Allen-Bradley

Logix5550 Controller

(Cat. No. 1756-L1, -L1M*x*)

Motion Instruction Set Reference Manual

Introduction

This release of this document contains new and updated information.

Updated Information

This document has been updated throughout. The most significant changes are:

- New Motion Calculate Cam Profile (MCCP) Motion Instruction
- New Motion Axis Position Cam (MAPC) Motion Instruction
- New Motion Axis Time Cam (MATC) Motion Instruction
- Added new error codes and structures to Appendix A

Notes:

Using This Manual

Introduction

This manual is one of several ControlLogix documents.

Task/Goal:	Documents:
Installing the controller and its components	*Logix5550 Controller Quick Start*, publication 1756-10.1 *Logix5550 Memory Board Installation Instructions*, publication 1756-5.33
Using the controller	*Logix5550 Controller User Manual*, publication 1756-6.5.12
Programming a sequential application	*Logix5550 Controller Instruction Set Reference Manual*, publication 1756-6.4.1
Programming a motion application **You are here** 	*Logix5550 Controller Motion Instruction Set Reference Manual*, publication 1756-6.4.3
Configuring and communicating with digital I/O modules	*Digital Modules User Manual*, publication 1756-6.5.8
Configuring analog I/O modules	*Analog Modules User Manual*, publication 1756-6.5.9
Configuring and using motion modules	*ControlLogix Motion Module Setup and Configuration Manual*, publication 1756-6.5.16
Detailed information for programming with motion	*ControlLogix Motion Module Programming Manual*, publication 1756-5.72
Selecting and installing a chassis	*ControlLogix Chassis Installation Instructions*, publication 1756-5.69
Selecting and installing a power supply	*ControlLogix Power Supply Installation Instructions*, publication 1756-5.1

Who Should Use This Manual

This document provides a programmer with details about the motion instructions that are available for a Logix5550 controller. You should already be familiar with how the Logix5550 controller stores and processes data.

Novice programmers should read all the details about an instruction before using the instruction. Experienced programmers can refer to the instruction information to verify details.

Purpose of This Manual

This manual provides information about each motion instruction that the Logix5550 controller supports. Each description follows this format.

This section:	Provides this type of information:
Instruction name	identifies the instruction defines whether the instruction is an input or an output instruction
Description	describes the instruction's use defines any differences when the instruction is enabled and disabled, if appropriate
Operands	lists all the operands of the instruction
Control structure	lists control status bits and values, if any, of the instruction
Execution	defines the specifics of how the instruction operates during: • prescan • rung-condition-in is false • rung-condition-in is true
Arithmetic status flags	defines whether or not the instruction affects arithmetic status flags see appendix A
Fault conditions	defines whether or not the instruction generates minor or major faults if so, defines the fault type and code
Error Codes	lists and defines the applicable error codes
Status Bits	lists affected status bits, their states, and definitions.
Example	provides at least one programming example includes a description explaining each example

Conventions and Related Terms

Set and clear

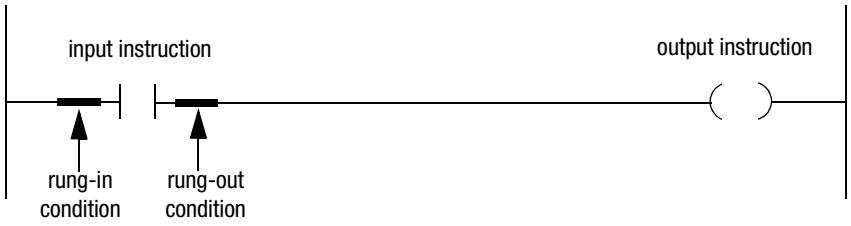
This manual uses set and clear to define the status of bits (booleans) and values (non-booleans):

This term:	Means:
set	the bit is set to 1 (ON) a value is set to any non-zero number
clear	the bit is cleared to 0 (OFF) all the bits in a value are cleared to 0

An instruction executes faster and requires less memory if all the operands of the instruction use the same optimal data type, typically DINT or REAL.

Rung condition

The controller evaluates ladder instructions based on the rung condition preceding the instruction (rung-condition-in). Based on the rung-condition-in and the instruction, the controller sets the rung condition following the instruction (rung-condition-out), which in turn, affects any subsequent instruction.



If the rung-in condition to an input instruction is true, the controller evaluates the instruction and sets the rung-out condition based on the results of the instruction. If the instruction evaluates to true, the rung-out condition is true; if the instruction evaluates to false, the rung-out condition is false.

Notes:

Motion Concepts

Introduction

This chapter covers concepts that are common to all the motion instructions. This chapter includes:

The motion instruction set consists of five groups of instructions:

Group:	For more information, see:
Motion state instructions	Chapter 2
Motion move instructions	Chapter 3
Motion group instructions	Chapter 4
Motion event instructions	Chapter 5
Motion configuration instructions	Chapter 6

These instructions operate on one or more axes. You must identify and configure axes before you can use them. For more information about configuring axes, refer to the *ControlLogix Motion Module Setup and Configuration Manual*, publication 1756-6.5.16.

Using Motion Parameters

Understanding motion status and configuration parameters

You can read motion status and configuration parameters in your ladder logic program using two methods.

Method	Example	For more information
Directly accessing the MOTION_GROUP and AXIS structures	• axis faults • motion status • servo status	See appendix A structures
Using the GSV instruction	• actual position • command position • actual velocity	See *Logix5550 Controller Instruction Set Reference Manual.* (Publication 1756-6.4.1)

Modifying 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.

For more information about the SSV instruction, see the *Logix5550 Controller Instruction Set Reference Manual,* publication 1756-6.4.1.

For more information about motion instructions and creating application programs, see the *ControlLogix Motion Module Setup and Configuration Manual*, publication 1756-6.5.16.

Understanding Instruction Timing

Motion instructions use three types of timing sequences:

Timing type:	Description:	See page:
Immediate	The instruction completes in one scan.	1-3
Message	The instruction completes over several scans because the instruction sends messages to the servo module.	1-4
Process	The instruction could take an indefinite amount of time to complete.	1-6

Immediate type instructions

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.

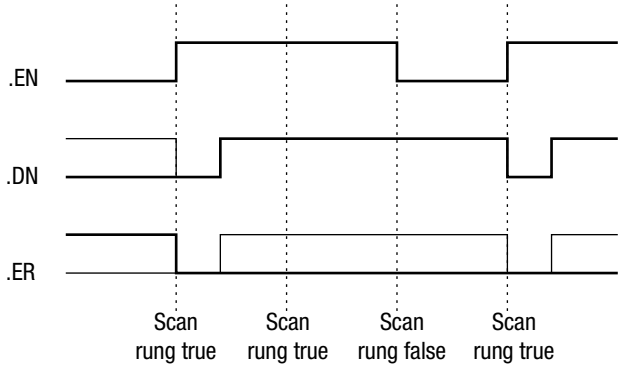
   The controller executes the instruction completely.

2.

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 instructions

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 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 begins to execute the instruction by setting up a message request to the servo module.

   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.

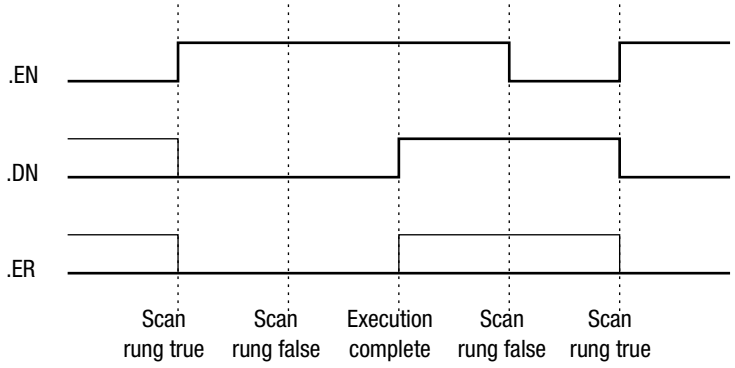
   This process may repeat several times if the instruction requires multiple messages.

6.

If the controller:	Then:
Does not detect an error when the instruction executes	The controller sets the .DN bit if all messaging to the module is completed.
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.



41385

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.
3.

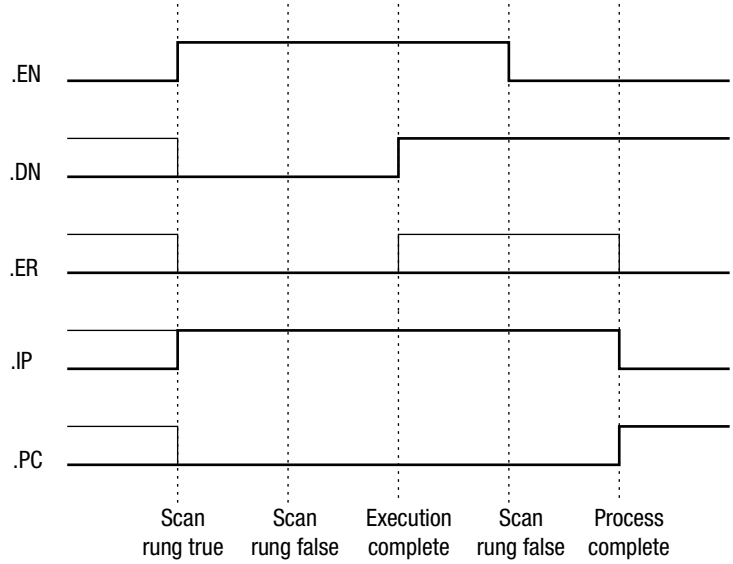
If:	Then the controller:
The controller does not detect an error when the instruction executes	• Sets the .DN bit. • Sets the in process (.IP) bit.
The controller detects an error when the instruction executes	• Sets the .ER bit. • Stores an error code in the control structure.
The controller detects 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. Note: 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	Clears the .IP bit.

4. After the initiation of the motion process, the program scan can continue.

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

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

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



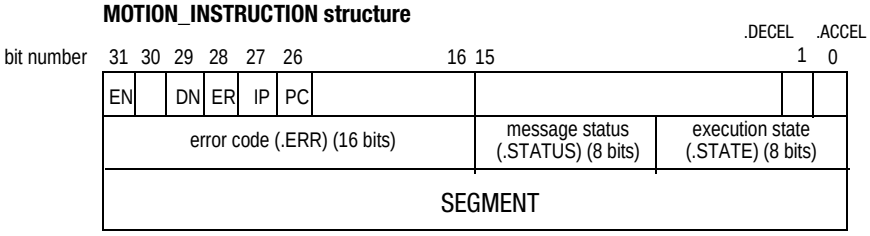
41385

Using the Motion Instruction Structure

The controller uses the MOTION_INSTRUCTION structure to store status information during the execution of motion instructions. Every motion instruction has an operand that requires a motion instruction structure. For each motion instruction you use, define a unique motion instruction structure.

The structure of the motion instruction structure is shown below:

MOTION_INSTRUCTION structure



Mnemonic:	Data Type:	Description:
.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.
.IP	BOOL	The in process bit indicates that a process is being executed.
.PC	BOOL	The process complete bit indicates that the operation is complete. Note: The .DN bit sets after an instruction has completed execution. The .PC bit sets when the initiated process has completed.
.ACCEL	BOOL	The .ACCEL bit indicates that the velocity has increased for the individual instruction that it is tied to i.e jog, move, gearing
.DECEL	BOOL	The .DECEL bit indicates that the velocity has decreased for the individual instruction that it is tied to i.e jog, move, gearing.
.ERR	INT	The error value contains the error code associated with a motion function. See page 1-8.
.STATUS	SINT	The message status value indicates the status condition of any message associated with the motion function. See page 1-10.
.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. See page 1-10.
.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.

Error codes (.ERR)

Error Code:	Error Message	Description:
3	Execution Collision	The instruction tried to execute while another instance of this instruction was executing. This can occur when the controller executes a messaging instruction without checking the .DN bit of the preceding instruction.
4	Servo On State Error	The instruction tried to execute on an axis with a closed servo loop.
5	Servo Off State Error	The instruction tried to execute on an axis with a servo loop that is not closed.
6	Drive On State Error	The axis drive is enabled.
7	Shutdown State Error	The axis is in the shutdown state.
8	Illegal Axis Type	The configured axis type is not correct.
9	Overtravel Condition	The instruction tried to execute in a direction that aggravates the current overtravel condition.
10	Master Axis Conflict	The master axis reference is the same as the slave axis reference.
11	Axis Not Configured	The axis is not configured.
12	Servo Message Failure	Messaging to the servo module failed.
13	Parameter Out Of Range	The instruction tried to use a parameter that is outside the range limit.
14	Tune Process Error	The instruction cannot apply the tuning parameters because of an error in the run tuning instruction.
15	Test Process Error	The instruction cannot apply the diagnostic parameters because of an error in the run diagnostic test instruction.
16	Home In Process Error	The instruction tried to execute with homing in progress.
17	Axis Mode Not Rotary	The instruction tried to execute a rotary move on an axis that is not configured for rotary operation.
18	Axis Type Unused	The axis type is configured as unused.
19	Group Not Synchronized	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.
20	Axis In Faulted State	The axis is in the faulted state.
21	Group In Faulted State	The group is in the faulted state.
22	Axis In Motion	An MSO (Motion Servo On) or MAH (Motion Axis Home) instruction was attempted while the axis was in motion.
23	Illegal Dynamic Change	An instruction attempted an illegal change of dynamics.
24	Illegal AC Mode Op	The controller attempted to execute an MDO, MSO, MAH, MAJ, MAM, MCD, MAPC, MATC, MAG, MRAT, or MRHD instruction when the controller was in the test mode.
25	Illegal Instruction	You attempted to execute an instruction that is not correct.
26	Illegal CAM Length	The cam array is of an illegal length.
27	Illegal CAM Profile Length	The cam profile array is of an illegal length.
28	Illegal CAM Type	You have an illegal segment type in the cam element.
29	Illegal CAM Order	You have an illegal order of cam elements.
30	CAM Profile Being Calculated	You tried to execute a cam profile while it is being calculated.
31	CAM Profile Being Used	The cam profile array you tried to execute is in use.
32	CAM Profile Not Calculated	The cam profile array you tried to execute has not been calculated.

Message status (.STATUS)

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.

Execution status (.STATE)

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.

Profile Segment (.SEGMENT)

A segment is the distance from one point up to but, not including the next point. A .SEGMENT instruction gives the relative position by segment number as the Cam is executing.

Motion State Instructions

(MSO, MSF, MASD, MASR, MDO,MDF, MAFR)

ATTENTION: 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 of the control variables.

Introduction

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:	See page:
Enable the servo drive and activate the axis servo loop.	MSO	2-4
Disable the servo drive and deactivate the axis servo loop.	MSF	2-7
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	2-11
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	2-15
Enable the servo drive and set the servo output voltage of an axis.	MDO	2-18
Deactivate the servo drive and set the servo output voltage to the output offset voltage.	MDF	2-22
Clear all motion faults for an axis.	MAFR	2-25

The five operating states of an 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-stop 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.

To determine the operating state of the axis, you can look at the DRIVE and FDBK LEDs on the servo module.

Important: If the DRIVE and FDBK LEDs are blank, the axis is not configured.

For a servo axis, the following LEDs apply:

If the DRIVE LED is:	And the FDBK LED is:	Then the operating state of the axis is:
Flashing green	Flashing green	Axis ready
Solid green	Flashing green	Direct drive control
Solid green	Solid green	Servo control
Flashing or solid red	Flashing or solid red Flashing or solid green	Axis faulted
Flashing red	Flashing green	Shutdown

For a position only axis, the following LEDs apply:

If the DRIVE LED is:	And the FDBK LED is:	Then the operating state of the axis is:
Blank	Flashing green	Axis ready
Blank	Flashing or solid red	Axis faulted

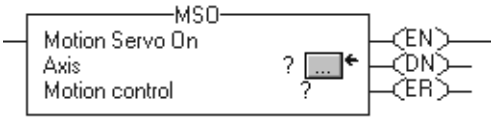
Motion Servo On (MSO)

The MSO instruction is an output instruction.Use the MSO instruction to enable the servo drive and to activate the axis servo loop. A common use for this instruction is activating an axis servo loop in preparation for commanding motion.

The MSO instruction uses message type timing.

To use the MSO instruction, configure the axis as a servo axis.

Operands:

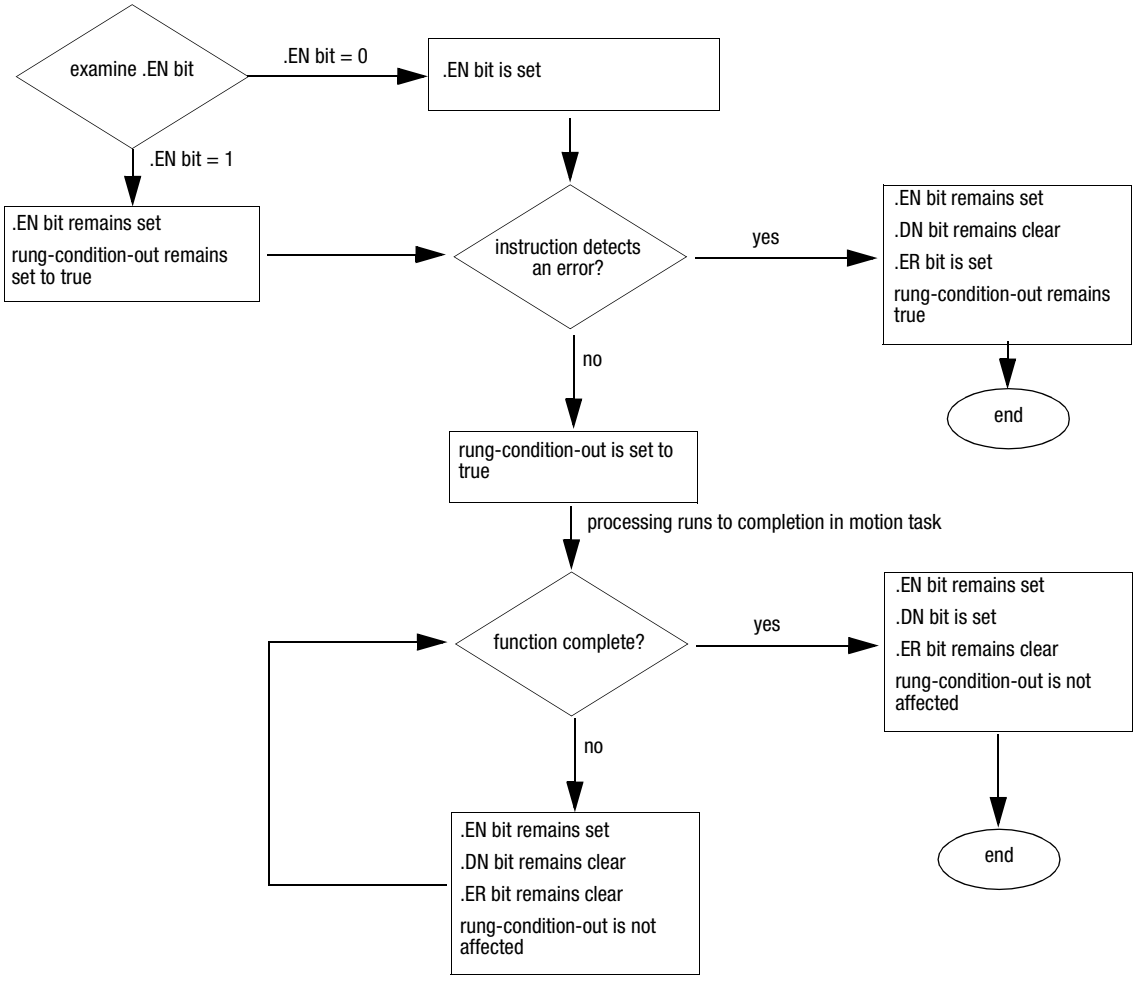


Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction enables axis servo action.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MSO Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
7	The axis is in the shutdown state.
8	The axis is not configured as a servo, position only, or virtual axis.
11	The axis is not configured.
12	Messaging to the servo module failed.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.
20	The axis is in the faulted state.
21	The group is in the faulted state.
22	The instruction was attempted while the axis was in motion.
24	The controller attempted to execute an MDO, MSO, MAH, MAJ, MAM, MCD, MAPC, MATC, MAG, MRAT, or MRHD instruction when the controller was in the test mode.

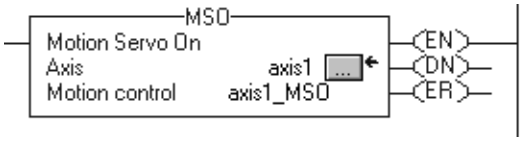
MSO Changes to Servo Module LEDs:

This LED:	Will change to:	Meaning:
FDBK	Solid green	Servo action is enabled.
DRIVE	Solid green	The drive enable output is active.

MSO Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
ServoActStatus	True	• Axis is in the servo on state. • Servo loop is active.
DriveEnableStatus	True	The drive enable output is active.

MSO Example:



When the input conditions are true, the controller enables the servo drive and activates the axis servo loop configured by *axis1*.

Other Formats:

Format:	Syntax:
neutral text	`MSO(axis,motion_control);`
ASCII text	`MSO axis motion_control`

Motion Servo Off (MSF)

The MSF instruction is an output instruction.

Use the MSF instruction to disable the servo drive and to deactivate the axis servo loop.

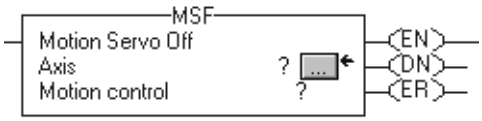
Important: If you execute an MSF instruction while the axis is moving, the axis will coast to an uncontrolled stop.

Use the MSF instruction to disable servo action. Although servo action is disabled, the controller continues to track the axis actual position. When the servo action is enabled using the Motion Servo On (MSO) instruction, the servo maintains the position.

The MSF instruction uses message type timing.

To use the MSF instruction, configure the axis as a servo axis.

Operands:

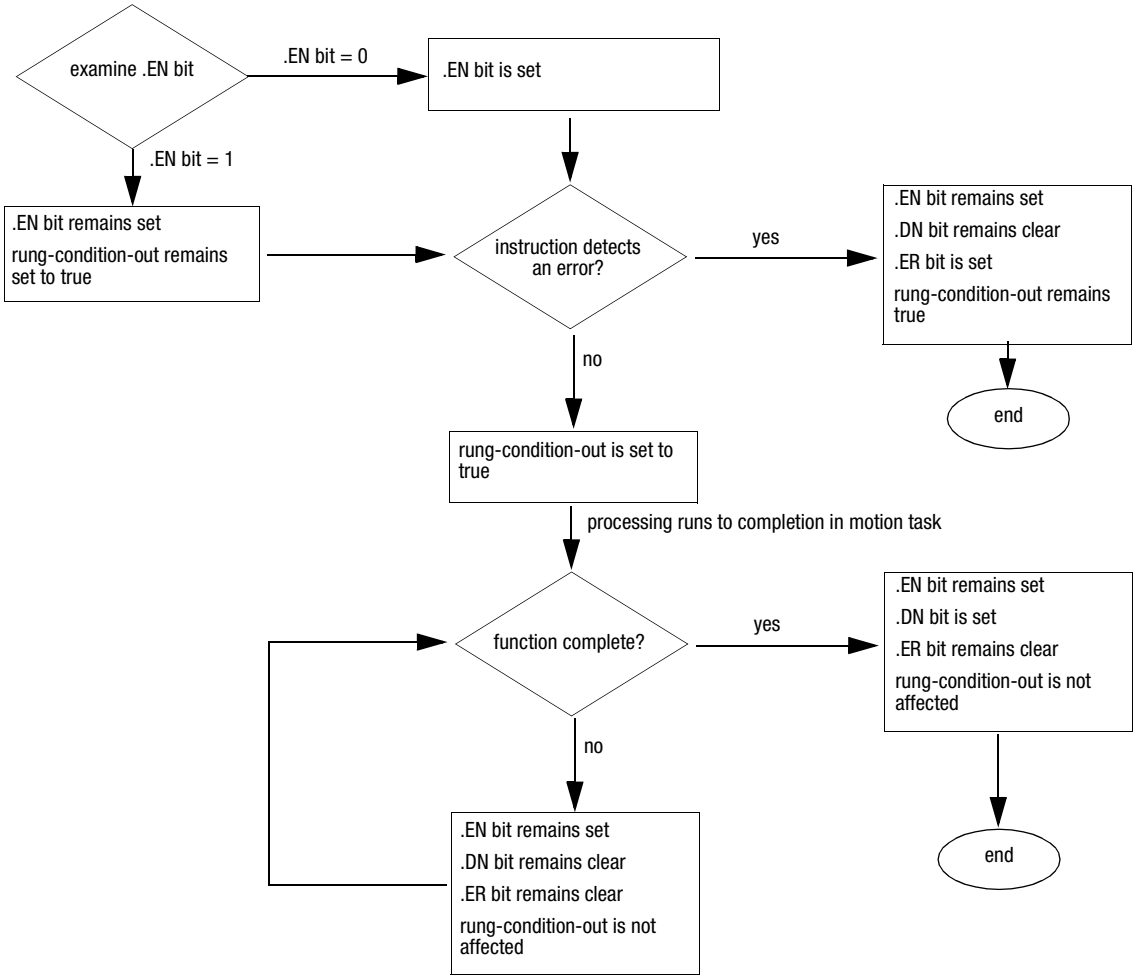


Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction disables axis servo action.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MSF Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
8	The axis is not configured as a servo axis.
11	The axis is not configured.
12	Messaging to the servo module failed.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

MSF Changes to Servo Module LEDs:

This LED:	Will change to:	Meaning:
FDBK	Flashing green	Servo action is disabled.
DRIVE	Flashing green	The drive enable output is inactive.

MSF Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
ServoActStatus	False	• Axis is in the axis ready state. • Servo loop is inactive.
DriveEnableStatus	False	The drive enable output is inactive.
AccelStatus	False	The axis is not accelerating.
StoppingStatus	False	The axis is not stopping.
JogStatus	False	The axis is not jogging.
MoveStatus	False	The axis is not moving.
GearingStatus	False	The axis is not gearing.
HomingStatus	False	The axis is not homing.
GearingLockedStatus	False	The axis is not clutching to a new gear speed.
PositionCamStatus	False	Pcam motion profile is not in progress.
TimeCamStatus	False	Tcam motion profile is not in progress.
PositionCamLockedStatus	False	The Pcam is stopped and the lock is cleared.
TimeCamLockedStatus	False	The Tcam is stopped and the lock is cleared.
TuneStatus	False	The axis is not running a tuning process.
TestStatus	False	The axis is not running a testing process.
PositionCamPendingStatus	False	The pending PCAM profile is cancelled.
TimeCamPendingStatus	False	The pending Tcam profile is cancelled.

MSF Example:

When the input conditions are true, the controller disables the servo drive and the axis servo loop configured by *axis1*.

Other Formats:

Format:	Syntax:
neutral text	`MSF(axis,motion_control);`
ASCII text	`MSF axis motion_control`

Motion Axis Shutdown (MASD)

The MASD instruction is an output instruction.

Use the MASD instruction to 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.

If an axis is in the shutdown operating state, it means:

- The axis servo action is disabled.
- The drive enable output is inactive.
- There is no servo output.
- The servo module OK contacts are open.

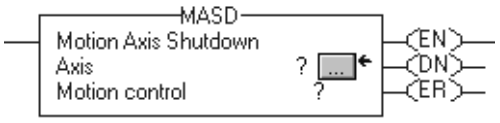
Note: You can use the OK contacts to open the contacts that open the E-stop string (if wired) that controls the power to the drive system.

The axis will remain in the shutdown state until either a Motion Axis Shutdown Reset (MASR) instruction or a Motion Group Shutdown Reset (MGSR) instruction executes.

The MASD instruction uses message type timing.

To use the MASD instruction, configure the axis as either a servo axis or a position only axis.

Operands:

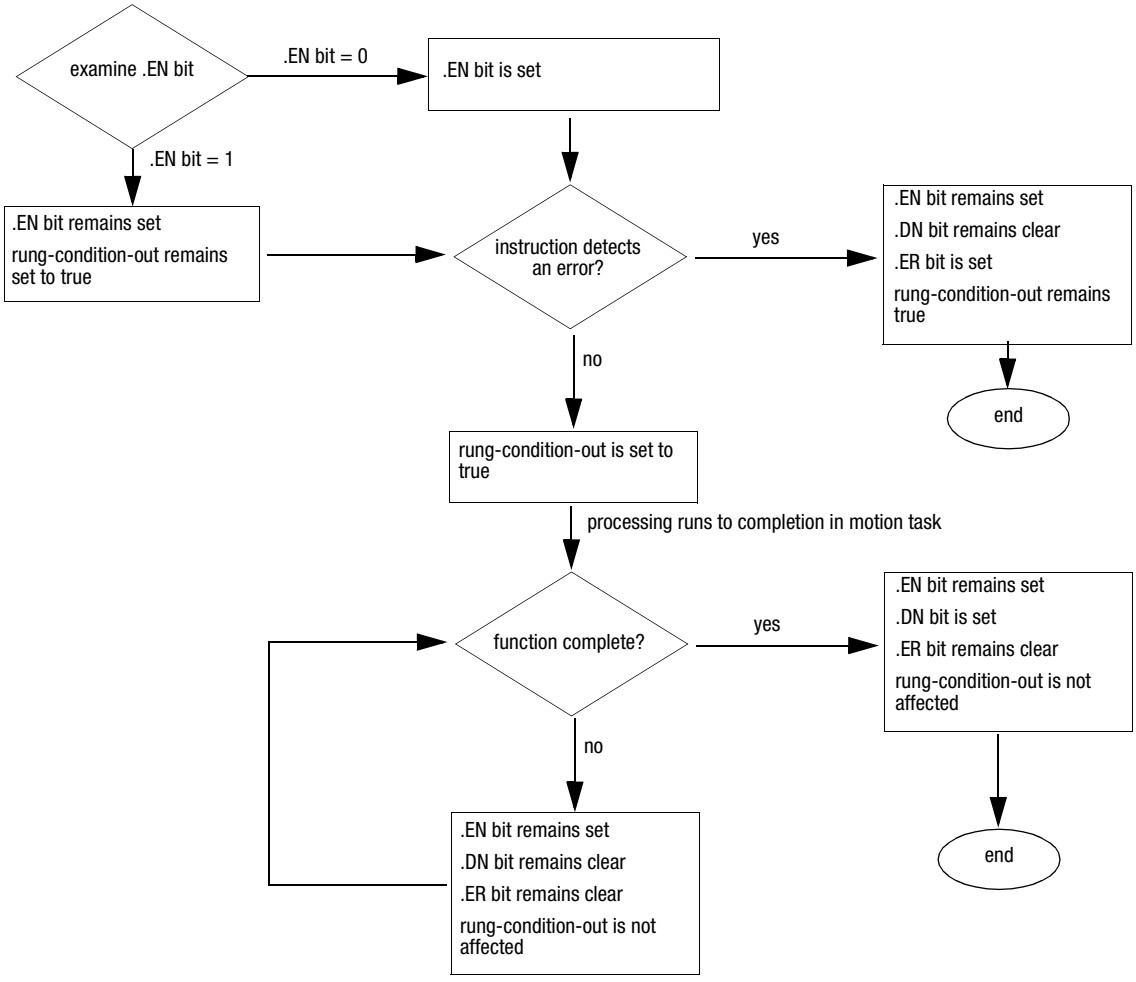


Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction sets the axis to the shutdown state.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MASD Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
8	The axis is not configured as a servo, position only, or virtual axis.
11	The axis is not configured.
12	Messaging to the servo module failed.
18	The axis type is configured as unused.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

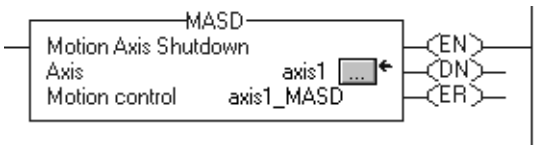
MASD Changes to Servo Module LEDs:

This LED:	Will change to:	Meaning:
FDBK	Flashing green	Servo action is inactive.
DRIVE	Flashing red	• The drive enable output is inactive. • The OK contact is open.

MASD Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
ServoActStatus	False	• The axis is in the axis ready state. • The servo loop is inactive.
DriveEnableStatus	False	The drive enable output is inactive.
ShutdownStatus	True	The axis is in the shutdown state.
AccelStatus	False	The axis is not accelerating.
StoppingStatus	False	The axis is not stopping.
JogStatus	False	The axis is not jogging.
MoveStatus	False	The axis is not moving.
GearingStatus	False	The axis is not gearing.
HomingStatus	False	The axis is not homing.
TuneStatus	False	The axis is not running a tuning process.
TestStatus	False	The axis is not running a testing process.
GearingLockedStatus	False	The axis is not clutching to a new gear speed.
PositionCamStatus	False	Pcam motion profile is not in progress.
TimeCamStatus	False	Tcam motion profile is not in progress.
PositionCamLockedStatus	False	The Pcam is stopped and the lock is cleared.
TimeCamLockedStatus	False	The Tcam is stopped and the lock is cleared.
PositionCamPendingStatus	False	The pending PCAM profile is cancelled.
TimeCamPendingStatus	False	The pending Tcam profile is cancelled.

MASD Example:



When the input conditions are true, the controller forces *axis1* into the shutdown operating state.

Other Formats:

Format:	Syntax:
neutral text	`MASD(axis,motion_control);`
ASCII text	`MASD axis motion_control`

Motion Axis Shutdown Reset (MASR)

The MASR instruction is an output instruction.

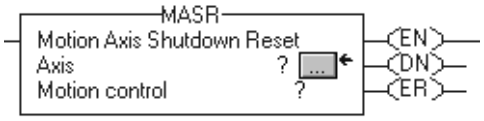
Use the MASR instruction to change an axis from an existing shutdown operating state to an axis ready operating state. If all of the axes of a servo module are removed from the shutdown state as a result of this instruction, the OK relay contacts for the module will close.

Note: Because this instruction can close the OK contacts, you can use this instruction to close the contacts in the E-stop string that control the power to the drive system. Once you close the E-stop string, you can reapply power to the drive.

The MASR instruction uses message type timing.

To use the MASR instruction, configure the axis as either a servo axis or a position-only axis.

Operands:

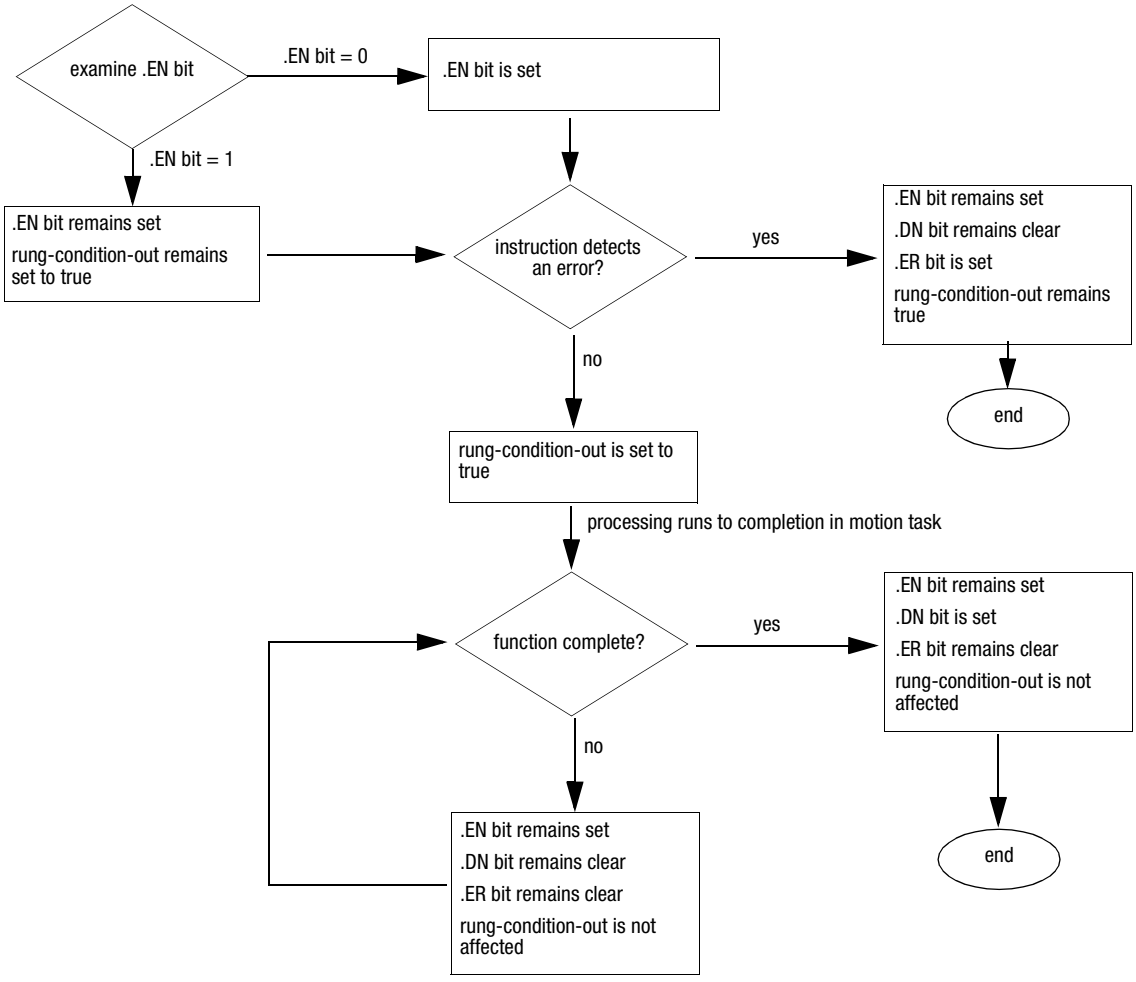


Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction clears the axis from the shutdown state.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MASR Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
8	The axis is not configured as a servo, position only, or virtual axis.
11	The axis is not configured.
12	Messaging to the servo module failed.
18	The axis type is configured as unused.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

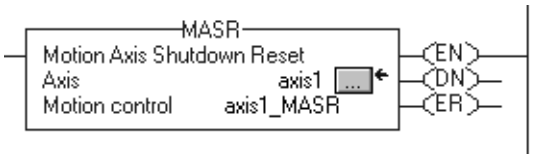
MASR Changes to Servo Module LEDs:

This LED:	Will change to:	Meaning:
FDBK	Flashing green	Servo action is inactive.
DRIVE	Flashing green	• The drive enable output is inactive. • The OK contact is closed.

MASR Changes to Axis Status Bits:

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

MASR Example:



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

Other Formats:

Format:	Syntax:
neutral text	`MASR(axis,motion_control);`
ASCII text	`MASR axis motion_control`

Motion Direct Drive On (MDO)

The MDO instruction is an output instruction.

Use the MDO instruction to enable the servo drive and to set the servo output voltage of an axis.

Common uses for this instruction include:

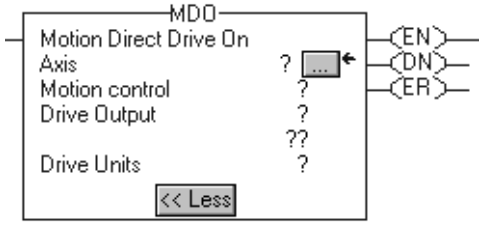
- Creating an independent programmable analog output as an open loop speed or torque reference for a drive.
- Testing a servo drive for closed loop operation.

The MDO instruction uses message type timing.

To use the MDO instruction:

- Configure the axis as a servo axis.
- Ensure that the axis operating state is axis ready.

Operands:



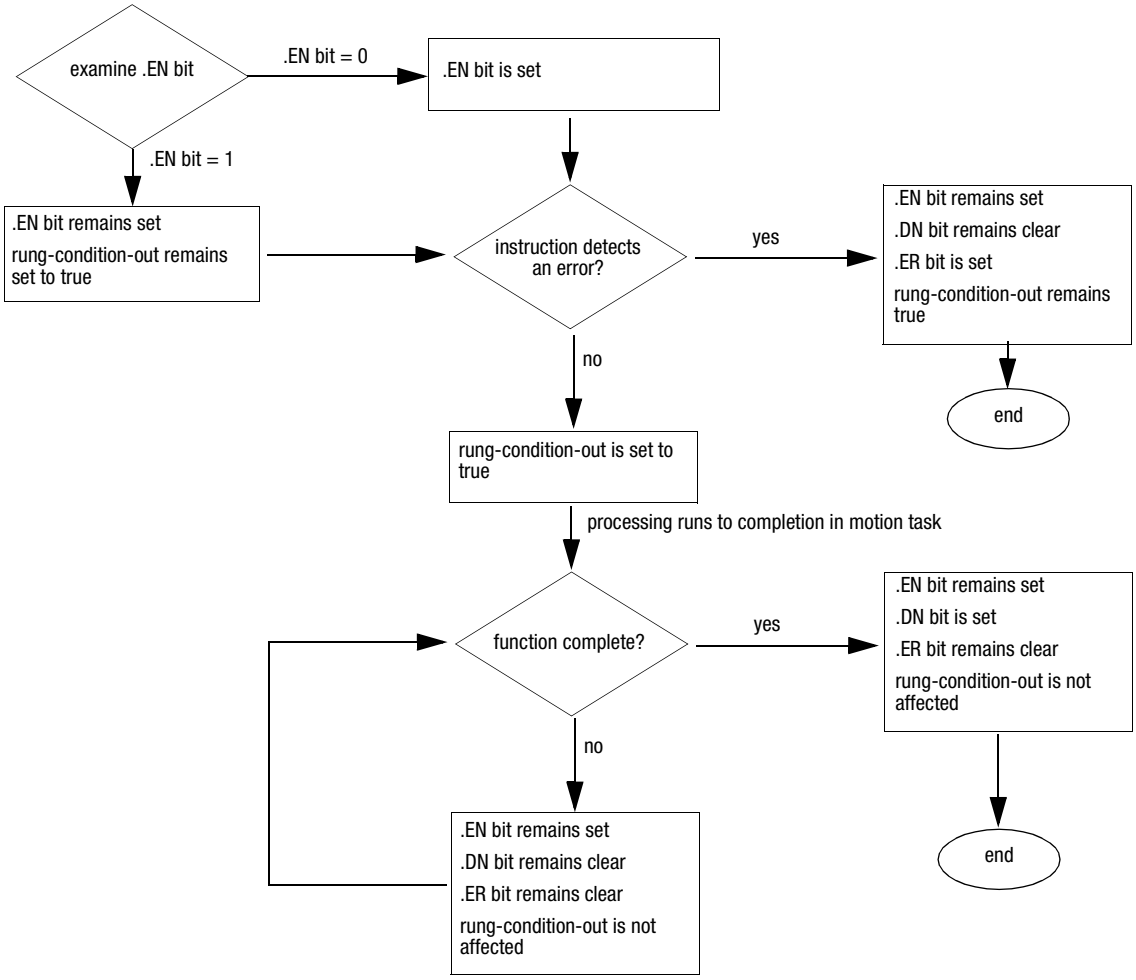
Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Drive Output[a]	SINT, INT, DINT, or REAL	immediate or tag	the new output voltage or percent value for the axis
Drive Units	DINT	immediate	0=volts 1=percent of maximum output limit

a. The 16-bit digital-to-analog convertor on the servo module limits the effective resolution of the MDO instruction to 305 mV or 0.003%. The servo output voltage is not limited by the output limit configuration parameter and is not affected by the servo output polarity configuration bit.

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction updates the axis servo output.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the drive output value was too large.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MDO Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
4	The axis servo loop is closed.
7	The axis is in the shutdown state.
8	The axis is not configured as a servo, position only, or virtual axis.
11	The axis is not configured.
12	Messaging to the servo module failed.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.
20	The axis is in the faulted state.
21	The group is in the faulted state.
24	The controller attempted to execute an MDO, MSO, MAH, MAJ, MAM, MCD, MAPC, MATC, MAG, MRAT, or MRHD instruction when the controller was in the test mode.

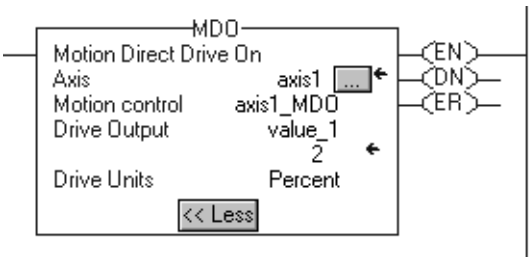
MDO Changes to Servo Module LEDs:

This LED:	Will change to:	Meaning:
FDBK	Flashing green	Servo action is disabled.
DRIVE	Solid green	The drive enable output is active.

MDO Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
DriveEnableStatus	True	• The axis is in the drive control state. • The drive enable output is active.

MDO Example:



When the input conditions are true, the controller activates the servo drive for *axis1* and sets the servo output voltage of *axis1*. In this example, the output will be 2% of the output value.

Other Formats:

Format:	Syntax:
neutral text	`MDO(axis,motion_control,drive_output,drive_units);`
ASCII text	`MDO axis motion_control drive_output drive_units`

Motion Direct Drive Off (MDF)

The MDF instruction is an output instruction.

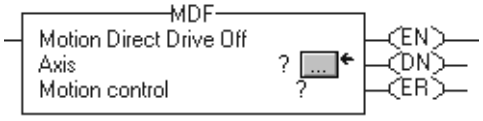
Use the MDF instruction to deactivate the servo drive and to set the servo output voltage to the output offset voltage. The output offset voltage is the output voltage that generates zero or minimal drive motion. You can specify this value during axis configuration.

Common uses for this instruction include:

- Stopping motion initiated by a preceding Motion Direct Drive On (MDO) instruction.
- Changing an axis from the direct drive control operating state to the axis ready operating state.

To use the MDF instruction, configure the axis as a servo axis.

Operands:

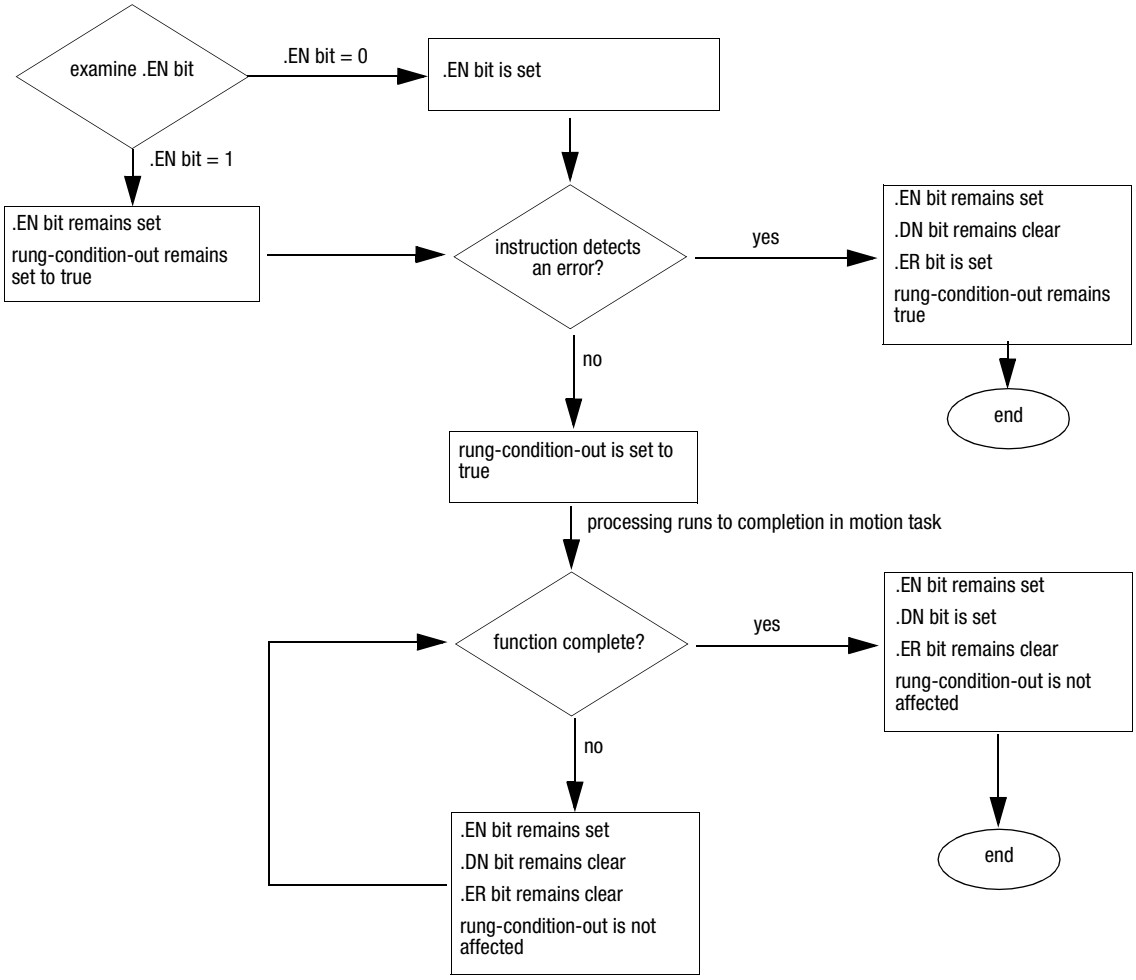


Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction disables the axis drive signals.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MDF Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
4	The axis servo loop is closed.
8	The axis is not configured as a servo axis.
11	The axis is not configured.
12	Messaging to the servo module failed.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

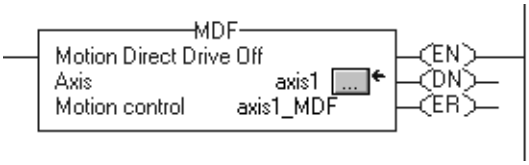
MDF Changes to Servo Module LEDs:

This LED:	Will change to:	Meaning:
FDBK	Flashing green	Servo action is disabled.
DRIVE	Flashing green	The drive enable output is inactive.

MDF Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
DriveEnableStatus	False	• The axis is in the axis ready state. • The drive enable output is inactive.

MDF Example:



When the input conditions are true, the controller deactivates the servo drive for *axis1* and sets the servo output voltage of *axis_* to the output offset value.

Other Formats:

Format:	Syntax:
neutral text	`MDF(axis,motion_control);`
ASCII text	`MDF axis motion_control`

Motion Axis Fault Reset (MAFR)

The MAFR instruction is an output instruction.

Use the MAFR instruction to clear all motion faults for an axis. This is the only method for clearing axis motion faults.

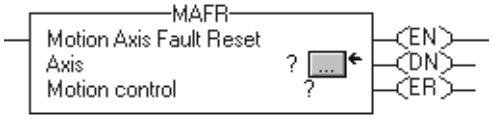
Important: The MAFR instruction removes the fault status, but does not perform any other recovery, such as enabling servo action. In addition, when the controller removes the fault status, the condition that generated the fault(s) may still exist. If the condition is not corrected before using the MAFR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

The MAFR instruction is usually used as a part of a fault handling program. A fault-handling program provides specific action in response to possible faults. Once the fault condition is removed, the MAFR block clears all the active fault status bits.

The MAFR instruction uses message type timing.

To use the MAFR instruction, configure the axis as either a servo axis or a position-only axis.

Operands:

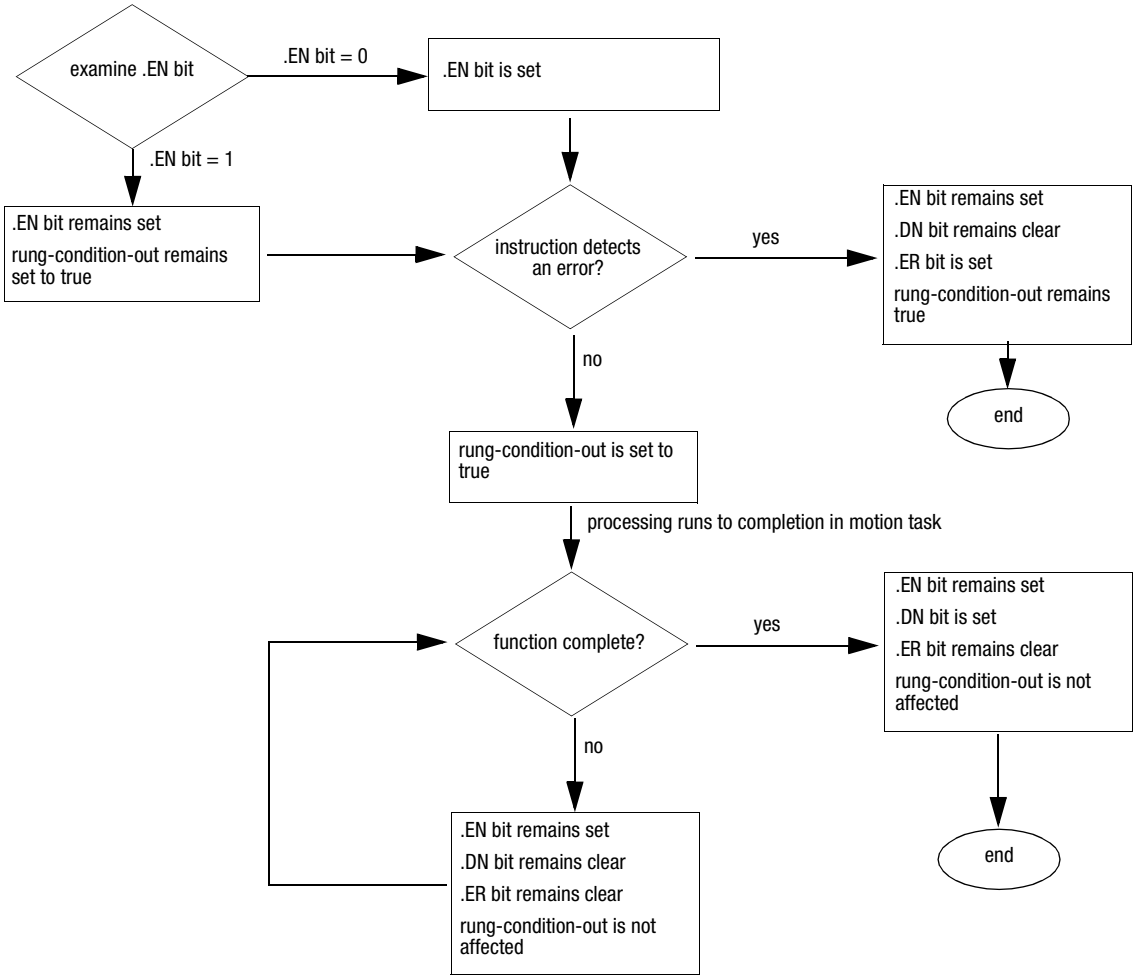


Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction clears all the fault status bits.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MAFR Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
8	The axis is not configured as a servo or position only axis type.
11	The axis is not configured.
12	Messaging to the servo module failed.
18	The axis type is configured as unused.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

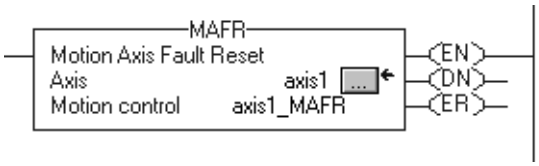
MAFR Changes to Servo Module LEDs:

This LED:	Will change to:	Meaning:
FDBK	Flashing or solid green	There are no feedback faults.
DRIVE	Not solid red	There are no drive faults.

MAFR Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
POtrvlFault	False	The positive overtravel fault is clear.
NOtrvlFault	False	The negative overtravel fault is clear.
PosErrorFault	False	The position error fault is clear.
EncCHALossFault	False	The encoder channel A loss fault is clear.
EncCHBLossFault	False	The encoder channel B loss fault is clear.
EncCHZLossFault	False	The encoder channel Z loss fault is clear.
EncNsFault	False	The encoder noise fault is clear.
DriveFault	False	The drive fault is clear.
HardFault	False	The servo hardware fault is clear.

MAFR Example:



When the input conditions are true, the controller clears all motion faults for *axis1*.

Other Formats:

Format:	Syntax:
neutral text	`MAFR(axis,motion_control);`
ASCII text	`MAFR axis motion_control`

Motion Move Instructions

(MAS, MAH, MAJ, MAM, MAG, MCD, MRP, MCCP, MAPC, MATC)

ATTENTION: 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 of the control variables.

Introduction

Motion move instructions control all aspects of axis position. The motion move instructions are:

If you want to:	Use this instruction:	See page:
Initiate a controlled stop of any motion process on an axis.	MAS	3-2
Home an axis.	MAH	3-7
Initiate a jog motion profile for an axis.	MAJ	3-12
Initiate a move profile for an axis.	MAM	3-17
Provide electronic gearing between any two axes.	MAG	3-22
Change the speed, acceleration rate, or deceleration rate of a move profile or a jog profile in progress.	MCD	3-27
Change the command or actual position of an axis.	MRP	3-31
Compute a cam profile.	MCCP	3-34
Initiate a position cam profile.	MAPC	3-37
Initiate a time cam profile	MATC	3-43

Motion Axis Stop (MAS)

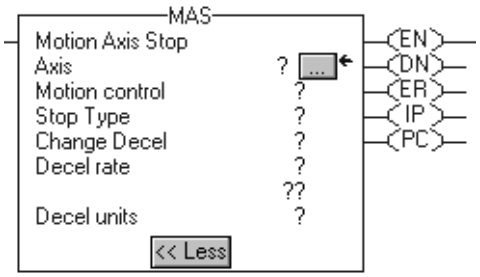
The MAS instruction is an output instruction.

Use a MAS instruction to initiate a controlled stop of any motion process on an axis and to clear associated motion status flags. If the axis is not moving when the MAS instruction executes, the MAS instruction has no effect on motion.

The MAS instruction uses immediate and process type timing.

To use the MAS instruction, configure the axis as a servo or virtual axis.

Operands:



Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Stop type	DINT	immediate	select the type of stop: • stop all motion • stop jogging • stop moving • stop gearing • stop homing • stop tuning • stop test • stop position camming • stop time camming
Change decel	DINT	immediate	select whether to change deceleration from the maximum value for the axis: • no • yes
Decel rate	SINT, INT, DINT, or REAL	immediate or tag	deceleration rate of the axis
Decel units	DINT	immediate	select the deceleration units: • % of maximum deceleration • units per sec2

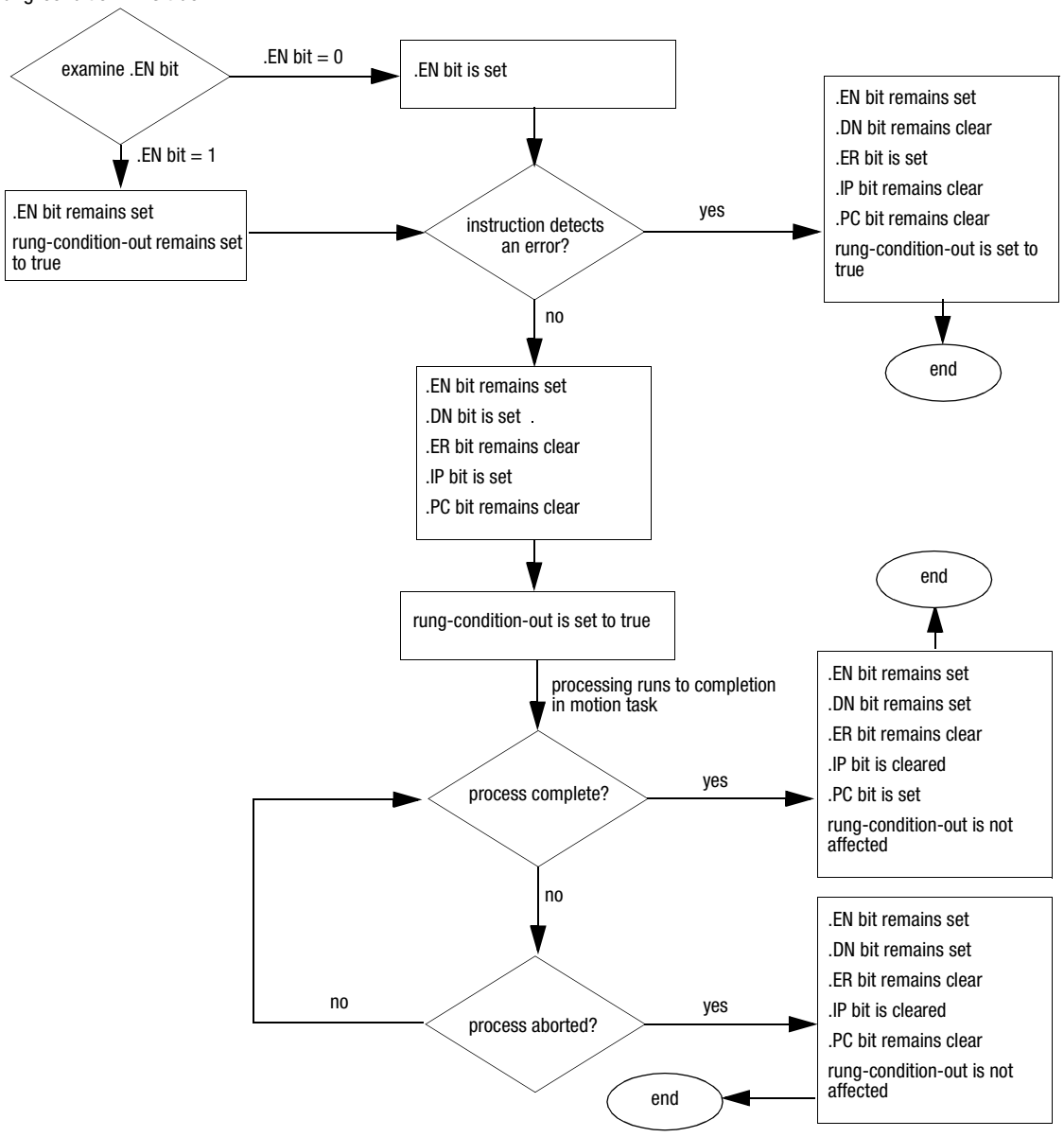
MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction initiates an axis stop.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

.IP	BOOL	• The in process bit sets when a stop process is successfully initiated.
		• It resets when one of the following events occurs:
		  • The MAS instruction completes.
		  • A servo fault terminates the MAS instruction.

.PC	BOOL	The process complete bit sets after the stop operation completes.
.ACCEL	BOOL	The .ACCEL bit indicates that the velocity has increased for the individual instruction that it is tied to i.e jog, move, gearing.
.DECEL	BOOL	The .DECEL bit indicates that the velocity has decreased for the individual instruction that it is tied to i.e jog, move, gearing.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The .IP bit is cleared. The .PC bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The .IP bit is not affected. The .PC bit is not affected The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

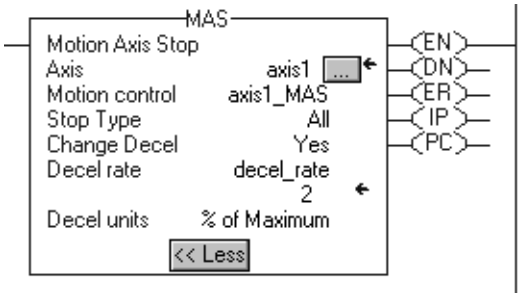
MAS Error Codes (.ERR):

Error Code:	Error Message	Description:
5	Servo Off State Error	The instruction tried to execute on an axis whose servo loop is not closed.
7	Shutdown State Error	The axis is in the shutdown state.
8	Illegal Axis Type	The axis is not configured as a servo or virtual axis type.
11	Axis Not Configured	The axis is not configured.
13	Parameter Out Of Range	The instruction tried to use a parameter that is outside the range limit.
19	Group Not Synchronized	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

MAS Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
StoppingStatus	True	The axis is stopping.
JogStatus	False	The axis is not jogging.
MoveStatus	False	The axis is not moving.
GearingStatus	False	The axis is not gearing.
HomingStatus	False	The axis is not homing.
DecelStatus	True	The axis is decelerating.
PositionCamStatus	False	Pcam motion profile is not in progress.
TimeCamStatus	False	Tcam motion profile is not in progress.
PositionCamLockedStatus	False	The Pcam is stopped and the lock is cleared.
TimeCamLockedStatus	False	The Tcam is stopped and the lock is cleared.
PositionCamPendingStatus	False	The pending PCAM profile is cancelled.
TimeCamPendingStatus	False	The pending Tcam profile is cancelled.

MAS Example:



When the input conditions are true, the controller stops motion on *axis1* and clears all associated motion status flags.

Other Formats:

Format:	**Syntax:**
neutral text	`MAS(axis,motion_control,stop_type,change_decel,rate_units);`
ASCII text	`MAS axis motion_control stop_type change_decel rate units`

Motion Axis Home (MAH)

The MAH instruction is an output instruction. Use the MAH instruction to home an axis.

Two different homing modes are available:

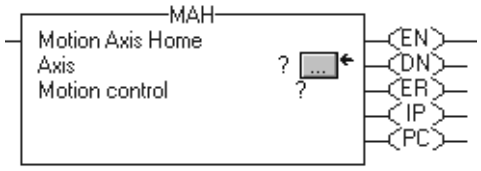
- Active—the axis executes the configured home sequence type and establishes an absolute axis position.
- Passive—the axis waits for the next marker pulse to establish an accurate home position.

The MAH instruction uses message and process type timing.

To use the MAH instruction:

If you want to use:	Then:
Active homing	• Configure the axis as a servo or virtual axis. • Ensure that the axis operating state is servo on or axis ready.
Passive homing	Configure the axis as either a servo axis or a position-only axis.

Operands:



Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure

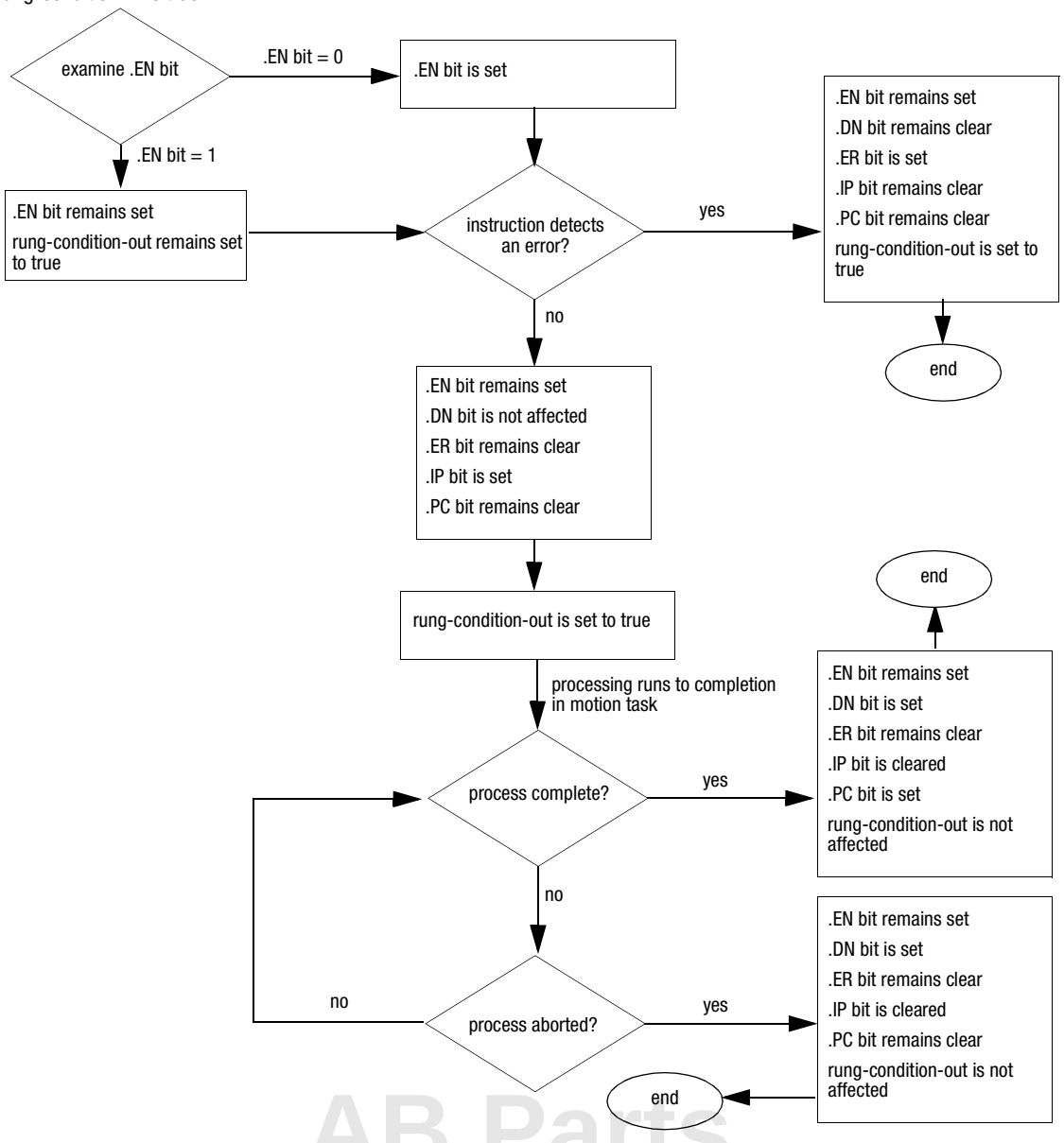
MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit sets when homing completes or is aborted.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

.IP	BOOL	• The in process bit sets when the homing process is successfully initiated.
		• It resets when one of the following events occurs:
		  • The MAH instruction completes.
		  • Homing is aborted.
		  • A servo fault occurs.

.PC	BOOL	The process complete bit sets when the instruction completes an axis home operation.
.ACCEL	BOOL	The .ACCEL bit indicates that the velocity has increased for the individual instruction that it is tied to i.e jog, move, gearing.
.DECEL	BOOL	The .DECEL bit indicates that the velocity has decreased for the individual instruction that it is tied to i.e jog, move, gearing.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The .IP bit is cleared. The .PC bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The .IP bit is not affected. The .PC bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MAH Error Codes (.ERR):

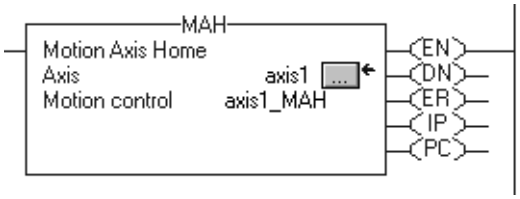
Error Code:	Error Message	Description:
3	Execution Collision	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
7	Shutdown State Error	The axis is in the shutdown state.
8	Illegal Axis Type	The axis is not configured as a servo or virtual axis.
11	Axis Not Configured	The axis is not configured.
12	Servo Message Failure	Messaging to the servo module failed.
18	Axis Type Unused	The axis type is configured as unused.
19	Group Not Synchronized	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.
20	Axis In Faulted State	The axis is in the faulted state.
21	Group In Faulted State	The group is in the faulted state.
22	Axis In Motion	An MSO (Motion Servo On) or MAH (Motion Axis Home) instruction was attempted while the axis was in motion.
24	Illegal AC Mode Op	The controller attempted to execute an MDO, MSO, MAH, MAJ, MAM, MCD, MAPC, MATC, MAG, MRAT, or MRHD instruction when the controller was in the test mode.

MAH Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
StoppingStatus	False	The axis is not stopping.
JogStatus	False/True[1]	The axis is not jogging.
MoveStatus	False/True[1]	The axis is not moving.
AccelStatus	True[1]	The axis is accelerating.
DecelStatus	True[1]	The axis is decelerating.
HomingStatus	True	The axis is homing.

1 If you select active homing, this status bit will be true.

MAH Example:



When the input conditions are true, the controller homes *axis1*.

Other Formats:

Format:	Syntax:
neutral text	`MAH(axis,motion_control);`
ASCII text	`MAH axis motion_control`

Motion Axis Jog (MAJ)

The MAJ instruction is an output instruction.

Use the MAJ instruction to initiate a jog motion profile for an axis.

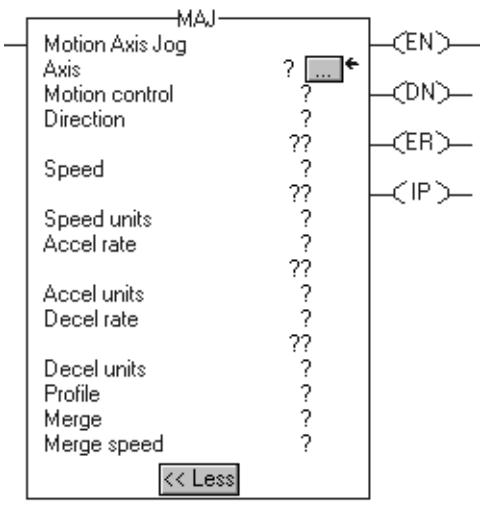
To stop an axis jog, use the Motion Axis Jog (MAJ) instruction with a speed of zero or the Motion Axis Stop (MAS) instruction.

The MAJ instruction uses immediate and process type timing.

To use the MAJ instruction:

- Configure the axis as a servo or virtual axis.
- Ensure that the axis operating state is servo on if the axis is a servo.

Operands:



Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Direction	SINT, INT, or DINT	immediate or tag	select the direction of the jog: • 0 = forward jog • 1 = reverse jog
Speed	SINT, INT, DINT, or REAL	immediate or tag	speed to jog the axis
Speed units	DINT	immediate	select the speed units: • % of maximum speed • units per sec
Accel rate	SINT, INT, DINT, or REAL	immediate or tag	acceleration rate of the axis
Accel units	DINT	immediate	select the acceleration units: • % of maximum acceleration • units per sec^2
Decel rate	SINT, INT, DINT, or REAL	immediate or tag	deceleration rate of the axis
Decel units	DINT	immediate	select the deceleration units: • % of maximum deceleration • units per sec^2
Profile	DINT	immediate	select the velocity profile to run the jog: • trapezoidal • S-curve
Merge	DINT	immediate	select whether to turn all axis movement into pure jog: • disabled • enabled

Operand:	Type:	Format:	Description:
Merge speed	DINT	immediate	if you enabled Merge, select the speed of the jog profile: • programmed value in the speed field • current axis speed

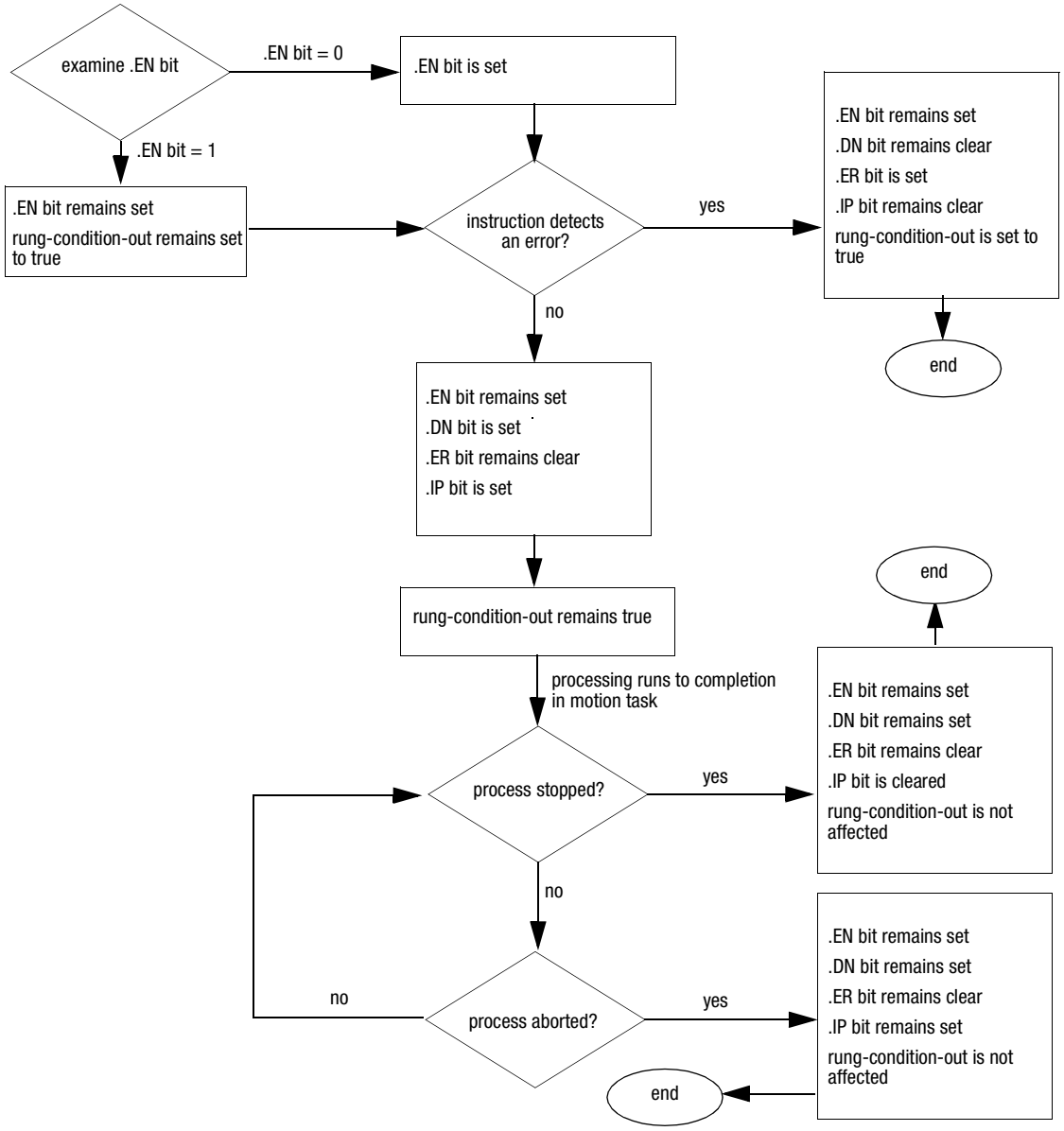
MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction initiates an axis jog.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

.IP	BOOL	• The in process bit sets when the jog process is successfully initiated.
		• It resets when one of the following events occurs:
		  • Another MAJ instance supersedes the current instruction.
		  • The jog is aborted.
		  • A servo fault terminates the MAJ instruction.

.ACCEL	BOOL	The .ACCEL bit indicates that the velocity has increased for the individual instruction that it is tied to i.e jog, move, gearing.
.DECEL	BOOL	The .DECEL bit indicates that the velocity has decreased for the individual instruction that it is tied to i.e jog, move, gearing.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The .IP bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The .IP bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MAJ Error Codes (.ERR):

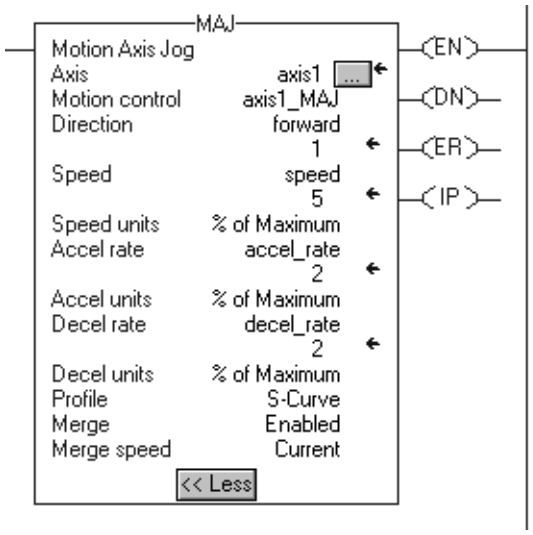
Error Code:	Error Message	Description:
5	Servo Off State Error	The instruction tried to execute on an axis whose servo loop is not closed.
7	Shutdown State Error	The axis is in the shutdown state.
8	Illegal Axis Type	The axis is not configured as a servo or virtual axis.
9	Overtravel Condition	The instruction tried to execute in a direction that aggravates the current overtravel condition.
11	Axis Not Configured	The axis is not configured.
13	Parameter Out Of Range	The instruction tried to execute with a parameter that is outside the range limit.
16	Home In Process Error	The instruction tried to execute with homing in process.
19	Group Not Synchronized	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.
23	Illegal Dynamic Change	An instruction attempted an illegal change of dynamics.
24	Illegal AC Mode Op	The controller attempted to execute an MDO, MSO, MAH, MAJ, MAM, MCD, MAPC, MATC, MAG, MRAT, or MRHD instruction when the controller was in the test mode.

MAJ Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
JogStatus	True	The axis is jogging.
AccelStatus	True	The axis is accelerating.
DecelStatus	True	The axis is decelerating.
MoveStatus[1]	False	The axis is not moving.
GearingStatus[1]	False	The axis is not gearing.
PositionCamStatus	False	Pcam motion profile is not in progress.
TimeCamStatus	False	Tcam motion profile is not in progress.
PositionCamPendingStatus	False	The pending PCAM profile is cancelled.
TimeCamPendingStatus	False	The pending Tcam profile is cancelled.

1 If you selected the merge option, the MAJ instruction will change this status bit.

MAJ Example:



When the input conditions are true, the controller initiates a jog motion for *axis1*.

Other Formats:

Format:	**Syntax:**
neutral text	`MAJ(`*`axis,motion_control,direction,speed,units,accel_rate,units,decel_rate,units,profile,merge,merge_speed`*`);`
ASCII text	`MAJ` *`axis motion_control direction speed units accel_rate units decel_rate units profile merge merge_speed`*

Motion Axis Move (MAM)

The MAM instruction is an output instruction.

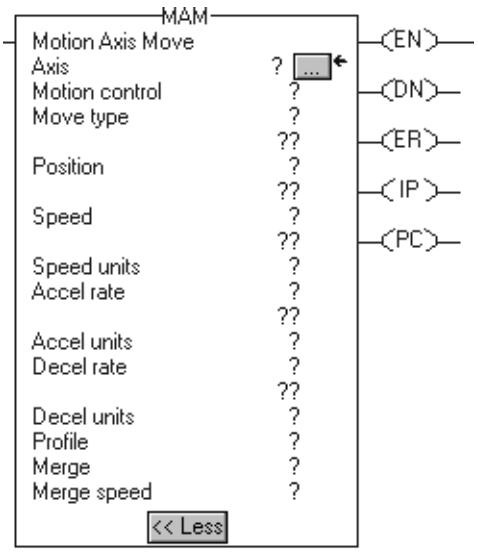
Use the MAM instruction to initiate a move profile for an axis.

The MAM instruction uses immediate and process type timing.

To use the MAM instruction:

- Configure the axis as a servo or virtual axis.
- Ensure that the axis operating state is servo on, if the axis is servo type.

Operands:



Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Move type	SINT, INT, or DINT	immediate or tag	select the type of move: • absolute move • incremental move • rotary shortest path move • rotary positive move • rotary negative move
Position	SINT, INT, DINT, or REAL	immediate or tag	value of the absolute command position to move to, or for incremental movement, the value of the distance to move from the current command position
Speed	SINT, INT, DINT, or REAL	immediate or tag	speed to move the axis
Speed units	DINT	immediate	select the speed units: • % of maximum speed • units per sec
Accel rate	SINT, INT, DINT, or REAL	immediate or tag	acceleration rate of the axis
Accel units	DINT	immediate	select the acceleration units: • % of maximum acceleration • units per sec^2
Decel rate	SINT, INT, DINT, or REAL	immediate or tag	deceleration rate of the axis
Decel units	DINT	immediate	select the deceleration units: • % of maximum deceleration • units per sec^2
Profile	DINT	immediate	select the velocity profile to run the move • trapezoidal • S-curve

Operand:	Type:	Format:	Description:
Merge	DINT	immediate	select whether to turn all axis movement into pure move: • disabled • enabled
Merge speed	DINT	immediate	if you enabled Merge, select the speed of the move profile: • programmed value in the speed field • current axis speed

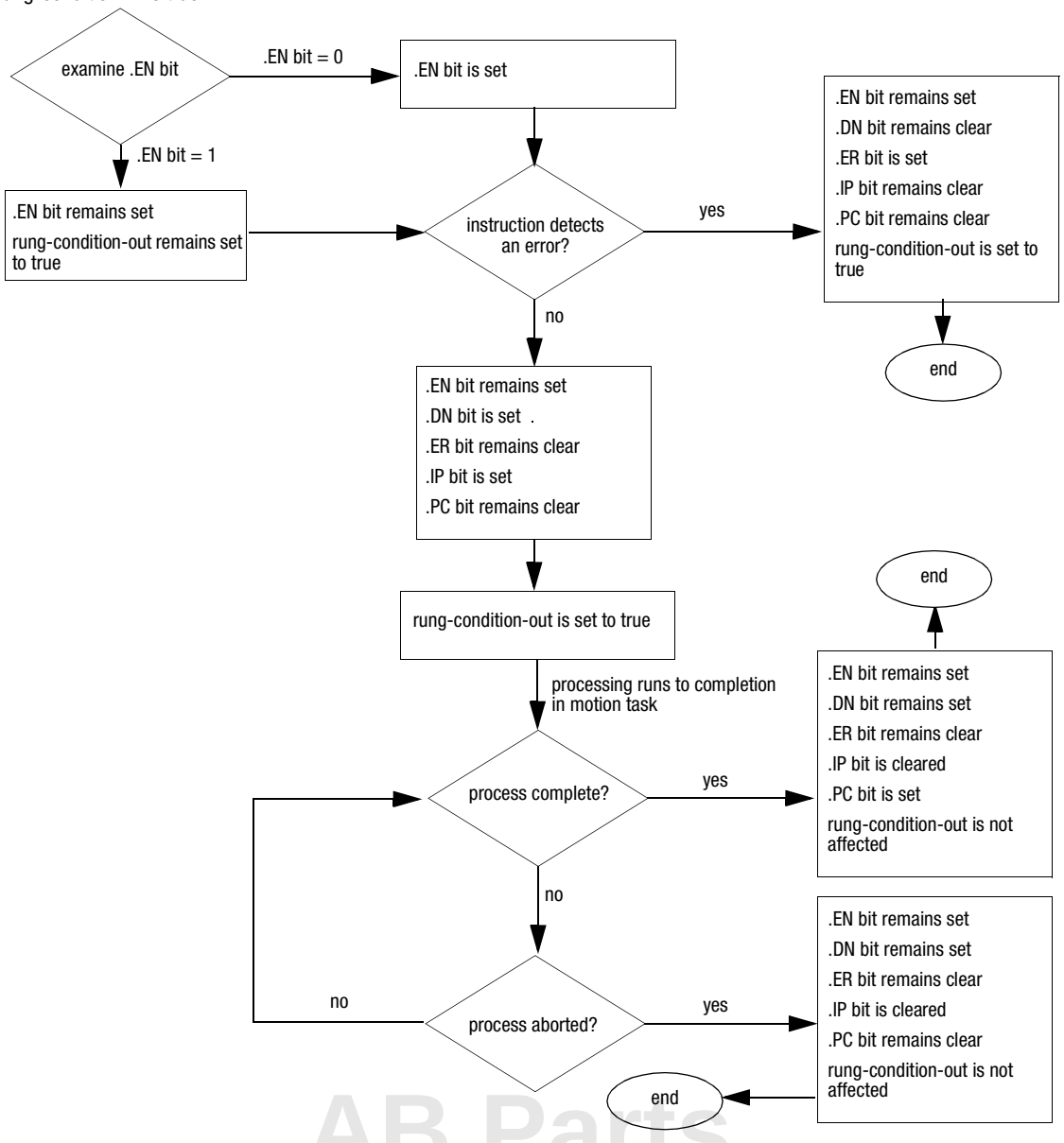
MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction initiates an axis move.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

.IP	BOOL	• The in process bit sets when the move process is successfully initiated.
		• It resets when one of the following events occurs:
		  • The MAM instruction completes.
		  • The move is aborted.
		  • A servo fault terminates the MAM instruction.

.PC	BOOL	The process complete bit sets when the instruction completes an axis move.
.ACCEL	BOOL	The .ACCEL bit indicates that the velocity has increased for the individual instruction that it is tied to i.e jog, move, gearing.
.DECEL	BOOL	The .DECEL bit indicates that the velocity has decreased for the individual instruction that it is tied to i.e jog, move, gearing.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The .IP bit is cleared. The .PC bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The .IP bit is not affected. The .PC bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MAM Error Codes (.ERR):

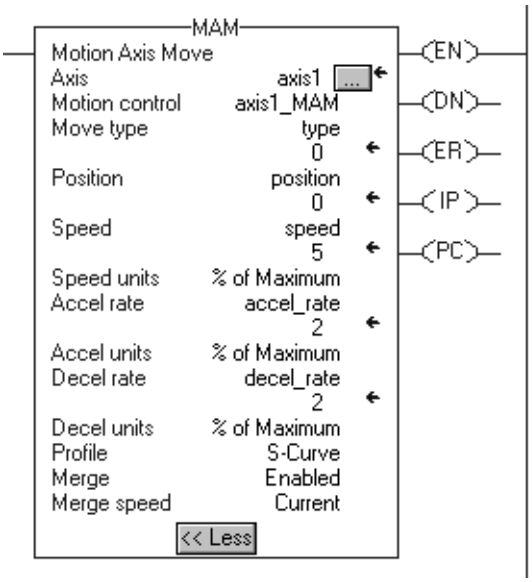
Error Code:	Error Message	Description:
5	Servo Off State Error	The instruction tried to execute on an axis whose servo loop is not closed.
7	Shutdown State Error	The axis is in the shutdown state.
8	Illegal Axis Type	The axis is not configured as a servo or virtual axis.
9	Overtravel Condition	The instruction tried to execute in a direction that aggravates the current overtravel condition.
11	Axis Not Configured	The axis is not configured.
13	Parameter Out Of Range	The instruction tried to use a parameter that is outside the legal range limit.
16	Home In Process Error	The instruction tried to execute with homing in process.
17	Axis Mode Not Rotary	The instruction tried to execute a rotary move on an axis that is not configured for rotary operation.
19	Group Not Synchronized	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.
23	Illegal Dynamic Change	An instruction attempted an illegal change of dynamics.
24	Illegal AC Mode Op	The controller attempted to execute an MDO, MSO, MAH, MAJ, MAM, MCD, MAPC, MATC, MAG, MRAT, or MRHD instruction when the controller was in the test mode.

MAM Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
JogStatus[1]	False	The axis is not jogging.
MoveStatus	True	The axis is moving.
GearingStatus[1]	False	The axis is not gearing.
AccelStatus	True	The axis is accelerating during the move.
DecelStatus	True	The axis is decelerating during the move.
PositionCamStatus	False	Pcam motion profile is not in progress.
TimeCamStatus	False	Tcam motion profile is not in progress.
PositionCamPendingStatus	False	The pending PCAM profile is cancelled.
TimeCamPendingStatus	False	The pending Tcam profile is cancelled.

1 If you selected the merge option, the MAM instruction will change this status bit.

MAM Example:



When the input conditions are true, the controller initiates a move for *axis1*.

Other Formats:

Format:	**Syntax:**
neutral text	`MAM(`*`axis,motion_control,move_type,position,speed,units,accel_rate,units,decel_rate,units,profile,merge,merge_speed`*`);`
ASCII text	`MAM` *`axis motion_control move_type position speed units accel_rate units decel_rate units profile merge merge_speed`*

Motion Axis Gearing (MAG)

The MAG instruction is an output instruction.

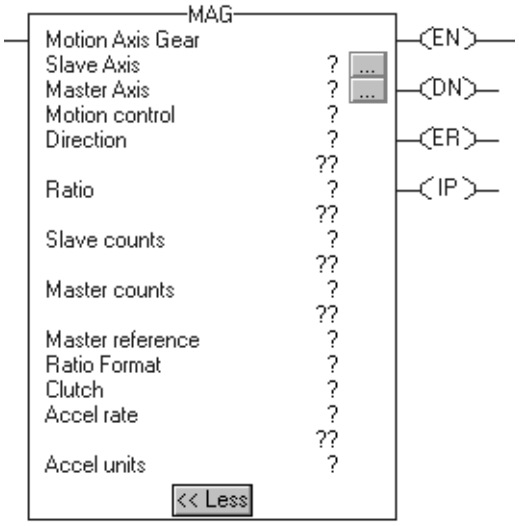
Use the MAG instruction to provide electronic gearing between any two axes.

The MAG instruction uses immediate and process type timing.

To use the MAG instruction:

- Configure the slave axis as a servo or virtual axis.
- Ensure that the slave axis operating state is servo on if the axis type is servo.

Operands:



Operand:	Type:	Format:	Description:
Slave axis	AXIS	tag	axis to perform the operation on
Master axis	AXIS	tag	axis that the slave axis follows
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Direction	SINT, INT, or DINT	immediate or tag	select the direction of the slave axis relative to the master axis: • 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	SINT, INT, DINT, or REAL	immediate or tag	signed, real value of the gear ratio of slave units per master units
Slave counts	SINT, INT, or DINT	immediate or tag	the slave encoder counts for an integer fraction
Master counts	SINT, INT, or DINT	immediate or tag	the master encoder counts for an integer fraction
Master reference	DINT	immediate	Sets the master position reference to either Command position or Actual position. • Actual – slave axis motion is generated from the current position of the master axis as measured by its encoder or other feedback device. • Command – slave axis motion is generated from the desired or commanded position of the master axis.

Operand:	Type:	Format:	Description:
Ratio format	DINT	immediate	select the format of the ratio between the slave and the master axis • real gear ratio • integer fraction of slave encoder counts to master encoder counts
Clutch	DINT	immediate	select whether or not to ramp the slave axis to gearing speed using the acceleration value: • disabled • enabled
Accel rate	SINT, INT, DINT, or REAL	immediate or tag	acceleration of the slave axis for clutching
Accel units	DINT	immediate	select the acceleration units for clutching: • % of maximum acceleration • units per sec2

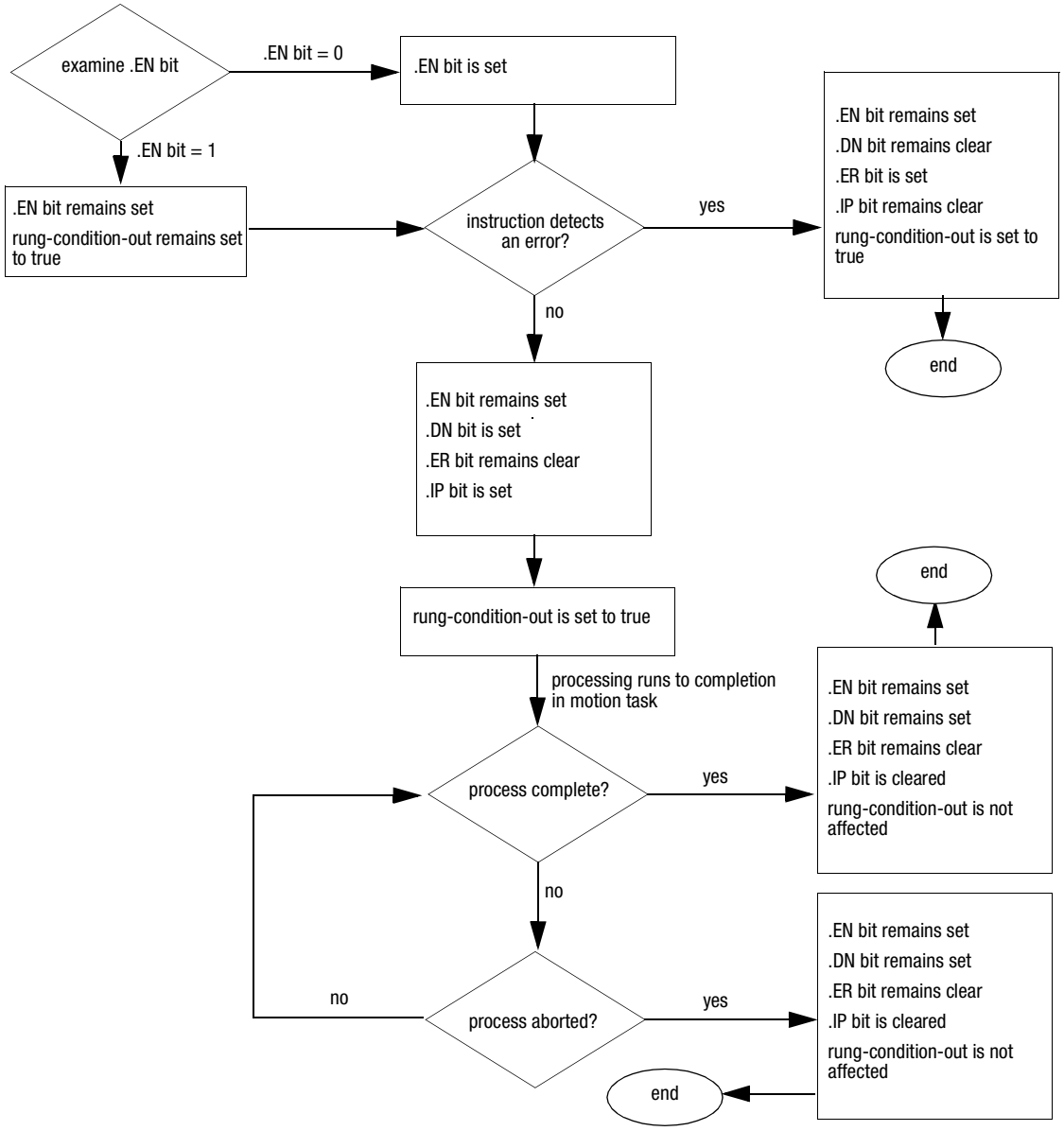
MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction initiates an axis gear.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

.IP	BOOL	• The in process bit sets when the gearing process is successfully initiated.
		• It resets when one of the following events occurs:
		  • The gear is aborted.
		  • A servo fault terminates the MAG instruction.

.ACCEL	BOOL	The .ACCEL bit indicates that the velocity has increased for the individual instruction that it is tied to i.e jog, move, gearing.
.DECEL	BOOL	The .DECEL bit indicates that the velocity has decreased for the individual instruction that it is tied to i.e jog, move, gearing.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The .IP bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The .IP bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MAG Error Codes (.ERR):

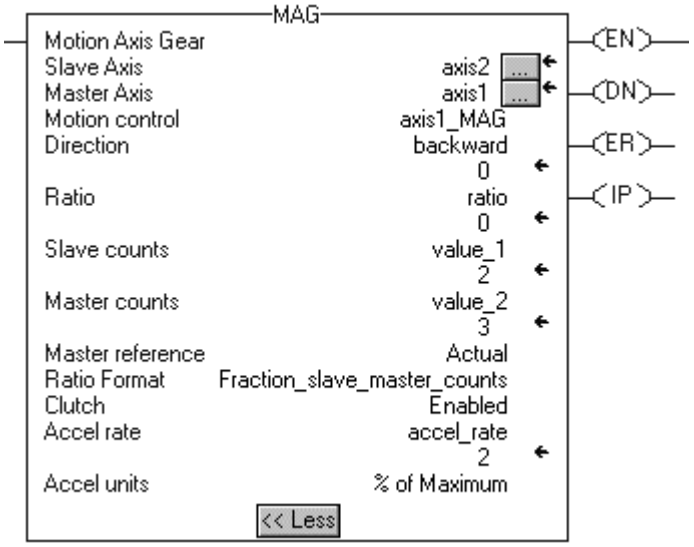
Error Code:	Error Message	Description:
5	Servo Off State Error	The instruction tried to execute on an axis whose servo loop is not closed.
7	Shutdown State Error	The axis is in the shutdown state.
8	Illegal Axis Type	The axis is not configured as a servo or virtual axis.
10	Master Axis Conflict	The master axis reference is the same as the slave axis reference.
11	Axis Not Configured	The axis is not configured.
13	Parameter Out Of Range	The instruction tried to use a parameter that is outside the range limit.
19	Group Not Synchronized	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.
24	Illegal AC Mode Op	The controller attempted to execute an MDO, MSO, MAH, MAJ, MAM, MCD, MAPC, MATC, MAG, MRAT, or MRHD instruction when the controller was in the test mode.

MAG Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
StoppingStatus[1]	True	The axis is stopping.
GearingStatus	True	The axis is gearing.
AccelStatus	True	The axis is accelerating when reaching the new speed.
DecelStatus	True	The axis is decelerating to reach the new speed.
PositionCamStatus	False	Pcam motion profile is not in progress.
TimeCamStatus	False	Tcam motion profile is not in progress.
PositionCamPendingStatus	False	The pending PCAM profile is cancelled.
TimeCamPendingStatus	False	The pending Tcam profile is cancelled.

1 If you selected the clutch option, the MAG instruction will change this status bit if the master current speed does not equal the slave current speed.

MAG Example:



When the input conditions are true, the controller provides electronic gearing between *axis2* and *axis1*.

Other Formats:

Format:	Syntax:
neutral text	`MAG(slave_axis,master_axis,motion_control,direction,ratio,slave_counts, master_counts,master_reference,ratio_format,clutch,accel_rate,units);`
ASCII text	`MAG slave_axis master_axis motion_control direction ratio slave_counts master_counts master_reference ratio_format clutch accel_rate units`

Motion Change Dynamics (MCD)

The MCD instruction is an output instruction. Use the MCD instruction to change the speed, acceleration rate, or deceleration rate of a move profile or a jog profile in progress.

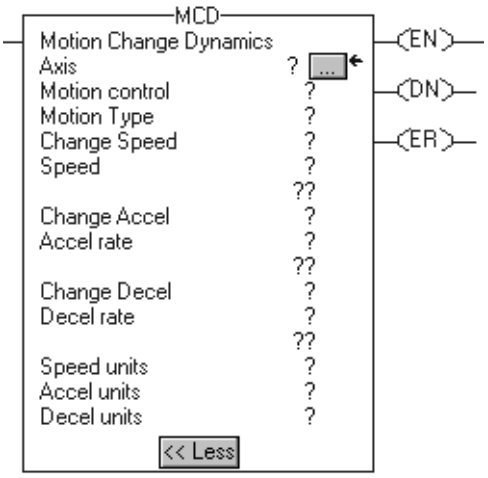
The MCD instruction uses immediate type timing.

To use the MCD instruction:

- Configure the axis as a servo or virtual axis.
- Ensure the axis operating state is servo on, if axis type is servo.

If the axis does not have a move or a jog in progress when the MCD instruction executes, the instruction will error.

Operands:

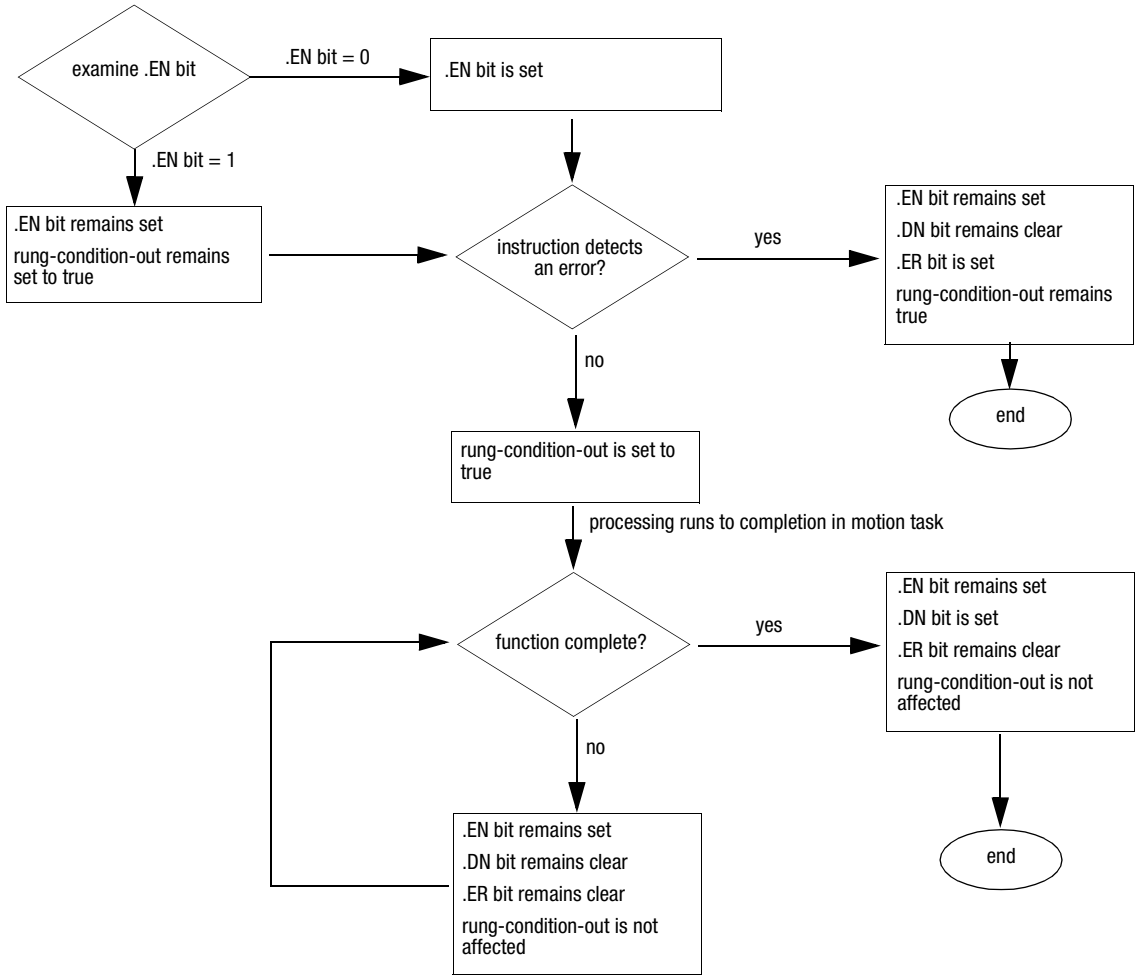


Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Motion type	DINT	immediate	select the motion profile to change: • jog • move
Change speed	DINT	immediate	select whether or not to change speed: • no • yes
Speed	SINT, INT, DINT, or REAL	immediate or tag	the new speed of the axis
Change accel	DINT	immediate	select whether to change acceleration: • no • yes
Accel rate	SINT, INT, DINT, or REAL	immediate or tag	acceleration rate of the axis
Change decel	DINT	immediate	select whether to change deceleration: • no • yes
Decel rate	SINT, INT, DINT, or REAL	immediate or tag	deceleration rate of the axis
Speed units	DINT	immediate	select the speed units: • % of maximum speed • units per sec
Accel units	DINT	immediate	select the acceleration units: • % of maximum acceleration • units per sec^2
Decel units	DINT	immediate	select the deceleration units: • % of maximum deceleration • units per sec^2

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled.
.DN	BOOL	The done bit indicates when the instruction initiates change of axis dynamics.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

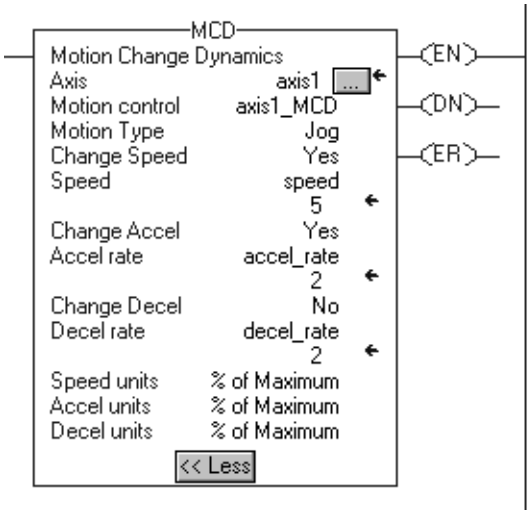
Arithmetic Status Flags: not affected

Fault Conditions: none

MCD Error Codes (.ERR):

Error Code:	Error Message	Description:
5	Servo Off State Error	The instruction tried to execute on an axis whose servo loop is not closed.
7	Shutdown State Error	The axis is in the shutdown state.
8	Illegal Axis Type	The axis is not configured as a servo or virtual axis.
11	Axis Not Configured	The axis is not configured.
13	Parameter Out Of Range	The instruction tried to use a parameter that is outside the range limit.
16	Home In Process Error	The instruction tried to execute with homing in progress.
19	Group Not Synchronized	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.
23	Illegal Dynamic Change	An instruction attempted an illegal change of dynamics.
24	Illegal AC Mode Op	The controller attempted to execute an MDO, MSO, MAH, MAJ, MAM, MCD, MAPC, MATC, MAG, MRAT, or MRHD instruction when the controller was in the test mode.

MCD Example:



When the input conditions are true, the controller changes the speed, acceleration, or deceleration rate of a move profile or jog profile in progress for *axis1*.

Other Formats:

Format:	Syntax:
neutral text	`MCD(a`*`xis,motion_control,motion_type,change_speed,speed,change_accel,accel_rate,change_decel,decel_rate,speed_units,accel_units,decel_units`*`);`
ASCII text	`MCD(a`*`xis motion_control motion_type change_speed speed change_accel accel_rate change_decel decel_rate speed_units accel_units decel_units`*

Motion Redefine Position (MRP)

The MRP instruction is an output instruction.

Use the MRP instruction to change the command or actual position of an axis.

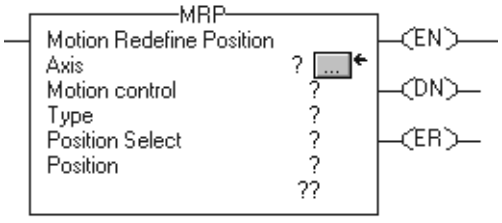
This instruction does not cause any motion; the axis position is only redefined. The controller can calculate new axis positions in two ways:

- Absolute, where the controller assigns the position value as the new actual or command position.
- Relative, where the controller adds the position value to the current actual or command position.

The MRP instruction uses message type timing.

To use the MRP instruction, configure the axis as either a servo axis, virtual axis, or a position-only axis.

Operands:

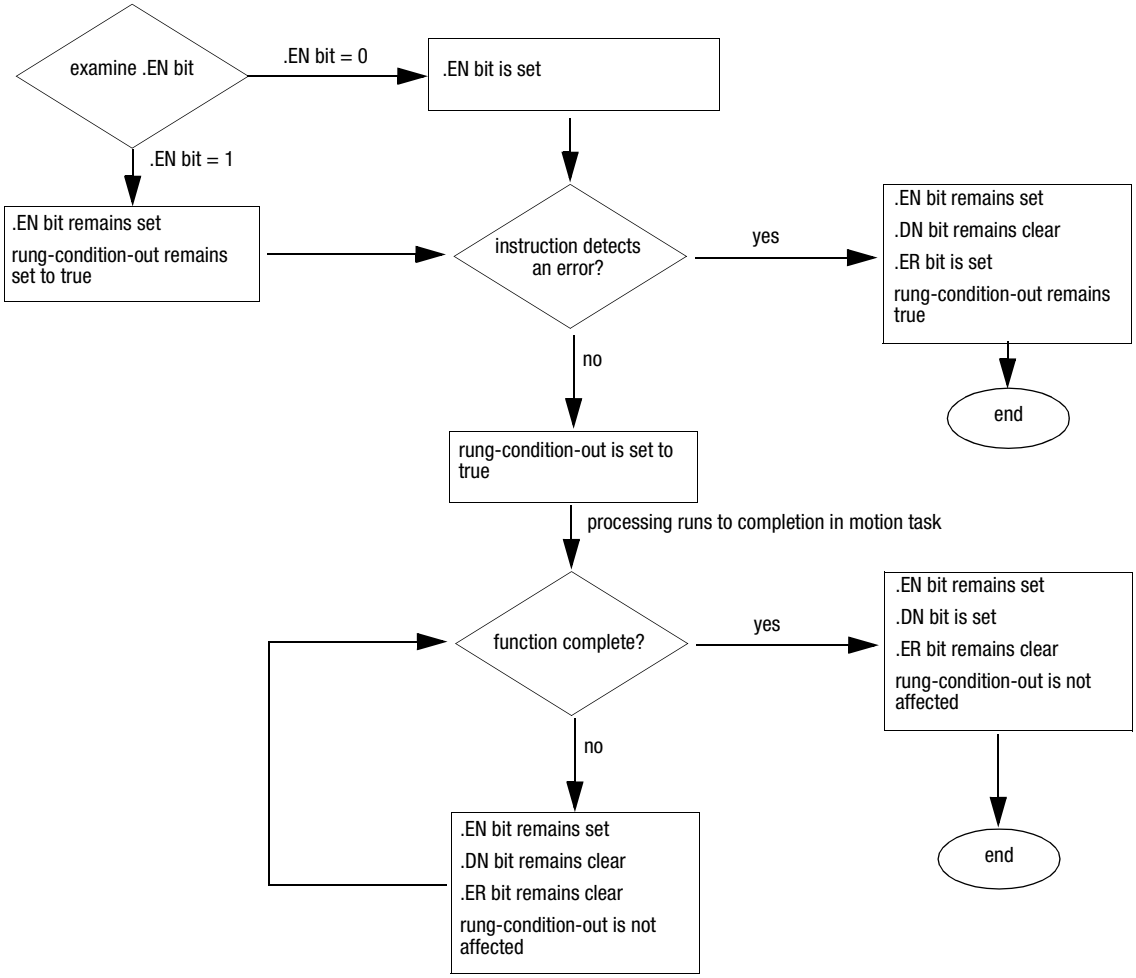


Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Type	DINT	immediate	select the type of operation: • absolute • relative
Position select	DINT	immediate	select what position to perform the redefinition operation on: • actual position • command position
Position	SINT, IN, DINT, or REAL	immediate or tag	value to change the axis position to or offset to current position

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction redefines the axis position.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

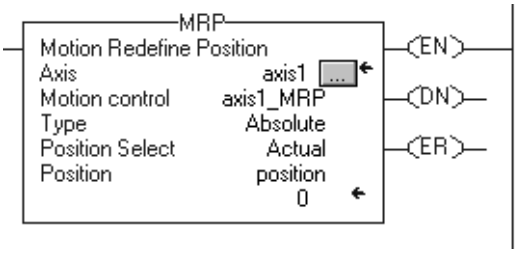
Arithmetic Status Flags: not affected

Fault Conditions: none

MRP Error Codes (.ERR):

Error Code:	Error Message	Description:
3	Execution Collision	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
8	Illegal Axis Type	The axis is not configured as a servo, position only, or virtual axis.
11	Axis Not Configured	The axis is not configured.
12	Servo Message Failure	Messaging to the servo module failed.
18	Axis Type Unused	The axis type is configured as unused.
19	Group Not Synchronized	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

MRP Example:



When the input conditions are true, the controller changes the position of *axis1*.

Other Formats:

Format:	Syntax:
neutral text	`MRP(a`*`xis,motion_control,type,position_select,position`*`);`
ASCII text	`MRP a`*`xis motion_control type position_select position`*

Motion Calculate Cam Profile (MCCP)

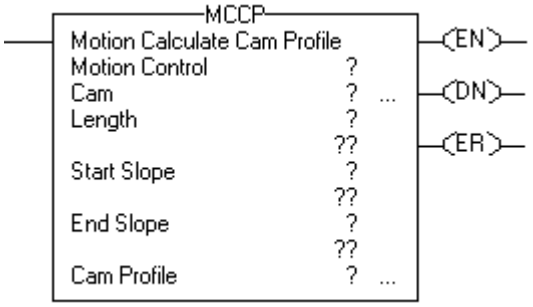
The MCCP instruction calculates a cam profile based on an array of cam points established programmatically or using the RSLogix5000 Cam Profile Editor. The primary purpose of an MCCP instruction is to let you calculate a cam profile in real time based on programmatic changes to the cam arrays. The MCCP instruction computes a cam profile based on a given set of points in a specified cam array. The cam profiles generated by this instruction are used by either MAPC or MATC camming instructions to perform complex motion of a slave axis in relation to a master axis position or to time.

To execute an MCCP instruction, you must configure a Cam array tag using either the RSLogix5000 Cam Profile Editor or the Cam Profile Editor.

The Cam array elements consist of slave and master point pairs and an interpolation type. The x and y point values are unitless because there is no association with a specific axis position or time. The interpolation type is specified for each point as either linear or cubic.

The MCCP instruction uses message type timing.

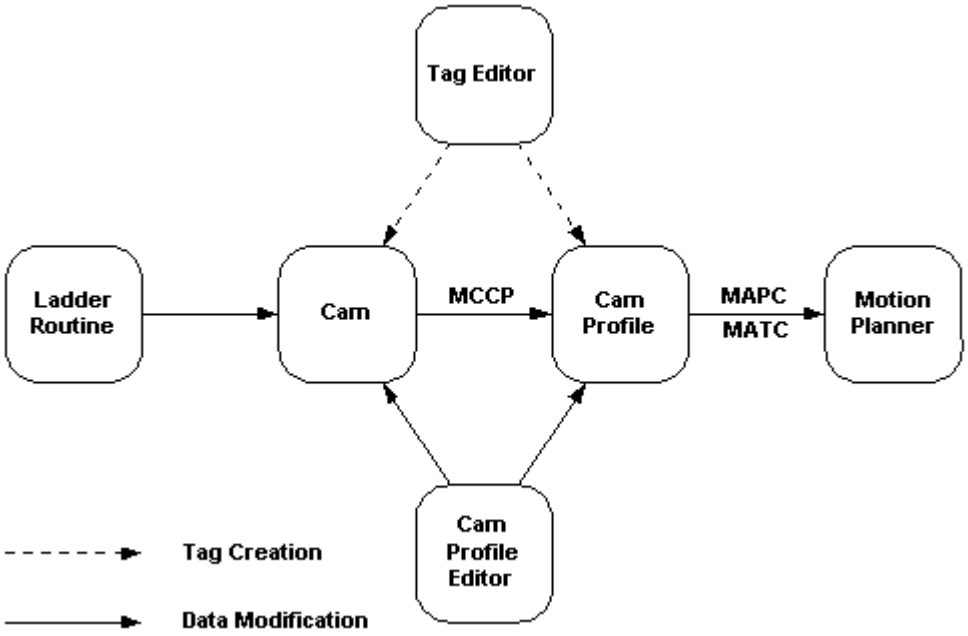
Operands:



Operand:	Type:	Format:	Description:
Motion control	MOTION_ INSTRUCTION	tag	Structure used to access block status parameters.
Cam	CAM	array	Tag name of the cam array used to compute the cam profile. The numerical array index indicates the stating cam element in the array used in the cam profile calculation. Ellipsis launches Cam Profile Editor.
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 is used to specify a slope through the first point.

Operand:	Type:	Format:	Description:
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 specify a slope through the last point.
Cam Profile	CAM_PROFILE	array	Tag name of the calculated cam profile array used as input to MAPC and MATC instructions. Ellipsis launches Cam Profile Editor.

Cam Operation Diagram



MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	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	BOOL	The done bit is set when the calculate cam instruction has been successfully executed and the Cam Profile array calculated.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the cam array is of an illegal length.

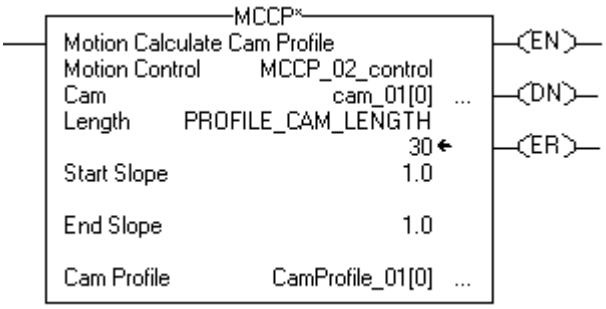
Arithmetic Status Flags: not affected

Fault Conditions: none

MCCP Error Codes (.ERR):

Error Code:	Error Message	Description:
26	Illegal Cam Length	The cam array is an illegal length.
27	Illegal Cam Profile Length	The cam profile array is of an illegal length.
28	Illegal Cam Type	You have an illegal segment type in the cam element.
29	Illegal Cam Order	You have an illegal order of cam elements.
30	Cam Profile Being Calculated	You tried to execute while a cam profile is being calculated.
31	Cam Profile Being Used	The cam profile array you tried to execute is in use.

MCCP Example:



Other Formats:

Format:	Syntax:
neutral text	`MCCP(motion_control,cam,length,start slope,end slope,cam profile);`
ASCII text	`MCCP motion_control cam length start slope end slope cam profile`

Motion Axis Position Cam (MAPC)

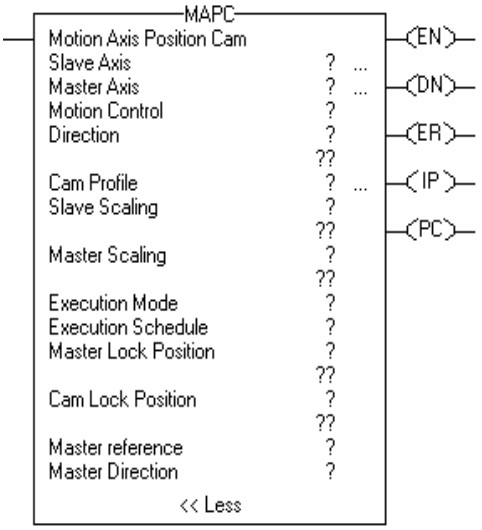
The MAPC function performs electronic camming between any two axes specified in the Cam Profile. Upon execution of this instruction the axis specified as the slave is synchronized to the axis designated as the master. Parameters for this instruction control direction, scaling, position, execution mode, and execution schedule.

The MAPC instruction executes a position cam profile set up by a previous MCCP block or by the RSLogix Cam Profile Editor. Position cams let you implement non-linear electronic gearing relationships between two axes. This instruction has no maximum limits for velocity, acceleration, or deceleration. The speed, acceleration, and deceleration are determined by the master axis motion and the designated cam profile.

To execute an MAPC instruction, a calculated Cam Profile data array must be specified.

The MAPC instruction uses immediate and process type timing.

Operands:

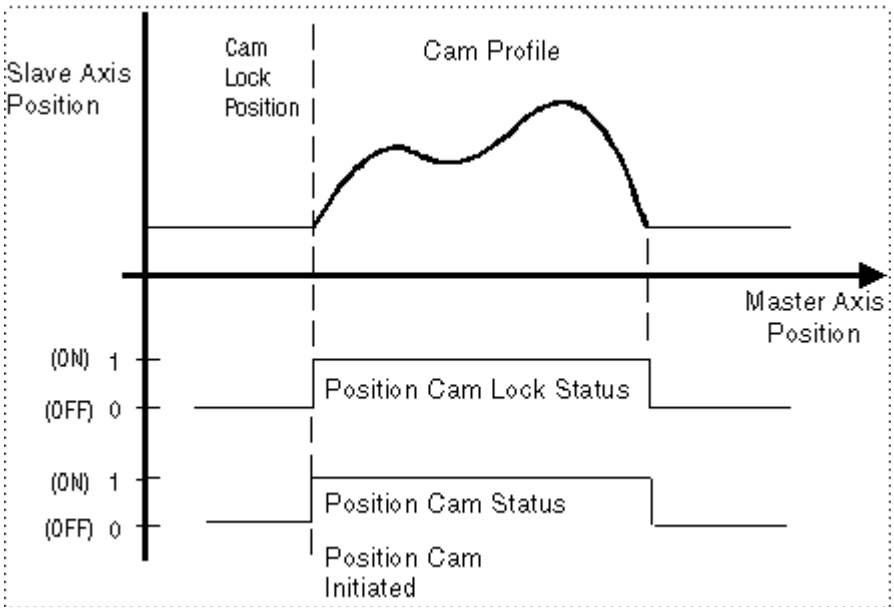


Operand:	Type:	Format:	Description:
Slave Axis	AXIS	tag	The name of the axis that the cam profile is applied to. Ellipsis launches Axis Properties dialog.
Master Axis	AXIS	tag	The axis that the slave axis follows according to the cam profile. Ellipsis launches Axis Properties dialog.
Motion Control	MOTION_ INSTRUCTION	tag	Structure used to access block status parameters.

Operand:	Type:	Format:	Description:
Direction	UINT32	immediate or tag	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 in the opposite sense of the master's. Or relative to the current or previous camming direction: • 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	array	Tag name of the calculated cam profile array used to establish the master/slave position relationship. Ellipsis launches Cam Profile Editor.
Slave Scaling	REAL	immediate or tag	Scales the total distance covered by the slave axis through the cam profile.
Master Scaling	REAL	immediate or tag	Scales the total distance covered by the master axis through the cam profile.
Execution Mode	UINT32		Determines if the cam profile is executed only one time or repeatedly: • Once – cam motion of slave axis starts only when the master axis moves into the range defined by the start and end points of the cam profile. When the master axis moves beyond the defined range cam motion on the slave axis stops and the Process Complete bit is set. Slave motion does not resume if the master axis moves back into the cam profile range. • 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	UINT32		Selects the method used to execute the cam profile. Options are: • Immediate – The slave axis is immediately locked to the master axis and the position camming process begins. • Forward only – the cam profile starts when the master position crosses the Master Lock Position in the forward direction. • Reverse only – the cam profile starts when the master position crosses the Master Lock Position in the reverse direction. • Bi-directional – the cam profile starts when the master position crosses the Master Lock Position in either direction. • Pending – lets you blend a new position cam execution after an in process position cam is finished.
Master Lock Position	REAL	immediate or tag	The Master axis absolute position where the slave axis locks to the master axis.
Cam Lock Position	REAL	immediate or tag	This determines the starting location in the cam profile.
Master Reference	UINT32		Sets the master position reference to either Command position or Actual position. • Actual – slave axis motion is generated from the current position of the master axis as measured by its encoder or other feedback device. • Command – slave axis motion is generated from the desired or commanded position of the master axis.

Operand:	Type:	Format:	Description:
Master Direction	UINT32		This determines the direction of the master axis that generates slave motion according to the cam profile. Options are: • Bi-directional – slave axis can track the master axis in either direction. • Forward only – slave axis tracks the master axis in the forward direction of the master axis. • Reverse only – slave axis tracks the master axis in the opposite direction of the master axis.

Position Cam Timing Diagram



MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit is set when the rung transitions from false-to-true and stays set until the rung goes false.
.DN	BOOL	The done bit is set when the axis position cam instruction is successfully initiated.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.
.IP	BOOL	The in process bit is set on positive rung transition and cleared when terminated by a stop command, merge, shutdown, servo fault or when the cam has completed.
.PC	BOOL	The Process Complete bit 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 currently active cam profile.
.ACCEL	BOOL	The .ACCEL bit indicates that the velocity has increased for the individual instruction that it is tied to i.e jog, move, gearing.
.DECEL	BOOL	The .DECEL bit indicates that the velocity has decreased for the individual instruction that it is tied to i.e jog, move, gearing.

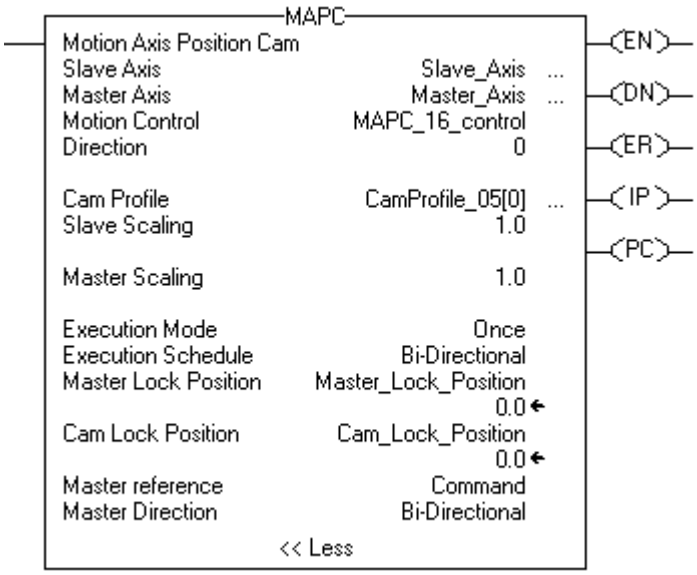
Arithmetic Status Flags: not affected

Fault Conditions: none

MAPC Error Codes (.ERR):

Error Code:	Error Message	Description:
5	Servo Off State Error	The servo loop wasn't closed at execution.
7	Shutdown State Error	The axis was in a shutdown state when you tried to execute it.
8	Illegal Axis Type	Your axis is not configured for servo or virtual functionality.
10	Master Axis Conflict	The master axis reference is the same as the slave axis reference.
11	Axis Not Configured	Your axis references an unconfigured axis.
13	Parameter Out of Range	An input parameter is out of range.
16	Home in Process Error	You tried to execute during a homing operation.
19	Group Not Synchronized	The axis and its associated axis group were not synchronized at time of execution.
23	Illegal Dynamic Change	You tried to execute while another Cam Profile was in process.
24	Illegal AC Mode Op	The controller attempted to execute an MDO, MSO, MAH, MAJ, MAM, MCD, MAPC, MATC, MAG, MRAT, or MRHD instruction when the controller was in the test mode.
32	Cam Profile Not Calculated	The cam profile array you tried to execute was not calculated.

MAPC Example:



Other Formats:

Format:	Syntax:
neutral text	`MAPC(slave_axis,master_axis,`*`motion_control,direction,cam_profile,slave_scaling,master_scaling,execution_mode,execution_schedule,master_lock_position,cam_lock_position,master_reference,master_direction`*`);`
ASCII text	`MAPC(slave_axis master_axis` *`motion_control direction cam_profile slave_scaling master_scaling execution_mode execution_schedule master_lock_position cam_lock_position master_reference master_direction`*`)`

Motion Axis Time Cam (MATC)

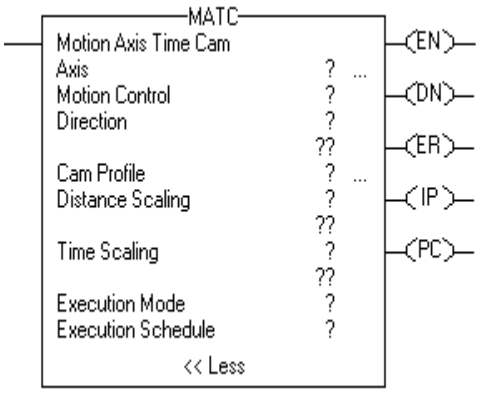
The MATC instruction performs electronic camming as a function of time based on the specified cam profile. Time cam lets you execute complex motion profiles in addition to the trapezoidal and S-curve move and jog profiles. At execution the specified axis is synchronized using the specified time cam profile. The parameters for this instruction let you establish the direction, distance scaling, time scaling, execution mode, and execution schedule for the camming.

The MATC instruction executes a time cam profile set up by a previous MCCP instruction or by the RSLogix5000 Cam Profile Editor. Time cams let you implement complex motion profiles other than the built-in trapezoidal and S-curve motion profiles. There are no maximum limits for velocity, acceleration, or deceleration in this instruction. The speed, acceleration, and deceleration of the slave axis are completely governed by the designated cam profile and the scaling values.

To execute an MATC instruction, a calculated Cam Profile data array tag must be specified.

The MATC instruction uses immediate and process type timing.

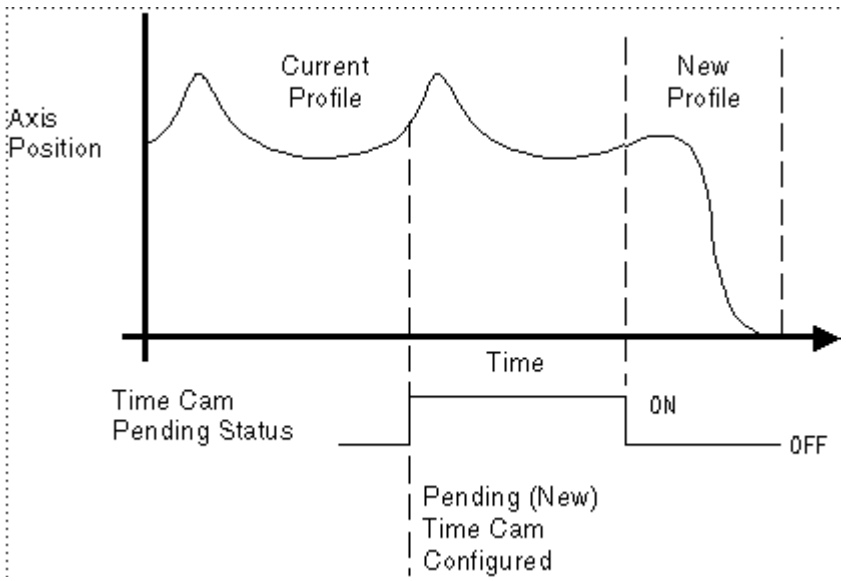
Operands:



Operand:	Type:	Format:	Description:
Axis	AXIS	tag	The name of the axis to which the cam profile is applied. Ellipsis launches Axis Properties dialog.
Motion Control	MOTION_ INSTRUCTION	tag	Structure used to access block status parameters.

Operand:	Type:	Format:	Description:
Direction	UINT32	immediate or tag	Relative direction of the slave axis to the master axis: • 0 = Same – the axis position values in the cam profile are added to the command position of the axis. • 1 = Opposite – the axis position values in the cam profile are subtracted from the command position of the axis creating axis motion in the other direction from that implied in the original cam table. Or relative to the current or previous camming direction: • 2 = Reverse – the current or previous direction of the position cam is changed either from Same to Opposite or vice versa. 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	array	Tag name of the calculated cam profile array. Ellipsis launches Cam Profile Editor.
Distance Scaling	REAL	immediate or tag	Scales the total distance covered by the axis through the cam profile.
Time Scaling	REAL	immediate or tag	Scales the time interval covered by the cam profile.
Execution Mode	UINT32		Determines how the cam motion behaves when the time moves beyond the end point of the cam profile. The options are: • Once – When the time cam execution time exceeds the time range in the cam profile, the MATC instruction completes, the axis motion stops, and the Time Cam Status bit is cleared. • Continuous – The cam profile motion is executed indefinitely.
Execution Schedule	UINT32		Selects the method used to execute the cam profile. Options are: • Immediate – instruction is scheduled to execute immediately with no delay enabling the time camming process. • Pending – Defers execution of the time cam until the completion of the currently or next immediate executing time cam. This is useful in blending a new time cam profile with an on going process to achieve a seamless transition.

Time Cam Timing Diagram:



MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit is set when the rung transitions from false-to-true and stays set until the rung goes false.
.DN	BOOL	The done bit is set when the axis time cam instruction is successfully initiated.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.
.IP	BOOL	The in process bit is set on positive rung transition and cleared when terminated by a stop command, merge, shutdown, or servo fault.
.PC	BOOL	The Process Complete bit is cleared on positive rung transition and set in Once Execution Mode, when the time leaves the time range defined by the currently active cam profile.
.ACCEL	BOOL	The .ACCEL bit indicates that the velocity has increased for the individual instruction that it is tied to i.e jog, move, gearing.
.DECEL	BOOL	The .DECEL bit indicates that the velocity has decreased for the individual instruction that it is tied to i.e jog, move, gearing.

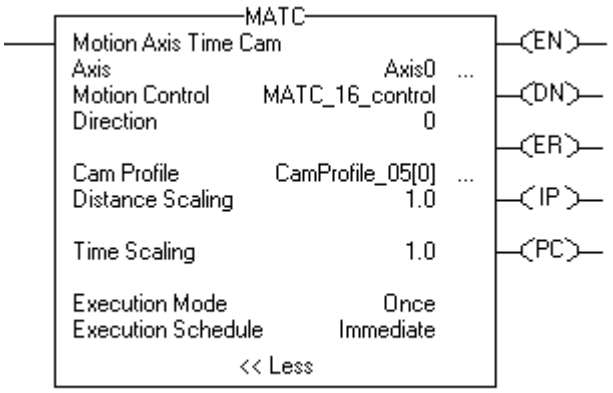
Arithmetic Status Flags: not affected

Fault Conditions: none

MATC Error Codes (.ERR):

Error Code:	Error Message	Description:
5	Servo Off State Error	The servo loop was not closed at execution.
7	Shutdown State Error	The axis was in a shutdown state when you tried to execute.
8	Illegal Axis Type	Your axis is not configured for servo or virtual functionality.
11	Axis Not Configured	Your axis value references an unconfigured axis.
13	Parameter Out of Range	An input parameter is out of range.
16	Home In Process Error	You tried to execute during a homing operation.
19	Group Not Synchronized	The axis and its associated axis group were not synchronized at time of execution.
23	Illegal Dynamic Change	You tried to execute while another Cam Profile was in process.
24	Illegal AC Mode Op	The controller attempted to execute an MDO, MSO, MAH, MAJ, MAM, MCD, MAPC, MATC, MAG, MRAT, or MRHD instruction when the controller was in the test mode.
32	Cam Profile Not Calculated	The cam profile array you tried to execute was not calculated.

MATC Example:



Other Formats:

Format:	Syntax:
Neutral text	`MATC(axis,motion_control,direction,cam_profile,distance_scaling, time_scaling,execution_mode,execution_schedule);`
ASCII text	`MATC(axis motion_control direction cam_profile distance_scaling time_scaling execution_mode execution_schedule)`

Motion Group Instructions

(MGS, MGPS, MGSD, MGSR, MGSP)

ATTENTION: 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 of the control variables.

Introduction

Motion group instructions control a group of axes. The motion group instructions are:

If you want to:	Use this instruction:	See page:
Initiate a stop of motion on a group of axes.	MGS	4-2
Initiate a stop of all motion on all the axes in a group, using the method that you set for each axis.	MGPS	4-6
Force all axes in a group into the shutdown operating state.	MGSD	4-10
Transition a group of axes from the shutdown operating state to the axis ready operating state.	MGSR	4-14
Latch the current command and actual position of all axes in a group.	MGSP	4-17

Motion Group Stop (MGS)

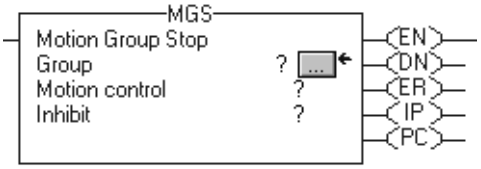
The MGS instruction is an output instruction.

Use the MGS instruction to initiate a stop on the motion on a group of axes.

The MGS instruction uses process type timing.

To use the MGS instruction, the group must be configured.

Operands:

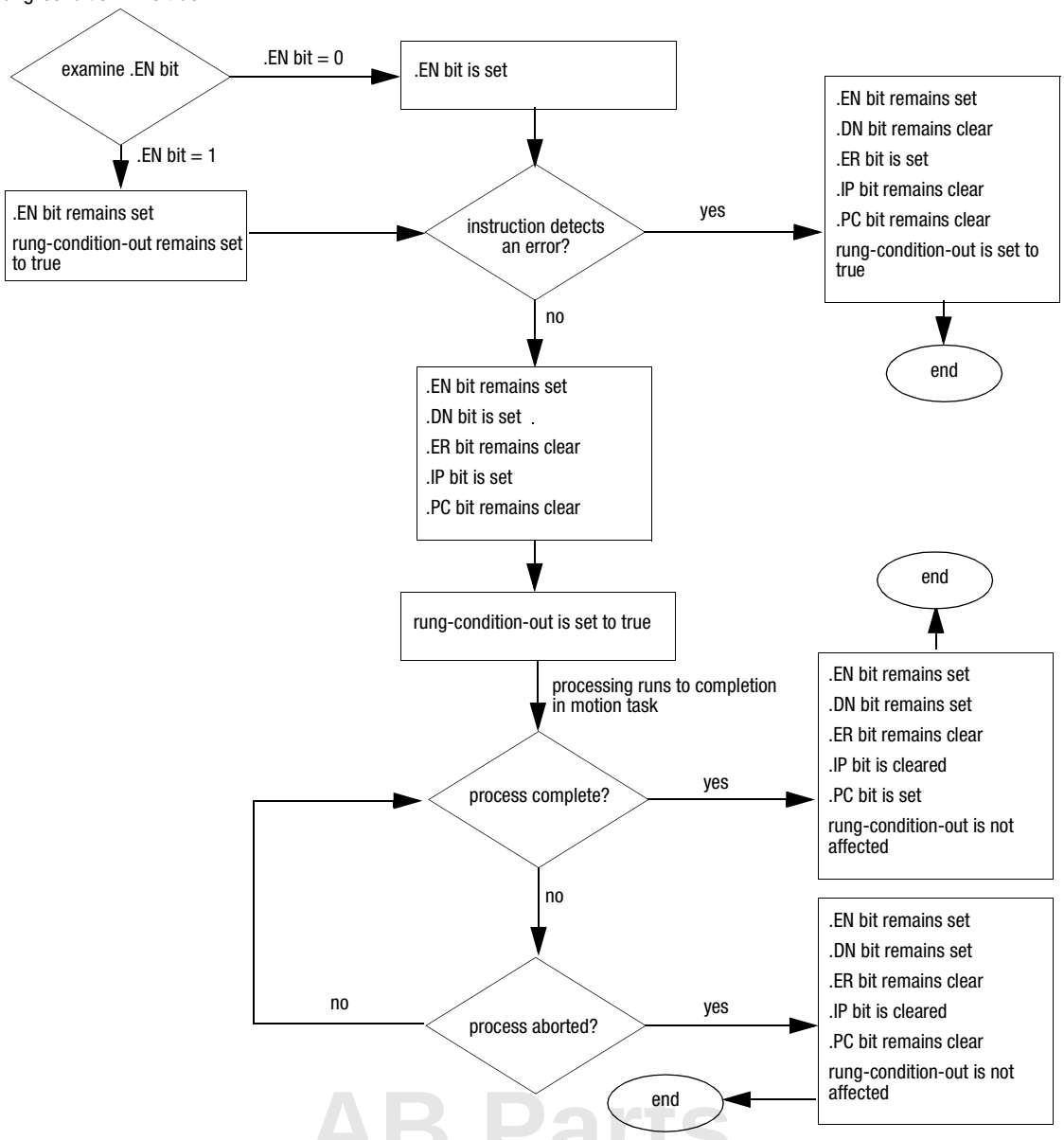


Operand:	Type:	Format:	Description:
Group	MOTION_ GROUP	tag	group structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Inhibit	DINT	immediate	select whether the group of axes servo loops are opened after they stop: • disabled • enabled

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction initiates a group stop for all the axes in a group.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if messaging to the servo module failed.
.IP	BOOL	• The in process bit sets when the motion group stop is successfully initiated. • It resets when the MGS instruction stops all the axes in the group and disables feedback (when the inhibit option is selected).
.PC	BOOL	The process complete bit sets after the instruction stops all the axes. If the Inhibit option is selected, the .PC bit sets after the instruction sets all the axes to the axis ready state.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The .IP bit is cleared. The .PC bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The .IP bit is not affected. The .PC bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MGS Error Codes (.ERR):

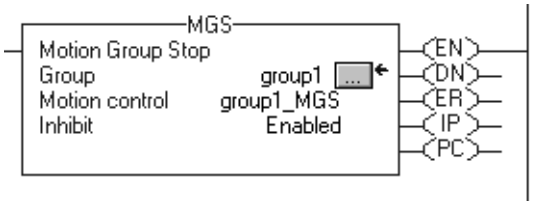
Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the controller executes a messaging instruction without checking the .DN bit of the preceding instruction.
12	Messaging to the servo module failed.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

MGS Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
StoppingStatus	True	The axis is stopping.
JogStatus	False	The axis is not jogging.
MoveStatus	False	The axis is not moving.
GearingStatus	False	The axis is not gearing.
HomingStatus	False	The axis is not homing.
DecelStatus	True	The axis is decelerating.
ServoActStatus	False[1]	The axis is in the axis ready state. The servo loop is inactive.
DriveEnableStatus	False[1]	The drive enable output is inactive.
PositionCamStatus	False	Pcam motion profile is not in progress.
TimeCamStatus	False	Tcam motion profile is not in progress.
PositionCamLockedStatus	False	The Pcam is stopped and the lock is cleared.
TimeCamLockedStatus	False	The Tcam is stopped and the lock is cleared.
PositionCamPendingStatus	False	The pending PCAM profile is cancelled.
TimeCamPendingStatus	False	The pending Tcam profile is cancelled.

1 If you selected the inhibit option, the MGS instruction will change this status bit.

MGS 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.

Other Formats:

Format:	**Syntax:**
neutral text	`MGS(group,motion_control,inhibit);`
ASCII text	`MGS group motion_control inhibit`

Motion Group Program Stop (MGPS)

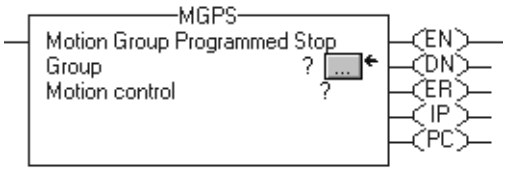
The MGPS instruction is an output instruction.

Use the MGPS instruction to initiate a stop of all motion on all the axes in a group. The instruction stops each axis using a method that you set for each axis. This method is contained in the ProgrammedStopMode attribute.

The MGPS instruction uses message and process type timing.

To use the MGPS instruction, the group must be configured.

Operands:

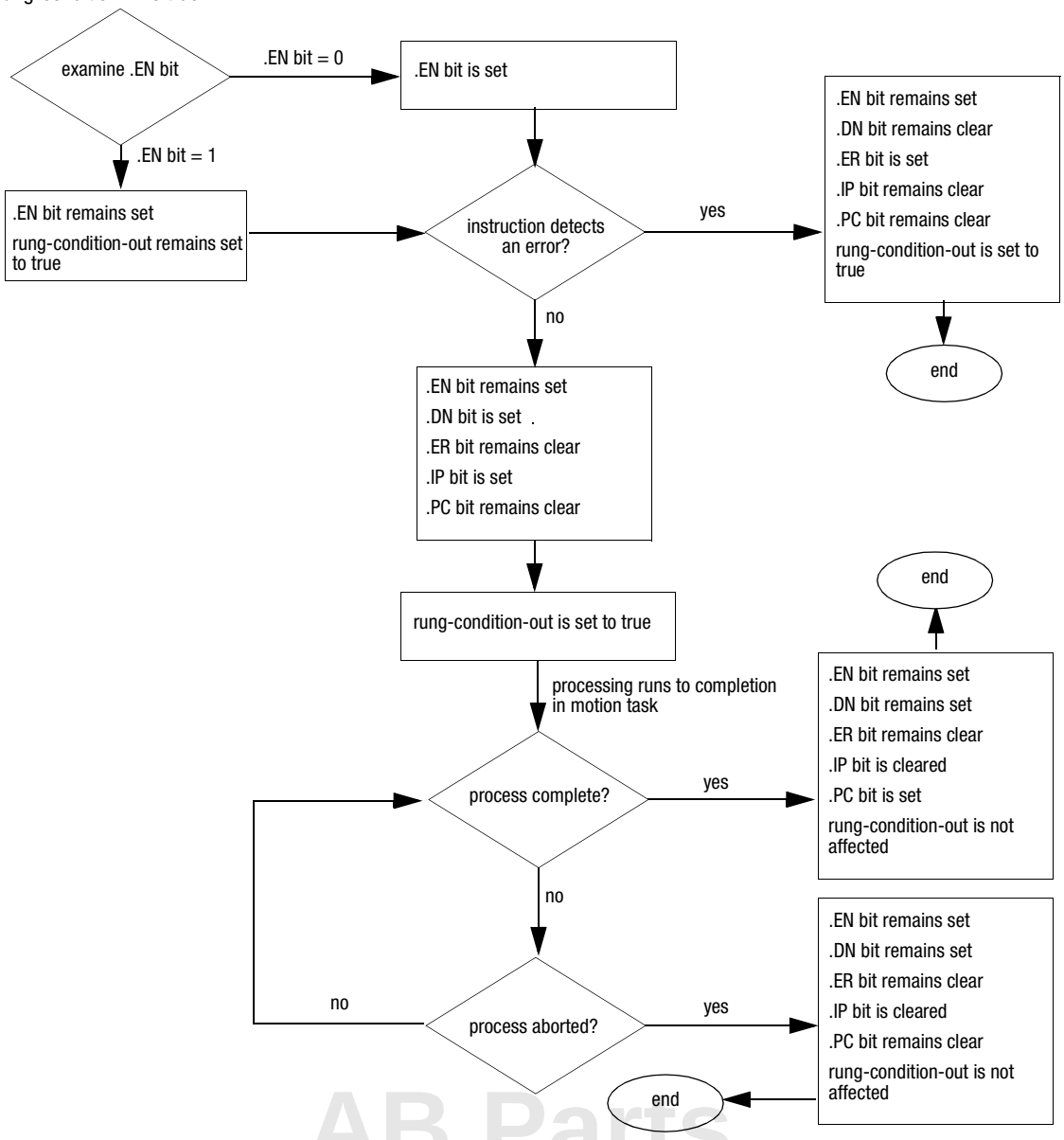


Operand:	Type:	Format:	Description:
Group	MOTION_ GROUP	tag	group structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the messaging completes. The length of time to complete messaging is based on the programmed stop mode configuration of the axes in the group.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if messaging to the servo module failed.
.IP	BOOL	• The in process bit sets when the group programmed stop is successfully initiated. • It resets when the commanded motion process completes.
.PC	BOOL	The process complete bit sets after the instruction stops each of the axes in the group according to the axis' programmed stop mode.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The .IP bit is cleared. The .PC bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The .IP bit is not affected. The .PC bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MGPS Error Codes (.ERR):

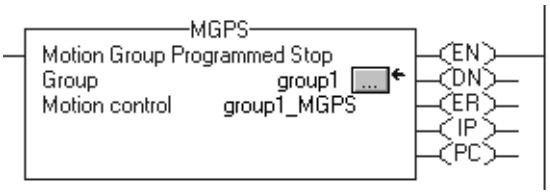
Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the controller executes a messaging instruction without checking the .DN bit of the preceding instruction.
12	Messaging to the servo module failed.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

MGPS Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
StoppingStatus	True	The axis is stopping.
JogStatus	False	The axis is not jogging.
MoveStatus	False	The axis is not moving
GearingStatus	False	The axis is not gearing.
HomingStatus	False	The axis is not homing.
AccelStatus	False	The axis is not accelerating.
TuneStatus	False	The axis is not running a tuning process.
TestStatus	False	The axis is not running a testing process.
DecelStatus	True	The axis is decelerating.
ShutdownStatus	True/False[1]	Depends on the programmed stop mode for each axis.
ServoActStatus	True/False[1]	Depends on the programmed stop mode for each axis.
DriveEnableStatus	True/False[1]	Depends on the programmed stop mode for each axis.
PositionCamStatus	False	Pcam motion profile is not in progress.
TimeCamStatus	False	Tcam motion profile is not in progress.
PositionCamLockedStatus	False	The Pcam is stopped and the lock is cleared.
TimeCamLockedStatus	False	The Tcam is stopped and the lock is cleared.
PositionCamPendingStatus	False	The pending PCAM profile is cancelled.
TimeCamPendingStatus	False	The pending Tcam profile is cancelled.
GearingLockedStatus	False	The axis is not clutching to a new gear speed.

1 This bit is set or cleared depending on the the type of programmed stop selected for each axis.

MGPS Example:



When the input conditions are true, the controller stops motion on all axes in *group1*, using a method that you set for each axis in its programmed stop action selection.

Other Formats:

Format:	**Syntax:**
neutral text	`MGPS(group,motion_control);`
ASCII text	`MGPS group motion_control`

Motion Group Shutdown (MGSD)

The MGSD instruction is an output instruction.

Use the MGSD instruction to force all axes in a group into the shutdown operating state.

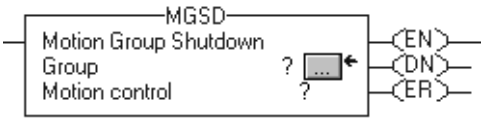
The shutdown state of an axis means:

- The servo is disabled.
- The drive enable output is immediately deactivated.
- The servo output level sets to the output offset value.
- The servo module OK relay contacts are open.

The MGSD instruction uses message type timing.

To use the MGSD instruction, the group must be configured.

Operands:

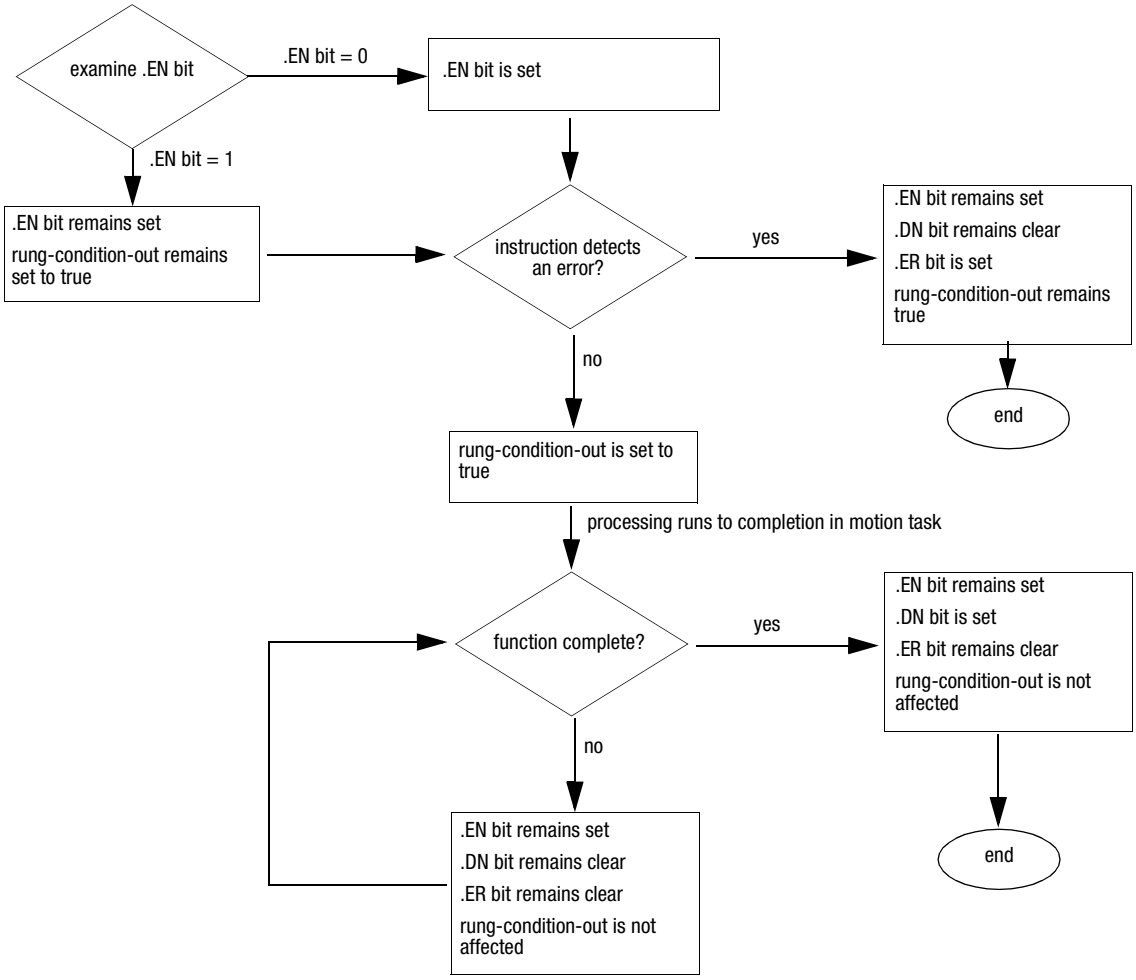


Operand:	Type:	Format:	Description:
Group	MOTION_ GROUP	tag	group structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction sets the group of axes to the shutdown operating state.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if messaging to the servo module failed.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MGSD Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the controller executes a messaging instruction without checking the .DN bit of the preceding instruction.
12	Messaging to the servo module failed.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

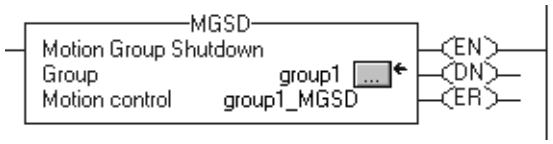
MGSD Changes to Servo Module LEDs:

This LED:	Will change to:	Meaning:
FDBK	Flashing green	Servo action is disabled.
DRIVE	Flashing red	• The drive enable output is inactive. • The OK contact is open.

MGSD Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
ServoActStatus	False	• Axis is in the axis ready state. • Servo loop is inactive.
DriveEnableStatus	False	The drive enable output is inactive.
ShutdownStatus	True	The axis is in the shutdown state.
AccelStatus	False	The axis is not accelerating.
DecelStatus	False	The axis is not decelerating.
StoppingStatus	False	The axis is not stopping.
JogStatus	False	The axis is not jogging.
MoveStatus	False	The axis is not moving.
GearingStatus	False	The axis is not gearing.
HomingStatus	False	The axis is not homing.
TuneStatus	False	The axis is not running a tuning process.
TestStatus	False	The axis is not running a testing process.
GearingLockedStatus	False	The axis is not clutching to a new gear speed.
PositionCamStatus	False	Pcam motion profile is not in progress.
TimeCamStatus	False	Tcam motion profile is not in progress.
PositionCamLockedStatus	False	The Pcam is stopped and the lock is cleared.
TimeCamLockedStatus	False	The Tcam is stopped and the lock is cleared.
PositionCamPendingStatus	False	The pending PCAM profile is cancelled.
TimeCamPendingStatus	False	The pending Tcam profile is cancelled.

MGSD Example:



When the input conditions are true, the controller forces all axes in *group1* into a shutdown operating state.

Other Formats:

Format:	Syntax:
neutral text	`MGSD(group,motion_control);`
ASCII text	`MGSD group motion_control`

Motion Group Shutdown Reset (MGSR)

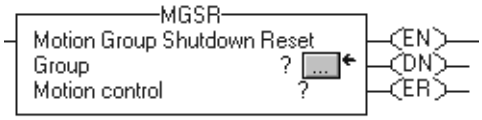
The MGSR instruction is an output instruction.

Use the MGSR instruction to transition a group of axes from the shutdown operating state to the axis ready operating state.

The MGS instruction uses message type timing.

To use the MGSR instruction, the group must be configured.

Operands:

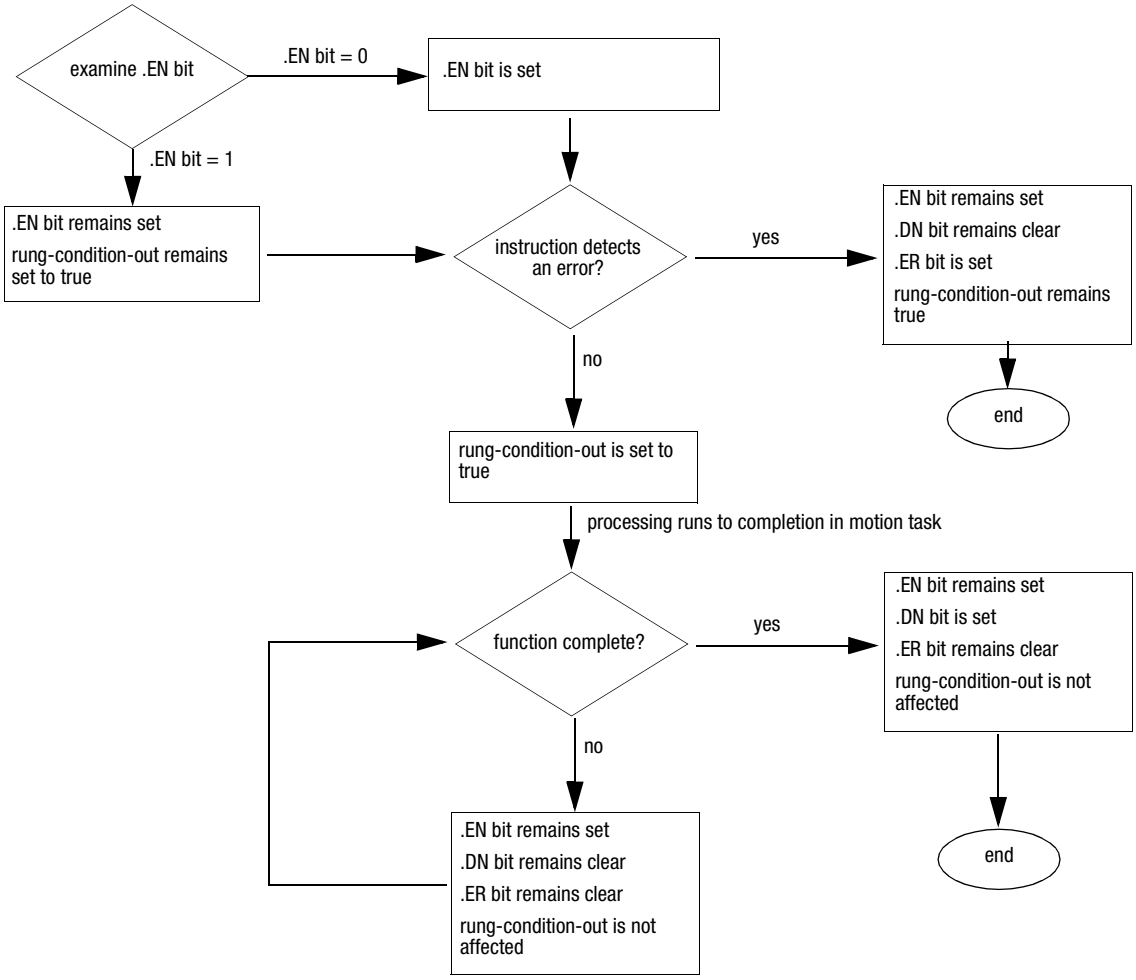


Operand:	Type:	Format:	Description:
Group	MOTION_ GROUP	tag	group structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction resets the group of axes from the shutdown operating state.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if messaging to the servo module failed.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MGSR Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the controller executes a messaging instruction without checking the .DN bit of the preceding instruction.
12	Messaging to the servo module failed.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

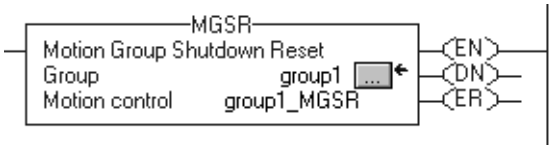
MGSR Changes to Servo Module LEDs:

This LED:	Will change to:	Meaning:
FDBK	Flashing green	Servo action is disabled.
DRIVE	Flashing green	• The drive enable output is inactive. • The OK contact is closed.

MGSR Changes to Axis Status Bits:

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

MGSR Example:



When the input conditions are true, the controller transitions all axes in *group1* from the shutdown operating state to the axis ready operating state.

Other Formats:

Format:	Syntax:
neutral text	`MGSR(group,motion_control);`
ASCII text	`MGSR group motion_control`

Motion Group Strobe Position (MGSP)

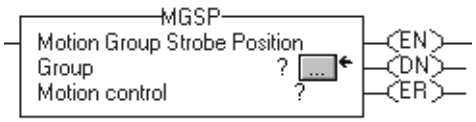
The MGSP instruction is an output instruction.

Use the MGSP instruction to latch the current command and actual position of all axes in a group. These values are latched in the StrobeActualPosition and StrobeCommandPosition attributes. You can read these values by using the GSV instruction.

The MGSP instruction uses immediate type timing.

To use the MGSP instruction, the group must be configured.

Operands:

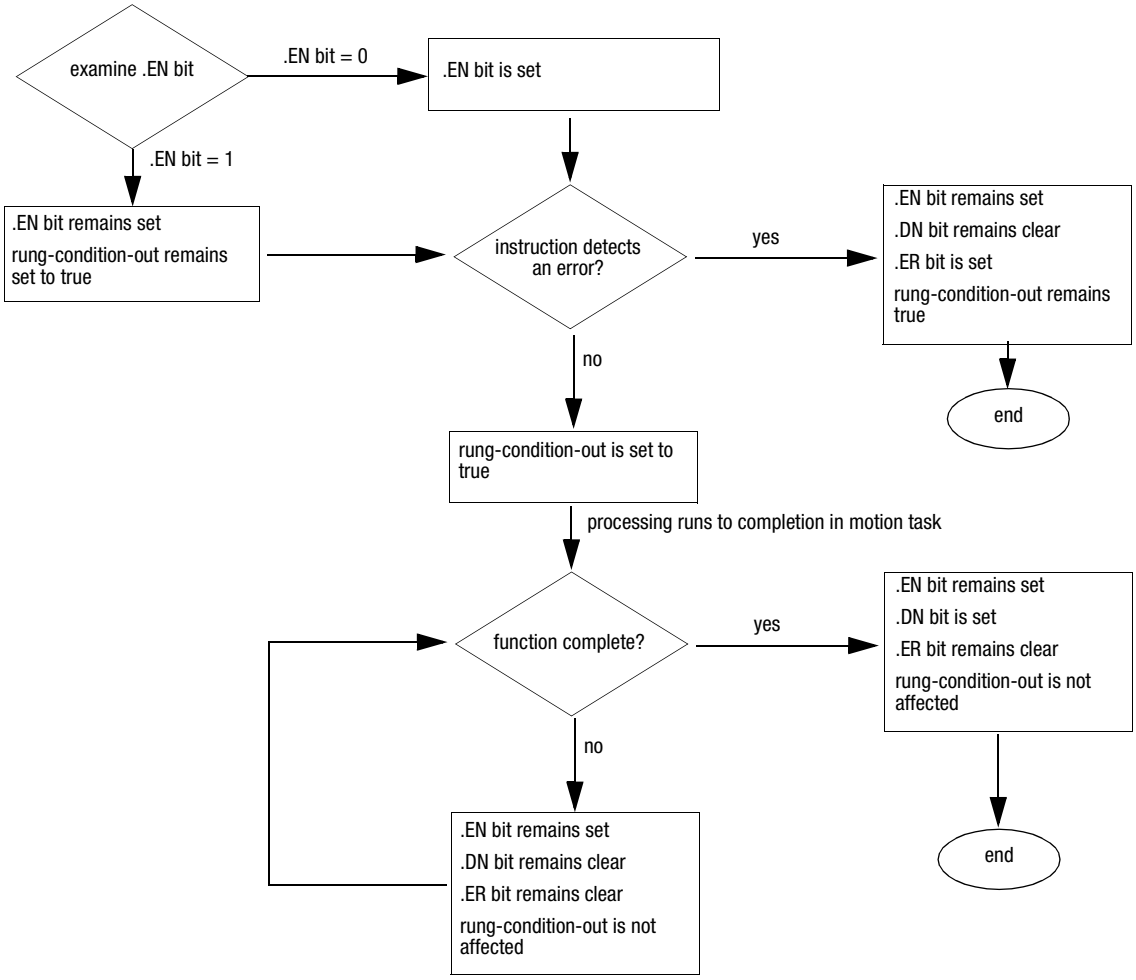


Operand:	Type:	Format:	Description:
Group	MOTION_ GROUP	tag	group structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled.
.DN	BOOL	The done bit indicates when the instruction latches the current Command and Actual positions of all the axes in a group.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the group is not synchronized.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

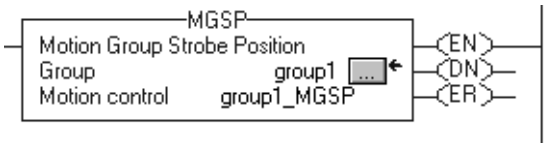
Arithmetic Status Flags: not affected

Fault Conditions: none

MGSR Error Codes (.ERR):

Error Code:	Description:
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

MGSP Example:



When the input conditions are true, the controller latches the current command and the actual position of all axes in *group1*.

Other Formats:

Format:	Syntax:
neutral text	`MGSP(group,motion_control);`
ASCII text	`MGSP group motion_control`

Motion Event Instructions

(MAW, MDW, MAR, MDR)

ATTENTION: 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 of the control variables.

Introduction

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:	See page:
Arm watch-position event-checking for an axis.	MAW	5-2
Disarm watch-position event-checking for an axis.	MDW	5-5
Arm servo-module registration-event checking for an axis.	MAR	5-8
Disarm servo-module registration-event checking for an axis.	MDR	5-12

Motion Arm Watch (MAW)

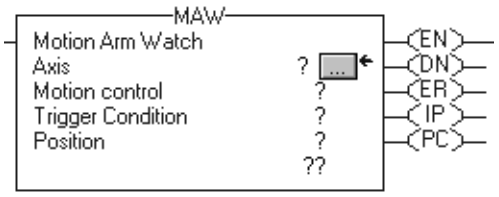
The MAW instruction is an output instruction.

Use the MAW instruction to arm watch-position event-checking for an axis.

The MAW instruction uses message and process type timing.

To use the MAW instruction, configure the axis as either a servo axis or a position only axis.

Operands:



Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Trigger condition	DINT	immediate	select the watch-event trigger condition: • forward The servo module looks for the actual position to change from less than the watch position to greater than the watch position. • reverse The servo module looks for the actual position to change from greater than the watch position to less than the watch position.
Position	SINT, INT, DINT, or REAL	immediate or tag	new value for the watch position

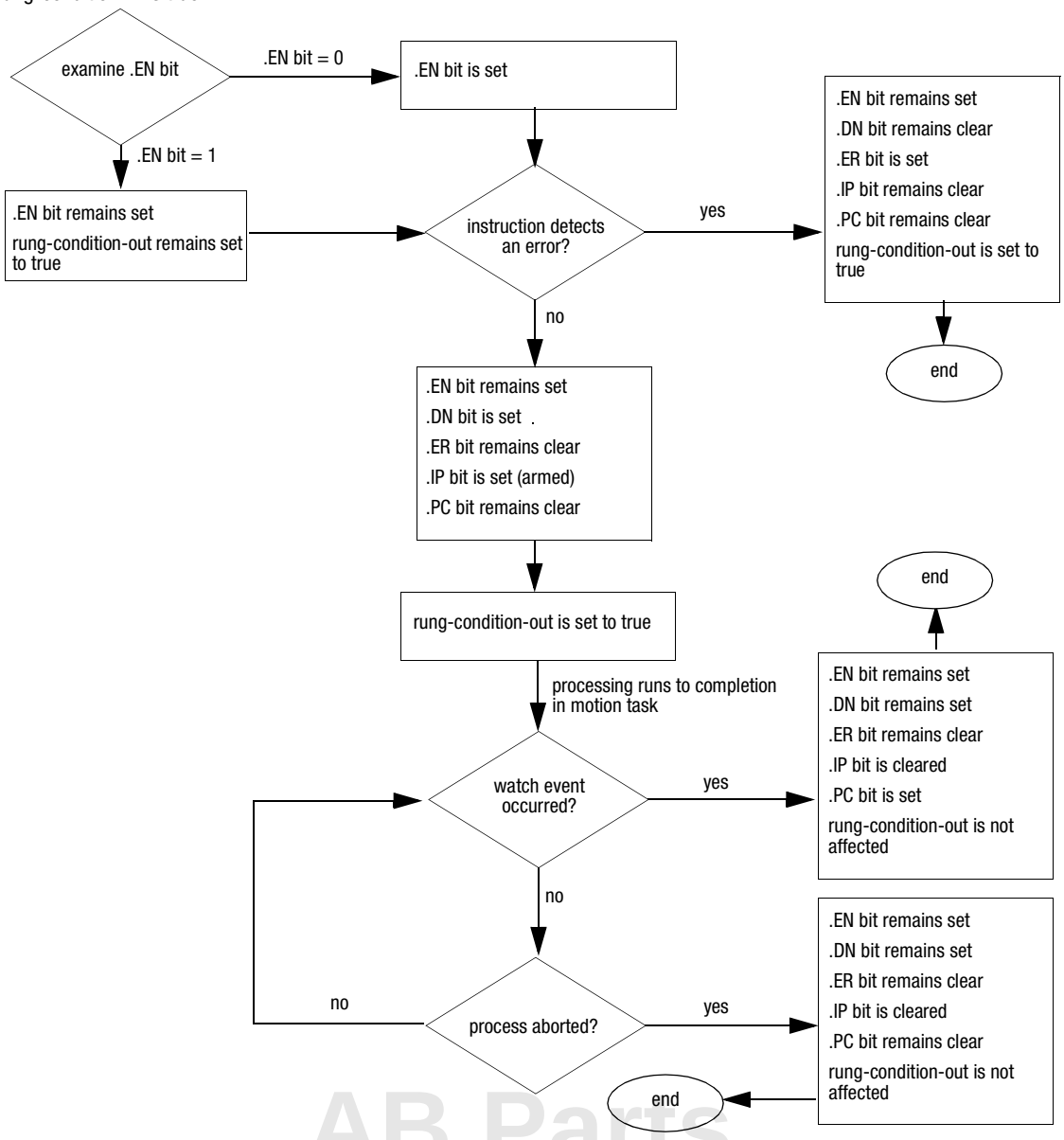
MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction arms watch-event checking.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

.IP	BOOL	• The in process bit sets when the MAW instruction is successfully initiated.
		• It resets when one of the following events occurs:
		  • A watch event occurs.
		  • Another MAW instruction supersedes the current instruction.
		  • A Motion Disarm Watch (MDW) instruction terminates the MAW instruction.

.PC	BOOL	The process complete bit is set when a watch event occurs.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The .IP bit is cleared. The .PC bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The .IP bit is not affected. The .PC bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

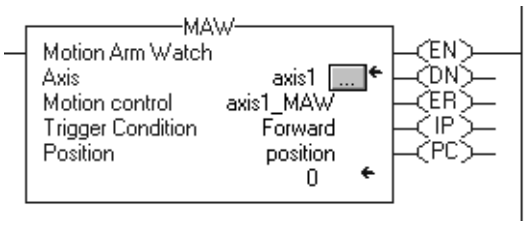
MAW Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
8	The axis is not configured as a servo or position only axis.
11	The axis is not configured.
12	Messaging to the servo module failed.
18	The axis type is configured as unused.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

MAW Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
WatchEvArmStatus	True	The axis is looking for a watch position event.
WatchEvStatus	False	The previous watch event is cleared.

MAW Example:



When the input conditions are true, the controller arms watch-position event-checking for *axis1*.

Other Formats:

Format:	Syntax:
neutral text	`MAW(axis,motion_control,trigger,position);`
ASCII text	`MAW axis motion_control trigger position`

Motion Disarm Watch (MDW)

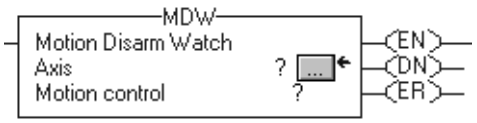
The MDW instruction is an output instruction.

Use the MDW instruction to disarm watch-position event-checking for an axis.

The MDW instruction uses message type timing.

To use the MDW instruction, configure the axis as either a servo axis or a position-only axis.

Operands:



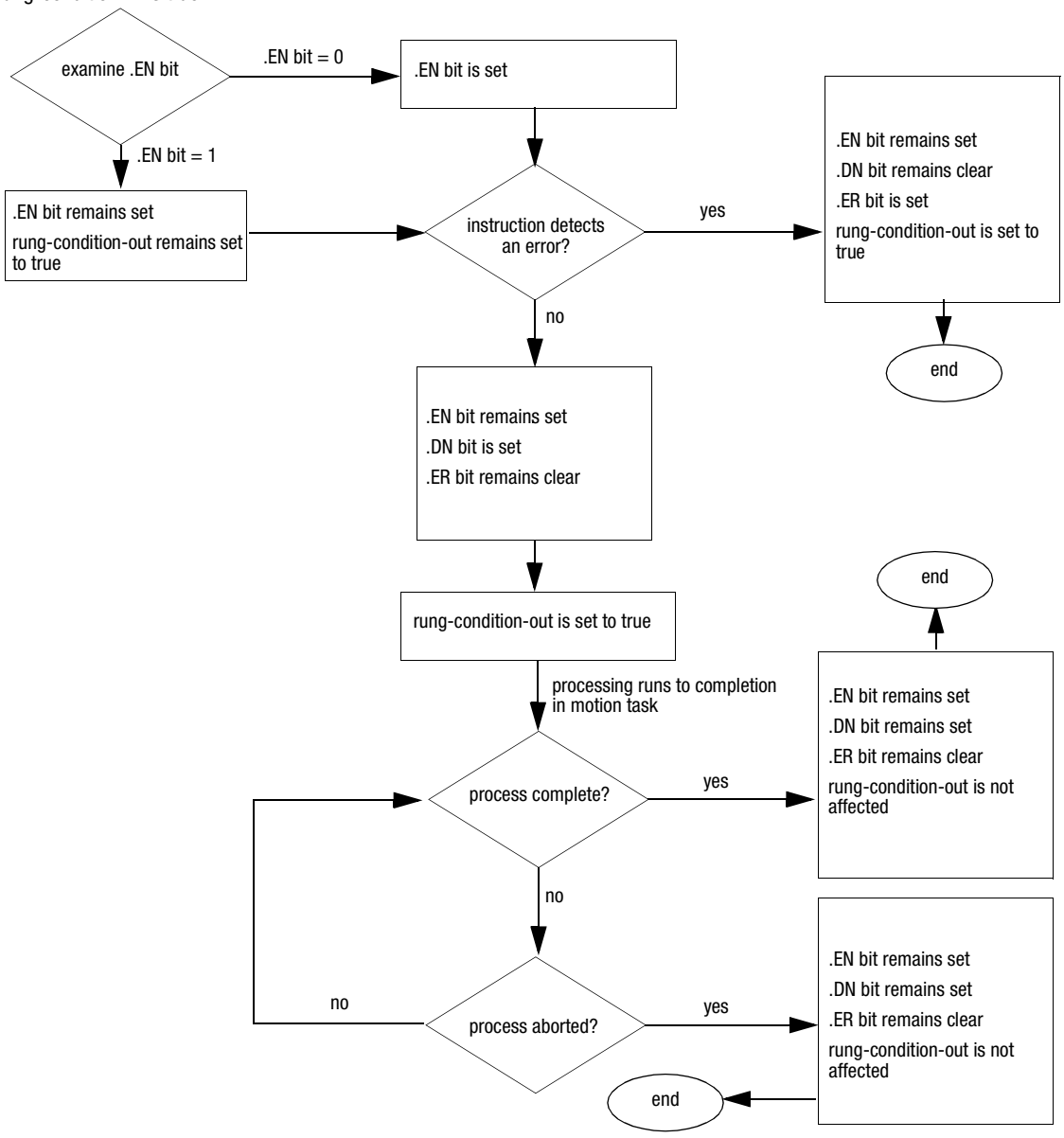
Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction disarms watch-event checking.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

When you use this instruction, the controller clears the watch-event status bit and the watch-armed status bits. This instruction also clears the .IP bit in the control structure of the MAW instruction.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

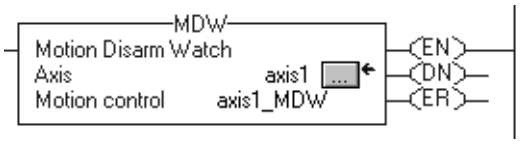
MDW Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
8	The axis is not configured as a servo or position only axis.
11	The axis is not configured.
12	Messaging to the servo module failed.
18	The axis type is configured as unused.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

MDW Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
WatchEvArmStatus	False	The axis is not looking for a watch position event.
WatchEvStatus	False	The previous watch event is cleared.

MDW Example:



When the input conditions are true, the controller disarms watch-position event-checking for *axis1*.

Other Formats:

Format:	Syntax:
neutral text	`MDW(axis,motion_control);`
ASCII text	`MDW axis motion_control`

Motion Arm Registration (MAR)

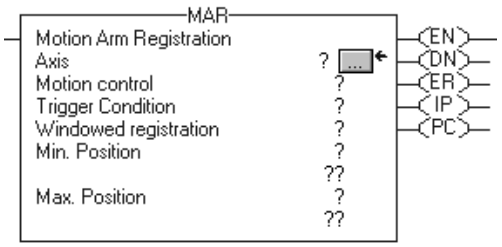
The MAR instruction is an output instruction.

Use the MAR instruction to arm servo module registration event checking for an axis.

The MAR instruction uses message and process type timing.

To use the MAR instruction, configure the axis as either a servo axis or a position only axis.

Operands:



Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Trigger condition	DINT	immediate	select the input condition that defines the registration event: • trigger on positive edge • trigger on negative edge
Windowed registration	DINT	immediate	select whether the registration position is within the position window: • disabled • enabled
Min. position	SINT, INT, DINT, or REAL	immediate or tag	minimum position for windowed registration event checking the registration position must be greater than this value before the controller accepts the registration event
Max. position	SINT, INT, DINT, or REAL	immediate or tag	maximum position for windowed registration event checking the registration position must be less than this value before the controller accepts the registration event

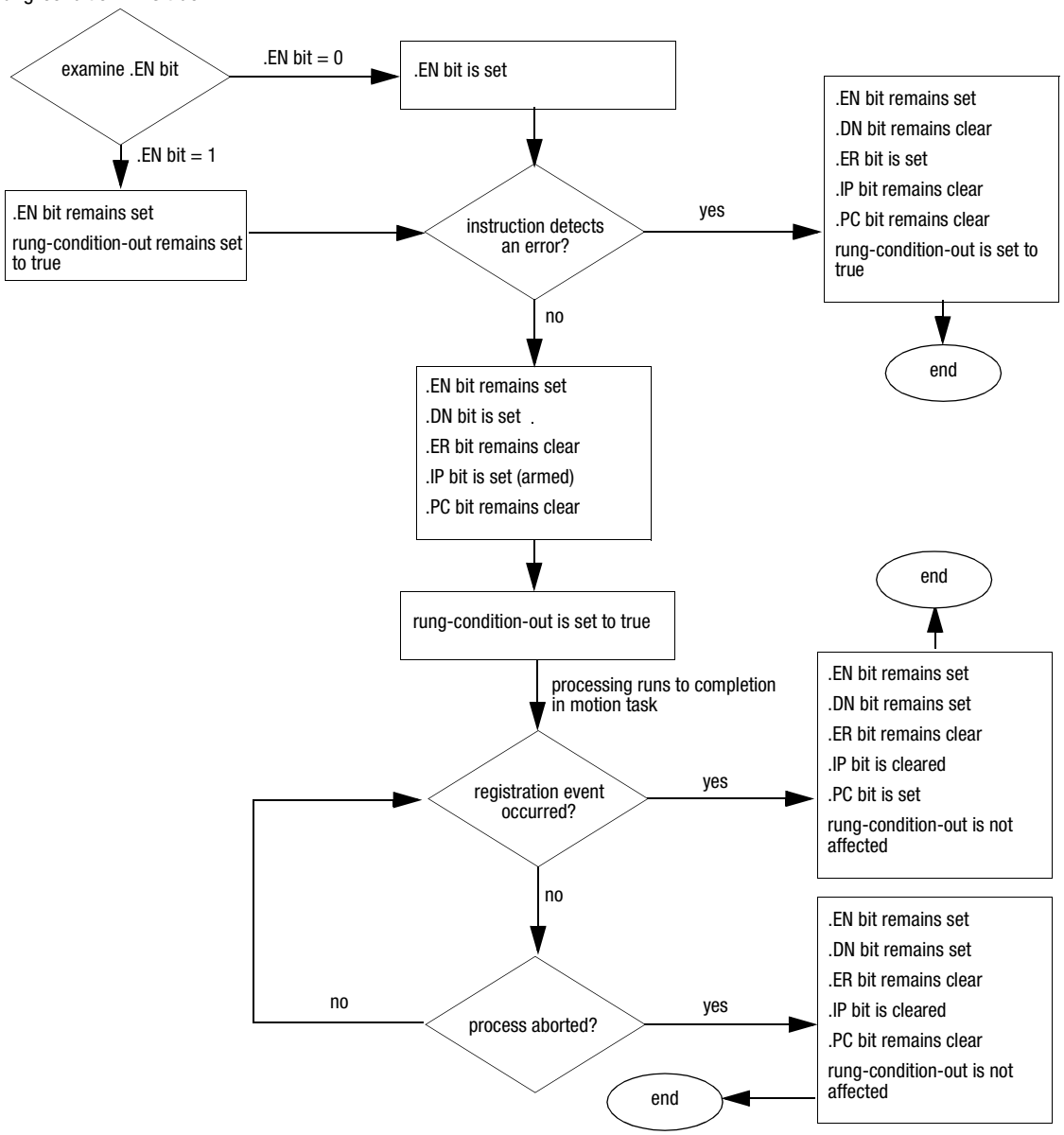
MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction arms registration event checking.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

.IP	BOOL	• The in process bit sets when arm registration is successfully initiated.
		• It resets when one of the following events occurs:
		  • A registration event occurs.
		  • Another MAR instruction supersedes the current instruction.
		  • A Motion Disarm Registration (MDR) instruction terminates the MAR instruction.

.PC	BOOL	The process complete bit sets when a registration event occurs.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The .IP bit is cleared. The .PC bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The .IP bit is not affected. The .PC bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

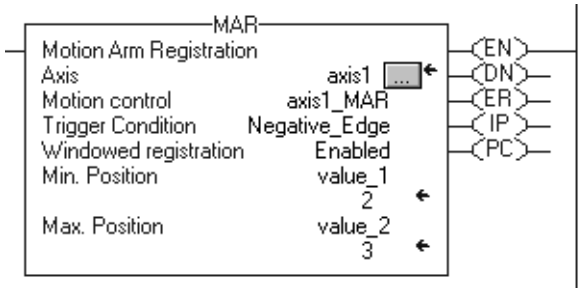
MAR Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
8	The axis is not configured as a servo or position only axis.
11	The axis is not configured.
12	Messaging to the servo module failed.
18	The axis type is configured as unused.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

MAR Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
RegEvArmStatus	True	The axis is looking for a registration event.
RegEvStatus	False	The previous registration event is cleared.

MAR Example:



When the input conditions are true, the controller arms servo-module registration-event checking for *axis1*.

Other Formats:

Format:	Syntax:
neutral text	`MAR(axis,motion_control,trigger,registration,minimum,maximum);`
ASCII text	`MAR axis motion_control trigger registration minimum maximum`

Motion Disarm Registration (MDR)

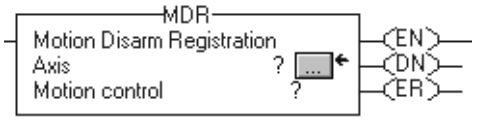
The MDR instruction is an output instruction.

Use the MDR instruction to disarm servo module registration event checking for an axis.

The MDR instruction uses message timing.

To use the MDR instruction, configure the axis as either a servo axis or a position only axis.

Operands:



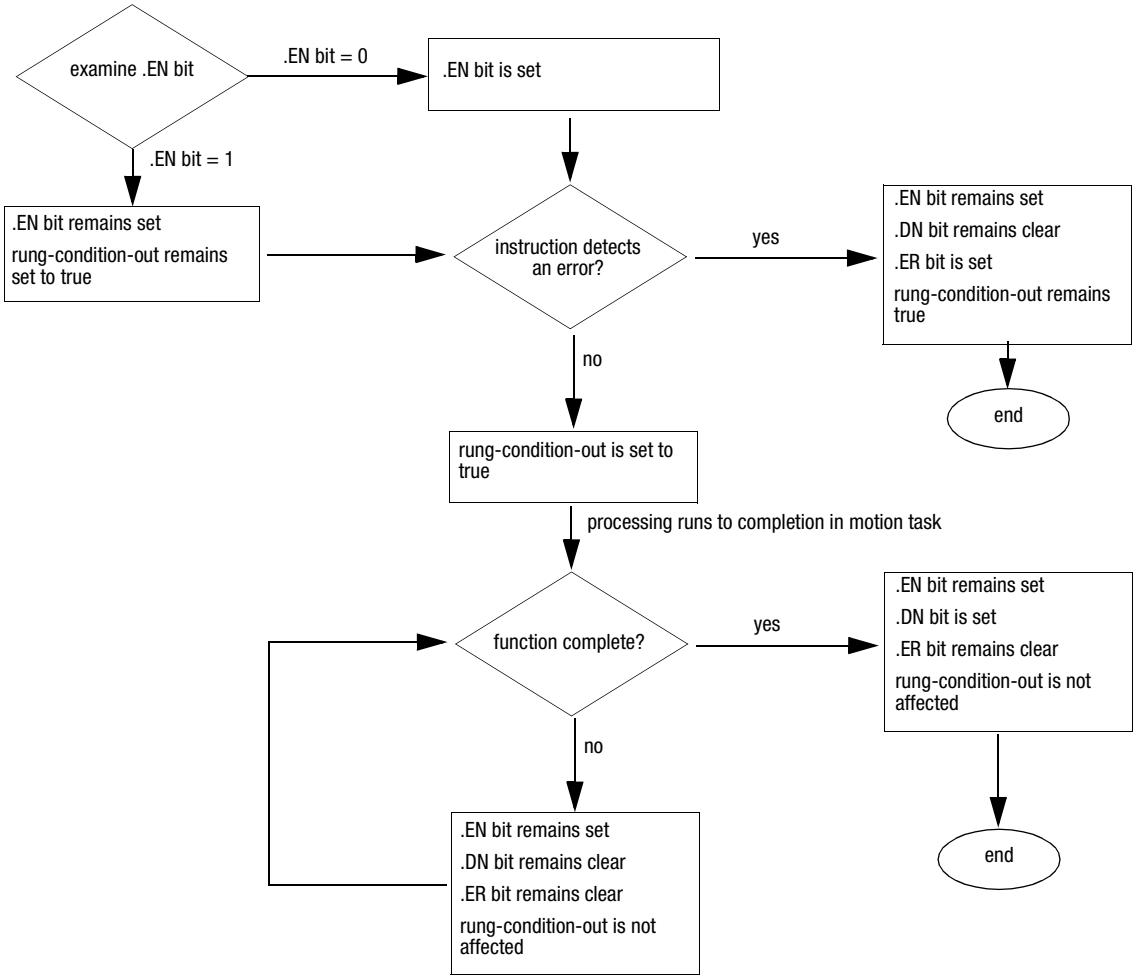
Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction disarms registration event checking.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

When you use this instruction, the controller resets the registration event status bit and the registration armed status bit.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

MDR Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
8	The axis is not configured as a servo or position only axis.
11	The axis is not configured.
12	Messaging to the servo module failed.
18	The axis type is configured as unused.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

MDR Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
RegEvArmStatus	False	The axis is not looking for a registration event.
RegEvStatus	False	The previous registration event is cleared.

MDR Example:

When the input conditions are true, the controller disarms registration-event checking for *axis1*.

Other Formats:

Format:	Syntax:
neutral text	`MDR(axis,motion_control);`
ASCII text	`MDR axis motion_control`

Motion Configuration Instructions

(MAAT, MRAT, MAHD, MRHD)

ATTENTION: 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 of the control variables.

Introduction

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

The motion configuration instructions are:

If you want to:	Use this instruction:	See page:
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	6-2
Command the servo module to run a tuning motion profile for an axis.	MRAT	6-5
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	6-8
Command the servo module to run one of three diagnostic tests on an axis.	MRHD	6-11

Motion Apply Axis Tuning (MAAT)

The MAAT instruction is an output instruction.

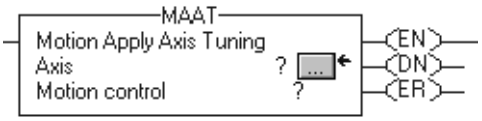
Use the MAAT instruction to compute a complete set of servo gains and dynamic limits based on a previously executed Motion Run Axis Tuning (MRAT) instruction. (The TuneStatus attribute should show the successful completion of an MRAT instruction.) This instruction also updates the servo module with the new gain parameters.

The MAAT instruction uses message type timing.

To use the MAAT instruction:

- Configure the axis as a servo axis.
- Ensure the axis operating state is axis ready.
- Ensure servo action is off.
- Use a Motion Run Axis Tuning (MRAT) instruction before the MAAT instruction.

Operands:

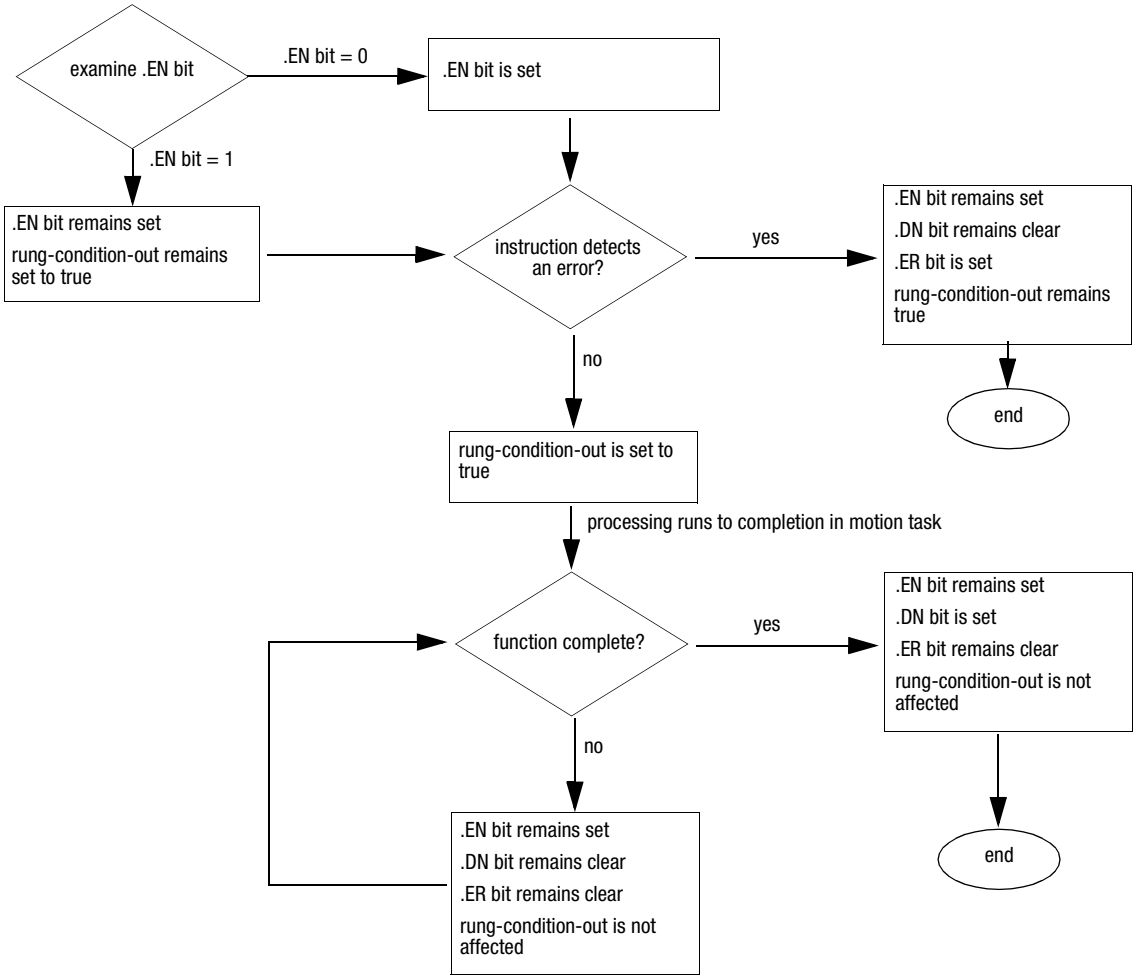


Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction completes an apply axis-tuning process.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

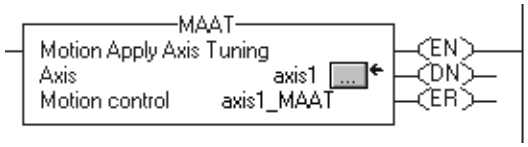
Arithmetic Status Flags: not affected

Fault Conditions: none

MAAT Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
4	The instruction tried to execute on an axis with a closed servo loop.
7	The axis is in the shutdown state.
8	The axis is not configured as a servo axis.
11	The axis is not configured.
12	Messaging to the servo module failed.
14	The instruction can not apply the tuning parameters because of an error in the run tuning instruction.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.
24	The controller attempted to execute an MDO, MSO, MAH, MAJ, MAM, MCD, MAPC, MATC, MAG, MRAT, or MRHD instruction when the controller was in the test mode.

MAAT Example:



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.

Other Formats:

Format:	Syntax:
neutral text	`MAAT(axis,motion_control);`
ASCII text	`MAAT axis motion_control`

Motion Run Axis Tuning (MRAT)

The MRAT instruction is an output instruction.

Use the MRAT to command the servo module to run a tuning motion profile for an axis. The input parameters for this function depend on the axis attributes listed on the Tune Servo tab of the Axis Property window.

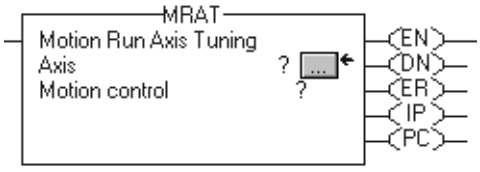
This instruction does not close the servo loop at any time.

The MRAT instruction uses message and process type timing.

To use the MRAT instruction:

- Configure the axis as a servo axis.
- Ensure the axis operating state is axis ready.

Operands:



Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure

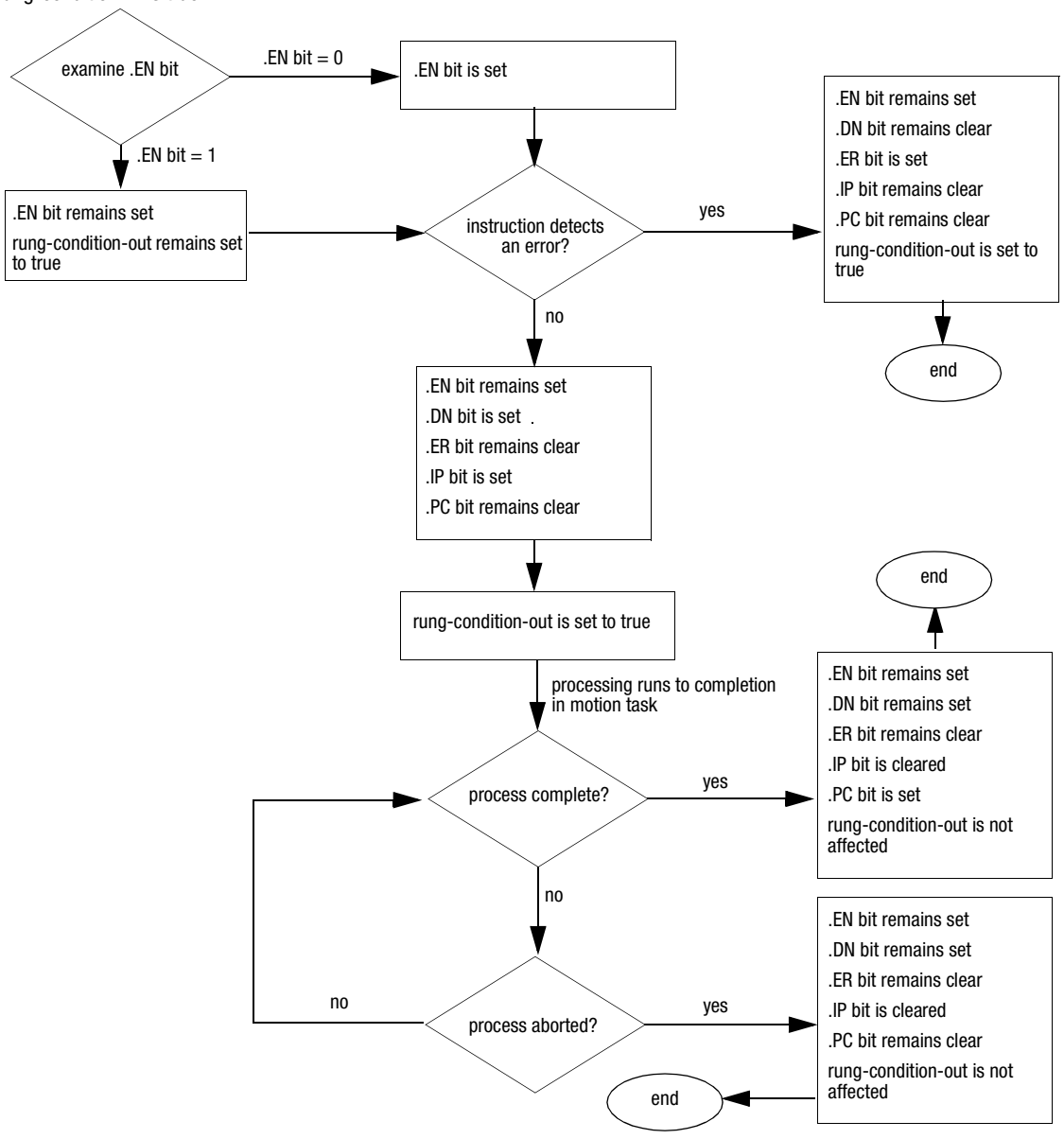
MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction initiates a tuning process.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

.IP	BOOL	• The in process bit sets when axis tuning is successfully initiated.
		• It resets when one of the following events occurs:
		  • The MRAT instruction completes.
		  • Tuning is aborted.

.PC	BOOL	The process complete bit sets when the instruction completes a tuning process.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The .IP bit is cleared. The .PC bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The .IP bit is not affected. The .PC bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

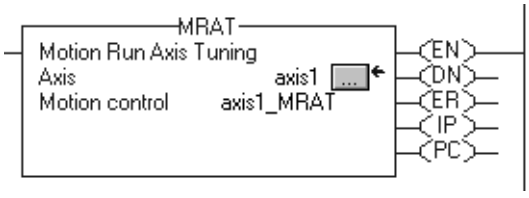
MRAT Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
4	The axis servo loop is closed.
6	The axis drive is enabled.
7	The axis is in the shutdown state.
8	The axis is not configured as a servo axis.
11	The axis is not configured.
12	Messaging to the servo module failed.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.
20	The axis is in the faulted state.
21	The group is in the faulted state.
24	The controller attempted to execute an MDO, MSO, MAH, MAJ, MAM, MCD, MAPC, MATC, MAG, MRAT, or MRHD instruction when the controller was in the test mode.

MRAT Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
DriveEnableStatus	True	• The axis is in the drive control state. • The drive enable output is active while the tuning profile is running.
TuneStatus	True	The axis is running a tuning process.

MRAT Example:



When the input conditions are true, the controller commands the servo module to run a tuning motion profile for *axis1*.

Other Formats:

Format:	Syntax:
neutral text	`MRAT(axis,motion_control);`
ASCII text	`MRAT axis motion_control`

Motion Apply Hookup Diagnostics (MAHD)

The MAHD instruction is an output instruction.

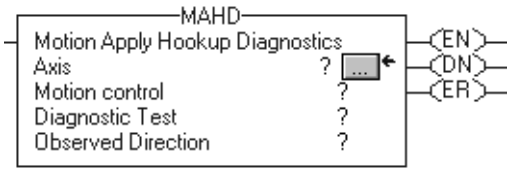
Use the MAHD instruction to apply the results of a previously executed Motion Run Hookup Diagnostics (MRHD) instruction. (The TestStatus attribute should show the successful completion of an MRHD instruction.) The MAHD instruction generates a new set of encoder and servo polarities based on the observed direction of motion during the MRHD instruction.

The MAHD instruction uses message type timing.

To use the MAHD instruction:

- Configure the axis as either a servo axis or a position-only axis.
- Make sure the axis operating state is axis ready.
- Make sure servo action is off.

Operands:

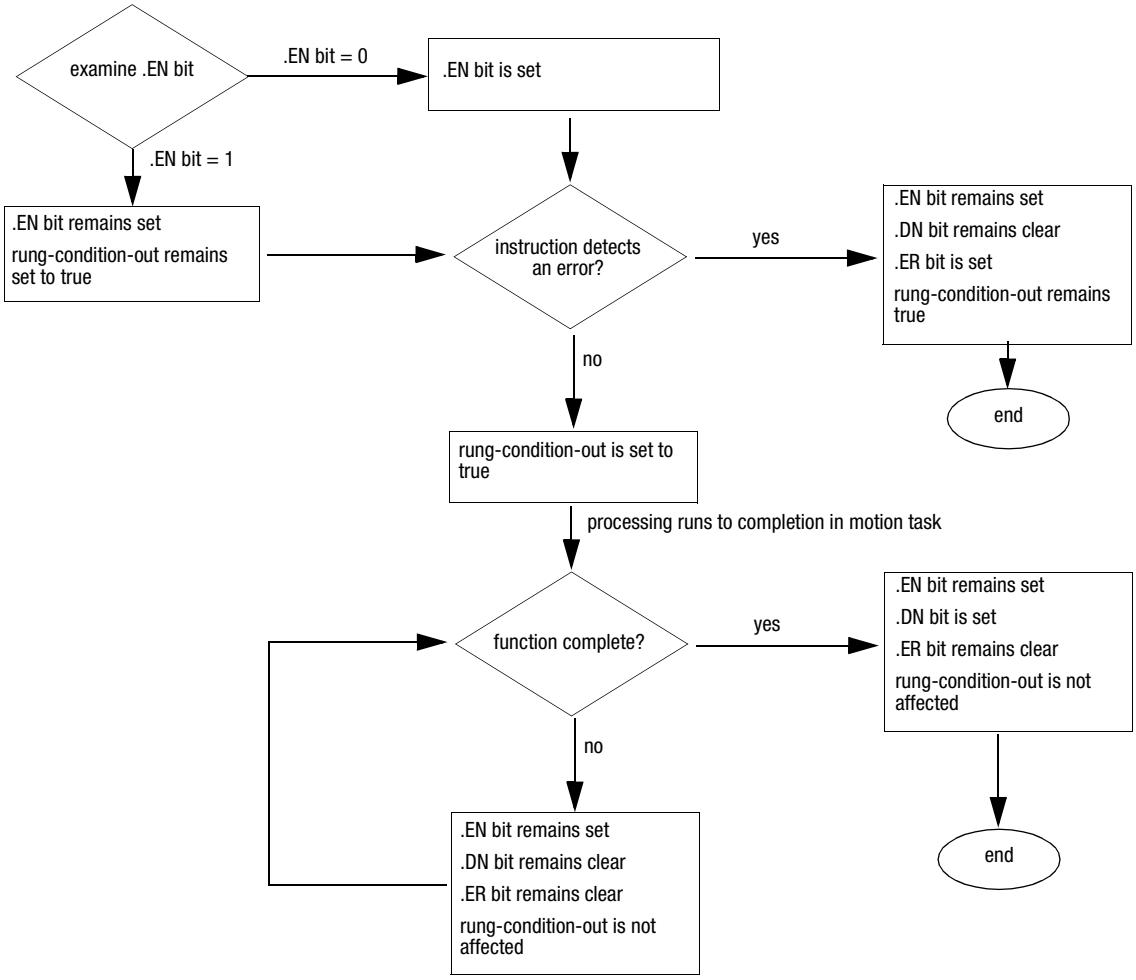


Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Diagnostic test	DINT	immediate	select the test to apply: • motor/encoder hookup test • encoder hookup test • encoder marker test
Observed direction	DINT	immediate	select the direction of motion during the diagnostic tests: • forward • reverse

MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction applies the results of a test process.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

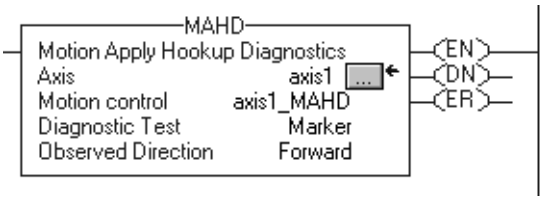
Arithmetic Status Flags: not affected

Fault Conditions: none

MAHD Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
4	The instruction tried to execute on an axis with a closed servo loop.
7	The axis is in the shutdown state.
8	The axis is not configured as a servo or position only axis.
11	The axis is not configured.
12	Messaging to the servo module failed.
15	The instruction can not apply the diagnostic parameters because of an error in the run diagnostic test instruction.
18	The axis type is configured as unused.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.

MAHD Example:



When the input conditions are true, the controller applies the results of a previously executed Motion Run Hookup Diagnostics (MRHD) instruction to *axis1*.

Other Formats:

Format:	Syntax:
neutral text	`MAHD(axis,motion_control,test,direction);`
ASCII text	`MAHD axis motion_control test direction`

Motion Run Hookup Diagnostics (MRHD)

The MRHD instruction is an output instruction.

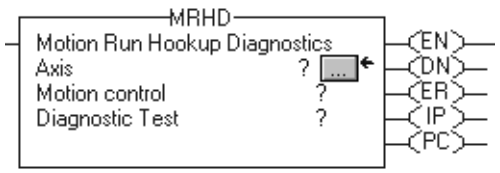
Use the MRHD instruction to command the servo module to run one of three diagnostic tests on an axis. The MRHD instruction also uses the TestDirection attribute as an input parameter.

The MRHD instruction uses message and process type timing.

To use the MRHD instruction, specify the diagnostic test to run:

If you want to run the:	Then:
Motor/encoder test	• Configure the axis as a servo axis. • Ensure that the axis is in the axis ready operating state.
Encoder hookup test	Configure the axis as either a servo axis or a position only axis.
Encoder marker test	Configure the axis as either a servo axis or a position only axis.

Operands:



Operand:	Type:	Format:	Description:
Axis	AXIS	tag	axis structure
Motion control	MOTION_ INSTRUCTION	tag	motion structure
Diagnostic test	DINT	immediate	select the diagnostic test to run: • motor/encoder hookup test • encoder hookup test • encoder marker hookup test

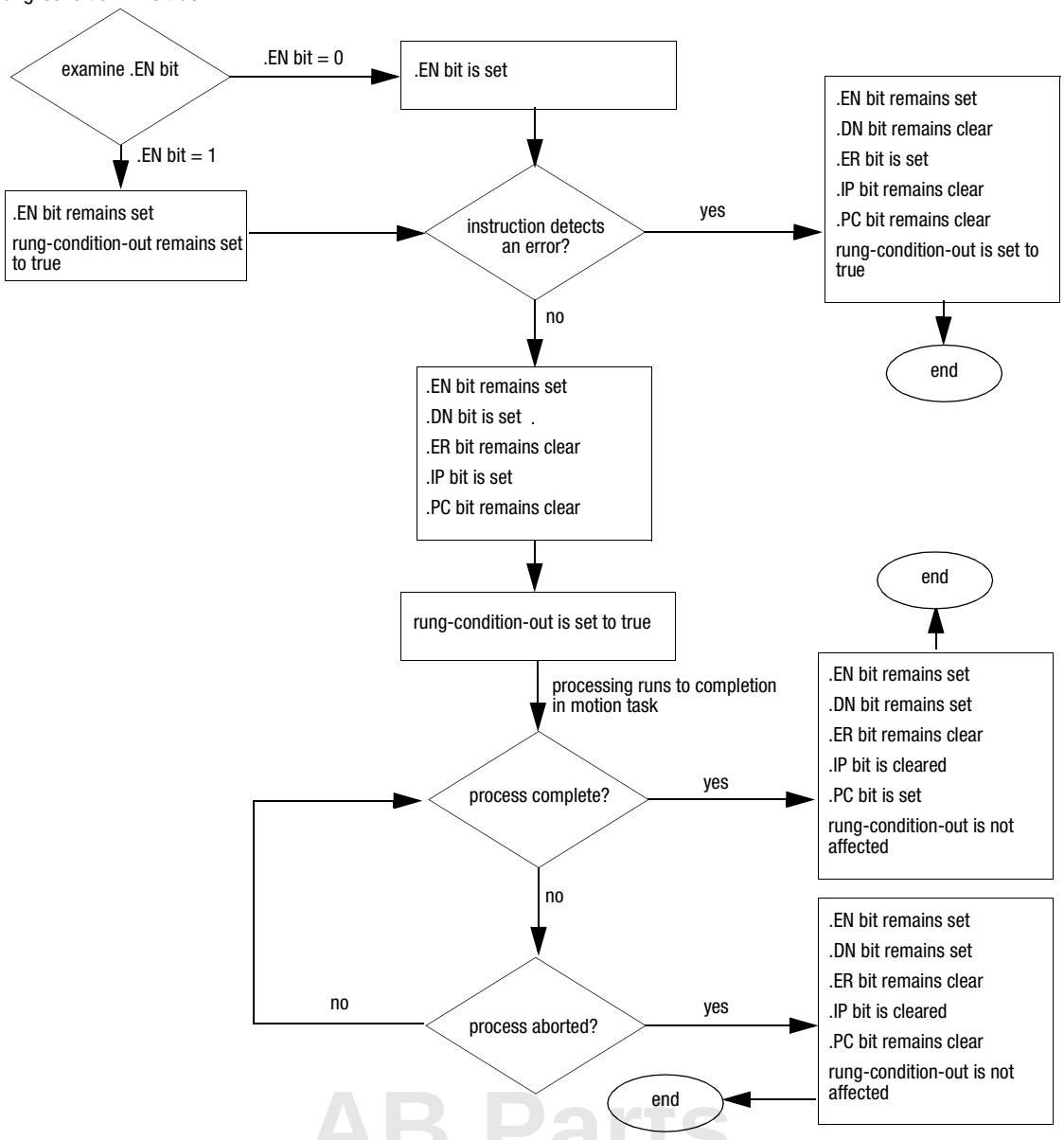
MOTION_INSTRUCTION Structure:

Mnemonic:	Data Type:	Description:
.EN	BOOL	The enable bit indicates when the instruction is enabled. It remains set until servo messaging completes and the rung-condition-in goes false.
.DN	BOOL	The done bit indicates when the instruction initiates a test process.
.ER	BOOL	The error bit indicates when the instruction detects an error, such as if the axis is not configured.

.IP	BOOL	• The in process bit sets when the hookup test is successfully initiated.
		• It resets when one of the following events occurs:
		  • The MRHD instruction completes.
		  • An instruction or a servo fault terminates the MRHD instruction.

.PC	BOOL	The process complete bit sets when the instruction completes the diagnostic test process.

Execution:

Condition:	Action:
prescan	The .EN bit is cleared. The .DN bit is cleared. The .ER bit is cleared. The .IP bit is cleared. The .PC bit is cleared. The rung-condition-out is set to false.
rung-condition-in is false	The .EN bit is cleared if either the .DN or .ER bit is set. Otherwise, the .EN bit is not affected. The .DN bit is not affected. The .ER bit is not affected. The .IP bit is not affected. The .PC bit is not affected. The rung-condition-out is set to false.
rung-condition-in is true	

Arithmetic Status Flags: not affected

Fault Conditions: none

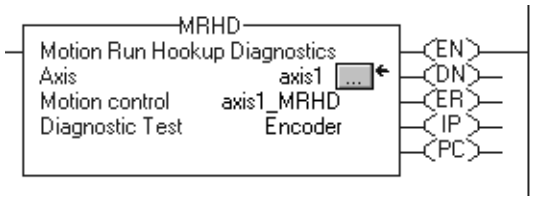
MRHD Error Codes (.ERR):

Error Code:	Description:
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the servo module executes a messaging instruction without checking the .DN bit of the preceding instruction.
4	The axis servo loop is closed.
6	The axis drive is enabled.
7	The axis is in the shutdown state.
8	The axis is not configured as a servo or position only axis.
11	The axis is not configured.
12	Messaging to the servo module failed.
18	The axis type is configured as unused.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.
20	The axis is in the faulted state.
21	The group is in the faulted state.
24	The controller attempted to execute an MDO, MSO, MAH, MAJ, MAM, MCD, MAPC, MATC, MAG, MRAT, or MRHD instruction when the controller was in the test mode.

MRHD Changes to Axis Status Bits:

Bit Name:	State:	Meaning:
DriveEnableStatus	True	• The axis is in the drive control state. • The drive enable output is active while the tuning profile is running.
TestStatus	True	The axis is running a testing process.

MRHD Example:



When the input conditions are true, the controller runs the *encoder* diagnostic test on *axis1*.

Other Formats:

Format:	Syntax:
neutral text	`MRHD(axis,motion_control,test);`
ASCII text	`MRHD axis motion_control test`

Notes:

Structures

Introduction

This appendix lists the predefined motion structures and the mnemonics for the members you can address within instructions.

Data type:	See page:
AXIS[1]	A-1
MOTION_GROUP[1]	A-4
MOTION_INSTRUCTION	A-5
CAM	A-7
CAM_PROFILE	A-7

1. These tags are controller tags only. These tags do not support arrays, cannot be nested in a user-defined structure, and cannot be passed to another routine via a JSR instruction.

Some of the more complex structures have attributes in addition to those described in this appendix. The additional attributes are accessible only from the configuration tabs for that structure.

AXIS Structure

Each AXIS structure contains status and configuration information for an axis.

Mnemonic:	Data Type:	Description:			
.MotionFault	DINT	The motion fault bits for your axis.			
		Bit:	**Number:**	**Data Type:**	**Description:**
		.ACAsyncConnFault	00	BOOL	asynchronous connection fault
		.ACSyncConnFault	01	BOOL	synchronous connection fault (controller declared)
.MotionStatus	DINT	The motion status bits for your axis.			
		Bit:	**Number:**	**Data Type:**	**Description:**
		.AccelStatus	00	BOOL	velocity increase relative to command position
		.DecelStatus	01	BOOL	velocity decrease relative to command position
		.MoveStatus	02	BOOL	set when move motion profile is in progress
		.JogStatus	03	BOOL	set when jog motion profile is in progress
		.GearingStatus	04	BOOL	set when axis is gearing to another axis
		.HomingStatus	05	BOOL	set when a home motion profile is in progress
		.StoppingStatus	06	BOOL	set when a stopping process is in progress
		.AxisHomedStatus	07	BOOL	set when absolute position reference is established
		.PositionCamStatus	08	BOOL	set when a Pcam motion profile is in progress
		.TimeCamStatus	09	BOOL	set when a Tcam motion profile is in progress
		.PositionCamPendingStatus	10	BOOL	set when Pcam profile is waiting for another to end
		.TimeCamPendingStatus	11	BOOL	set when Tcam profile is waiting for another to end
		.GearingLockedStatus	12	BOOL	clear when axis is clutching to a new gear speed
		.PositionCamLockStatus	13	BOOL	set when master axis meets starting Pcam condition

Mnemonic:	Data Type:	Description:			
.ServoFault	DINT	The servo fault bits for your servo loop.			
		Bit:	**Number:**	**Data Type:**	**Description:**
		.POtrvlFault	00	BOOL	positive overtravel fault
		.NOtrvlFault	01	BOOL	negative overtravel fault
		.PosErrorFault	02	BOOL	position error fault
		.EncCHALossFault	03	BOOL	encoder channel A loss fault
		.EncCHBLossFault	04	BOOL	encoder channel B loss fault
		.EncCHZLossFault	05	BOOL	encoder channel Z loss fault
		.EncNsFault	06	BOOL	encoder noise fault
		.Drivefault	07	BOOL	drive fault
		Bit:	**Number:**	**Data Type:**	**Description:**
		.SyncConnFault	00	BOOL	synchronous connection fault (servo declared)
		.HardFault	01	BOOL	servo hardware fault
.ServoStatus	DINT	The status bits for your servo loop.			
		Bit:	**Number:**	**Data Type:**	**Description:**
		.ServoActStatus	00	BOOL	servo action
		.DriveEnableStatus	01	BOOL	drive enable
		.OutLmtStatus	02	BOOL	output limit
		.PosLockStatus	03	BOOL	position lock
		.HomeSwitchStatus	05	BOOL	shows the current state of the home input switch
		Bit:	**Number:**	**Data Type:**	**Description:**
		.TuneStatus	13	BOOL	tuning process
		.TestStatus	14	BOOL	test diagnostic
		.ShutdownStatus	15	BOOL	axis shutdown
.EventStatus	DINT	The servo event bits for your servo loop.			
		Bit:	**Number:**	**Data Type:**	**Description:**
		.WatchEvArmStatus	00	BOOL	watch event armed
		.WatchEvStatus	01	BOOL	watch event
		.RegEvArmStatus	02	BOOL	registration event armed
		.RegEvStatus	03	BOOL	registration even
		.HomeEvArmStatus	04	BOOL	home event armed
		.HomeEvStatus	05	BOOL	home event

Mnemonic:	Data Type:	Description:			
.UpdateStatus	DINT	The servo status update bits for your axis.			
		Bit:	**Number:**	**Data Type:**	**Description:**
		.AxisTypeStatus	00	BOOL	axis type
		.PosUnwindStatus	01	BOOL	position unwind
		.MaxPTrvlStatus	02	BOOL	maximum positive travel
		.MaxNTrvlStatus	03	BOOL	maximum negative travel
		.PosErrorTolStatus	04	BOOL	position error tolerance
		.PosLockTolStatus	05	BOOL	position lock tolerance
		.PosPGainStatus	06	BOOL	position proportional gain
		.PosIGainStatus	07	BOOL	position integral gain
		.VelFfGainStatus	08	BOOL	velocity feedforward gain
		.AccFfGainStatus	09	BOOL	acceleration feedforward gain
		.VelPGainStatus	10	BOOL	velocity proportional gain
		.VelIGainStatus	11	BOOL	velocity integral gain
		.OutFiltBwStatus	12	BOOL	output filter bandwidth
		.OutScaleStatus	13	BOOL	output of the servo loop into the equivalent voltage to the drive
		.OutLimitStatus	14	BOOL	maximum servo output voltage
		.OutOffsetStatus	15	BOOL	effects of the cumulative offsets of the servo module DAC output and the servo drive input
		.FricCompStatus	16	BOOL	friction compensation
		.POtrvlFaultActStatus	17	BOOL	positive overtravel fault
		.PosErrorFaultActStatus	18	BOOL	position fault
		.EncLossFaultActStatus	19	BOOL	encoder loss fault
		.EncNsFaultActStatus	20	BOOL	encoder noise fault
		.DriveFaultActStatus	21	BOOL	drive fault
		.ServoConfigBitsStatus	22	BOOL	configuration bits

MOTION_GROUP Structure

There is one MOTION_GROUP structure per controller. This structure contains status and configuration information about the motion group.

Mnemonic:	Data Type:	Description:			
.GroupStatus	DINT	The status bits for the group.			
		Bit:	**Number:**	**Data Type:**	**Description:**
		.InhibStatus	00	BOOL	inhibit status
		.GroupSynced	01	BOOL	synchronization status
.MotionFault	DINT	The motion fault bits for the group.			
		Bit:	**Number:**	**Data Type:**	**Description:**
		.ACAsyncConnFault	00	BOOL	asynchronous connection fault
		.ACSyncConnFault	01	BOOL	synchronous connection fault (controller . declared)
.ServoFault	DINT	The servo-module fault bits for the group.			
		Bit:	**Number:**	**Data Type:**	**Description:**
		.POtrvlFault	00	BOOL	positive overtravel fault
		.NOtrvlFault	01	BOOL	negative overtravel fault
		.PosErrorFault	02	BOOL	position error fault
		.EncCHALossFault	03	BOOL	encoder channel A loss fault
		.EncCHBLossFault	04	BOOL	encoder channel B loss fault
		.EncCHZLossFault	05	BOOL	encoder channel Z loss fault
		.EncNsFault	06	BOOL	encoder noise fault
		.DriveFault	07	BOOL	drive fault
		Bit:	**Number:**	**Data Type:**	**Description:**
		.SyncConnFault	00	BOOL	synchronous connection fault (servo declared)
		.HardFault	01	BOOL	servo hardware fault
.GroupFault	DINT	The fault bits for the group.			
		Bit:	**Number:**	**Data Type:**	**Description:**
		.GroupOverlapFault	00	BOOL	group overlap fault

MOTION_INSTRUCTION Structure

Each motion instruction has a MOTION_INSTRUCTION structure that contains status information about the instruction.

Mnemonic:	Data Type:	Description:			
FLAGS	BOOL	The instruction status bits are:			
		Bit:	**Number:**	**Data Type:**	**Description:**
		.ACCEL	00	BOOL	The .ACCEL bit indicates that the velocity has increased for the individual instruction that it is tied to i.e jog, move, gearing.
		.DECEL	01	BOOL	The .DECEL bit indicates that the velocity has decreased for the individual instruction that it is tied to i.e jog, move, gearing.
		.PC	26	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	27	BOOL	The in process bit indicates that a process is being executed.
		.ER	28	BOOL	The error bit indicates when the operation generates an overflow.
		.DN	29	BOOL	The done bit indicates that the operation is complete.
		.EN	31	BOOL	The enable bit indicates that the instruction is enabled.

Mnemonic:	Data Type:	Description:
.ERR	INT	The error value contains the error code associated with a motion instruction.

Value:	Description
3	The instruction tried to execute while another instance of this instruction was executing. This can occur when the controller executes a messaging instruction without checking the .DN bit of the preceding instruction.
4	The instruction tried to execute on an axis whose servo loop is closed.
5	The instruction tried to execute on an axis whose servo loop is not closed.
6	The axis drive is enabled.
7	The axis is in the shutdown state.
8	The axis is not configured as a servo, position only, or virtual axis type.
9	The instruction tried to execute in a direction that aggravates the current overtravel condition.
10	The master axis reference is the same as the slave axis reference.
11	The axis is not configured.
12	Messaging to the servo module failed.
13	The instruction tried to use a parameter that is outside the range limit.
14	The instruction can not apply the tuning parameters because of an error in the run tuning instruction.
15	The instruction can not apply the diagnostic parameters because of an error in the run diagnostic test instruction.
16	The instruction tried to execute with homing in process.
17	The instruction tried to execute a rotary move on an axis that is not configured for rotary operation.
18	The axis type is configured as unused.
19	The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration.
20	The axis is in the faulted state.
21	The group is in the faulted state.
22	An MSO (Motion Servo On) or MAH (Motion Axis Home) instruction was attempted while the axis was in motion.
23	An instruction attempted an illegal change of dynamics.
24	The controller tried to execute an MDO, MSO, MAH, MAJ, MAM, MCD, MAPC, MATC, MAG, MRAT, or MRHD instruction when the controller was in the test mode.
25	The instruction you tried to execute is not a legal instruction.
26	The cam array is of an illegal length.
27	The cam profile array is of an illegal length.
28	You have an illegal segment type in the cam element.
29	You have an illegal order of cam elements.
30	You tried to execute while a cam profile is being calculated.
31	The cam profile array you tried to execute is in use.
32	The cam profile array you tried to execute was not calculated.

.STATE	SINT	The execution state is always set to 0 when the controller sets the .EN bit for a motion instruction. Other execution states depend on the motion instruction.

Mnemonic:	Data Type:	Description:
.STATUS	SINT	The message status value indicates the status condition of any message associated with the motion function. **Value:** **Description** 0000 The message was successful. 0001 The module is processing another message. 0002 The module is waiting for a response to a previous message. 0003 The response to a message failed. 0004 The module is not ready for messaging.
.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.

CAM Structure

The Cam data type consists of slave and master point pairs as well as an interpolation type. Since there is no association with a specific axis position or time, the point values are unit-less. The interpolation type can be specified for each segment as either linear or cubic. The format of the cam array element is shown in the following table.

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. **Value:** **Description** 0 linear. 1 cubic.

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 to the user is Status which is defined in the following table.

Mnemonic:	Data Type:	Description:
Status	DINT	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 an uncalculated element in a Cam Profile, the instruction produces an error. **Value:** **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 and MATC instructions.

ControlLogix and Logix5550, Allen-Bradley, and RSLogix are trademarks of Rockwell Automation.

Reach us now at www.rockwellautomation.com

Wherever you need us, Rockwell Automation brings together leading brands in industrial automation including Allen-Bradley controls, Reliance Electric power transmission products, Dodge mechanical power transmission components, and Rockwell Software. Rockwell Automation's unique, flexible approach to helping customers achieve a competitive advantage is supported by thousands of authorized partners, distributors and system integrators around the world.

Americas Headquarters, 1201 South Second Street, Milwaukee, WI 53204, USA, Tel: (1) 414 382-2000, Fax: (1) 414 382-4444
European Headquarters SA/NV, avenue Herrmann Debroux, 46, 1160 Brussels, Belgium, Tel: (32) 2 663 06 00, Fax: (32) 2 663 06 40
Asia Pacific Headquarters, 27/F Citicorp Centre, 18 Whitfield Road, Causeway Bay, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Rockwell Automation

Publication 1756-6.4.3 October 1999

PN 957208-33