Rockwell Automation Application Content

Rockwell Automation Robotics Libraries

Reference Manual

Device Status – Robot

raM_Robot_Dvc_DeviceStatus **V2.2**

April, 2024

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_Robot_Dvc_DeviceStatus:

Instruction provides status of robot and summarizes Device Handler information.

Use when:

- Using a Device Handler for Robot Management
- Desire to view overall status of robot

Do NOT use when:

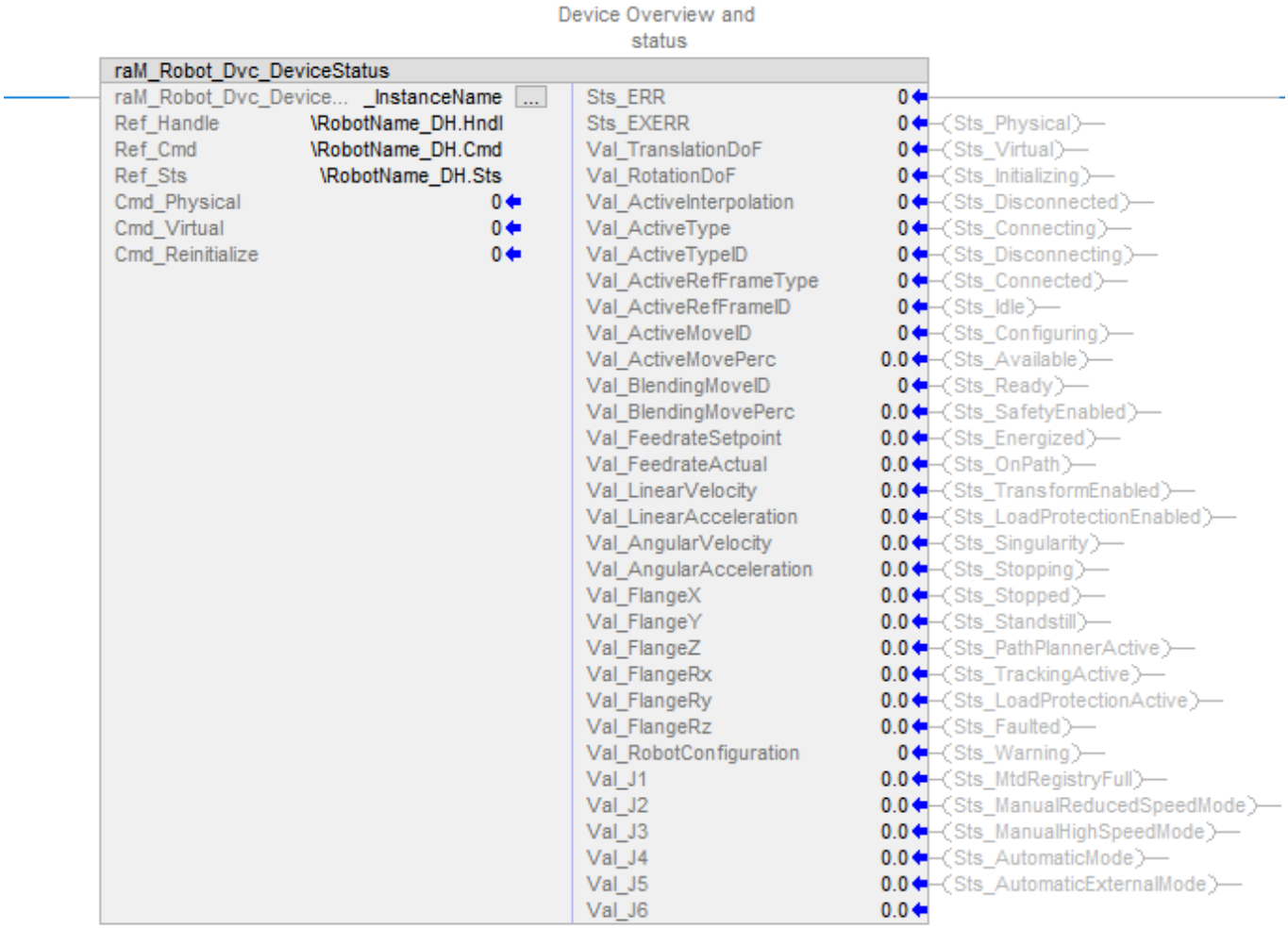
- Not using a Device Handler for Robot Management

1.1 Prerequisites

- Robot Device Handler
  - Rockwell Automation Robotics Libraries v2.2 →
- Studio 5000 – Logix Designer
  - v35.0 →
- Studio 5000 – Application Code Manager
  - v4.03.00 →

1.2 Functional Description

Instruction provides status of robot and summarizes Device Handler information.



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

- Continuous

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.Virtual	
*.Sts.Physical	

2 Instruction

2.1 Input Data

Input	Function / Description	DataType
Ref_Handle	Device Handler Data Structure	raM_UDT_Robot_Dvc_DataHndl
Ref_Cmd	Handler Command interface	raM_UDT_Robot_Dvc_Command
Ref_Sts	Handler Status interface	raM_UDT_Robot_Dvc_Status
Cmd_Physical	Input to set the Device Handler in physical mode	BOOL
Cmd_Virtual	Input to set the Device Handler in virtual mode	BOOL
Cmd_Reinitialize	Input to reinitialize the Device Handler	BOOL

2.2 Output Data

Output	Function / Description	DataType
Sts_Physical	Device Handler is in physical mode	BOOL
Sts_Virtual	Device Handler is in virtual mode	BOOL
Sts_Initializing	Device Handler in Initializing state	BOOL
Sts_Disconnected	Device Handler in Disconnected state	BOOL
Sts_Connecting	Device Handler in Connecting state	BOOL
Sts_Disconnecting	Device Handler in Disconnecting state	BOOL
Sts_Connected	Device Handler in Connected state	BOOL
Sts_Idle	Device Handler in Idle state	BOOL
Sts_Configuring	Device Handler in Configuring state	BOOL
Sts_Available	Device Handler in Available state	BOOL
Sts_Ready	Device Handler is ready for motion	BOOL
Sts_SafetyEnabled	All axes are in operational safety state	BOOL
Sts_Energized	Robot is energized	BOOL
Sts_OnPath	Motor axes are synchronized with path axes	BOOL
Sts_TransformEnabled	Bidirectional transform between Cartesian and Joint axes of a Robot system	BOOL
Sts_LoadProtectionEnabled	Load protection is configured and running	BOOL
Sts_Singularity	Current Pose represents a singularity condition	BOOL
Sts_Stopping	Robot is decelerating to a stop	BOOL
Sts_Stopped	Robot is stopped. Path planner is inactive	BOOL
Sts_Standstill	All robot axes are below configure standstill speed	BOOL
Sts_PathPlannerActive	Path points are being processed	BOOL
Sts_TrackingActive	Actively tracking external axes	BOOL
Sts_LoadProtectionActive	Load protection is actively reduing the robot feedrate	BOOL
Sts_Faulted	Device Handler is faulted	BOOL
Sts_Warning	Device Handler has a alarm/warning	BOOL
Sts_MtdRegistryFull	Method Registry array is full	BOOL
Sts_ManualReducedSpeedMode	rOS move operations allowed	BOOL
Sts_ManualHighSpeedMode	rOS move operations allowed	BOOL
Sts_AutomaticMode	rOS move operations allowed	BOOL
Sts_AutomaticExternalMode	rOS move operations disabled	BOOL
Sts_ERR	Error code	DINT

Output	Function / Description	DataType
Sts_EXERR	Extra Error Code	DINT
Val_TranslationDoF	Number of degrees of freedom for translational motion	SINT
Val_RotationDoF	Number of degrees of freedom for rotational motion	SINT
Val_ActiveInterpolation	0 - PTP 1- CP-L 2- CP-W	DINT
Val_ActiveType	Flange, 3 Tool, 4	DINT
Val_ActiveTypeID	Frame unique identifier (for tool frames)	DINT
Val_ActiveRefFrameType	Reference frame type (0: World, 1: Robot, 2: Flange, 3: Tool, 4: User)	DINT
Val_ActiveRefFrameID	Reference frame unique identifier (for tool and user frames)	DINT
Val_ActiveMoveID	Move ID of active path point	DINT
Val_ActiveMovePerc	Percentage along active move	REAL
Val_BlendingMoveID	Move ID of blending path point	DINT
Val_BlendingMovePerc	Percentage along blended move	REAL
Val_FeedrateSetpoint	Feed rate set point in %	REAL
Val_FeedrateActual	Actual feed rate in %	REAL
Val_LinearVelocity	Linear Velocity of flange mm/s	REAL
Val_LinearAcceleration	Linear Acceleration of flange mm/s2	REAL
Val_AngularVelocity	Angular Velocity of flange deg/s	REAL
Val_AngularAcceleration	Angular Acceleration of flange deg/s2	REAL
Val_FlangeX	X coordinate of Flange position in mm	REAL
Val_FlangeY	Y coordinate of Flange position in mm	REAL
Val_FlangeZ	Z coordinate of Flange position in mm	REAL
Val_FlangeRx	Rotation about X-axis in degrees	REAL
Val_FlangeRy	Rotation about Y-axis in degrees	REAL
Val_FlangeRz	Rotation about Z-axis in degrees	REAL
Val_RobotConfiguration	Current configuration	DINT
Val_J1	Joint J1 value in degrees	REAL
Val_J2	Joint J2 value in degrees	REAL
Val_J3	Joint J3 value in degrees	REAL
Val_J4	Joint J4 value in degrees	REAL
Val_J5	Joint J5 value in degrees	REAL
Val_J6	Joint J6 value in degrees	REAL

2.3 Error Codes

Sts_ERR	Description
0	No errors present
<Number>	See errors in Device Handler manual

3 Application Code Manager

3.1 Definition Object: raM_Robot_Dvc_DeviceStatus

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_Dvc_DeviceStatus

Implementation Language: Ladder
Content Type: Routine

This implement contains only a rung with an instance of the raM_Robot_Dvc_DeviceStatus 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

Linked Library

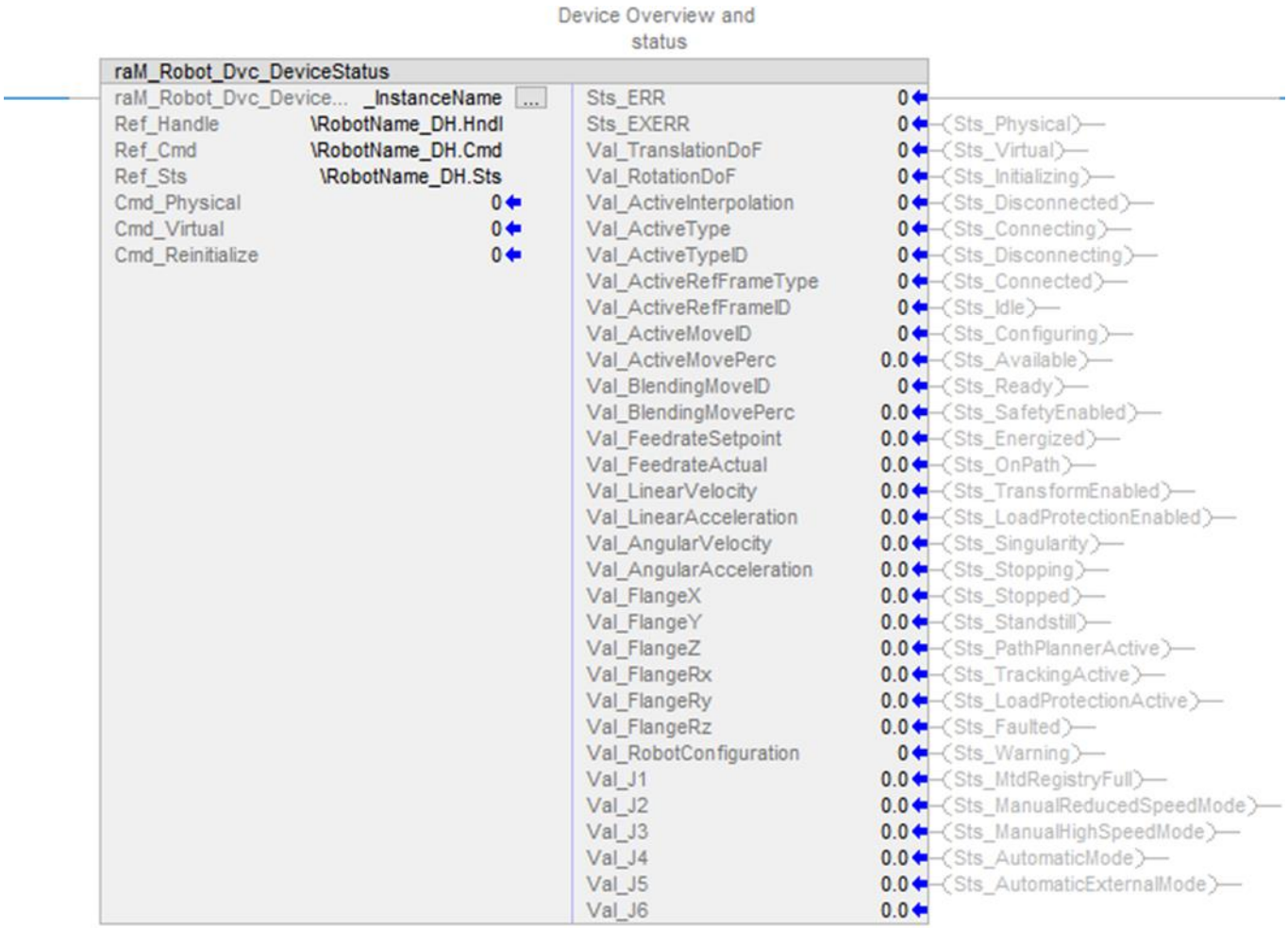
Link Name	Catalog Number	Revision	Solution	Category
RobotHandler	raM_Robot_Dvc_DeviceHandler	2	(RA-LIB) Robotics	Robot Handler
raM_Robot_Dvc_DeviceStatus	raM_Robot_Dvc_DeviceStatus	2	(RA-LIB) Robotics	Asset-Control

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_Dvc_DeviceStatus



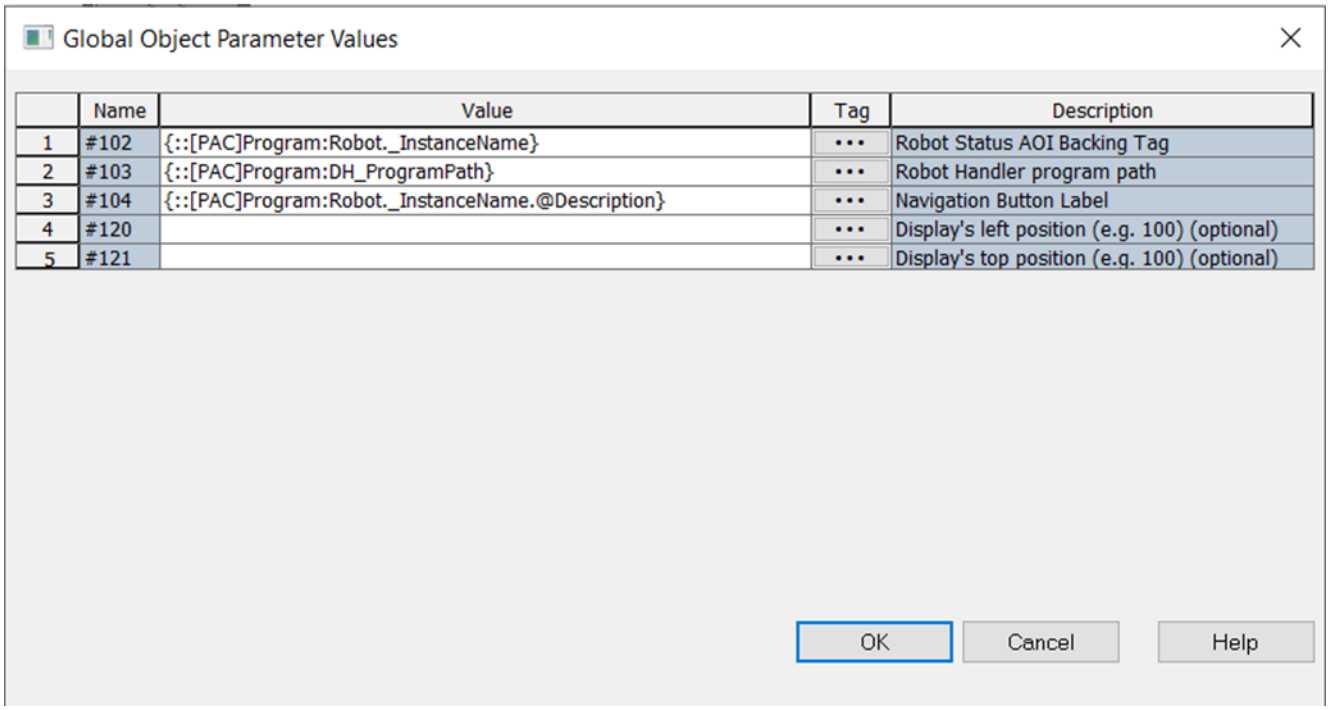
5 Optional User Interface

The optional faceplate files allow you to quickly load, configure and use a preconfigured status and diagnostic display available for FactoryTalk View Machine Edition. This faceplate is supported in version v2.2 → of this AOI.

5.1 Faceplate Launch Button

The Launch button is available on the user developed screen of the FactoryTalk View Studio application. When the button is pressed, it will launch the Robot Status device object faceplate with default home tab.

The Global Object file contains the “call-to-action” button that allows navigation to the faceplate display.

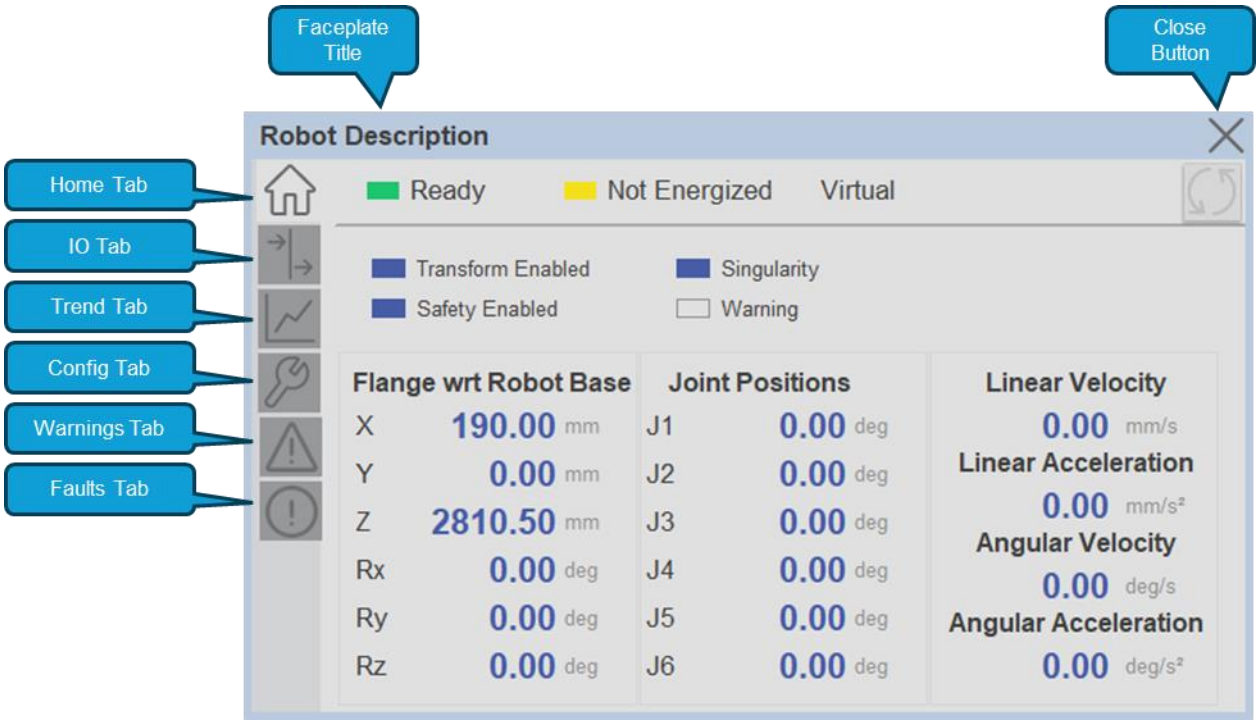


- #102 – raM_Robot_Dvc_DeviceStatus Instruction Backing Tag
- #103 – Robot Handler Program Path
- #104 – Device Object Instruction Backing Tag Description
- #120 – Display’s left position
- #121 – Display’s right Position

When the application launches, the faceplate’s Launch Button caption and the faceplate title are linked to the raM_Robot_Dvc_DeviceStatus AOI instance tag and _InstanceName.@Description.

Name	Data Type	Class	Description
▸ _InstanceName	raM_Robot_Dvc_DeviceStatus	Standard	Robot Description

5.2 Faceplate Navigation



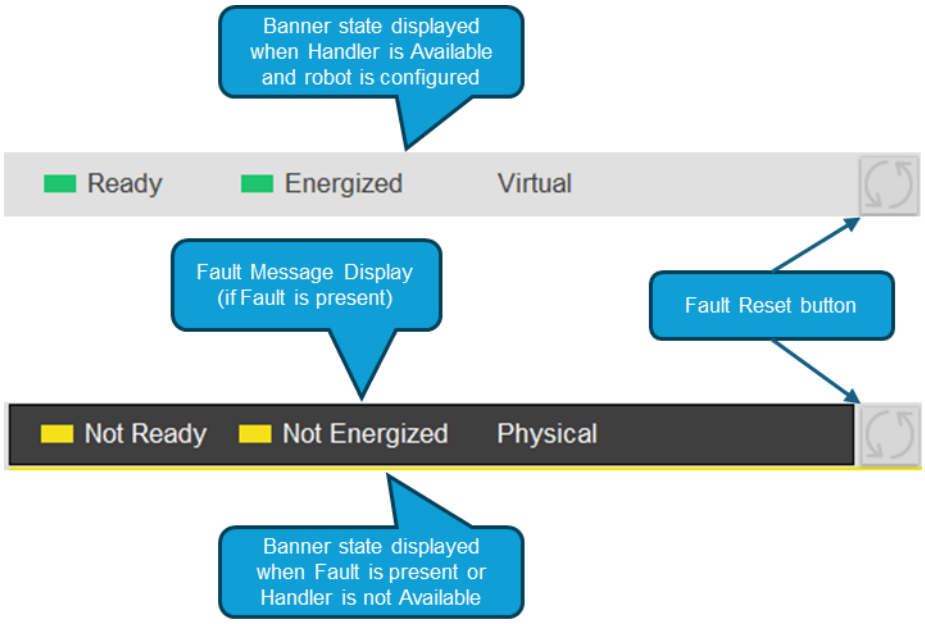
Tab descriptions:

- The **Home** tab provides a robot status summary.
- The **I/O** Tab provides the status indication.
- The **Trend** tab displays real-time values of linear and angular velocity for the robot.
- The **Config** tab provides trend configuration along with a limited number of functions provided by the Robot Device Status AOI.
- The **Warnings** tab displays a running history of the last (8) warnings with descriptions.
- The **Faults** tab displays a running history of the last (8) faults and alarms with descriptions.

5.3 Common Status

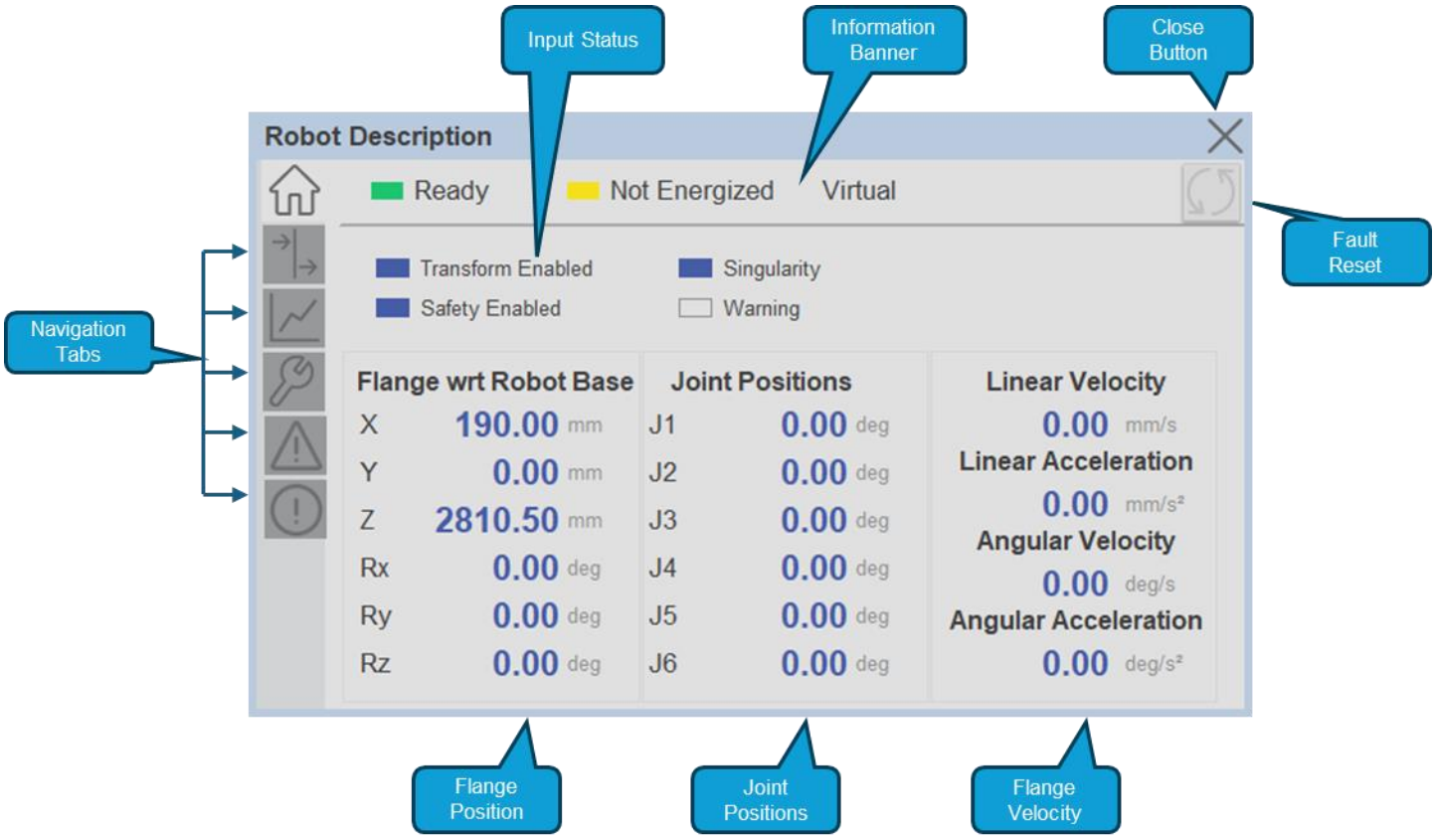
A banner will be displayed at the top of each tab to provide the following information of the device:

1. Ready or Not Ready
2. Faulted (device will show Not Ready and fault message)
3. Virtualized or Physical
4. Robot Energized or Not Energized
5. Device Handler availability



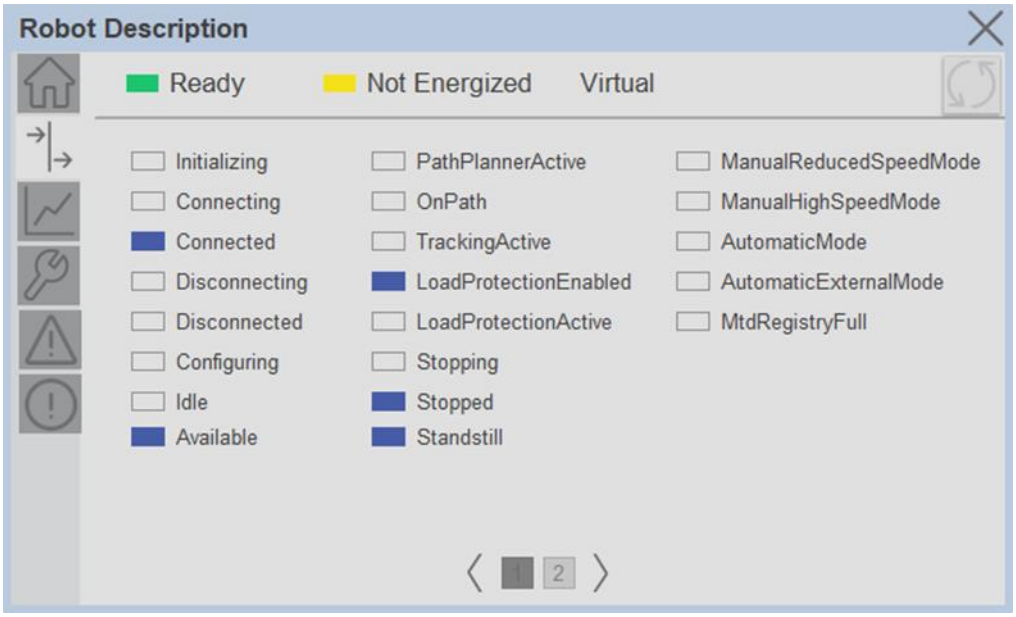
5.4 Home Tab

The Home tab is the main tab of the faceplate. It contains Numeric display objects for position and velocity. A limited number of input status points are also displayed. Additional inputs can be viewed by navigating to the IO tab.

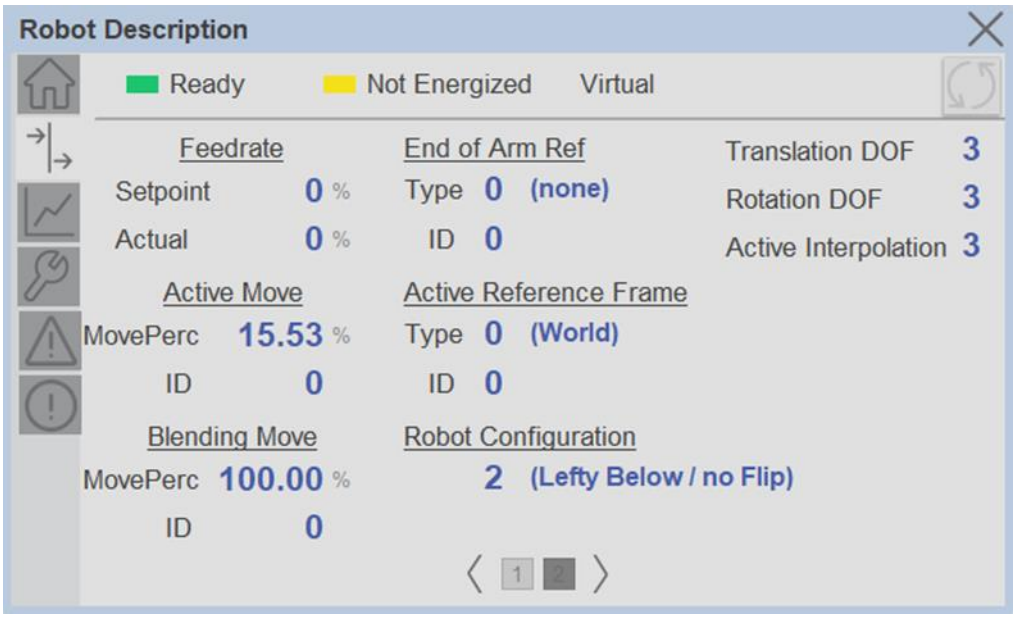


5.5 IO Tab

The I/O tab displays the on/off status of the device handler's I/O as well as additional numerical status information. The LEDs show no color when they are in the OFF condition and show blue when they are in the ON condition. Page 1 of the IO Tab displays the discrete information obtained from raM_Robot_Dvc_DeviceStatus.

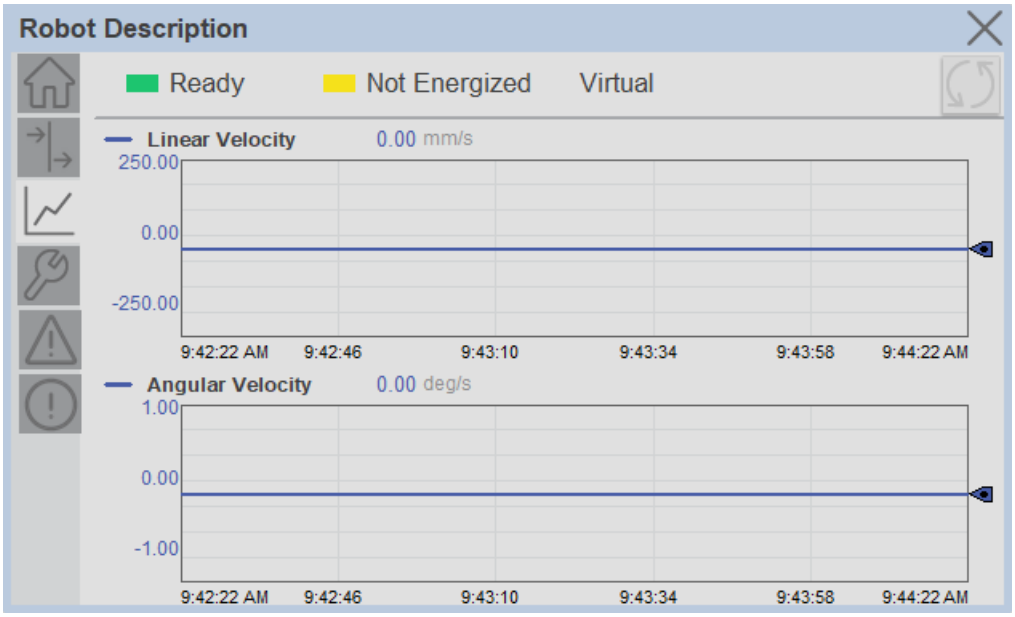


Page 2 of the IO Tab displays the numeric information obtained from raM_Robot_Dvc_DeviceStatus.



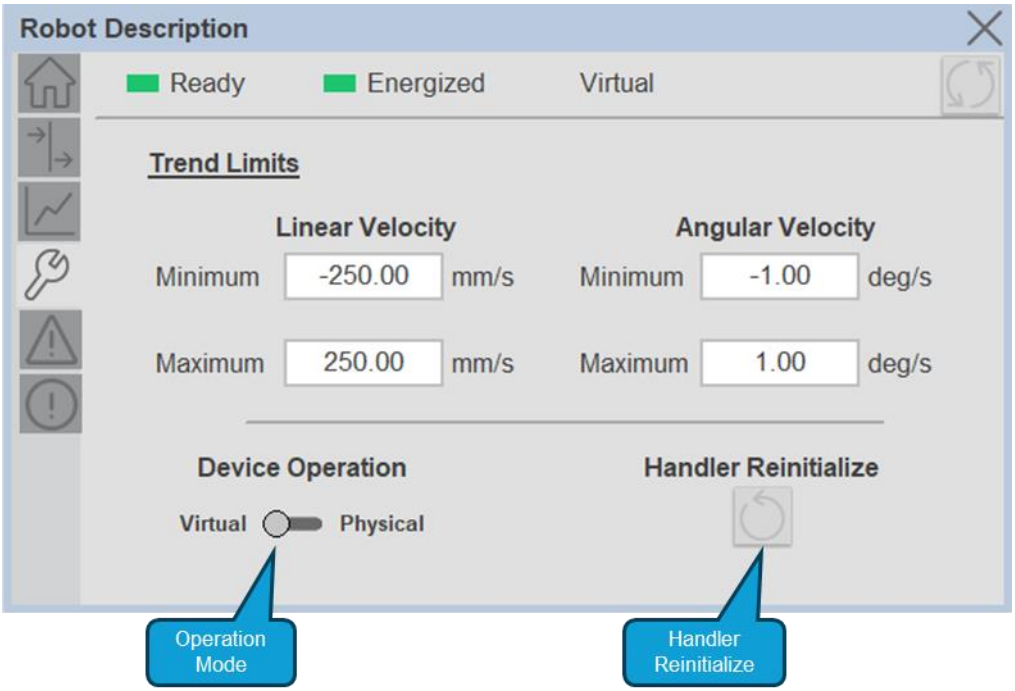
5.6 Trend Tab

Trends display values over time. They are often used to compare similar or related values and to allow operators to predict future states to make control action decisions. Two values are trended on this screen. The top trend plots the flange linear velocity, and the bottom trend plots the flange angular velocity.



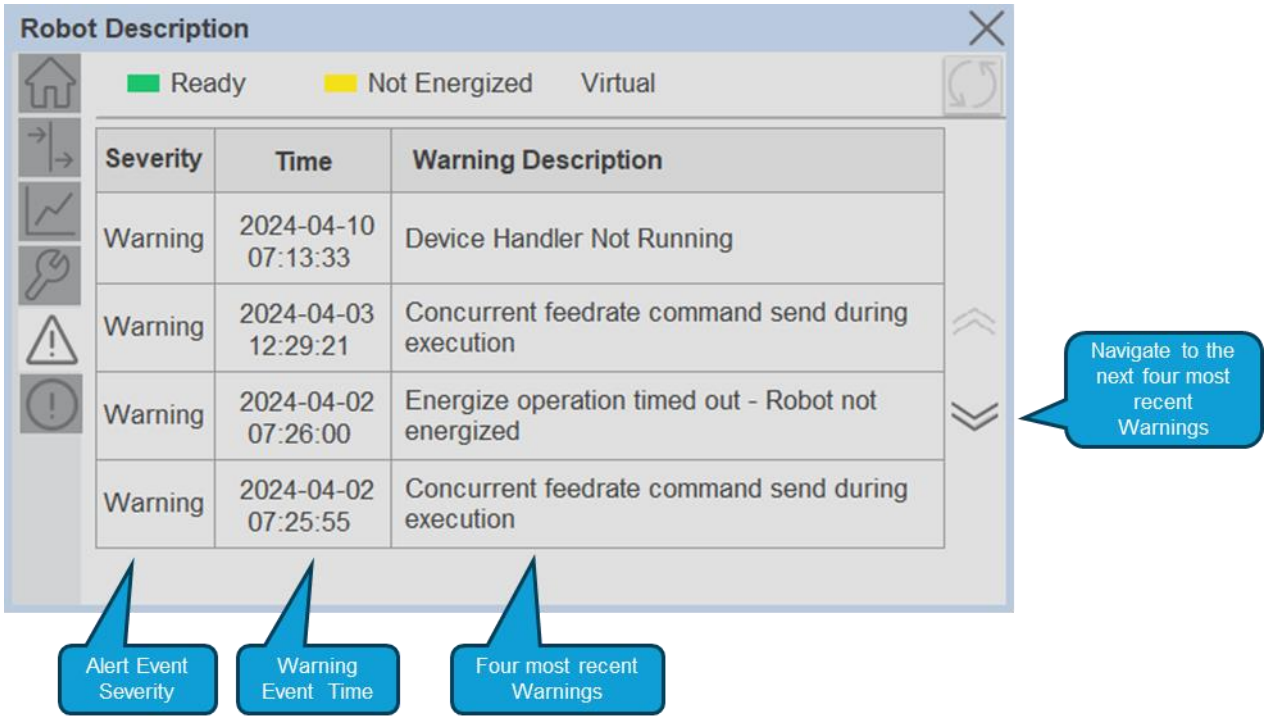
5.7 Config Tab

The Config display provides the ability to change the velocity scaling on the two trend charts. When the Device Handler is in the "Available" state and the robot is deenergized, the device operation can be switched between "Physical" and "Virtual" modes from this display. In addition, if required, the Device Handler can be re-initialized.



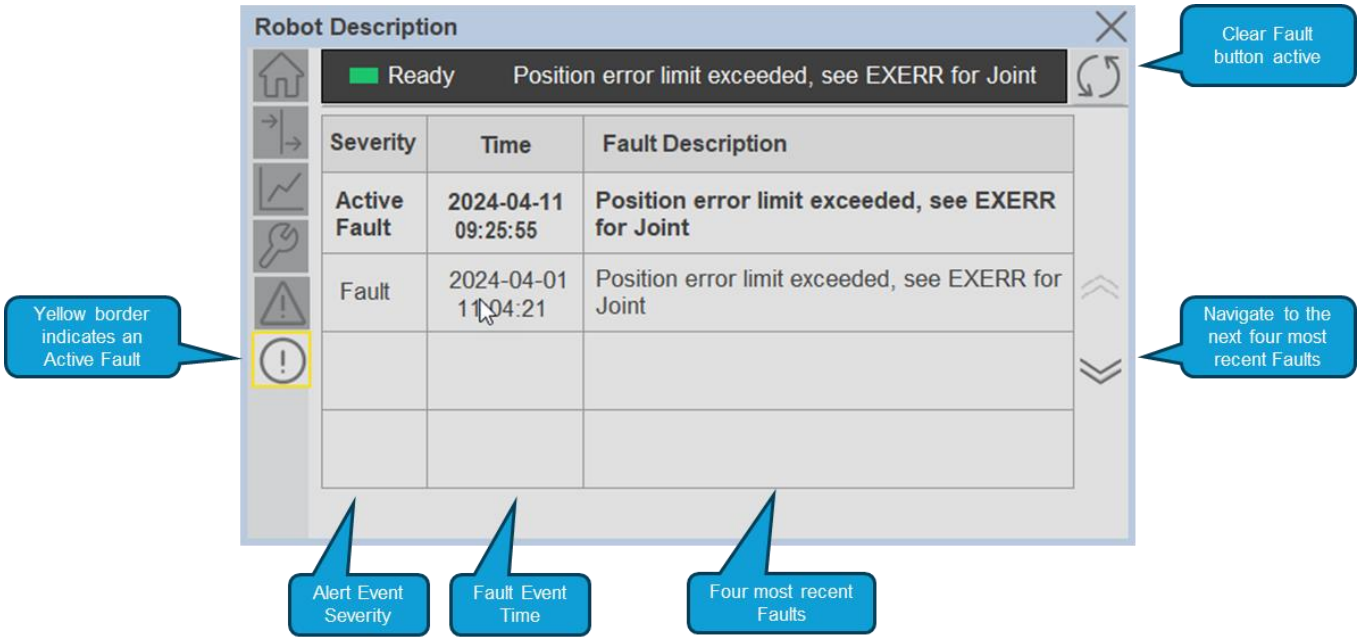
5.8 Warning Tab

The warnings tab displays the Severity level (Warning), date, time, and a description of the warning. Rows 1-4 display warnings sequentially as they are captured. Navigating to the second page, using the controls on the right side of the display, shows the next 4 rows of warnings.



5.9 Faults Tab

The faults tab displays the Severity level (Fault or Active Fault), date, time, and a description of the fault. Note, only page 1, row 1 on the Faults Summary will display the “Active Fault” in the severity column if there is a current active fault, otherwise it will display the last fault. Rows 2-4 only display past faults, not an active fault. Navigating to the second page, using the controls on the right side of the display, shows the next 4 rows of faults.



6 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.