



# Proficiency Batch Execution 5.6

PLI Development Manual



**Proprietary Notice**

The information contained in this publication is believed to be accurate and reliable. However, General Electric Company assumes no responsibilities for any errors, omissions or inaccuracies. Information contained in the publication is subject to change without notice.

No part of this publication may be reproduced in any form, or stored in a database or retrieval system, or transmitted or distributed in any form by any means, electronic, mechanical photocopying, recording or otherwise, without the prior written permission of General Electric Company. Information contained herein is subject to change without notice.

© 2020, General Electric Company. All rights reserved.

**Trademark Notices**

GE, the GE Monogram, and Predix are either registered trademarks or trademarks of General Electric Company.

Microsoft® is a registered trademark of Microsoft Corporation, in the United States and/or other countries.

All other trademarks are the property of their respective owners.

We want to hear from you. If you have any comments, questions, or suggestions about our documentation, send them to the following email address:

[doc@ge.com](mailto:doc@ge.com)

# Table of Contents

About This Guide .....	1
Reference Documents .....	1
Introduction .....	1
Understanding the PLI .....	1
Memory Variables .....	2
Allocating Variables on a PLC and DCS .....	3
PLI Configurations .....	3
Writing a Single-Instance PLI .....	4
Developing a PLI .....	4
Task Overview .....	5
Using This Manual .....	6
Allocating Phase Memory Variables .....	7
Overview .....	7
Phase Memory Variable Naming Convention .....	7
Batch Execution Communication .....	7
Communication from the Batch Execution Server to the PLI .....	8
Communication from the PLI to the Batch Execution Server .....	9
Communication from the HMI to the PLI .....	12
Communication from the Batch Execution Server or the HMI to the Phase Logic .....	14
Communication from the Phase Logic to the Batch Execution Server or the HMI .....	15
State Transition Logic and Batch Execution Commands .....	16
Overview .....	16
State Transition Paths .....	17
State Transition Matrix .....	18
Phase States .....	19

Active States.....	19
Inactive States .....	20
Changing States .....	21
Batch Execution Commands .....	22
Batch Execution Command Mechanism.....	23
Example: Normal Phase Execution Scenario.....	24
When the Batch Execution Server starts the phase, the Server: .....	26
The PLI: .....	26
The Phase Logic:.....	26
The PLI: .....	26
The Batch Execution Server:.....	26
The PLI: .....	26
The Batch Execution Server:.....	27
External Control.....	27
Understanding the Batch Execution and External Control Relationship .....	27
Understanding the PLI.....	28
Flowchart Conventions .....	28
Programming the Main Calling Program .....	29
First Scan Initialization (3:1) .....	30
PLI Overview .....	31
Checking the Watchdog .....	32
Processing the Phase.....	34
Copying Commands .....	36
Checking for Phase Failure .....	37
Checking Phase Flags.....	38
Process Aborting Complete.....	39
Process Stopping Complete .....	40
Process Holding Complete .....	41

Process Restarting Complete .....	42
Process Running Complete .....	44
Processing Commands .....	45
Phase Mode.....	46
Setting Flags.....	47
Processing the Idle Phase State .....	48
Processing Commands in the PLI .....	49
Overview: Processing Commands .....	49
Operator-Issued Commands .....	50
Process Abort Command .....	51
Process Hold Command.....	53
Process Stop Command.....	56
Process Reset Command.....	59
Process Restart Command .....	61
Process Start Command .....	63
Process Clear Failure Command .....	64
Handshaking Commands .....	65
Process Request Confirmed Command.....	65
Process Cancel Previous Request Command .....	66
Process Request Unsuccessfully Canceled.....	67
Process Clear Request Register Command .....	68
Mode Change Commands.....	68
Process Phase Mode P-Auto Command.....	69
Process Phase Mode O-Auto Command .....	70
Process Phase Mode Manual Command.....	71
Download Request Command Process .....	72
Phase Debugging Commands.....	72
Process Single Step Command.....	73

Process Pause Command.....	74
Process Resume Command.....	75
Quick References .....	75
Batch Execution Command Values .....	76
Phase Status Values .....	77
Batch Execution Requests .....	78
iFIX Database Tags.....	80
Sample PLI .....	81
Sample AB5 PLI Ladder Logic Notes .....	82
Data Table File Overview .....	82
Configuring the Request Qualifier Array.....	82
Configuring Phase and Report Parameters .....	83
Data Table File Descriptions .....	83
Other Tags.....	91
Troubleshooting.....	91
Index .....	95

---

# About This Guide

The Proficy Batch Execution PLI Development Manual is a comprehensive guide to developing the Phase Logic Interface (PLI) through the use of programmable controllers or other similar type of control equipment.

This manual is intended for those who wish to develop and use the standard Batch Execution application interface targeted for programmable controllers. The manual assumes the reader has a good understanding of batch control processing, programmable or process controllers, and ladder logic.

---

## Reference Documents

For additional information about developing phase logic and the Phase Logic Interface, refer to the following documents:

- Phase Programming Manual
- ISA-S88.01, Batch Control, Part 1: Models & Terminology

---

## Introduction

The sections that follow provide an overview of the Batch Execution Phase Logic Interface (PLI) and its place in the communication stream that ultimately controls the execution of phases in Batch Execution:

- Understanding the PLI
- Allocating Variables on a PLC and DCS
- PLI Configurations
- Developing a PLI
- Task Overview
- Using This Manual

---

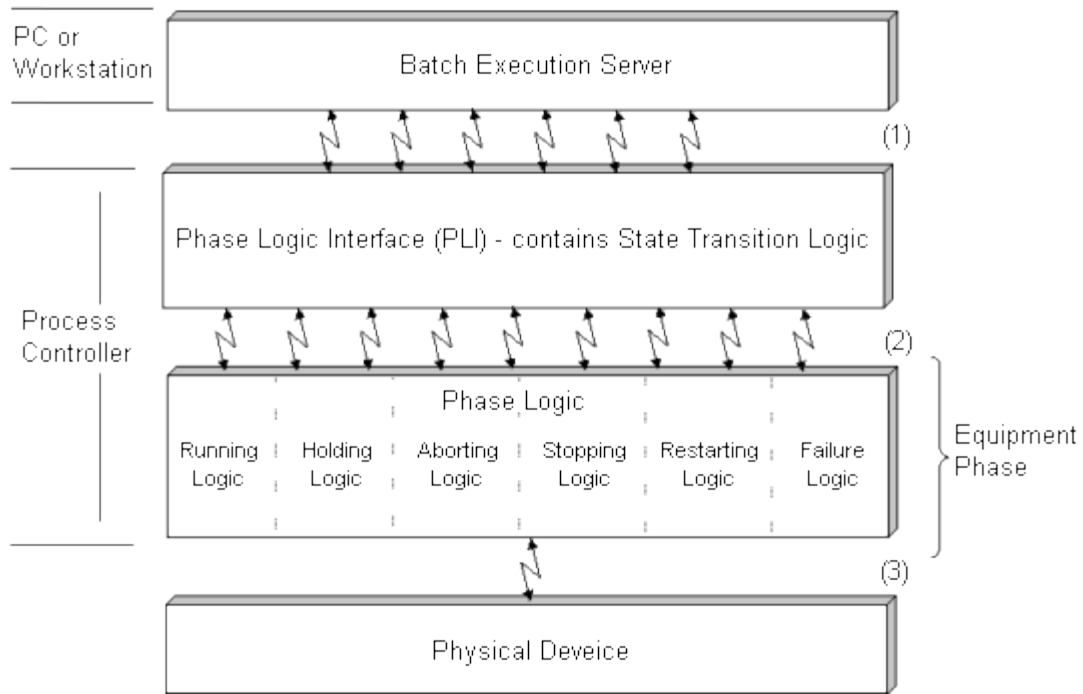
## Understanding the PLI

The PLI is the standard interface between the Batch Execution Server and the phase logic in the process controller. The PLI's purpose is to control:

- The state transitions for phases.
- The communication between the Batch Execution Server and the process I/O phase logic.

The PLI receives commands from the Batch Execution Server, the operator, or an external interface, and then initiates the appropriate module of phase logic in the process controller.

The following figure illustrates this communication flow. (1) The Batch Execution Server writes and receives commands from the PLI. (2) The PLI writes and receives commands from the phase logic, which (3) ultimately controls the physical device on the plant floor.



*Communication Flow*

For example, when the PLI receives a START command from the Batch Execution Server, the PLI initiates the Running module in the phase logic. If the PLI receives a STOP command from the Batch Execution Server, it initiates the Stopping module of code.

## Memory Variables

Memory variables store the values that Batch Execution and the phase logic use to communicate. Certain variables store data that the Batch Execution Server sends to the PLI and others store data that propagate up from the phase logic. A *memory variable* is a named storage space that exists in the process controller's memory. When a process controller program runs, this space is allocated and can be used to store data. A process controller program references this space by the variable name.

The Batch Execution Server, the PLI, and the phase logic store and retrieve data such as command values, status values, parameter values, and request values in the appropriate memory variable. This value triggers an action by the Batch Execution Server, the PLI, or the phase logic.

For example, when the phase logic completes the running logic, it sets the value of the Running Complete variable to complete. The PLI sees this value and sets the Phase Status variable value to complete. The Batch Execution Server sees that the Phase Status variable value is set to complete and begins execution of the next phase in the process, if appropriate. This cycle continues for each phase until the batch completes.



---

## Allocating Variables on a PLC and DCS

For specific information pertaining to your process controller, refer to your process controller manual. The general allocation procedures are described below.

When allocating phase memory variables:

1. Choose a memory location for the variable.
2. Associate a memory variable name with the chosen memory location.

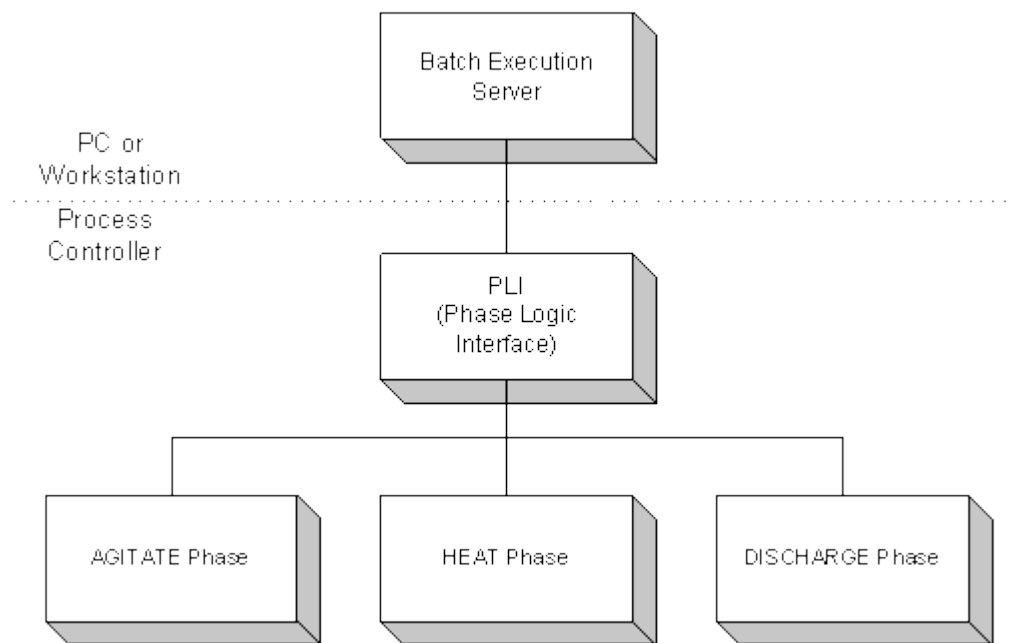
---

## PLI Configurations

How you configure your PLI depends on the capabilities of your process controller. There are two basic types of PLI configurations: single-instance (recommended) and multiple-instance. However, depending on your equipment, you may choose to implement a combination of the two configurations.

### Single-Instance PLI Configuration

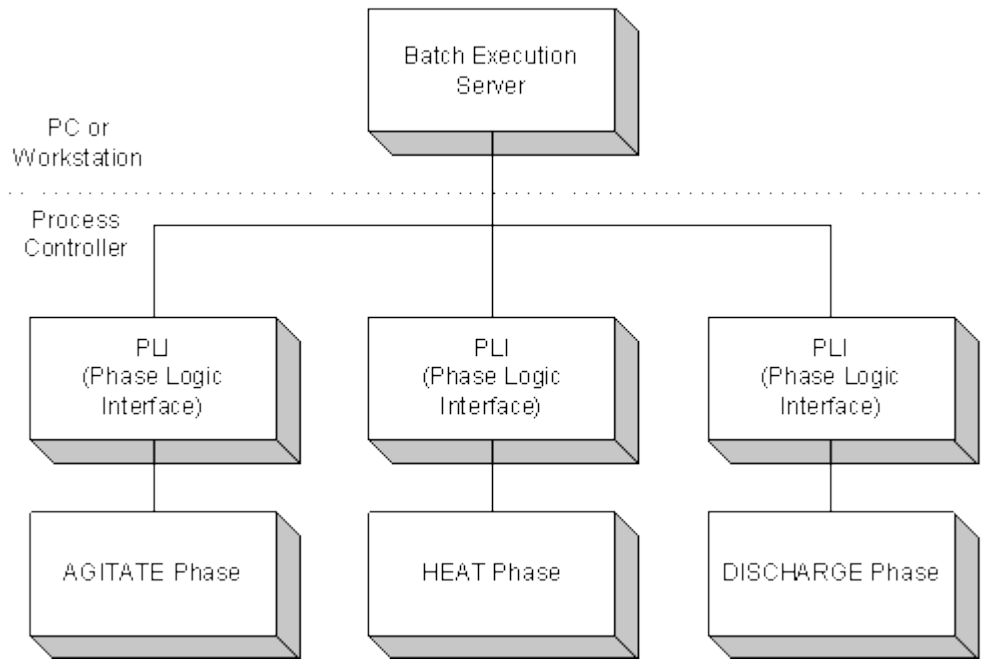
If your process controller supports indirect addressing, configure one, generic PLI to communicate with all of your phases. Using indirect addressing to create one PLI is the best design approach because you only need to maintain one PLI. The following figure illustrates a single-instance PLI configuration.



*Single-Instance PLI Configuration*

### Multiple-Instance PLI Configuration

If your process controller does not support indirect addressing, you may need to create individual PLIs for each phase. The following figure illustrates a multiple-instance PLI configuration.



*Multiple-Instance PLI Configuration*

## Writing a Single-Instance PLI

A single-instance PLI uses indirect addressing to communicate with all phases. Indirect addressing uses offsets to determine the phase number. Each phase must have a unique phase number. Use a Main program in the process controller to calculate what offset is being addressed.

For example, the Main program in the sample AB5 ladder logic in the Sample AB5 PLI Ladder Logic section, uses N26:0 to hold the phase number for the currently processing phase. For example:

```
N23[N26:0]
```

N26:0 represents the offset for the currently processing phase. If Phase 3 is currently processing, this address is interpreted as:

```
N23:3
```

---

## Developing a PLI

Many of the examples in this manual are specific to a certain PLI; they refer to the sample AB5 Ladder logic PLI or the sample Modicon Concept PLI. If you choose to create your own PLI and it is significantly different than those provided at the end of this book, the following is a list of the elements you must address in your PLI:

**Commands** – you must include a process in your PLI for recognizing the commands that Batch Execution sends to the process controller.

**Status** – the PLI must update the status of the phases to the Batch Execution Server.

**State Transitions** – the PLI must control state transition logic for the phases.

**Equipment Phase Tags** – you must provide a method for Batch Execution to communicate with each equipment phase tag defined for the project. These tags will always include:

- COMMAND
- FAILURE
- OWNER
- PAUSE
- PAUSED
- REQUEST
- SINGLE\_STEP
- STATUS
- STEP\_INDEX
- UNIT

Optional tags are PARAMETERS, REPORTS, and REQUEST DATA.

***NOTE:** These tags are listed in the Equipment Editor's Edit Equipment Phase dialog box.*

## Task Overview

The following provides an overview of the tasks you need to perform to program the PLI.

1. Design your control strategy. A solid control strategy must be in place before you begin PLI development. Keep in mind that the PLI development and phase logic development can be done independently. The elements that must be clearly communicated from the phase logic developer to the PLI developer are the flag register sets.
2. Allocate the required phase memory variables in your basic control device, such as a PLC. Allocate additional variables if necessary, depending on your control strategy.
3. Program the PLI. The PLI is grouped into several functional components. Each component is listed below. Refer to the section indicated for detailed programming information.

To program this PLI Component...	Refer to this section...
Check the watchdog flag.	Checking the Watchdog
Determine if the phase needs to be processed.	Processing the Phase
Copy the command to the temporary command register.	Copying Commands
Check for phase failure.	Checking for Phase Failure

<b>To program this PLI Component...</b>	<b>Refer to this section...</b>
Check the phase completion flags.	Checking Phase Flags
Process the completion logic.	Checking Phase Flags
Process the phase mode command.	Phase Mode
Set general use flags.	Setting Flags
Process the idle phase.	Processing the Idle Phase State
Process the command.	Processing Commands in the PLI

---

## Using This Manual

The manual uses a combination of text, graphics, flowcharts, and sample code (presented in both ladder logic and in structured text) to define and describe the Batch Execution PLI. The Understanding the PLI and Processing Commands in the PLI sections break down and describe each step of a PLI through a graphical representation (flowchart) and accompanying text. Each of the flowcharts in these two sections, as well as many examples throughout the book, refer to the Sample AB5 PLI Ladder Logic section. You can analyze the concepts presented in these sections and compare them to the sample ladder, and apply that information to your own specific PLI.

The phase memory variables discussed in this manual are identified by the phase name followed by an underscore character and a two or three character extension. For example:

PHASE\_xxx

If you need additional information on any of the phase memory variables, you can quickly locate this information in the manual's appendices.

<b>To find additional information on this component of the PLI...</b>	<b>Refer to this section...</b>
Command values	Batch Execution Command Values
Status values	Phase Status Values
Request functions	Batch Execution Requests
Corresponding iFIX tags	iFIX Database Tags

To find additional information on this component of the PLI...	Refer to this section...
Detailed description of the memory variables	Sample AB5 PLI Ladder Logic
Sample AB5 Ladder Logic	Sample AB5 PLI Ladder Logic

---

## Allocating Phase Memory Variables

The sections that follow describe the function of the phase memory variables. These sections also discuss the communication between the Batch Execution Server and the PLI. Communication between the PLI and the Phase Logic is discussed in subsequent sections.

---

### Overview

Batch Execution requires that you define phase memory variables for unique data items. A *memory variable* is a named storage space that exists in the process controller's memory. When a process controller program runs, this space can be used to store data. A process controller program references this space by the variable name or address.

Memory variables store the values that the Batch Execution Server and the phase logic use to communicate. Certain variables store data that the Batch Execution Server sends to the PLI and others store data that propagates up from the phase logic.

### Phase Memory Variable Naming Convention

The recommended phase memory variable name begins with the phase name followed by an underscore character and a two or three character extension:

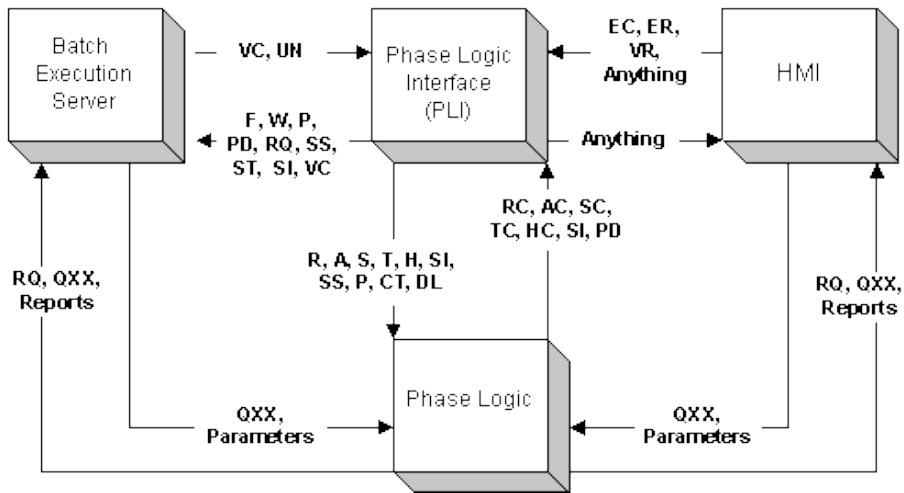
PHASE\_XXX

For example, the recommended name for the Batch Execution Command variable for a phase called Agitate is:

AGITATE\_VC

### Batch Execution Communication

The following figure shows each component of Batch Execution and how communication occurs between the different components. The arrows between each of the components indicate the flow (direction) of the communication. The variables found along the arrows are identified by their two or three character extension.



*Batch Execution Communication Overview*

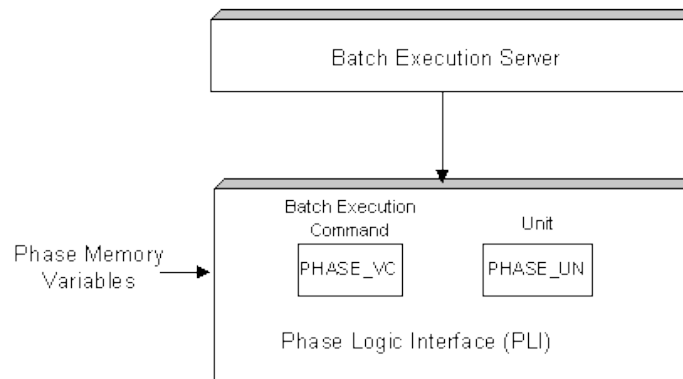
The following sections describe the purpose, data type, and recommended variable name for each data item. The items are grouped into the following categories, based on the communication flow:

- Communication from the Batch Execution Server to the PLI.
- Communication from the PLI to the Batch Execution Server.
- Communication from the HMI to the PLI.
- Communication from the Batch Execution Server or the HMI to the Phase Logic.
- Communication from the Phase Logic to the Batch Execution Server or the HMI.

For additional information on the variables that pass between the phase logic and the PLI, refer to the Phase Programming Manual.

## Communication from the Batch Execution Server to the PLI

The Batch Execution Server can send command values to the PLI and indicate on which unit a phase is executing. The following figure shows this communication between the Batch Execution Server and the PLI.



*Communication between the Batch Execution Server and the PLI*

The following table describes the data that is passed from the Batch Execution Server to the PLI.

<b>Communication from the Batch Execution Server to the PLI</b>			
<b>Description</b>	<b>Recommended Variable Name</b>	<b>Data Type</b>	<b>Purpose</b>
Batch Execution Command	PHASE_VC	Integer	Commands the phase to enter a particular state as defined by the state transition logic. The state transition logic uses these commands to transition batches to different states.
Unit	PHASE_UN	Integer	Indicates the unit on which a phase is executing, when a phase is common to two or more units. The integer corresponds to a particular unit.

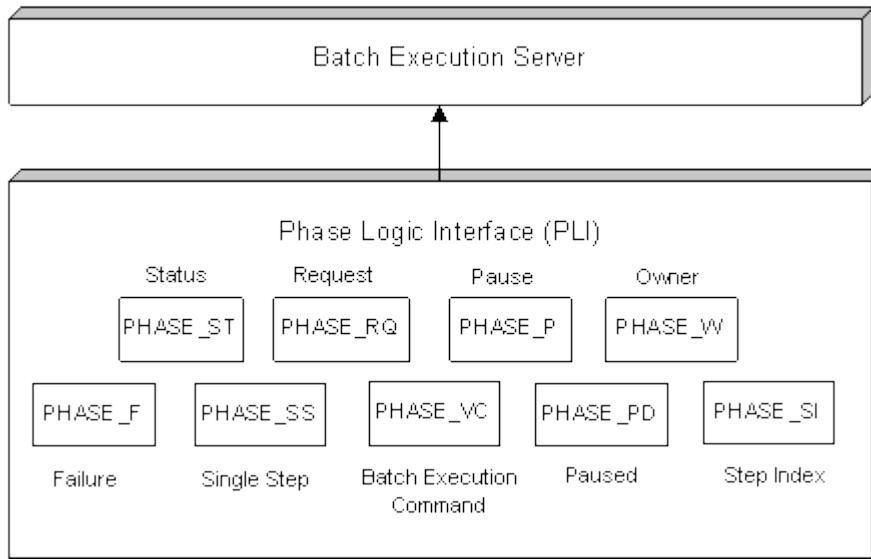
---

## Communication from the PLI to the Batch Execution Server

The following data items propagate up from the phase logic. The basic control device monitors and sends these data items to the Batch Execution Server.

- Status
- Requests
- Failure
- Pause
- Paused
- Owner
- Single-step
- Batch Execution Command
- Step Index

The following figure shows the communication from the PLI to the Batch Execution Server.



*Communication from the PLI to the Batch Execution Server*

The information in the following table is passed from the PLI to the Batch Execution Server.

<b>Communication from the PLI to the Batch Execution Server</b>			
<b>Description</b>	<b>Recommended Variable Name</b>	<b>Data Type</b>	<b>Purpose</b>
Failure Number	PHASE_F	Integer	The Failure Number variable is set when any device detects a fail condition or when a process condition occurs that requires a Hold command. An increase in the failure priority when the phase is in the Running, Holding, Held, or Restarting state causes the phase to transition to the start of the Holding logic. A failure number greater than zero can also inhibit the state transitions of Idle to Running and Held to Restarting states.
Pause	PHASE_P	Boolean	Allows the operator to request a single pause by issuing a PAUSE command to the phase. This sets the Pause bit, indicating that the phase will pause at the next programmed pause transition. The PLI resets the pause bit when the next RESUME command occurs, thus causing only a single pause to occur.



<b>Communication from the PLI to the Batch Execution Server</b>			
<b>Description</b>	<b>Recommended Variable Name</b>	<b>Data Type</b>	<b>Purpose</b>
Paused	PHASE_P D	Boolean	Indicates that the phase has paused at the programmed transition. The operator can request a single pause by issuing a PAUSE command to the phase. Once the phase has paused at the programmed pause transition, the phase logic sets the Paused bit, indicating that the phase has paused at the programmed transition.
Request	PHASE_RQ	Integer	Initiates the request from the phase logic to the Batch Execution Server. Requests to download parameter values, acquire resources, and so on, are all made using this variable. Refer to the Phase Programming Manual for additional information on requests.
Status	PHASE_ST	Integer	Contains values that correspond to the state of the phase. Refer to the section the Phase States section for more information.
Step Index	PHASE_SI	Integer	When the PLI changes its current state to any active state, the PLI initializes the Step Index register to the value set by the user in PHASE_IS. This value is usually set to zero (0) or one (1). The user's phase logic uses the Step Index register to step through the steps or states by changing the value in the Step Index register.
Single Step	PHASE_SS	Boolean	Stores an attribute that indicates when the phase is in the single step mode. Single step mode causes the phase to pause at every programmed pause transition and wait for the operator to issue a RESUME command.
Batch Execution Command	PHASE_VC	Integer	Commands the phase to enter a particular state as defined by the state transition logic. The state transition logic uses these commands to transition phases to different states.

Communication from the PLI to the Batch Execution Server			
Description	Recommended Variable Name	Data Type	Purpose
Owner	PHASE_W	Boolean	Identifies the control mode, either Batch Execution or External. When false (0), the owner is Batch Execution; when true (1), the owner is External. The owner is determined by the external interface. Two request data items, Batch Execution Request (PHASE_VR) and External Request (PHASE_ER) are used to toggle between Batch Execution and External mode.

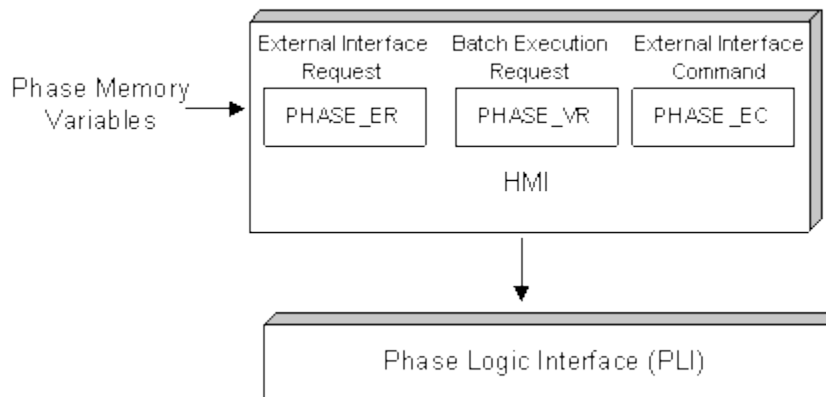
---

## Communication from the HMI to the PLI

The HMI can:

- Request external ownership of a phase.
- Request Batch Execution ownership of a phase.
- Send commands to a phase by storing data in the PHASE\_ER, PHASE\_VR, and PHASE\_EC memory variables.

The following figure illustrates the communication from the HMI to the PLI.



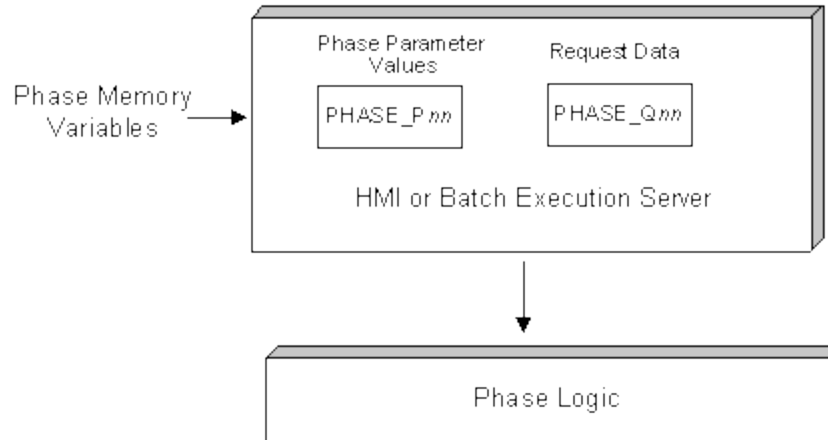
*Communication from the HMI to the PLI*

The following table describes the purpose of the variables that are passed from the HMI to the PLI.

<b>Communication from the HMI to the PLI</b>			
<b>Description</b>	<b>Recommended Variable Name</b>	<b>Data Type</b>	<b>Purpose</b>
External Requests	PHASE_ER	Boolean	Allows an external interface to manually control a phase. Upon execution of the request by the PLI, the PLI places the owner (PHASE_W) into External mode. This flag is usually set externally, such as from an HMI system.
Batch Execution Request	PHASE_VR	Integer	The Batch Execution Request item provides a flag that can be turned on to switch from External mode to Batch Execution mode. This returns the phase to Batch Execution control. This flag is set externally, from an HMI system.
External Commands	PHASE_EC	Integer	Allows an external interface to manually control a phase. Batch Execution does not use this variable. The state transition logic uses these commands to transition batches to different states.

## Communication from the Batch Execution Server or the HMI to the Phase Logic

The Batch Execution Server or an HMI can send phase parameter values and respond to requests by the phase logic by storing data in the PHASE\_P $_{nn}$  and PHASE\_Q $_{nn}$  variables. The following figure illustrates the communication from the Batch Execution Server or the HMI to the Phase Logic.



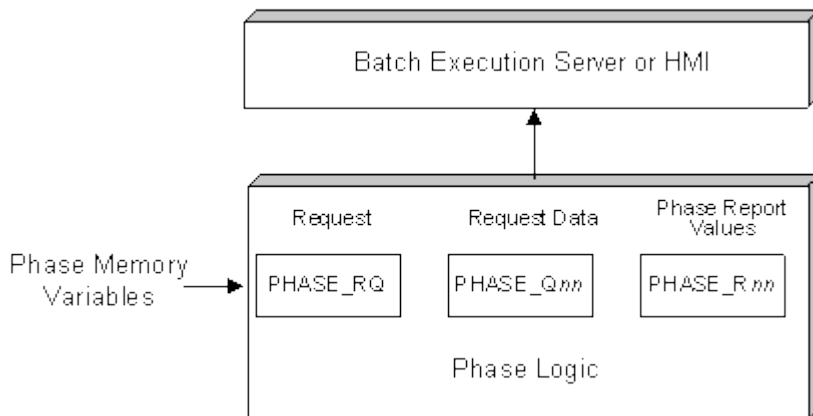
*Communication from the Batch Execution Server or HMI to the Phase Logic*

The following table describes the purpose of the variables that are passed from either the Batch Execution Server or the HMI to the Phase Logic.

Communication from the Batch Execution Server or HMI to the Phase Logic			
Description	Recommended Variable Name	Data Type	Purpose
Phase Parameter Values	PHASE_P $_{nn}$ , where $nn$ is a number. You can allocate several memory variables for the Request Data values.	Integer, Real, or String	Stores phase parameter values within the process controller. The Batch Execution Server can write a single parameter value or multiple parameter values. For example, the parameter values may contain information about the amount of ingredient the phase should add, or the amount of time that the phase should run.
Request Data	PHASE_Q $_{nn}$ , where $nn$ is a number. You can allocate an array of several memory variables for the Request Data values.	Integer	Provides a buffer to store values that clarify the request initiated with the Request data item. Refer to the Phase Programming Manual for additional information on requests.

## Communication from the Phase Logic to the Batch Execution Server or the HMI

The phase logic can upload report values and make requests by sending the, PHASE\_RQ, PHASE\_Qnn, and PHASE\_Rnn items to the Batch Execution Server or to the HMI. The following figure illustrates the communication from the Phase Logic to either the Batch Execution Server or the HMI.



*Communication from the Phase Logic to the Batch Execution Server or the HMI*

The following table describes the communication from the Phase Logic to the Batch Execution Server or the HMI.

Communication from the Phase Logic to the Batch Execution Server or the HMI			
Description	Recommended Variable Name	Data Type	Purpose
Phase Report Values	PHASE_Rnn, where nn is a number. You can allocate memory variables for the Request Data values.	Integer, Real, or String	Stores either a single report value or multiple report values. The PLI can write a single report value or multiple report values. The Batch Execution Server can read these report values.
Request	PHASE_RQ	Integer	Initiates the request from the phase logic to the iBatch Server. Requests to download parameter values, acquire resources, and so on, are all made using this variable. Refer to the Phase Programming Manual for additional information on requests.

Communication from the Phase Logic to the Batch Execution Server or the HMI			
Description	Recommended Variable Name	Data Type	Purpose
Request Data	PHASE_Qnn, where <i>nn</i> is a number. You can allocate an array or several memory variables for the Request Data values.	Integer	Provides a buffer to store values that clarify the request initiated with the Request variable. Refer to the Phase Programming Manual for additional information on requests.

---

## State Transition Logic and Batch Execution Commands

The sections that follow describe the state transition logic and how the PLI (Phase Logic Interface) controls the transitions between the phase states.

- Overview
- State Transition Paths
- Understanding Phase States
- Understanding Batch Execution Commands
- External Control

---

### Overview

The PLI is the standard interface to the project-specific phase logic. The PLI receives commands from the Batch Execution Server or the operator and then initiates the different components of the phase-specific logic. The state transition logic is based on the procedural states as defined in the ISA S88.01 Batch Control Standard.

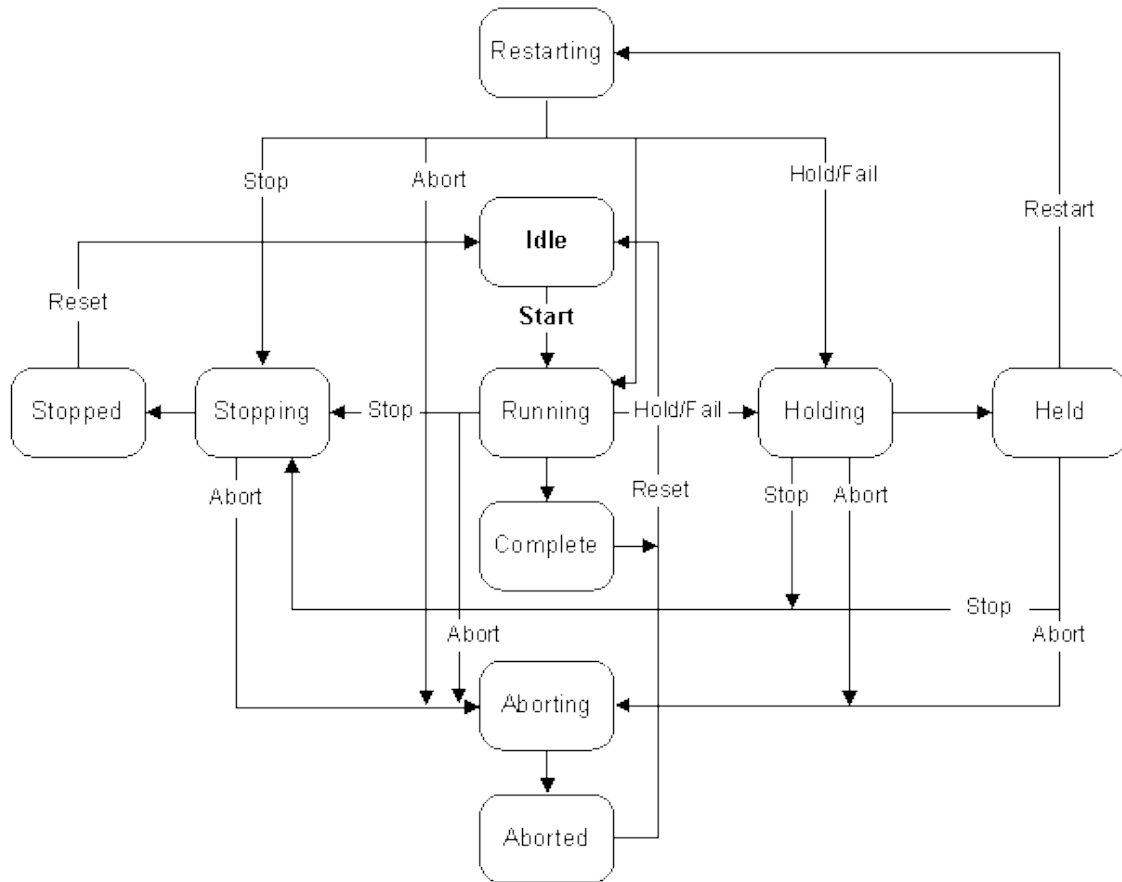
Two components of the state transition logic include the:

- Phase states supported by Batch Execution.
- Transition paths between the states.

Each component is described in the following sections.

## State Transition Paths

The state transition logic in the PLI enables the phase to transition from one state to another and must follow the valid transition paths, which are illustrated in the following figure.



*State Transition Diagram*

## State Transition Matrix

The following table provides a matrix representation of the state transition paths illustrated in the State Transition Paths section.

State Transition Matrix							
Initial State	No Command End State	Start	Stop	Hold	Restart	Abort	Reset
Idle		Running					
Running	Complete		Stopping	Holding		Aborting	
Complete							Idle
Holding	Held		Stopping			Aborting	
Held			Stopping	Holding	Restarting	Aborting	
Restarting	Running		Stopping	Holding		Aborting	
Stopping	Stopped					Aborting	



State Transition Matrix							
Initial State	No Command End State	Start	Stop	Hold	Restart	Abort	Reset
Stopped							Idle
Aborting	Aborted						
Aborted							Idle

---

## Phase States

Batch Execution supports 10 phase states, shown in the state transition diagram illustrated in the State Transition Paths section. Phase states fall into three categories: active, inactive, and changing states. The following sections describe each state:

- Active States
- Inactive States
- Changing States

### Active States

The following table describes the five active phase states supported by Batch Execution and lists their phase status variable (PHASE\_ST) values.

Active States			
State	Description	Status Value	Corresponding Flag
Aborting	The phase is executing its Aborting sequence. Upon completion of the aborting sequence, the phase automatically transitions to the Aborted state.	10	PHASE_A

<b>Active States</b>			
<b>State</b>	<b>Description</b>	<b>Status Value</b>	<b>Corresponding Flag</b>
Holding	The phase received a HOLD command or detected a device Failure and is executing its holding logic sequence to put the process into a safe state. If sequencing is not required, the phase immediately transitions to the Held state.	20	PHASE_H
Stopping	The phase was issued a STOP command. Upon completion of the stopping sequence, the phase automatically transitions to the Stopped state.	30	PHASE_S
Restarting	The phase received a RESTART command after being in the Held state. It is executing its restarting logic to return to the Running state. If no sequencing is required, the phase immediately transitions to the Running state.	40	PHASE_T
Running	The phase is exhibiting normal sequence execution.	50	PHASE_R

*NOTE: Only one active or inactive state flag is set at a given time. The PLI clears all other flags.*

## **Inactive States**

The following table describes the inactive phase states supported by Batch Execution and lists their phase status variable (PHASE\_ST) values.

<b>Inactive Phase States</b>			
<b>State</b>	<b>Description</b>	<b>Status Value</b>	<b>Corresponding Flag</b>
Held	The phase completed its Holding sequence, and the process has been brought to a safe state. The phase is waiting for a RESTART command to proceed to the Restarting state.	60	PHASE_HE
Complete	Normal sequence execution ran to completion. The phase is now waiting for a RESET command before returning to the Idle state.	70	PHASE_C

Inactive Phase States			
State	Description	Status Value	Corresponding Flag
Stopped	The phase completed the Stopping sequence. The phase is waiting for a RESET command to transition back to the Idle state.	80	PHASE_SD
Aborted	The phase completed the Aborting sequence. The phase is waiting for a RESET command to transition back to the Idle state.	90	PHASE_AD
Idle	The phase is waiting for a START command. Device failures are monitored and are displayed to the operator if detected. A phase with a failure ignores the START command.	100	PHASE_I

For additional information on the Active and Inactive state flags, refer to the Sample AB5 PLI Ladder Logic section.

## Changing States

The following table describes the actions that may cause a phase to change from one state to another.

Changing States	
Action...	Result...
Normal Completion of a Sequence	The phase transitions to the next state as defined by the path in the state transition diagram. For example, if the Running state runs to completion, the phase transitions to the Complete state.
Command is Issued	The phase transitions to the commanded state as specified by the Batch Execution Server or the process operator. For example, if the current phase state is Running, an ABORT command causes the phase to transition to the Aborting state.
Device Failure	The phase transitions to the appropriate state. For example, if a pump that the phase is responsible for fails to start, the phase transitions to the Holding state.  <i>NOTE: The phase programmer can choose what events represent a failure of the phase. Two common examples are a device un-commanded state change and loss of communication between the Batch Execution Server and the phase.</i>

<b>Changing States</b>	
<b>Action...</b>	<b>Result...</b>
Communication Loss	The phase logic behaves as if a HOLD command was issued.

---

## Batch Execution Commands

The PLI responds to the command values issued by the Batch Execution Server or an external interface. This value is stored in the temporary command status register (TEMP\_CD).

The following table lists the commands that the Batch Execution Server or the operator issues to the PLI and the corresponding values that the Batch Execution Server writes to the phase.

<b>Batch Execution Commands</b>		
<b>This Command...</b>	<b>Orders the Phase to...</b>	<b>Value</b>
ABORT	Transition to the Aborting state and is valid in every phase state except for Idle, Complete, and Stopped. If the phase is in any other state, it transitions to the Aborting state to perform any required sequencing.	10
HOLD	Transition from the Running or Restarting state to the Holding state. This command is only valid when the phase is in the Running or Restarting states.	20
STOP	Transition to the Stopping state. This command is only valid when the phase is in one of the following states: <ul style="list-style-type: none"> <li>• Running</li> <li>• Holding</li> <li>• Held</li> <li>• Restarting</li> </ul>	30

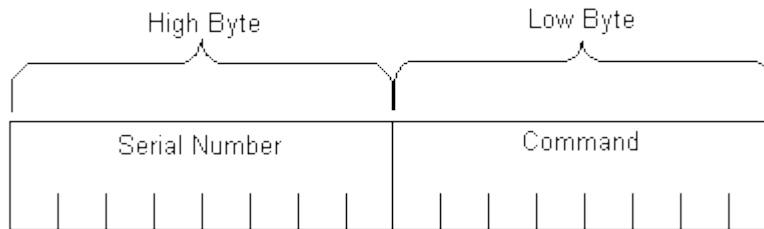
<b>Batch Execution Commands</b>		
<b>This Command...</b>	<b>Orders the Phase to...</b>	<b>Value</b>
RESET	Transition from the Stopped, Aborted, and Complete states back to the Idle state. This command is only valid when the phase is in one of the following states: <ul style="list-style-type: none"> <li>• Stopped</li> <li>• Aborted</li> <li>• Complete</li> </ul>	40
PAUSE	Pause at the next programmed pause transition within its sequencing logic and wait for a RESUME command before proceeding. This command causes the Pausing/Paused mode attribute to be active, indicating the current pause condition of the phase. This command is only valid when the Pausing/Paused mode attribute is inactive.	50
SINGLE STEP	Set the single step mode attribute that allows the phase to transition to the Paused state at each programmed pause transition and wait for the operator to issue a RESUME command.	60
DOWNLOAD	Request the phase to ask for recipe data. Upon execution of this command, the PLI sets the PHASE_DRQ flag. The DOWNLOAD command is used only when the phase is programmed to respond to this command. Other phases will merely reset the command word without responding in any manner. Typically, this command is used for continuous operations that may require setpoint changes during the execution of a batch, such as temperature control, agitation control, and so on.	70

## Batch Execution Command Mechanism

The Batch Execution Server writes commands to the phase logic using a technique that allows the logic to trace individual commands with a high degree of reliability. This technique ensures that the phase:

- Responds only to the lower byte of the command word.
- Clears only the lower byte of the command word.

The following figure illustrates the use of the high-byte and low-byte areas in a 16-bit integer.



*16-Bit Integer*


The High-Byte/Low-Byte technique works as follows:

- The command word in the phase logic consists of a 16-bit integer:
  - High-byte = Serial Number
  - Low-byte = Batch Execution Command
- Batch Execution, when writing a command, simultaneously writes the command into the low-byte and a unique serial number into the high-byte of the Batch Execution command word.
- The phase processes the command, clears out the low-byte of the command, and leaves the high-byte as is.
- Batch Execution waits for the following conditions to be met, guaranteeing that the phase receives and processes the command:
  - High-byte = Serial Number
  - Low-byte = 0

---

## Example: Normal Phase Execution Scenario

To help illustrate the communication dialog between the Batch Execution Server, the PLI, and the phase logic, the following sections describe a possible phase execution scenario. Your actual phase execution may be very different from this example, or it may use only some of the described steps. This scenario, also shown in the following figure, begins when a batch is running and we reach a point in the recipe where a phase needs to execute.

Time	Batch Execution Server	PLI	Phase
	Writes phase unit ID on to PHASE_UN.		
	PHASE_VC = 356		
		TEMP_CD = 100 (Clears the lower byte of PHASE_VC.)	
		PHASE_R = 1 (Clears all other state flags)  PHASE_SI = PHASE_IS  PHASE_ST = 50	
			Executes the running logic. When the running logic is complete, then:  PHASE_RC = 1
		PHASE_ST = 70  PHASE_C = 1 (Clears all other state flags)	
	Reads PHASE_ST		
	PHASE_VC = 552		
		TEMP_CD = 40 (Clears the lower byte of PHASE_VC.)	
		PHASE_ST = 100  PHASE_F = 0  PHASE_IS = 0  PHASE_RQ = 0  PHASE_Qrm = 0  PHASE_I = 1 (Clears all other state flags)	
	Sees that phase has reset. Allows recipe to continue.		

*Normal Phase Execution Scenario*

### **When the Batch Execution Server starts the phase, the Server:**

1. Writes the unit ID of the unit that the phase is running onto the phase's unit tag (PHASE\_UN).
2. Writes a start (100) command (with the serial number) to the phase's command register (PHASE\_VC).

### **The PLI:**

1. Sees the non-zero command in the PHASE\_VC register and copies the command (without the serial number) to the temporary command register (TEMP\_CD).
2. Clears the lower byte of PHASE\_VC.
3. Sees the start (100) command in the temporary command register and does the following:
  - a. Sets the phase's running bit (PHASE\_R).
  - b. Sets the phase's step index register (PHASE\_SI) to the initial step (PHASE\_IS).
  - c. Changes the phase's status register (PHASE\_ST) to running (50).
  - d. Clears all other state flags (PHASE\_RC, PHASE\_AC, PHASE\_HC, and so on).

### **The Phase Logic:**

1. Sees the running bit (PHASE\_R) is set and executes the running logic.
2. When the phase completes, sets the running complete bit (PHASE\_RC).

### **The PLI:**

1. Sees the running complete bit (PHASE\_RC) and does the following:
  - a. Changes the phase's status (PHASE\_ST) to complete (70).
  - b. Clears all other state flags (PHASE\_RC, PHASE\_AC, PHASE\_HC, and so on).

### **The Batch Execution Server:**

1. Is notified of the updated state, through the phase status register (PHASE\_ST). Typically this results in the transition firing so the recipe can continue.
2. When the transition fires, writes a reset (40) command (with the serial number) to the phase's command register (PHASE\_VC).

### **The PLI:**

1. Sees the non-zero command in the PHASE\_VC register and copies the command (without the serial number) to the temporary command register.
2. Sees the reset command (40) in the temporary command register and does the following:
  - a. Changes the phase's status register (PHASE\_ST) to idle (100).
  - b. Clears the phase's failure register (PHASE\_F).
  - c. Clears the phase's step index register (PHASE\_SI).



- d. Clears the phase's request register (PHASE\_RQ) and request data array registers (PHASE\_Qnn).
- e. Sets the phase's idle bit.
- f. Clears all other state flags (PHASE\_R, PHASE\_RC, PHASE\_AC, PHASE\_HC, and so on).

### The Batch Execution Server:

Receives notification that the phase has reset and allows the recipe to continue.

## External Control

Batch Execution lets you control phases from external interfaces. An example of an external interface is an iFIX picture. To prevent conflicts between Batch Execution commands and those from the external interface, the phase logic must select between these two interfaces.

### Understanding the Batch Execution and External Control Relationship

The source of phase control is determined by the owner flag (PHASE\_W). When the External Request (PHASE\_ER) flag is set, PHASE\_W is also set, indicating that an external interface controls the phase. When the Batch Execution Request (PHASE\_VR) flag is set, PHASE\_W is cleared, indicating that Batch Execution controls the phase.

When...	Then...
An External request (PHASE_ER) is received.	The mode transitions to External control. PHASE_ER is cleared after the transition occurs.
A Batch Execution request (PHASE_VR) is received.	The mode transitions to Batch Execution control. PHASE_VR is cleared after the transition occurs.

Commands to the phase to change its state or attribute are sent to two registers: the Batch Execution Command (PHASE\_VC) register and the External Command (PHASE\_EC) register. The Batch Execution Server sends commands to PHASE\_VC; an external interface sends commands to PHASE\_EC. When PHASE\_W is set, the PLI reads PHASE\_EC. When PHASE\_W is cleared, the PLI reads PHASE\_VC.

The command values used in each register are nearly identical. However, external commands do not use a serial number. When the PLI processes external commands, the entire register is cleared. When the PLI processes Batch Execution commands, only the low byte is cleared.

The phase logic owner is determined by the external interface. Because the external interface controls the owner, it is the responsibility of Batch Execution to check PHASE\_W prior to sending any commands. If PHASE\_W is set, Batch Execution will not send any commands or process any requests. It is the responsibility of the external interface to return phase ownership to Batch Execution by setting PHASE\_VR when it is done with the phase.

---

# Understanding the PLI

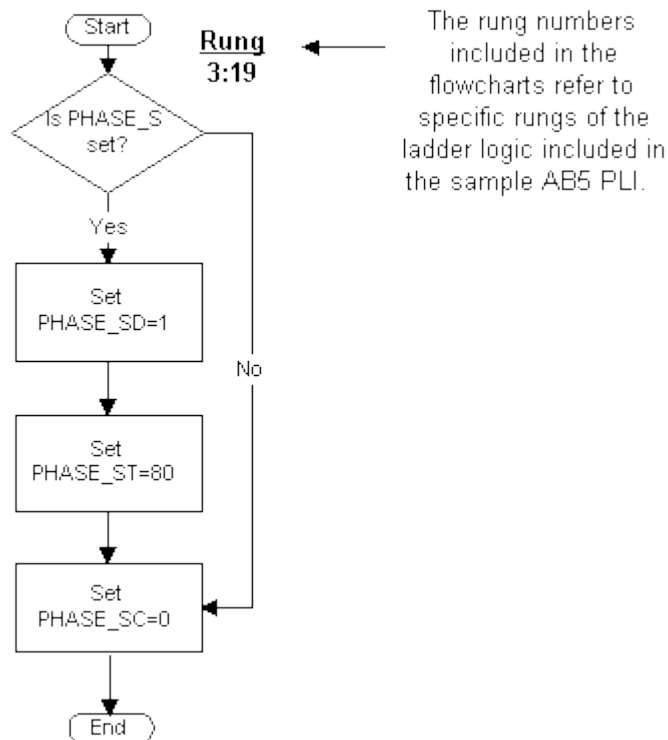
The sections that follow use examples from the AB5 PLI ladder logic in the iFIX Database Tags section to describe:

- The Main Calling program used to invoke the PLI.
- An overview of the PLI.
- A detailed examination of each step of the PLI.

A combination of flowcharts and text is used to explain the steps of the AB5 PLI.

## Flowchart Conventions

Each flowchart includes rung numbers, which refer to the equivalent rungs of ladder in the Sample AB5 PLI Ladder Logic section, as shown in the following figure.

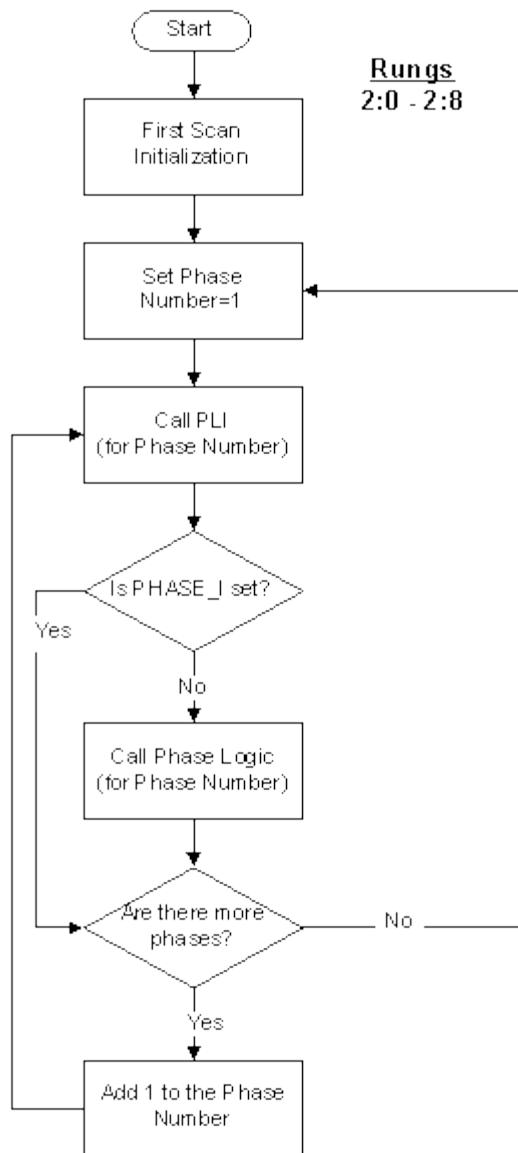


*Flowchart Conventions*

Consider the PLI Overview diagram (in the PLI Overview section) the parent diagram in the sections that follow. The additional child diagrams in the sections that follow present a more detailed view into the steps of the PLI. Each of these child diagrams includes a number, which also represents the specific rung of the ladder logic depicted in the illustration.

## Programming the Main Calling Program

The Main Calling Program invokes the PLI, as illustrated in the following figure.



*Main Calling Program*

The Main Calling Program performs the following tasks:

1. Initialize all variables.
2. Set the number of the first phase that is evaluated to one.
3. Call the PLI for the current phase.
4. Check the Idle State Active (PHASE\_I) flag. If PHASE\_I is not set, call the phase logic for the first phase.

5. If PHASE\_I is set, or if the phase logic has been called for the first phase, check if there are more phases to be evaluated:
  - a. If there is another phase to be evaluated, increment the phase number by one and repeat the process by calling the PLI for the next phase.
  - b. If there are no other phases to be evaluated, set the phase number to one.

This completes the Main Calling program.

### **First Scan Initialization (3:1)**

The first scan initialization of the AB5 PLI occurs during the first scan of the process controller. The following flags may be set as desired for each phase:

**PHASE\_II** – Idle Initialization flag.

**PHASE\_HA** – Hold Active Phases on First Scan flag.

**PHASE\_WC** – Hold on Watchdog Time-out flag.

**PHASE\_RE** – Re-execute Hold flag.

**PHASE\_MR** – Mode Request flag.

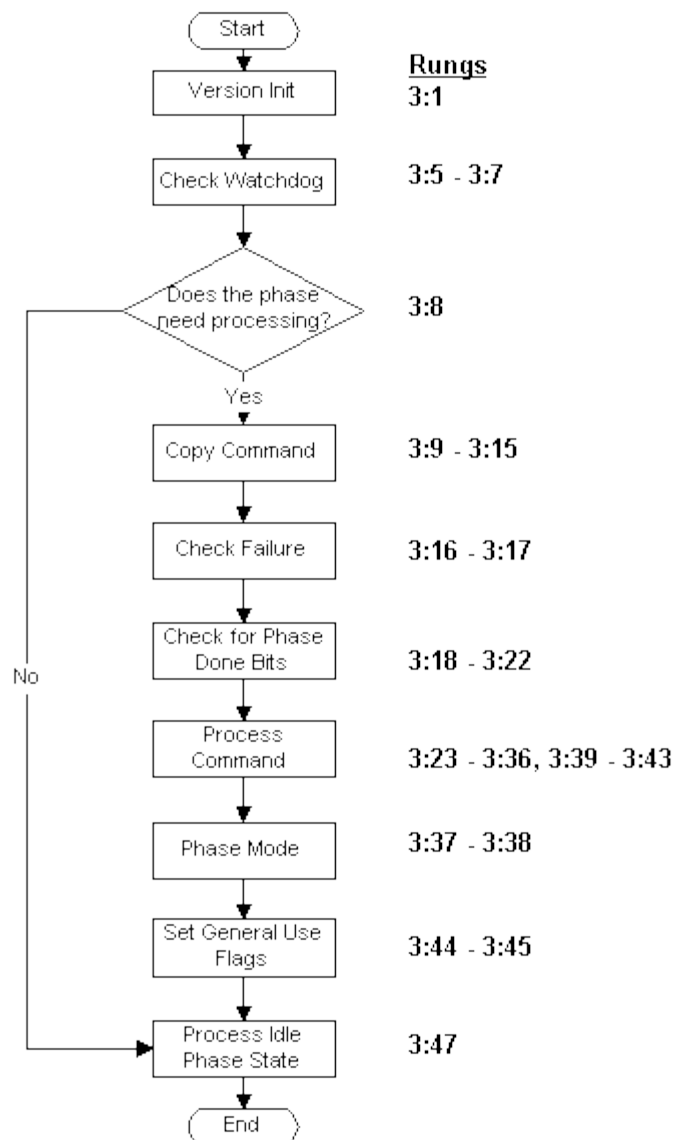
Having the appropriate values loaded for PHASE\_II and PHASE\_HA, especially when first loading the phase logic controller, is essential for the phases to initialize and run as desired.

PHASE\_IS holds the initial value of the Step Index. You can set the value in this register to the desired initial Step Index value, but this value can be set only once in each PLI.

For additional information on these flags and registers, refer to table in the Sample AB5 PLI Ladder Logic section.

## PLI Overview

The following figure presents an overview of the PLI.



*PLI Overview*

As shown in the previous figure, the PLI proceeds almost entirely in a linear fashion through the following tasks:

1. The version number is copied to a register to maintain revision control.
2. Check the Watchdog Flag bit.
3. Check if the phase needs to be processed.
4. Copy commands to the temporary command register (TEMP\_CD).
5. Check if there is a phase failure.

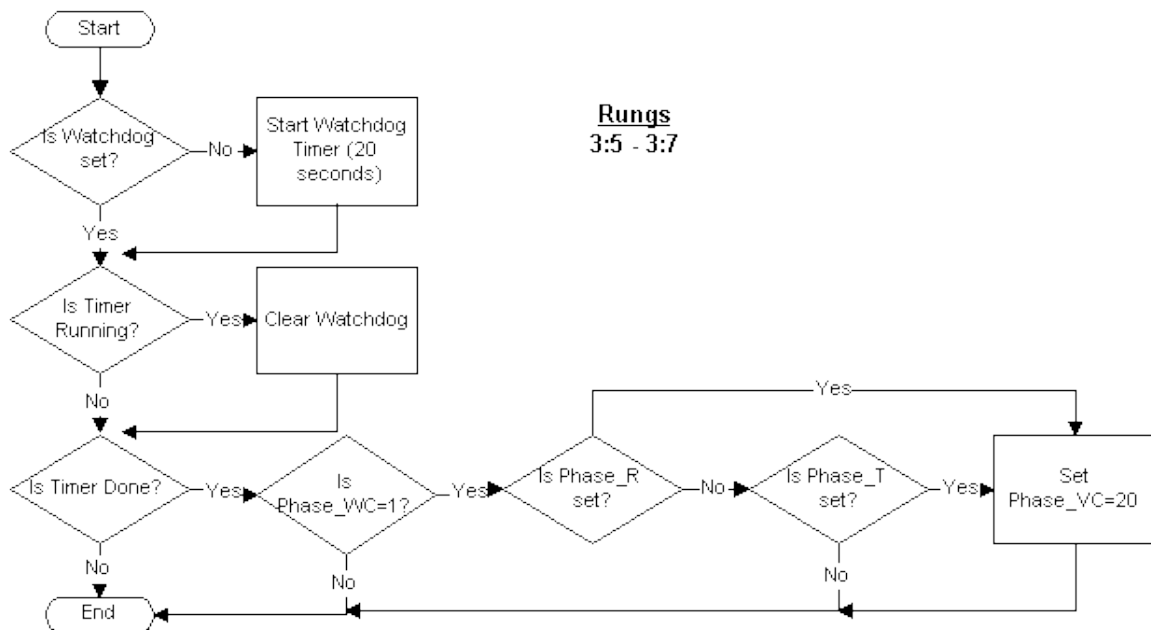
6. Check for the state complete bits (\_AC, \_RC, \_SC, \_TC, \_HC).
7. Process commands.
8. Evaluate the phase mode (Batch Execution or External).
9. Set general use flags.
10. Process the Idle phase state.

The following sections describe each step in the flowchart.

## Checking the Watchdog

The Watchdog register is defined in the process controller. Batch Execution reads this register to ensure that the process controller is available to handle data requests.

The following figure illustrates the process of checking the Watchdog.



*Check Watchdog*

The process of checking the Watchdog includes the following tasks:

1. The PLI checks the Watchdog flag. If this flag is not set, the PLI starts the Watchdog Timer.
2. The PLI verifies that the timer is running. If the timer is running, clear the watchdog and proceed to step 3a.
3. If the timer is not running, check if the timer is done. If the timer is done, perform the following tasks:
  - a. Check the Hold on Watchdog Time-out Configuration (PHASE\_WC) flag. If it is not set, this step of the PLI is complete. If it is set, put the phase into Hold and perform the next task.

- b. Check the Running State Active (PHASE\_R) flag. The PLI sets this flag when the phase state is Running. If this flag is not set, check the Restarting State Active (PHASE\_T) flag. The PLI sets this flag when the phase state is Restarting. If neither of these flags is set, this step of the PLI is complete.
- c. If either PHASE\_R or PHASE\_T are set, set the Batch Execution Command (PHASE\_VC) register to 20. The PLI reads this register and changes the phase state to Holding.

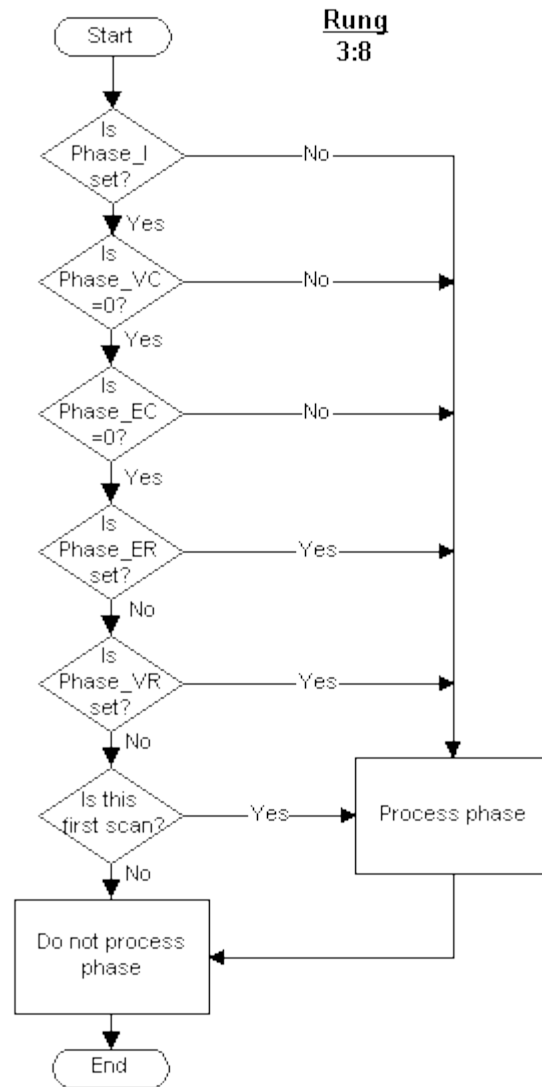
This step of the PLI is now complete.

**NOTE:** *The PLI should issue a Hold command in the case of a watchdog failure. In order to issue a Hold command, the PLI assumes that you either already shut down the Batch Execution Server or that the software put the batch into a Held state for a warm restart. This is necessary, because when the phases are set to Hold you cannot restart the batch from the Batch Execution Client if the procedure, unit procedure, or operation still have a status of Running.*

---

## Processing the Phase

The next step in the PLI evaluates whether to process the phase, as illustrated in the following figure.



*Process Phase*

This step of the PLI performs the following tasks:

1. Check the Idle State Active (PHASE\_I) flag. If this flag is not set, process the phase. If the flag is set, proceed to the next task.
2. Check the Batch Execution Command (PHASE\_VC) register. If there are commands pending, process the phase. If there are no commands pending, proceed to the next task.
3. Check the External Command (PHASE\_EC) register. If there are external commands pending, process the phase. If there are no external commands pending, proceed to the next task.
4. Check the External Request (PHASE\_ER) flag. If this flag is set, the PLI changes ownership

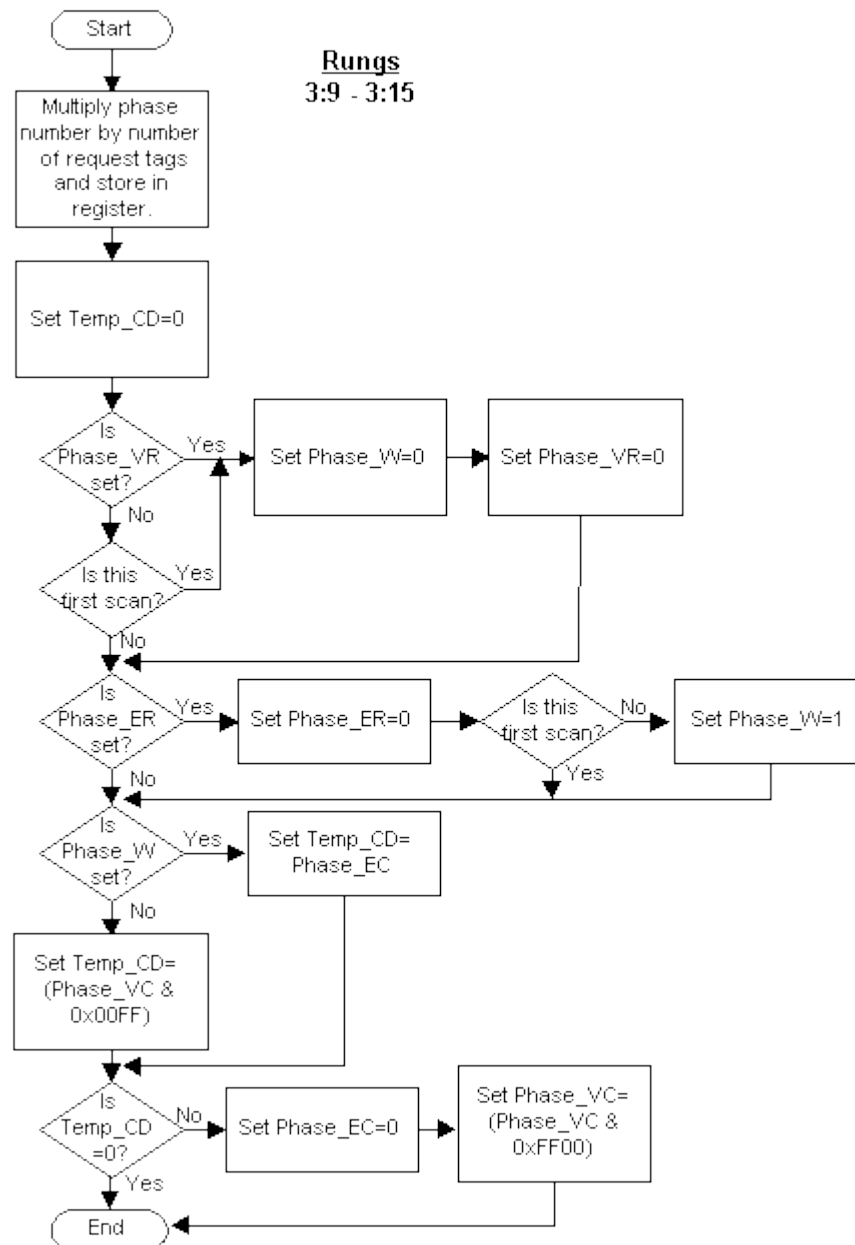


- of the phase to external and processes the phase. If the flag is not set, proceed to the next task.
5. Check the Batch Execution Request (PHASE\_VR) flag. If this flag is set, the PLI changes ownership of the phase from external to Batch Execution and processes the phase. If the flag is not set, proceed to the next task.
  6. Check if this is the first scan. If it is, process the phase. If it is not the first scan, do not process the phase.

This step of the PLI is now complete.

## Copying Commands

The Copy Command step of the PLI copies the command value from the PHASE\_VC (Batch Execution Command) register to the TEMP\_CD (Temporary Command) register. This step is illustrated in .



*Copy Command to TEMP\_CD*

The following tasks occur in this step of the PLI:

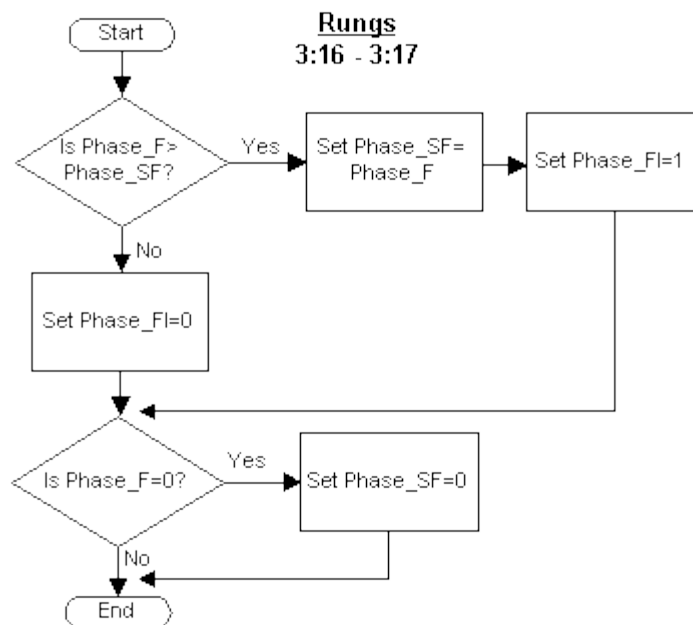
1. Multiply the phase number by the number of request tags and store this value in a temporary register.

2. Clear the TEMP\_CD register.
3. Check the Batch Execution Request (PHASE\_VR) flag. When this flag is set, the PLI changes ownership to Batch Execution.
  - a. If PHASE\_VR *is* set, clear the Owner (PHASE\_W) flag and then clear PHASE\_VR.
  - b. If PHASE\_VR is *not* set, check if this is the first scan, and:
    - If this is the first scan, clear the Owner (PHASE\_W) flag and then clear PHASE\_VR.
    - If it is not the first scan, proceed to the next step.
4. Check the External Request (PHASE\_ER) flag. When this flag is set, the PLI changes ownership of the phase to external.
  - a. If PHASE\_ER *is* set, clear the flag. If this is not the first scan, set PHASE\_W.
  - b. If PHASE\_ER is *not* set or if this is the first scan, proceed to the next step.
5. Check PHASE\_W.
  - a. If PHASE\_W *is* set, set TEMP\_CD equal to the External Command (PHASE\_EC ) register.
  - b. If PHASE\_W is *not* set, copy the low byte of PHASE\_VC to TEMP\_CD.
6. Check if TEMP\_CD is clear. If it is not, clear PHASE\_EC and then clear the low byte of PHASE\_VC. If the TEMP\_CD is clear, this step of the PLI is complete.

---

## Checking for Phase Failure

This step of the PLI checks for any phase failures, as illustrated in the following figure.



*Check Phase Failure*

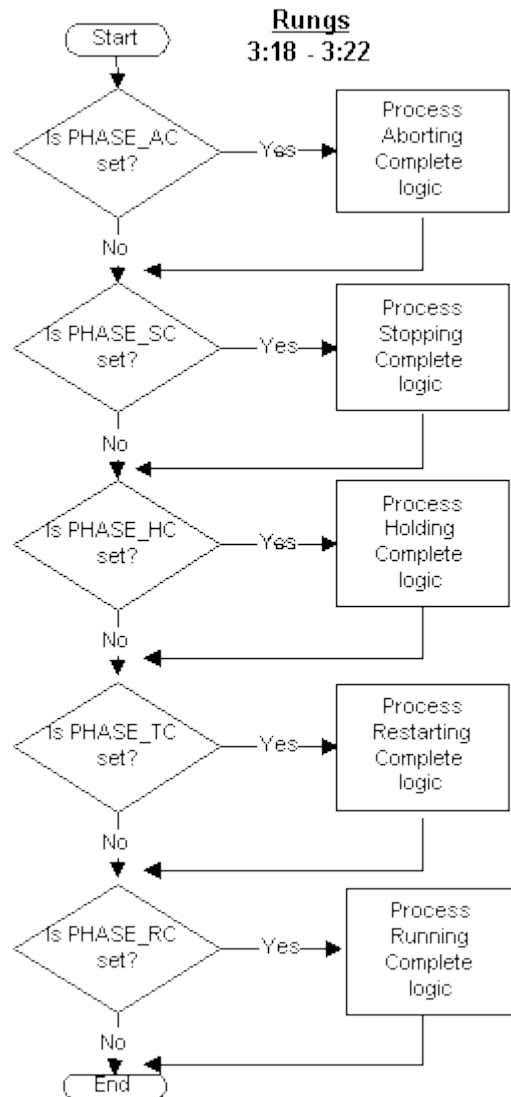
The following tasks occur in this step of the PLI:

1. Check if the Phase Failure (PHASE\_F) register is greater than the Phase Stored Failure (PHASE\_SF) register. If PHASE\_F is not greater, clear the Fail Increase (PHASE\_FI) flag. If PHASE\_F is greater, set the two registers to be equal and then set PHASE\_FI. This flag is set for one scan when the value in PHASE\_F is greater than the value in PHASE\_SF.
2. Check PHASE\_F. If this register is clear, clear PHASE\_SF. If PHASE\_F is not clear, this step of the PLI is complete.

---

## Checking Phase Flags

This step of the PLI checks the state of the phase by checking a series of phase flags, as shown in the following figure.



*Check Phase Flags*

The phase logic sets these flags to indicate to the PLI that the associated logic has run to completion. The PLI: (1) monitors the flags and (2) processes the state's completion logic when the flag is set.

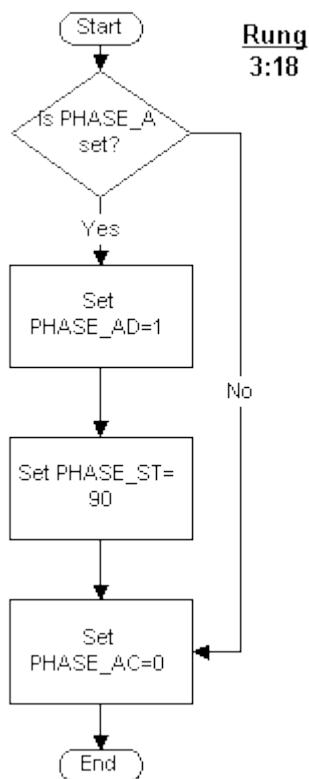
This step of the PLI performs the following tasks:

1. Check the Aborting Complete (PHASE\_AC) flag. If the flag is set, process the Aborting complete logic. If the flag is not set, proceed to the next task.
2. Check the Stopping Complete (PHASE\_SC) flag. If the flag is set, process the Stopping complete logic. If the flag is not set, proceed to the next task.
3. Check the Holding Complete (PHASE\_HC) flag. If the flag is set, process the Holding complete logic. If the flag is not set, proceed to the next task.
4. Check the Restarting Complete (PHASE\_TC) flag. If the flag is set, process the Restarting complete logic. If the flag is not set, proceed to the next task.
5. Check the Running Complete (PHASE\_RC) flag. If the flag is set, process the Running complete logic.

The following sections describe the PLI's process complete states.

### Process Aborting Complete

The following figure provides a closer look at the Process Aborting Complete logic, which is one step of the Checking Phase Flags portion of the PLI.



*Process Aborting Complete*

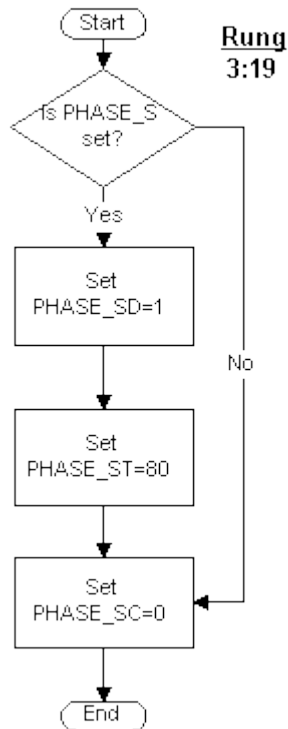
This step of the PLI performs the following tasks:

1. Check the Aborting State Active (PHASE\_A) flag. The PLI sets this flag when the phase state is Aborting. If PHASE\_A is not set, proceed to step 2. If PHASE\_A is set, perform the following tasks:
  - a. Set the Aborted State Active (PHASE\_AD) flag. The PLI sets this flag when the phase state is Aborted.
  - b. Set the Phase Status (PHASE\_ST) register to 90, which is the Aborted value. The PLI processes the command and changes phase states accordingly.
2. Clear the Aborting Complete (PHASE\_AC) flag. The phase logic sets this flag to indicate to the PLI that the Aborting logic has run to completion. The PLI monitors this flag and transitions to the Aborted state when the flag is set.

The Aborting Complete process is complete.

## Process Stopping Complete

The following figure provides a closer look at the Process Stopping Complete logic, which is one step of the Checking Phase Flags portion of the PLI.



*Process Stopping Complete*

This step of the PLI performs the following tasks:

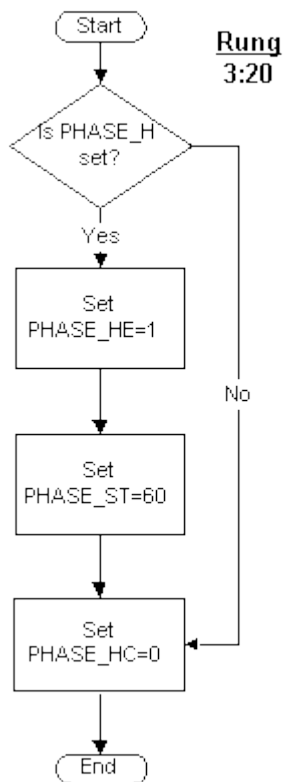
1. Check the Stopping State Active (PHASE\_S) flag. The PLI sets this flag when the phase state is Stopping. If PHASE\_S is not set, proceed to step 2. If PHASE\_S is set, perform the following tasks:

- a. Set the Stopped State Active (PHASE\_SD) flag. The PLI sets this flag when the phase state is Stopped.
  - b. Set the Phase Status (PHASE\_ST) register to 80, which is the Stopped value. The PLI processes commands and changes phase states accordingly.
2. Clear the Stopping Complete (PHASE\_SC) flag. The phase logic sets this flag to indicate to the PLI that the Stopping logic has run to completion. The PLI monitors this flag and transitions to the Stopped state when the flag is set.

This Stopping Complete process is complete.

## Process Holding Complete

The following figure provides a closer look at the Process Holding Complete logic, which is one step of the Checking Phase Flags portion of the PLI.



*Process Holding Complete*

This step of the PLI performs the following tasks:

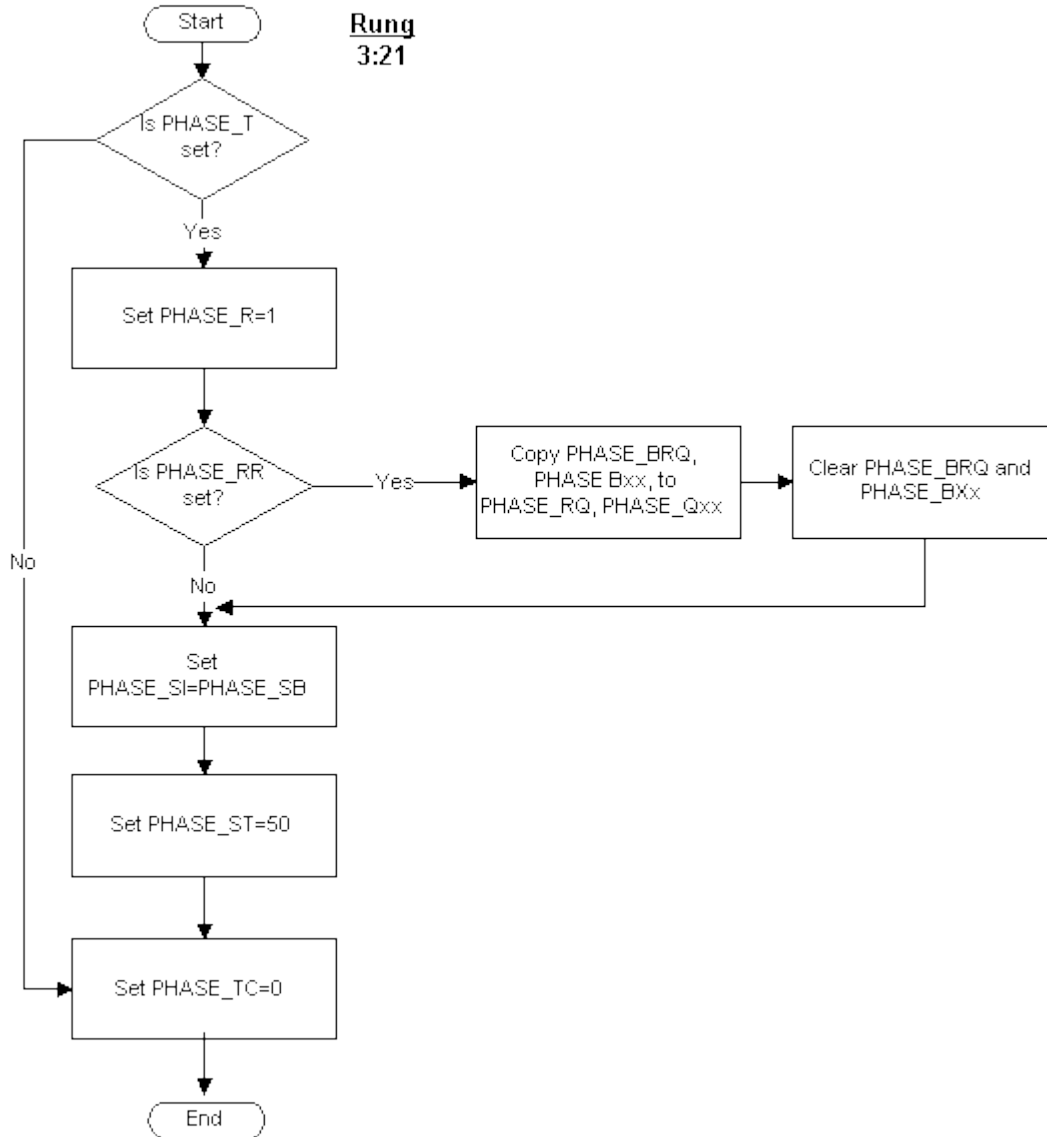
1. Check the Holding State Active (PHASE\_H) flag. The PLI sets this flag when the phase state is Holding. If PHASE\_H is not set, proceed to step 2. If PHASE\_H is set, perform the following tasks:
  - a. Set the Held State Active (PHASE\_HE) flag. The PLI sets this flag when the phase state is Held.
  - b. Set the Phase Status (PHASE\_ST) file to 60, which is the Held value. The PLI processes commands and changes phase states accordingly.

2. Clear the Holding Complete (PHASE\_HC) flag. The phase logic sets this flag to indicate to the PLI that the Holding logic has run to completion. The PLI monitors this flag and transitions to the Held state when the flag is set.

This Holding Complete process is complete.

## Process Restarting Complete

The following figure provides a closer look at the Process Restarting Complete logic, which is one step of the Checking Phase Flags portion of the PLI.



*Process Restarting Complete*



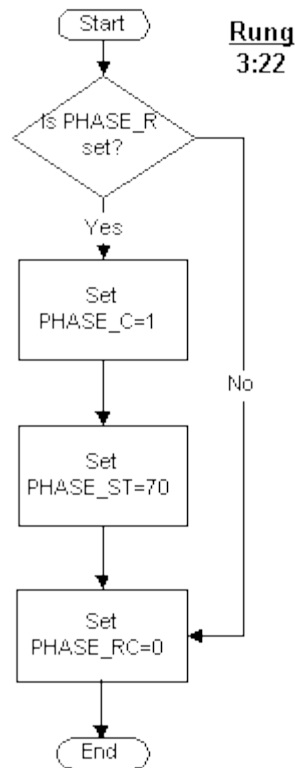
This step of the PLI performs the following tasks:

1. Check the Restarting State Active (PHASE\_T) flag. The PLI sets this flag when the phase state is Restarting. If PHASE\_T is not set, proceed to step 2. If PHASE\_T is set, perform the following tasks:
  - a. Set the Running State Active (PHASE\_R) flag. The PLI sets this flag when the phase state is Running.
  - b. Check the Restore Request (PHASE\_RR) flag. If the flag is set, copy the Phase Request Data Buffer (PHASE\_BRQ) to the Phase Request Data Array (PHASE\_RQ) and then proceed to the next task. Proceed directly to the next task if the flag is not set.
  - c. Clear the Phase Request Data Buffer PHASE\_BRQ and PHASE\_Bxx.
  - d. Set the Phase Step Index (PHASE\_SI) register equal to the Phase Step Buffer (PHASE\_SB) register.
  - e. Set the Phase Status (PHASE\_ST) register to 50, which is the Running value. The PLI processes commands and changes phase states accordingly.
2. Clear the Restarting Complete (PHASE\_TC) flag. The phase logic sets this flag to indicate to the PLI that the Restarting logic has run to completion. The PLI monitors this flag and transitions to the Running state when the flag is set.

This Restarting Complete process is complete.

## Process Running Complete

The following figure provides a closer look at the Process Running Complete logic, which is one step of the Checking Phase Flags portion of the PLI.



*Process Running Complete*

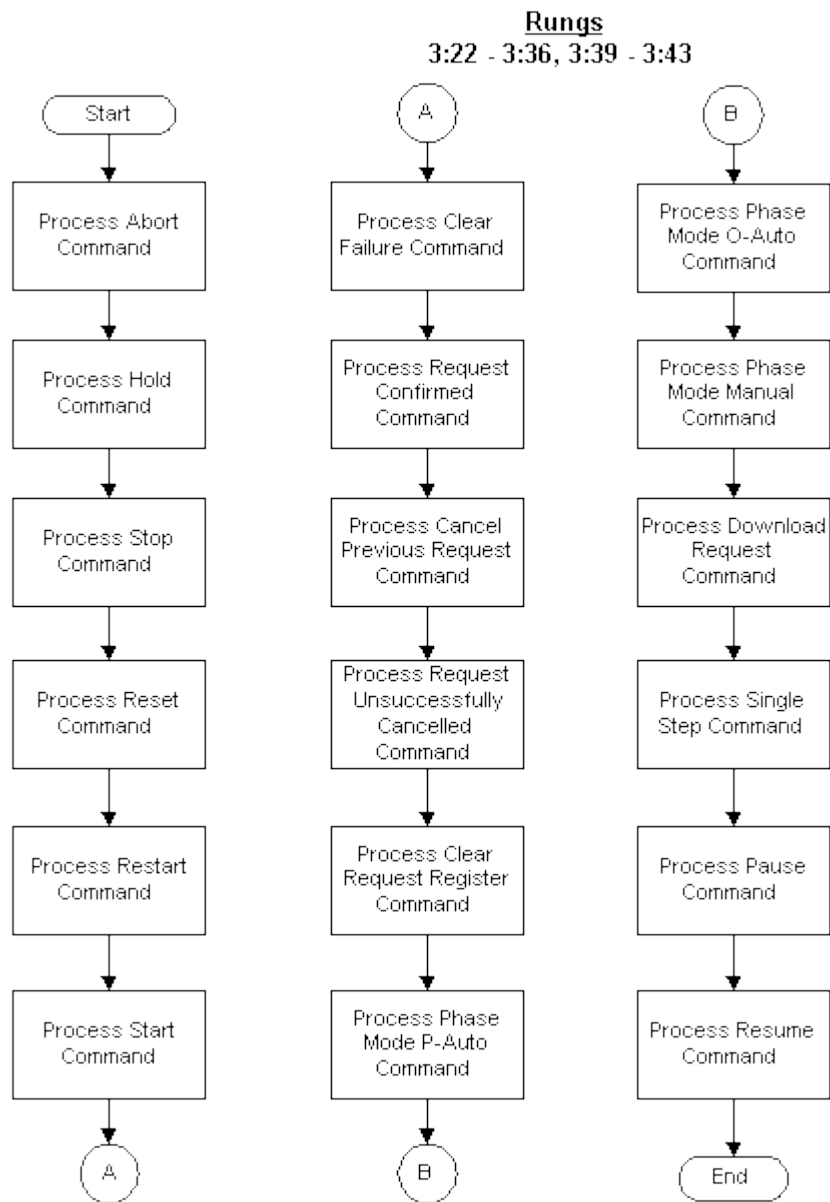
This step of the PLI performs the following tasks:

1. Check the Running State Active (PHASE\_R) flag. The PLI sets this flag when the phase state is Running. If PHASE\_R is not set, proceed to step 2. If PHASE\_R is set, perform the following tasks:
  - a. Set the Complete State Active (PHASE\_C) flag. The PLI sets this flag when the phase state is Aborted.
  - b. Set the Phase Status (PHASE\_ST) register to 70, which is the Complete value. The PLI processes commands and changes phase states accordingly.
2. Clear the Running Complete (PHASE\_RC) flag. The phase logic sets this flag to indicate to the PLI that the Running logic has run to completion. The PLI monitors this flag and transitions to the Complete state when the flag is set.

This Running Complete process is complete.

## Processing Commands

This step of the PLI processes all commands. The following figure illustrates an overview of how the PLI processes commands, in the order that the commands are processed. Refer to the Processing Commands in the PLI section for a detailed discussion of how the PLI processes specific commands.

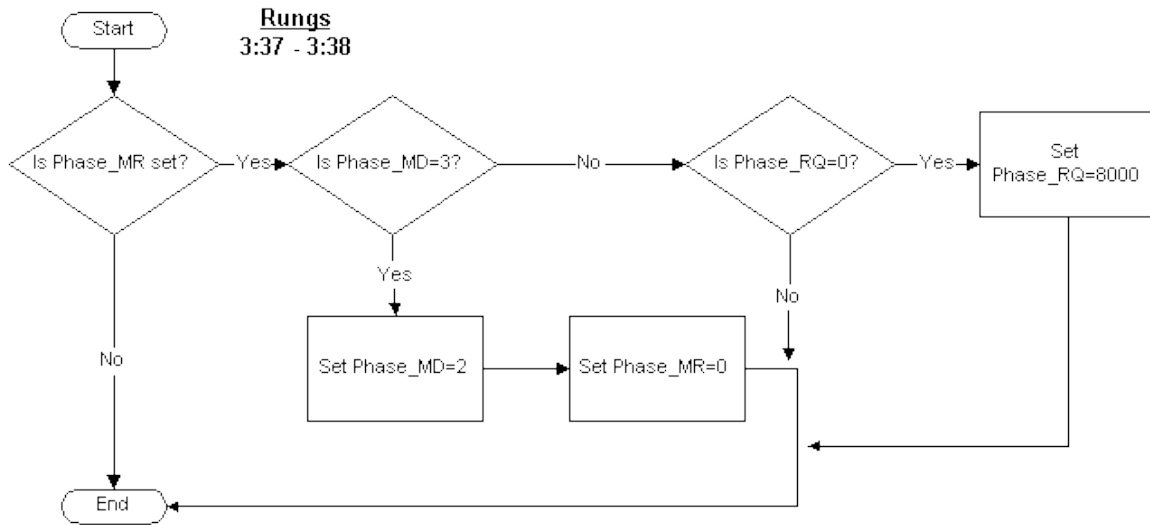


*Process Commands*

---

## Phase Mode

The Phase Mode portion of the PLI allows a user to request a change in the phase mode. The following figure illustrates the Phase Mode portion of the PLI.



### *Phase Mode*

This step of the PLI performs the following tasks:

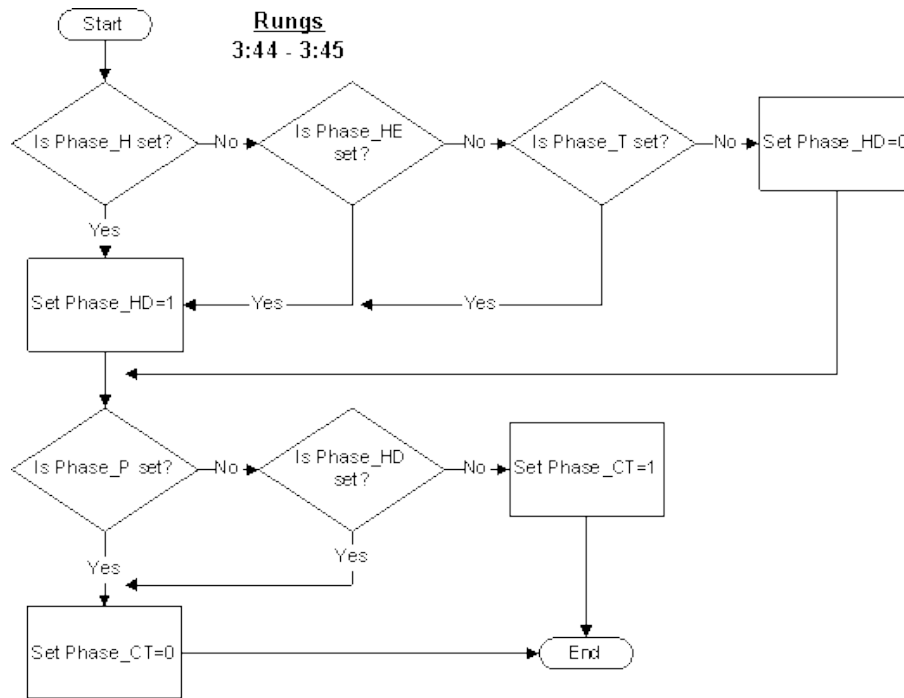
1. Check the Mode Request (PHASE\_MR) flag. If this flag is not set, this step of the PLI terminates. If PHASE\_MR is set, proceed to the next step.
2. Check the Phase Mode (PHASE\_MD) register.
  - a. If PHASE\_MD is set to 3, which indicates Manual mode, set the register to 2, which indicates O-Auto mode, and clear PHASE\_MR. This step of the PLI terminates.
  - b. If PHASE\_MD is not set to 3, check the Request Data (PHASE\_RQ) register. If PHASE\_RQ is not clear, this step of the PLI terminates. If PHASE\_RQ is clear, set the register to 8000, which is a request to put the phase into Manual mode.

**NOTE:** *The Batch Execution Server does not respond to the 8000 request.*

The processing of the Phase Mode is complete.

## Setting Flags

The following figure illustrates how general use flags are set in the PLI. The general use flags are the Holding Indicator flag and the Continue flag. For more information on these particular flags, refer to the Sample AB5 PLI Ladder Logic section.



*Set Flags*

This step of the PLI performs the following tasks:

1. Check the following flags:
  - Holding State Active (PHASE\_H) flag.
  - Held State Active (PHASE\_HE) flag.
  - Restarting State Active (PHASE\_T) flag.

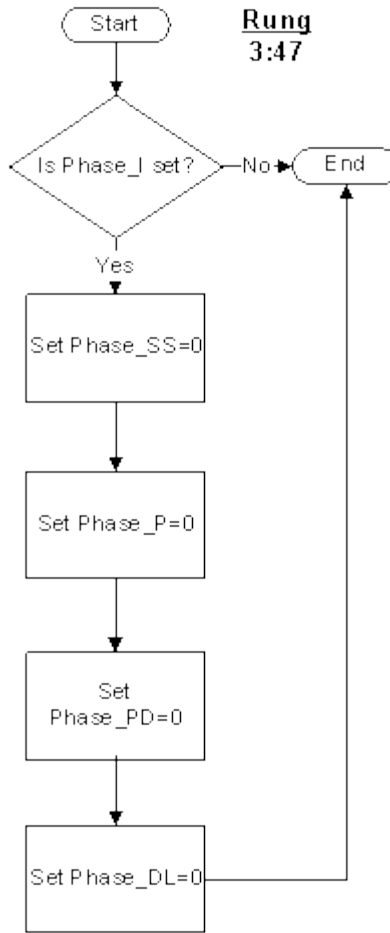
If none of the flags are set, clear the Holding Indicator (PHASE\_HD) flag. If any of the flags are set, set PHASE\_HD.
2. Check the Pause (PHASE\_P) flag.
  - a. If PHASE\_P is set, clear the Continue (PHASE\_CT) flag.
  - b. If PHASE\_P is not set, proceed to the following step.
3. Check PHASE\_HD.
  - a. If PHASE\_HD is set, clear PHASE\_CT.
  - b. If PHASE\_HD is not set, set PHASE\_CT.

The Setting Flags portion of the PLI is complete.

---

## Processing the Idle Phase State

The final step in the PLI processes the Idle phase state, as illustrated in the following figure.



*Process Idle Phase*

This step of the PLI performs the following tasks:

1. Check the Idle State Active (PHASE\_I) flag. If this flag is not set, this step of the PLI terminates. If PHASE\_I is set, proceed to the next step.
2. Clear the Single Step (PHASE\_SS) flag.
3. Clear the Pause (PHASE\_P) flag.
4. Clear the Paused (PHASE\_PD) flag.

---

# Processing Commands in the PLI

The sections that follow present a high level of detail of the various commands that are processed in the PLI:

- Operator-Issued Commands
- Handshaking Commands
- Mode Change Commands
- Download Request Command Process
- Phase Debugging Commands

These sections use a combination of flowcharts and text to explain this section of the AB5 PLI.

---

## Overview: Processing Commands

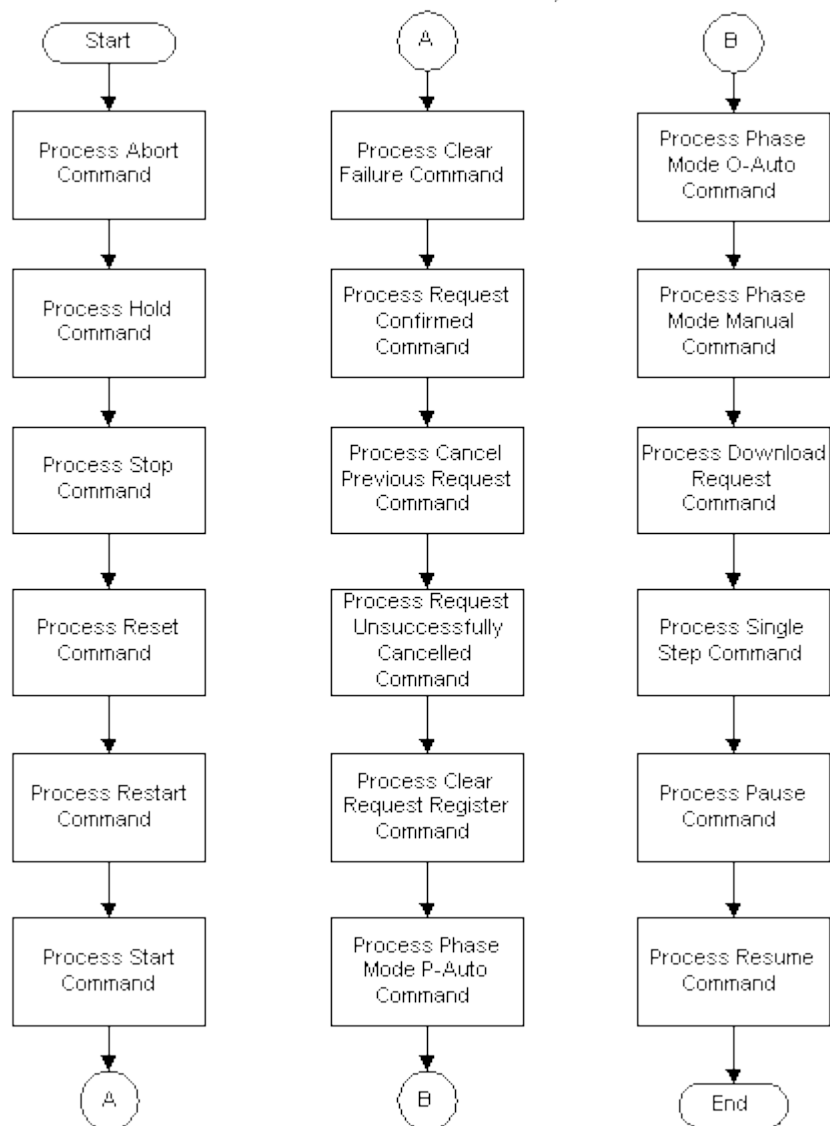
The Process Commands step of the PLI processes all commands. The following figure illustrates an overview of how the PLI processes commands in the order that the commands are processed. The illustrations in the sections that follow are children to this parent figure.

The commands are grouped into the following categories:

- Operator-issued commands
- Handshaking commands
- Mode change commands
- Download request command
- Phase debugging commands

The following sections describe each of the steps in the following figure in greater detail.

**Rungs**  
3:22 - 3:36, 3:39 - 3:43



*Processing Commands in the PLI*

---

## Operator-Issued Commands

The following sections describe how the PLI processes operator-issued commands:

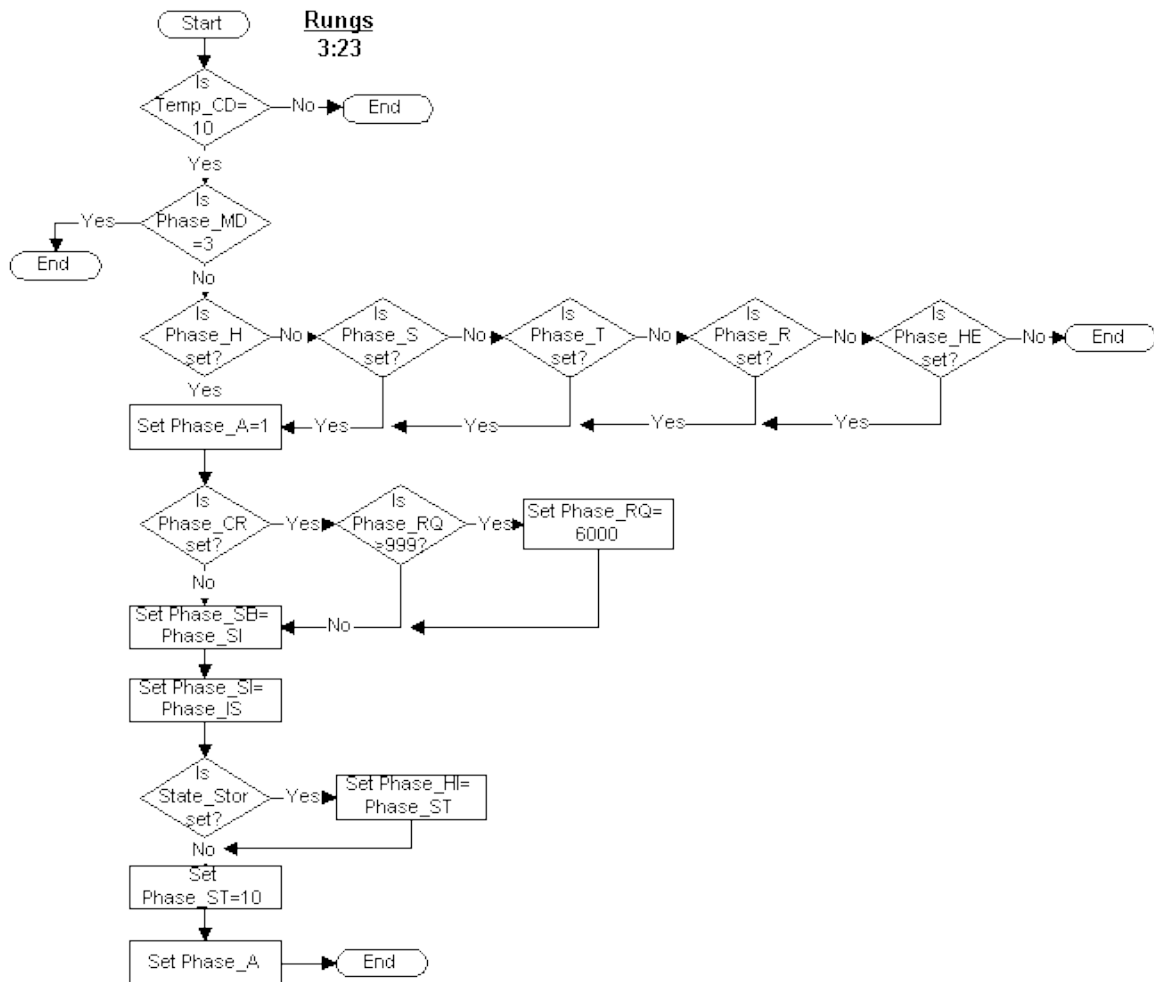
- Process Abort Command
- Process Hold Command
- Process Stop Command
- Process Reset Command
- Process Restart Command



- Process Start Command
- Process Clear Failure Command

## Process Abort Command

The following figure illustrates how the PLI processes the Abort command. This figure is a child to the parent illustration, in the Overview: Processing Commands section.



*Process Abort Command*

The Abort command process performs the following tasks:

1. Check the TEMP\_CD register. If it contains the Abort phase command value (10), proceed to the next step. If it does not, this step of the PLI terminates.
2. Check the Phase Mode (PHASE\_MD) register. If PHASE\_MD contains a value of 3, the value for Manual mode, this step of the PLI terminates. If the value is not 3, proceed to the next step.
3. Check the following flags:
  - Holding State Active (PHASE\_H) flag

- Stopping State Active (PHASE\_S) flag
- Restarting State Active (PHASE\_T) flag
- Running State Active (PHASE\_R) flag
- Held State Active (PHASE\_HE) flag

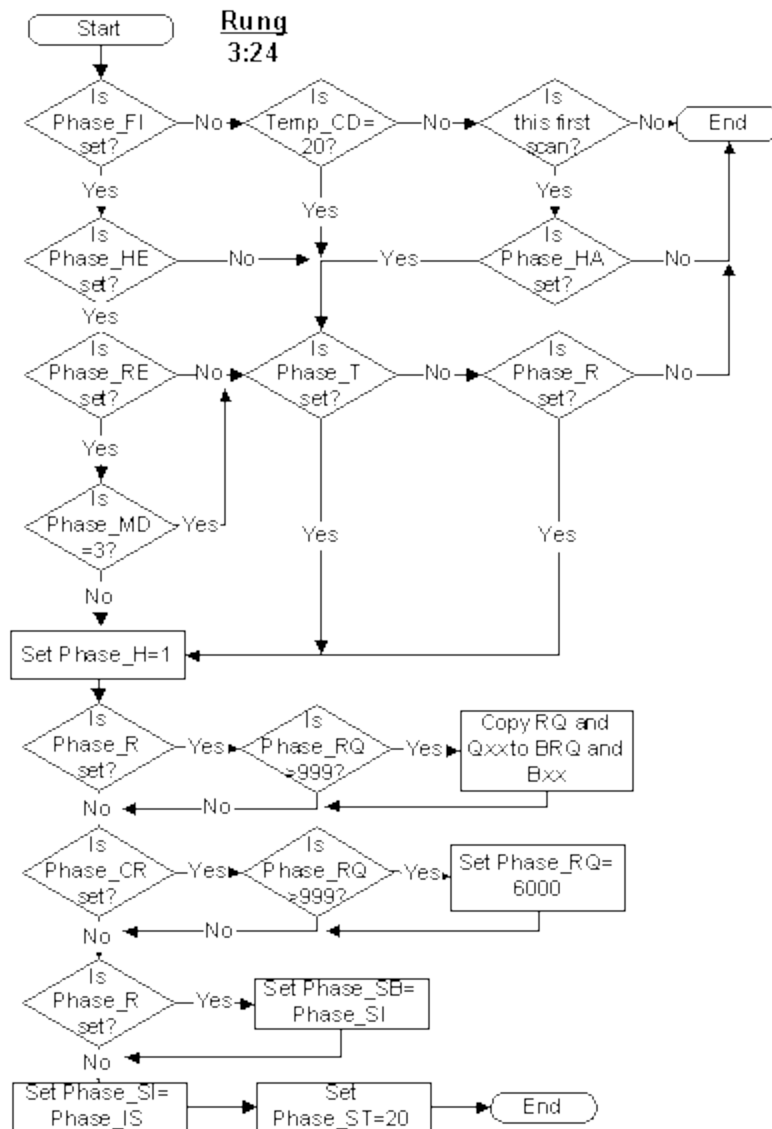
If any of these flags are set, proceed to the next step. If none of the flags are set, this step of the PLI terminates.

4. Set the Aborting State Active (PHASE\_A) flag.
5. Check the Clear (PHASE\_CR) flag.
  - a. If PHASE\_CR is set, check the value of the PHASE\_RQ register. If this register contains a value that is greater than 999, set PHASE\_RQ to 6000 to cancel the previous request. This step clears requests that the Batch Execution Server has not processed.
  - b. If the PHASE\_CR is not set, or if PHASE\_RQ is not greater than 999, proceed to the next step.
6. Set the Phase Step Buffer (PHASE\_SB) register equal to the Phase Step Index (PHASE\_SI) register.
7. Set PHASE\_SI equal to the register containing the initial value of the Step Index (PHASE\_IS).
8. Check the STATE\_STOR flag. This flag determines whether the Hold Index register will hold the value of the last state. If this flag is set, copy the value of the current state to the Phase Hold Index (PHASE\_HI) register before transitioning to the new state. If the flag is not set, proceed to the next step.
9. Set the Phase Status (PHASE\_SI) register to 10, the Aborting value.
10. Set PHASE\_A.

The processing of the Abort command is complete.

## Process Hold Command

The following figure illustrates how the PLI processes the Hold command. This figure is a child to the parent illustration, in the Overview: Processing Commands section.



### *Process Hold Command*

The Hold command process performs the following tasks:

1. Check the Fail Increase (PHASE\_FI) flag.
  - If PHASE\_FI is not set, check the value of the TEMP\_CD register:

<b>If...</b>	<b>And...</b>	<b>Then...</b>
TEMP_CD is not equal to 20	This is not the first scan	This step of the PLI terminates.
<ul style="list-style-type: none"> <li>TEMP_CD is not equal to 20</li> </ul> and <ul style="list-style-type: none"> <li>This is the first scan</li> </ul>	PHASE_HA is not set	This step of the PLI terminates.
<ul style="list-style-type: none"> <li>TEMP_CD is not equal to 20</li> </ul> and <ul style="list-style-type: none"> <li>This is the first scan</li> </ul>	<ul style="list-style-type: none"> <li>PHASE_HA is set</li> </ul> and <ul style="list-style-type: none"> <li>PHASE_T is set</li> </ul>	Proceed to step 4.
<ul style="list-style-type: none"> <li>TEMP_CD is not equal to 20</li> </ul> and <ul style="list-style-type: none"> <li>This is the first scan</li> </ul>	<ul style="list-style-type: none"> <li>PHASE_HA is set</li> </ul> and <ul style="list-style-type: none"> <li>PHASE_T is not set</li> </ul> and <ul style="list-style-type: none"> <li>PHASE_R is set</li> </ul>	Proceed to step 4.
<ul style="list-style-type: none"> <li>TEMP_CD is not equal to 20</li> </ul> and <ul style="list-style-type: none"> <li>This is the first scan</li> </ul>	<ul style="list-style-type: none"> <li>PHASE_HA is set</li> </ul> and <ul style="list-style-type: none"> <li>PHASE_T is not set</li> </ul> and <ul style="list-style-type: none"> <li>PHASE_R is not set</li> </ul>	This step of the PLI terminates.
TEMP_CD equals 20	PHASE_T is set	Proceed to step 4.
TEMP_CD equals 20	<ul style="list-style-type: none"> <li>PHASE_T is not set</li> </ul> and <ul style="list-style-type: none"> <li>PHASE_R is set</li> </ul>	Proceed to step 4.
TEMP_CD equals 20	<ul style="list-style-type: none"> <li>PHASE_T is not set</li> </ul> and <ul style="list-style-type: none"> <li>PHASE_R is not set</li> </ul>	This step of the PLI terminates.

- If PHASE\_FI is set, check the Held State Active (PHASE\_HE ) flag. If PHASE\_HE is set, proceed to the next step.

<b>If...</b>	<b>And...</b>	<b>Then...</b>
PHASE_HE is not set	PHASE_T is set	Proceed to step 4.
PHASE_HE is not set	<ul style="list-style-type: none"> <li>• PHASE_T is not set</li> <li>and</li> <li>• PHASE_R is set</li> </ul>	Proceed to step 4.
PHASE_HE is not set	<ul style="list-style-type: none"> <li>• PHASE_T is not set</li> <li>and</li> <li>• PHASE_R is not set</li> </ul>	This step of the PLI terminates.

2. Check the Re-execute Hold (PHASE\_RE) flag. If PHASE\_RE is set, proceed to the next step.

<b>If...</b>	<b>And...</b>	<b>Then...</b>
PHASE_RE is not set	PHASE_T is set	Proceed to step 4.
PHASE_RE is not set	<ul style="list-style-type: none"> <li>• PHASE_T is not set</li> <li>and</li> <li>• PHASE_R is set</li> </ul>	Proceed to step 4.
PHASE_RE is not set	<ul style="list-style-type: none"> <li>• PHASE_T is not set</li> <li>and</li> <li>• PHASE_R is not set</li> </ul>	This step of the PLI terminates.

3. Check the value of the Batch Execution Phase Mode (PHASE\_MD) register. If PHASE\_MD is not equal to 3 (the value for Manual Mode), proceed to the next step.

<b>If...</b>	<b>And...</b>	<b>Then...</b>
PHASE_MD = 3	PHASE_T is set	Proceed to step 4.
PHASE_MD = 3	<ul style="list-style-type: none"> <li>• PHASE_T is not set</li> <li>and</li> <li>• PHASE_R is set</li> </ul>	Proceed to step 4.
PHASE_MD = 3	<ul style="list-style-type: none"> <li>• PHASE_T is not set</li> <li>and</li> <li>• PHASE_R is not set</li> </ul>	This step of the PLI terminates.

4. Set the Holding State Active (PHASE\_H) flag.
5. Check the Running State Active (PHASE\_R) flag. If PHASE\_R is not set, proceed to the next step.

If...	And...	Then...
PHASE_R is set	PHASE_RQ > 999 <i>NOTE: A value greater than 999 indicates that the request has not been processed by the Batch Execution Server.</i>	Copy PHASE_RQ to PHASE_BRQ.
PHASE_R is set	PHASE_RQ < 999	Proceed to the next step.

6. Check the Clear (PHASE\_CR) flag. If PHASE\_CR is not set, proceed to the next step.

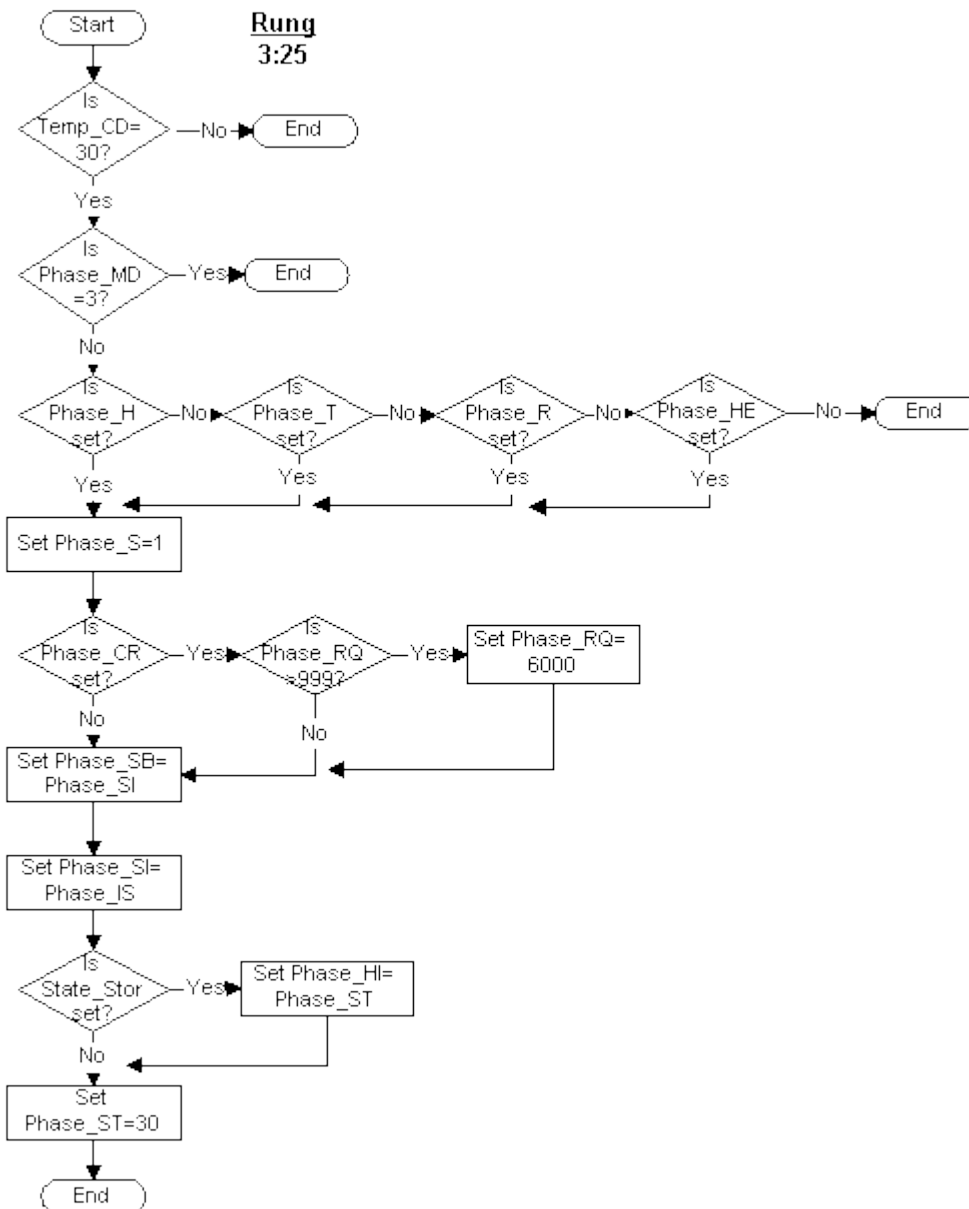
If...	And...	Then...
PHASE_CR is set	PHASE_RQ > 999	Set PHASE_RQ = 6000.
PHASE_CR is set	PHASE_RQ < 999	Proceed to the next step.

7. Check the Running State Active (PHASE\_R) flag.
  - If PHASE\_R is set, set the Phase Step Buffer (PHASE\_SB) register equal to the Phase Step Index (PHASE\_SI) register. Proceed to the next step.
  - If PHASE\_R is not set, proceed to the next step.
8. Set PHASE\_SI equal to the register containing the initial value of the Step Index (PHASE\_IS).
9. Set the Phase Status (PHASE\_ST) register to 20, the Holding value.

The processing of the Hold command is complete.

## Process Stop Command

The following figure illustrates how the PLI processes the Stop command. This figure is a child to the parent illustration, in the Overview: Processing Commands section.



### *Process Stop Command*

The Stop command process performs the following tasks:

1. Check the TEMP\_CD register. If this register contains a value of 30, the Stop phase command, proceed to the next step. If TEMP\_CD does not contain a value of 30, this step of the PLI terminates.
2. Check the Phase Mode (PHASE\_MD) register. If this register contains a value of 3, the Manual mode value, this step of the PLI terminates. If it does not contain a value of 3, proceed to the next step.
3. Check the following flags:
  - Holding State Active (PHASE\_H) flag

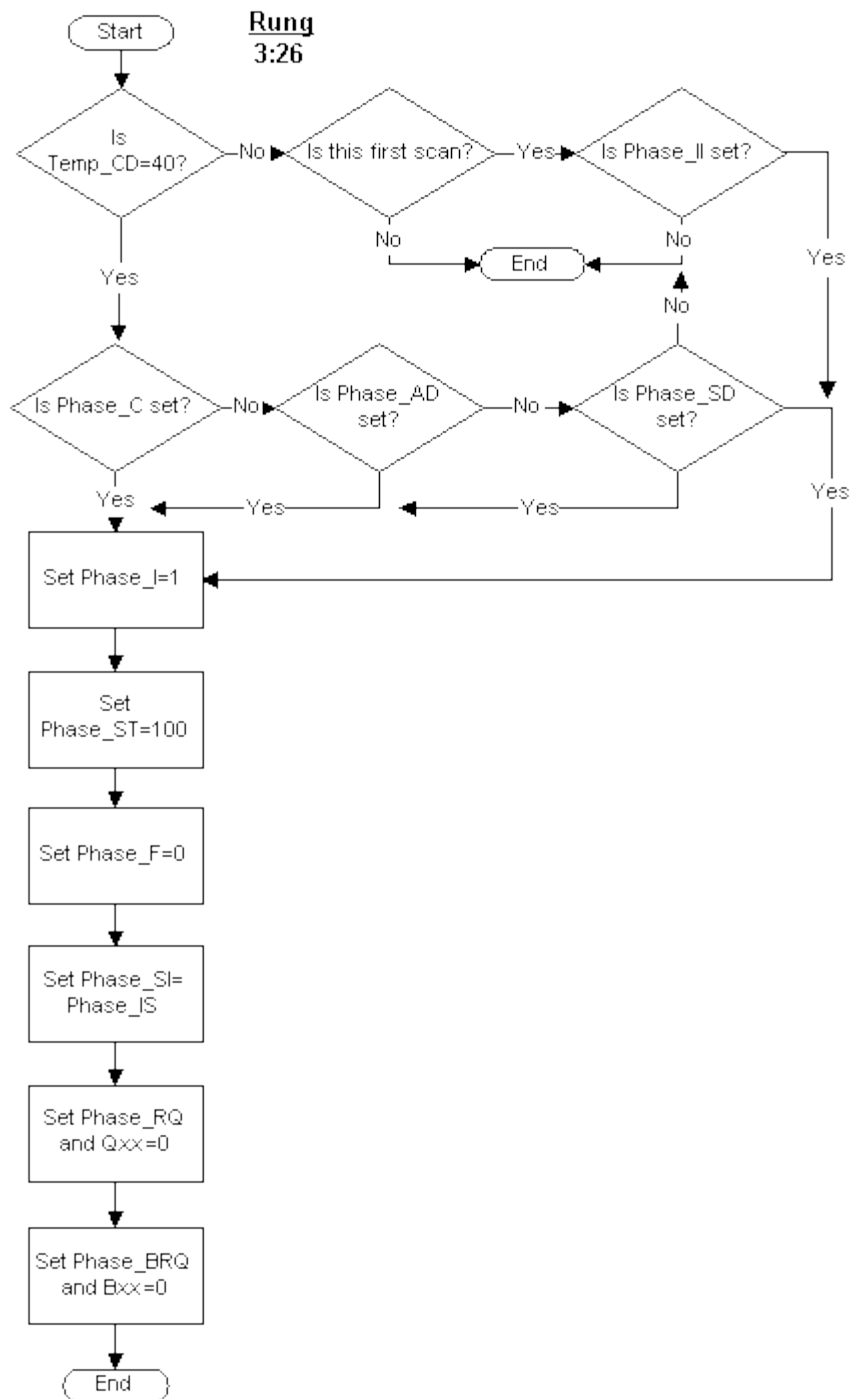
- Restarting State Active (PHASE\_T) flag
  - Running State Active (PHASE\_R) flag
  - Held State Active (PHASE\_HE) flag
  - If none of the flags are set, this step of the PLI terminates. If any of the flags are set, proceed to the next step.
4. Set the Stopping State Active (PHASE\_S) flag.
  5. Check the Clear (PHASE\_CR) flag.
    - a. If PHASE\_CR is set, check the value of the PHASE\_RQ register. If this register contains a value that is greater than 999, set PHASE\_RQ to 6000 to cancel the previous request. This step clears any requests that the Server has not processed.
    - b. If the Clear flag is not set, or if PHASE\_RQ is not greater than 999, proceed to the next step.
  7. Set the Phase Step Buffer (PHASE\_SB) register equal to the Phase Step Index (PHASE\_SI) register.
  8. Set PHASE\_SI equal to the register containing the initial value of the Step Index (PHASE\_IS).
  9. Check the STATE\_STOR flag. This flag determines whether the Hold Index (PHASE\_HI) register will hold the value of the last state. If this flag is set, copy the value of the current state to PHASE\_HI before transitioning to the new state. If the flag is not set, proceed to the next step.
  10. Set the Phase Status (PHASE\_ST) register to 30, the Stopping value.

The processing of the Stop command is complete.



## Process Reset Command

The following figure illustrates how the PLI processes the Reset command. This figure is a child to the parent illustration, in the Overview: Processing Commands section.



*Process Reset Command*

The Reset command process performs the following tasks:

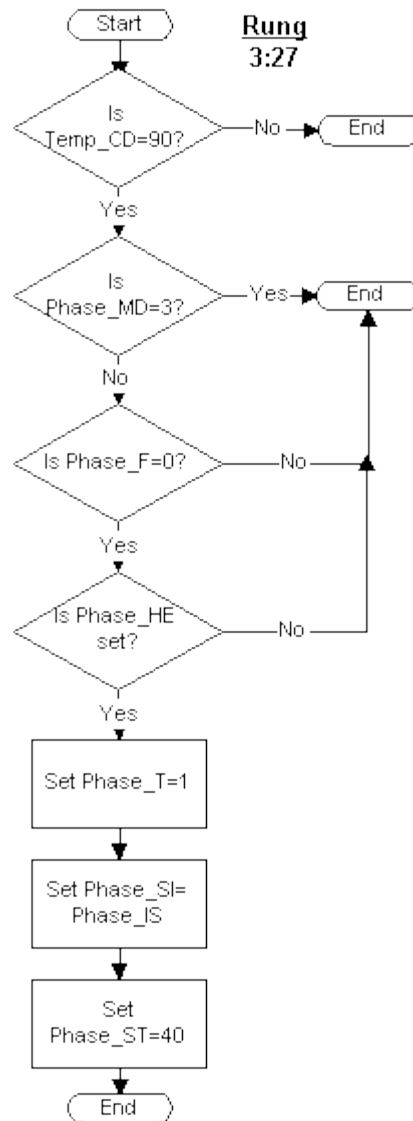
1. Check the temporary command (TEMP\_CD) register. If this register contains a value of 40, the Reset phase command, continue to the next step.
  - a. If TEMP\_CD does not contain a value 40 and this is not the first scan, this step of the PLI terminates.
  - b. If TEMP\_CD does not contain a value of 40, and this is the first scan, check the Idle Initialization (PHASE\_II) flag. If this flag is not set, this step of the PLI terminates. If PHASE\_II is set, proceed to step 3 of this process.
2. Check the following flags:
  - Complete State Active (PHASE\_C) flag
  - Aborted State Active (PHASE\_AD) flag
  - Stopped State Active (PHASE\_SD) flag

If none of these flags are set, this step of the PLI terminates. If any of these flags are set, continue to the next step.
3. Set the Idle State Active (PHASE\_I) flag.
4. Set the Phase Status (PHASE\_ST) register to 100, the Idle value.
5. Clear the Phase Failure (PHASE\_F) register.
6. Set the Step Index (PHASE\_SI) register equal to the register containing the initial value of the Step Index (PHASE\_IS).
7. Clear the Phase Request Data Array (PHASE\_RQ). When the PLI changes from any active state to any other active state, values in PHASE\_RQ are cleared.
8. Clear the Phase Request Data Buffer file (PHASE\_BRQ).

The processing of the Reset command is complete.

## Process Restart Command

The following figure illustrates how the PLI processes the Restart command. This figure is a child to the parent illustration, in the Overview: Processing Commands section.



*Process Restart Command*

The Restart command process performs the following tasks:

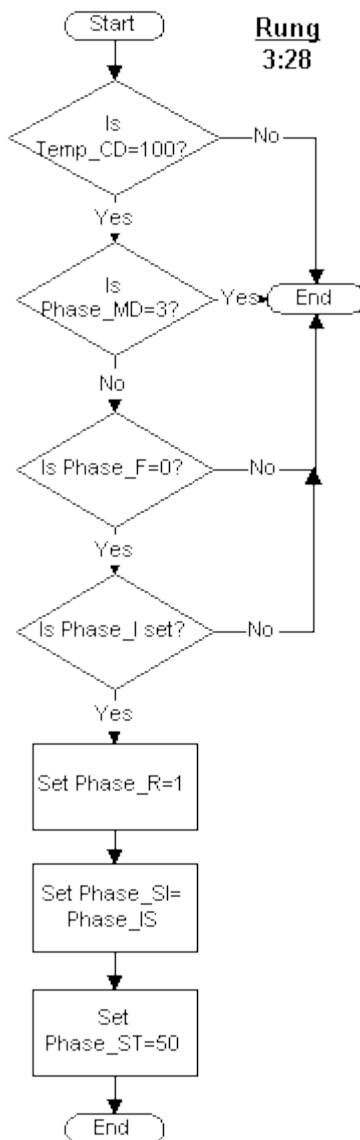
1. Check the temporary command (TEMP\_CD) register. If the register is equal to 90, the Restart phase command, proceed to the next step. If the TEMP\_CD register is not set to 90, this step of the PLI terminates.
2. Check the Batch Execution Phase Mode (PHASE\_MD) register. If this register contains a value of 3, which indicates Manual mode, this step of the PLI terminates. If PHASE\_MD does not contain a value of 3, continue to the next step.
3. Check the Phase Failure (PHASE\_F) register. If this register is equal to zero, proceed to the

- next step. If there is a non-zero value in this register, this step of the PLI terminates. When the PLI detects a non-zero value in this register, the phase changes from Running to Holding.
4. Check the Held State Active (PHASE\_HE) flag. If this flag is not set, this step of the PLI terminates. If PHASE\_HE is set, proceed to the next step.
  5. Set the Restarting State Active (PHASE\_T) flag.
  6. Set the Step Index (PHASE\_SI) register equal to the register containing the initial value of the Step Index (PHASE\_IS).
  7. Set the Phase Status (PHASE\_ST) register to 40, the Restarting value.

The processing of the Restart command is complete.

## Process Start Command

The following figure illustrates how the PLI processes the Start command. This figure is a child to the parent illustration, in the Overview: Processing Commands section.



*Process Start Command*

The Start command process performs the following tasks:

1. Check the temporary command (TEMP\_CD) register. If this register is equal to 100, the Start phase command, proceed to the next step. If TEMP\_CD is not set to 100, this step of the PLI terminates.
2. Check the Batch Execution Phase Mode (PHASE\_MD) register. If PHASE\_MD contains a value of 3, which indicates Manual mode, this step of the PLI terminates. If PHASE\_MD does not contain a value of 3, continue to the next step.
3. Check the Phase Failure (PHASE\_F) register. If this register is equal to zero, proceed to the

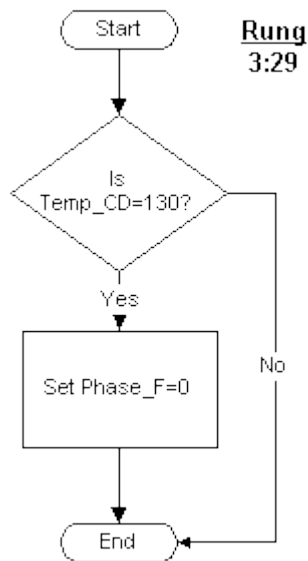
next step. If there is a non-zero value in this register, this step of the PLI terminates. When the PLI detects a non-zero value in this register, the phase changes from Running to Holding.

4. Check the Idle State Active (PHASE\_I) flag. If PHASE\_I is not set, this step of the PLI terminates. If PHASE\_I is set, proceed to the next step.
5. Set the Running State Active (PHASE\_R) flag.
6. Set the Step Index (PHASE\_SI) register equal to the register containing the initial value of the Step Index (PHASE\_IS).
7. Set the Phase Status (PHASE\_ST) register to 50, the Running value.

The processing of the Start command is complete.

## Process Clear Failure Command

The following figure illustrates how the PLI processes the Clear Failure command. This figure is a child to the parent illustration, in the Overview: Processing Commands section.



*Process Clear Failure Command*

The Clear Failure command process performs the following task:

- Check the temporary command (TEMP\_CD) register.
  - a. If this register contains a value of 130, the Clear Failure phase command, clear the Phase Failure (PHASE\_F) register.
  - b. If TEMP\_CD does not contain a value of 130, this step of the PLI terminates.

The processing of the Clear Failure command is complete.

## Handshaking Commands

Certain commands use a protocol known as handshaking to ensure that commands are read, processed, and completed. The handshaking protocol is a method of communication between the Batch Execution Server, the PLI, and the phases in a process. In this protocol:

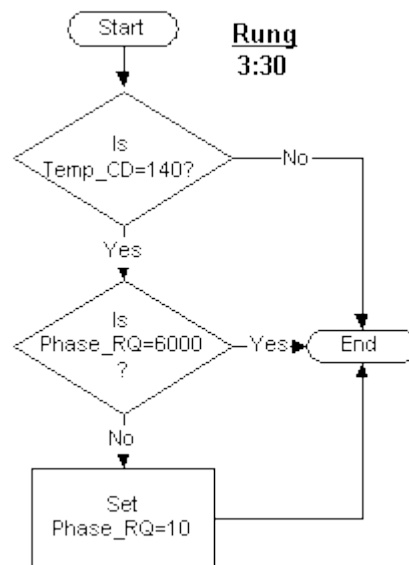
1. A command message is sent from the Server to the controller's Phase Logic via the PLI.
2. The Server writes the command to the low word (16 bits) of the command register, along with a command serial number to the high word (message ID).
3. The controller (PLI) acknowledges the command by zeroing out the low word of the command register, leaving the serial number in the high word of the command register.

The Batch Server will not send further commands if a command acknowledgement is outstanding. All commands must be acknowledged.

The following sections describe the commands that use handshaking to communicate the command's completion back to the Batch Execution Server.

### Process Request Confirmed Command

The following figure illustrates how the PLI processes the Request Confirmed command. This figure is a child to the parent illustration, in the Overview: Processing Commands section.



*Process Request Confirmed Command*

The Request Confirmed command process performs the following task:

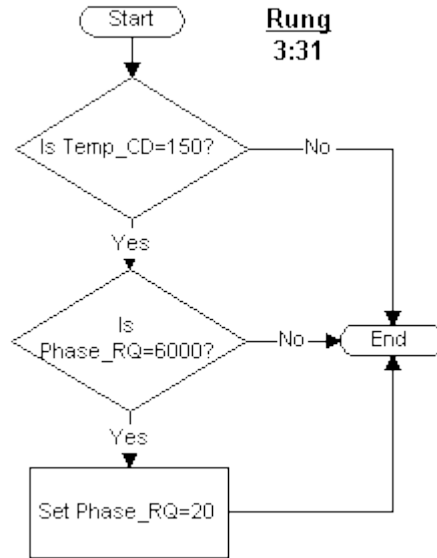
- Check the temporary command (TEMP\_CD) register. If TEMP\_CD does not contain a value of 140, this step of the PLI terminates. If TEMP\_CD contains a value of 140, the Request Confirmed command, check the Phase Request (PHASE\_RQ) register.
  - a. If PHASE\_RQ is equal to 6000, this step of the PLI terminates.
  - b. If PHASE\_RQ has a value other than 6000, set PHASE\_RQ to 10. This functions as

handshaking to the Batch Execution Server, so the Server recognizes the command has been processed.

The processing of the Request Confirmed command is complete.

## Process Cancel Previous Request Command

The following figure illustrates how the PLI processes the Cancel Previous Request command. This figure is a child to the parent illustration, in the Overview: Processing Commands section.



*Process Cancel Previous Request Command*

The Cancel Previous Request command process performs the following task:

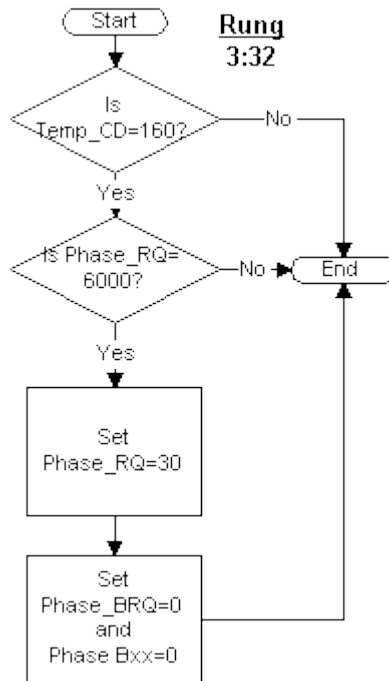
- Check the temporary command (TEMP\_CD) register. If TEMP\_CD does not contain a value of 150, this step of the PLI terminates. If TEMP\_CD contains a value of 150, the Request Successfully Canceled command, check the Phase Request (PHASE\_RQ) register.
  - a. If PHASE\_RQ has a value other than 6000, this step of the PLI terminates.
  - b. If PHASE\_RQ is equal to 6000, set PHASE\_RQ to 20. This functions as handshaking to the Batch Execution Server, so the Server recognizes the command has been processed.

The processing of the Cancel Previous Request command is complete.



## Process Request Unsuccessfully Canceled

The following figure illustrates how the PLI processes the Request Unsuccessfully Canceled command. This figure is a child to the parent illustration, in the Overview: Processing Commands section.



*Process Request Unsuccessfully Canceled Command*

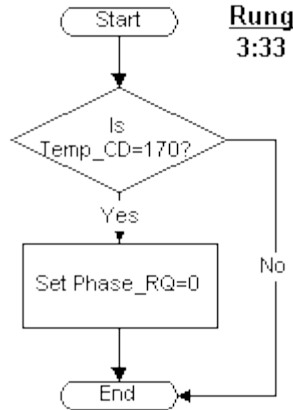
The Request Unsuccessfully Canceled command process performs the following task:

- Check the temporary command (TEMP\_CD) register. If TEMP\_CD does not contain a value of 160, this step of the PLI terminates. If TEMP\_CD contains a value of 160, the Request Unsuccessfully Canceled command, check the Phase Request (PHASE\_RQ) register.
  - a. If PHASE\_RQ is not equal to 6000, this step of the PLI terminates.
  - b. If PHASE\_RQ equals 6000, set PHASE\_RQ to 30. This functions as handshaking to the Batch Execution Server, so the Server recognizes the command has been processed.
  - c. Set the Phase Request Data Buffer File (PHASE\_BRQ) to 0 and Phase\_Bxx to 0.

The processing of the Request Unsuccessfully Canceled command is complete.

## Process Clear Request Register Command

The following figure illustrates how the PLI processes the Clear Request Register command. This figure is a child to the parent illustration, in the Overview: Processing Commands section.



*Process Clear Request Register Command*

The Clear Request Register command process performs the following task:

- Check the temporary command (TEMP\_CD) register. If TEMP\_CD does not contain a value of 170, this step of the PLI terminates. If TEMP\_CD contains a value of 170, the Clear Request Register command, check the Phase Request (PHASE\_RQ) register.

The processing of the Clear Request Register command is complete.

---

## Mode Change Commands

The following sections describe how the AB5 PLI processes commands that change the phase mode. When designing your own PLI, you only need to include these commands if:

- You are using an external system to control your phases.
- You want to change phase modes during your process.

**NOTE:** *The Batch Execution Server will never send commands to change the phase mode.*

The mode change commands are as follows:

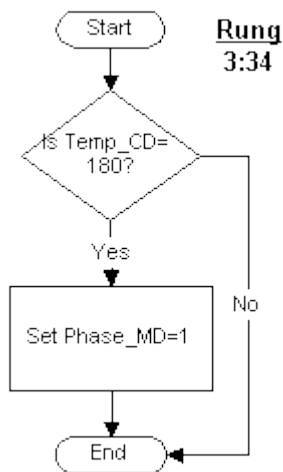
**P-Auto mode** – allows the step's transition to execute, but prevents an operator from sending commands to a batch. Batch Execution is in control.

**O-Auto mode** – allows the step's transition to execute and allows an operator to send commands to a procedure. The operator is in direct control.

**Manual mode** – prevents the step's transition from executing, but allows an operator to send commands to a procedure. The phase cannot execute its logic and ignores all commands.

## Process Phase Mode P-Auto Command

The following figure illustrates how the PLI processes the Request Confirmed command. This figure is a child to the parent illustration, in the Overview: Processing Commands section.



*Process Phase Mode P-Auto Command*

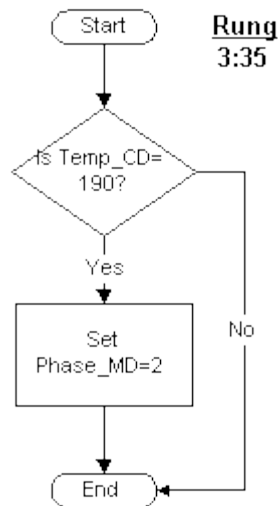
The Phase Mode P-Auto command process performs the following tasks:

1. Check the temporary command (TEMP\_CD) register. If TEMP\_CD contains a value of 180, the Phase Mode P-Auto command, proceed to the next step. If it does not contain a value of 180, this step of the PLI terminates.
2. Set the Phase Mode (PHASE\_MD) register to 1, the ordinal state for P-Auto.

The processing of the Phase Mode P-Auto command is complete.

## Process Phase Mode O-Auto Command

The following figure illustrates how the PLI processes the Phase Mode O-Auto command. This figure is a child to the parent illustration, in the Overview: Processing Commands section.



*Process Phase Mode O-Auto Command*

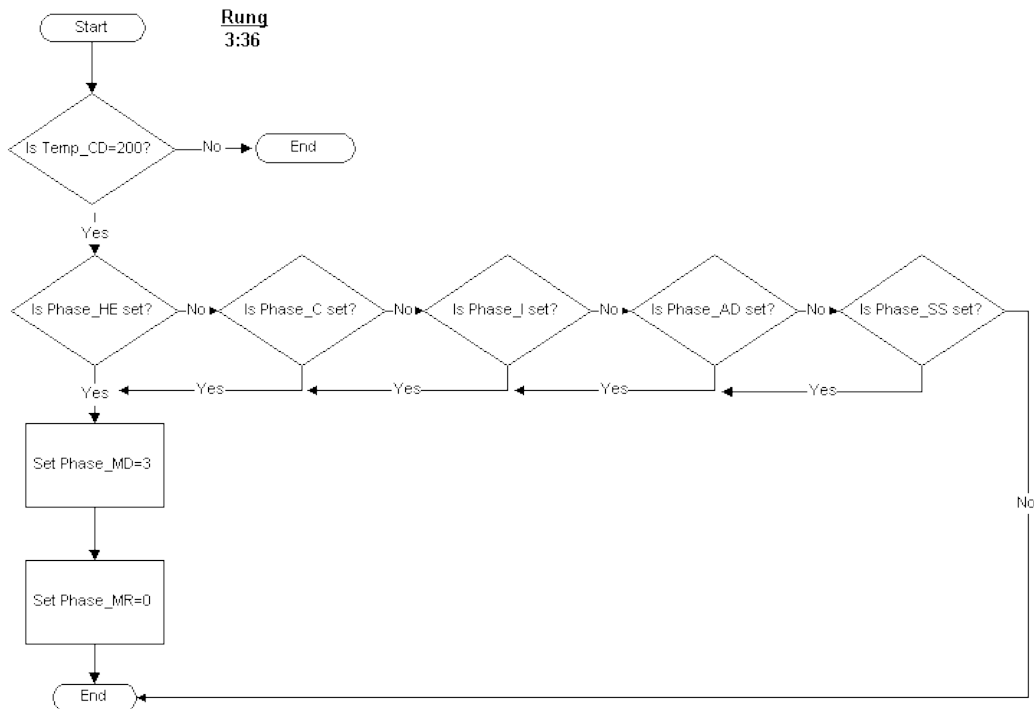
The Phase Mode O-Auto command process performs the following tasks:

1. Check the temporary command (TEMP\_CD) register. If TEMP\_CD contains a value of 190, the Phase Mode O-Auto command, proceed to the next step. If it does not contain a value of 190, this step of the PLI terminates.
2. Set the Phase Mode (PHASE\_MD) register to 2, the ordinal state for O-Auto.

The processing of the Phase Mode O-Auto command is complete.

## Process Phase Mode Manual Command

The following figure illustrates how the PLI processes the Phase Mode Manual command. This figure is a child to the parent illustration, in the Overview: Processing Commands section.



*Process Phase Mode Manual Command*

The Phase Mode Manual command process performs the following tasks:

1. Check the temporary command (TEMP\_CD) register. If TEMP\_CD contains a value of 200, the Phase Mode Manual command, proceed to the next step. If it does not contain a value of 200, this step of the PLI terminates.
2. Check the following flags:
  - Held State Active (PHASE\_HE) flag
  - Complete State Active (PHASE\_C) flag
  - Idle State Active (PHASE\_I) flag
  - Aborted State Active (PHASE\_AD) flag
  - Single Step (PHASE\_SS) flag

If none of these flags are set, this step of the PLI terminates. If any of the flags are set, continue to the next step of this process.

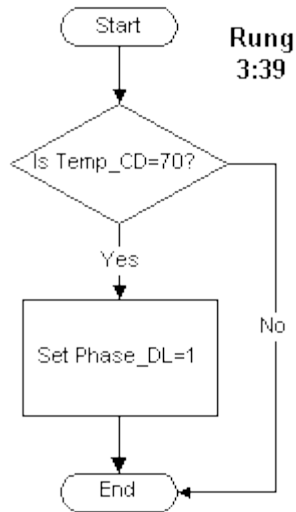
3. Set the Phase Mode (PHASE\_MD) register to 3, which indicates Manual mode.
4. Clear the Mode Request (PHASE\_MR) flag.

The processing of the Phase Mode Manual command is complete.

---

## Download Request Command Process

The following figure illustrates how the PLI processes the Download Request command.



*Process Download Request Command*

The Download Request command process performs the following tasks:

1. Check the temporary command (TEMP\_CD) register. If TEMP\_CD contains a value of 70, the Download Request command, proceed to the next step. If the register does not contain a value of 70, this step of the PLI terminates.
2. Set the Download Request (PHASE\_DRQ) flag.

The processing of the Download Request command is complete.

---

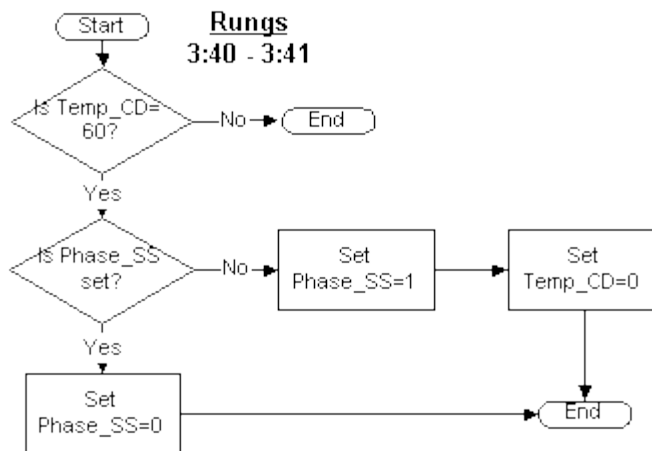
## Phase Debugging Commands

The following sections describe how the PLI processes commands that are used to debug phases. The following commands are described:

- Process Single Step Command
- Process Pause Command
- Process Resume Command

## Process Single Step Command

The following figure illustrates how the PLI processes the Single Step command.



*Process Single Step Command*

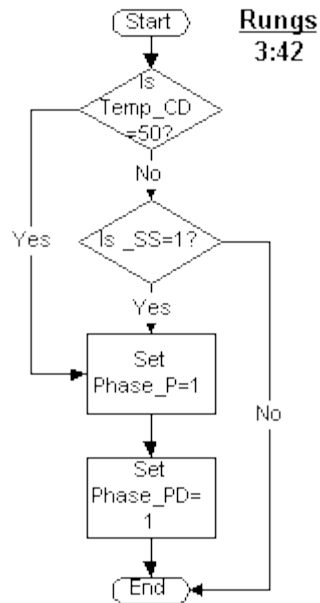
The Single Step command process performs the following tasks:

1. Check the temporary command (TEMP\_CD) register. If TEMP\_CD contains a value of 60, the Single Step command, proceed to the next step. If it does not contain a value of 60, this step of the PLI terminates.
2. Check the Single Step (PHASE\_SS) flag.
  - a. If PHASE\_SS is not set, set the flag and then clear the TEMP\_CD. This step of the PLI terminates.
  - b. If PHASE\_SS is set, clear the flag.

The processing of the Single Step command is complete.

## Process Pause Command

The following figure illustrates how the PLI processes the Pause command.



*Process Pause Command*

The Pause Command process performs the following tasks:

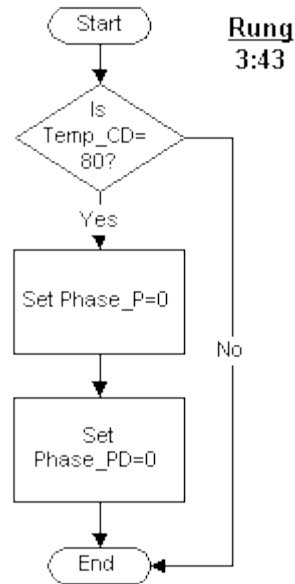
1. Check the temporary command (TEMP\_CD) register. If TEMP\_CD contains a value of 50, proceed to step number 3. If the register does not contain a value of 50, proceed to the next step.
2. Check the Single Step (PHASE\_SS) flag. If the flag is not set, this step of the PLI terminates. If PHASE\_SS is set, proceed to the next step.
3. Set the Pause (PHASE\_P) flag.
4. Set the Paused (PHASE\_PD) flag.

The processing of the Pause command is complete.



## Process Resume Command

The following figure illustrates how the PLI processes the Resume command.



*Process Resume Command*

The Resume command process performs the following tasks:

1. Check the temporary command (TEMP\_CD) register. If TEMP\_CD contains a value of 80, the Resume command, clear the Pause (PHASE\_P) flag. If TEMP\_CD does not contain a value of 80, this step of the PLI terminates.
2. Clear the Paused (PHASE\_PD) flag.

The processing of the Resume command is complete.

---

## Quick References

The sections that follow provide several tables that allow you to quickly get information on the following:

- Batch Execution Command Values
- Phase Status Values
- Batch Execution Requests
- iFIX Database Tags

---

## Batch Execution Command Values

The following table contains a list of the Batch Execution commands and their associated values. These values are stored in the temporary command register (TEMP\_CD) in the PLI.

Batch Execution Commands	
This Command...	Writes This Value...
ABORT	10
HOLD	20
STOP	30
RESET	40
PAUSE	50
SINGLE STEP	60
DOWNLOAD REQUEST	70
RESUME	80
RESTART	90
START	100
CLEAR FAILURE	130
REQUEST CONFIRMED	140
REQUEST SUCCESSFULLY CANCELED	150
REQUEST UNSUCCESSFULLY CANCELED	160
CLEAR REQUEST REGISTER	170
PHASE MODE P-AUTO	180
PHASE MODE O-AUTO	190

<b>Batch Execution Commands</b>	
<b>This Command...</b>	<b>Writes This Value...</b>
PHASE MODE MANUAL	200

---

## Phase Status Values

The following table lists the phase states supported by Batch Execution and the phase status values. These values are stored in the Phase Status (PHASE\_ST) register.

<b>Phase Status Values</b>	
<b>This State...</b>	<b>Has This Status Value...</b>
Aborting	10
Holding	20
Stopping	30
Restarting	40
Running	50
Held	60
Complete	70
Stopped	80
Aborted	90
Idle	100

---

## Batch Execution Requests

The following table provides a quick-reference for the Batch Execution request functions.

Requests Quick-Reference	
Use this request...	To...
PHASE_RQ = 1000	Download all phase parameters.
PHASE_RQ = 1100	Download a range of phase parameters.
PHASE_RQ = 1200	Download a single phase parameter starting at parameter index 1.
PHASE_RQ = 1300	Download a single phase parameter value stored in a specific parameter index.
PHASE_RQ = 2000	Upload all report parameter values.
PHASE_RQ = 2100	Upload a range of report parameter values.
PHASE_RQ = 2200	Upload a single report parameter retrieved from parameter index 1.
PHASE_RQ = 2300	Upload a single report parameter retrieved from a specific parameter index.
PHASE_RQ = 3000	Send a message to an operator.
PHASE_RQ = 4000	Acquire a single resource.
PHASE_RQ = 4100	Acquire multiple resources.
PHASE_RQ = 4200	Release a single resource.

<b>Requests Quick-Reference</b>	
<b>Use this request...</b>	<b>To...</b>
PHASE_RQ = 4300	Release multiple resources.
PHASE_RQ = 4400	Release all currently acquired resources.
PHASE_RQ = 5000	Send a message to a phase.
PHASE_RQ = 5100	Send a message to a phase and wait for a response from all receiving phases.
PHASE_RQ = 5200	Send a message to a phase and wait for a response from one receiver.
PHASE_RQ = 5300	Cancel a specific message that was sent to a phase.
PHASE_RQ = 5400	Cancel all messages.
PHASE_RQ = 5500	Wait for a message from a phase.
PHASE_RQ = 6000	Abort a request.
PHASE_RQ = 7001	Send electronic work instructions to the operator.
PHASE_RQ = 7100	Download the user-defined Batch ID.
PHASE_RQ = 7200	Download the Batch Execution Serial Number.
PHASE_RQ = 7300	Download the phase ID.

<b>Requests Quick-Reference</b>	
<b>Use this request...</b>	<b>To...</b>
PHASE_RQ = 7400	Download the Batch Execution node name.
PHASE_RQ = 7500	Download the fully qualified phase path.

---

## **iFIX Database Tags**

Batch Execution lets you assign iFIX database tags to Batch Execution equipment phase tags and unit tags. The following table lists the recommended iFIX tag types to assign to Equipment Phase tags.

<b>Recommended iFIX Tag Types for Equipment Phase Tags</b>		
<b>Equipment Module Tag</b>	<b>Equivalent PLI Register</b>	<b>Recommended iFIX Tag Type</b>
COMMAND	Command Register (PHASE_VC)	Analog Input (AI) *
FAILURE	Failure Register (PHASE_F)	Analog Input (AI) *
OWNER	Owner Register (PHASE_W)	Digital Input (DI)
PARMTR0n	Parameter Value Register (PHASEP0n)	Analog Input (AI) *
REQUEST	Request Register (PHASE_RQ)	Analog Input (AI) *
REQUEST0n	Request Data Register (PHASEQ0n)	Analog Input (AI) *
REPORT0n	Report Value Register (PHASER0n)	Analog Input (AI) *
PAUSE	Pause Register (PHASE_P)	Digital Input (DI)
PAUSED	Paused Register (PHASE_PD)	Digital Input (DI)

Recommended iFIX Tag Types for Equipment Phase Tags		
Equipment Module Tag	Equivalent PLI Register	Recommended iFIX Tag Type
STATUS	Status Register (PHASE_ST)	Analog Input (AI)*
SINGLE STEP	Single Step Register (PHASE_SS)	Digital Input (DI)
UNIT	Unit Register (PHASE_UN)	Analog Output (AO)
STEP INDEX	Step Index Register (PHASE_SI)	Analog Input (AI)*

\* **NOTE:** For the Analog Input (AI) tags, make sure you select the Enable Output option (on the Advanced tab of the Analog Input dialog box) in the Proficy iFIX Database Manager.

Unit tags represent data that is associated with a particular unit, such as a temperature or tank level indicator. The type of iFIX tag that you assign to a unit tag will vary. For example, if the unit tag represents a temperature sensor, an Analog Input tag may be appropriate.

In the case of the UNIT\_READY and UNIT\_PRIORITY tags, use the iFIX tag types listed in the following table.

iFIX Tag Types for Unit Status Tags	
For these tags...	Use this iFIX Database Tag Type...
UNIT_READY tags	Analog Input (AI), with the Enable Output option
UNIT_PRIORITY tags	Analog Input (AI), with the Enable Output option

## Sample PLI

Sample PLI ladder logic for the following programmable controllers are included in the Samples folder with the Proficy Batch Execution product:

- The Allen-Bradley™ PLC5 Series (PLI\_REV2\_06.RSP). This PLI was written in RSLogix™ 5.
- The Allen-Bradley™ SLC/500 (PLI\_REV2\_06.RSS). This PLI was written in RSLogix 500.
- The Allen-Bradley™ ControlLogix Processor (PLI\_REV4\_00.ACD). This PLI was written in RSLogix 5000. This sample uses Data Structures or Direct Addressing. It does not use PLC-5 addressing (recommended), and instead uses RSLinx as an OPC Server providing information directly to Batch Execution.

- The Allen-Bradley™ ControlLogix™ Processor (PLI\_REV2\_06.ACD). This PLI was written in RSLogix 5000. The sample provides a more robust communications method using Allen-Bradley driver and RSLinx™ through Proficy iFIX as opposed to OPC directly to RSLinx.
- The Siemens S7-300 and S7-400 PLCs (S7\_pli\_6.zip). This PLI was written with Siemens Step 7 Series software.

If you installed Proficy Batch Execution to the default folder, you can find the PLI samples in the C:\Program Files\Proficy\Proficy Batch Execution\samples\PLI folder.

---

## Sample AB5 PLI Ladder Logic Notes

The sections below provide more sample information for the PLI ladder logic for Allen-Bradley™ PLC5 Series programmable controllers example (PLI\_REV2\_06.RSP) included in the Batch Execution/Samples/PLI folder.

### Data Table File Overview

The Address Descriptions table that follows contains descriptions for the PLC5 data file structures for the Batch Execution PLI. The number of phases within the PLC determines the length of files N101 - N117, and N120. The length is the number of phases plus one (1). Element zero (0) is not used for phase configuration.

### Configuring the Request Qualifier Array

For most applications, the phase logic will be required to request information from the Batch Execution Server. This is accomplished through the use of the PHASE\_RQ Request Register stored in File N107. Some requests cannot be properly defined with a single Request word. Requests can be further defined through the use of Request Qualifiers, which complement the original Request Register.

The PLI assumes that all phases have the same number of Request Qualifiers. The number of phases multiplied by the number of qualifiers plus one (1) determines the size of the Phase Qualifier file (F118) and the Buffered Phase Qualifiers file (F119). Element zero (0) in these files is not used.

The Sample PLI assumes that each phase has three qualifiers. If your application requires less, no modifications are necessary. If you require more than the default, you must make the following modifications to the PLI.

Rung Number	Instruction	Field	Change
LAD10: 26, 29	COP	LENGTH	Change the length from 3 to the required number of Request Qualifiers
LAD10: 26, 31, 43	FLL	LENGTH	Change the length from 3 to the required number of Request Qualifiers

**NOTE:** There are two FLL instructions and you must modify both.



You must also modify N100:1 from three to the required number of qualifiers.

## Configuring Phase and Report Parameters

Neither phase nor report parameters are used in the PLI. Therefore, you can manipulate them at will. However, the correct address for these tags must be available to the Batch Execution Server.

## Data Table File Descriptions

The following table contains descriptions for the PLC5 data file structures for the Batch Execution PLI.

Address Descriptions		
File Number	Tag Name	Description
N101	PHASE_EC	External Command register file. Commands are sent from an External source, such as an HMI system, into these registers to make phase state and attribute changes. The command values used here are identical to the Batch Execution command values.
N102	PHASE_F	Phase Failure register file. The user's phase logic can detect abnormal conditions within the phase and store a failure code in this register. When the PLI detects a non-zero value in this register, the phase changes states from RUNNING to HOLDING. The failure codes can be enumerated within the Batch Execution area model. Batch Execution can detect the value in this register and log the equivalent enumerated text into the Event Journal as well as display it in the Batch Execution Client's SFC and Alarm Summary screens. More critical alarms should be given higher Failure values. (For example, 100 is a more critical failure than 80.)

Address Descriptions		
File Number	Tag Name	Description
N103	PHASE_IPE	<p>Internal Phase Error:</p> <ul style="list-style-type: none"> <li>• Error 0 indicates there is no error.</li> <li>• Error 1 indicates that Batch Execution issued a command but the phase cannot respond because it is externally controlled.</li> <li>• Error 2 indicates that an External Command was given but the phase cannot respond because Batch Execution controls it.</li> <li>• Errors 3-9 denote that a phase indicated a completion but was not in the state for which the completion was given.</li> <li>• x1 indicates that a command was given, but the mode of the phase was invalid for the given command.</li> <li>• x2 indicates that a command was given but the state of the phase was invalid for the given command.</li> </ul>
N104	PHASE_ISI	Initial Step Index. Set the value in this register to the desired initial Step Index value.
N105	PHASE_MD	<p>Batch Execution Phase Mode register file. A phase has three modes: P-Auto, O-Auto, and Manual. When the phase is in P-Auto, a recipe in the Batch Execution Server is controlling the phase. When the phase is in O-Auto, the operator is controlling the phase. When the phase is in Manual mode, the phase cannot execute. The ordinal states of the Phase Mode register are defined as follows:</p> <p>0 = Undefined, 1 = P-Auto, 2 = O-Auto, 3 = Manual.</p>
N106	PHASE_PF	Previous Failure. This register contains the last failure reported by the phase logic.
N107	PHASE_RQ	Phase Request. The phase logic may be required to request that the Batch Execution Server perform tasks such as downloading phase parameters or uploading report parameter values. For a listing of the available Requests, refer to the Quick References section.

<b>Address Descriptions</b>		
<b>File Number</b>	<b>Tag Name</b>	<b>Description</b>
N108	PHASE_BF	Buffered Failure. The current Failure is compared with the Buffered Failure to determine if a more critical failure has occurred during the last scan. Once this has been checked, the Buffered Failure is updated to the current Failure.
N109	PHASE_SI	Phase Step Index register file. When the PLI changes the state to any active state, the PLI initializes the Step Index register to the value in N104. The value is user-configurable and is usually set to 0 or 1. The user's phase logic can then use the Step Index register to step through different steps or states by changing the value in the Step Index register.
N110	PHASE_BRQ	Phase Buffered Request file. When the PLI changes the active state of RUNNING to HOLDING, the Request Register is copied into this file. When the PLI changes states from RESTARTING to RUNNING, the Buffered Request Register value for the phase are copied back to the Request Register if the Restore Request flag (N116:x.11) is on.
N111	PHASE_BSI	Buffered Step Index. This register holds the last step executed by the phase logic.
N112	PHASE_BST	Buffered State. This register holds the last state of the phase prior to the current state.
N113	PHASE_ST	Phase Status register file. The PLI processes commands and changes phase states accordingly. The value of the corresponding state is stored in these registers by the PLI.
N114	PHASE_UN	Unit ID register file. The Unit ID of the operating unit is stored in this register for this phase. If a particular phase is shared between two or more different units, programmed phase logic decisions can be made to change the operating characteristics of the phase.

Address Descriptions		
File Number	Tag Name	Description
N115	PHASE_VC	Batch Execution Command register file. Commands are sent from the Batch Execution Server into these registers to make phase state and attribute changes. The PLI reads this register and changes phase states and attributes accordingly. The command uses the lower byte of each command. The Batch Execution Server uses the upper byte of each command to detect when the PLI has processed a command.
N116	PHASE_CFLAG	Command Flag. Each bit represents a command/status for the phase.
N116 Bit 0	PHASE_AMD	Phase Alter Mode Flag.
N116 Bit 1	PHASE_CRQ	Clear Request flag. This flag is a user-configurable setting. When the phase transitions from RUNNING to HOLDING, ABORTING, or STOPPING, the current request is cleared if this flag is set. If the phase programmer wants to clear the request in the Request Data Array, the phase programmer must set this flag in either the Data Table Monitor or a Rung Output instruction.
N116 Bit 2	PHASE_DRQ	Download Request flag. The Download Request flag is set by a Download command (70). The Download Request command is sent to the phase when there is Transfer of Control within the recipe. The phase programmer can use this flag to determine when the phase has transitioned between two steps within the recipe. The programmer may choose to download new parameters when this flag is set. The phase programmer must reset this flag immediately upon making the request in order to detect another transfer of control.
N116 Bit 3	PHASE_ER	External Request flag. This flag is usually set externally, such as from an HMI system. When this flag is set, the PLI changes ownership of the phase to external.
N116 Bit 4	PHASE_FI	Fail Increase flag. The PLI sets this flag for one scan when the value in the PHASE_BF register is greater than the value in the PHASE_PF register. The phase logic may read this flag, but writing to this flag is not permitted.

Address Descriptions		
File Number	Tag Name	Description
N116 Bit 5	PHASE_NCHOLD	Phase Non Command Hold Condition. Failure Increases and Watchdog timeouts also cause phases to go to Hold. This bit is used to distinguish these Hold conditions from a Batch Execution Hold command.
N116 Bit 6	PHASE_II	Idle Initialization flag. This flag is a user-configurable setting. If this flag is set, the phase will transition to the IDLE state on First Scan. The first scan bit is set when the processor mode changes from Program Mode to Run Mode or when the processor powers up in Run Mode.
N116 Bit 7	PHASE_W	Owner flag. This flag is toggled via the Batch Execution and External Request flags. If the Batch Execution Request flag is set, then the PLI clears this flag. If the External Request flag is set, then the PLI sets this flag. If this flag is clear, the phase owner is Batch Execution.
N116 Bit 8	PHASE_PD	Paused flag. This flag is set by sending a Pause command to the phase or when the phase is in Single Step mode. This flag is reset by sending a Resume command to the phase.
N116 Bit 9	PHASE_P	Pause flag. This flag is set by sending a Pause command to the phase or when the phase is in Single Step mode. This flag is reset by sending a Resume command to the phase. The user's phase logic can examine this flag to pause at any transition within the phase logic. If the phase is in single step mode, a Resume command resets this flag for only one scan.
N116 Bit 10	PHASE_RHG	Phase Re-Execute Holding Flag (for failure increases). This flag is set by the user to determine if Holding logic should be re-executed when a new, more critical failure occurs.
N116 Bit 11	PHASE_RRQ	Restore Request flag. This flag is a user-configurable setting. When the phase transitions from RUNNING to HOLDING, then any current request is buffered. If the phase programmer wants to restore the buffered request to the Request Data Array when transitioning from RESTARTING to RUNNING, the phase programmer must set this flag in either the Data Table Monitor or a Rung Output instruction.

<b>Address Descriptions</b>		
<b>File Number</b>	<b>Tag Name</b>	<b>Description</b>
N116 Bit 12	PHASE_SS	Single Step flag. The Batch Execution Server or an external source can set this flag by sending a single step command to the Command register. If this flag is already set, another single step command resets it "Off." If this flag is set, the phase pauses at every pre-programmed pause transition.
N116 Bit 13	PHASE_VR	Batch Execution Request flag. This flag is usually set externally, such as from an HMI system. When this flag is set, the PLI changes ownership of the phase to Batch Execution.
N116 Bit 14	PHASE_WHG	Hold on Watchdog Time-out flag. This flag is a user-configurable setting. If this flag is set, the phase is put into hold on Watchdog failure.
N116 Bit 15	–	Not Used
N117	PHASE_SFLAG	Phase Status Flag. Each bit represents a state for the phase.
N117 Bit 0	PHASE_AD	Aborted State Active flag. The PLI sets this flag when the phase state is ABORTED. This flag is not read by any phase logic.
N117 Bit 1	PHASE_AG	Aborting State Active flag. The PLI sets this flag when the phase state is ABORTING. The phase logic must read this flag to determine if the ABORTING state is active and to process the ABORTING logic.
N117 Bit 2	PHASE_AC	Aborting Complete flag. The user's phase logic must set this flag to indicate to the PLI that the ABORTING logic has run to completion. The PLI monitors this flag and transitions to the ABORTED state when this flag is on.
N117 Bit 3	PHASE_C	Complete State Active flag. The PLI sets this flag when the phase state is COMPLETE. This flag is not read by any phase logic.

<b>Address Descriptions</b>		
<b>File Number</b>	<b>Tag Name</b>	<b>Description</b>
N117 Bit 4	PHASE_HD	Holding Indicator flag. This is a general use flag. This flag may be read by the phase logic to determine if the phase is in the HOLDING, HELD or RESTARTING states.
N117 Bit 5	PHASE_HG	Holding State Active flag. The PLI sets this flag when the phase state is HOLDING. The phase logic must read this flag to determine if the HOLDING state is active and to process the HOLDING logic.
N117 Bit 6	PHASE_HC	Holding Complete flag. The user's phase logic must set this flag to indicate to the PLI that the HOLDING logic has run to completion. The PLI monitors this flag and transitions to the HELD state when this flag is on.
N117 Bit 7	PHASE_I	Idle State Active flag. The PLI sets this flag when the phase state is IDLE. This flag is not read by any phase logic.
N117 Bit 8	PHASE_RG	Running State Active flag. The PLI sets this flag when the phase state is RUNNING. The phase logic must read this flag to determine if the RUNNING state is active and to process the RUNNING logic.
N117 Bit 9	PHASE_RC	Running Complete flag. The user's phase logic must set this flag to indicate to the PLI that the RUNNING logic has run to completion. The PLI monitors this flag and transitions to the COMPLETE state when this flag is on.
N117 Bit 10	PHASE_SD	Stopped State Active flag. The PLI sets this flag when the phase state is STOPPED. This flag is not read by any phase logic.
N117 Bit 11	PHASE_SG	Stopping State Active flag. The PLI sets this flag when the phase state is STOPPING. The phase logic must read this flag to determine if the STOPPING state is active and to process the STOPPING logic.
N117 Bit 12	PHASE_SC	Stopping Complete flag. The user's phase logic must set this flag to indicate to the PLI that the STOPPING logic has run to completion. The PLI monitors this flag and transitions to the STOPPED state when this flag is on.

Address Descriptions		
File Number	Tag Name	Description
N117 Bit 13	PHASE_TG	Restarting State Active flag. The PLI sets this flag when the phase state is RESTARTING. The phase logic must read this flag to determine if the RESTARTING state is active and to process the RESTARTING logic.
N117 Bit 14	PHASE_TC	Restarting Complete flag. The user's phase logic must set this flag to indicate to the PLI that the RESTARTING logic has run to completion. The PLI monitors this flag and transitions to the RUNNING state when this flag is on.
N117 Bit 15	–	Not Used
F118	PHASE_Qxx	Phase Request Qualifiers. These registers are used to further define a Request to the Batch Execution Server.
F119	PHASE_BQ	Phase Buffered Request Qualifiers. These registers store the Phase Request Qualifiers when the phase State is changed from Running to Holding.
N120	PHASE_M	Mode Expressed as an Integer.
F121	PHASE_Pxx	Floating Point Phase Parameters.
N122	PHASE_Pxx	Integer Phase Parameters.
F123	PHASE_Rxx	Floating Point Report Parameters.
N124	PHASE_Rxx	Integer Report Parameters.
N125	PHASE_BVC	Buffered Batch Execution Command. This register holds the last command processed by the PLI for the phase. This value includes both the command and Serial Number for the command.
N126	PHASE_BPC	Buffered Phase Command. This register holds the last command issued for the phase. This value added to the value in the VC register should always equal the PHASE_BVC.



## Other Tags

The following table contains descriptions for other data file structures that are used by the Batch Execution PLI.

Other Tag Descriptions		
Address	Tag Name	Description
N100:0/0	WDOG	Watchdog flag. This flag is set by the Batch Execution Server on a periodic time interval. The time interval in which this flag is configurable is set in the VBEXEC.INI file. The PLI checks the value of this flag every scan. If the value is set or on, a timer, T1304:0, is reset to 0 (zero). The Watchdog flag is then reset.
N100:1	PLI_NRQ	Number of Request Qualifiers per phase.
N100:2	PLI_PC	Phase Command. This command is the local register without the Batch Execution serial number.
N100:3	PLI_PID	Phase ID number for the phase that PLI is currently evaluating.
N100:4	PLI_RQI	Request Qualifier Index. Index that PLI uses in the Qualifier array to determine the starting point for the request qualifiers for the current phase.
N100:5	PLI_EPN	Ending Phase Number (when the PLI_PID reaches this value, it resets to 1 and starts again)
T130:0	WDOG_TMR	Watchdog Timer

---

## Troubleshooting

When trying to troubleshoot your PLI, you must first determine if the PLI is the actual point of failure. The failure of a batch to properly execute can originate in many locations including:

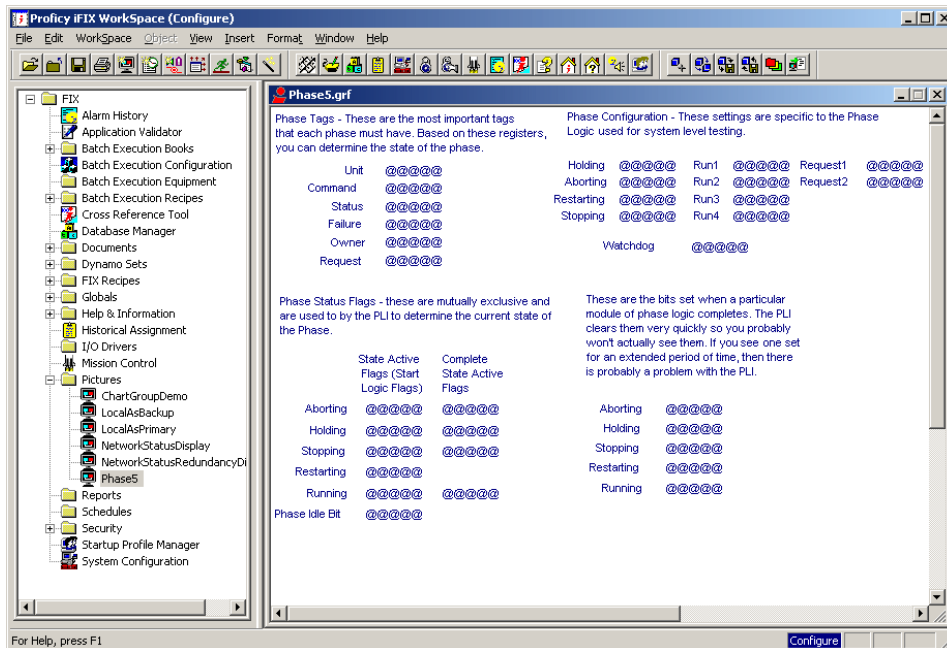
- Batch Execution Server
- OPC Server
- iFIX
- I/O Driver
- Phase Logic
- Computer Network

- PLI

As with many problems, diagnosing the component causing the problem is best determined through a process of elimination. For example, suppose you can't start any phases.

The following steps can help you to quickly determine the source of the problem:

1. Create a picture in the Proficy iFIX WorkSpace, as shown in the following figure.
2. Attempt to send the Start command to the PLI from this screen. If this is unsuccessful, you can eliminate the Batch Execution Server, the OPC Server, and possibly the network as the source of your problem.
3. Write the Start command directly from the process controller programming software. If this step is unsuccessful, you can eliminate iFIX, the I/O Driver, and most network connections as the source of your problem.
4. Set the Start Running Logic flag directly from the process controller. This can help to eliminate the PLI as the source of the problem.
5. If the phase still won't start, you should re-examine the phase logic.



Sample iFIX Troubleshooting Screen

There are other methods to quickly determine the cause of a failure.

- If you are running a 6.x driver iFIX, check Mission Control, which lists all driver errors.
- If you are running a 7.x driver iFIX, click the Statistics button in the driver's Run-time toolbar to display the statistics for the currently selected item in the Tree Browser.

**NOTE:** You also can use Mission Control with 7.x drivers, but you get more detailed information from the driver's Statistics mode.

- Check the VBEXEC.LOG. Communication issues, phase failures, any exception errors, or crashes are recorded in this file.

**NOTE:** *In the vbexec.log file, you may see the error: "Phase Parameter Download - Failed to Verify the Correct Value was Written Warning in vbexec.log file". This message occurs when the Batch server attempts to verify that the correct value was downloaded before the I/O driver has written the value to the PLC. This is normally a timing issue and does not indicate a problem with the system.*



---

# Index

## 1

1000 request .....	78
1100 request .....	78
1200 request .....	78
1300 request .....	78

## 2

2000 request .....	78
2100 request .....	78
2200 request .....	78
2300 request .....	78

## 3

3000 request .....	78
--------------------	----

## 4

4000 request .....	78
4100 request .....	78
4200 request .....	78
4300 request .....	78
4400 request .....	78

## 5

5000 request .....	78
5100 request .....	78
5200 request .....	78
5300 request .....	78
5400 request .....	78
5500 request .....	78

## 6

6000 request.....	78
-------------------	----

## 7

7001 request.....	78
7100 request.....	78
7200 request.....	78
7300 request.....	78
7400 request.....	78
7500 request.....	78

## A

abort	
command .....	22
process command .....	51
abort.....	51
aborted state.....	20
aborted state active flag .....	82
aborting	
process complete.....	39
state.....	19
aborting.....	19
aborting complete flag .....	82
aborting state active flag .....	82
active states	
aborting.....	19
holding.....	19

restarting.....	19	communication loss .....	21
running.....	19	device failure .....	21
stopping .....	19	issuing commands.....	21
active states.....	19	normal completion of a sequence .....	21
allocating memory variables		changing states.....	21
DCS .....	3	checking	
PLC.....	3	for phase failure .....	37
allocating memory variables.....	3	the watchdog.....	32
<b>B</b>		checking.....	32
Batch Execution command		checking phase status flags.....	38
described.....	8	clear failure	
mechanism.....	23	command .....	76
Batch Execution command.....	23	process command .....	64
Batch Execution commands		clear failure.....	64
processing.....	23	clear request	
understanding .....	22	process command .....	68
writing .....	23	clear request.....	68
Batch Execution commands .....	23	clear request flag.....	82
Batch Execution request flag.....	82	clear request register	
buffered Batch Execution command .....	82	command .....	76
buffered failure .....	82	clear request register .....	76
buffered phase command.....	82	clear request register command .....	76
buffered phase qualifiers .....	82	command flag .....	82
buffered state .....	82	command register .....	82
buffered step index .....	82	commands	
<b>C</b>		abort.....	22
cancel previous request command.....	66	Batch Execution.....	8
changing states		clear failure .....	64

clear request.....	76	complete state .....	20
copying .....	36	complete state active flag.....	82
download .....	22	control	
external .....	12	external .....	27
hold.....	22	control.....	27
mode change.....	46	copying commands .....	36
operator issued.....	50	<b>D</b>	
pause.....	22	developing a PLI.....	4
phase debugging .....	72	device failure .....	21
processing.....	45	download request	
reset .....	22	command .....	22
restart.....	61	described.....	15
resume .....	75	process command .....	72
single step.....	22	download request.....	72
stop .....	22	download request flag.....	82
using handshaking .....	65	<b>E</b>	
commands.....	65	equipment phase tags.....	4
communication		example	
Batch Execution.....	7	AB5 PLI.....	82
Batch Execution Server or HMI to the phase logic.....	14	normal phase execution scenario .....	24
Batch Execution Server to PLI .....	8	example.....	24
HMI to the PLI .....	12	external	
loss of .....	21	commands.....	12
overview .....	7	requests .....	12
PLI to Batch Execution Server .....	9	external .....	12
communication .....	9	external command register.....	82
communication loss.....	21	external control .....	27
		external requests .....	82

## F

fail increase flag .....	82	holding indicator.....	47
failure		holding state active .....	41
checking phase.....	37	idle initialization.....	82
number.....	9	idle state active .....	30
register .....	37	mode request.....	30
failure .....	37	owner .....	36
failure register.....	82	pause .....	47
flags		paused .....	48
aborted state active .....	39	phase failure.....	37
aborting complete .....	38	phase hold index .....	51
aborting state active.....	39	phase mode .....	46
Batch Execution command.....	34	phase request data array.....	42
Batch Execution phase mode.....	61	phase request data buffer .....	42
Batch Execution request .....	34	phase state buffer .....	51
clear .....	51	phase status .....	39
complete state active.....	44	phase step buffer .....	42
continue .....	47	phase step index.....	42
download request.....	82	re-execute hold .....	30
external command .....	34	request data .....	46
external commands .....	36	restarting complete .....	38
external request.....	34	restarting state active .....	42
fail increase.....	37	restore request.....	42
held state active .....	41	running complete .....	38
hold active phases on first scan .....	30	running state active .....	42
hold index .....	56	setting .....	47
hold on watchdog time-out .....	30	single step .....	48
holding complete .....	38	state store .....	56
		step index.....	51



stopped state active.....	40	holding.....	19
stopping complete.....	38	holding complete flag.....	82
stopping state active.....	40	holding indicator flag.....	82
stored failure.....	37	holding state active flag.....	82
watchdog.....	82	<b>I</b>	
flags.....	82	idle	
Floating Point Phase Parameters.....	82	processing phase state.....	48
Floating Point Report Parameters.....	82	state.....	20
flowchart		idle.....	20
conventions.....	28	idle initialization flag.....	82
examples.....	28	idle phase state	
flowchart.....	28	processing.....	48
<b>H</b>		idle phase state.....	48
handshaking		idle state active flag.....	82
described.....	65	iFIX	
example.....	65	tags.....	80
handshaking.....	65	iFIX.....	80
held state.....	20	inactive states	
high byte.....	23	aborted.....	20
hold		complete.....	20
command.....	76	held.....	20
process command.....	53	idle.....	20
hold.....	22	stopped.....	20
hold on watchdog time-out flag.....	82	inactive states.....	20
holding		initial step index.....	82
complete.....	41	Integer Phase Parameters.....	82
process complete.....	41	Integer Report Parameters.....	82
state.....	77	internal phase error.....	82

<b>L</b>		
low byte .....	23	
<b>M</b>		
main calling program		
first scan initialization .....	29	
programming .....	29	
main calling program.....	29	
manual mode .....	71	
memory variables		
allocating on PLC and DCS.....	3	
described.....	1	
naming convention .....	7	
overview .....	7	
memory variables .....	7	
mode		
change.....	68	
process manual command.....	71	
process O-Auto command .....	70	
process P-Auto command.....	69	
mode .....	69	
mode expressed as an integer .....	82	
<b>O</b>		
operator issued commands .....	50	
owner .....	9	
owner flag.....	82	
<b>P</b>		
pause		
command .....	76	
process command .....	74	
pause .....	9	
pause flag.....	82	
paused .....	9	
paused flag .....	82	
P-Auto mode.....	69	
phase		
debugging .....	72	
failure, checking for.....	37	
mode .....	46	
processing .....	34	
phase .....	34	
phase alter mode flag.....	82	
phase mode .....	46	
phase mode manual		
command .....	76	
process command .....	71	
phase mode manual .....	71	
phase mode O- Auto		
process command .....	70	
phase mode O- Auto .....	70	
phase mode O-Auto		
command .....	76	
phase mode O-Auto .....	76	
phase mode P-Auto		
command .....	76	
process command .....	69	
phase mode P-Auto.....	69	

phase mode register file.....	82	PHASE_A.....	39
phase non command hold condition .....	82	PHASE_AC	
phase parameter values.....	14	using .....	38
phase qualifiers.....	82	PHASE_AC .....	82
phase re-execute holding flag .....	82	PHASE_AD	
phase report values .....	15	using .....	39
phase request data array.....	82	PHASE_AD.....	82
phase request data buffer .....	82	PHASE_AG.....	82
phase states		PHASE_AMD .....	82
aborted.....	20	PHASE_BF.....	82
aborting.....	77	PHASE_Bnn.....	82
active .....	19	PHASE_BPC.....	82
changing .....	21	PHASE_BQ.....	82
complete .....	20	PHASE_BRQ	
held.....	20	using .....	42
holding.....	19	PHASE_BRQ .....	82
idle.....	20	PHASE_BSI .....	82
inactive .....	20	PHASE_BST .....	82
restarting.....	19	PHASE_BVC .....	82
running.....	19	PHASE_C	
stopped.....	20	using .....	44
stopping .....	19	PHASE_C.....	82
understanding .....	19	PHASE_CFLAG .....	82
phase states.....	19	PHASE_CR	
phase status flag.....	82	using .....	51
phase status register.....	82	PHASE_CR .....	51
PHASE_A		PHASE_CRQ .....	82
using .....	39	PHASE_CT	

using .....	47	using .....	41
PHASE_CT .....	47	PHASE_HE .....	82
PHASE_DRQ .....	82	PHASE_HG .....	82
PHASE_EC		PHASE_HI	
using .....	34	using .....	51
PHASE_EC .....	12	PHASE_HI .....	51
PHASE_ER		PHASE_I	
using .....	34	using .....	34
PHASE_ER .....	12	PHASE_I .....	82
PHASE_F		PHASE_II	
using .....	37	using .....	30
PHASE_F .....	9	PHASE_II .....	82
PHASE_FI		PHASE_IPE .....	82
using .....	37	PHASE_IS	
PHASE_FI .....	82	using .....	30
PHASE_H		PHASE_IS .....	82
using .....	41	PHASE_M .....	82
PHASE_H .....	41	PHASE_MD	
PHASE_HA		using .....	46
using .....	30	PHASE_MD .....	82
PHASE_HA .....	30	PHASE_MR	
PHASE_HC		using .....	30
using .....	38	PHASE_MR .....	30
PHASE_HC .....	82	PHASE_NCHOLD .....	82
PHASE_HD		PHASE_P	
using .....	47	using .....	47
PHASE_HD .....	47	PHASE_P .....	9
PHASE_HE		PHASE_PD	

using .....	48	PHASE_S .....	40
PHASE_PD .....	9	PHASE_SB	
PHASE_PF .....	82	using .....	42
PHASE_Pnn .....	14	PHASE_SB .....	42
PHASE_Pxx .....	82	PHASE_SC	
PHASE_Qnn .....	14	using .....	38
PHASE_Qxx .....	82	PHASE_SC .....	82
PHASE_R		PHASE_SD	
using .....	42	using .....	40
PHASE_R .....	42	PHASE_SD .....	82
PHASE_RC		PHASE_SF	
using .....	38	using .....	37
PHASE_RC .....	82	PHASE_SF .....	37
PHASE_RE		PHASE_SFLAG .....	82
using .....	30	PHASE_SG .....	82
PHASE_RE .....	30	PHASE_SI	
PHASE_RG .....	82	using .....	42
PHASE_RHQ .....	82	PHASE_SI .....	9
PHASE_Rmn .....	15	PHASE_SS	
PHASE_RQ		using .....	48
using .....	42	PHASE_SS .....	9
PHASE_RQ .....	9	PHASE_ST	
PHASE_RR		using .....	39
using .....	42	PHASE_ST .....	9
PHASE_RR .....	82	PHASE_T	
PHASE_Rxx .....	82	using .....	42
PHASE_S		PHASE_T .....	42
using .....	40	PHASE_TC	

using .....	38	PLI_NRQ.....	82
PHASE_TC .....	82	PLI_PC .....	82
PHASE_TG .....	82	PLI_PID.....	82
PHASE_UN.....	8	PLI_RQI .....	82
PHASE_VC		previous failure.....	82
using .....	34	process	
PHASE_VC.....	8	abort command .....	51
PHASE_VR		aborting complete .....	39
using .....	34	commands.....	49
PHASE_VR.....	82	download request command .....	72
PHASE_W		hold command .....	53
using .....	36	pause command .....	74
PHASE_W.....	9	phase mode manual command.....	71
PHASE_WC		phase mode O-Auto command .....	70
using .....	30	phase mode P-Auto command .....	69
PHASE_WC.....	30	request unsuccessfully canceled command..	67
PHASE_WHG.....	82	reset command.....	59
PLI		resume command.....	75
configurations .....	3	single step command .....	73
multiple-instance .....	3	start command.....	63
necessary elements .....	4	stop command.....	56
overview .....	1	process .....	56
single-instance .....	3	process cancel previous request command	
task overview.....	5	tasks .....	66
tasks.....	31	process cancel previous request command .....	66
understanding .....	1	process holding complete .....	41
PLI.....	1	process phase .....	34
PLI_EPN .....	82	process request confirmed command	

tasks .....	65	command .....	76
process request confirmed command.....	65	process command .....	61
process restarting complete .....	42	restart .....	61
process running complete .....	44	restart command	
process stopping complete.....	40	process .....	61
processing		restart command .....	61
idle phase state.....	48	restarting	
processing .....	48	state.....	77
processing commands.....	45	restarting .....	19
processing idle phase state.....	48	restarting complete flag .....	82
<b>R</b>		restarting state active flag .....	82
request		restore request flag.....	82
data .....	14	resume	
external .....	12	command .....	76
request .....	9	process command .....	75
request confirmed		resume .....	75
command .....	76	running	
process command .....	65	complete .....	44
request confirmed .....	65	process command .....	44
request successfully canceled		state.....	77
command .....	76	running.....	19
request successfully canceled .....	76	running complete flag .....	82
request unsuccessfully canceled command .....	67	running state active flag.....	82
reset		<b>S</b>	
command .....	76	setting flags.....	47
process command .....	59	single step	
reset .....	22	command .....	76
restart		process command .....	73

single step .....	9	status .....	9
single step flag .....	82	status values	
single-instance PLI		Aborted .....	77
configuration .....	3	Aborting .....	77
writing .....	4	Complete .....	77
single-instance PLI .....	4	Held .....	77
start		Holding .....	77
command .....	63	Idle .....	77
process command .....	63	Restarting .....	77
start .....	63	Running .....	77
state		Stopped .....	77
aborted .....	20	Stopping .....	77
aborting .....	19	status values .....	77
complete .....	77	step index .....	9
held .....	77	step index register .....	82
holding .....	77	stop	
idle .....	77	command .....	76
restarting .....	77	process command .....	56
running .....	77	stop .....	22
stopped .....	77	stopped	
stopping .....	77	state .....	77
state .....	77	stopped .....	20
state transition logic		stopped state active flag .....	82
components of .....	16	stopping	
matrix .....	18	complete .....	40
overview .....	16	process command .....	40
state transition logic .....	16	state .....	77
STATE_STOR .....	56	stopping .....	19



stopping complete flag .....	82	Unit Status tags.....	80
stopping state active flag .....	82	<b>V</b>	
<b>T</b>		variables	
tags		memory variables .....	1
iFIX .....	80	variables.....	1
recommended .....	80	<b>W</b>	
Unit Status .....	80	watchdog	
tags .....	80	checking.....	32
TEMP_CD.....	22	watchdog.....	32
troubleshooting .....	91	watchdog flag .....	82
<b>U</b>		WDOG.....	82
unit.....	8	WDOG_TMR .....	82
unit ID register.....	82		