Rockwell Automation Application Content

Machine Builder Libraries

Reference Manual

Recover To Path

raM_Opr_SyncPthPhyAx_CD **v2.x**

December, 2022

Rockwell Automation

Important User Information

Solid-state equipment has operational characteristics differing from those of electromechanical equipment. Safety Guidelines for the Application, Installation and Maintenance of Solid State Controls (publication SGI-1.1 available from your local Rockwell Automation sales office or online at http://literature.rockwellautomation.com) describes some important differences between solid-state equipment and hard-wired electromechanical devices. Because of this difference, and because of the wide variety of uses for solid-state equipment, all persons responsible for applying this equipment must satisfy themselves that each intended application of this equipment is acceptable.

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

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

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

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

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

WARNING 	Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.
IMPORTANT	Identifies information that is critical for successful application and understanding of the product.
ATTENTION 	Identifies information about practices or circumstances or death, property damage, or economic loss. Attentions avoid a hazard, and recognize the consequence.
SHOCK HAZARD 	Labels may be on or inside the equipment, that dangerous voltage may be present.
BURN HAZARD 	Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.

Table of Contents

1 Overview

raM_Opr_SyncPthPhyAx_CD:

- Instruction is used to recover physical axis position based on path axis position or path axis position based on physical axis position.

Use when:

- Using a Device Handler for Axis Management
- Need to recover physical axis position after physical axis lost relation to the path axis (axis fault, e-stop, door opening, etc.)
- Need to recover path axis position to physical axis. When physical axis has absolute encoder and path axis needs to recover after power recycle.

Do NOT use when:

- Not using a Device Handler

1.1 Prerequisites

- Device Handler for Axis Management
- Studio 5000 - Logix Designer
  - v30.0 →
- Studio 5000 - Application Code Manager
  - V4.0 →

1.2 Functional Description

The 'Synch Path Physical Axis' object provides the functionality of recovering physical axis position to path position when physical axis loses its relation to the path axis (i.e. during servo axis fault, e-stop, door opening, etc.). The object can also recover path axis to physical axis position. The recovery relation can be configured in the Cfg_SyncType parameter.

The physical axis should be energized prior to executing this object. If allowed by machine conditions, object can execute an axis home prior to recover to path operation (automatic home configurable option) or it can generate fault when a recovery attempt has been made and axis is not homed.

Note: After controller power recycle or Program to Run mode, the object should be executed once as path to physical synch (Cfg_SyncType =0) before attempting physical to path synch (Cfg_Synch=1).

General Status Bit Behavior:

Note: Status bit not shown on the output side of the instruction are not used and will not exist in the instruction backing tag.

Status Bit	Description / Behavior
*.Sts_EO	• Enable Out indicated the status of the output line of the instruction. • If false (logically LO) any instruction on the ladder rung between the instruction and the neutral rail will not be energized. • If the instruction is removed from ladder scan either in a conditional subroutine, MCR zone, JMP/LBL etc., the bit will remain in its last evaluated state.
*.Sts_EN	• The rung-in condition of the ladder rung is true and the instruction is being evaluated. • If the instruction is removed from ladder scan either in a conditional subroutine, MCR zone, JMP/LBL etc., the bit will remain in its last evaluated state.
*.Sts_ER	• If the instruction experiences an internal error, the *. Sts_ER bit will be set. Error codes / Extended codes can be found by monitoring the backing tag *.Sts_ERR / *.Sts_EXERR members respectively. • If the instruction is removed from ladder scan either in a conditional subroutine, MCR zone, JMP/LBL etc., the bit will remain in its last evaluated state.
*.Sts_IP	• Used to identify the instruction is in the process • If the instruction is removed from ladder scan either in a conditional subroutine, MCR zone, JMP/LBL etc., the bit will remain in its last evaluated state.
*.Sts_PC	• Used when the execution of the instruction requires more than a single scan to complete, and indicates the 'process' carried out by the instruction has successfully completed. • If the instruction is removed from ladder scan either in a conditional subroutine, MCR zone, JMP/LBL etc., the bit will remain in its last evaluated state.

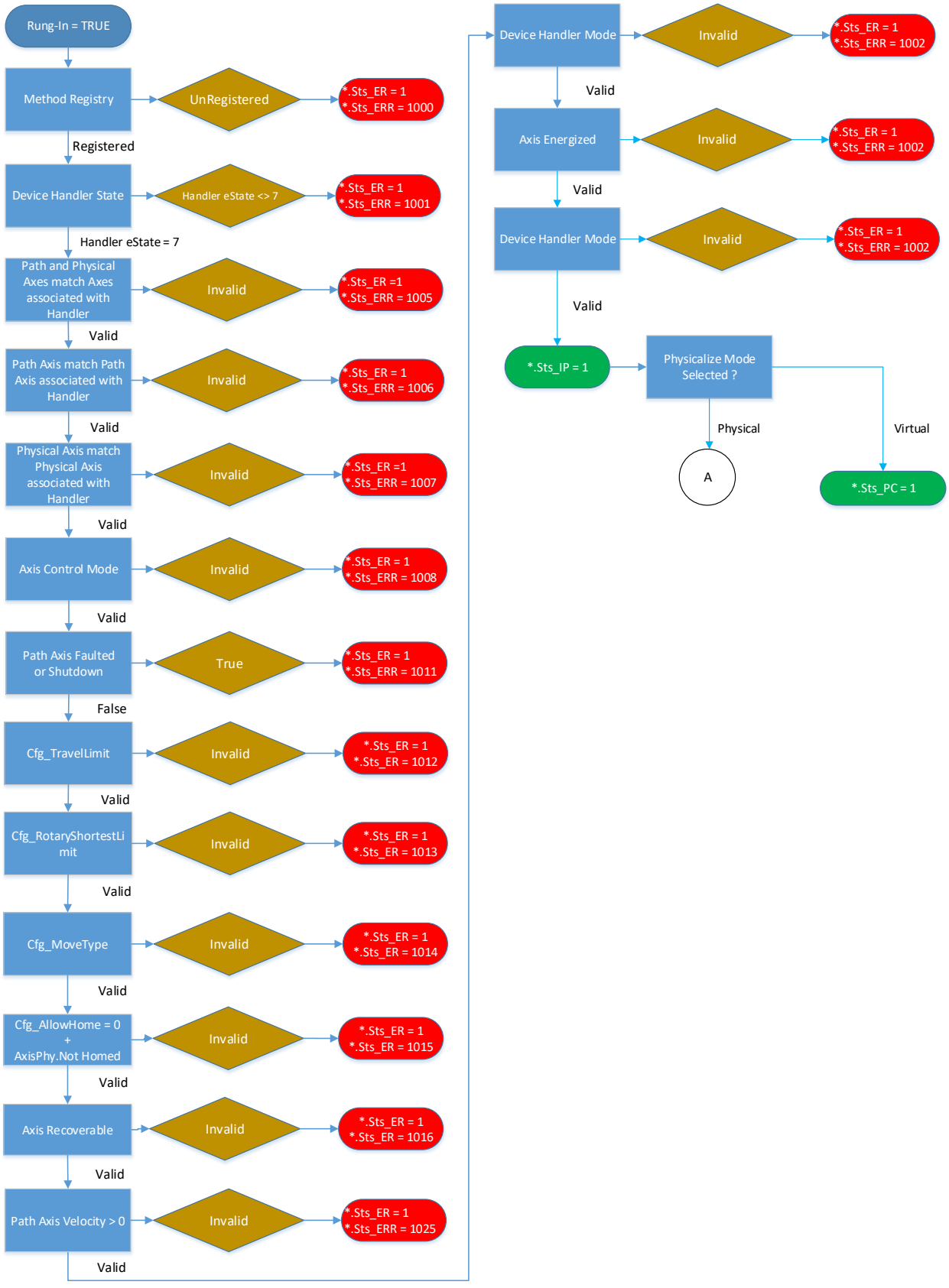
1.3 Execution

- Level

1.3.1 Affected Device Handler Status

Status	Value
*.Sts.OnPath	1

1.3.2 Execution Table



2 Instruction

2.1 Footprint

Characteristic	Description	Value	Unit
Definition	Estimated memory required to store the object definition, including all dependents with exception of components already installed by Device Handler.	32	kB
Instance	Estimated memory required per object instantiated. This includes the object instance and all datatypes required to verify the project. In the case of user configurable arrays, an application relevant array length will be used for estimation.	0.5	kB
Execution L8x	Estimated execution time / scan footprint evaluated in 1756-L8x PAC	70	us

2.2 Input Data

Input	Function / Description	Data Type
Ref_AxisPhy	Physical Axis	AXIS_CIP_DRIVE
Ref_AxisPth	Path Axis (axis which keeps the relation to the master and to which PHY axis is geared)	AXIS_VIRTUAL
Ref_Handle	Axis Handle Data	raM_UDT_Dvc_xADH_DataHndl
Cfg_SyncType	0 = Path to Physical 1 = Physical to Path (position recovery)	DINT
Cfg_AllowHome	Enable bit to allow axis homing of physical axis position during the recovery. Applicable when Cfg_SyncType = 1.	BOOL
Cfg_TravelLIM	Maximum difference between Path and Physical axis when recovery is still performed otherwise error is set. Enter 0 to disable this check. Applicable when Cfg_SyncType = 1.	REAL
Cfg_RotaryShortestTOL	Axis Positions differences below the limit are always handled as rotary shortest movement type to avoid moving by whole axis unwind just because of a very small position discrepancy. Applicable when Cfg_SyncType = 1.	REAL
Cfg_MoveType	Move type for recovery (i.e. rotary shortest, absolute, forward only etc.) 0 = Absolute 2 = Rotary Shortest 3 = Rotary Positive 4 = Rotary Negative Applicable when Cfg_SyncType = 1.	DINT
Set_Speed	Define speed of recovery movement, acceleration and deceleration values are calculated. Applicable when Cfg_SyncType = 1.	REAL

2.3 Output Data

Output	Function / Description	DataType
raM_Opr_SyncPthPhyAx_CD	Instruction Identification Bit	BOOL
Sts_EO	Instruction has enabled the rung output. Provides a visible indicator of the EnableOut system parameter for use during ladder instantiation	BOOL
Sts_EN	Instruction is Being Scanned – Rung-In Condition = TRUE	BOOL
Sts_ER	Instruction is in Error - See Sts_ERR / Sts_EXERR for Additional Error Information	BOOL
Sts_ERR	Instruction Error Code - See Instruction Help for Code Definition	DINT
Sts_EXERR	Instruction Extended Error Code - See Instruction Help for Code Definition	DINT

Output	Function / Description	DataType
Sts_IP	Recovery is 'In Process'	BOOL
Sts_PC	Recovery is 'Process Complete' – Command Position has reached target position.	BOOL
Sts_MtdID	Method ID Number	DINT

2.4 Error Codes

Sts_ERR	Description
0	No errors present
1000	Method failed to register. Method will not execute until registered. Method Registry Array must be larger.
1001	Device Handler is not in a running state. Commands to the device cannot be processed.
1002	Device Handler is not in a supported mode. Neither in Physical nor Virtual.
1005	Path and Physical Axes referenced by instruction do not match Axes associated with the Handler.
1006	Path Axis referenced by instruction does not match Path Axis associated with the Handler.
1007	Physical Axis referenced by instruction does not match Physical Axis associated with the Handler.
1008	Invalid Axis Control Mode. Axis must be configured for Velocity or Position.
1010	Physical axis faulted or shutdown at invocation. Applicable when AxisHandler mode is Physical.
1011	Path axis faulted or shutdown at invocation.
1012	Cfg_TravelLIM Actual Recovery distance is bigger than the configured limit or smaller then 0. Applicable when Cfg_SyncType = 1.
1013	Cfg_RotaryShortestTOL - Rotary shortest distance is less than zero. Applicable when Cfg_SyncType = 1.
1014	Cfg_MoveType - Movement type is different then allowed movement types (0,2,3,4). Applicable when Cfg_SyncType = 1.
1015	Cfg_AllowHome - Axis is not homed and automatic homing during recovery is disabled. Applicable when Cfg_SyncType = 1.
1016	Path position integrity not verified – Recover to path terminated. Execute the object once with Cfg_SycnType = 0.
1017	Set_Speed is set less than 0
1118	Axis not energized (energize method execution is required prior sync)
1020	MAH Instruction Error. See *.Sts_EXERR for Motion Instruction error code. Applicable when Cfg_SyncType = 1.
1022	MAM Instruction Error. See *.Sts_EXERR for Motion Instruction error code. Applicable when Cfg_SyncType = 1.
1023	MAG Instruction Error. See *.Sts_EXERR for Motion Instruction error code.
1025	Path axis moving at invocation
1027	MRP Instruction Error. See *.Sts_EXERR for Motion Instruction error code. Applicable when Cfg_SyncType = 0.

Sts_EXERR	Description
< Number >	If a native instruction error occurs internally, the value of the instruction *.ERR DINT will be placed in Sts_EXERR.

3 Application Code Manager

3.1 Definition Object: raM_Opr_SyncPthPhyAx_CD

This object contains the AOI definition and used as linked library to implement object. This gives flexibility to choose to instantiate only definition and create custom implement code. User may also create their own implement library and link with this definition library object.

3.2 Implement Object: raM_LD_SyncPthPhyAx_CD

Implement Language: Ladder Diagram

Parameter Name	Default Value	Instance Name	Definition	Description
ObjectName	raM_LD_SyncPthPhyAx_CD			Object Name
RoutineName	{ObjectName}	{RoutineName}	Routine	Name of the routine where the object will be placed
TagName	_{ObjectName}		Local Tag	Instruction backing tag
StartBitTagName	Cmd_Start{ObjectName}		Local Tag	Bit used to trigger object

Linked Library

Link Name	Catalog Number	Revision	Solution	Category
raM_LD_AxisHandler_CD	raM_LD_AxisHandler_CD	>=2.0	(RA-LIB) Machine	DvcHdlr – CIP Motion
raM_Opr_SyncPthPhyAx_CD	raM_Opr_SyncPthPhyAx_CD	>=2.0	(RA-LIB) Machine	General Motion

Interface

Interface Name	Linked Library	Revision
MethodInterface	raM_LD_AxisHandler_CD	1.0

MethodInterface Members

Member Name	Description
AHPrgName	Program name where Axis Handler resides
AxisName_PHY	Name of Physical Axis attached to Handler
AxisName_PTH	Name of Path Axis attached to Handler
AHTagName	Handle Tag name of Axis Handler

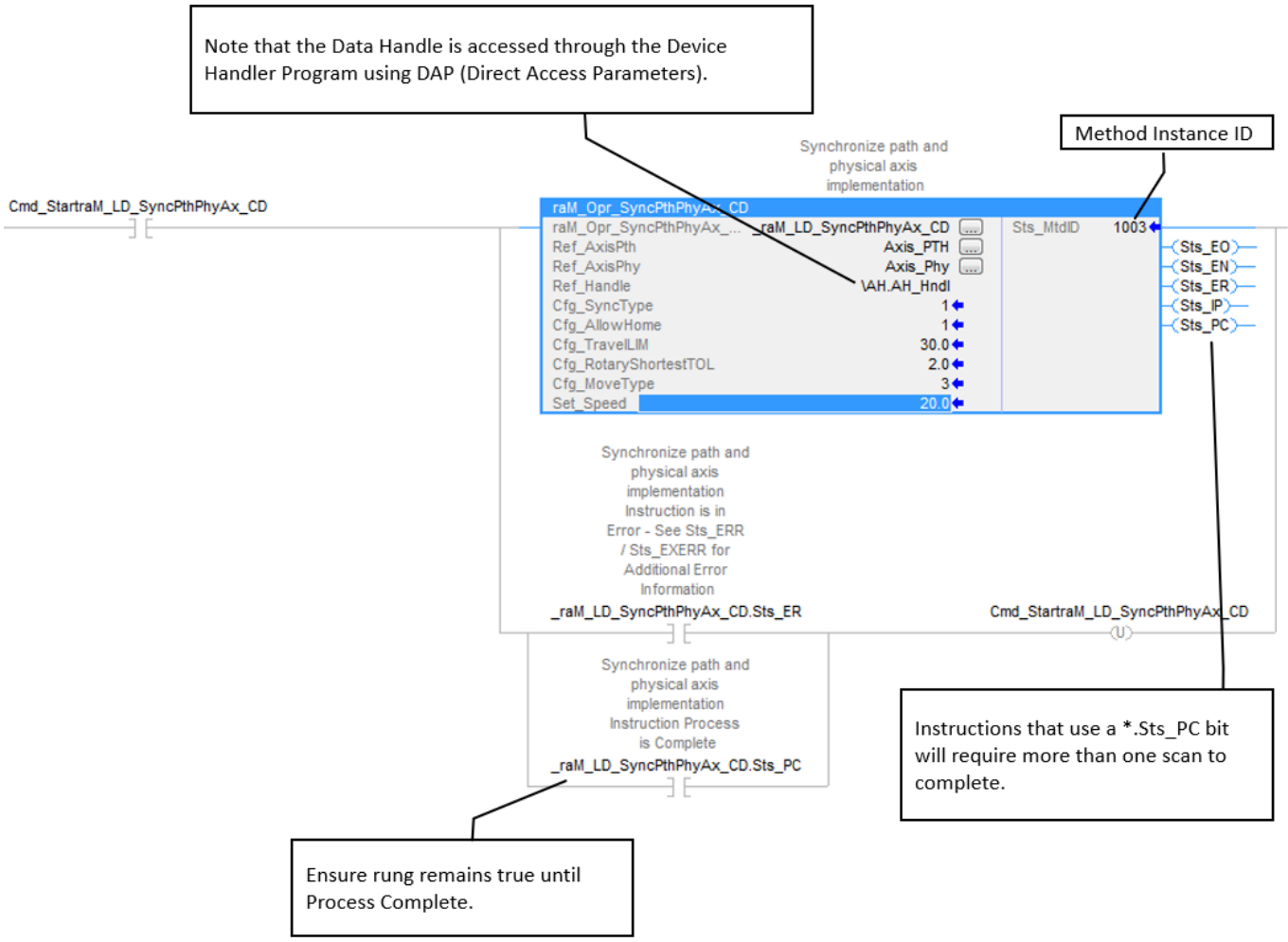
3.3 Attachments

Name	Description	File Name	Extraction path
V2_{LibraryName}	Reference Manual	RM-{LibraryName}.pdf	{ProjectName}\Documentation

4 Application

Configure a raM_Opr_SyncPthPhyAx_CD example:

- Axis recovery is allowed only up to 30 units and always forward only (if difference is less than 2 units, then it can go shortest path, no matter which direction). Recover Physical to Path Axis.



- Instruction Parameters
  - Home is Allowed during Recovery
    - Cfg_AllowHome = 1
  - Recovery is allowed only up to 30 units of position difference between Axis Path and Axis Physical
    - Cfg_TravelLIM = 30.0
  - Go shortest path if position difference between Axis Path and Axis Physical is less than 2 position units
    - Cfg_RotaryShortestTOL=2.0
  - Move Type is Rotary Positive
    - Cfg_MoveType =3
  - Configured speed = 20 units/seconds (units defined in axis definition)
    - Set_Speed = 20.0
  - Recover Physical to Path Axis
    - Cfg_SyncType = 1

- Recover to Path Instruction is a Method for the Handler
    - Handle Tag Name = AH_Hndl
    - Path Axis (Virtual Axis) = Axis_PTH
    - Physical Axis (CIP Axis) = Axis_Phy
    - In this example, the register Method ID is 1003
        - Handler ID = 1
        - Method is the 3rd to be register to this Handler

5 Appendix

General

This document provides a programmer with details on this instruction for a Logix-based controller, its Application Code Manager library content, and visualization content, if applicable. This document assumes that the programmer is already familiar with how the Logix-based 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.

IMPORTANT This object includes a Logix Designer Asset for use with Version 30 or later of Studio 5000 Logix Designer.

Common Information for All Instructions

Rockwell Automation Application Content may contain many common attributes or objects. Refer to the following reference materials for more information:

- Foundations of Modular Programming, **IA-RM001C-EN-P**

Conventions and Related Terms

Data - 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

Signal Processing - Edge and Level

This manual uses Edge and Level to describe how bit (BOOL) Commands, Settings, Configurations and Inputs to this instruction are sent by other logic and processed by this instruction.

Send/Receive Method:	Description:
Edge	• Action is triggered by "rising edge" transition of input (0-1) • Separate inputs are provided for complementary functions (such as "enable" and "disable") • Sending logic SETS the bit (writes a 1) to initiate the action; this instruction CLEARS the bit (to 0) immediately, then acts on the request if possible • LD: use conditioned OTL (Latch) to send • ST: use conditional assignment [if (condition) then bit:=1;] to send • FBD: OREF writes a 1 or 0 every scan, should use Level, not Edge Edge triggering allows multiple senders per Command, Setting, Configuration or Input (many-to-one relationship)
Level	• Action ("enable") is triggered by input being at a level (in a state, usually 1) • Opposite action ("disable") is triggered by input being in opposite state (0) • Sending logic SETS the bit (writes a 1) or CLEARS the bit (writes a 0); this instruction does not change the bit • LD: use OTE (Energize) to send • ST: use unconditional assignment [bit:= expression_resulting_in_1_or_0;] or "if-then-else" logic [if (condition) then bit:= 1; else bit:= 0;] • FBD: use OREF to the input bit Level triggering allows only one sender can drive each Level

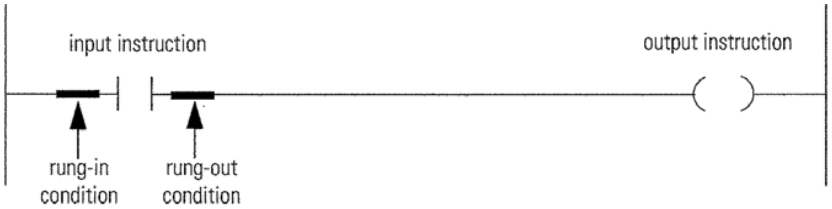
Instruction Execution - Edge and Continuous

This manual uses Edge and Continuous to describe how an instruction is designed to be executed.

Method:	Description:
Edge	• Instruction Action is triggered by "rising edge" transition of the rung-in-condition
Continuous	• Instruction Action is triggered by input being at a level (in a state, usually 1) • Opposite action is triggered by input being in opposite state (0) • Instructions designed for continuous execution should typically be used on rungs without input conditions present allowing the instruction to be continuously scanned

Relay Ladder Rung Condition

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

IMPORTANT

The rung-in condition is reflected in the EnableIn parameter and determines how the system performs each Add-On Instruction. If the EnableIn signal is TRUE, the system performs the instruction's main logic routine. Conversely, if the EnableIn signal is FALSE, the system performs the instruction's EnableInFalse routine.

The instruction's main logic routine sets/clears the EnableOut parameter, which then determines the rung-out condition. The EnableInFalse routine cannot set the EnableOut parameter. If the rung-in condition is FALSE, then the EnableOut parameter and the rung-out condition will also be FALSE.

Pre-scan

On transition into RUN, the controller performs a pre-scan before the first scan. Pre-scan is a special scan of all routines in the controller. The controller scans all main routines and subroutines during pre-scan, but ignores jumps that could skip the execution of instructions. The controller performs all FOR loops and subroutine calls. If a subroutine is called more than once, it is performed each time it is called. The controller uses pre-scan of relay ladder instructions to reset non-retentive I/O and internal values.

During pre-scan, input values are not current and outputs are not written. The following conditions generate pre-scan:

- Transition from Program to Run mode.
- Automatically enter Run mode from a power-up condition.

Pre-scan does not occur for a program when:

- Program becomes scheduled while the controller is running.
- Program is unscheduled when the controller enters Run mode.

IMPORTANT The Pre-scan process performs the Process Add-On Instruction's logic routine as FALSE and then performs its Pre-scan routine as TRUE.