subscript is an expression, the expression and the instruction could generate minor faults.

Tip: If an array subscript is too large (out of range), a major fault (type 4, code 20) generates.

Bit Addressing

Bit addressing is used access a particular bit within a larger container. Larger containers include any integer, structure or BOOL array. For example:

Definition	Example	Description

Definition	Example	Description
Variable0 defined as LINT has 64 bits	variable0.42	This example references the bit 42 of variable0.
variable1 defined as DINT has 32 bits	variable1.2	This example references the bit 2 of variable1.
variable2 defined as INT has 16 bits	variable2.15	This example references the bit 15 of variable2.
variable3 defined as SINT holds 8 bits	variable3.[4]	This example references bit 4 of variable3.
variable4 defined as COUNTER structure has 5 status bits	variable4.DN	This example references the DN bit of variable4.
MyVariable defined as BOOL[100] MyIndex defined as SINT	MyVariable[(MyIndex AND NOT 7) / 8].[MyIndex AND 7]	This example references a bit within a BOOL array.
MyArray defined as BOOL[20]	MyArray[3]	This example references the bit 3 of MyArray.
variable5 defined as ULINT holds 64 bits	variable5.53	This example references the bit 53 of variable5.

Use Bit Addressing anywhere a BOOL typed tag is allowed.

See also

Index Through Arrays on page 691

Index

C

common attributes 671
Logix instructions 671

M

MAAT 302
MAFR 25
MAHD 319
MAOC 261
MAR 248
MASD 30
MASR 35
MAW 238
MCCD 388
MCCM 411
MCLM 434
MCPM 374
MCSD 454
MCSR 458
MCT 471
MCTO 363
MCTP 356
MCTPO 401
MDCC 350
MDF 40
MDO 44
MDOC 293
MDR 257
MDS 50
MDW 244
MGS 216
MGSD 223
MGSR 227
motion change dynamics MCD 121
motion apply axis tuning MAAT 302
motion apply hookup diagnostics MAHD 319
motion arm output cam MAOC 261
motion arm registration MAR 248
motion arm watch MAW 238
motion axis fault reset MAFR 25
motion axis gear MAG 110
motion axis home MAH 80
motion axis jog MAJ 86
motion axis move MAM 96
motion axis position cam MAPC 150
motion axis shutdown MASD 30
motion axis shutdown reset MASR 35
motion axis stop MAS 70
motion axis time cam MATC 184
motion calculate cam profile MCCP 139
motion calculate slave values MCSV 147
motion direct drive off MDF 40
motion direct drive on MDO 44
motion disarm output cam MDOC 293
motion disarm registration MDR 257
motion disarm watch MDW 244
motion drive start MDS 50
motion group shutdown MGSD 223
motion group shutdown reset MGSR 227
motion group stop MGS 216
motion group strobe position MGSP 231
motion redefine position MRP 132
motion run axis tuning MRAT 309
motion run hookup diagnostics MRHD 325
motion servo off MSF 57
motion servo on MSO 63
MRHD 325
MSF 57
MSO 63

O

OUTPUT_CAM structure 644

Rockwell Automation support

Use these resources to access support information.

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

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)

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

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

Allen-Bradley, expanding human possibility, Logix, Rockwell Automation, and Rockwell Software are trademarks of Rockwell Automation, Inc.

EtherNet/IP is a trademark of ODVA, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

Rockwell Otomayson Ticaret A.Ş. Kar Plaza İş Merkezi E Blok Kat:6 34752, İçerenköy, İstanbul, Tel: +90 (216) 5698400 EEE YÖnetmeliğine Uygundur

Connect with us.

AMERICAS: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
EUROPE/MIDDLE EAST/AFRICA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
ASIA PACIFIC: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Rockwell Automation Publication MOTION-RM002M-EN-P-November 2023
Supersedes MOTION-RM002L-EN-P-November 2022

Copyright © 2023 Rockwell Automation Technologies, Inc. All Rights Reserved. Printed in the U.S.A.