Rockwell Automation Application Content

Rockwell Automation Robotics Libraries

Reference Manual

Configure – Robot

raM_Robot_Opr_ConfigureDelta V2.0

January, 2024

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_Robot_Opr_ConfigureDelta:

Instruction applies a configuration to a Robot Device Handler specifying active components and motion configuration for a delta type robot geometry.

Use when:

- Using a Device Handler for Robot Management
- Defining robot geometry parameters
- Setting the coupling relationship between joints and motors
- Setting joint position limit checking parameters
- Configuring a robot to accept commands

Do NOT use when:

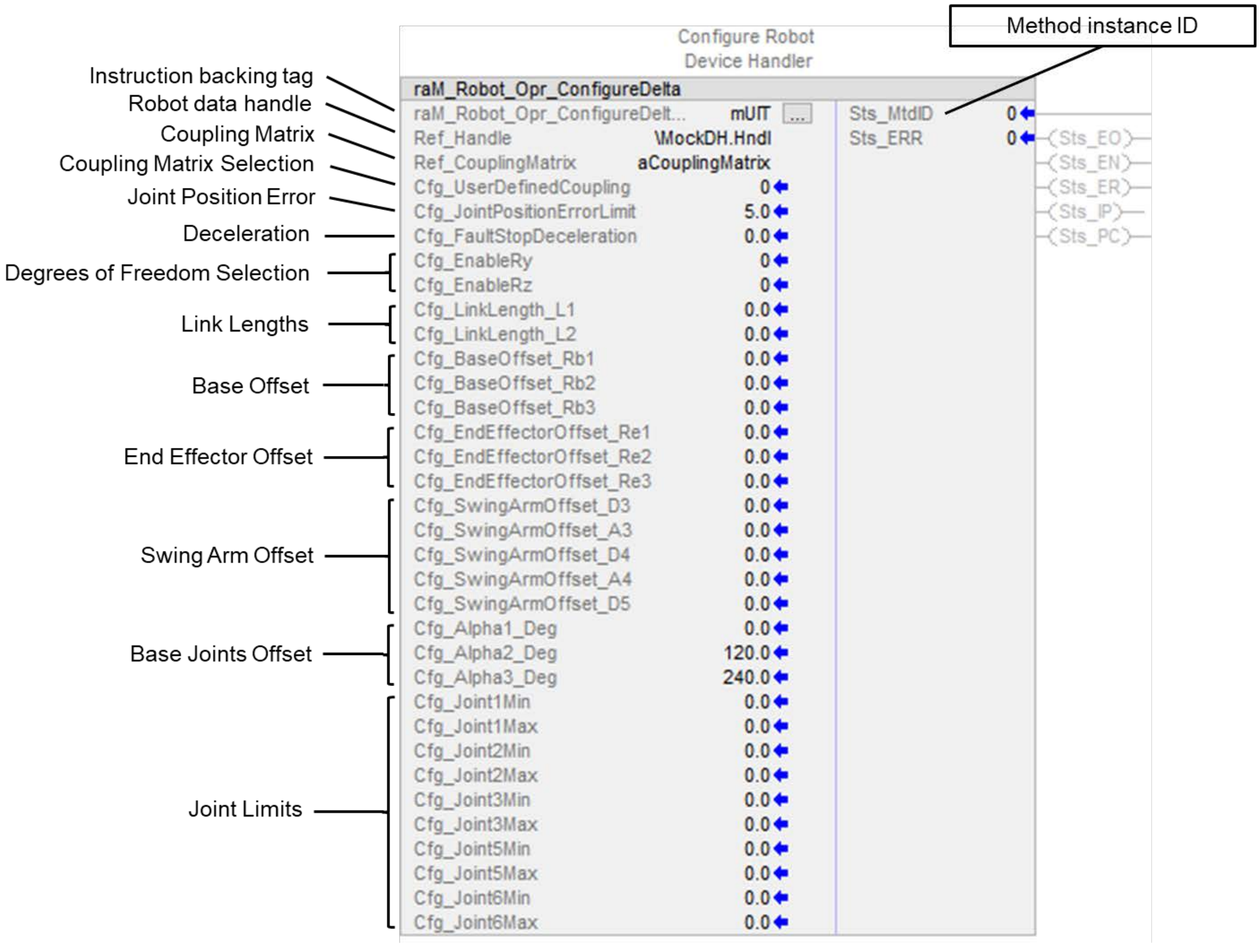
- Not using a Device Handler for Robot Management

1.1 Prerequisites

- Robot Device Handler
    - Rockwell Automation Robotics Libraries v2.0 →
- Studio 5000 – Logix Designer
    - v35.0 →
- Studio 5000 – Application Code Manager
    - v4.03.00 →

1.2 Functional Description

The instruction applies a configuration for a delta robot to a Robot Device Handler



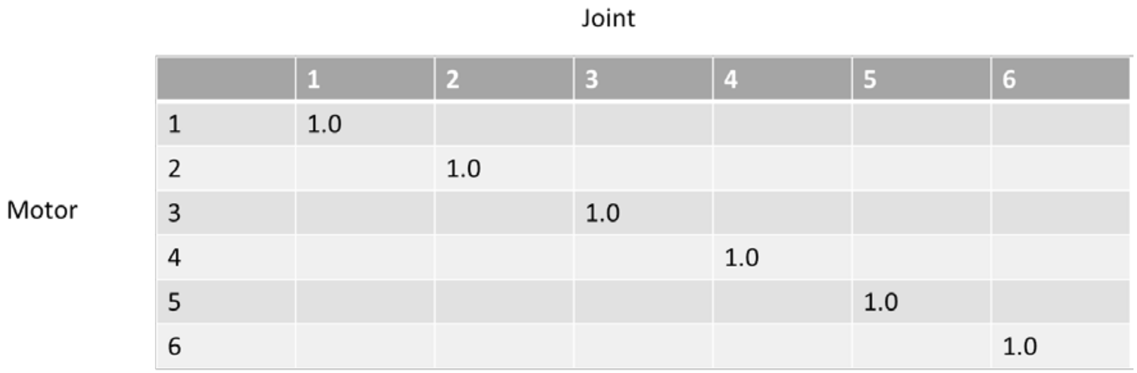
Configure Robot
Device Handler

raM_Robot_Opr_ConfigureDelta

raM_Robot_Opr_ConfigureDelt...	mUIT ...	Sts_MtdID	0
Ref_Handle	\MockDH.Hndl	Sts_ERR	0
Ref_CouplingMatrix	aCouplingMatrix		
Cfg_UserDefinedCoupling	0		
Cfg_JointPositionErrorLimit	5.0		
Cfg_FaultStopDeceleration	0.0		
Cfg_EnableRy	0		
Cfg_EnableRz	0		
Cfg_LinkLength_L1	0.0		
Cfg_LinkLength_L2	0.0		
Cfg_BaseOffset_Rb1	0.0		
Cfg_BaseOffset_Rb2	0.0		
Cfg_BaseOffset_Rb3	0.0		
Cfg_EndEffectorOffset_Re1	0.0		
Cfg_EndEffectorOffset_Re2	0.0		
Cfg_EndEffectorOffset_Re3	0.0		
Cfg_SwingArmOffset_D3	0.0		
Cfg_SwingArmOffset_A3	0.0		
Cfg_SwingArmOffset_D4	0.0		
Cfg_SwingArmOffset_A4	0.0		
Cfg_SwingArmOffset_D5	0.0		
Cfg_Alpha1_Deg	0.0		
Cfg_Alpha2_Deg	120.0		
Cfg_Alpha3_Deg	240.0		
Cfg_Joint1Min	0.0		
Cfg_Joint1Max	0.0		
Cfg_Joint2Min	0.0		
Cfg_Joint2Max	0.0		
Cfg_Joint3Min	0.0		
Cfg_Joint3Max	0.0		
Cfg_Joint5Min	0.0		
Cfg_Joint5Max	0.0		
Cfg_Joint6Min	0.0		
Cfg_Joint6Max	0.0		

A coupling matrix is used to establish the relationship between Motors and Joints. In many cases, a logical Joint is made up of a number of contributing motors. Put another way, when one motor moves, it affects the positions of multiple joints. The system must be made aware of these relationships to properly report Joint positions and to offset the relationship to avoid unintended or undesired motion. If there are not any physical couplings between multiple motors and joints, then the coupling matrix becomes a unity matrix where each motor is solely mapped to a corresponding joint. If there is not physical couplings between multiple motors/joints, the coupling matrix becomes a unity matrix where each motor is solely mapped to a single corresponding joint.

For example, a unity matrix for a 6-Degree of Freedom robot with a 1:1 relationship between motor and joints would appear as follows (all entries without a value are 0.0 but left out for readability):

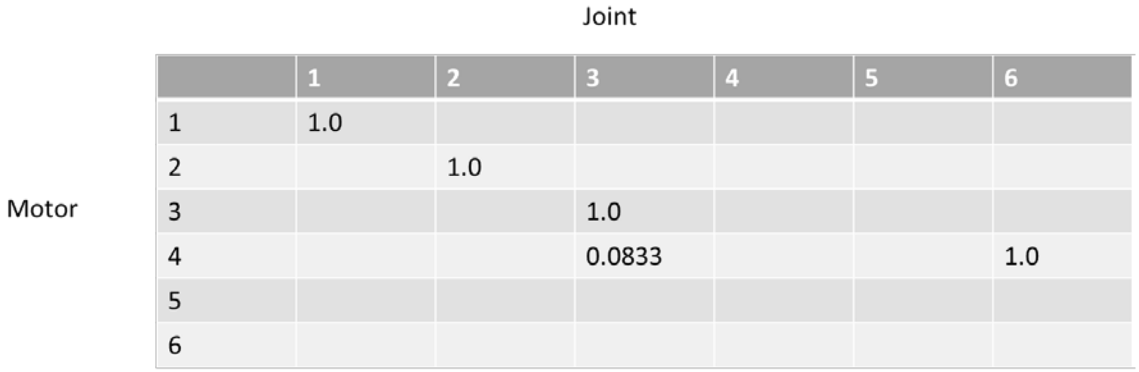


Joint

Motor

	1	2	3	4	5	6
1	1.0					
2		1.0				
3			1.0			
4				1.0		
5					1.0	
6						1.0

In a system where physical coupling is present, for example a SCARA geometry that has J3 coupled to J6, the matrix would look appear as follows:



Joint

Motor

	1	2	3	4	5	6
1	1.0					
2		1.0				
3			1.0			
4			0.0833			1.0
5						
6						

In this example Joint 3 has contributions from Motor 3 and Motor 4. The value for the coupling is determined by physical relationships. In this case, there is a 30:1 ratio between Motor 3 and Motor 4 (for every 360 degrees Motor 3 moves, Motor 4 moves 30 degrees, 30/360 = 0.0833). Note that Motors 5 and 6 and Joints 4 and 5 are not present in this system so there are no values associated with them.

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_DN	• Used when the execution of the instruction completes within a single scan. • 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-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; Process Complete. • 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

- Edge

1.3.1 Overview

Rung in condition transition response:

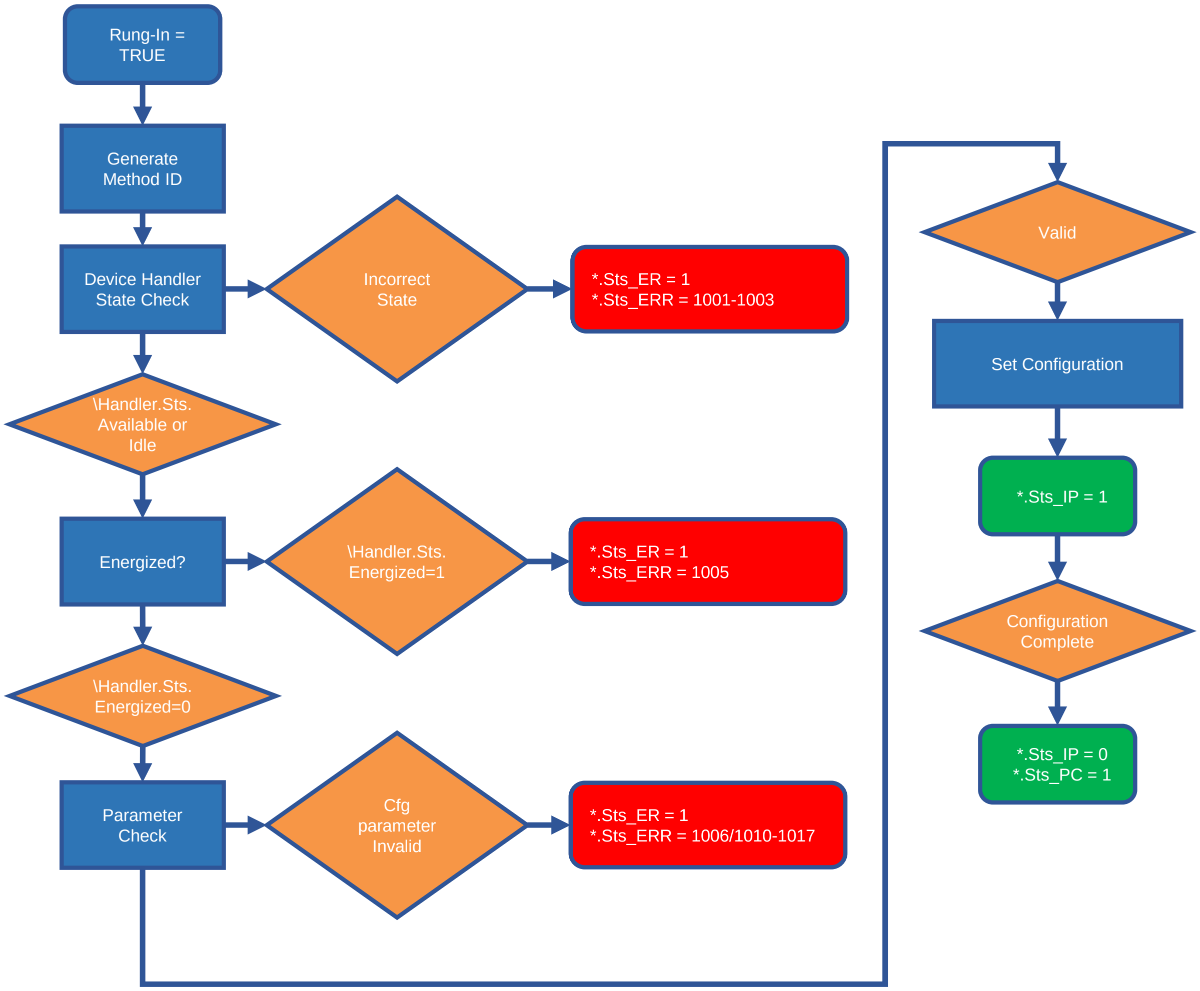
- False → True
    - Initialization
        - *.Sts_EO = 0
        - *.Sts_ER = 0
        - *.Sts_PC = 0
        - *.Sts_IP = 0
    - Running
        - *.Sts_EO = 1
        - *.Sts_EN = 1
        - *.Sts_IP = 1
            - Apply Device Handler configuration
            - IF: Configure applied successfully
                - THEN: *.Sts_PC = 1 and *.Sts_IP = 0
            - IF: Error
                - THEN: *.Sts_IP = 0 and *.Sts_PC = 0 and *.Sts_ER = 1

- True → False
    - *.Sts_EO = 0
    - *.Sts_EN = 0
    - *.Sts_IP = 0
        - IF: Error
            - THEN: *.Sts_ER = 1

1.3.2 Affected Device Handler Status

Status	Value
*.Sts.Idle	TRUE → FALSE
*.Sts.Configuring	FALSE → TRUE → FALSE
*.Sts.Available	FALSE → TRUE
*.Sts.EXERR	0

1.3.3 Execution Table



2 Instruction

2.1 Input Data

Input	Function / Description	DataType
Ref_Handle	Device Handler Data Structure	raM_UDT_ Robot_Dvc_DataHndl
Ref_CouplingMatrix	Coupling relationship between motors and joints	raM_UDT_Robot_Opr_CouplingMatrix
Cfg_UserDefinedCoupling	0 = Identity matrix for coupling, 1 = Input coupling matrix is applied	BOOL
Cfg_JointPositionErrorLimit	Maximum allowed position error on the joints	REAL
Cfg_FaultStopDeceleration	Deceleration rate for joint axes on fault	REAL
Cfg_EnableRy	Set to enable 5 axis operation	BOOL
Cfg_EnableRz	Set to enable 4 or 5 axis operation	BOOL
Cfg_LinkLength_L1	Length of link attached to each actuated joint (J1, J2, and J3) (mm)	REAL
Cfg_LinkLength_L2	Length of the parallel bar assembly attached to each Link (mm)	REAL
Cfg_BaseOffset_Rb1	Base plate offset value equal to the distance from the origin of the robot coordinate system to J1. (mm)	REAL
Cfg_BaseOffset_Rb2	Base plate offset value equal to the distance from the origin of the robot coordinate system to J2. (mm)	REAL
Cfg_BaseOffset_Rb3	Base plate offset value equal to the distance from the origin of the robot coordinate system to J3. (mm)	REAL
Cfg_EndEffectorOffset_Re1	End plate offset value equal to the distance from the center of the moving end plate to the lower spherical joints of the parallel arm for J1. (mm)	REAL
Cfg_EndEffectorOffset_Re2	End plate offset value equal to the distance from the center of the moving end plate to the lower spherical joints of the parallel arm for J2. (mm)	REAL
Cfg_EndEffectorOffset_Re3	End plate offset value equal to the distance from the center of the moving end plate to the lower spherical joints of the parallel arm for J3. (mm)	REAL
Cfg_SwingArmOffset_D3	The distance on Z axis from the center of end plate to the J4 axis of rotation. (mm)	REAL
Cfg_SwingArmOffset_A3	The distance on X axis from center of end plate to the J4 axis of rotation. (mm)	REAL
Cfg_SwingArmOffset_D4	The distance on Z axis from the J4 axis of rotation to the J5 axis of rotation. (mm)	REAL
Cfg_SwingArmOffset_A4	The distance on X axis from the J4 axis of rotation to the J5 axis of rotation. (mm)	REAL
Cfg_SwingArmOffset_D5	The distance on Z axis from the J5 axis of rotation to the EOA frame. (mm)	REAL
Cfg_Alpha1_Deg	The angle between J1 and the X-axis of the Robot Frame.(deg) Leave = 0	REAL
Cfg_Alpha2_Deg	The angle between J1 and J2 in the XY plane of the robot frame or around the Z axis in anti-clockwise direction. .(deg)	REAL
Cfg_Alpha3_Deg	The angle between J1 and J3 in the XY plane of the robot frame or around the Z axis in anti-clockwise direction. .(deg) Alpha2+Alpha3 <= 360.	REAL
Cfg_Joint1Min	Joint 1 minimum position limit (deg for revolute, mm for prismatic)	REAL
Cfg_Joint1Max	Joint 1 maximum position limit (deg for revolute, mm for prismatic)	REAL
Cfg_Joint2Min	Joint 2 minimum position limit (deg for revolute, mm for prismatic)	REAL
Cfg_Joint2Max	Joint 2 maximum position limit (deg for revolute, mm for prismatic)	REAL
Cfg_Joint3Min	Joint 3 minimum position limit (deg for revolute, mm for prismatic)	REAL
Cfg_Joint3Max	Joint 3 maximum position limit (deg for revolute, mm for prismatic)	REAL

Input	Function / Description	DataType
Cfg_Joint5Min	Joint 5 minimum position limit (deg for revolute, mm for prismatic)	REAL
Cfg_Joint5Max	Joint 5 maximum position limit (deg for revolute, mm for prismatic)	REAL
Cfg_Joint6Min	Joint 6 minimum position limit (deg for revolute, mm for prismatic)	REAL
Cfg_Joint6Max	Joint 6 maximum position limit (deg for revolute, mm for prismatic)	REAL

2.2 Output Data

Output	Function / Description	DataType
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
Sts_MtdID	Method ID	DINT
Sts_IP	Instruction is 'In Process'	BOOL
Sts_PC	Instruction Process is Complete	BOOL

2.3 Error Codes

Sts_ERR	Description
0	No errors present
1001	Device Handler is not in a running state. Commands to the device cannot be processed.
1002	Device Handler faulted
1003	Device Handler is not in a supported state. Device Handler must be in state Idle or Available
1004	Incorrect Configure used, DH hardware requires vendor specific configuration instruction
1005	Robot is energized, DH cannot be reconfigured. Power down and reapply the method
1006	Invalid coupling matrix. No inverse found.
1007	Configuration failed, DH could not message drive which may affect robot performance. Execute configure AOI and try again
1008	Device handler does not support minimum number of motors for this geometry
1009	Concurrent Command or DH is already aplying a configuration
1010	Invalid Position error limit, must be greater than 0
1011	Stop deceleration out of range, must be greater than or equal to 0
1012	Joint 1 minimum position cannot be smaller than max limit
1013	Joint 2 minimum position cannot be smaller than max limit
1014	Joint 3 minimum position cannot be smaller than max limit
1016	Joint 5 minimum position cannot be smaller than max limit
1017	Joint 6 minimum position cannot be smaller than max limit
1018	L1 link length must be > 0
1019	L2 link length must be > 0
1020	Enable Ry is only support for Delta 5 with Rz enabled
1021	Cfg_Zb2 is invalid. Angle between J1 and J2 must be greater than Zb1
1022	Cfg_Zb3 is invalid. Angle between J1 and J3 must be greater than Zb2

3 Application Code Manager

3.1 Definition Object: raM_Robot_Opr_Configure

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 Implementation Object: raM_LD_Robot_Configure

Implementation Language: Ladder
Content Type: Routine

This implement contains only a rung with an instance of the raM_Robot_Opr_Configure object.

Parameter Name	Default Value	Instance Name	Definition	Description
RoutineName	{ObjectName}	{RoutineName}	Routine	Name of the routine where the object will be placed
TagName	_{ObjectName}	{TagName}	Tag	Instruction backing tag
StartBitTagName	Cmd_{ObjectName}	{ StartBitTagName }	Local Tag	Tag name for start command enabling bit
GeometryType	ArticulatedIndependent	Delta	List	Select desired geometry configuration for implement

Linked Library

Link Name	Catalog Number	Revision	Solution	Category
RobotHandler	raM_Robot_Dvc_DeviceHandler	2	(RA-LIB) Robotics	Robot Handler
raM_Robot_Opr_Configure	raM_Robot_Opr_Configure	2	(RA-LIB) Robotics	Asset-Control

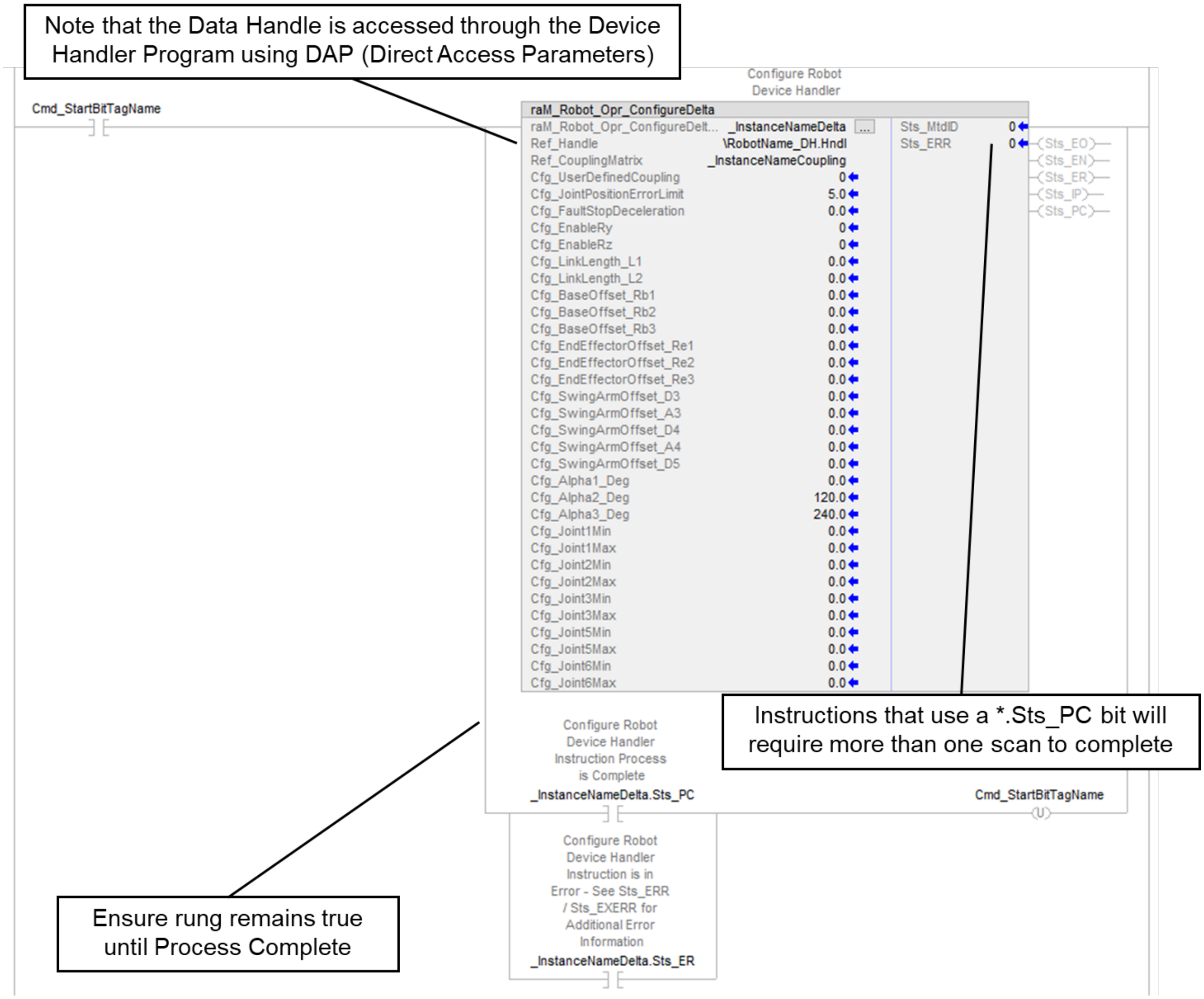
3.3 Attachments

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

4 Application

4.1 Using raM_Robot_Opr_ConfigureDelta

NOTE. The Configure instruction must be executed when the Device Handler is in Idle state or Available state to trigger a configuration or reconfiguration. The instruction will move into the Configuring state while the inputs are being applied and, if successful, transition into the Available state.



Note that the Data Handle is accessed through the Device Handler Program using DAP (Direct Access Parameters)

Cmd_StartBitTagName

Configure Robot Device Handler

raM_Robot_Opr_ConfigureDelta

Parameter	Value
raM_Robot_Opr_ConfigureDelt...	_InstanceNameDelta
Ref_Handle	\RobotName_DH.Hndl
Ref_CouplingMatrix	_InstanceNameCoupling
Cfg_UserDefinedCoupling	0
Cfg_JointPositionErrorLimit	5.0
Cfg_FaultStopDeceleration	0.0
Cfg_EnableRy	0
Cfg_EnableRz	0
Cfg_LinkLength_L1	0.0
Cfg_LinkLength_L2	0.0
Cfg_BaseOffset_Rb1	0.0
Cfg_BaseOffset_Rb2	0.0
Cfg_BaseOffset_Rb3	0.0
Cfg_EndEffectorOffset_Re1	0.0
Cfg_EndEffectorOffset_Re2	0.0
Cfg_EndEffectorOffset_Re3	0.0
Cfg_SwingArmOffset_D3	0.0
Cfg_SwingArmOffset_A3	0.0
Cfg_SwingArmOffset_D4	0.0
Cfg_SwingArmOffset_A4	0.0
Cfg_SwingArmOffset_D5	0.0
Cfg_Alpha1_Deg	0.0
Cfg_Alpha2_Deg	120.0
Cfg_Alpha3_Deg	240.0
Cfg_Joint1Min	0.0
Cfg_Joint1Max	0.0
Cfg_Joint2Min	0.0
Cfg_Joint2Max	0.0
Cfg_Joint3Min	0.0
Cfg_Joint3Max	0.0
Cfg_Joint5Min	0.0
Cfg_Joint5Max	0.0
Cfg_Joint6Min	0.0
Cfg_Joint6Max	0.0

Output	Value
Sts_MtdID	0
Sts_ERR	0

Instructions that use a *.Sts_PC bit will require more than one scan to complete

Configure Robot Device Handler Instruction Process is Complete

_InstanceNameDelta.Sts_PC

Cmd_StartBitTagName

Configure Robot Device Handler Instruction is in Error - See Sts_ERR / Sts_EXERR for Additional Error Information

_InstanceNameDelta.Sts_ER

Ensure rung remains true until Process Complete

5 Appendix

General

This document provides a programmer with details on this OEM Building Block instruction for a Logix-based controller. You should already be 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 OEM Building Block Instruction includes an Add-On Instruction for use with Version 24 or later of Studio 5000 Logix Designer.

Common Information for All Instructions

Rockwell Automation Building Blocks 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

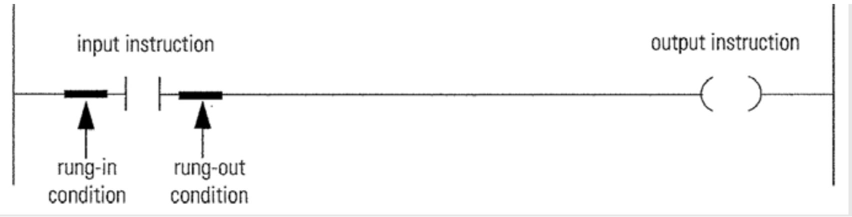
Instruction Execution - Edge and Continuous

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.