



# Interfacing Sepasoft Batch and Procedure Module with PLCs

## Version History

Version Number	Revision Date	Revision Description	Change By	Additional Notes
0.1	01/09/2023	Draft	Tom Hechtman	
0.2	03/08/2023	Peer review update	Tom Hechtman	
0.3	03/29/2023	Added Ignition UDT information	Tom Hechtman	
1.0	11/12/2025	Updated Command State diagram Added legend Added TOC	Natalie Calkins	

Contents

**Interfacing**

**Sepasoft Batch and Procedure Module**

**with PLCs..... 1**

**Version History..... 2**

**Introduction.....4**

**Basic Communication Between Batch Phase and Equipment Phase..... 5**

    Commands and States..... 6

**State Diagram.....7**

**Ignition UDTs.....10**

**Handshaking..... 15**

**ControlLogix Processors..... 16**

    Command Control Logic..... 18

    State Control Logic..... 19

**Batch Message Control Logic..... 20**

# Introduction

This document provides an overview of interfacing the Sepasoft Batch & Procedure module with PLCs. The ISA-88 standard defined this interface at a high level, which the Sepasoft Batch & Procedure module follows. Rockwell Automation has extended this interface in their ControlLogix processors beyond the ISA-88 standard, and this document will discuss what is required in the ControlLogix processor to support interfacing to the Sepasoft Batch & Procedure module.

The basic interface used to pass data back and forth between a batch engine (in this case, the Sepasoft Batch & Procedure module) and a PLC is known as PLI (Phase Logic Interface). This is the default method unless the PLC supports an alternate method and it is selected to be used.

It is important to understand the nomenclature and concepts that are defined in the ISA-88 standard. Processes are broken down into granular tasks that are generically called phases. On the batch engine side, these are technically called Batch Phases, and on the PLC side, they are Equipment Phases.

There is a one-to-one mapping of these phases. For example, if there is an Add H<sub>2</sub>O batch phase defined in the batch engine, then there is a matching Add H<sub>2</sub>O equipment phase implemented in the PLC. When implementing an equipment phase in a PLC where extended functionality is not supported or selected to be used, standard logic is used to handle the execution of equipment phases.

The matching batch phase and equipment phase communicate using tags. The structure and definition of these tags are based on PLI. Many default parameters are defined in the ISA-88 standard, while others are included by Sepasoft for common helpful functionality. The user (typically the engineer) can add additional parameters specific to the phase. For example, a Mix Time parameter can be added to specify the duration of the mixing step.

The Sepasoft Batch & Procedure module will automatically create tags in the Ignition platform for the appropriate parameters defined for a batch phase. These tags can be mapped to OPC devices or Ignition reference tags that are passed to the PLC. The primary parameters, and associated tags, that control the execution and monitoring of equipment phases are the Command, Command\_Number, State, and State\_Number. The Command will have string values, such as Start and Stop, that are human-readable. The Command\_Number will have a numeric value that is easier to handle in PLC logic.

# Basic Communication Between Batch Phase and Equipment Phase

There are several scenarios, and the following represents just the basic communication between the batch phase and the equipment phase during the execution of a batch recipe:

- System and user-defined parameter values defined in the recipe will be written to the tags by the batch engine. The PLC can read these values and are typically used in the equipment phase logic.
- The batch engine will set the Command to START and the Command\_Number to 1.
- The batch engine will wait for the State to become Running. This is done when State\_Number is set to 2 by the PLC.
- The Command will be set back to None, and the Command\_Number set to 0.
- When the PLC has completed the phase, it will set the State\_Number to 3 (Complete).
- The batch engine will set the Command to RESET and Command\_Number to 7.
- The batch engine will wait until the PLC sets the State\_Number to 1 (Idle).
- The phase is deactivated, and the batch continues with the next step.

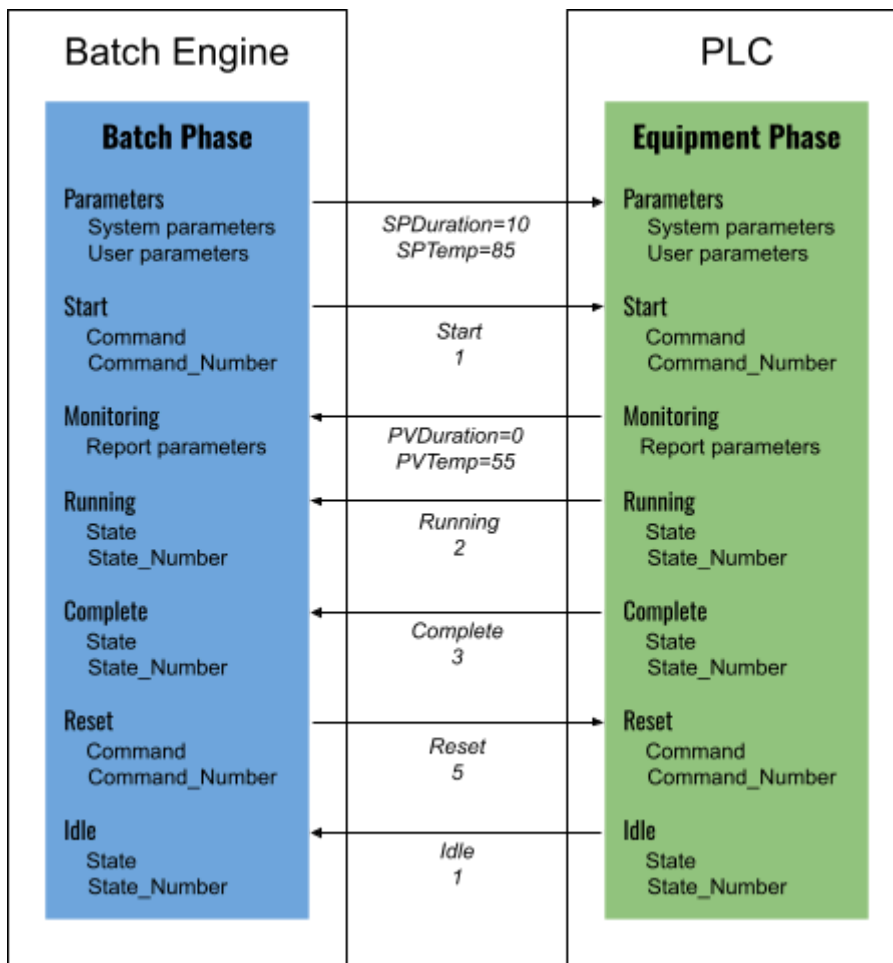


Figure 1: Batch Phase and Equipment Phase relationship

## Commands and States

Because PLCs are better at detecting and setting numbers compared to text, the following integer values are used by the PLC to detect commands and set the states.

<b>Command Name</b>	<b>Command Number</b>
NONE	0
START	1
PAUSE	2
RESUME	3
HOLD	4
RESTART	5
STOP	6
RESET	7
ABORT	8

Table 1: Commands

<b>State Name</b>	<b>State Number</b>
IDLE	1
RUNNING	2
COMPLETE	3
RESTARTING	4
RESETTING	5
PAUSING	6
PAUSED	7
HOLDING	8
HELD	9
STOPPING	10
STOPPED	11
ABORTING	12
ABORTED	13

Table 2: States

# State Diagram

The following diagram shows the appropriate commands for all of the states. The states are shown in boxes, and the commands are on the lines between the states. The appropriate command depends on the current state. For example, in Idle (green box on the left), the only appropriate command is Start which will change the state to Running.

State Transition Diagram

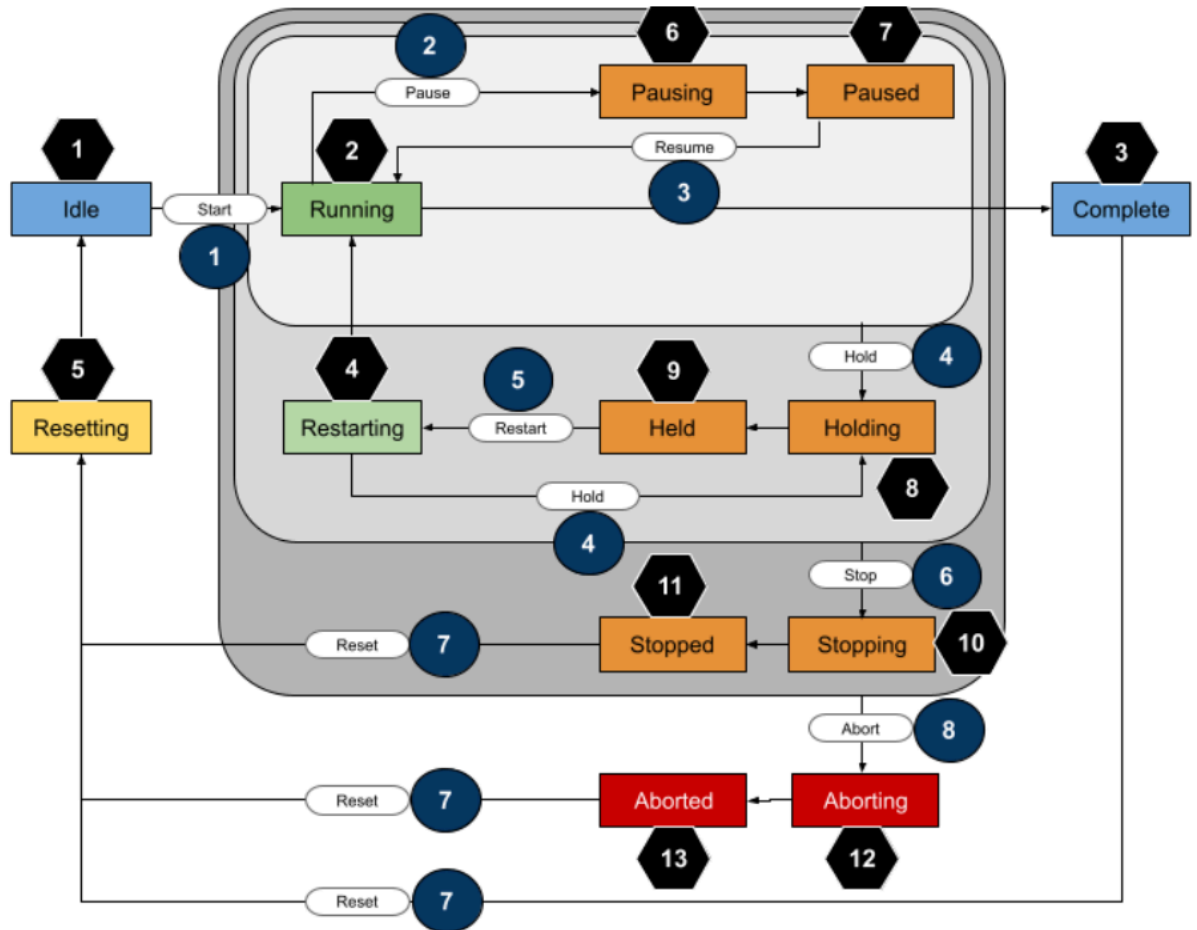


Figure 2: State diagram

Command Name	Command Number		State Name	State Number
Start			Idle	
Pause			Running	
Resume			Complete	
Hold			Restarting	
Restart			Resetting	
Stop			Pausing	
Reset			Paused	
Abort			Holding	
			Held	
			Stopping	
			Stopped	

			Aborting	
			Aborted	

Figure 2: State diagram

Current State	Command								New Final State
	Start	Stop	Hold	Restart	Abort	Reset	Pause	Resume	
Idle	Running								
Running		Stopping	Holding		Aborting		Pausing		Complete
Complete						Resetting			
Pausing		Stopping	Holding		Aborting				Paused
Paused		Stopping	Holding		Aborting			Running	
Holding		Stopping		Restarting	Aborting				Held
Held		Stopping			Aborting				
Restarting		Stopping	Holding		Aborting				Running
Stopping					Aborting				Stopped
Stopped					Aborting	Resetting			
Aborting									Aborted
Aborted						Resetting			
Resetting									Idle

Table 3: Command and state transition table

In the event that communications are lost or a restart of the batch engine, the batch engine will adapt to the current state read from the PLC. In the scenario that communications are lost between the two and the PLC continues the phase to completion, the following will occur when communications are reestablished:

1. For safety reasons, when the batch engine starts, all running batches are HELD
2. The batch is RESTARTED by the user or from a script
3. Each phase state is read from the PLC
4. If the state is:
  - a. RUNNING, then the phase will continue to run
  - b. COMPLETE, then the phase will be considered complete, and a RESET command will be issued

- c. HELD or HOLDING, then the phase will remain held until a RESTART command is issued by the user or scripting
- d. PAUSED or PAUSING, then the phase will remain paused until a RESUME command is issued by the user or scripting
- e. STOPPED or STOPPING, then the phase will remain stopped until a RESET command is issued by the user or scripting
- f. ABORTED or ABORTING, then the phase will remain aborted until a RESET command is issued by the user or scripting

## Ignition UDTs

On the Sepasoft Batch and Procedure module end, Phases are managed using the Phase Manager component or by using scripting. When the Exposed option is set to True as shown in figure 3, a UDT (User Defined Type) for the Agitate phase is automatically created in Ignition. The UDT will include a member for each system and user-defined parameter. The Tag Type setting, as shown in figure 4, will determine the type of UDT member that can be one of the following.

- Memory - simple tags that do not poll or update their values. These can be bound to Ignition Perspective views, access in scripting, etc.
- OPC - tags that are linked to field devices (typically PLCs) through an OPC server.
- Reference - tags that refer to other existing tags. This type can be used for linking to MQTT tags.

\* Edit Phase

Phase Name: Agitate

Description: [Empty]

Exposed:  true

Base Phase: Equipment Phase

Parameters

Name	Value	Value Source	Tag Type	Data Type
Mode	Unknown	Neither	Memory	String
Phase_Exception		Execution	Memory	String
Propagate_Hold_To_Parent	true	Recipe	Memory	Boolean
State	idle	Neither	OPC	String
State_Number	1	Execution	Memory	Integer
State_Transition_Handling	PLI	Recipe	Memory	Enum (State_Transition_
Step_Name		Neither	Memory	String
Step_Sequence	0	Neither	Memory	Integer
Unit_Path		Neither	Memory	String
SP_Duration_Minutes		Recipe	OPC	Integer

Figure 3: Batch and Procedure phase with the Expose option selected.

The SP\_Duration\_Minutes duration parameter is set to OPC tag type and a Data Type of Integer, as shown in figure 4. Note that many PLC vendors do not support spaces in the tag names, and for this reason, underscores should be used instead.

The image shows a dialog box titled "Edit Param" with the following fields and options:

- Parameter Name: SP\_Duration\_Minutes
- Recording Type \*: All to Object and Last to History
- Value Source \*: Recipe
- Tag Type: OPC
- Calculation: (empty field)
- Data Type \*: Integer
- Min Value: (empty field)
- Max Value: (empty field)
- Parameter Value: (empty field)

At the bottom right, there are two buttons: "CANCEL" and "SAVE".

Figure 4: Batch parameter setting, including the tag type.

When the phase configuration is saved, the Agitate UDT will be automatically created in the MES tag provider. Refer to Figure 5 and note that the SP\_Duration\_Minutes member of the UDT has a lock next to the name. This is because the parameter Value Source is set to Recipe. This means the value can only be set in the master recipe and not changed thereafter. If the Value Source is set to Recipe and Execution, the lock will not appear, and the value can be changed during execution originating from within the batch engine or PLC.

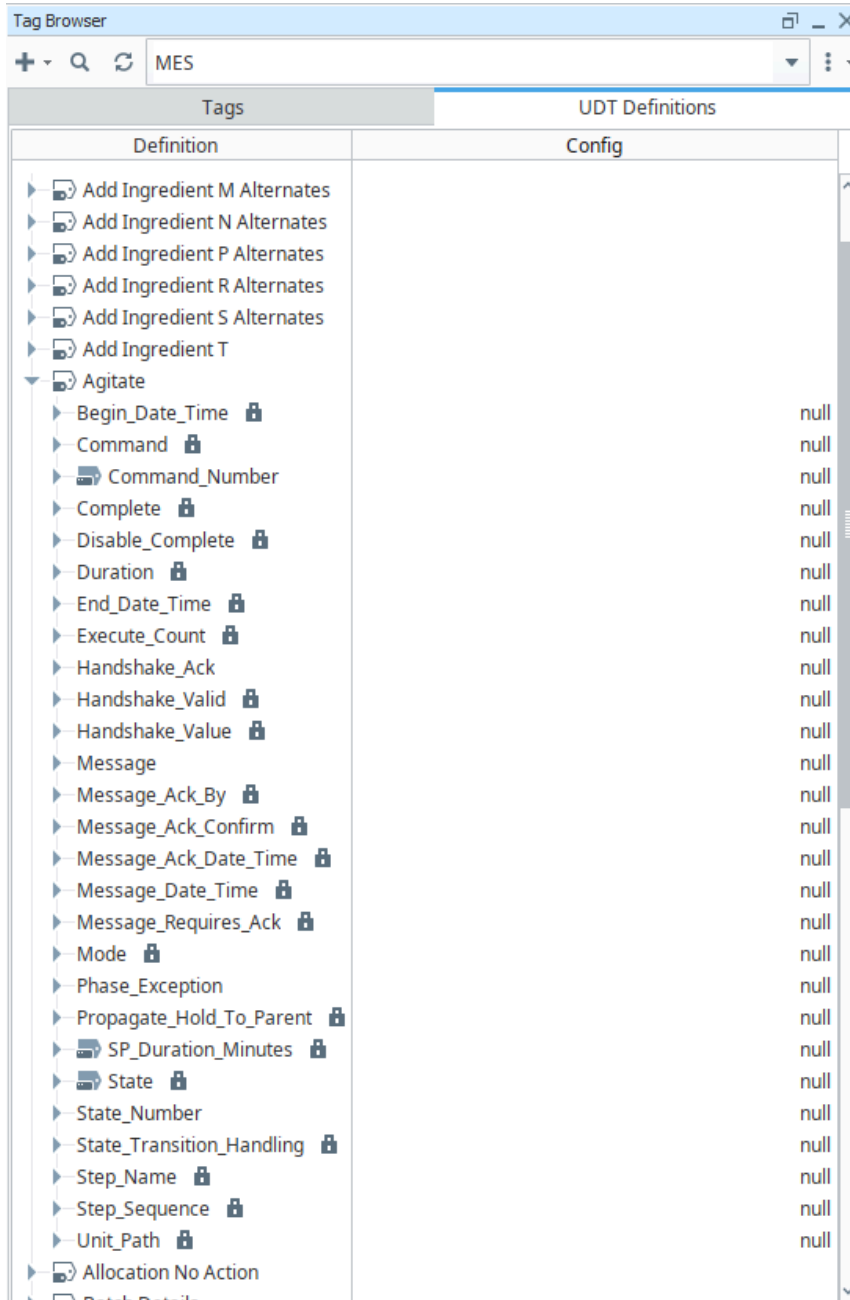


Figure 5: Tag Browser in the Ignition Designer showing the Agitate UDT

The UDT can be edited to assign the OPC item paths or reference tag paths. Both can be parameterized to allow dynamic paths based on the unit name or other UDT parameter values. As shown in figure 6, the {Unit} portion of the OPC Item Path will be replaced with the unit name configured in the Batch and Procedure module for all instances of the UDT.

When scaling to multiple units that support the same phases, the use of parameterized OPC Item Paths will eliminate manually mapping them. This structure should be used for greenfield implementations. For an existing implementation that does not have a good structure, manually assigning the OPC Item Paths in the instance must be done.

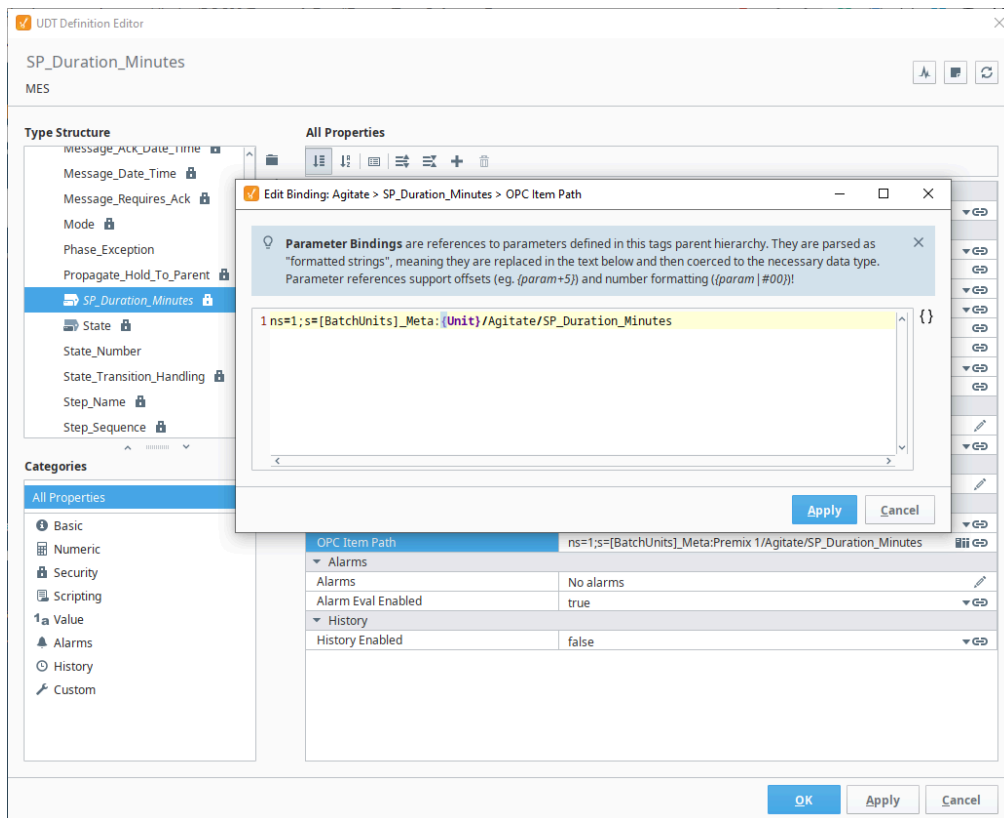


Figure 6: UDT Editor in the Ignition Designer showing an OPC Item Path

Once the UDT is configured, an instance of it will be created for each unit that is configured for the associated phase. Note in figure 7 that the {Unit} UDT parameter of OPC Item Path has been replaced with Test C.

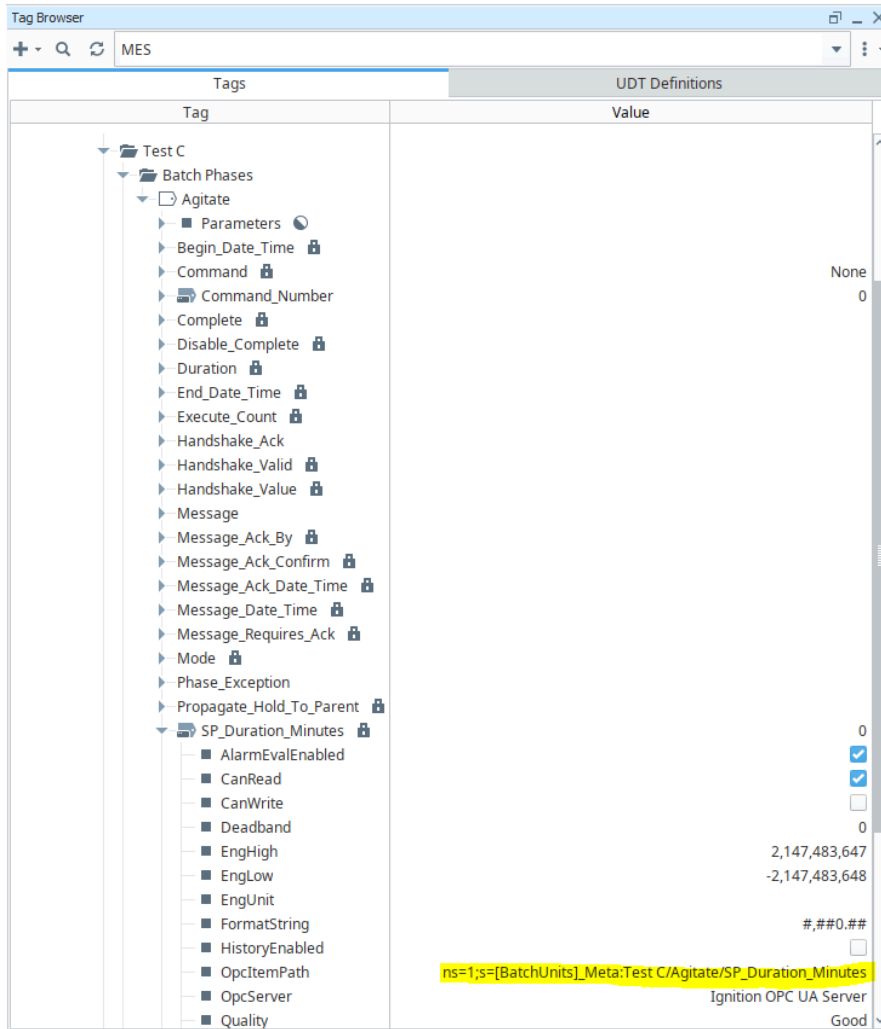


Figure 7: UDT Editor in the Ignition Designer showing an OPC Item Path

# Handshaking

The batch engine generates a new sequential handshake value every second that can be used to verify communications between the batch engine and the PLC. Referring to Figure 8, if the PLC doesn't detect the Handshake\_Value tag value changing every second, it can respond appropriately to the loss of communications to maintain safety. In the batch engine, the Handshake\_Valid parameter or tag can be used to handle lost communications.

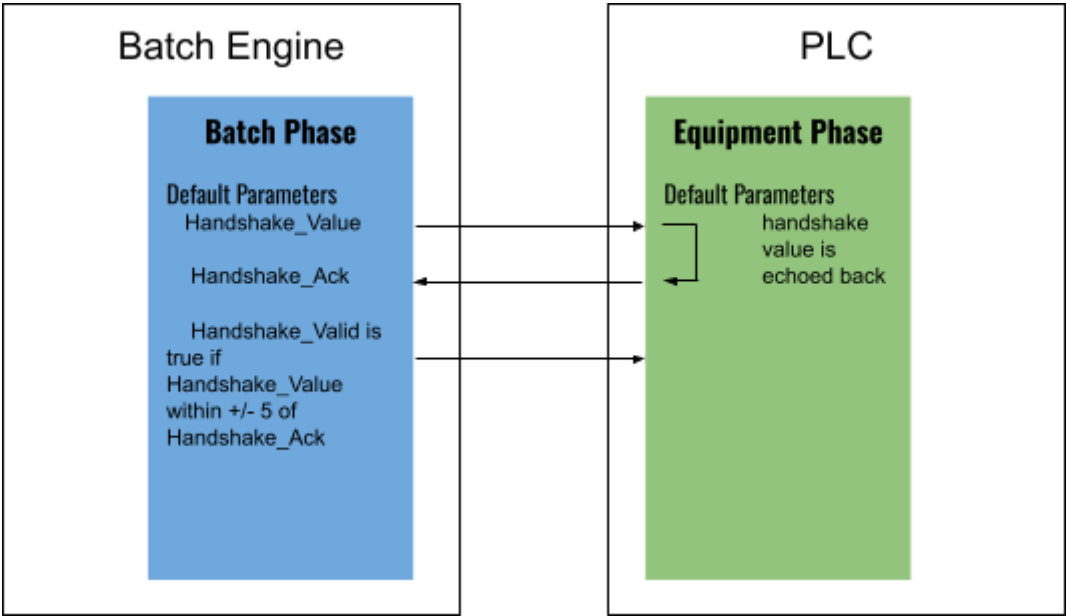


Figure 8: Handshake relationship between batch engine and PLC

# ControlLogix Processors

ControlLogix processors from Rockwell Automation support extended equipment phase functionality by enabling the CIP Phase Manager. This is optional, and if it is not enabled, the functionality described above can be used. If it is being used, phase instructions and separate program logic files are used for each state. Going back to our Add H2O example, when a phase transitions to the Running state, a program file contains logic to control the valves, pumps, monitor the amount added, handle safety, and handle exceptions. There is a separate program file for the Resetting state that can handle items like zeroing the flow meter, etc.

Because the CIP Phase Manager doesn't use tags, there is a program file specific to handling the PLI interface. This will need logic to monitor the Command\_Number tag and execute a PLC instruction against the equipment phase. For example, when the Command\_Number = 1, it executes the Phase Start command. And when the Running bit on the Equipment Phase is detected, it writes 2 into the State\_Number tag.

## PLI Logic

The following diagram shows an example of a Unit (MashKettle2) folder followed by a phase folder (MK\_Phase) which contains all the equipment phases with all the state's subroutines. All the logic, conditions, and commands will be programmed under the "Running" subroutine.

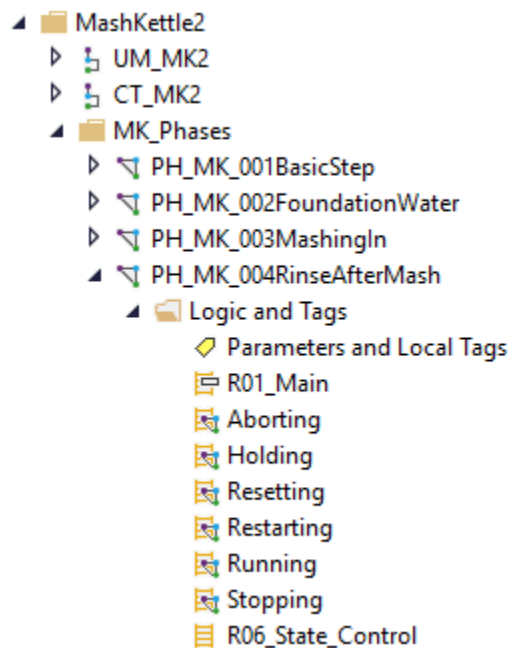


Figure 9: Phase routine structure in ControlLogix logical organizer

Figure 10 shows an example of an equipment phase's options and configurations

General Configuration Parameters Monitor

Assigned routines:

Prestate:  ▾

Fault:  ▾

Inhibit equipment phase

Synchronize redundancy data after execution

Initial state:  ▾

Complete state immediately if not implemented or empty

Initial step index:  ▴ ▾

External sequencer loss of communication command:  ▾

External request hold action:  ▾

Automatically assign the values of sequence step inputs to phase inputs

Automatically assign the values of phase outputs to sequence step outputs when the phase is:

- Complete
- Stopped
- Aborted

Figure 10: Phase routine structure in ControlLogix logical organizer

# Command Control Logic

Figure 11 shows an example of equipment phase command logic to interface the command value received from the batch engine and execute the appropriate PLC instruction. Command\_Number holds the command value from the batch engine as an integer value and executes the appropriate command on the equipment phase via the “PCMD” PLC instruction.

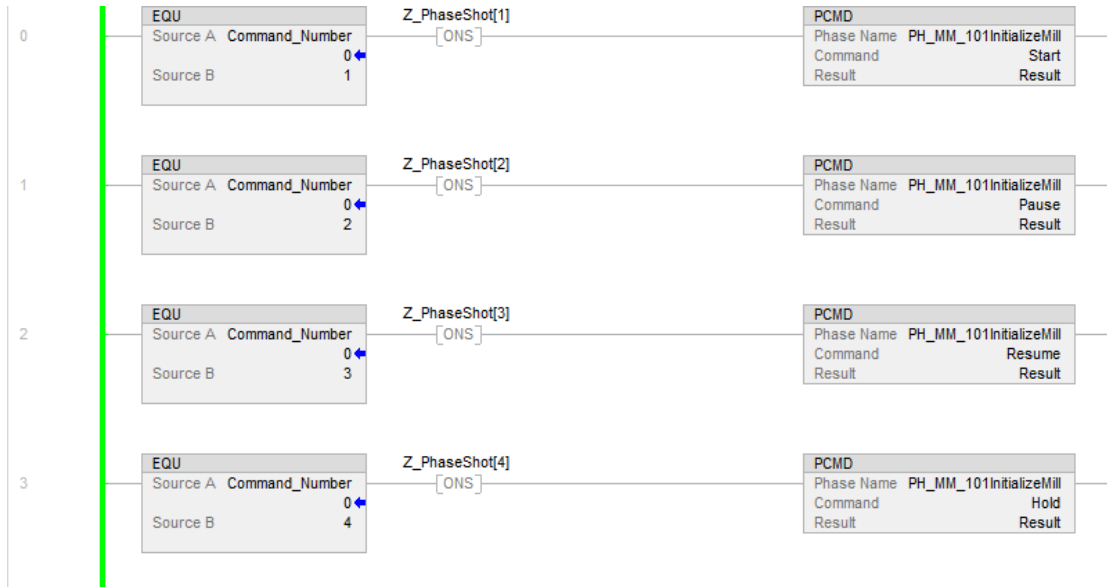


Figure 11: ControlLogix command logic

# State Control Logic

Figure 12 shows an example of equipment phase state logic to interface echoing the state from the PLC to the batch engine. Equipment phase tags in the PLC represent the state of a phase as a boolean value and then set the corresponding state number tag (State\_Number) as an integer used by the batch engine. This is accomplished with the “MOV” PLC instruction.

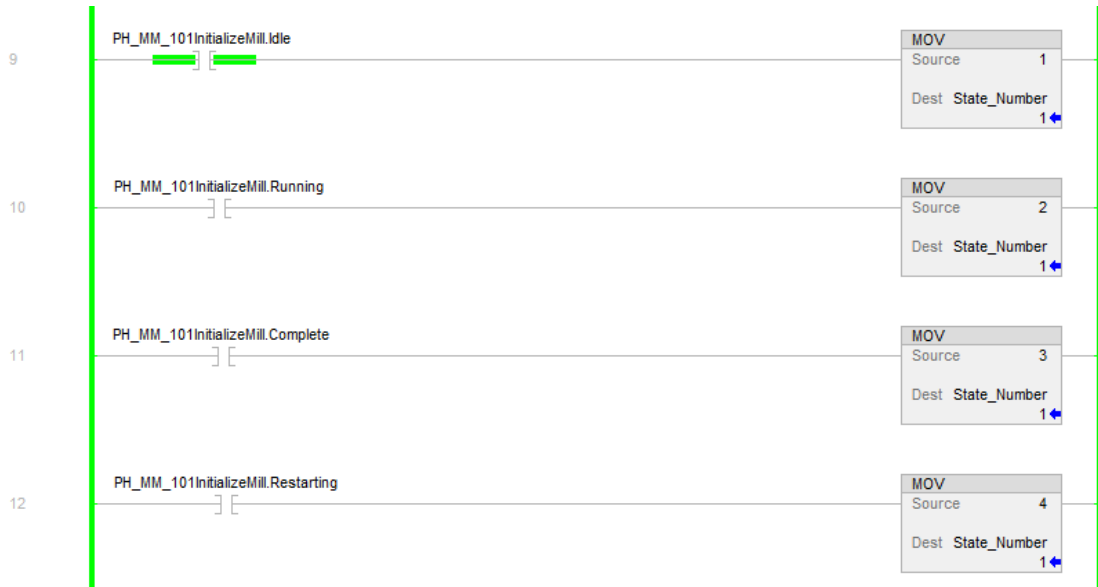


Figure 12: ControlLogix state logic

# Batch Message Control Logic

Figure 13 shows an example of equipment phase message logic from the PLC to the Batch engine. When the “TimerMonitor.Out\_Reached” tag activates, it will modify the “Message” tag of the batch engine, which is a string value. This modification of string value will initiate a Batch message on the HMI side if the step is in the “Running” state.

Then if the “TimerMonitor.Out\_Reached” tag deactivates or the “Message\_Ack\_Confirm” tag from the Batch engine activates, then the “Null” value will be moved to the “Message” string tag, which clears the Batch message on the batch engine side.

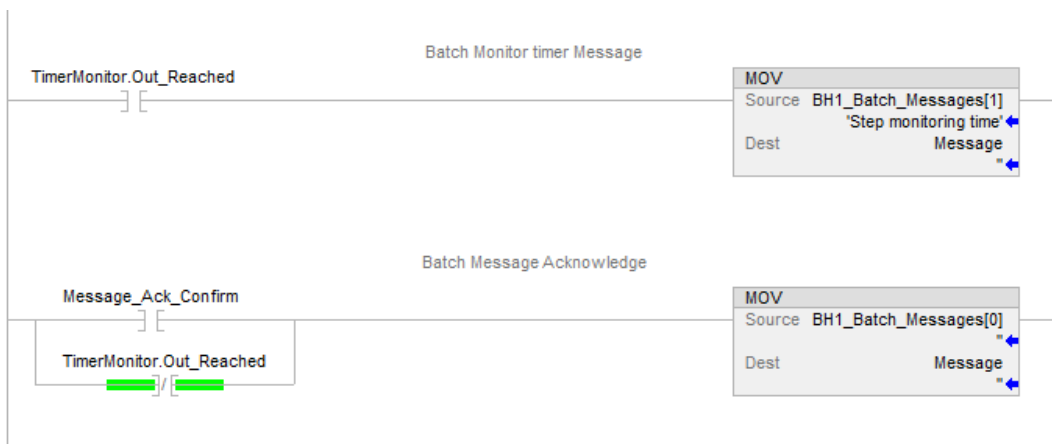


Figure 13: PLC-generated message logic