



PlantPAx Equipment Module (EM) and Equipment Phase (EP)



Allen-Bradley

by ROCKWELL AUTOMATION

User Manual

Original Instructions

Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

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

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

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

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

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



WARNING: Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.



ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.

IMPORTANT Identifies information that is critical for successful application and understanding of the product.

These labels may also be on or inside the equipment to provide specific precautions.



SHOCK HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.



BURN HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.



ARC FLASH HAZARD: Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE).

The following icon may appear in the text of this document.



Identifies information that is useful and can help to make a process easier to do or easier to understand.

Table of Contents

	About This Publication	5
	Summary of Changes	5
	Download Firmware, Add-On Profile, EDS, and Other Files	5
	Additional Resources	5
Overview	Prerequisites	7
	Overview	7
	Equipment Phase (EP) Example System	8
	Equipment Module (EM) Example System	8
Equipment Phases	Overview	9
	raP_Opr_EPGen	
	Block Configuration	10
	Inp_PermOK and Inp_NBPermOK	10
	Inp_IntlkOK and Inp_NBIntlkOK	11
	Inp_IntlkAvailable	13
	Inp_IntlkTriplnh	15
	Sts_BypActive	16
	Inp_RdyReset	17
	Inp_DvcAlms	18
	Inp_PromptRdy	19
	Inp_EnableNavTrans	20
	MSet_Step (Manual Step)	22
	MCmd_StateForce (Force State Complete)	24
	EP Control Strategy (defaults)	24
	EP Routines (defaults)	25
	Use Case Examples	33
	Equipment Phase One - TK001_EP_TXIn_PHS	33
	Routine Structure - TK001_EP_TXIn_PHS	35
	PhaseCommands - TK011_EP_TXIn_PHS	36
	S88 State Routines - TK011_EP_TXIn_PHS	37
	Equipment Phase Two - TK001_EP_AgiCtrl	38
	Equipment Phase Three - TK001_EP_TXOut	39
Equipment Modules	Overview	41
	raP_Opr_EMGen	
	Block Configuration	41
	Inp_PermOK and Inp_NBPermOK	42
	Inp_IntlkOK and Inp_NBIntlkOK	44
	Inp_IntlkAvailable	47
	Inp_IntlkTriplnh	48
	Sts_BypActive	50
	Inp_RdyReset	51
	Inp_DvcAlms	53
	Inp_PromptRdy	54
	Inp_EnableNavTrans	54
	MSet_Step (Manual Step)	56
	MCmd_StateForce (Force State Complete)	57
	State Settings (Maintenance Configuration)	57
	EM Control Strategy (defaults)	58
	EM Routines (defaults)	59

	Use Case Examples.	65
	Equipment Module One - TK011_EM_TXIn	65
	Routine Structure - TK011_EM_TXIn	66
	StateModel - TK011_EM_TXIn	67
	Equipment Module Two - TK011_EM_AgiCtrl.	68
	Equipment Module Three - TK011_EM_TXOut.	69
Additional Configuration	Equipment Module	
	State Management	71
	Parameters and Reports	73
	Parameters	73
	Reports.	75
	Extended Alarms.	77
	Operator Prompts.	79
	Configuration From Faceplate.	80
	Display Values	81
	Input Values.	82
	Selection Tab.	84
	Responses Tab	85

About This Publication

This publication describes the use of raP_Opr_EPGen and raP_Opr_EMGen and programming examples.

Summary of Changes

This publication contains the following new or updated information. This list includes substantive updates only and is not intended to reflect all changes.

Topic	Page
Updated EM Parameter section and screen capture	73
Updated EM Reports section and screen capture	75

Download Firmware, Add-On Profile, EDS, and Other Files

Download firmware, associated files (such as Add-on Profile, EDS, and DTM), and access product release notes from the Product Compatibility and Download Center at rok.auto/pcdc.

Additional Resources

These documents contain additional information concerning related products from Rockwell Automation. You can view or download publications at rok.auto/literature.

Resource	Description
Rockwell Automation Library of Process Objects, publication PROCES-RM200	Describes the use of the Library of Process Objects and the Add-On Instruction in the Library of Process Objects.
PlantPAx Faceplates for Process Controller Instructions, publication PROCES-RM203	Describes the PlantPAx® Process instructions, and associated faceplates that are available to develop applications.
EtherNet/IP Network Devices User Manual, ENET-UM006	Describes how to configure and use EtherNet/IP™ devices to communicate on the EtherNet/IP network.
Ethernet Reference Manual, ENET-RM002	Describes basic Ethernet concepts, infrastructure components, and infrastructure features.
System Security Design Guidelines Reference Manual, SECURE-RM001	Provides guidance on how to conduct security assessments, implement Rockwell Automation products in a secure system, harden the control system, manage user access, and dispose of equipment.
UL Standards Listing for Industrial Control Products, publication CMPNTS-SR002	Assists original equipment manufacturers (OEMs) with the construction of panels, to help confirm that they conform to the requirements of Underwriters Laboratories.
American Standards, Configurations, and Ratings: Introduction to Motor Circuit Design, publication IC-AT001	Provides an overview of American motor circuit design based on methods that are outlined in the NEC.
Industrial Components Preventive Maintenance, Enclosures, and Contact Ratings Specifications, publication IC-TD002	Provides a quick reference tool for Allen-Bradley® industrial automation controls and assemblies.
Safety Guidelines for the Application, Installation, and Maintenance of Solid-state Control, publication SGI-1.1	Designed to harmonize with NEMA Standards Publication No. ICS 1.1-1987 and provides general guidelines for the application, installation, and maintenance of solid-state control in the form of individual devices or packaged assemblies incorporating solid-state components.
Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1	Provides general guidelines for installing a Rockwell Automation industrial system.
ProposalWorks configuration software, rok.auto/systemtools	Helps configure complete, valid catalog numbers and build complete quotes that are based on detailed product information.
Rockwell Automation Global SCCR tool, rok.auto/sccr	Provides coordinated high-fault branch circuit solutions for motor starters, soft starters, and component drives.
Product Certifications website, rok.auto/certifications	Provides declarations of conformity, certificates, and other certification details.

Notes:

Overview

Prerequisites

Before you build a system with the guidelines that are contained in this publication and use the raP_Opr_EMGen and raP_Opr_EPGen objects, familiarize yourself with the following:

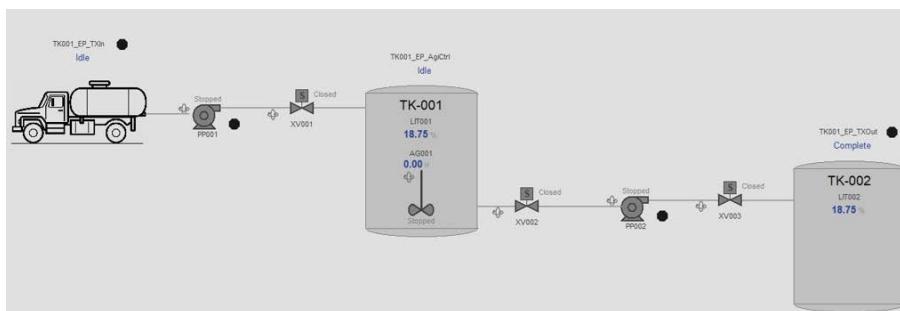
- ISA-88 guidelines and principles, with emphasis on proper application of equipment modules and equipment phases
- S88 phase state model
- How the Equipment PhaseManager™ functions
- How to create code, either via Application Code Manager (ACM) or import of raP_Opr_EMGen/raP_Opr_EPGen control strategies from base PlantPax® DCS download
- When using the raP_Opr_EPGen object, an understanding of how external sources, including FactoryTalk® Batch and SequenceManager™, can be used with Equipment PhaseManager

Install the PlantPax process library, version 5.xx, from the Rockwell Automation Product Compatibility and Download Center (PCDC).

Overview

This document provides descriptions and examples of two systems that achieve the same goal. One system is controlled using the raP_Opr_EPGen Equipment Phase block and the other is controlled using the raP_Opr_EMGen Equipment Module block. The goal in either case is the transfer of liquid from one location to another. The differences between these two examples are explained further in this guide with use case examples.

This document does not cover the organizational tree objects such as the Bus and OrgView. For information on those tree objects, see the PlantPax Distributed Control System Configuration and Implementation User Manual, publication [PROCES-UM100](#).



These are examples. You can always add code specific to your application.

Equipment Phase (EP) Example System

In the Equipment Phase (EP) example system, liquid is unloaded from a truck into mixing tank TK-001. In TK-001, the liquid is agitated and then dispensed into holding tank TK-002. The system is split into three equipment phases:

1. TK001_EP_TXIn commands the objects that dispense liquid into the mixing tank.
2. TK001_EP_AgiCtrl commands the objects that agitate and monitors the liquid level inside the tank.
3. TK001_EP_TXOut commands the objects that transfer the liquid from the mixing tank to the holding tank.

Equipment Module (EM) Example System

In the Equipment Module (EM) example, liquid is unloaded from a truck into mixing tank TK-011. In TK-011, the liquid is agitated and then dispensed into holding tank TK-012. The system is split into three equipment modules:

1. TK011_EM_TXIn commands the objects that dispense liquid into the mixing tank.
2. TK011_EM_AgiCtrl is responsible for the objects that agitate the liquid and monitor liquid level inside the tank.
3. TK011_EM_TXOut is responsible for the objects that transfer the liquid from the mixing tank to the holding tank.

Equipment Phases

Overview

Equipment phases (EPs) interact with the raP_Opr_EPGen block in the code. You can control the EP through the following methods: external sources such as FactoryTalk® Batch application, controller logic, or operator or maintenance modes.

If the program or sequence code issuing commands to the phase owns the phase, the command source is listed as 'Program.'



If the operator owns the phase, the command source is listed as 'Operator.' In this configuration, the operator can make commands to the EP via the HMI faceplate.

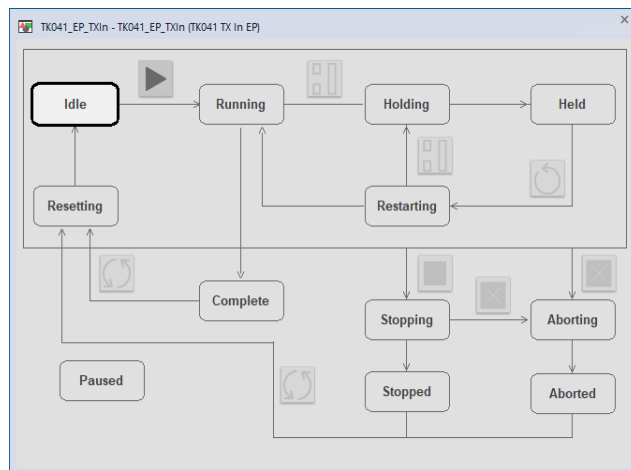


When FTBatch is used, the command source changes from 'Operator' or 'Program' to 'Batch.' In the controller, when FTBatch owns the phase, the 'External Sequencer' owns the EP.

ISA-88 State Model standards determine state control. Depending upon the ownership setting, you can shift the phase state into another permitted phase state through an external source, a program command, or an operator interaction.



ISA-88 Batch Control standards dictate all transitions between states. To transfer from one phase state to another, you must follow a path on the following chart:



For example, from the running state, an EP could go into one of the following states:

- Holding
- Stopping
- Aborting
- Complete

An EP is not able to go from the running state directly to an idle state — as this configuration goes against the ISA-88 state transition diagram. The availability of the operator HMI state commands changes based on the current state of the EP.

In the idle state, when all other necessary conditions required to start an EP are met — such as Object Ready, Interlocks OK — the only operator command available on the HMI is the start command. The start command is the only command available because you can only transition to the running phase state from the idle state.

raP_Opr_EPGen Block Configuration

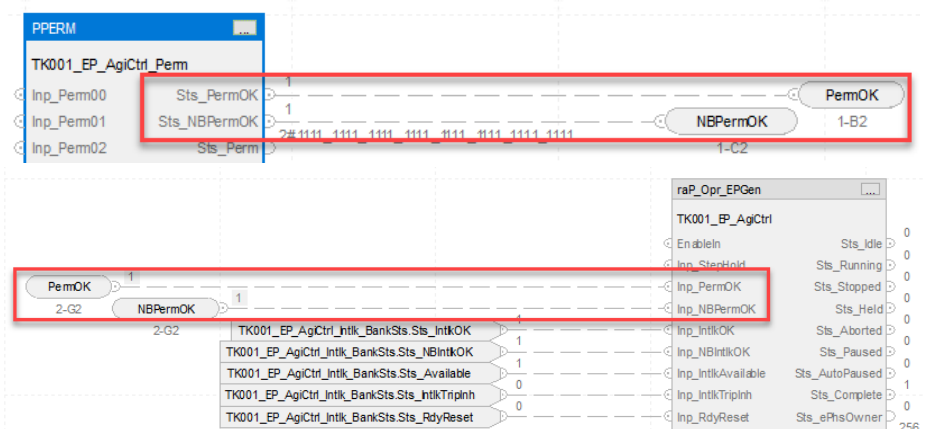
Before writing EM/EP sequence code, the raP_Opr_EPGen must be configured to allow for proper functionality.

IMPORTANT Rockwell Automation recommends setting individual permissives and individual interlocks to the same state, whether 0 or 1.
You can configure either state based on the needs of your environment.

Inp_PermOK and Inp_NBPermOK

The Inp_PermOK and Inp_NBPermOK parameters monitor the status of the EP's permissives on the associated permissive blocks. To monitor the status of the associated permissive block properly, a wire connection must be made between the following:

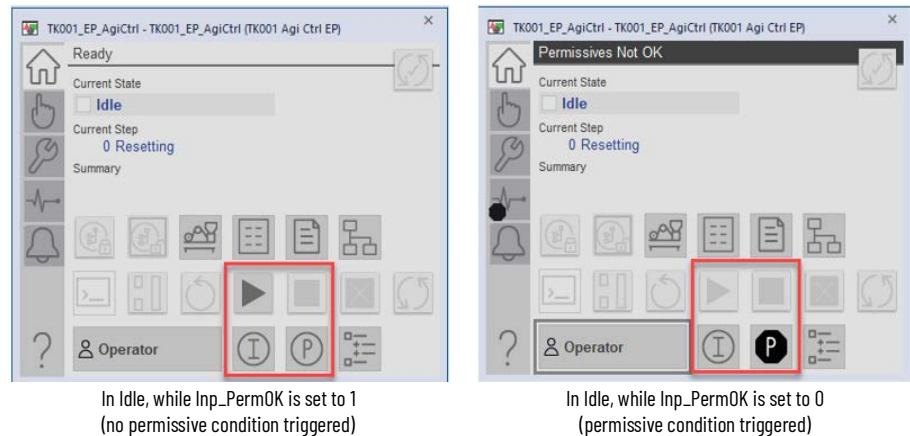
- PPERM Sts_PermOK output and the raP_Opr_EPGen Inp_PermOK input
- PPERM Sts_NBPermOK output and the raP_Opr_EPGen.Inp_NBPermOK input



For more information on the PPERM instruction, see Rockwell Automation Library of Process Objects, publication [PROCES-RM203](#).

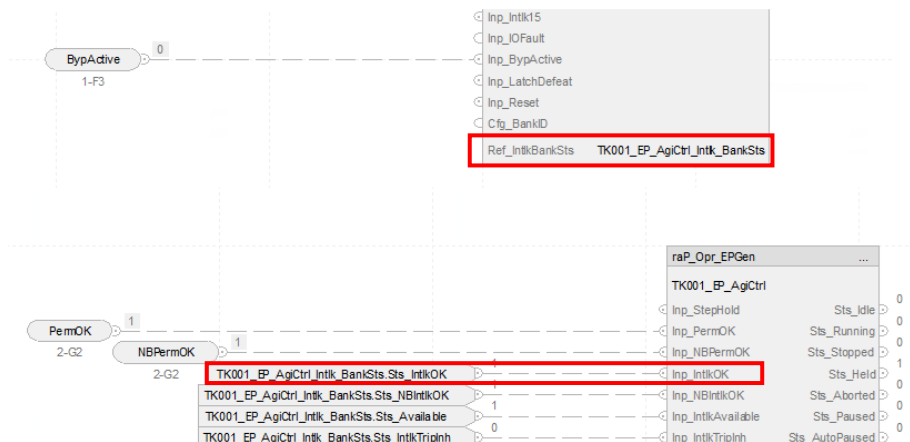
If no bypass-enabled permissive condition is triggered, the PPERM Sts_PermOK is set to 1, and raP_Opr_EPGen Inp_PermOK is also set to 1. A value of 1 indicates that you are able to start the EP in operator mode. During a permissive condition trigger, you cannot start the EP.

Permissive conditions function the same regardless of the command source – the Operator, Program, and External EP start commands are prohibited until the permissives return to the OK state.



Inp_IntlkOK and Inp_NBIntlkOK

The Inp_IntlkOK and Inp_NBIntlkOK parameters monitor the status of the EP's associated interlock blocks. To monitor the status of the interlock blocks properly, create and associate a bank status tag with the Ref_IntlkBankSts pin for each PINTLK block of the EP. Then associate the raP_Opr_EPGen Inp_IntlkOK with the Sts_IntlkOK of the bank status tag that was created.



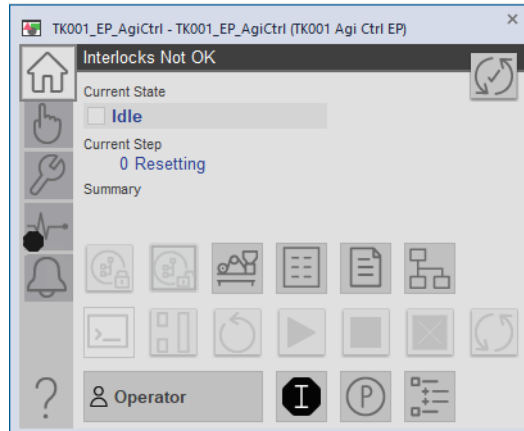
For more information on the PINTLK instruction, see Rockwell Automation Library of Process Objects, publication [PROCES-RM203](#).

If no bypassable interlock conditions are triggered and the PINTLK bank Sts_IntlkOK is set to 1, set the raP_Opr_EPGen Inp_IntlkOK to 1.

An interlock trip affects the EP state without the need of external logic. You can program the Phase Action to one of the following: Hold, Stop, or No Action.

The Cfg_ShedOnIntlk and Cfg_IntlkIssuesHold bitwise parameters are used to configure which Phase Action is taken. Similar to other PlantPax® objects, the Sts_NotRdy and Sts_IntlkTrip status parameters can be used with additional user logic to achieve the results you need for your environment.

The EP interlock trip depends on how you have configured the EP state and the EP. When the EP is Idle, an interlock trip will not allow operator commands. When Cfg_ShedOnIntlk and Cfg_IntlkIssuesHold are set to 0, the EP ignores interlock trips. In this scenario, the EP remains in the same state, which is typically the running state.

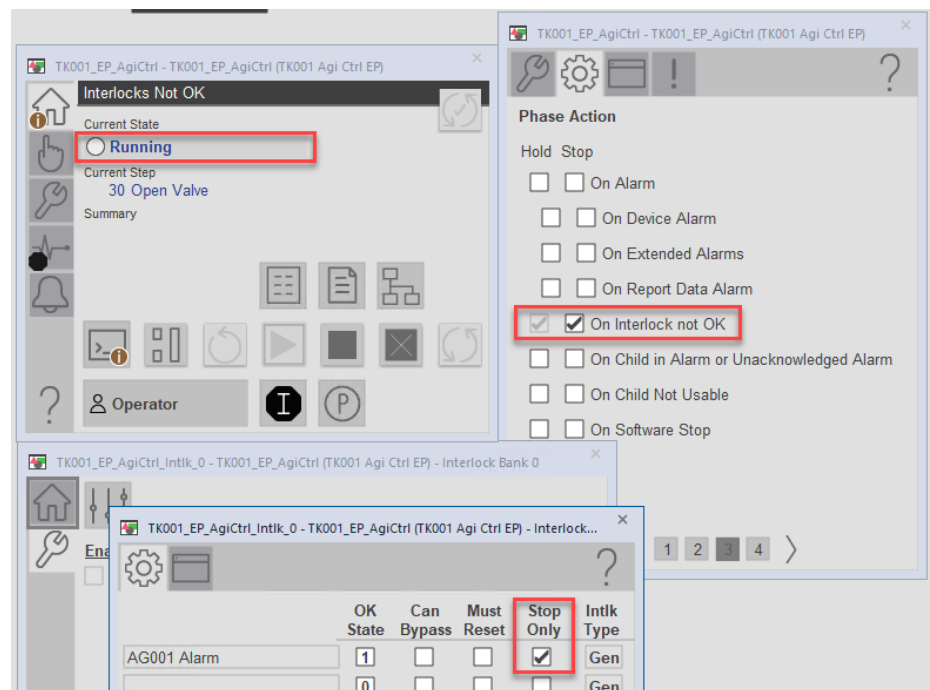


Cfg_ShedOnIntlk takes precedence over Cfg_IntlkIssuesHold and Stop takes precedence over Hold.

The Hold option is not available if Stop on Interlocks is configured.



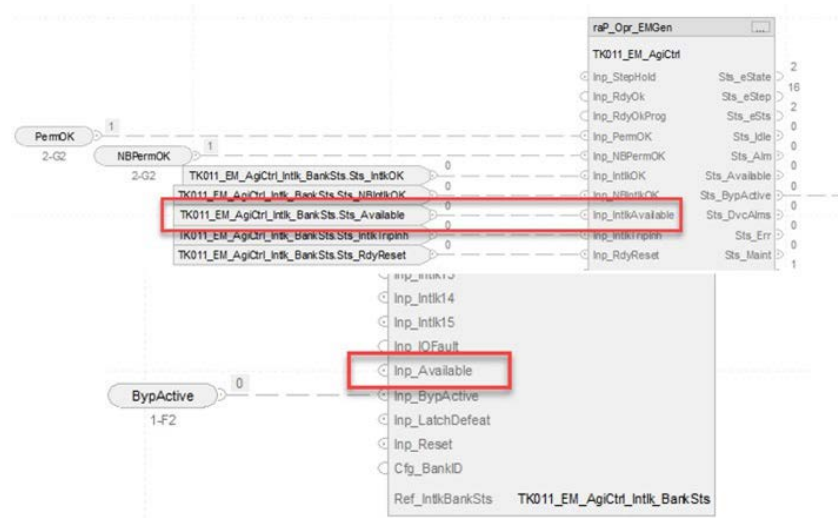
If the Interlock instruction condition is configured as 'Stop Only', the EP ignores the interlock trip, even if the EP Phase Action is configured to Stop 'On Interlock not OK.' In this case, the EP continues to run because the trip is masked. This is not a typical configuration combination.



Per the ISA-88 state model, interlock trips do not affect the ability of the EP to go into the Stopping/Stopped, Resetting, or Aborting/Aborted state. If an interlock is tripped in the Stopped or Aborted state, the EP remains in that same state. If an interlock is tripped in the Stopping, Resetting, or Aborting state, the EP goes into the Stopped, Idle, or Aborted state, respectively.

Inp_IntlkAvailable

The Inp_IntlkAvailable parameter monitors whether the EP is available for an interlock trip. To monitor the status of the interlock blocks, create a bank status tag and associate it with the Ref_IntlkBankSts pin, for each PINTLK block of the EP. Then associate the raP_Opr_EPGen Inp_IntlkAvailable parameters with the Sts_Available parameter of the bank status tag that was created.

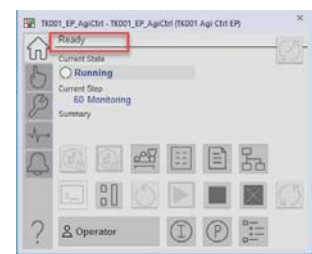


The Sts_Available parameter value is determined by a combination of the Inp_Available parameter's value and the configuration type of each of the interlocks block's input conditions. When Inp_Available=1 and the interlock block has an input condition that is configured as protective, device, mechanical, or process interlock type and that input condition is not in the OK state, the Sts_Available is set to 1 and the Sts_IntlOK/ Sts_NBIntlOK is set to 0. When Inp_Available=1 and the interlock block has an input condition that is configured as safety, electrical, operational, or general interlock type and that input condition not in the OK state, the Sts_Available and the Sts_IntlOK/ Sts_NBIntlOK are set to 0.

The Sts_IntlkAvailable output indicates that the equipment is ready to operate and all safety, electrical, operational, or general type interlocks that affect availability are working properly.

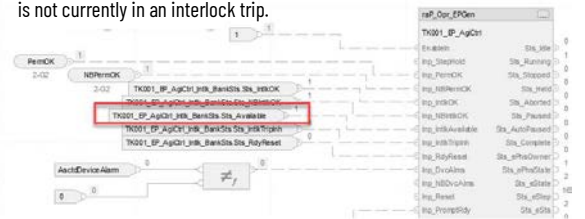


During an interlock trip condition, the EP.Sts_IntlkAvailable is set to 0 regardless of the configuration of the Stop Phase Cfg_ShedOnIntlk parameter. As discussed in [Inp_IntlkOK](#) and [Inp_NBIntlkOK](#) section, the Cfg_ShedOnIntlk parameter affects the EP Sts_IntlkTrip and Sts_NotRdy parameters, but not the Sts_IntlkAvailable parameter.

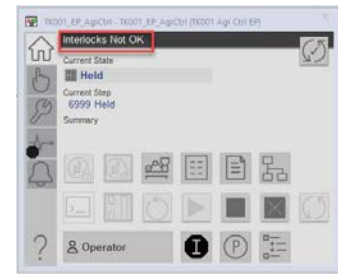


The screenshot shows the 'Ready' status of the EP. The 'Current State' is 'Ready', and the 'Current Step' is '60 Monitoring'. The 'Summary' tab is selected, showing various status icons.

Inp_IntlkAvailable is true, because EP is not currently in an interlock trip.




The ladder logic diagram shows the logic for Inp_IntlkAvailable. It includes inputs like PermOK, NDPermOK, and various status signals. The output is Inp_IntlkAvailable, which is set to 1 when the EP is not in an interlock trip.



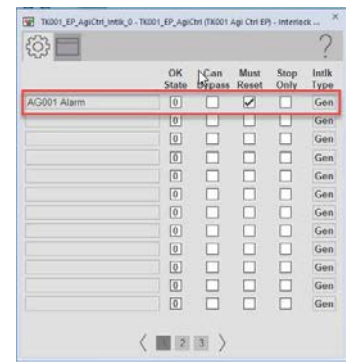
The screenshot shows the 'Interlocks Not OK' status of the EP. The 'Current State' is 'Held', and the 'Current Step' is '60999 Held'. The 'Summary' tab is selected, showing various status icons.

Inp_IntlkAvailable is false, because EP is currently in an interlock trip.




The ladder logic diagram shows the logic for Inp_IntlkAvailable. It includes inputs like PermOK, NDPermOK, and various status signals. The output is Inp_IntlkAvailable, which is set to 0 when the EP is in an interlock trip.

If Bypass Enabled is selected on the EP, and Enabled Bypass is selected on the PINTLK on a bypassable interlock, the Bank Sts_Available interlock retains a value of 1. If an interlock is configured to require a reset — as set by Cfg_Latched in the PINTLK block — the Interlock Bank Sts_IntlkAvailable parameter retains a value of 0. The Interlock Bank Sts_IntlkAvailable is reset to 0 when the EP is reset from the interlock trip.



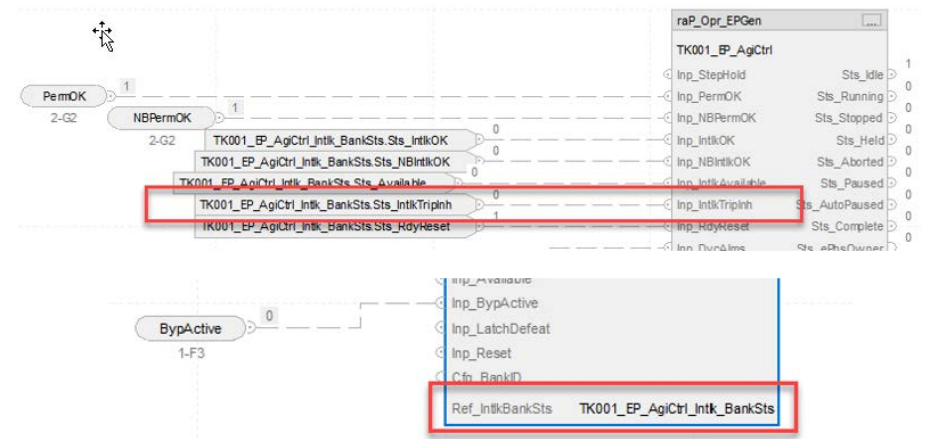
The screenshot shows the 'Enabled' status of the EP. The 'Current State' is 'Enabled', and the 'Current Step' is '60999 Held'. The 'Summary' tab is selected, showing various status icons.



The screenshot shows the 'Gen AG001 Alarm' status of the EP. The 'Current State' is 'Gen', and the 'Current Step' is '60999 Held'. The 'Summary' tab is selected, showing various status icons.

Inp_IntlkTriplnh

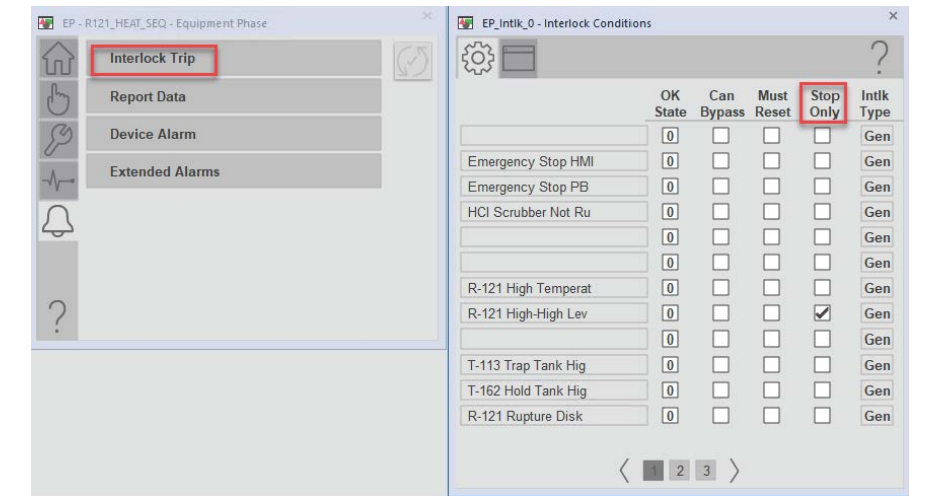
The Inp_IntlkTriplnh EP parameter monitors the system for any interlock trip alarms that are inhibited. To monitor this tag properly, create a bank status tag and connect it to each PINTLK block of the EP and associate it with the Ref_IntlkBankSts pin. Then tie the raP_Opr_EPGen Inp_IntlkTriplnh parameter to the bank status tag Sts_IntlkTriplnh parameter.



To inhibit the EP Interlock Trip Alarm, use the Cfg_StopOnly parameter of the PINTLK block to determine which interlock conditions need to be configured as 'Stop Only.' An interlock Cfg_StopOnly value of 1 indicates that when the interlock condition trips, the EP Interlock Trip will not go into alarm.

To create a Trip Alarm notification on the EP faceplate and the HMI Alarm Summary/Banner, go to the controller Alarm Manager and set the EP Interlock Trip Alarm to 'Use.'

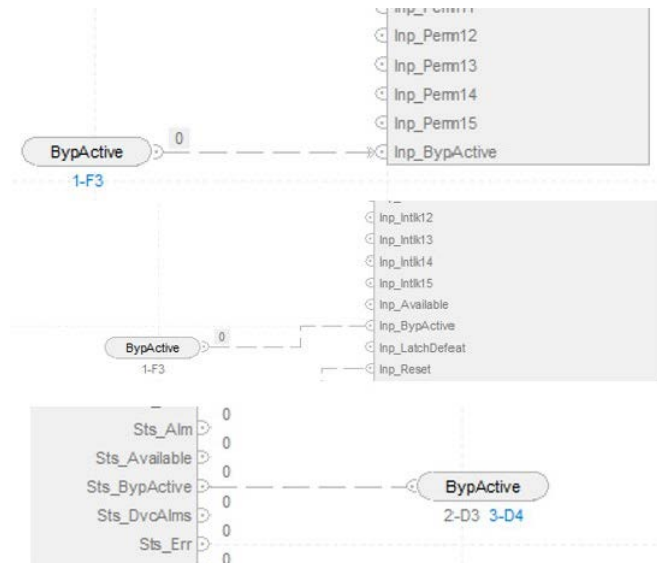
State	Use	Owner	Name	Type	Input	Expression
	<input type="checkbox"/>	\Template_EP.EP	Alm_DvcAlms	TRIP	\Template_EP.EP.Sts_DvcAlms	= 1
	<input checked="" type="checkbox"/>	\Template_EP.EP	Alm_IntlkTrip	TRIP	\Template_EP.EP.Sts_IntlkTrip	= 1
	<input type="checkbox"/>	\Template_EP.EP	Alm_RptData	TRIP	\Template_EP.EP.Sts_RptData	= 1
	<input checked="" type="checkbox"/>	\Template_EP.ExtddAlm_00	Alm_Alarm	TRIP	\Template_EP.ExtddAlm_00.Sts	= 1



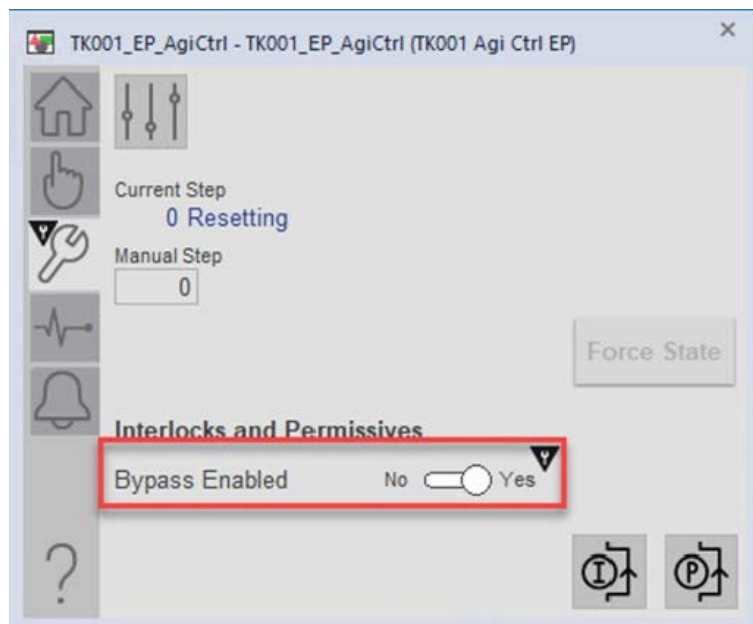
Sts_BypActive

The Sts_BypActive parameter monitors whether bypass is enabled on the EP. Some interlocks/permissions for the EP are considered bypassable, which means that an engineer or maintenance can ignore the conditions and proceed with the phase as normal. Some conditions are considered non-bypassable, which means that the only way to resume normal phase conditions is to first remedy the tripped interlock/permission.

If you need to bypass interlocks and permissions, tie a wire connector to each instance of a PINTLK or PPERM block to the Inp_BypActive pin. Tie the wire connector that you assign to the Sts_BypActive pin of the EP block.



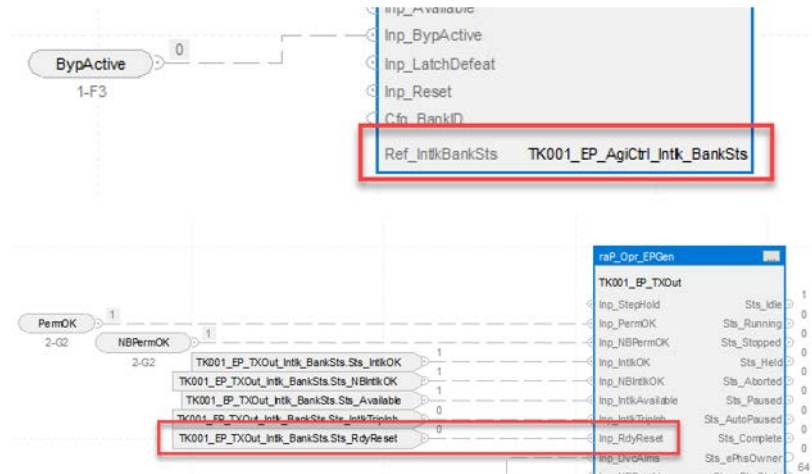
You can enable a bypass via the HMI faceplate.



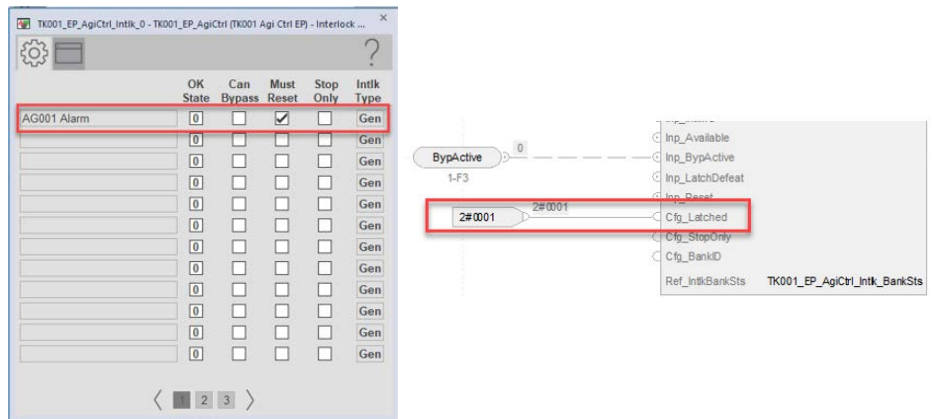
Inp_RdyReset

The Inp_RdyReset EP parameter monitors whether a latched interlock is ready to be reset. To monitor a reset- requested status, create and associate a bank status tag with the Ref_IntlkBankSts pin for each PINTLK block of the EP. Then associate the raP_Opr_EPGen Inp_RdyReset parameter with the Sts_RdyReset parameter of the bank status tag. This functionality is consistent with the other PlantPAx Interlock control strategies.

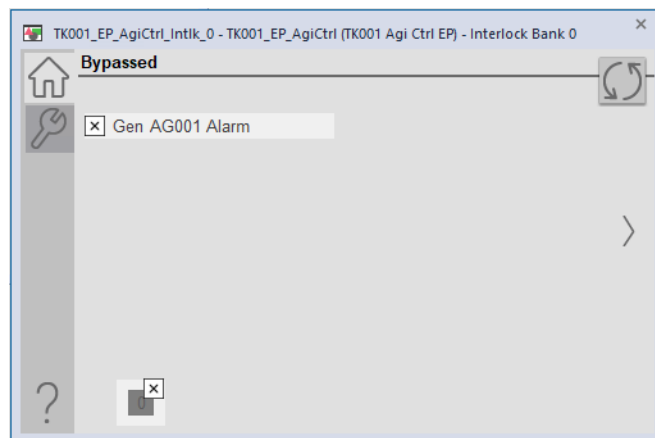
See PlantPAx Process Control Instructions User Manual, publication [PROCES-RM215](#) for more information.



Interlocks can be configured individually to require a reset by modifying the PINTLK Cfg_Latched parameter. For example, if you set a Cfg_Latched value to 1, an interlock trip remains active until the interlock trigger condition returns to normal and you reset the interlock.



If an interlock condition is configured as 'reset required,' the `Inp_RdyReset` remains set to 0, as long as the interlock condition remains tripped. When the interlock condition clears, the `Inp_RdyReset` value changes to 1 and an HMI reset becomes available.



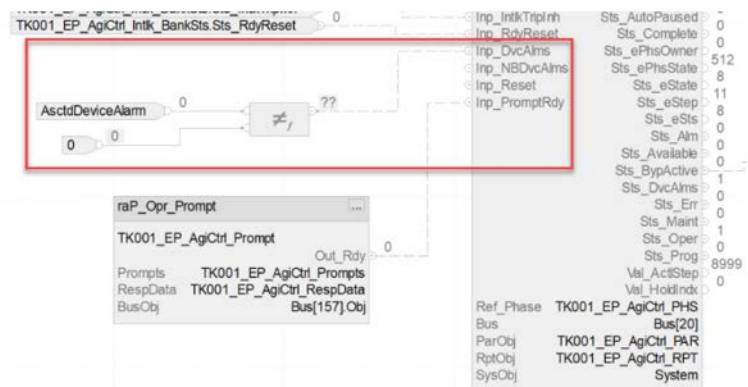
The interlock condition is now false, but the EP is still in an interlock trip until a reset occurs. Inp_RdyReset is set to 1 and therefore the reset is available.

When a reset has been issued, the Inp_RdyReset parameter is reset to 0 and the Inp_IntlkOK and/or Inp_NBIntlkOK parameters are set to 1.

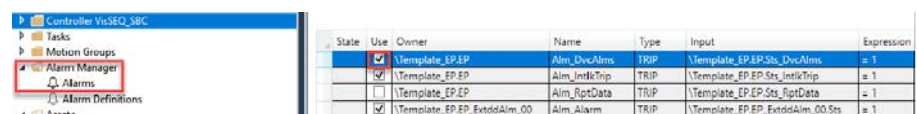
Inp_DvcAlms

When the Bus organizer is not used, you can use the `Inp_DvcAlms` parameter to monitor the alarm state of devices that are associated with the EP – including children device alarms connected to the EP.

Connect each object to one parameter of an associated alarms tag — listed as 'AsctdDeviceAlarm' in the image below — and pin the objects the Inp_DvcAlrms input as follows.

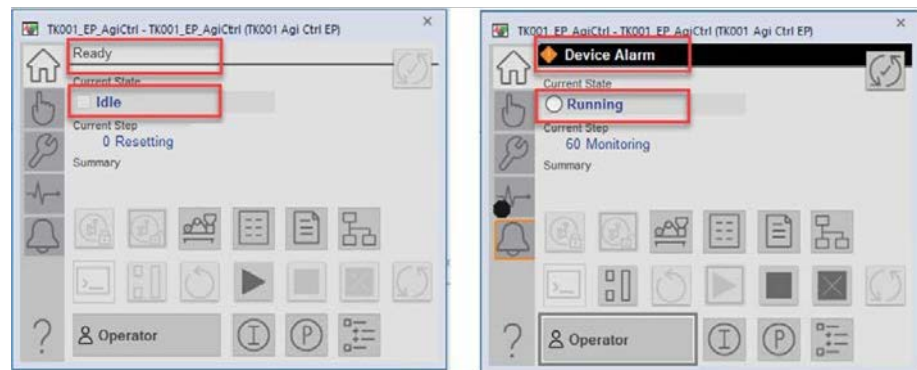


A nonzero indicates that one or more objects that are associated with the EP are in alarm. In the controller Alarm Manager, the EP Device Alarm must be set to 'Use,' which then displays the Device Alarm on the EP faceplate and makes the alarm active.



The `Inp_DvcAlms` parameter of the EP is reflected in the `Sts_DvcAlms` parameter, even when the Alarm Manager alarm is set to 'Use.'

To trigger a device alarm in the associated EP, first verify that the EP is not in the Idle state. The Idle state acts as a gate for the EP Device Alarm, while the EP status parameter `Sts_DvcAlms` remains set to 0.



Shows that the EP is in Idle State 0 during a device alarm

Shows that the EP is in a non-Idle State during a device alarm

IMPORTANT The `Inp_DvcAlms` must only be used for device alarms that you want to bypass. When the EP bypass is enabled, the EP `Sts_DvcAlms` is not triggered and remains set to 0.

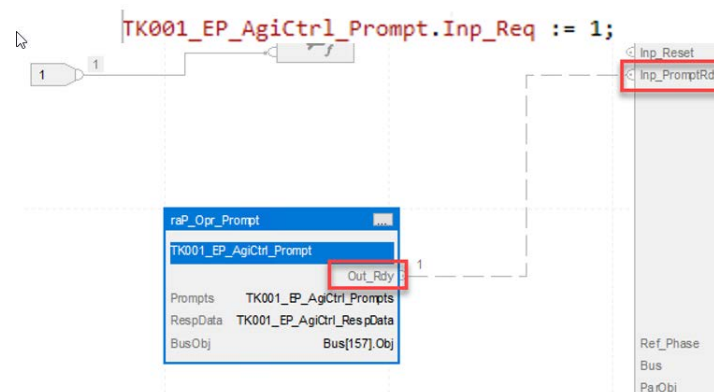
Extended Alarms: Can be used to display and track unique alarm conditions relative to the EM or EP instance. See [Additional Configuration](#) for more information.

Inp_PromptRdy

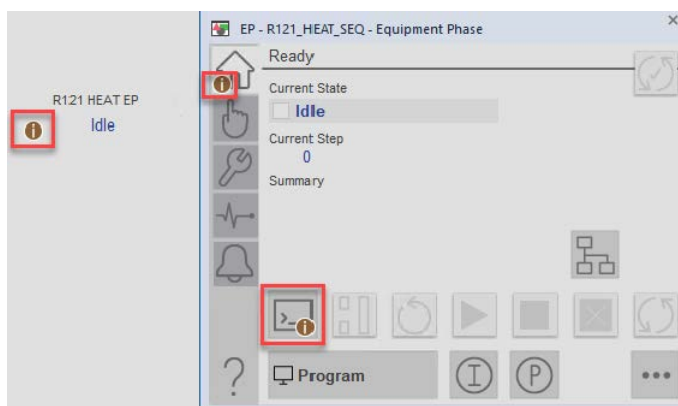
The `Inp_PromptRdy` parameter indicates that the associated `raP_Opr_Prompt` block is ready to send a prompt request to the operator.

To enable the prompt object to appear on the `raP_Opr_Prompt` block, set the `Inp_Req` parameter on the `raP_Opr_Prompt` block that is associated with the EP to 1. If you want to enable operator prompts, use logic to set the `Inp_Req` to 1 and the `Inp_Ref` to the desired Prompt array value. Program the logic so that when you acknowledge the prompt, `Inp_Req` resets to 0.

To cause the EP display and notify the operator prompt in the HMI faceplate, pin the prompt block `Out_Rdy` to the EP `Inp_PromptRdy` parameter.



When the EP Inp_PromptRdy input is set to 1, an informational breadcrumb image is visible on the EP global object and on the EP Home tab faceplate. In this scenario, no additional programming is required. The Prompt button is available and used for the operator to view and acknowledge the prompt.



For more information on the PROMPT instruction, see Rockwell Automation Library of Process Objects, publication [PROCES-RM203](#).

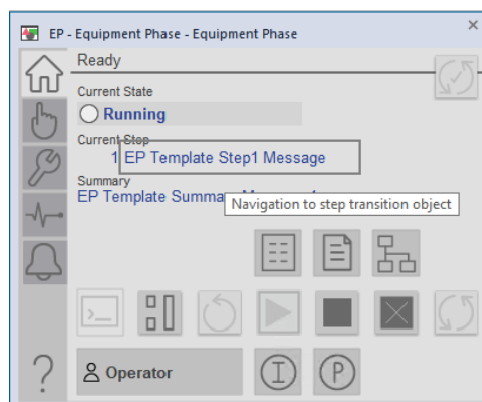
Inp_EnableNavTrans

The Inp_EnableNavTrans parameter is used to enable visibility navigation for step transitions, via additional programming. When enabled and set to =1, a touch point with a tooltip is displayed on the EP faceplate. When you use this feature, system operators are able to troubleshoot what conditions are preventing step transitions.

Two common tag types are: PPERM (Permissive) and PBL (Boolean logic). Both of these tags provide instructions for gaining insight into transition conditions. You can use any PlantPAx object because the touchpoint uses the NavToDisplay macro. However, the PPERM and the PBL are typically the most flexible. With a PINTLK block, multiple interlocks can trip simultaneously, so knowing which interlock tripped first is not useful for gaining insight into transition conditions.

Name the transition tag syntax the same as the EP tag. Concatenated the name with '_Tran_' and then add the step number. For example, in step 20, EP_Tran_20 is the transition tag for the local EP tag. This tag must have the same path location as the EP tag, which is typically a programmed scoped tag.

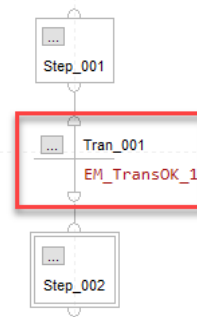
EP HMI faceplate: When the Inp_EnableNavTrans input parameter is true (1), the Current Step description has touch navigation that lets you open the step transition object faceplate.



Transition logic: Consider leveraging the Sts_BypActive to let certain conditions be bypassed from the faceplate. You can customize the Boolean tag name for the SFC transition – for example, 'EP_TransOk_1.'



State Step Sequence logic: Use a local Boolean tag on the output of the PPERM instruction for the actual SFC Transition Boolean logic. When the tag is true, the SFC will move to the next step.



MSet_Step (Manual Step)

To force the EP to a specific State Step, perform the following steps.

1. Put the EP into Maintenance mode.



Operators and supervisors do not have this security privilege.

2. From the Maintenance tab on the EP faceplate, enter a Manual Step number.



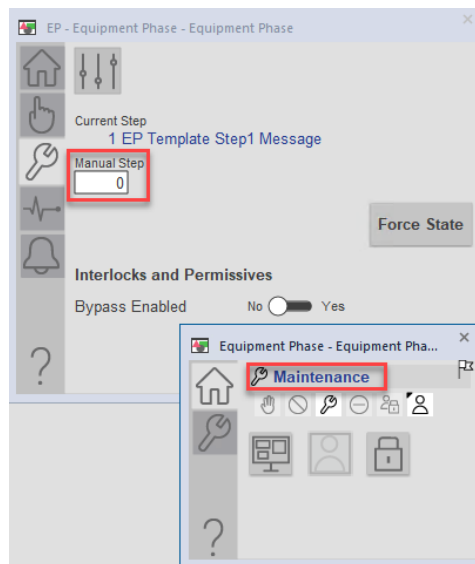
This numeric Manual Step entry writes directly to the EP MSet_Step parameter.

3. The EP instruction processes the command internally and sets the EP Val_ActStep.

IMPORTANT

Make sure that the entry value is a valid and configured step, especially when using SFC step logic.

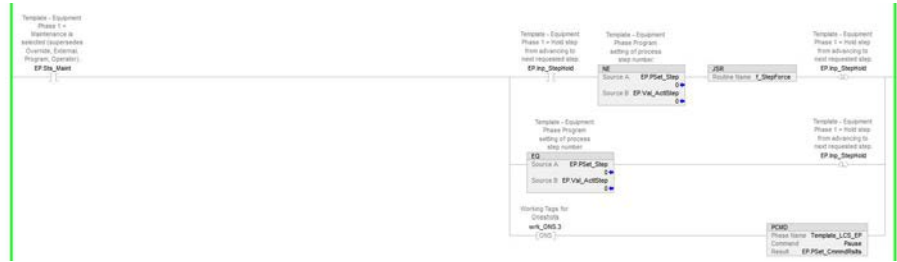
4. When you press Enter on the keyboard, the Manual Entry value immediately resets to 0.



Monitor the EP Val_ActStep or EP PSet_Step parameter value when the EP is in Maintenance Mode to determine the associated step logic that is needed to execute for manual step jumps.

You can also use EP Inp_StepHold to verify that the MSet_Step value is valid by monitoring the PSet_Step before the step request is processed. When you use Inp_StepHold in this way, you can validate or ignore the step value by monitoring the PSet_Step value. You can choose if you want to make step changes within the same state only. The PSet_Step is available in any mode and does not require the EP to be in Program mode.

The following contains an example additional developer code that is used to verify MSet_Step and PSet_Step (optional).



```

1  STEP FORCE: 07/2024 Rockwell Automation
2  Used for MSet_Step and PSet_Step force entries.
3  Ensures the Step number entry is a valid step and matches it with the associated SFC Step tagname.
4  ****
5
6
7  if EP.Sts_Running then
8      case EP.PSet_Step of
9          1000: SFR(Running,E_SFCStep_1000);
10         1010: SFR(Running,E_SFCStep_1010);
11         1020: SFR(Running,E_SFCStep_1020);
12         1030: SFR(Running,E_SFCStep_1030);
13         1040: SFR(Running,E_SFCStep_1040);
14         1050: SFR(Running,E_SFCStep_1050);
15         1060: SFR(Running,E_SFCStep_1060);
16         1070: SFR(Running,E_SFCStep_1070);
17         1080: SFR(Running,E_SFCStep_1080);
18         1090: SFR(Running,E_SFCStep_1090);
19         1100: SFR(Running,E_SFCStep_1100);
20         1110: SFR(Running,E_SFCStep_1110);
21     else
22         EP.PSet_Step := EP.Val_ActlStep;
23     end_case;
24
25  elsif EP.Sts_Holding then
26      case EP.PSet_Step of
27          6000: SFR(Holding,E_SFCStep_6000);
28          6010: SFR(Holding,E_SFCStep_6010);
29          6020: SFR(Holding,E_SFCStep_6020);
30      else
31          EP.PSet_Step := EP.Val_ActlStep;
32      end_case;
33
34  elsif EP.Sts_Resetting then
35      case EP.PSet_Step of
36          5000: SFR(Resetting,E_SFCStep_5000);
37      else
38          EP.PSet_Step := EP.Val_ActlStep;
39      end_case;
40  end_if;

```

MCmd_StateForce (Force State Complete)

To program the EP to force the current state to State Complete, perform the following steps.

1. Put the EP into Maintenance mode.



Operators and supervisors do not have this security privilege.

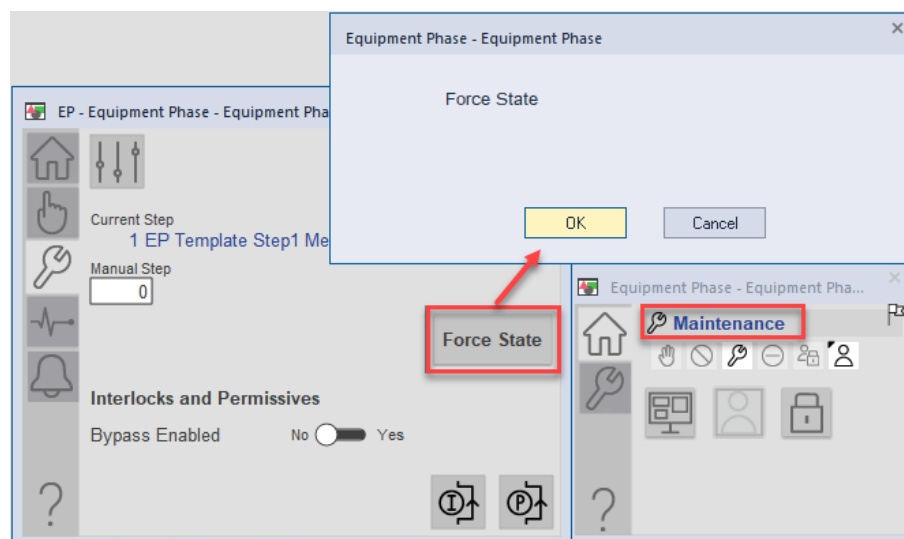
2. From the Maintenance tab on the EP faceplate, select the Force State button.

The Force State button writes to the EP MCmd_StateForce parameter directly and the EP sets the Sts_StateCompleteRqst internally before self-resetting.

3. Use the SFR command to connect the Sts_StateCompleteRqst to the required location in the State Model logic.
4. Set the desired state/step for SFC logic.

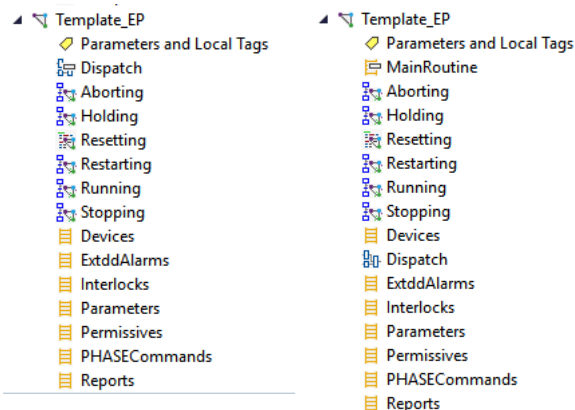


The EP has a PCmd_StateForce that can be used to set the Sts_StateCompleteRqst, via programming.



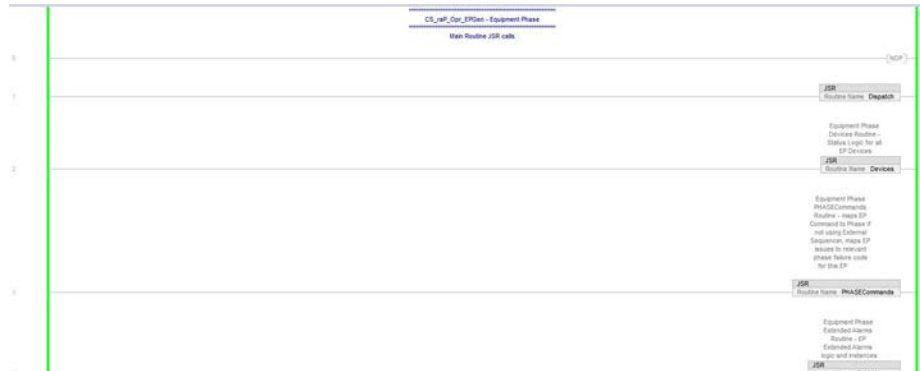
EP Control Strategy (defaults)

The following example shows a ControlLogix® Template EP Program that was created using Application Code Manager (ACM). The number and type of routines vary depending on the ACM build configuration. The following example shows two templates — one with a MainRoutine and one without a MainRoutine.

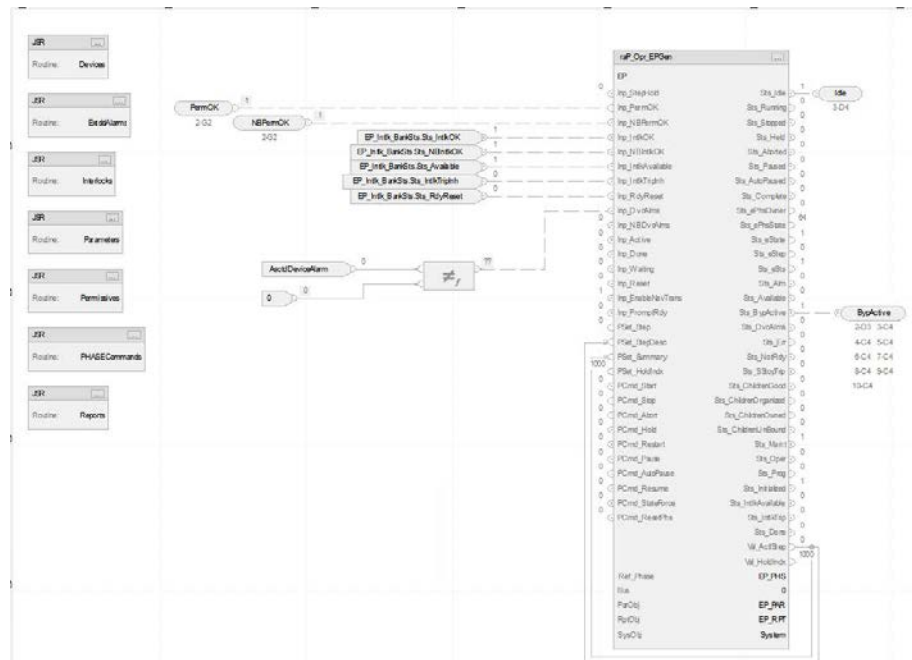


EP Routines (defaults)

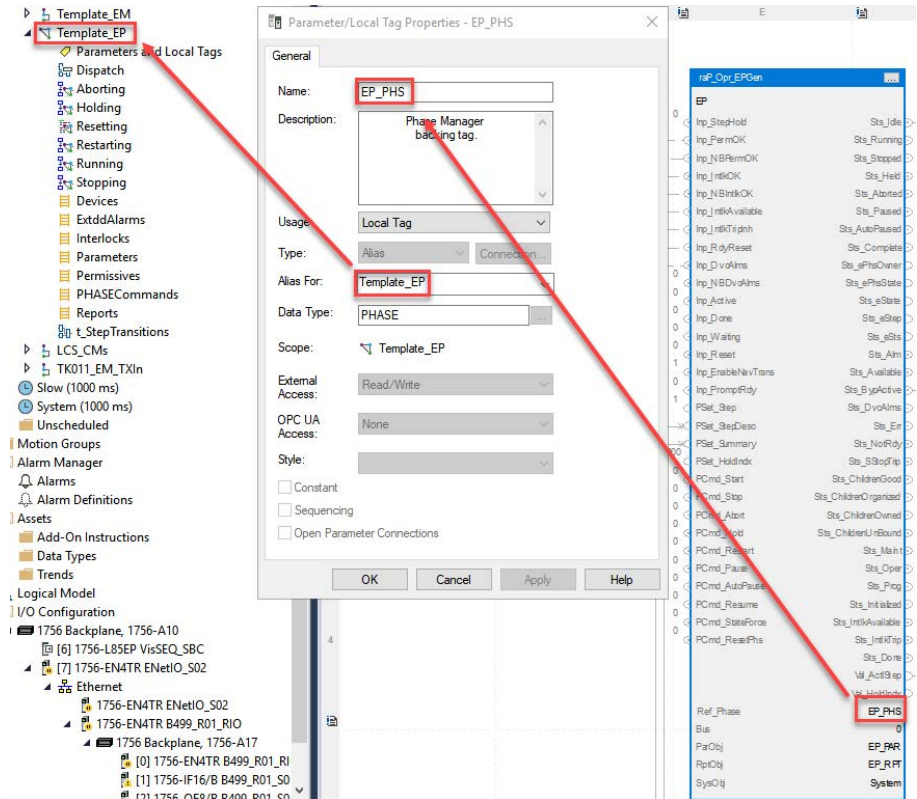
MainRoutine: Contains the Jump to Subroutine JSR instructions. When you use this routine, you do not need to use additional/similar JSR calls in the Dispatch Routine. The MainRoutine also allows for routine order of execution.



Dispatch: Contains the raP_Op_EPGen instruction, the Interlock and Permissives instructions (additional sheets), and the calls to the other routines (non-MainRoutine cases). If you use a MainRoutine, you do not need the JSR instructions on the left of the sheet.

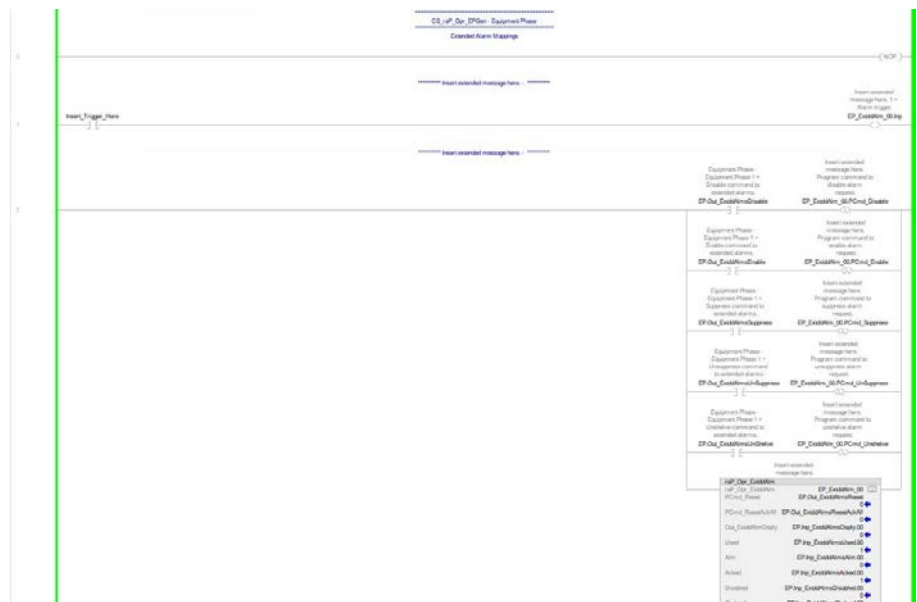


Ref_Phase: The reference phase tag must be defined as an alias to the Equipment Phase name.



Devices (optional): Contains the Control Module device mapping. You can use parameter connections or a direct tag reference in the state/step logic.

ExtddAlarms (optional): Extended Alarm. Replace the placeholder tag 'Insert_Trigger_Here' with the alarm condition trigger.

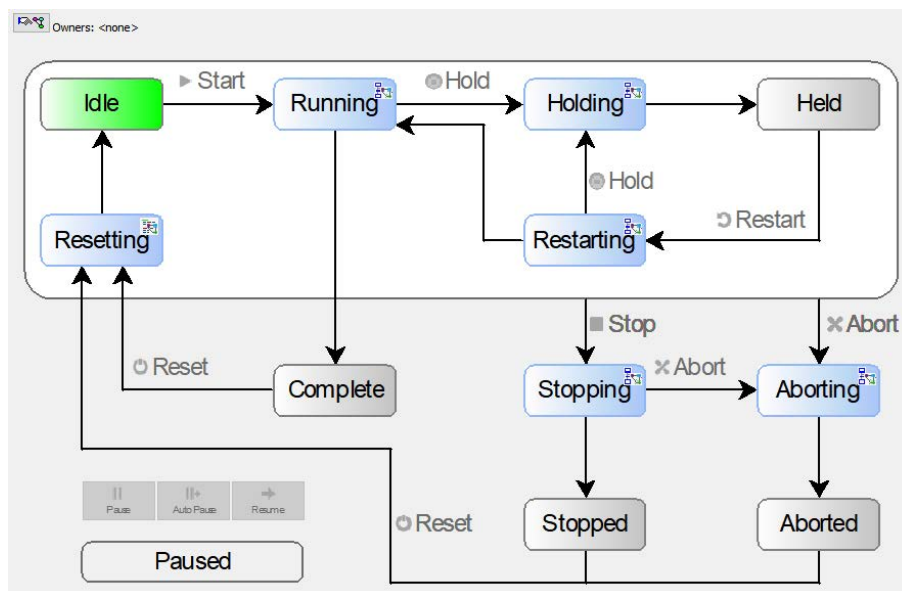


Interlocks and Permissives (optional): Interlock/Permissive conditions can be mapped in this ladder logic routine. For complex interlock/permissive conditions, use ladder logic. In less complex scenarios, you can pin the expressions directly to the interlock/permissive input pins in function block.

Parameters (optional): PlantPax has four different types of parameters that use the raP_Tec_ParRpt instruction, including: Enumeration, Integer, Real, and String parameter types.

Use the default local parameter tagname 'EP' for the EPGen. The ParObj parameter must be set as the local parameter tag named 'EP_PAR.'



S88 State Routings: Aborting, Holding, Resetting, Restarting, Running, Stopping

State step logic: At the time of ACM configuration, you can choose to use Sequential Function Chart (SFC) or Ladder logic. Each State has a dedicated routine. Rockwell Automation recommends that you standardize local tag names for the SFC Steps, Actions, and Transitions.

When a transition is true, the logic progresses to the next step and the code from the corresponding action executes. Rockwell Automation recommends that you increase the EM step index incrementally for each new step in the SFC. To do so, increase the value of the Pset_Step Parameter.

IMPORTANT When you restart the system, you must also set the Holding Index.

PSet_StepDesc is a message index and is used on the EP faceplate references to a local message. See the next section for more information on how the local message is used.

In the following example, the level transmitter (LIT001) Ctrl Hi Limit is set to the initial value of an EP parameter. Valve XV011 is commanded open. When you receive confirmation that valve XV011 is open, pump PP011 is commanded to start.

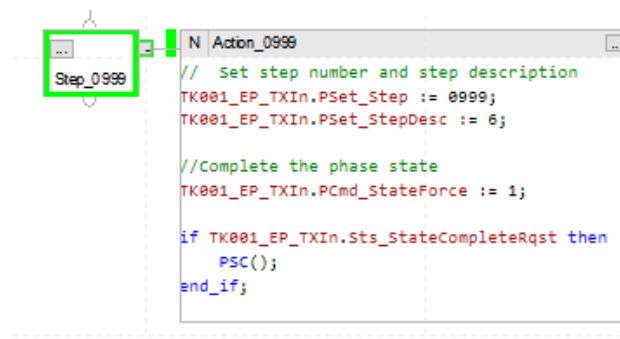


State Complete Command (optional): The following example shows the last step in an SFC state routine. Since PhaseManager™ is used as an outside wrapper, the Phase Complete command must be issued to complete the state.

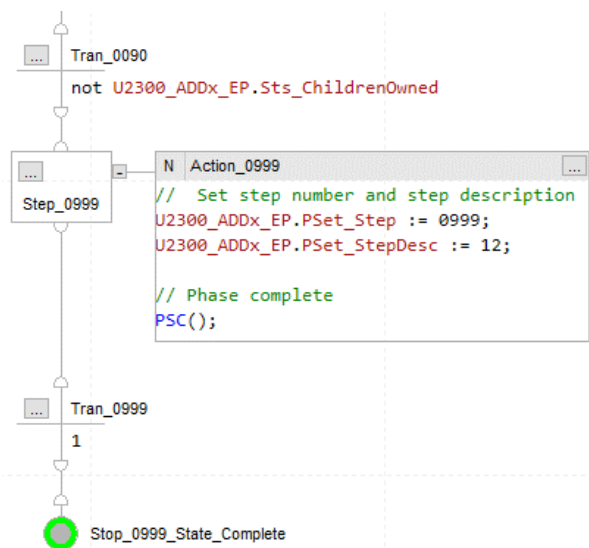


The optional SFC_Stop instruction has no functional impact and is not required.

You can choose to program the EP.PCmd_StateForce parameter to 1, which causes the EP to process this command and sets the EP.Sts_StateCompleteRqst. To confirm this status parameter and issue the Phase Complete command, use the step action.

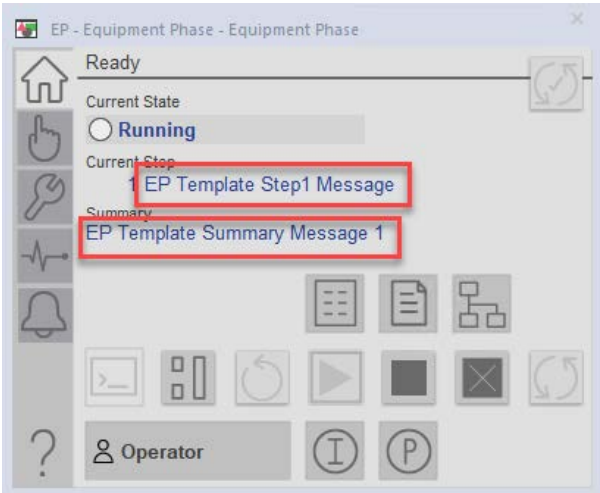


The PCmd_StateForce is not required. As an alternative, you can determine to issue the PSC() command directly in the last step action.

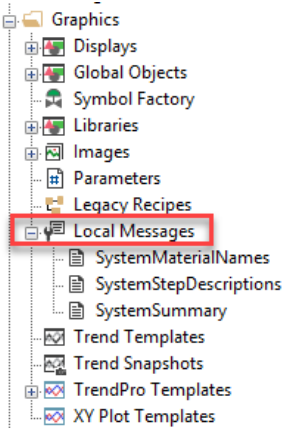


Step Descriptions: HMI local message displays are used to display additional EP Current Step information on the EP faceplate. You can use the EP.PSet_StepDesc to display a custom message. The program PSet command functions regardless of the mode, which means it can be used even if the EP is in operator or maintenance mode. It is also independent of the state or step values.

Summary Descriptions: Local message displays are used to display additional EP Summary information on the EP faceplate. You can use the EP.PSet_Summary to display a custom message. Similar to the Step Description, EP.PSet_Summary functions even if the EP is in operator or maintenance mode.



HMI Local Messages (legacy method): The default SystemStepDescriptions local message is a legacy method that provides backwards compatibility. When you use this function, the faceplate references the local message object. The controller shortcut path for this function is Global parameter #3.



SystemStepDescriptions - /LCS_SAMPLE/HMI (Local Messages)		
	Trigger Value	Message
1	1	/S:0 {#3System.Enum.Step_Desc[1].@Description}*/
2	2	/S:0 {#3System.Enum.Step_Desc[2].@Description}*/
3	3	/S:0 {#3System.Enum.Step_Desc[3].@Description}*/
4	4	/S:0 {#3System.Enum.Step_Desc[4].@Description}*/
5	5	/S:0 {#3System.Enum.Step_Desc[5].@Description}*/
6	6	/S:0 {#3System.Enum.Step_Desc[6].@Description}*/
7	7	/S:0 {#3System.Enum.Step_Desc[7].@Description}*/
8	8	/S:0 {#3System.Enum.Step_Desc[8].@Description}*/
9	9	/S:0 {#3System.Enum.Step_Desc[9].@Description}*/
10	10	/S:0 {#3System.Enum.Step_Desc[10].@Description}*/
11	11	/S:0 {#3System.Enum.Step_Desc[11].@Description}*/
12	12	/S:0 {#3System.Enum.Step_Desc[12].@Description}*/
13	13	/S:0 {#3System.Enum.Step_Desc[13].@Description}*/
14	14	/S:0 {#3System.Enum.Step_Desc[14].@Description}*/
15	15	/S:0 {#3System.Enum.Step_Desc[15].@Description}*/
16	16	/S:0 {#3System.Enum.Step_Desc[16].@Description}*/
17	17	/S:0 {#3System.Enum.Step_Desc[17].@Description}*/
18	18	/S:0 {#3System.Enum.Step_Desc[18].@Description}*/
19	19	/S:0 {#3System.Enum.Step_Desc[19].@Description}*/
20	20	/S:0 {#3System.Enum.Step_Desc[20].@Description}*/
21	21	/S:0 {#3System.Enum.Step_Desc[21].@Description}*/
22	22	/S:0 {#3System.Enum.Step_Desc[22].@Description}*/
23	23	/S:0 {#3System.Enum.Step_Desc[23].@Description}*/
24	24	/S:0 {#3System.Enum.Step_Desc[24].@Description}*/
25	25	/S:0 {#3System.Enum.Step_Desc[25].@Description}*/
26	26	/S:0 {#3System.Enum.Step_Desc[26].@Description}*/
27	27	/S:0 {#3System.Enum.Step_Desc[27].@Description}*/
28	28	/S:0 {#3System.Enum.Step_Desc[28].@Description}*/
29	29	/S:0 {#3System.Enum.Step_Desc[29].@Description}*/

The System tag is controller-scoped and common for the EPs in the controller.

Scope: [VaSEQ_SBC] Show: All Tags [T_system]

Name	Force Mask	Style	Data Type	Description
System.Enum.Step_Desc		(...)	Decimal	System Global Structure Step Descriptions
System.Enum.Step_Desc[0]			Decimal	Step Description 0
System.Enum.Step_Desc[1]			Decimal	Step Description 1
System.Enum.Step_Desc[2]			Decimal	Step Description 2
System.Enum.Step_Desc[3]			Decimal	System Global Structure Step Descriptions
System.Enum.Step_Desc[4]			Decimal	System Global Structure Step Descriptions
System.Enum.Step_Desc[5]			Decimal	System Global Structure Step Descriptions
System.Enum.Step_Desc[6]			Decimal	System Global Structure Step Descriptions

HMI Local Messages (most current method): This is the preferred method as it provides the most flexibility and can be used to create class-based local message files for each class/type.

As part of the EP tag structure, you can use the Sts_eSummary and Sts_eStep extended tag properties for the Navigation field to define which local messages are used.

In the following example, the step descriptions use 'Template_EP_Steps' and the summary displays 'Template_EP_Summary.' In this scenario, no other controller tags or fields are required.

EP.Sts_eStep | Navigation | Template_EP_Steps

EP.Sts_eSummary | Navigation | Template_EP_Summary

You can copy and paste local messages from and into a spreadsheet or notepad application. This capability allows you to make bulk edits and/or updates to your system.

Template_EP_Steps - /LCS_SAMPLE/HMI (Local Messages)

	Trigger Value	Message
1	1	EP Template Step1 Message
2	2	EP Template Step2 Message
3	3	EP Template Step3 Message
4	4	EP Template Step4 Message

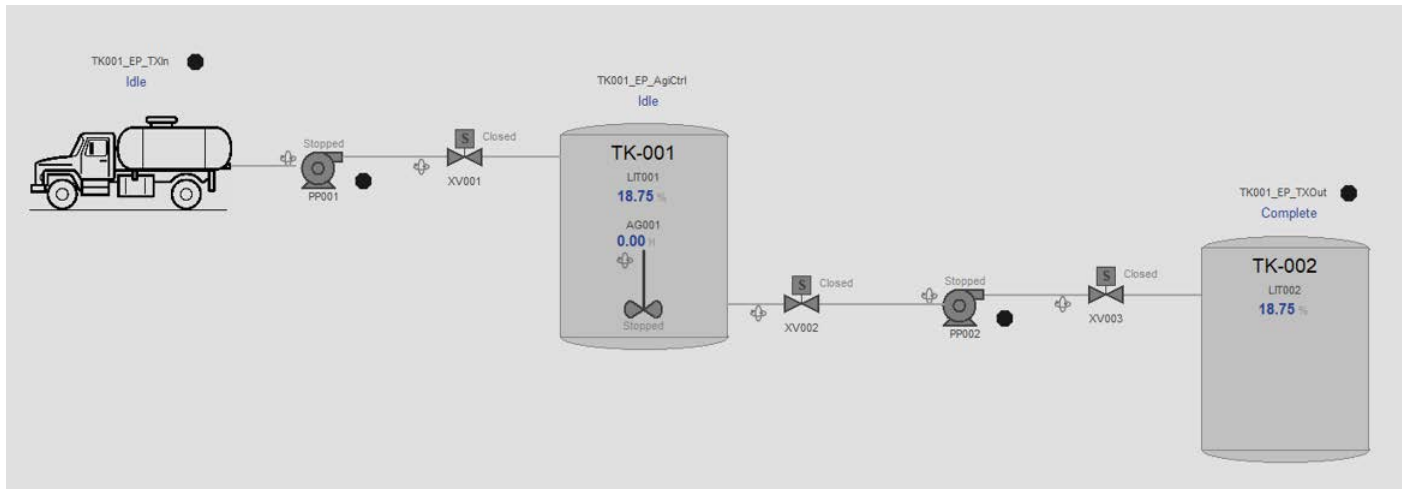
Template_EP_Summary - /LCS_SAMPLE/HMI (Local Messages)

	Trigger Value	Message
1	1	EP Template Summary Message 1
2	2	EP Template Summary Message 2
3	3	EP Template Summary Message 3
4	104	EP Template Summary Message 4

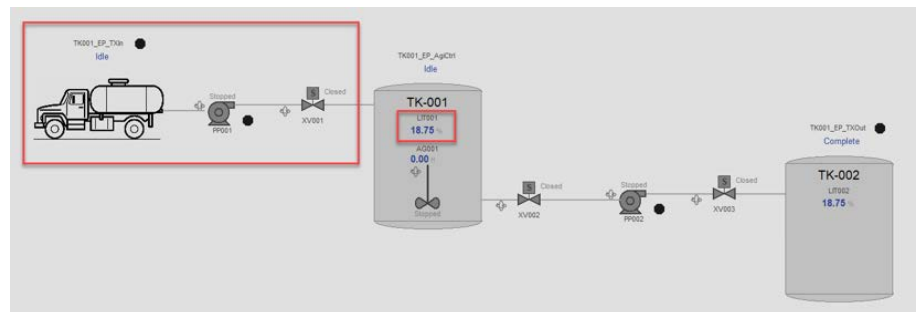
Use Case Examples

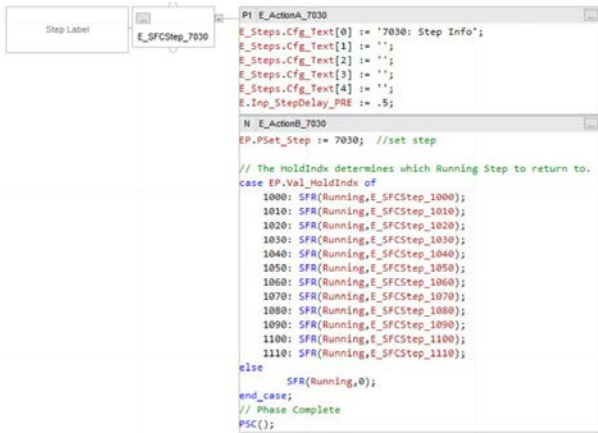
In the Equipment Phase (EP) example system, liquid is unloaded from a truck into a mixing tank, labeled TK-001. In TK-001, the liquid is agitated and then dispensed into a holding tank, labeled TK-002. The system is split into three equipment phases:

1. TK001_EP_TXIn commands the objects that dispense liquid into the mixing tank.
2. TK001_EP_AgiCtrl commands the objects that agitate and monitor the liquid level inside the tank.
3. TK001_EP_TXOut commands the objects that transfer the liquid from the mixing tank to the holding tank.



Equipment Phase One - TK001_EP_TXIn_PHS



State	Program the Logic as Follows
Running	<ol style="list-style-type: none"> 1. When ownership of the EP is acquired, the EP parameters set and capture the target fill level of tank TK-001. 2. When the fill level is set, valve XV001 is commanded to open. 3. When XV001 status is opened, pump PP001 is commanded to start. 4. When pump PP001 runs, the EP begins to monitor the tank level indicator LIT001. 5. When level indicator LIT001 indicates that the target level has been reached, pump PP001 is commanded to stop. 6. When PP001 is stopped, XV001 is commanded to close. 7. When the objects are closed, the state that the EP ends the sequence in (Complete, Aborted, Held, or Stopped) is reported to the EP. 8. The EP is released from PhaseManager ownership. 9. The running state is complete (Complete).
Aborting	<ol style="list-style-type: none"> 1. When the operator commands the EP to go to the aborting state, pump PP001 is commanded to stop. 2. Inlet valve XV001 is commanded to close via programming. Note: There is no status check for these two actions before transitioning steps. 3. When the objects are commanded to close via programming, the state that the EP leaves the sequence in (Complete, Aborted, Held, or Stopped) is captured (as 'Aborted'). 4. The report is captured. 5. Ownership of the EP is commanded to be released. 6. After the EP is commanded to release, the aborting state is complete (Aborted).
Holding	<ol style="list-style-type: none"> 1. If an EP is commanded into the Holding state, the state that the EP leaves the sequence in is captured (as 'Held'). 2. The report is captured. 3. If the report is captured properly, ownership of the EP is commanded to release. Note: You can decide if you want to release ownership in Holding logic. 4. All devices are stopped and released. 5. When the EP has been released, the holding state is complete (Held).
Resetting	<ol style="list-style-type: none"> 1. If the EP is commanded into the resetting state, the other states (Complete, Aborted, and Stopped) are reset to the beginning of their sequences. 2. Ownership of the EP is released. 3. The resetting state is complete (Idle).
Restarting	<ol style="list-style-type: none"> 1. When an operator commands the EP into the restarting state, the running phase sequence is reset to the beginning of the sequence. 2. The holding phase sequence step is set to the end of the sequence. 3. The restarting state is complete (Running). 4. By using the PSet_HoldIdx and Val_HoldIdx, the Running logic can restart at the desired step. Following is an example:  <pre> P1 E_ActionA_7030 E_Steps.Cfg_Text[0] := '7030: Step Info'; E_Steps.Cfg_Text[1] := ''; E_Steps.Cfg_Text[2] := ''; E_Steps.Cfg_Text[3] := ''; E_Steps.Cfg_Text[4] := ''; E_StepDelay_PSE := .5; N E_ActionB_7030 EP.PSet_Step := 7030; //set step // The HoldIdx determines which Running Step to return to. case EP.Val_HoldIdx of 1000: SFR(Running, E_SFCStep_1000); 1010: SFR(Running, E_SFCStep_1010); 1020: SFR(Running, E_SFCStep_1020); 1030: SFR(Running, E_SFCStep_1030); 1040: SFR(Running, E_SFCStep_1040); 1050: SFR(Running, E_SFCStep_1050); 1060: SFR(Running, E_SFCStep_1060); 1070: SFR(Running, E_SFCStep_1070); 1080: SFR(Running, E_SFCStep_1080); 1090: SFR(Running, E_SFCStep_1090); 1100: SFR(Running, E_SFCStep_1100); 1110: SFR(Running, E_SFCStep_1110); else SFR(Running, 0); end_case; // Phase Complete PSC(); </pre>
Stopping	<ol style="list-style-type: none"> 1. When the operator commands the EP into the stopping state, pump PP001 is commanded to stop. 2. When pump PP001 has stopped, valve XV001 is commanded to close. 3. When valve XV001 is closed, the status that the EP leaves the sequence in (Complete, Aborted, Held, or Stopped) is captured (as 'Stopped'). 4. The report is captured. 5. When the report is captured, ownership of the EP is commanded to release. 6. When ownership is released, the stopping state is complete (Stopped).

Routine Structure - TK001_EP_TXIn_PHS

In the following example, the letter-number combination prefix on the routine name helps organize the list of routines.

Most of the tags that are used with the EP are Parameter and Local Tags and are not controller-scoped. This lets you create reusable code and you do not need to change tag names for each instance.

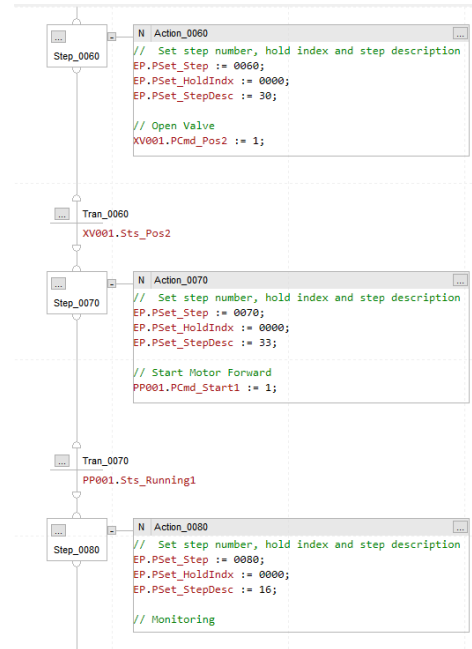
Scope: TK001_EP_TXIn_PHS Show: All Tags

Name	Usage	Data Type
Action_9010	Local	SFC_ACTION
Action_9020	Local	SFC_ACTION
Action_9030	Local	SFC_ACTION
Action_9040	Local	SFC_ACTION
Action_9050	Local	SFC_ACTION
Action_9999	Local	SFC_ACTION
AsctdDeviceAlarm	Local	LINT
EP	Local	raP_Opr_EPGen
Step_0000	Local	SFC_STEP
Step_0010	Local	SFC_STEP
Step_0020	Local	SFC_STEP
Step_0030	Local	SFC_STEP
Step_0040	Local	SFC_STEP
Step_0050	Local	SFC_STEP
Step_0060	Local	SFC_STEP
Step_0070	Local	SFC_STEP
Step_0080	Local	SFC_STEP
Step_0090	Local	SFC_STEP
Step_0100	Local	SFC_STEP
Step_0110	Local	SFC_STEP
Step_0120	Local	SFC_STEP
Step_0999	Local	SFC_STEP
Step_6000	Local	SFC_STEP
Step_6010	Local	SFC_STEP
Step_6020	Local	SFC_STEP
Step_6030	Local	SFC_STEP
Step_6999	Local	SFC_STEP
Step_7000	Local	SFC_STEP
Step_7010	Local	SFC_STEP

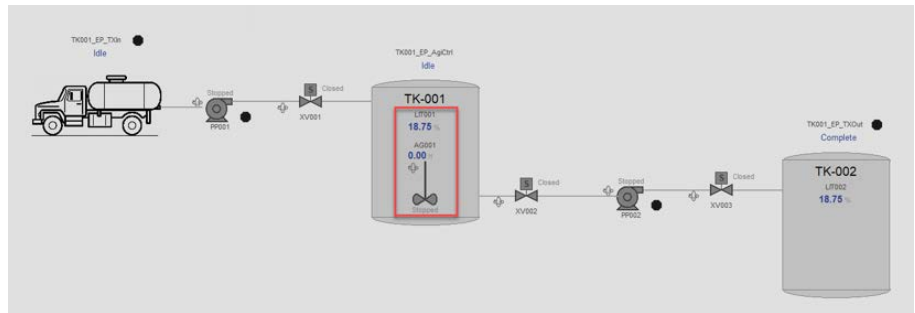
S88 State Routines - TK011_EP_TXIn_PHS

Each step has a continuous action instruction. This action sets the PSet_Step, PSet_HoldIdx, and PSet_StepDesc. It also sets the desired control module or equipment module commands and any other necessary code for each unique step.

In the following example, step0060 shows that XV001 is commanded to open (position2) and step0070 PP001 is commanded to start. The transitions confirm the state of the devices.

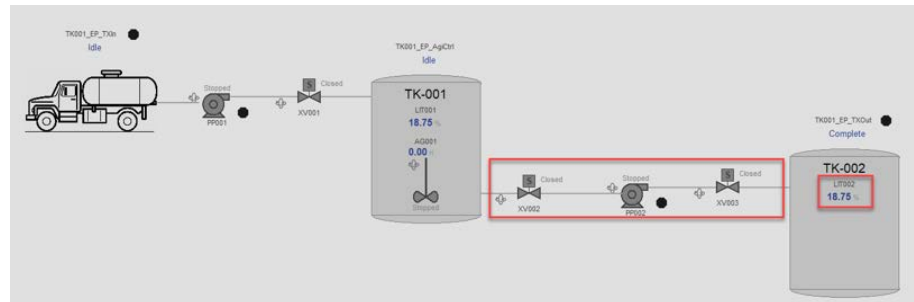


Equipment Phase Two - TK001_EP_AgiCtrl



State	Program the Logic as Follows
Running	<ol style="list-style-type: none"> 1. While TK001_EP_TXIn is filling the tank, TK001_EP_AgiCtrl is monitoring the level of transmitter LIT001. 2. When the EP is acquired, the parameter values are captured and the agitation timer is set. 3. When the agitation timer is set, the agitator on-delay and off-delay times are set. Note: There is a pre-determined level that LIT001 must read while filling before the agitator starts. There is also a pre-determined level that LIT001 must empty to before the agitator stops. 4. The on-delay and off-delay timers determine how much time it will take for the agitator to turn off or turn on after LIT001 achieves the desired value. 5. When the agitator is in the stopped state and LIT001 fills to the desired level, the on-delay timer begins. 6. When the on-delay timer is complete, the agitator begins agitating for the set amount of time – based on the agitator timer parameter. 7. The agitator stops in one of two cases: <ul style="list-style-type: none"> – When the agitator timer is complete or – When the tank level drops below a certain point because TK001_EP_TXOut transferred liquid out of the tank 8. When stopped, the exit condition (Complete, Aborted, Held, or Stopped) is reported and captured. 9. When captured, ownership of the EP is released and the phase is complete (Complete).
Aborting	<ol style="list-style-type: none"> 1. When the operator commands the EP into the aborting state, the agitator AG001 is commanded to stop. Note: There is no status check before exiting this step. 2. The state in which the EP leaves the sequence is set to 'Aborted,' and the report is captured. 3. Ownership of the EP is commanded to be released. 4. After the EP is commanded to release, the aborting state is complete (Aborted).
Holding	<ol style="list-style-type: none"> 1. When the operator commands the EP into the holding state, a system check is performed to determine the next course of action. 2. If the agitator is running forward, the speed reference is set to 10 percent. Otherwise, the following occurs: <ol style="list-style-type: none"> a. If the level of transmitter LIT001 is greater than the desired maximum value and the agitator is stopped, then the agitator is commanded to start again with a speed of 10 percent. b. If neither of these cases (Step 2 and Step 2a) are true, the level of transmitter LIT001 is below the target maximum value, and the agitator is running forward, the agitator is commanded to stop. Note: There is no transition condition to leave this step. 3. The state in which the EP leaves the sequence is captured as 'Held' and the report is captured. 4. If the report is captured properly, ownership of the EP is commanded to release. 5. When ownership is released, the holding state is complete (Held).
Resetting	<ol style="list-style-type: none"> 1. When the operator commands the EP into the resetting state, each of the other states (Complete, Aborted, and Stopped) are reset to the beginning of their sequences. 2. Ownership of the EP is released. 3. The resetting state is complete (Idle).
Restarting	<ol style="list-style-type: none"> 1. When an operator commands the EP into the restarting state, the running phase sequence is reset to the beginning of the sequence. 2. The holding phase sequence is set to the end of the sequence. 3. The restarting state is complete (Running).
Stopping	<ol style="list-style-type: none"> 1. When an operator commands the EP into the stopping state, agitator AG001 is commanded to stop. 2. When the pump is stopped, the status that the EP leaves the sequence in is captured as 'Stopped'. 3. The report is captured. 4. When the report is captured, ownership of the EP is commanded to release. 5. When ownership is released, the stopping state is complete (Stopped).

Equipment Phase Three - TK001_EP_TXOut



State	Program the Logic as Follows
Running	<ol style="list-style-type: none"> When ownership is requested and acquired, the desired level indicator LIT002 level parameter is captured. After capturing the LIT002 level parameter, valve XV002 is commanded to open. When XV002 is open, valve XV003 is commanded to open. When valves XV002 and XV003 statuses are open, pump PP002 is commanded to start. When pump PP002 reaches the running state, the EP monitors the level of tank LIT002. When tank LIT002 reaches the desired level, pump PP002 is commanded to stop. When pump PP002 is stopped, valve XV003 is commanded to close. When valve XV002 is closed, XV002 is commanded to close. After all objects are stopped/closed, a report sets and captures the exit status. When the report is captured, ownership of the EP is released and the running state is complete (Complete).
Aborting	<ol style="list-style-type: none"> When the operator commands the EP into the aborting state, pump PP002, inlet valve XV003 and outlet valve XV002 are commanded to close. Note: There is no status check for these actions before the next transition steps begin. When the objects in Step 1 are commanded to close via programming, the state that the EP leaves the sequence is in captured as 'Aborted'. The report is captured. Ownership of the EP is commanded to be released. After the EP is commanded to release, the aborting state is complete (Aborted).
Holding	<ol style="list-style-type: none"> When the operator commands the EP into the holding state, the state in which the EP leaves the sequence is captured as 'Held' and the report is captured. If the report is captured properly, ownership of the EP is commanded to release. When ownership is released, the holding state is complete (Held).
Resetting	<ol style="list-style-type: none"> When the operator commands the EP into the resetting state, ownership of the EP is released. When ownership is released, the resetting state is complete (Idle).
Restarting	When the operator commands the EP into the restarting state, the restarting state is set to complete (Running).
Stopping	<ol style="list-style-type: none"> When the operator commands the EP into the stopping state, pump PP002 is commanded to stop. When pump PP002 is stopped, inlet valve XV003 is commanded to close. When inlet valve XV003 is closed, outlet valve XV002 is commanded to close. When outlet valve XV002 is closed, the status that the EP leaves the sequence in is captured as 'Stopped'. The report is captured. When the report is captured, ownership of the EP is commanded to release. When ownership is released, the stopping state is complete (Stopped).

Notes:

Equipment Modules

Overview

Equipment modules (EMs) interact with the raP_Opr_EMGen block in the code. The EM can be controlled by several sources: external sources such as an EPGen block, programmatically (either independently or by being acquired by an EP), or via an operator.

If the program owns the module and the sequence code and is issuing commands to the module, the command source is listed as 'Program.'



If the operator owns the module and the operator is making commands to the EM via the HMI faceplate, the command source is listed as 'Operator.'



Each specific EM state design determines the module state control. However, depending upon the set ownership, an external source, program command, or operator interaction can shift the module state into another permitted module state.

raP_Opr_EMGen Block Configuration

For these parameters to function properly, configure the raP_Opr_EMGen before you program EM/EP sequence code.

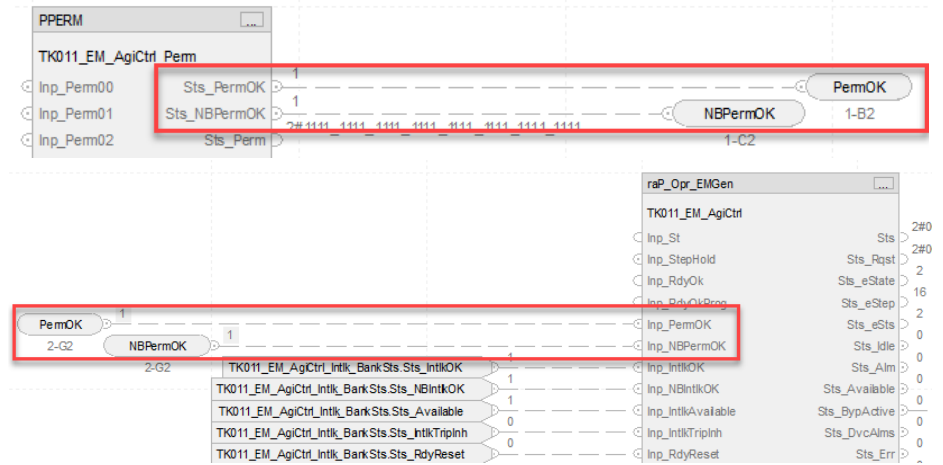
IMPORTANT

Rockwell Automation recommends setting individual permissives and individual interlocks to the same state, whether 0 or 1.
You can configure either state based on the needs of your environment.

Inp_PermOK and Inp_NBPermOK

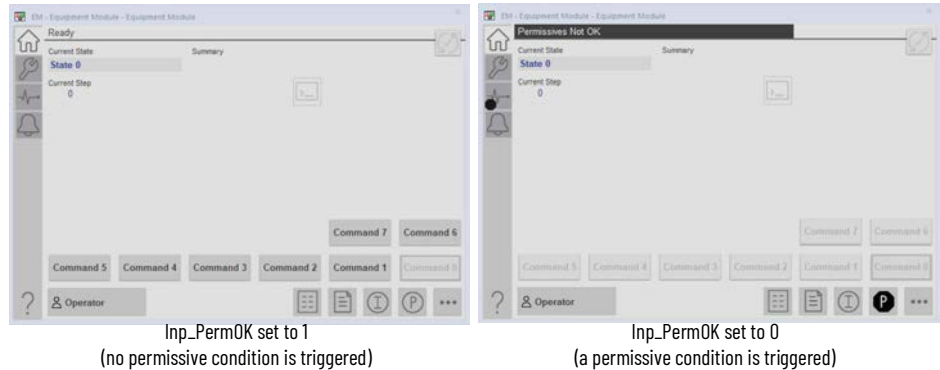
The Inp_PermOK and Inp_NBPermOK parameters monitor the status of the EM's permissives on the associated permissive blocks. To monitor the status of the associated permissive block properly, make a connection between the following:

- PPERM Sts_PermOK output and the raP_Opr_EMGen Inp_PermOK
- PPERM Sts_NBPermOK output and the raP_Opr_EMGen.Inp_NBPermOK input



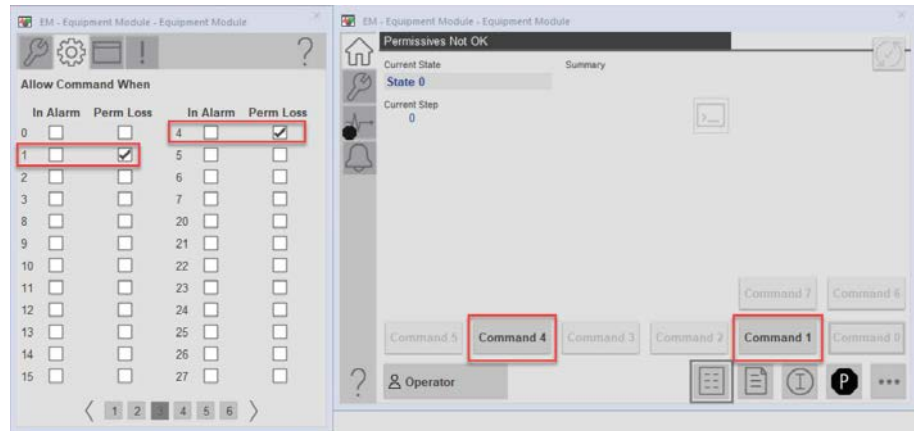
For more information on the PPERM instruction, see Rockwell Automation Library of Process Objects, publication [PROCES-RM203](#).

If no bypass-enabled permissive condition is triggered, the PPERM Sts_PermOK is set to 1. In this scenario, raP_Opr_EMGen Inp_PermOK is also set to 1. When the Inp_PermOK is set to 1, you can change the state of the EM in operator mode. During a permissive condition trigger, you cannot change the state of the EM.



When you use the Cfg_PermAllowCmd bitwise parameter, the EM can be commanded based on the configured states — even if the permissive conditions are not satisfied. In operator mode, the EM faceplate state buttons are available for those states that are configured to allow commands.

The following figure shows states 1 and 4 configured to Allow Command When Perm Loss.



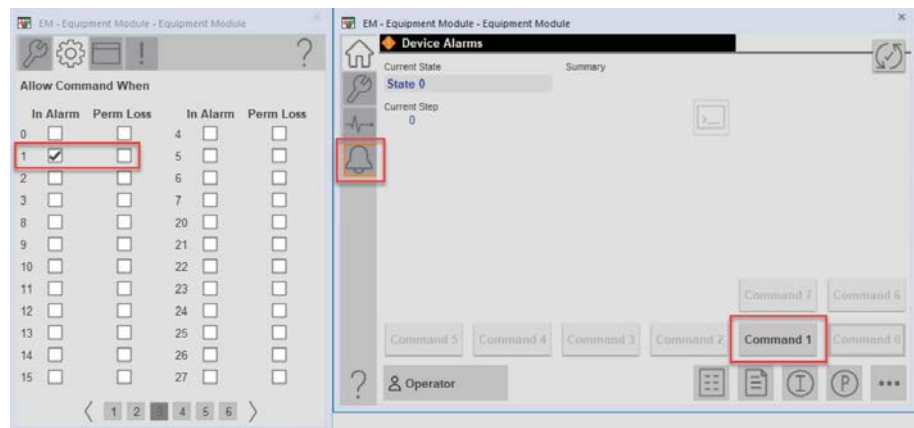
States 1 and 4 are permitted to be commanded during a permissive trip.

Permissive conditions function the same regardless of the command source: Operator, Program, and External EP state commands are prohibited until the permissives are OK.

Similarly, while an alarm state is active, you can use the Cfg_AlmAllowCmd bitwise parameter to command the EM based on the configuration states.



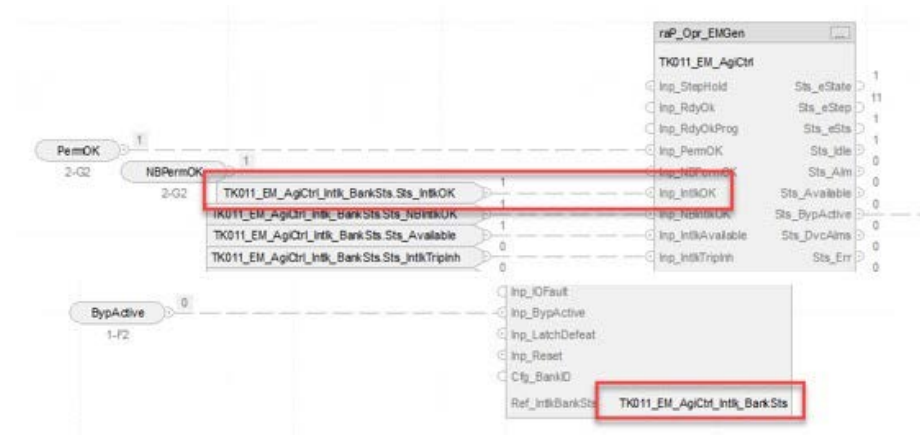
Alarm trigger inputs alone do not affect the available state command buttons. An alarm must be configured to be 'Used' in Alarm Manager and 'In Alarm' state. Do not configure the In Alarm for your configured 'Idle' state so that you do not get all alarms while the EM is Idle.



State 1 is permitted to be commanded during an active alarm.
The Idle state is configured as State 1.

Inp_IntlkOK and Inp_NBIntlkOK

The Inp_IntlkOK and Inp_NBIntlkOK parameters monitor the status of the associated EM interlock blocks. To monitor the status of the interlock blocks properly, create a bank status tag and associate it with the Ref_IntlkBankSts pin for each PINTLK block of the EM. Then associate raP_Opr_EMGen Inp_IntlkOK with Sts_IntlkOK of the bank status tag that was created.



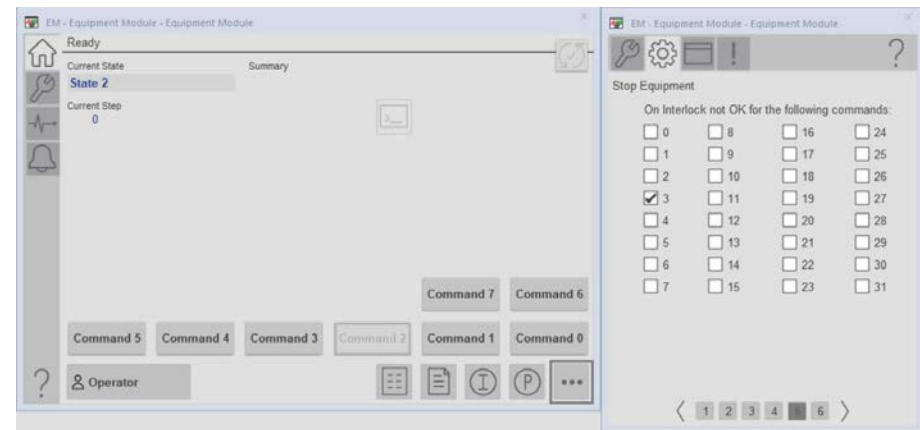
For more information on the PINTLK instruction, see Rockwell Automation Library of Process Objects, publication [PROCES-RM203](#).

If no bypassable interlock trip conditions are present, the PINTLK bank Sts_IntlkOK is set to 1. In this scenario, set raP_Opr_EMGen Inp_IntlkOK to 1.

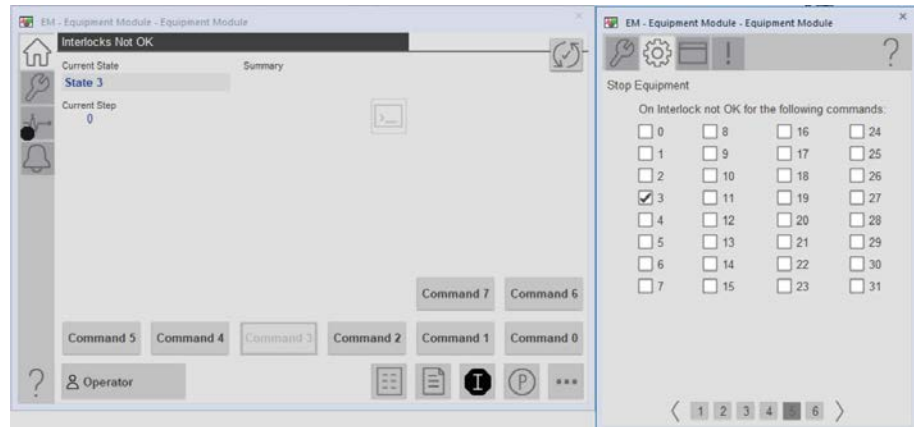
If you want interlock trips to affect EM functionality, use the Sts_NotRdy and Sts_IntlkTrip status parameters. You can also use the Cfg_ShedOnIntlk bitwise parameter to configure which states monitor the Interlock status block.

In the following example, State 3 is configured to stop equipment on an Interlock trip. State 2 is not configured to stop equipment.

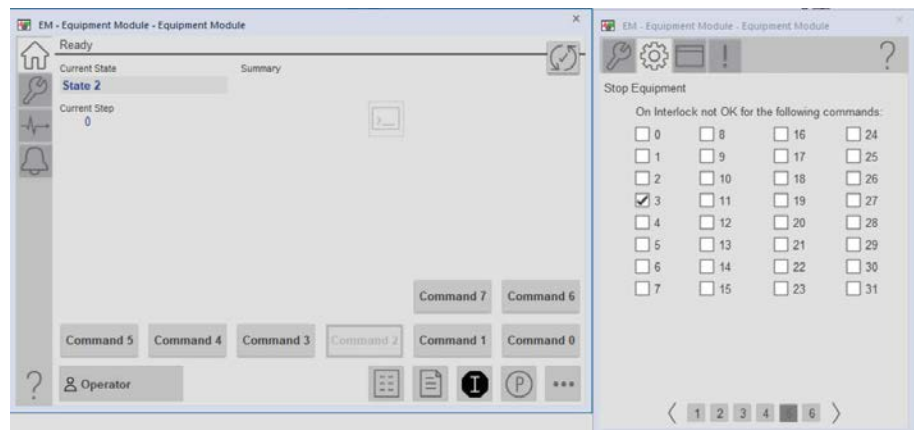
Case 1: The current state is State 2. Interlocks Ok, both Sts_NotRdy and Sts_IntlkTrip = 0



Case 2: Interlocks Not Ok, both Sts_NotRdy and Sts_IntlkTrip = 1. The status in 'Not Ready' because State 3 is configured to Stop Equipment on an interlock trip.

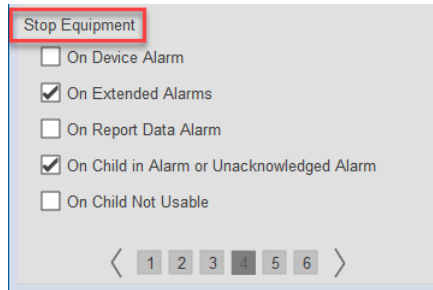


Case 3: Interlocks Not Ok, Sts_NotRdy = 0, Sts_IntlkTrip = 1. The status is 'Ready' because State 2 is configured not to Stop Equipment on an interlock trip.



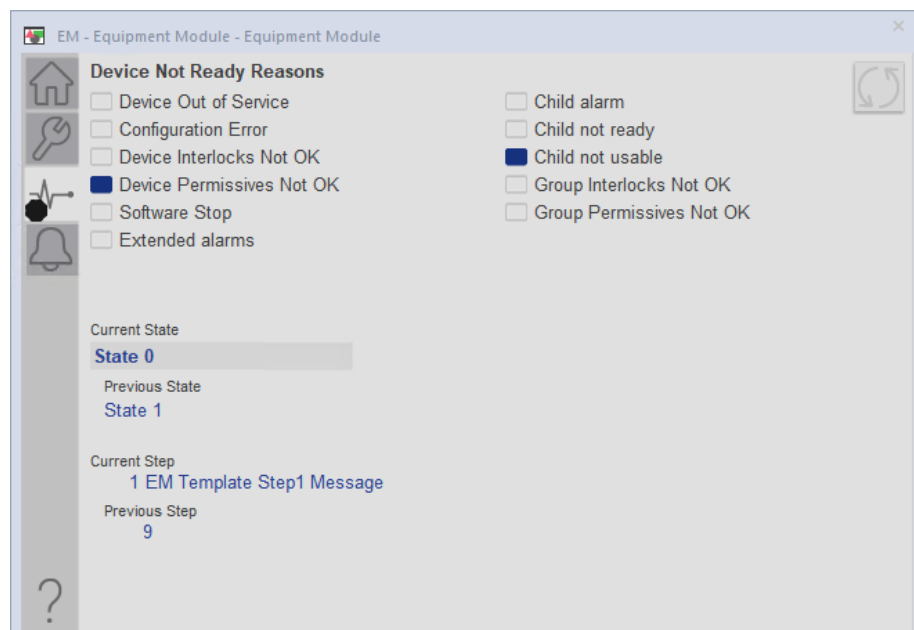
The EM Sts_IntlkAvailable is evaluated regardless of the configuration of the Stop Equipment. EM Sts_IntlkAvailable is set to 1 when the interlock conditions are OK.

In addition to the Interlock Trip, there are five other optional configurations to Stop Equipment that affect Sts_NotRdy.



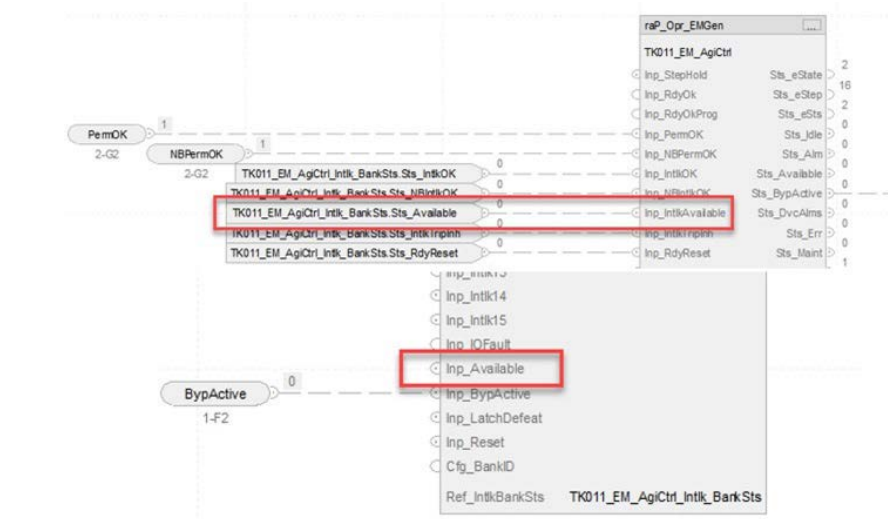
- **On Device Alarm:** Stops equipment if the EM pin Inp_DvcAlms is true as a result of the AsctdDeviceAlarm tag are triggered.
- **On Extended Alarms:** Stops equipment if the raP_Opr_ExtddAlm instructions associated with the EM are triggered.
- **On Report Data Alarm:** Stops equipment if the EM_RPT report parameters associated with the EM are in error.
- **On Child in Alarm or Unacknowledged Alarm:** When a child is in alarm or has an unacknowledged alarm and you use the Bus organizer, this configuration stops equipment.
- **On Child Not Usable:** When you request ownership of the EM, children use the Bus organizer and set the EM.Sts_ChildrenGood = 0. This configuration stops the equipment. You can configure the EM delay for this status.

You can use the Diagnostic tab on the EM faceplate to troubleshoot a Not Ready status.



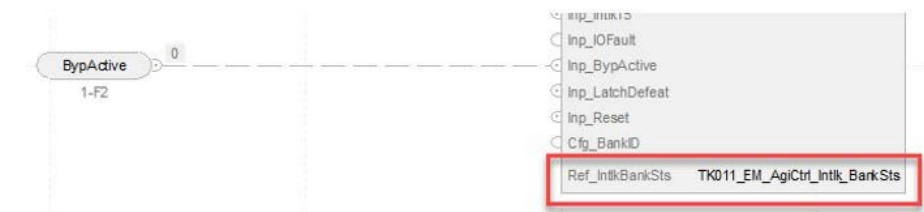
Inp_IntlkAvailable

The Inp_IntlkAvailable parameter monitors whether the EM is available for an interlock trip. To monitor the status of the interlock blocks, create a bank status tag and associate it with the Ref_IntlkBankSts pin, for each PINTLK block of the EM. Then associate the raP_Opr_EMGen Inp_IntlkAvailable parameters with the Sts_Available parameter of the bank status tag that was created.

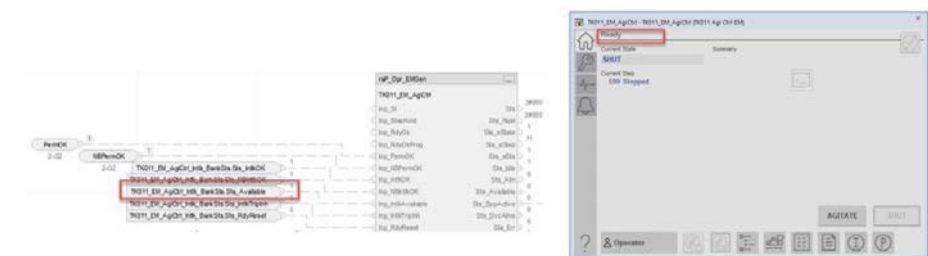


The Sts_Available parameter value is determined by a combination of the Inp_Available parameter's value and the configuration type of each of the interlocks block's input conditions. When Inp_Available=1 and the interlock block has an input condition that is configured as protective, device, mechanical, or process interlock type and that input condition is not in the OK state, the Sts_Available is set to 1 and the Sts_IntlkOK/ Sts_NBIntlkOK is set to 0. When Inp_Available=1 and the interlock block has an input condition that is configured as safety, electrical, operational, or general interlock type and that input condition not in the OK state, the Sts_Available and the Sts_IntlkOK/ Sts_NBIntlkOK are set to 0.

The Sts_IntlkAvailable output indicates that the equipment is ready to operate and all safety, electrical, operational, or general type interlocks that affect availability are working properly.



On an interlock trip condition, the EM.Sts_IntlkAvailable is 0 regardless of the configuration of the Stop Equipment Cfg_ShedOnIntlk in that state. As discussed in the [Inp_IntlkOK](#) and [Inp_NBIntlkOK](#) section, Cfg_ShedOnIntlk affects the EM Sts_IntlkTrip and Sts_NotRdy parameters, but not the Sts_IntlkAvailable parameter.





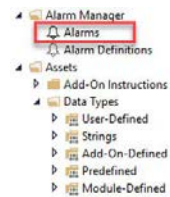
Inp_IntlkTriplnh

The Inp_IntlkTriplnh EM parameter monitors the system for any interlock trip alarms that are inhibited. To monitor this parameter properly, create a bank status tag and connect it to each PINTLK block of the EM and associate it with the Ref_IntlkBankSts pin. Then tie the raP_Opr_EMGen Inp_IntlkTriplnh parameter to the bank status tag Sts_IntlkTriplnh parameter.

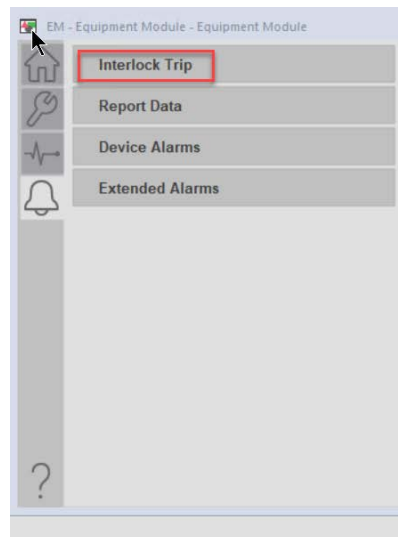
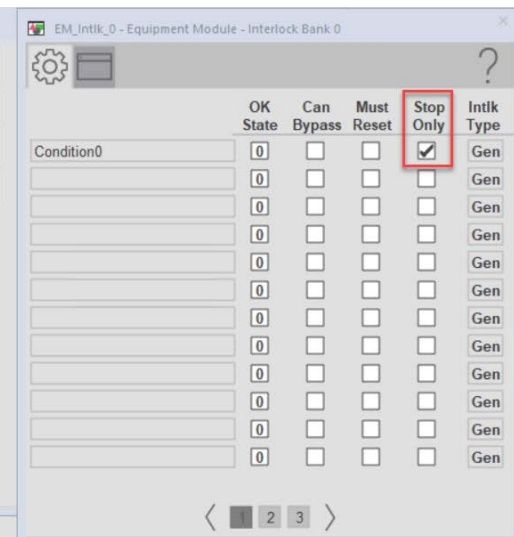


To inhibit the EM Interlock Trip Alarm, use the Cfg_StopOnly parameter of the PINTLK block to determine which interlock conditions must be configured as 'Stop Only.' A Cfg_StopOnly value of 1 for an interlock condition indicates that when the interlock condition trips, the EM Interlock Trip does not go into alarm.

To create a Trip Alarm notification on the EM faceplate and the HMI Alarm Summary/Banner, go to the controller Alarm Manager and set the EM Interlock Trip Alarm to 'Use.'



<input checked="" type="checkbox"/>	TK011_EM_AgiCtrl	Alm_DvcAlm	TRIP	TK011_EM_AgiCtrl.Sts_DvcAlms	= 1
<input checked="" type="checkbox"/>	TK011_EM_AgiCtrl	Alm_IntlkTri	TRIP	TK011_EM_AgiCtrl.Sts_IntlkTrip	= 1
<input checked="" type="checkbox"/>	TK011_EM_AgiCtrl	Alm_AprcAlm	TRIP	TK011_EM_AgiCtrl.Sts_AprcAlms	= 1
<input checked="" type="checkbox"/>	TK011_EM_AgiCtrl.Ex	Alm_Alarm	TRIP	TK011_EM_AgiCtrl.ExtddAlm_00.Sts	= 1
<input checked="" type="checkbox"/>	TK011_EM_AgiCtrl.Pr	Alm_AlertTri	TRIP	TK011_EM_AgiCtrl.Prompt.Sts_AlertTimeOut	= 1

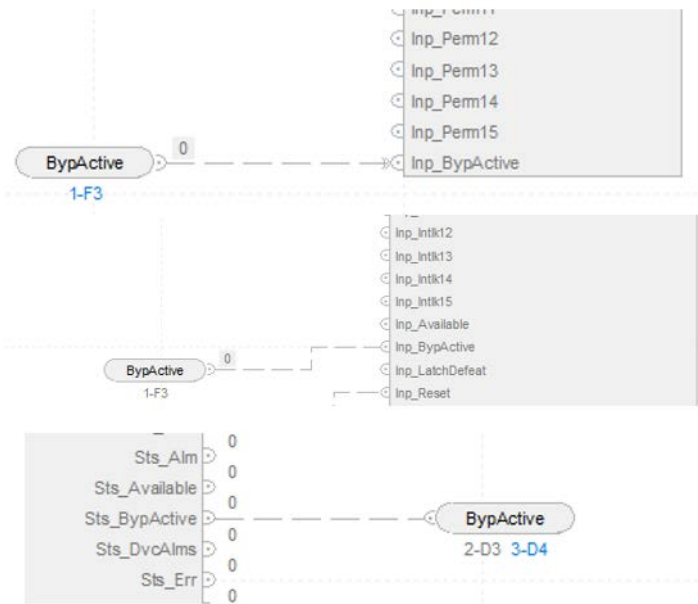



	OK State	Can Bypass	Must Reset	Stop Only	Intlk Type
Condition0	0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Gen
	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Gen
	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Gen
	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Gen
	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Gen
	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Gen
	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Gen
	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Gen
	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Gen
	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Gen
	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Gen
	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Gen
	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Gen
	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Gen

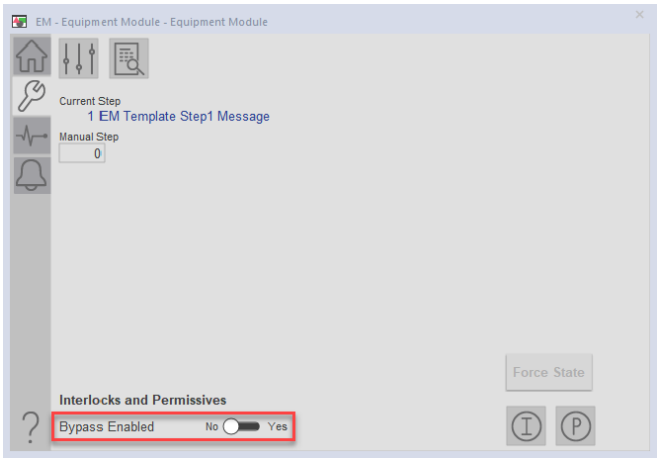
Sts_BypActive

The Sts_BypActive parameter monitors whether bypass is enabled on the EM. Some interlocks/permisives for the EM are considered bypassable, which allows an engineer or maintenance to ignore the conditions and proceed with the state as normal. Some conditions are considered non-bypassable, which means that the only way to resume normal state conditions is to remedy the tripped interlock/permisive.

If you need to bypass interlocks and permisives, tie a tag to each instance of a PINTLK or PPERM block to the Inp_BypActive pin. Then tie that tag to the Sts_BypActive pin of the EM block.



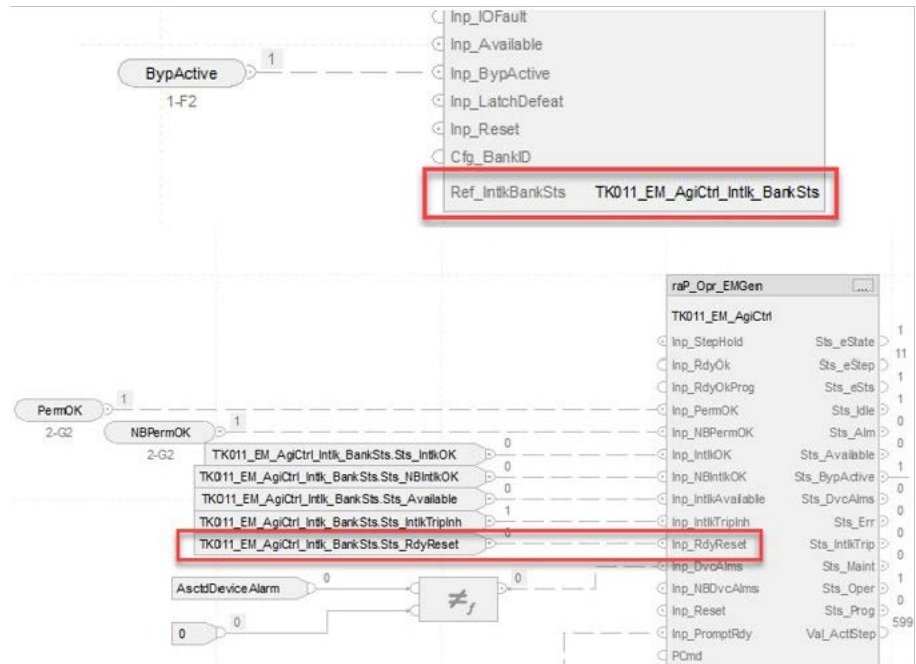
You can enable the EM bypass from the HMI faceplate as follows:



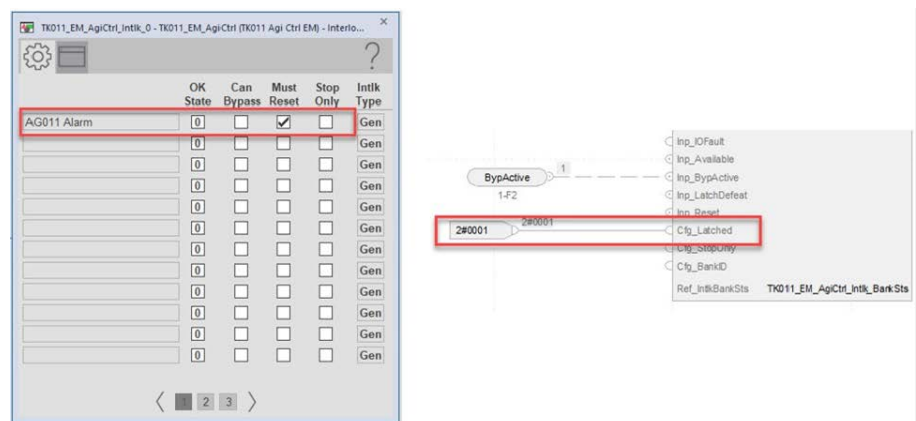
Inp_RdyReset

The Inp_RdyReset EM parameter monitors whether a latched interlock is ready to be reset. To monitor a reset- requested status, create and associate a bank status tag with the Ref_IntlkBankSts pin for each PINTLK block of the EM. Then associate the raP_Opr_EMGen Inp_RdyReset parameter with the Sts_RdyReset parameter of the bank status tag. This functionality is consistent with the other PlantPAx® Interlock control strategies.

See PlantPAx Process Control Instructions User Manual, publication [PROCES-RM215](#) for more information.



Interlocks can be configured individually to require a reset by modifying the PINTLK Cfg_Latched parameter. For example, if you set a Cfg_Latched value to 1, an interlock trip remains active until the interlock trigger condition returns to normal and you reset the interlock.

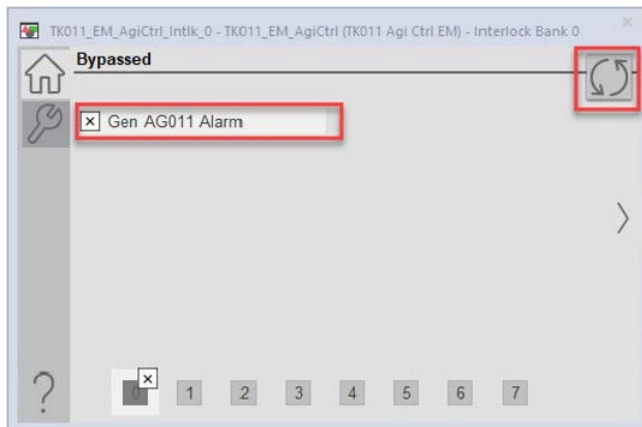


If an interlock condition is configured with 'reset required' while the interlock condition remains tripped, the `Inp_RdyReset` remains reset to 0. When the interlock condition clears, `Inp_RdyReset` becomes 1 and an HMI reset becomes available.



Because the interlock condition is true, `Inp_RdyReset` is reset to 0. You cannot perform a reset.

When the interlock condition is no longer tripped, `Inp_RdyReset` is set to 1 and the reset on the HMI becomes available.



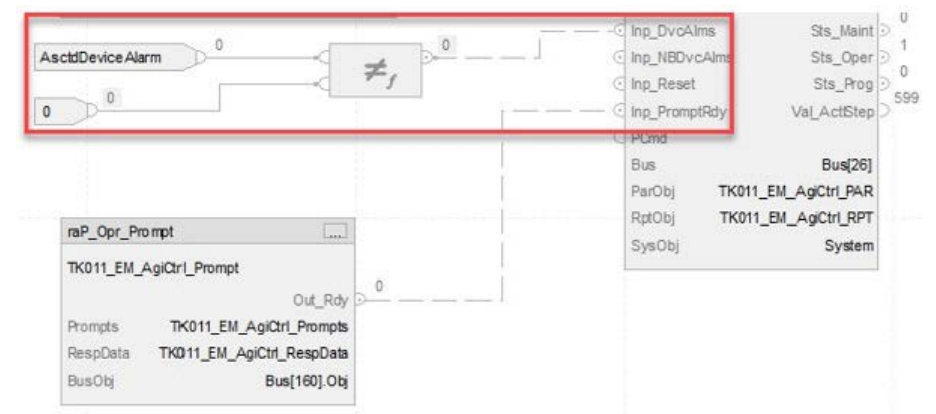
The interlock condition is false, but the EM is in an interlock trip until a reset occurs. A reset can occur.

When a reset has been issued, the `Inp_RdyReset` parameter is cleared to 0, and the `Inp_IntlkOK` and/or `Inp_NBIntlkOK` parameters are set to 1.

Inp_DvcAlms

When the Bus organizer is not used, you can use the Inp_DvcAlms parameter to monitor the alarm state of devices that are associated with the EM – including children device alarms connected to the EM.

Connect each object to one parameter of an associated alarms tag – listed as ‘AsctdDeviceAlarm’ in the image below – and pin the objects the Inp_DvcAlms input as follows.

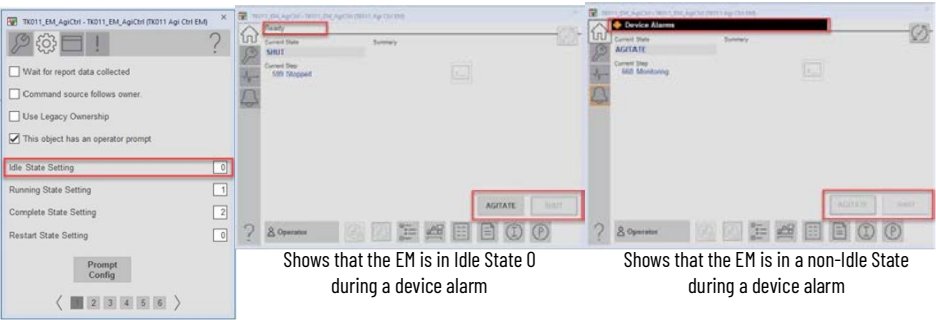


A nonzero indicates that one or more objects that are associated with the EM are in alarm. In the controller Alarm Manager, the EM Device Alarm must be set to ‘Use,’ which then displays the Device Alarm on the EM faceplate and makes the alarm active.

State	Use	Owner	Name	Type	Input	Expression	Limit
	<input checked="" type="checkbox"/>	TK011_EM_AgiCtrl	Alm_DvcAlms	TRIP	TK011_EM_AgiCtrl.Sts_DvcAlms	= 1	
	<input checked="" type="checkbox"/>	TK011_EM_AgiCtrl	Alm_IntlkTrip	TRIP	TK011_EM_AgiCtrl.Sts_IntlkTrip	= 1	
	<input type="checkbox"/>	TK011_EM_AgiCtrl	Alm_RptData	TRIP	TK011_EM_AgiCtrl.Sts_RptData	= 1	
	<input type="checkbox"/>	TK011_EM_AgiCtrl_Ex	Alm_Alarm	TRIP	TK011_EM_AgiCtrl.ExtddAlm_00.Sts	= 1	
	<input checked="" type="checkbox"/>	TK011_EM_AgiCtrl_Pr	Alm_AlertTimeOut	TRIP	TK011_EM_AgiCtrl.Prompt.Sts_AlertTimeOut	= 1	

he Inp_DvcAlms parameter of the EM is reflected in the Sts_DvcAlms parameter, even when the Alarm Manager alarm is set to ‘Use.’

To trigger the EM into a Device Alarm, be sure that the EM is not in the EM defined Idle state. The Idle state acts as a gate for the EM Device Alarm and keeps the EM status parameter Sts_DvcAlms reset to 0. In the example below, Shut (state 0) is defined as the Idle state.



IMPORTANT The Inp_DvcAlms must only be used for device alarms that you want to bypass. When the EM bypass is enabled, the EM Sts_DvcAlms is not triggered and remains set to 0.

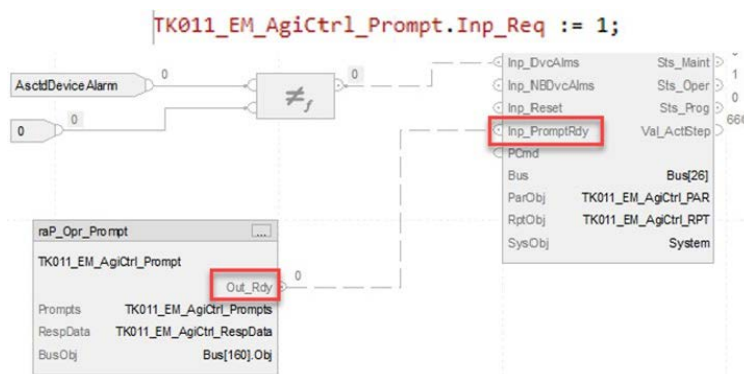
Extended Alarms: Can be used to display and track unique alarm conditions relative to the EM or EP instance. See [Additional Configuration](#) for more information.

Inp_PromptRdy

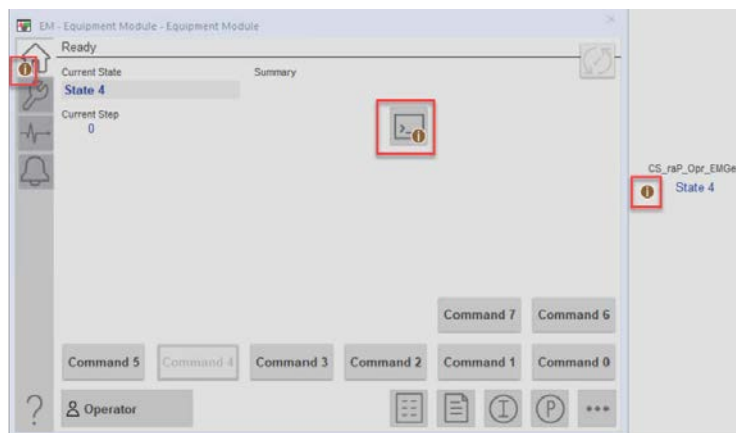
The Inp_PromptRdy parameter indicates that the associated raP_Opr_Prompt block is ready to send a prompt request to the operator.

To enable the prompt object to appear on the raP_Opr_Prompt block, set the Inp_Req parameter on the raP_Opr_Prompt block that is associated with the EM to 1. If you want to enable operator prompts, use logic to set the Inp_Req to 1 and the Inp_Ref to the desired Prompt array value. Program the logic so that when you acknowledge the prompt, Inp_Req resets to 0.

To cause the EM to display and notify the operator prompt in the HMI faceplate, pin the Prompt block Out_Rdy to the EM Inp_PromptRdy parameter.



When the EM Inp_PromptRdy input is set to 1, an informational breadcrumb image is visible on the EM global object and on the EM Home tab faceplate. In this scenario, no additional programming is required. The Prompt button is available and used for the operator to view and acknowledge the prompt.



For more information on the PROMPT instruction, see Rockwell Automation Library of Process Objects, publication [PROCES-RM203](#).

Inp_EnableNavTrans

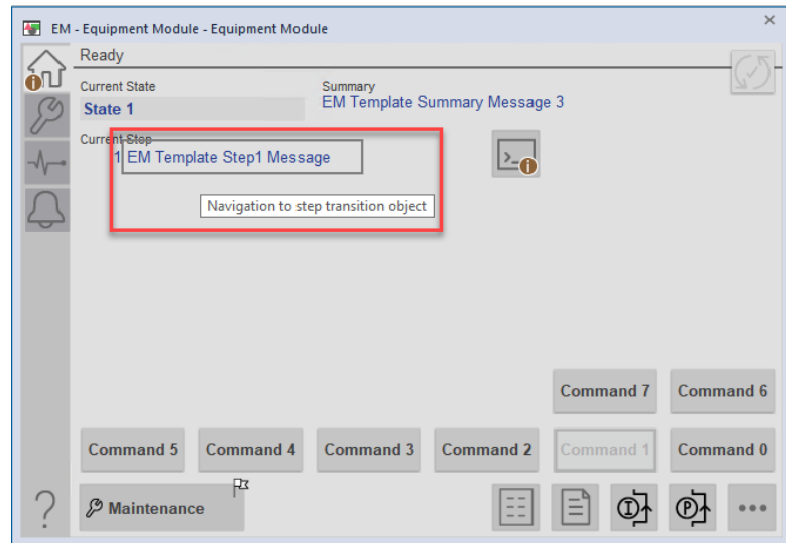
The Inp_EnableNavTrans parameter is used to enable visibility navigation for step transitions, via additional programming. When enabled and set to =1, a touch point with a tooltip is displayed on the EP faceplate. When you use this feature, system operators are able to troubleshoot what conditions are preventing step transitions.

Two common tag types are: PPERM (Permissive) and PBL (Boolean logic). Both of these tags provide instructions for gaining insight into transition conditions. You can use any PlantPAx object because the touchpoint uses the NavToDisplay macro. However, the PPERM and the PBL are typically the most flexible. With a PINTLK block, multiple interlocks can trip simultaneously,

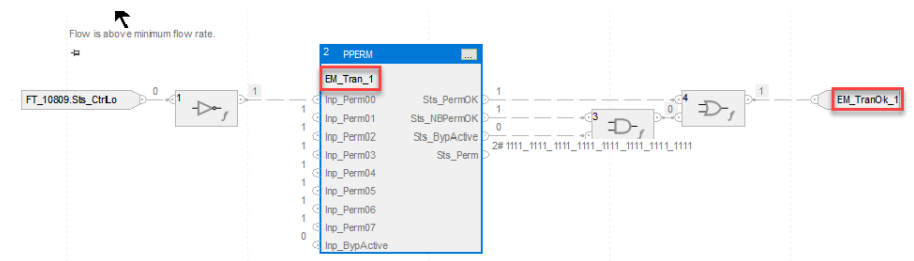
so knowing which interlock tripped first is not useful for gaining insight into transition conditions.

Name the transition tag syntax the same as the EM tag. Concatenated the name with '_Tran_' and then add the step number. For example, in step 20, EM_Tran_20 is the transition tag for the local EM tag. This tag must have the same path location as the EM tag, which is typically a programmed scoped tag.

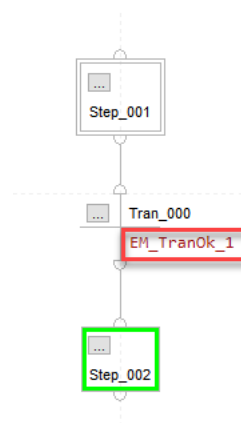
EM HMI faceplate: When the `Inp_EnableNavTrans` input parameter is true (1), the Current Step description has touch navigation that lets you open the step transition object faceplate.



Transition logic: Consider leveraging the `Sts_BypActive` to let certain conditions be bypassed from the faceplate. You can customize the Boolean tag name for the SFC transition – for example, 'EM_TranOk_1'.



State Step Sequence logic: Use a local Boolean tag on the output of the PPERM instruction for the SFC Transition Boolean logic. When the tag is set to true, the SFC moves to the next step.



MSet_Step (Manual Step)

The EM can be forced to move to a specific State Step by performing the following steps.

1. Put the EM into Maintenance mode.



Operators and supervisors do not have this security privilege.

2. From the Maintenance tab on the EM faceplate, enter a Manual Step number.



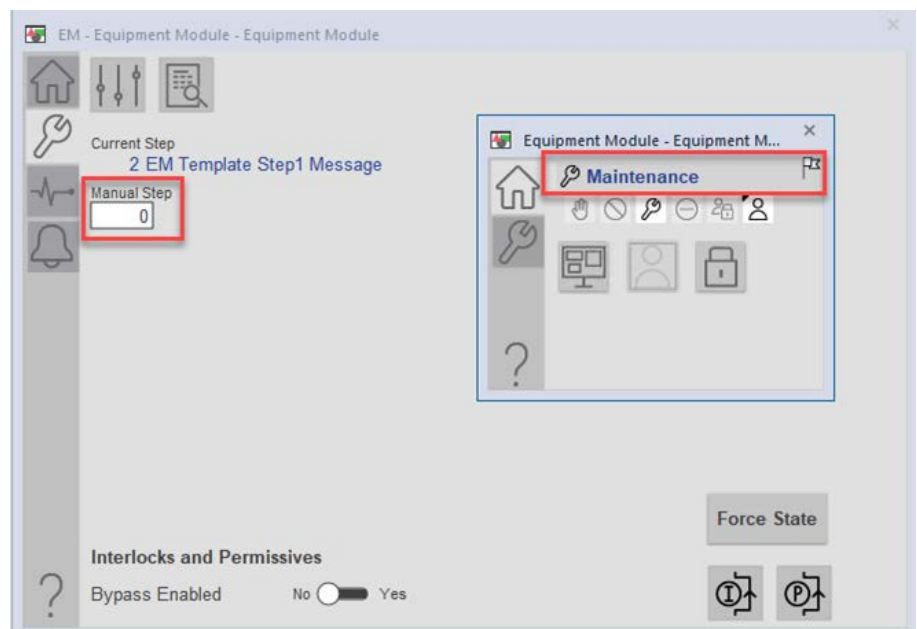
This numeric Manual Step entry writes directly to the EM MSet_Step parameter.

3. The EM instruction processes the command and sets the EM Val_ActStep.

IMPORTANT

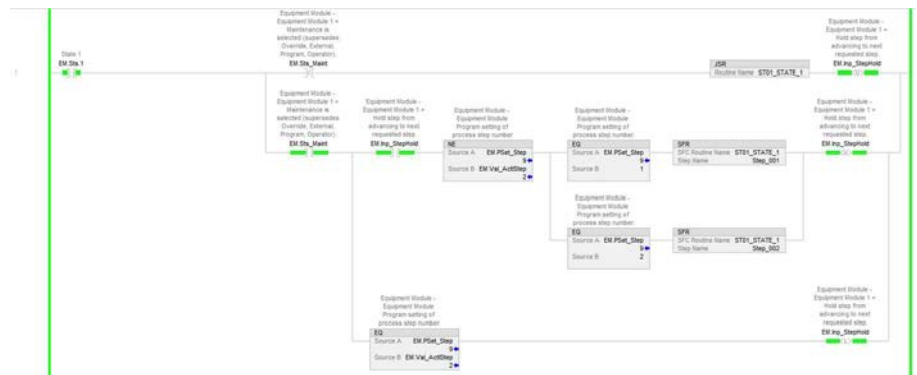
Make sure that the entry value is a valid and configured step, especially when using SFC step logic.

4. When you press Enter on the keyboard, the Manual Entry value immediately resets to 0.



Monitor the EM Val_ActStep or EM PSet_Step parameter value when the EM is in Maintenance Mode to determine the associated step logic that is needed to execute for manual step jumps.

You can also use EP Inp_StepHold to verify that the MSet_Step value is valid by monitoring the PSet_Step before the step request is processed. When you use Inp_StepHold in this way, you can validate or ignore the step value by monitoring the PSet_Step value. You can choose if you want to make step changes within the same state only. The PSet_Step is available in any mode and does not require the EP to be in Program mode.



MCmd_StateForce (Force State Complete)

To program the EM to force the current state to State Complete, perform the following steps.

1. Put the EM into Maintenance mode.

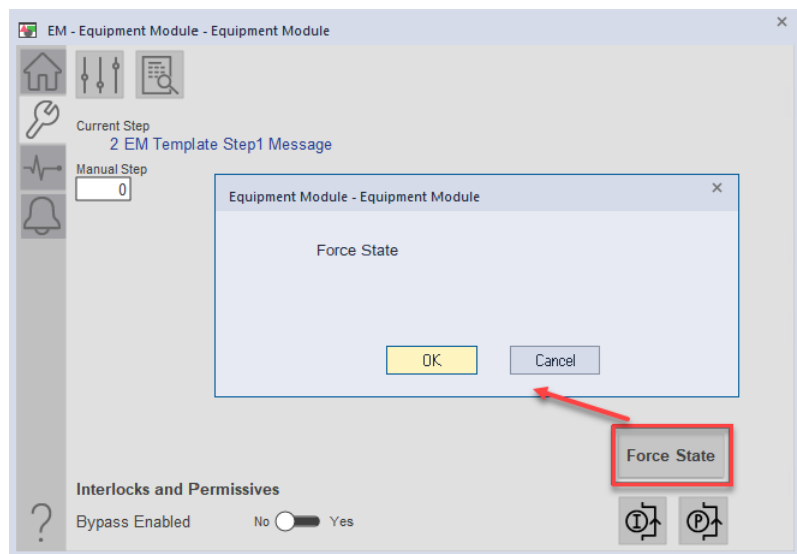


Operators and supervisors do not have this security privilege.

2. From the Maintenance tab on the EM faceplate, select the Force State button.
The Force State button writes to the MCmd_StateForce parameter directly and the EM sets the Sts_StateCompleteRqst internally before self-resetting.
3. Use the SFR command to connect the Sts_StateCompleteRqst to the required location in the State Model logic.
4. Set the desired state/step for SFC logic.

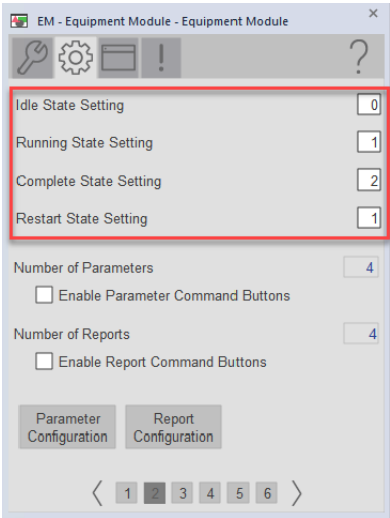


- The EM has a PCmd_StateForce that can be used to set the Sts_StateCompleteRqst, via programming.



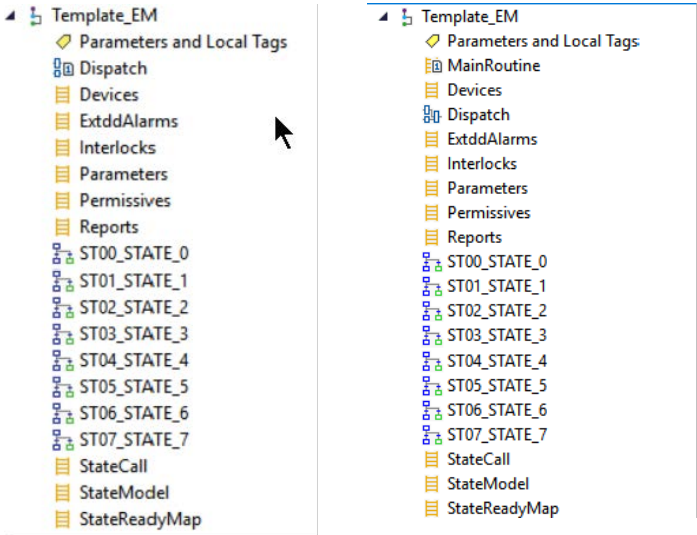
State Settings (Maintenance Configuration)

The EM can define the Idle, Running, Complete, and Restarting States. You do not need to make these state definitions unique — they can follow the state model. The EM Cfg_IdleState, Cfg_RunState, Cfg_CompleteState, and Cfg_RestartState can be used in external logic. Alarms do not inhibit state commands when the EM is Idle.



EM Control Strategy (defaults)

The following example shows a ControlLogix® Template EM Program that was created using Application Code Manager (ACM). The number and type of routines vary depending on the ACM build configuration. The following example shows two templates — one with a MainRoutine and one without a MainRoutine.

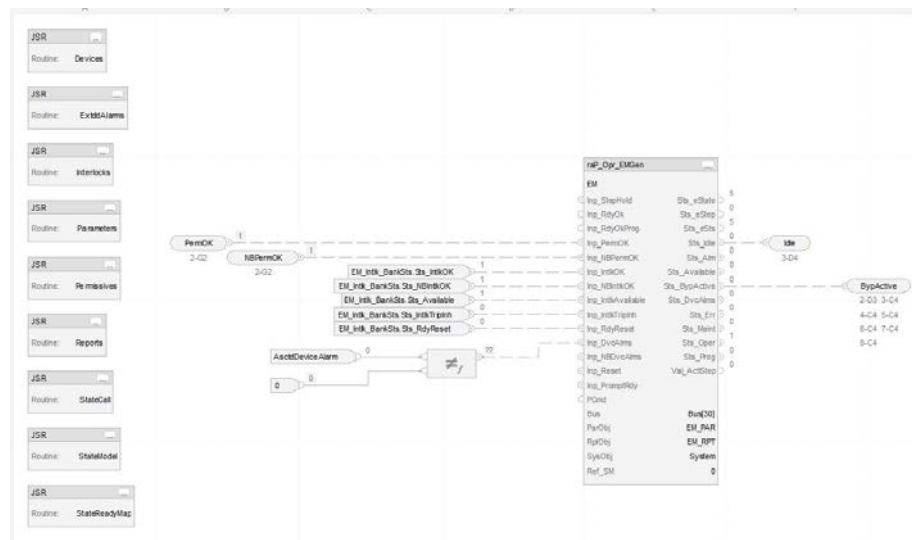


EM Routines (defaults)

MainRoutine: Contains the Jump to Subroutine JSR instructions. When you use this routine, you do not need to use additional/similar JSR calls in the Dispatch Routine. The MainRoutine also allows for routine order of execution.

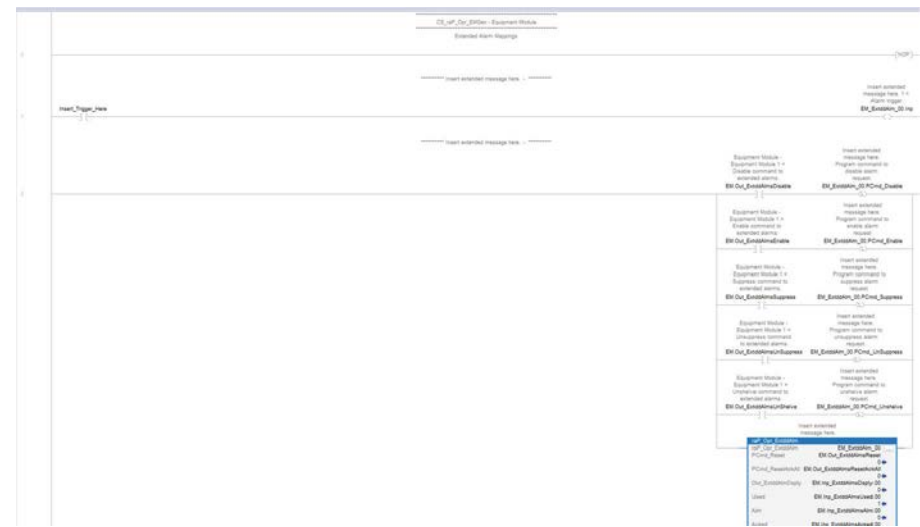


Dispatch: Contains the raP_Op_EMGen instruction, the Interlock and Permissives instructions (additional sheets), and the calls to the other routines (non-MainRoutine cases). If you use a MainRoutine, you do not need the JSR instructions on the left of the sheet.



Devices (optional): Contains the Control Module device mapping. You can use parameter connections or direct tag references in the state/step logic.

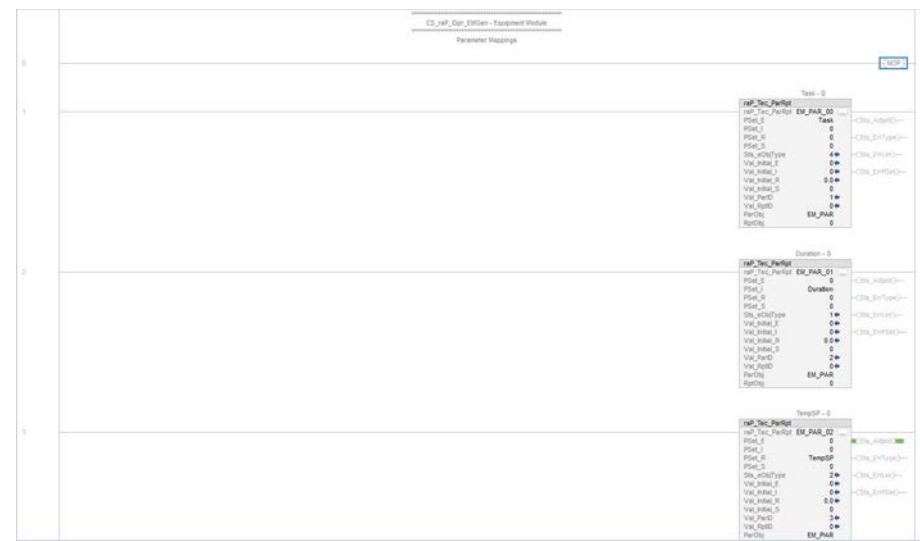
ExtddAlarms (optional): Extended Alarm. Replace the placeholder tag 'Insert_Trigger_Here' with the alarm condition trigger.



Interlocks and Permissives (optional): Interlock/Permissive conditions can be mapped in this ladder logic routine. For complex interlock/permissive conditions, use ladder logic. In less complex scenarios, you can pin the expressions directly to the interlock/permissive input pins in function block.

Parameters (optional): PlantPax has four different types of parameters that use the raP_Tec_ParRpt instruction, including: Enumeration, Integer, Real, and String parameter types.

Use the default local parameter tagname 'EM' for the EMGen. The ParObj parameter must be set as the local parameter tag named 'EM_PAR.'



Reports (optional): Reports the mapping routine. See Parameters (optional) routine for more information. The RptObj must be set to use the local parameter tag titled 'EM_RPT.'

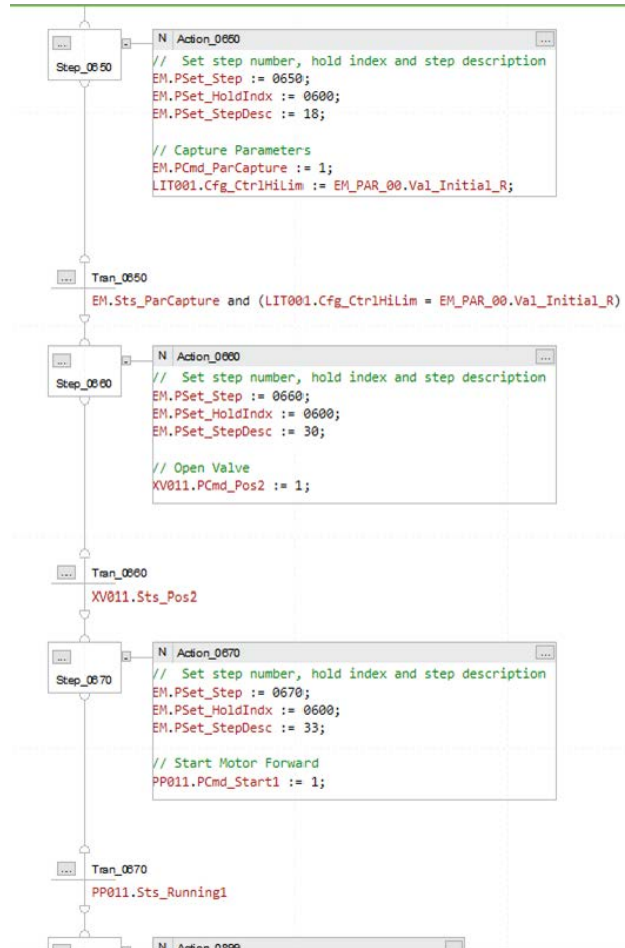
STOx_State_x: State step logic. When you configure the ACM, use either Sequential Function Chart (SFC) or Ladder logic. Each State has a dedicated routine. Rockwell Automation recommends that you standardize local tagnames for the SFC Steps, Actions, and Transitions.

When a transition is true, the logic progresses to the next step. Once in that step, the Action sets the PSet step to the desired value.

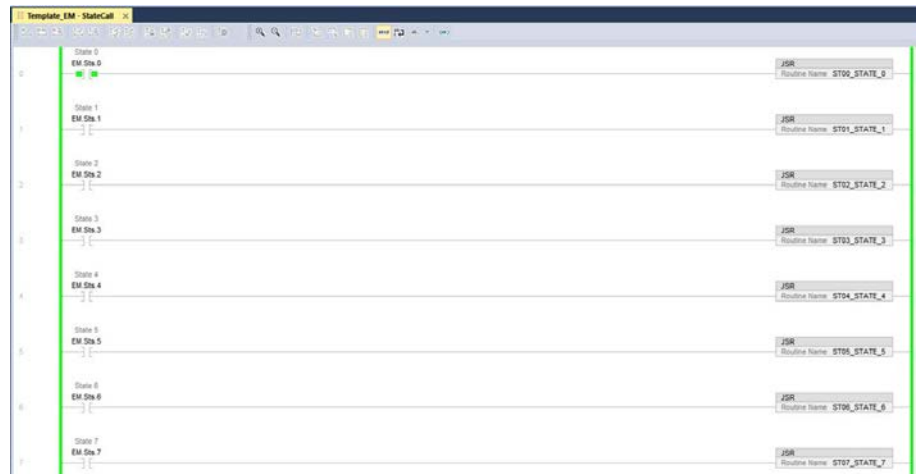
IMPORTANT When you restart the system, you must also set the Holding Index.

PSet_StepDesc is a message index that is used on the EM faceplate and references a local message. See the next StateCall section for more information on how the local message is used.

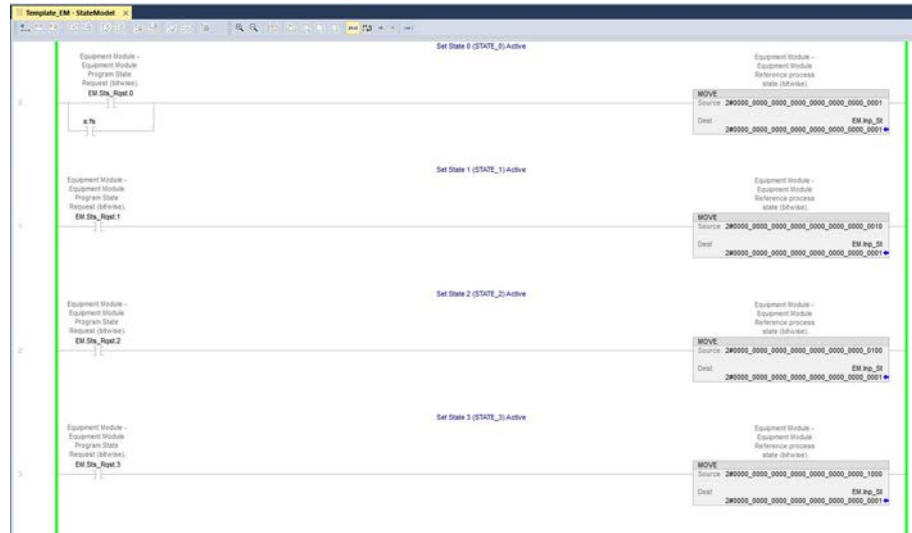
In the following example, the level transmitter (LIT001) Ctrl Hi Limit is set to the initial value of an EM parameter and valve XV011 is commanded open. When valve XV011 is confirmed open, pump PP011 is commanded to run.



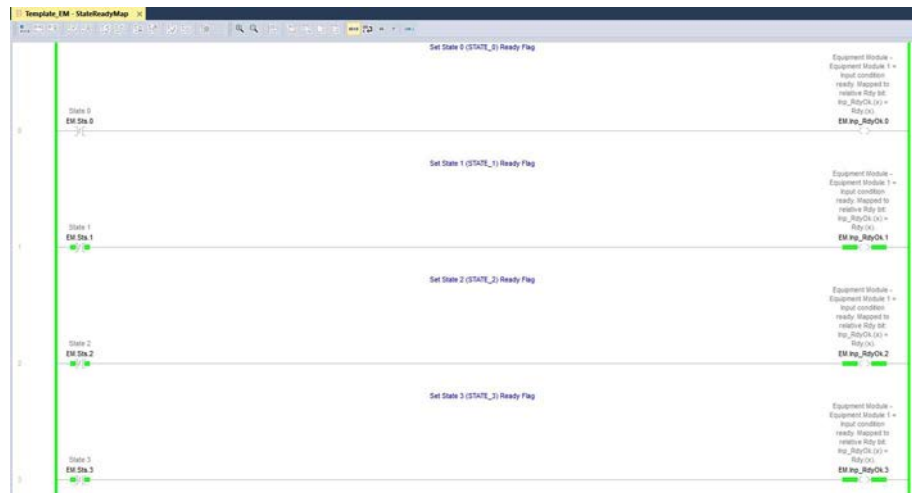
StateCall: The StateCall routine uses the EM State status bitwise (EM.Sts.x) to call the corresponding State routine.



StateModel: The StateModel routine uses the EM Request status bitwise (EM.Sts_Rqst.x) to move the constant state bitwise value into the EM Input State bitwise (EM.Inp_St). Requests can be sent from the operator faceplate, program, or external commands. In the following example, programming suggestions are used to clear the SFC Stop/End tag and Reset SFC instruction SFR.

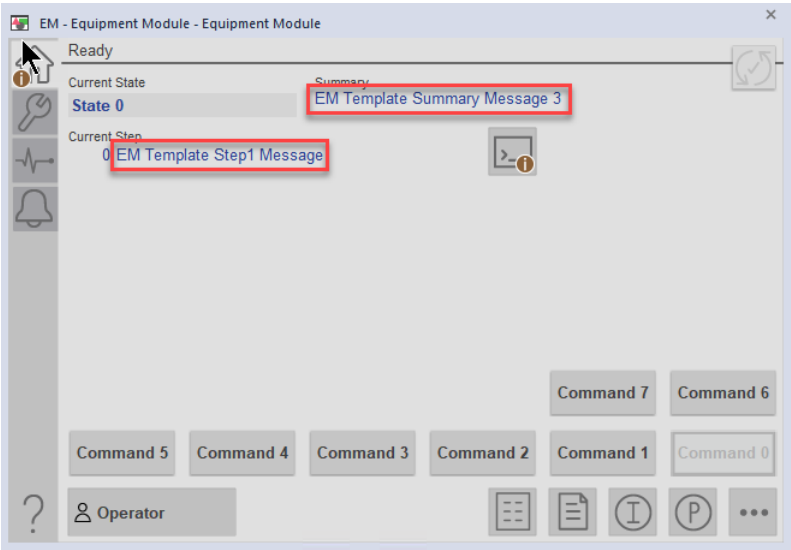


StateReadyMap: Use the StateReadyMap routine to control the visibility of the State buttons on the EM faceplate. When the state is active, the button is disabled. In operator mode, you can determine which buttons are visible.

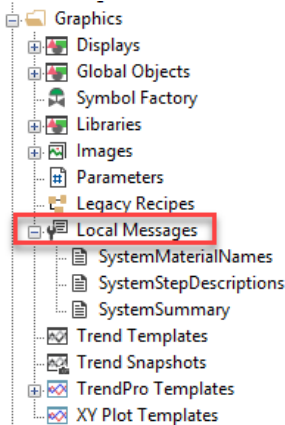


Step Descriptions: HMI local message displays are used to display additional EM Current Step information on the EM faceplate. Use the EM.PSet_StepDesc to display messages. The PSet program command functions regardless of the mode, which means it can be used even if the EM is in operator or maintenance mode. It is also independent of the state or step values.

Summary Descriptions: Local message displays are used to display additional EP Summary information on the EM faceplate. You can use the EM.PSet.Summary to display a custom message. Similar to the Step Description, EM.PSet.Summary functions even if the EM is in operator or maintenance mode.



HMI Local Messages (legacy method): The default SystemStepDescriptions local message is a legacy method that provides backwards compatibility. When you use this function, the faceplate references the local message object. The controller shortcut path for this function is Global parameter #3.



SystemStepDescriptions - /LCS_SAMPLE/HMI (Local Messages)		
	Trigger Value	Message
1	1	/*S:0 {#3System.Enum.Step_Desc[1].@Description}*/
2	2	/*S:0 {#3System.Enum.Step_Desc[2].@Description}*/
3	3	/*S:0 {#3System.Enum.Step_Desc[3].@Description}*/
4	4	/*S:0 {#3System.Enum.Step_Desc[4].@Description}*/
5	5	/*S:0 {#3System.Enum.Step_Desc[5].@Description}*/
6	6	/*S:0 {#3System.Enum.Step_Desc[6].@Description}*/
7	7	/*S:0 {#3System.Enum.Step_Desc[7].@Description}*/
8	8	/*S:0 {#3System.Enum.Step_Desc[8].@Description}*/
9	9	/*S:0 {#3System.Enum.Step_Desc[9].@Description}*/
10	10	/*S:0 {#3System.Enum.Step_Desc[10].@Description}*/
11	11	/*S:0 {#3System.Enum.Step_Desc[11].@Description}*/
12	12	/*S:0 {#3System.Enum.Step_Desc[12].@Description}*/
13	13	/*S:0 {#3System.Enum.Step_Desc[13].@Description}*/
14	14	/*S:0 {#3System.Enum.Step_Desc[14].@Description}*/
15	15	/*S:0 {#3System.Enum.Step_Desc[15].@Description}*/
16	16	/*S:0 {#3System.Enum.Step_Desc[16].@Description}*/
17	17	/*S:0 {#3System.Enum.Step_Desc[17].@Description}*/
18	18	/*S:0 {#3System.Enum.Step_Desc[18].@Description}*/
19	19	/*S:0 {#3System.Enum.Step_Desc[19].@Description}*/
20	20	/*S:0 {#3System.Enum.Step_Desc[20].@Description}*/
21	21	/*S:0 {#3System.Enum.Step_Desc[21].@Description}*/
22	22	/*S:0 {#3System.Enum.Step_Desc[22].@Description}*/
23	23	/*S:0 {#3System.Enum.Step_Desc[23].@Description}*/
24	24	/*S:0 {#3System.Enum.Step_Desc[24].@Description}*/
25	25	/*S:0 {#3System.Enum.Step_Desc[25].@Description}*/
26	26	/*S:0 {#3System.Enum.Step_Desc[26].@Description}*/
27	27	/*S:0 {#3System.Enum.Step_Desc[27].@Description}*/
28	28	/*S:0 {#3System.Enum.Step_Desc[28].@Description}*/
29	29	/*S:0 {#3System.Enum.Step_Desc[29].@Description}*/

The System tag is controller-scoped and can be used for the EMs in the controller.

Scope:	VaSEQ_SBC	Show:	All Tags		T_system
Name	Force Mask	Style	Data Type	Description	
System.Enum.Step_Desc		(...)	Decimal	System Global Structure Step Descriptions	
System.Enum.Step_Desc[0]			Decimal	Step Description 0	
System.Enum.Step_Desc[1]			Decimal	Step Description 1	
System.Enum.Step_Desc[2]			Decimal	Step Description 2	
System.Enum.Step_Desc[3]			Decimal	System Global Structure Step Descriptions	
System.Enum.Step_Desc[4]			Decimal	System Global Structure Step Descriptions	
System.Enum.Step_Desc[5]			Decimal	System Global Structure Step Descriptions	
System.Enum.Step_Desc[6]			Decimal	System Global Structure Step Descriptions	

HMI Local Messages (most current method): This is the preferred method as it provides the most flexibility and can be used to create class-based local message files for each class/type.

As part of the EM tag structure, the Sts_eSummary and Sts_eStep extended tag properties for the Navigation field to define which local messages are used.

In the following example, the step descriptions use 'Template_EM_Steps' and the summary displays 'Template_EM_Summary.' In this scenario, no other controller tags or fields are required

EM.Sts_eStep	1	Navigation	Template_EM_Steps
EM.Sts_eSummary	0	Navigation	Template_EM_Summary

You can copy and paste local messages from and into a spreadsheet or notepad application. This capability allows you to make bulk edits and/or updates to your system.

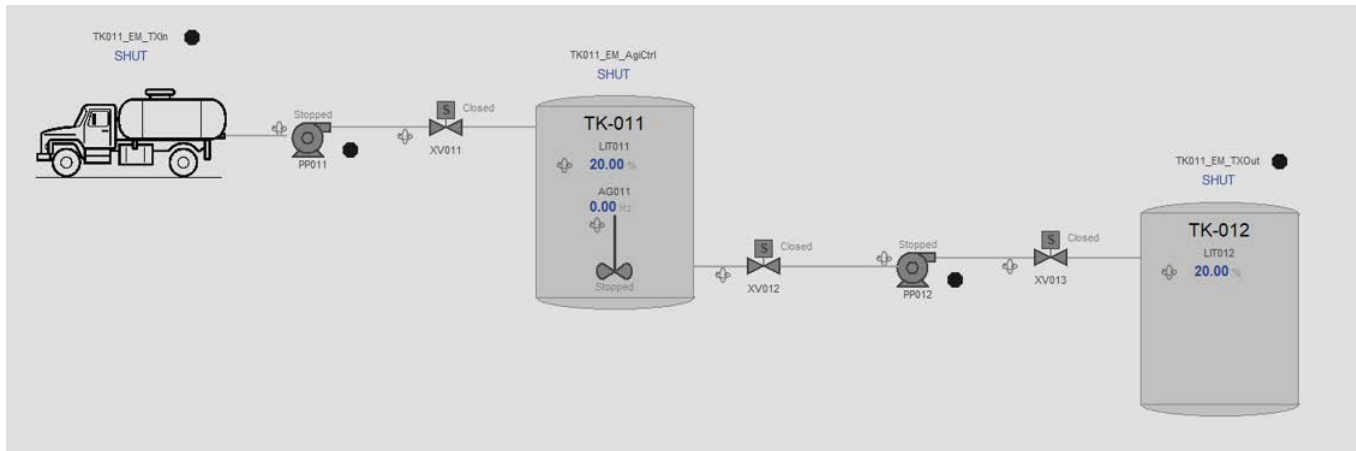
Template_EM_Steps - /LCS_SAMPLE/HMI (Local Messages)		
	Trigger Value	Message
1	1	EM Template Step1 Message
2	2	EM Template Step2 Message
3	3	EM Template Step3 Message
4	4	EM Template Step4 Message
5	105	EM Template Step105 Message

Template_EM_Summary - /LCS_SAMPLE/HMI (Local Messages)		
	Trigger Value	Message
1	1	EM Template Summary Message 1
2	2	EM Template Summary Message 2
3	3	EM Template Summary Message 3
4	104	EM Template Summary Message 4
5		

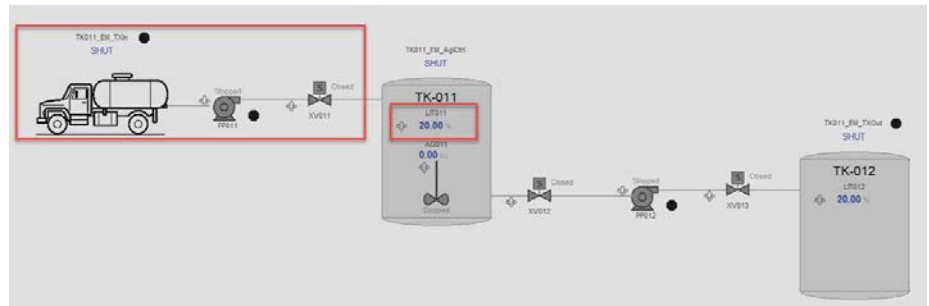
Use Case Examples

In the Equipment Module (EM) example system, liquid is unloaded from a truck into a mixing tank, labeled TK-011. In TK-011, the liquid is agitated and then dispensed into a holding tank, labeled TK-012. The system is split into three equipment phases:

1. TK011_EM_TXIn commands the objects that dispense liquid into the mixing tank.
2. TK011_EM_AgiCtrl commands objects that agitate the liquid and monitor liquid level inside the tank.
3. TK011_EM_TXOut commands objects that transfer the liquid from the mixing tank to the holding tank.



Equipment Module One - TK011_EM_TXIn



There are two states for TK011_EM_TXIn:

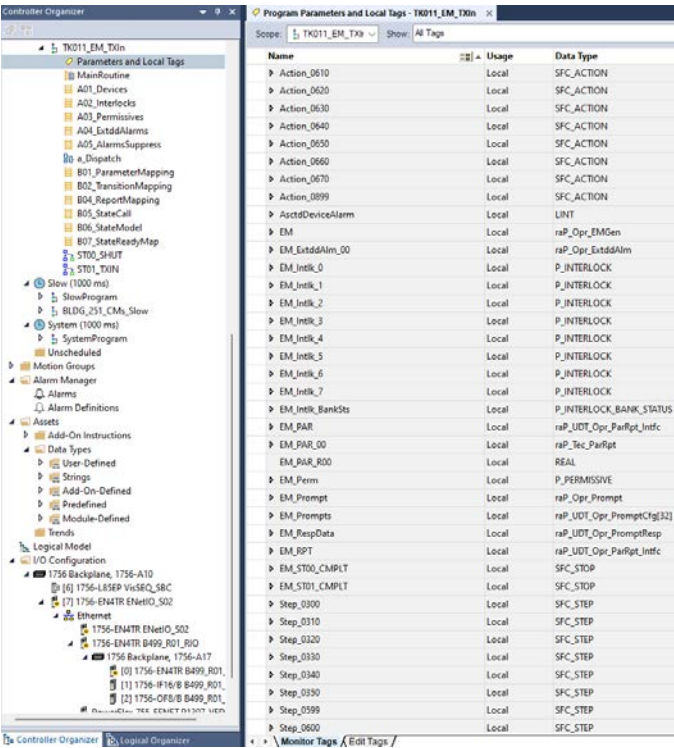
1. SHUT
2. TXIN

State	Program the Logic as Follows
SHUT	<ol style="list-style-type: none"> 1. When commanded into the SHUT state, pump PP011 is commanded to stop. 2. When pump PP011 has stopped, inlet valve XV011 is commanded to close. 3. If there are any values to report when inlet valve XV011 is closed, a report capture is requested. 4. When the report capture is complete, the EM ownership is released. 5. When ownership is released, the EM state is complete.
TXIM	<ol style="list-style-type: none"> 1. When commanded into the TXIn state, ownership of the EM is requested. 2. When ownership is acquired, the high limit of the level indicator LIT001 is set with one of the EM parameters. 3. The parameters are captured. 4. When the LIT001 high limit is set and the parameters are captured, command inlet valve XV011 to open. 5. When inlet valve XV011 is open, command pump PP011 to start. 6. When pump PP011 is running, the TXIN state is complete.

Routine Structure - TK011_EM_TXIn

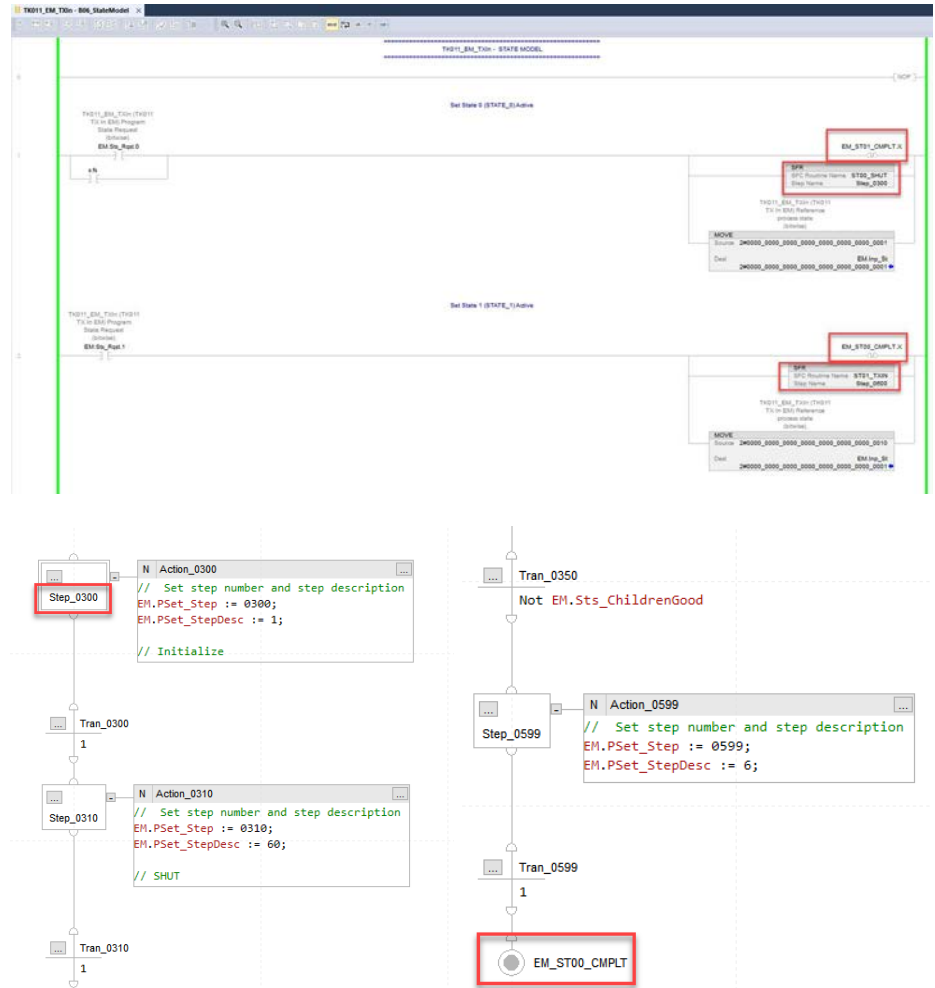
In the following example, the letter-number combination prefix on the routine name helps organize the list of routines.

Most tags that are used with the EM are Parameter and Local Tags and are not controller-scoped. This lets you create reusable code and you do not need to change tag names for each instance.

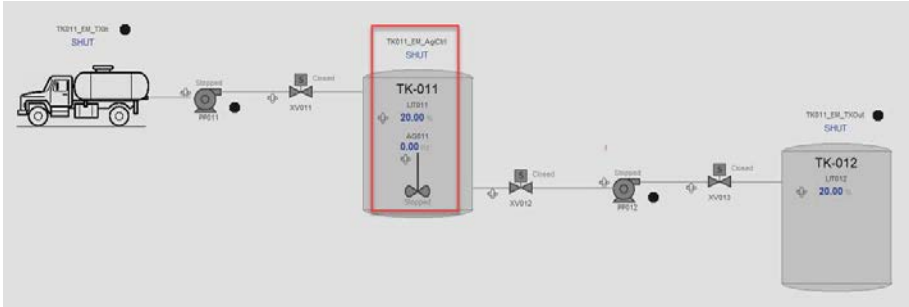


StateModel - TK011_EM_TXIn

The first instruction in the branch unlatches the SFC Stop instruction complete parameter. The Stop complete parameter is typically set after the last step in the opposing State SFC logic. The SFR calls the SFC and specifies which step begins executing, which is typically the first SFC Step instruction in the state routine. The first scan parameter S:FS initializes the EM to the SHUT state after a download or when switching into controller run mode.



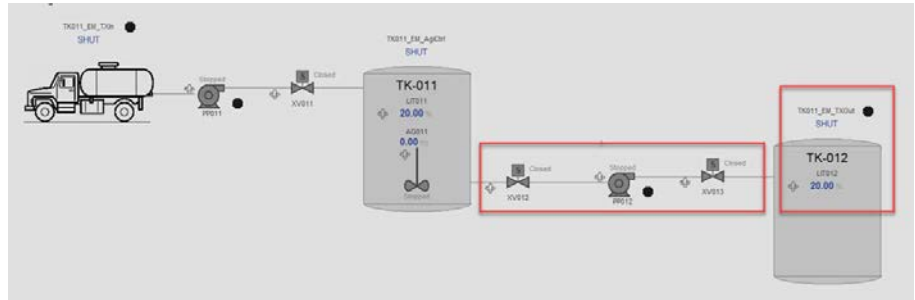
Equipment Module Two - TK011_EM_AgiCtrl



There are two states for TK011_EM_AgiCtrl: SHUT or AGITATE.

State	Program the Logic as Follows
SHUT	<div>1. When commanded into the SHUT state, command agitator AG011 to stop.</div> <div>2. When agitator AG011 has stopped, capture the report parameters.</div> <div>3. When the report parameters are captured, command the EM to be released.</div> <div>4. When the EM is released, this EM state is complete.</div>
AGITATE	<div>1. When commanded into the TXIN state, the program acquires the EM.</div> <div>2. When the EM is owned, capture the parameter values.</div> <div>3. When the capture is complete, trigger the on-delay timer for agitator starts and stops.</div> <div>4. The EM state is set to complete.</div> <div>5. If the level indicator LIT011 is greater than the set parameter value, the agitator stops, the EM goes into the monitoring step, and the on-delay completes.</div> <div>6. The agitator AG011 is commanded to start with a speed set by the EM parameters, via programming.</div> <div>7. When agitator AG011 runs forward, the EM resumes monitoring the state of the agitator.</div> <div>8. If the level indicator LIT011 is lower than the value set by the EM parameters, agitator AG011 runs forward.</div> <div>9. The EM goes into the monitoring step and the on-delay timer for the transition completes.</div> <div>10. Command the agitator AG011 to stop.</div> <div>11. Once the agitator has stopped, the EM continues to monitor the state until it is commanded into the SHUT state.</div>

Equipment Module Three - TK011_EM_TXOut



There are two states for TK011_EM_TXOut:

1. SHUT
2. TXOUT

State	Program the Logic as Follows
SHUT	<ol style="list-style-type: none"> 1. When commanded into the SHUT state, command pump PP012 to stop. 2. When pump PP012 is stopped, command Tank 012 inlet valve XV013 to close. 3. When inlet valve XV013 is closed, command Tank 011 outlet valve XV012 to close. 4. When outlet valve XV012 is closed, capture report parameters. 5. When the report parameters are captured, command the ownership of EM to release. 6. When the ownership is released, the EM state is complete.
TXOUT	<ol style="list-style-type: none"> 1. When commanded into the TXOUT state, ownership of the EM is requested. 2. When ownership is acquired, capture report parameters and set the Tank 012 level indicator LIT002 to the high limit. 3. When the parameter values are captured and the level indicator LIT002 high limit is set to the correct EM parameter value, command Tank 011 outlet valve XV012 to open. 4. When outlet valve XV012 is open, command Tank 012 inlet valve to open. 5. When the Tank012 inlet valve is open, command pump PP012 to start. 6. When pump PP012 is running, the TXOUT EM status is complete.

Notes:

Additional Configuration

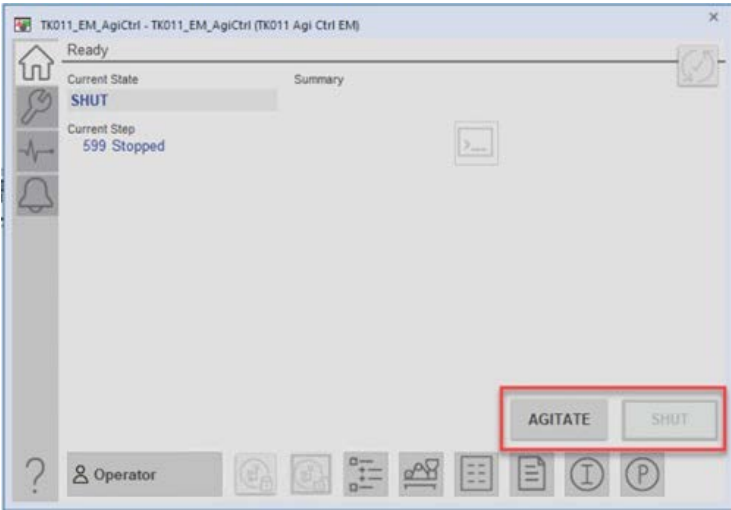
Equipment Module
State Management

To configure an EM state, create a routine for each desired state. Each routine contains the code that commands objects, captures parameters, and captures reports. However, unlike the ISA-88 phase state model, EM routines do not have an inherent module state model. When you develop an EM state model, write the code to reflect the model.

To enable a module state button on the HMI faceplate, create the descriptions within the EM.XCmd parameter. Then make each parameter correspond to a module state button. Set the text displayed under the 'Current State' with descriptors within the EM.Sts parameter:

EM.XCmd	Binary	DINT	TK011_EM_TXIn (TK011 TX in EM) External command to State (bitwise).
EM.XCmd.0	Decimal	BOOL	SHUT
EM.XCmd.1	Decimal	BOOL	TXIN

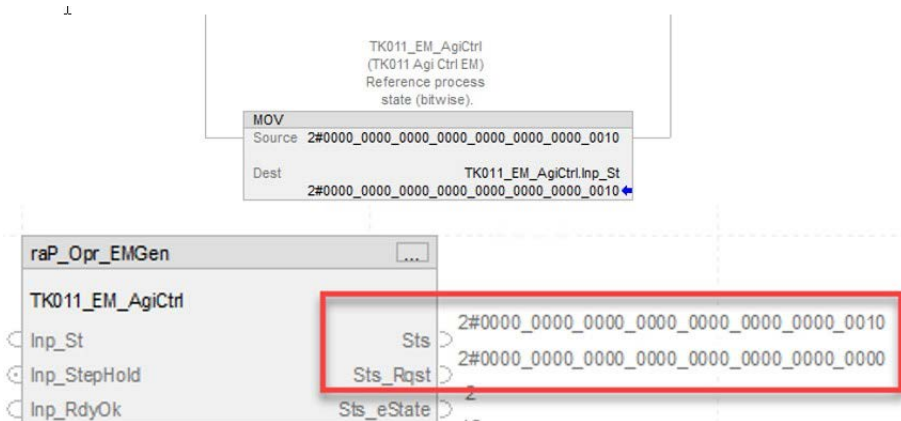
EM.Sts	Binary	DINT	TK011_EM_TXIn (TK011 TX in EM) State Status (bitwise).
EM.Sts.0	Decimal	BOOL	SHUT
EM.Sts.1	Decimal	BOOL	TXIN



To command the raP_Opr_EMGen block into the correct state, modify the bitwise request Inp_St parameter. To modify this request within the code, issue a PCmd to the corresponding state parameter. Or you can also click the corresponding state button, which then issues an OCmd on a specific parameter within the HMI. This input request sets the corresponding Sts_Rqst parameter to the corresponding bitwise state but does not change the state of the EM. To transfer the state of the EM, you must create additional code. As seen in the following example, a command is issued to transfer the EM to module state 1. Without additional code, the system is stuck in module state 0:



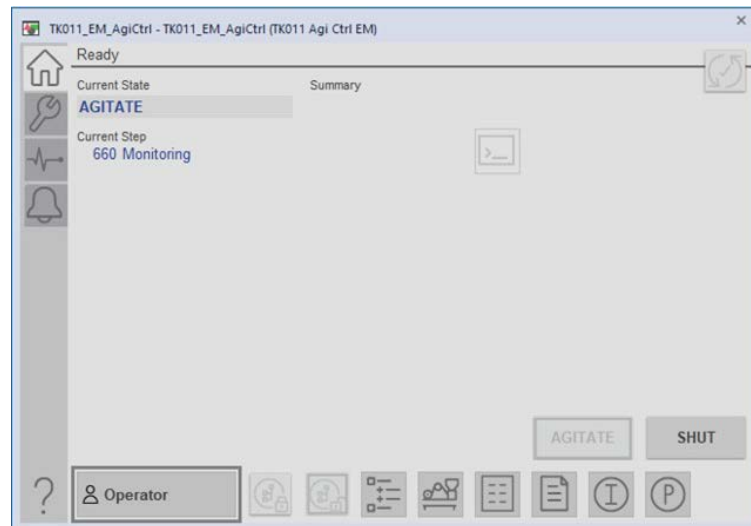
To set the EM to the requested module state, associate the Sts_Rqst parameter with the Inp_St code — for example, through a MOV block. Move the requested state parameter into the Inp_St parameter. When you move into this parameter, the Sts parameter is set to the requested state and the EM is put into the required state. The requests are cleared when the Sts parameter matches the Sts_Rqst parameter.



When you set the Sts parameter, you must implement additional code to execute the module state routine. One way you can control EM code execution is to use the individual Sts parameters to command the module to execute specified state routines.



With the EM state model configuration, you can define which buttons are visible on the faceplate via the `Inp_RdyOk` parameters. When set to 1, the corresponding `Inp_RdyOk` parameter lets you change the EM to this state. This then executes the phase state code for that state. When you set the `Inp_RdyOk` parameter to 0, you cannot execute the code of that state and the button is not visible.



If `Inp_RdyOk.0` is set to 1, you can transition it to state 0 (Shut).
If `Inp_RdyOk.0` is set to 0, you cannot transition it to state 1 (Agitate).

Parameters and Reports

Parameters

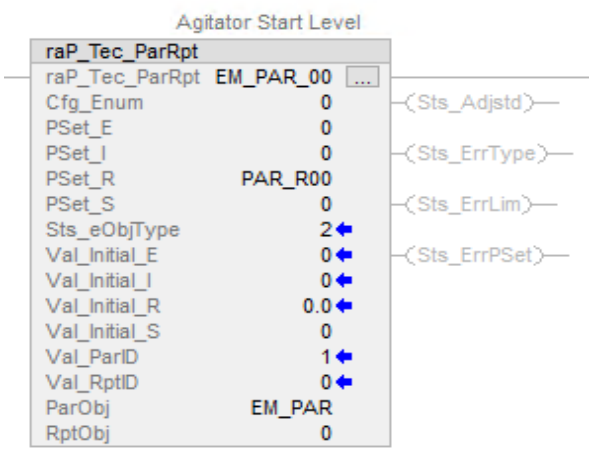
The Operating and Engineering parameters are values that determine the functionality of the EM/EP. These parameters contain attributes for the system, including the setpoint of a dispensed ingredient, the level of a tank, the amount of time to agitate ingredients in a tank.

To configure `raP_Tec_ParRpt` as a parameter, create a parameter object tag and associate it with the EM/ EP block. In the following example, the parameter object tag is `EM_PAR`. Create an instance of the `raP_Tec_ParRpt` block for each parameter. For each instance of the `raP_Tec_ParRpt` block you create, you must associate it with the same parameter object tag. To indicate that the instance of an `raP_Tec_ParRpt` is for a parameter and not a report, place a 0 in the report object tag location.

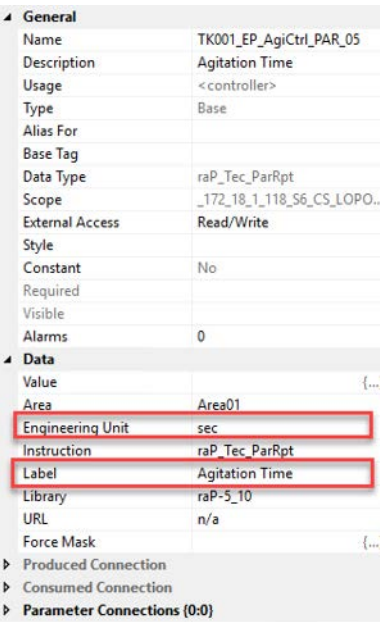
You can use a parameter for an enumeration, integer, real, or string. To configure the value of the parameter, create a tag of the same data type and associate it with the corresponding `PSet` parameter. You can only set a parameter as one of these four data types:

- Enumerations are pinned to `PSet_E`
 - The enumeration set is a string array that is configured in the `Cfg_Enum` parameter. The string value in the array will correspond to the value of `Pset_E`.
- Integer tags are pinned to `PSet_I`
- Real tags are set to `PSet_R`
- String tags are pinned to `PSet_S`

All other data types must have a 0 put in its place. These data type connections can only be set offline.

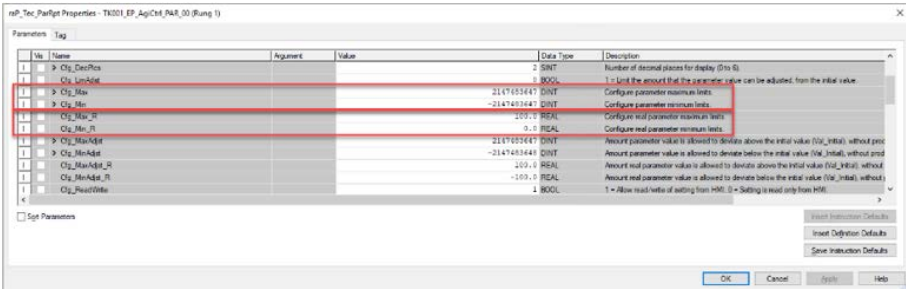


Set the label and engineering units in the extended tag properties of each parameter tag:



The configured minimum and maximum values that are used in the raP_Tec_ParRpt block will depend on the parameter data type.

- If the parameter is an integer, Cfg_Max and Cfg_Min will determine the maximum and minimum values.
- If the parameter is real, Cfg_Max_R and Cfg_Min_R will determine the maximum and minimum values.



To set the parameter values for a run of the phase and log any changes to the values of the parameters that are made since the last run, create a parameter capture in the sequence code for the EP or EM.

```
TK001_EP_AgiCtrl.PCmd_ParCapture := 1;
```

If you update any values after the capture, the previous value is used until the phase sequence runs again. You can update values via the faceplate or within the code by modifying the tag value:

Parameter Description	Value	Snapshot	Default
Agitator Start Level	25.00 %	20.00	20.00

Reports

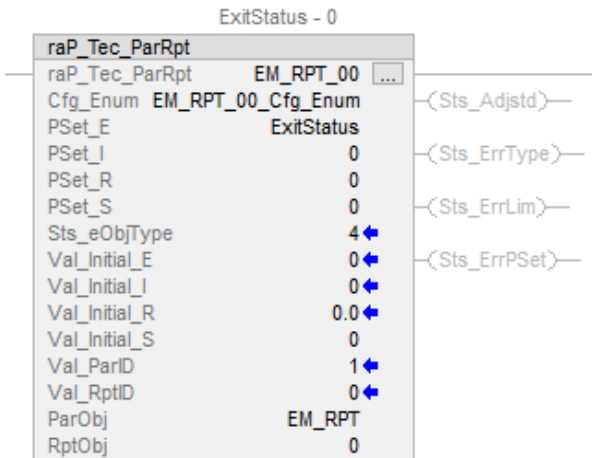
Reports contain key EM/EP values that provide information about a specific phase or module run. Reports include – but are not limited to – actual mass of the product that is dispensed, actual motor speeds, and EM/EP status. Unlike parameters, which are often used to define a system setpoint, reports usually measure the results of a phase or module run.

To configure raP_Tec_ParRpt as a report, create a report object tag and associate it with the EM/ EP block. In the following example, the report object tag is EM_RPT. Create an instance of the raP_Tec_ParRpt block for each report. For each instance of the raP_Tec_ParRpt block you create, you must associate it with the same report object tag. To indicate that the instance of an raP_Tec_ParRpt is for a report and not a parameter, place a 0 in the parameter object tag location.

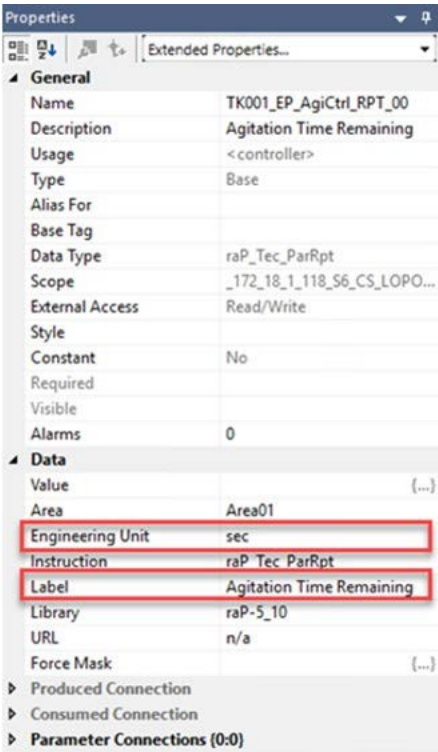
You can use a report as an enumeration, integer, real, or string. To configure the value of a report, create and associate a tag of the same data type with the corresponding PSet parameter:

- Enumerations are pinned to PSet_E
 - The enumeration set is a string array that is configured in the Cfg_Enum parameter. The string value in the array will correspond to the value of Pset.E.
- Integer tags are pinned to PSet_I
- Real tags are set to PSet_R
- String tags are pinned to PSet_S

All other data types must have a 0 put in its place. These data type connections can only be set offline.

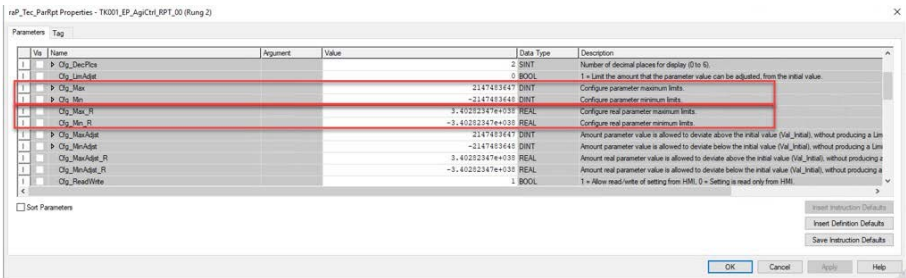


Set label and engineering units in the extended tag properties of each parameter.



The configured minimum and maximum values that are used in the raP_Tec_ParRpt block will depend on the report data type.

- If the report is an integer, Cfg_Max and Cfg_Min will determine the maximum and minimum values.
- If the report is real, Cfg_Max_R and Cfg_Min_R will determine the maximum and minimum values.



To set the parameter values for a run of the phase and log any changes to the values of the parameters that are made since the last run, include a parameter capture in the sequence code for the EP.

```
TK001_EP_AgiCtrl.PCcmd_RptCapture := 1;
```

If you update any values after the capture, the previous value is used until the phase sequence is run again.

Extended Alarms

With PlantPAX® Bus object extended alarms, you can propagate changes in the organizational tree. Extended alarms are generic, which means you can configure them as needed for your specific system. The only requirement for an extended alarm is that you must specify if you want an alarm to trigger any parent objects alarms in the organizational tree – such as to the unit or area.

To configure extended alarms for an EM/EP, perform the following steps.

1. Create an instance of a raP_Opr_ExtddAlm per extended alarm.
2. When you name the extended alarm, use an underscore and two digits at the end of the alarm name. Then increment each alarm that you extend by a value of 1. For example:
 - EM_ExtddAlm_00
 - EM_ExtddAlm_01

Insert Extended Message Here

raP_Opr_ExtddAlm	EM_ExtddAlm_00
PCmd_Reset	EM.Out_ExtddAlmsReset
	0
PCmd_ResetAckAll	EM.Out_ExtddAlmsResetAckAll
	0
Out_ExtddAlmDsply	EM.Inp_ExtddAlmsDsply.0
	0
Used	EM.Inp_ExtddAlmsUsed.0
	1
Alm	EM.Inp_ExtddAlmsAlm.0
	0
Acked	EM.Inp_ExtddAlmsAcked.0
	1
Disabled	EM.Inp_ExtddAlmsDisabled.0
	0
Shelved	EM.Inp_ExtddAlmsShelved.0
	0
Suppressed	EM.Inp_ExtddAlmsSuppressed.0
	0
AlarmFault	EM.Inp_ExtddAlmsAlarmFault.0
	0
Sts_Alminh	EM.Inp_ExtddAlmsAlminh.0
	0
Sts_RdyReset	EM.Inp_ExtddAlmsRdyReset.0
	0
Inp_ExtddAlmeNotify	EM.Inp_ExtddAlmeNotify
Inp_ExtddAlmDsply	EM.Inp_ExtddAlmsDsply



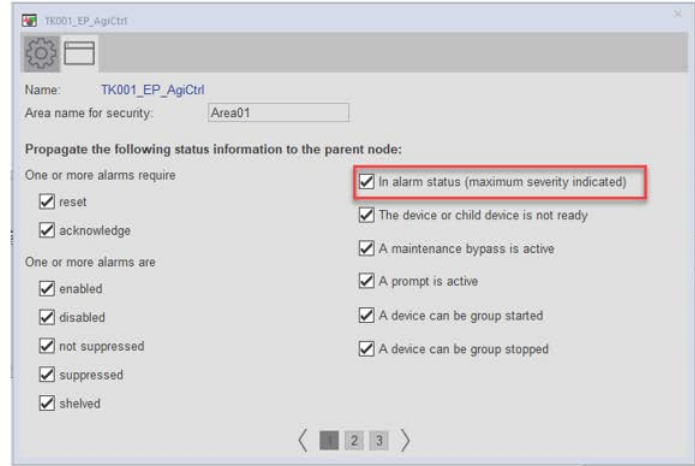
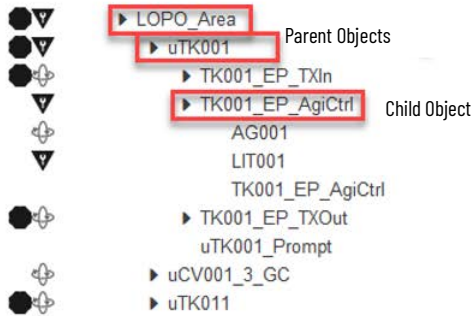
Extended alarms can be used even if the Bus is not used.

3. Associate the trigger condition with the Inp parameter of the associated extended alarm tag.
4. Enable the extended alarm tag for the EM/EP in the Alarm Manager.
5. Enable each level of propagation that you wish the alarm to go through, including area and unit tags.
6. Configure the organizational tree of child and parent objects with the bus object.

Insert extended Message Here: 1 = alarm trigger

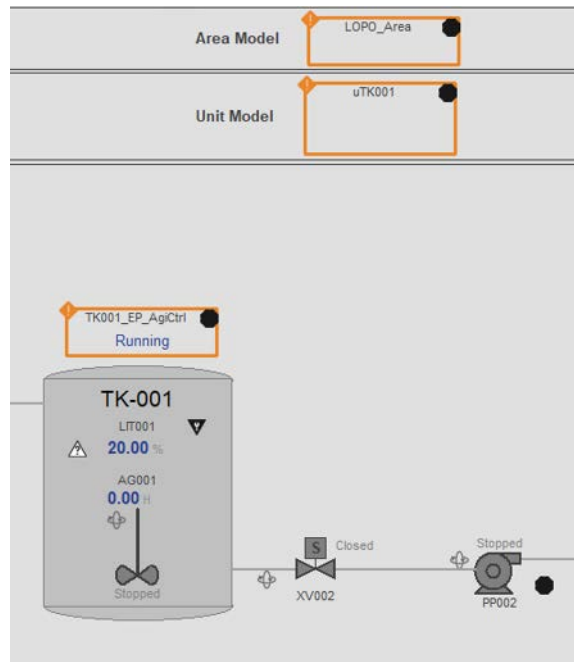
Insert, Trigger, Name	Alarm, Trigger, Name	Alarm, Trigger, Name	Alarm, Trigger, Name	Alarm, Trigger, Name	Alarm, Trigger, Name	Alarm, Trigger, Name	Alarm, Trigger, Name
1	170001 EP_AlgCvt	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s
2	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s
3	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s
4	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s
5	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s
6	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s
7	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s
8	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s
9	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s
10	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s
11	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s
12	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s	Alarm, Alarm	17001	170001 EP_AlgCvt, 00.00s

7. For each level of desired propagation, enable alarm status propagation on each parent object:
 - a. Identify each desired parent node that needs to go into alarm upon a child alarm
 - b. Enable alarm propagation



8. When you perform these steps tests, you can test if a child alarm triggers parent objects.

This example uses the extended alarm TK001_EP_AgiCtrl.



Operator Prompts

Operator prompts are universal mechanisms that interact with the `raP_Opr_Prompt` block in the code. You can configure operator prompts within a control scheme to provide system operators with the ability to interact with custom messages and data fields and use a prompt to request input. You can customize the following inputs:

- Acknowledge prompts
- View and confirm data
- Make selections
- Enter numeric data
- Enter text data

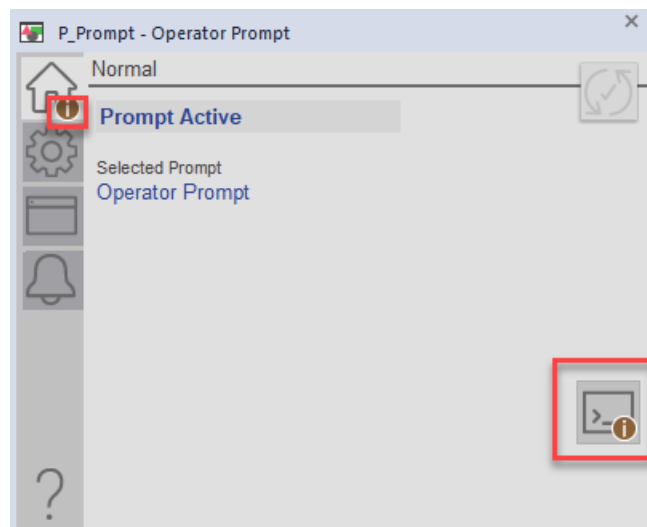


- The `raP_Opr_Prompt` Add-On Instruction does not use command sources.
- The `raP_Opr_Prompt` Instruction has no Virtualization capability.

You can only configure the **Go_Prompt** graphic on the screen:



When operator input is required, an information icon is displayed. To respond to the prompts, click the **Go_prompt** object. You can then configure prompts in the configuration tab.



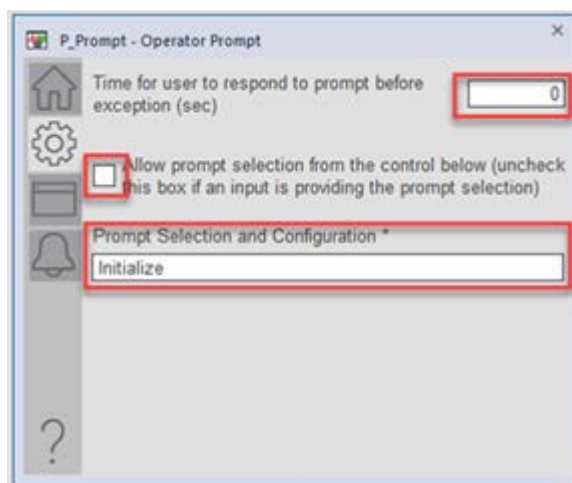
These are the associated logic Instructions.



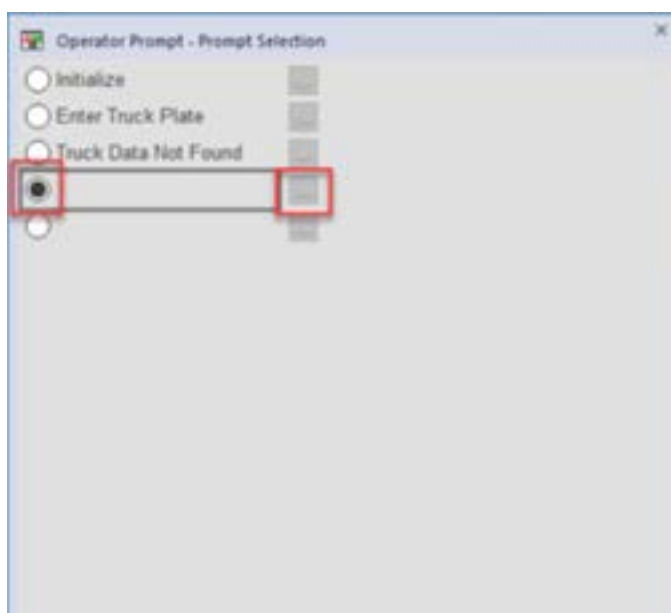
Configuration From Faceplate

To configure the maximum amount of time that can pass without a response before an exception occurs, use the configuration tab on the faceplate and set the value in seconds.

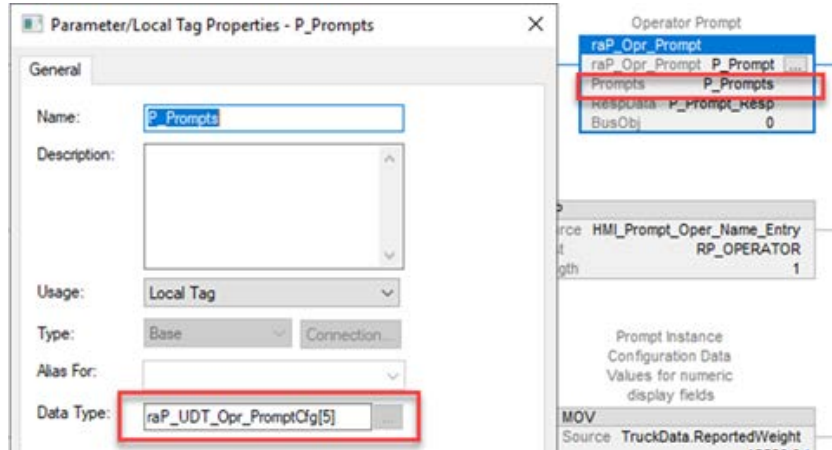
If you want system operators to be able to navigate prompt selections and configure displays, set the 'Prompt Selection and Configuration' to 'Initialize.'



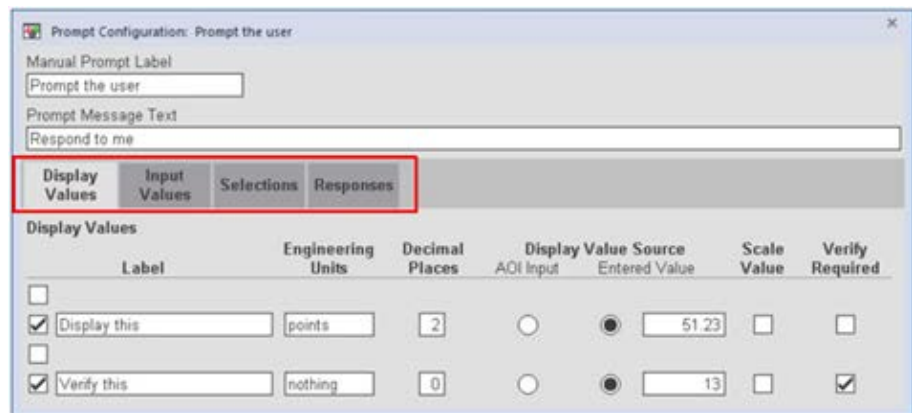
To access the prompt configuration tab, click Prompt Selection and Configuration to display the Prompt Selection tab. Select an option and then click the ellipses (...) to make changes.



To configure the number of prompts required, edit the Prompt tag of the raP_Opr_Prompt object.

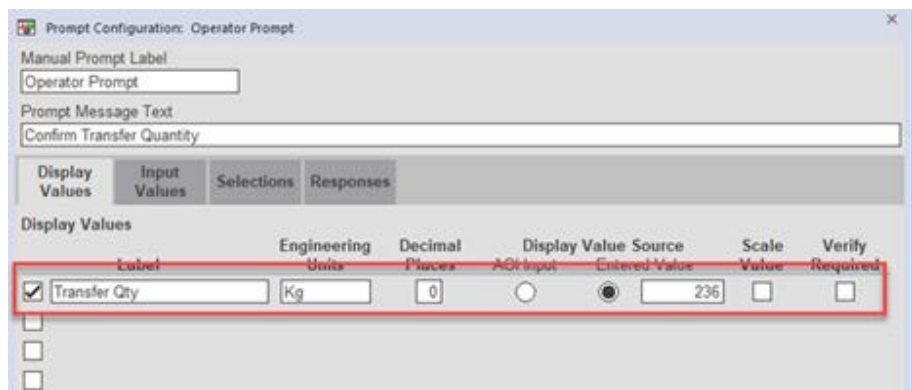


The Prompt Configuration dialog box has four sections to configure a prompt: Display Values, Input Values, Selections, and Responses. You can create a manual prompt label and prompt message text for each section.

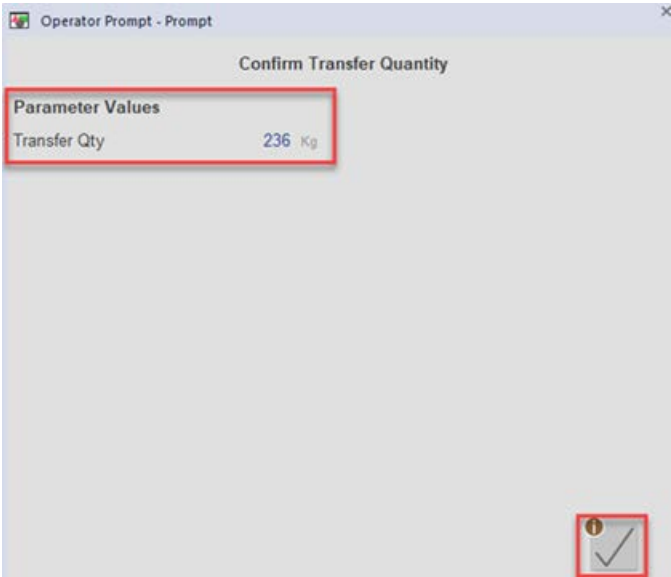


Display Values

If you need to display a numeric value on a prompt, configure the display value section. To configure a numeric display, enter the required information in the display value fields.



The values that you set are displayed when a prompt is triggered, and require acknowledgment to proceed.



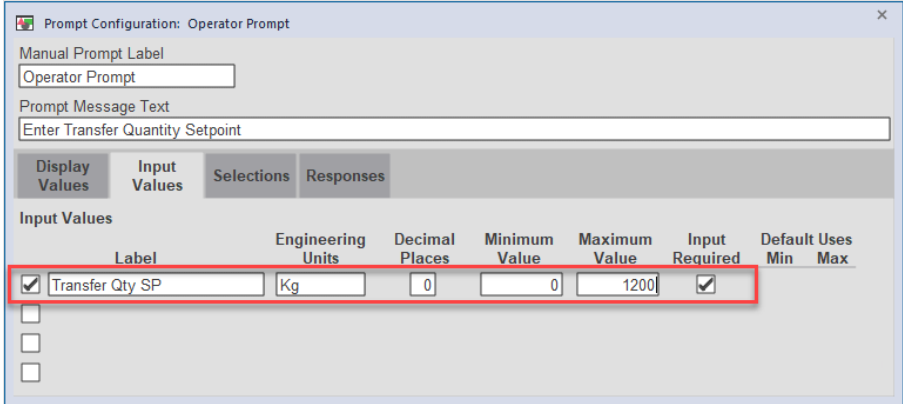
Display values are linked to the Cfg_DispVal[x] parameters of the raP_Opr_Prompt object input tag which has a data type of raP_UDt_Opr_PromptCfg. Each display value corresponds to a respective prompt and display value array.

▲ P_Prompts[4]		<input type="checkbox"/>	(...)	(...)	raP_UDt_Opr_PromptCfg
▶ P_Prompts[4].Cfg_Message		<input type="checkbox"/>	'Confirm Transfer Qu...	(...)	STRING
▶ P_Prompts[4].Cfg_Label		<input type="checkbox"/>	'Operator Prompt'	(...)	STRING_20
▶ P_Prompts[4].Cfg_InpValLabel		<input type="checkbox"/>	(...)	(...)	STRING_20[4]
▶ P_Prompts[4].Cfg_DispValLabel		<input type="checkbox"/>	(...)	(...)	STRING_20[4]
▶ P_Prompts[4].Cfg_SelectLabel		<input type="checkbox"/>	(...)	(...)	STRING_20[4]
▶ P_Prompts[4].Cfg_RespLabel		<input type="checkbox"/>	(...)	(...)	STRING_20[4]
▶ P_Prompts[4].Cfg_DispValEU		<input type="checkbox"/>	(...)	(...)	STRING_8[4]
▶ P_Prompts[4].Cfg_InpValEU		<input type="checkbox"/>	(...)	(...)	STRING_8[4]
▲ P_Prompts[4].Cfg_DispVal		<input type="checkbox"/>	(...)	(...) Float	REAL[4]
▶ P_Prompts[4].Cfg_DispVal[0]		<input type="checkbox"/>	236.0	Float	REAL
▶ P_Prompts[4].Cfg_DispVal[1]		<input type="checkbox"/>	0.0	Float	REAL
▶ P_Prompts[4].Cfg_DispVal[2]		<input type="checkbox"/>	0.0	Float	REAL
▶ P_Prompts[4].Cfg_DispVal[3]		<input type="checkbox"/>	0.0	Float	REAL

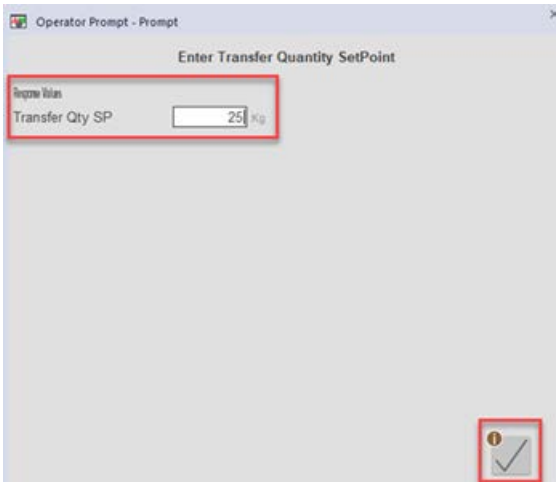
Input Values

If you need to enter a numeric value during the process to proceed to the next step of operation, configure the input value section.

To configure input values, select the input value checkbox and enter the required information such as Label, Engineering Units, Decimal places, and Minimum Value, and Maximum Value.



The operator prompt value defines what input value is needed to acknowledge the prompt and proceed.

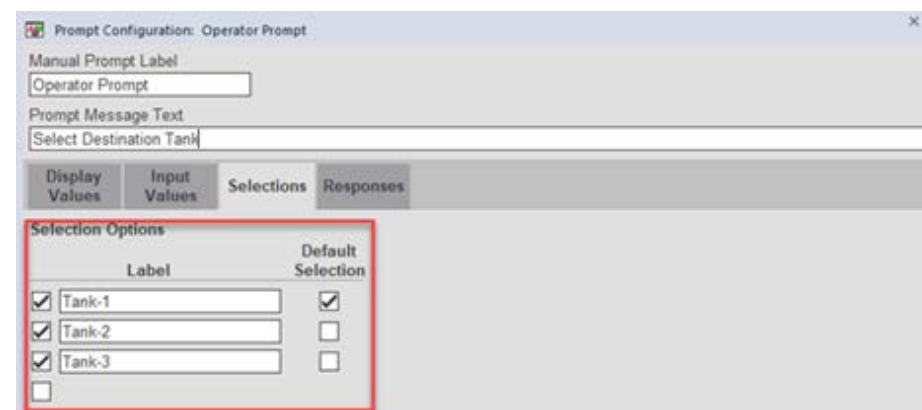


The values that you enter during the operation are logged in the RespData tag of raP_Opr_Prompt object.

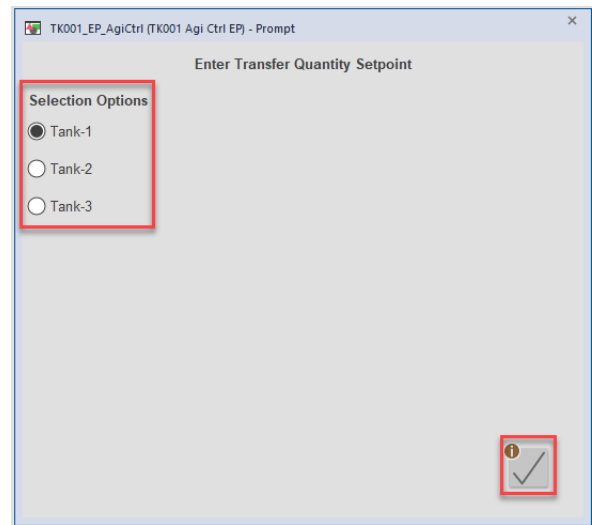
▲ P_Prompt_Resp	Local	<input type="checkbox"/>	[...]	[...]	raP_UDT_Opr_PromptResp
▲ P_Prompt_Resp.Selection		<input type="checkbox"/>	--	[...]	STRING_20
▶ P_Prompt_Resp.Selection.LEN		<input type="checkbox"/>	0	Decimal	DINT
▶ P_Prompt_Resp.Selection.DATA		<input type="checkbox"/>	[...]	[...] ASCII	SINT[20]
▲ P_Prompt_Resp.NumericInput		<input type="checkbox"/>	[...]	[...]	STRING_16[4]
▶ P_Prompt_Resp.NumericInput[0]		<input type="checkbox"/>	25.36	[...]	STRING_16
▶ P_Prompt_Resp.NumericInput[1]		<input type="checkbox"/>	--	[...]	STRING_16
▶ P_Prompt_Resp.NumericInput[2]		<input type="checkbox"/>	--	[...]	STRING_16
▶ P_Prompt_Resp.NumericInput[3]		<input type="checkbox"/>	--	[...]	STRING_16
▶ P_Prompt_Resp.CharInput		<input type="checkbox"/>	[...]	[...]	STRING 40[4]

Selection Tab

You can configure as many as four prompts on the selection tab.



When the operator prompt is triggered during the process, you must select any one of the predefined selection options.

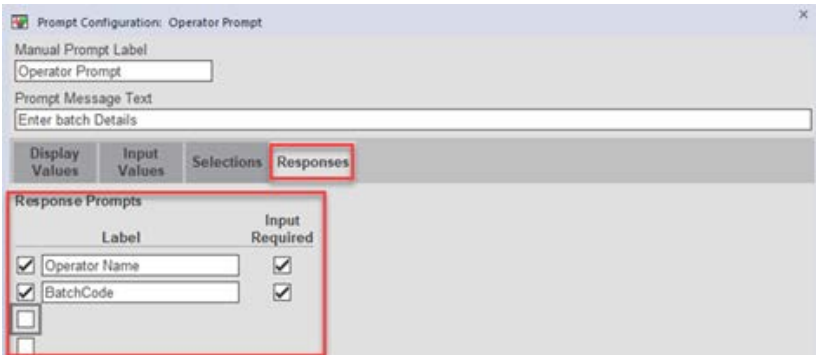


The input that you select is logged in the RespData input tag of raP_Opr_Prompt object.

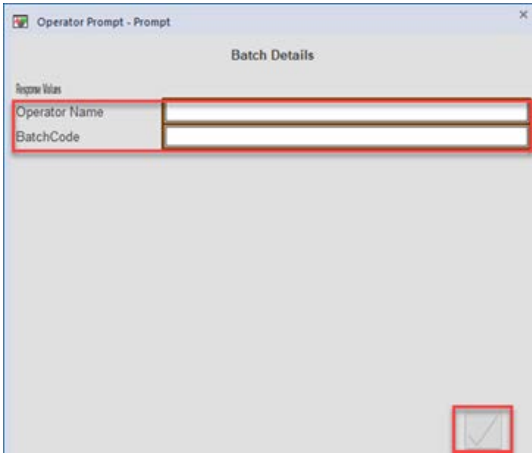
P_Prompt_Resp	Local					raP_UDT_Opr_PromptResp
P_Prompt_Resp.Selection				Tank-1		STRING_20
P_Prompt_Resp.Selection.LEN				8	Decimal	DINT
P_Prompt_Resp.Selection.DATA					ASCII	SINT[20]
P_Prompt_Resp.NumericInput						STRING_16[4]
P_Prompt_Resp.CharacterInput						STRING_40[4]

Responses Tab

Configure required responses on the Responses tab.



When the operator prompt is triggered, the operator must enter the details that you configure.



The selected input that you enter during the operation is logged in the RespData tag of raP_Opr_Prompt object.

P_Prompt_Resp	Local			(...)	(...)	raP_UOT_Opr_PromptResp
P_Prompt_Resp.Selection				--	(...)	STRING_20
P_Prompt_Resp.Selection.LEN				0	Decimal	DINT
P_Prompt_Resp.Selection.DATA				(...)	(...) ASCII	SINT[20]
P_Prompt_Resp.NumericInput				(...)	(...)	STRING_16[4]
P_Prompt_Resp.NumericInput[0]				--	(...)	STRING_16
P_Prompt_Resp.NumericInput[1]				--	(...)	STRING_16
P_Prompt_Resp.NumericInput[2]				--	(...)	STRING_16
P_Prompt_Resp.NumericInput[3]				--	(...)	STRING_16
P_Prompt_Resp.CharInput				(...)	(...)	STRING_40[4]
P_Prompt_Resp.CharInput[0]				'Operator-A'	(...)	STRING_40
P_Prompt_Resp.CharInput[1]				'Test Batch'	(...)	STRING_40
P_Prompt_Resp.CharInput[2]				--	(...)	STRING_40
P_Prompt_Resp.CharInput[3]				--	(...)	STRING_40

Notes:

Rockwell Automation Support

Use these resources to access support information.

Technical Support Center	Find help with how-to videos, FAQs, chat, user forums, Knowledgebase, and product notification updates.	rok.auto/support
Local Technical Support Phone Numbers	Locate the telephone number for your country.	rok.auto/phonesupport
Technical Documentation Center	Quickly access and download technical specifications, installation instructions, and user manuals.	rok.auto/techdocs
Literature Library	Find installation instructions, manuals, brochures, and technical data publications.	rok.auto/literature
Product Compatibility and Download Center (PCDC)	Download firmware, associated files (such as AOP, EDS, and DTM), and access product release notes.	rok.auto/pcdc

Documentation Feedback

Your comments help us serve your documentation needs better. If you have any suggestions on how to improve our content, complete the form at rok.auto/docfeedback.

Waste Electrical and Electronic Equipment (WEEE)



At the end of life, this equipment should be collected separately from any unsorted municipal waste.





Rockwell Automation maintains current product environmental compliance information on its website at rok.auto/pec.

Allen-Bradley, expanding human possibility, FactoryTalk, PhaseManager, PlantPAx, Rockwell Automation, and SequenceManager are trademarks of Rockwell Automation, Inc.

EtherNet/IP is a trademark of ODVA, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

Rockwell Otomasyon Ticaret A.Ş. Kar Plaza İş Merkezi E Blok Kat:6 34752, İçerenköy, İstanbul, Tel: +90 (216) 5698400 EEE Yönetmeliğine Uygundur

Connect with us.    

rockwellautomation.com — expanding human possibility®

AMERICAS: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000

EUROPE/MIDDLE EAST/AFRICA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2663 0600

ASIA PACIFIC: Rockwell Automation SEA Pte Ltd, 2 Corporation Road, #04-05, Main Lobby, Corporation Place, Singapore 618494, Tel: (65) 6510 6608

UNITED KINGDOM: Rockwell Automation Ltd., Pitfield, Kiln Farm, Milton Keynes, MK11 3DR, United Kingdom, Tel: (44)(1908) 838-800

Publication PROCES-UM110C-EN-P - November 2025

PROCES-UM110B-EN-P December 2024

Copyright © 2025 Rockwell Automation, Inc. All rights reserved. Printed in the U.S.A.