Proficy Batch Execution 5.6

Phase Programming Manual

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

Table of Contents

About This Guide

The Proficy Batch Execution Phase Programming Manual is a comprehensive guide to developing and using the application interface to Batch Execution 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, ladder logic, and sequential function charts (SFC).

Reference Documents

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

- PLI Development Manual
- ISA-S88.01, Batch Control, Part 1: Models & Terminology

Introduction

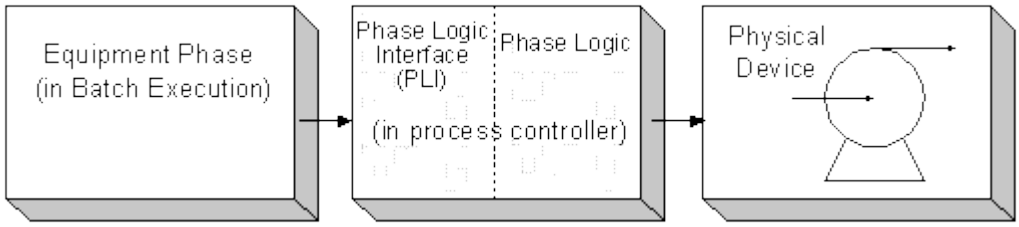
The sections that follow provide an overview of phases in Batch Execution, including the:

- Components that make up a phase.
- Communication process between Batch Execution and the phase logic in the process controller.
- Phase programming requirements that enable equipment to respond to and interact with Batch Execution.

Understanding Phases

A *phase* is a series of steps that cause one or more equipment or process-oriented actions. These actions issue commands to set or change controller constants, modes, or algorithms.

In Batch Execution, *equipment phases* are defined in the equipment database to trigger the execution of the phase logic that resides in the process controller. The phase logic in the process controller contains the instructions to control the physical devices, such as a pump or a motor. The following figure illustrates an equipment phase.



Equipment Phase

Phase Logic Components

The logic in the process controller consists of two components: the phase logic interface (PLI) and the phase logic:

PLI – is the standard interface between the Batch Execution Server and the phase logic. The PLI is the Batch Execution-specific portion of the phase that controls the state transitions for phases. A program listing for a sample PLI is provided in the PLI Development Manual.

Phase Logic – contains the instructions to sequence the individual equipment connected to the physical devices. It is the code that contains the control steps, such as opening a valve, starting a pump, or stopping a totalizer.

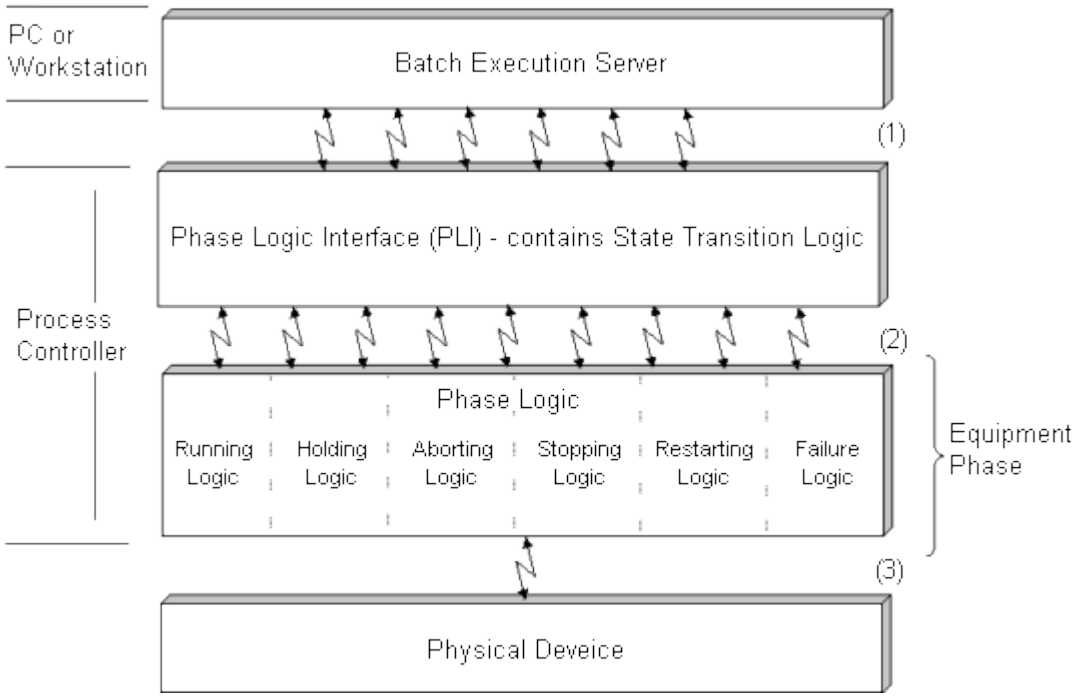
Understanding Communications

The Batch Execution Server and the PLI communicate using a standard set of commands, requests, and other data items. In general, the Batch Execution Server sends commands and phase parameter values to the PLI. In return, requests, status information, and phase report values propagate up from the phase logic.

Communication Interface

In order for the Batch Execution Server, the PLI, and the phase logic to communicate, a specific data structure is necessary. This data structure makes up the communication interface that enables the Batch Execution Server, the PLI, and the phase logic to exchange data, which ultimately controls the execution of a batch.

The following figure illustrates the layers of communication between the Batch Execution Server, the PLI, the phase logic, and the physical device.



Phase Components

Programming Phase Logic

The phase logic contains the instructions to operate a piece of equipment. For example, the control logic in a heat phase may contain the instructions to open and close a steam valve on a heating jacket.

To implement the operating sequences and detect equipment failures, Batch Execution requires that each phase contain five modules of code:

- Running
- Aborting
- Stopping
- Restarting
- Holding

Variables set by the state transition logic initiate these modules of code. When these modules of code terminate, they set a variable that indicates completion. The completion status is read by the state transition logic, allowing the phase to transition to the next state.

Refer to the Programming Project-Specific Phase Logic section for more information.

Variable Names

The recommended naming conventions for the variables set by the state transition logic begin with a unique phase name, followed by an underscore and up to three character identifiers for the variable:

```
PHASE_xxx
```

For example, for a phase called Charge, the recommended variable name for the Running logic is:

```
CHARGE_R
```

Unit Status Tags

Batch Execution includes two mechanisms for defining a unit's status:

- The Unit Ready tag
- The Unit Priority tag

Both of these items are optional and may not be necessary for your particular process. Refer to the Equipment Configuration Manual for additional information on both these items.

Unit Ready Tag

The Unit Ready tag is an integer programmed into the process controller that indicates whether a unit is ready to be used in a batch process. A value of zero (0) indicates that the unit is available to a process or an operator. A value other than zero indicates that the unit is currently unavailable.

Example

On a plant floor, you can program an ON/OFF button on a mixer to use the Unit Ready tag. If the mixer is shut down for any reason, due to equipment failure or routine maintenance, the Unit Ready tag is set to a non-zero integer. This mixer is then unavailable to any operator, as well as any batch process.

Unit Priority Tag

Active Binding allows you to bind and re-bind a physical piece of equipment to a recipe at different times during the production cycle. There are three methods of binding a unit to a unit procedure:

- Manually, when the recipe is created.
- During run time, using an operator prompt.
- Automatically, allowing the Batch Execution Server to make the selection dynamically.

You can use the Unit Priority tag during Active Binding to determine the relative importance of a particular unit, as compared to other units in the same unit class. To do so, assign an integer to each unit's Unit Priority tag. During Active Binding, Batch Execution selects the unit with the highest Unit Priority value from the list of available units.

For additional information on Active Binding, refer to the Recipe Development Manual.

Example

During Active Binding, Batch Execution uses the Unit Priority tag. Assume a plant is running a batch and using Active Binding to automatically determine on which mixer the batch will run. There are three mixers to choose from: MIX1, MIX2, and MIX3. The Batch Execution Server will select the unit with the highest value as its Unit Priority tag, as the following table describes:

Unit Priority Tag Example		
If this unit...	**Has a Unit Priority tag with this value...**	**Then...**
MIX1	7	Batch Execution does *not* select this unit.
MIX2	25	Batch Execution selects this unit.
MIX3	2	Batch Execution does *not* select this unit.

Now, consider the same example, but with the following modifications:

- Add an additional mixer, MIX4.
- Include the evaluation of the Unit Ready tag.

The following table displays the results.

Unit Status Tags Example			
If this unit...	**Has a Unit Priority tag with this value...**	**Has a Unit Ready tag with this value...**	**Then...**
MIX1	7	0	Batch Execution does *not* select this unit.
MIX2	25	3	Batch Execution does *not* select this unit.
MIX3	2	5	Batch Execution does *not* select this unit.
MIX4	16	0	Batch Execution selects this unit.

Design Strategies

When designing your phases, your goal should be to design modular, flexible, and generic phases that you can run on multiple units and use for multiple recipes. Define phases without a particular product in mind. This way, if you alter a product or process, it is less likely you will need to change the phase logic. Essentially, you should have a library of phases that you can select from when creating your recipe procedures. However, you can only effectively design the types and quantities of phases you need if you have a complete understanding of your process, including how the production lines are configured at your facility.

Unit Design

An integral part of phase design is unit design. Your unit design is important in that phases control the equipment that makes up a unit. As defined in the S88.01 Batch Control standard, a unit is comprised of the control modules that perform a specific function. You can't effectively design a phase to run on a unit until you have a complete understanding of how the unit is configured and which I/O points are controlled by the unit.

For example, assume that you have a unit that consists of all the equipment modules that make up a mixer. One of the equipment modules associated with this unit is a cooling jacket. To operate the cooling jacket, you would create a phase that controls the I/O defined for the cooling jacket. For more information on unit design, refer to the Equipment Configuration Manual.

Task Overview: Phase Design

The following list provides an overview of the tasks involved in designing phases.

1. Identify the phases in your process.
2. Identify the phase requirements, including:
    - Requests
    - Parameters
    - Reports
    - Interlocks
    - Interphase communications
3. Write modular, flexible, and generic phases.

The sections that follow include information on identifying the phases in your process, as well as suggested strategies for designing flexible and generic phases. Subsequent sections address the identification and creation of specific phase requirements.

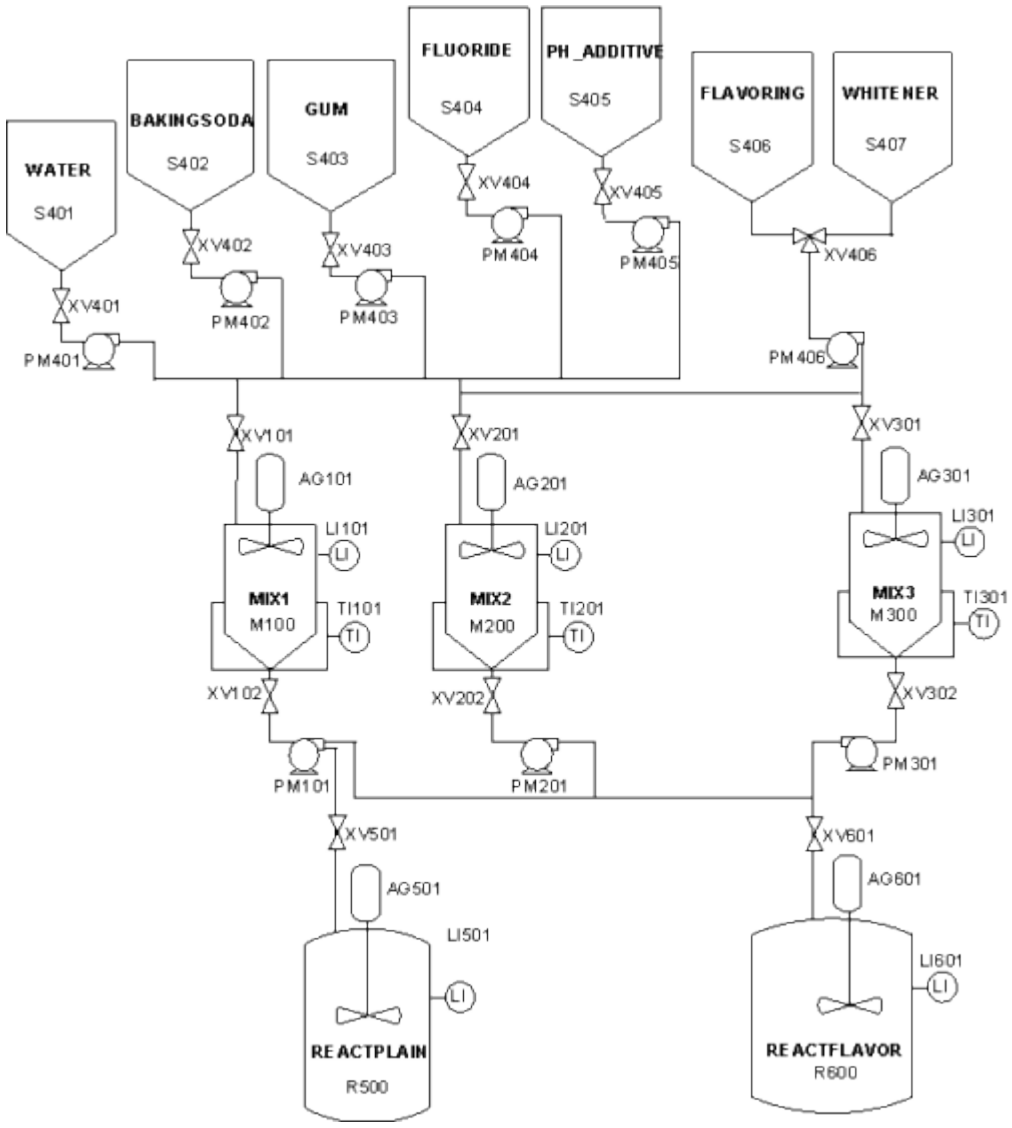
Identifying the Phases in the Process

Before you can begin programming phases, you must first identify the phases in the process. A phase is an independent process-oriented action within an operation. There may be one or more control steps within a phase that may be executed sequentially or concurrently.

Phases execute on equipment modules. When grouped together, equipment modules make up a unit. To identify phases, isolate each equipment or control module and group them by the function they perform. To identify the individual phases that make up the process, refer to your plant's Process and Instrumentation Drawing (P&ID).

Example: Process and Instrumentation Drawing

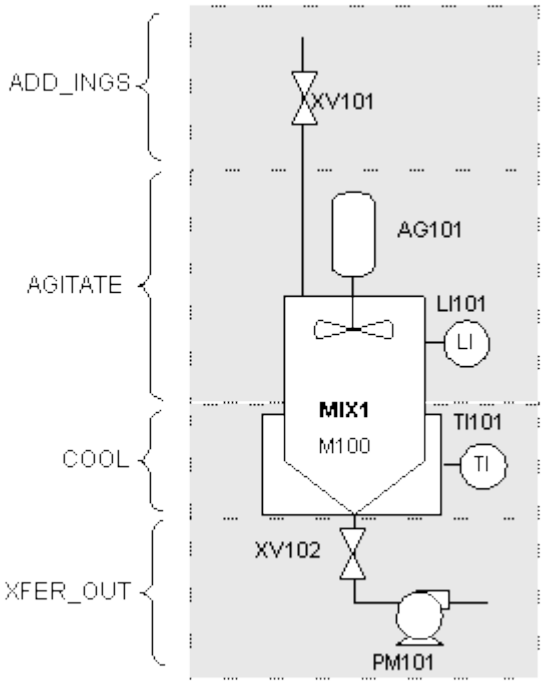
The following figure illustrates a sample Process and Instrumentation Drawing (P&ID) based on the sample toothpaste application.



Sample P&ID

Example: Agitator Phases

For example, the MIX1 mixer contains several phases, all of which operate the equipment and control modules associated with the mixer. The following figure illustrates the phases for the MIX1 unit.



Agitator Phases

The following table describes each phase that controls the Agitator unit:

Sample Agitator Phases	
Phase...	**Function...**
ADD_INGS	Controls valve XV101 to transfer raw material into the mixer.
AGITATE	Controls the agitator motor AG101 to agitate the ingredients in the mixer.
COOL	Controls the temperature of the cooling jacket.
XFER_OUT	Controls valve XV102 and pump PM101 to transfer material out of the mixer.

NOTE: *LI101 and TI101 are level and temperature indicators for the unit. These are defined as unit tags in the Equipment Editor. For more information on unit tags, refer to the Designing Generic Phases section.*

Designing Generic Phases

During phase design, consider the use of phase parameters and unit tags to create generic phases. This is especially important if you have a multi-stream configuration, meaning that you can produce batches of a product on multiple production lines.

Using Phase Parameters

Phase parameters allow you to set values, such as temperature setpoints, that are appropriate for a particular unit on which the phase is executing. Using phase parameters allows you to create one generic phase that can execute on multiple units. Phase parameters also allow you to use the same phase in multiple recipes.

Using Unit Tags

Unit tags are configured for a unit class in the Batch Execution Equipment Editor. For example, the sample P&ID in the Example: Process and Instrumentation Drawing section, the figure of sample P&ID contains three identical mixers. You can assign unit tags to each mixer's temperature and level indicator. You can then define a unit tag class, which provides a generic label that resolves to a specific unit tag at batch run time.

For more information on phase parameters and unit tags, refer to the Equipment Configuration Manual.

Example: Add Ingredient Phase Logic

The phase logic for a typical Add Ingredient phase may contain the following steps to add material to a unit:

1. Download the target amount.
2. Acquire the pump PM401.
3. Reset the totalizer.
4. Open valve VLV01.
5. Open valve VLV02.
6. Start pump PMP01.
7. Wait for the totalizer to reach the target amount.
8. Stop pump PMP01.
9. Close valves.
10. Release pump PMP01.
11. Upload actual amount.
12. Reset phase and clear out old values.

Example: Structured Text

The following code is an example in structured text format of the Add Ingredient phase logic. The step numbering reflects the suggested step numbering scheme in the Implementing a Phase Step Numbering Scheme section.

```
(* CHARGE Running Logic *)
ELSIF (CHARGE_R = 1)
     IF (CHARGE_SI = 1)
          (* Perform Step 1 - Download All Parameters *)
          CHARGE_RQ := 1000
```

```
     (* Change to step 2 so we only set RQ once *)
     CHARGE_SI := 2
ELSIF (CHARGE_SI = 2)
     (* Perform Step 2 - Wait for Param Download *)
     IF (CHARGE_RQ = 0)
          (* Copy downloaded value into local variable *)
          CHARGE_TARGET := CHARGE_P01
          CHARGE_SI := 3
     ENDIF
ELSIF (CHARGE_SI = 3)
     (* Perform Step 3 - Acquire the Pump *)
     CHARGE_RQ := 4011 (* Pump's Resource ID is 11 *)
     CHARGE_SI := 4
ELSIF (CHARGE_SI = 4)
     (* Perform Step 4 - Wait for resource *)
     IF (CHARGE_RQ = 0)
          CHARGE_SI := 101
     ENDIF
ELSIF (CHARGE_SI = 101)
     (* Perform Step 101 - Prepare to Charge *)
     CHARGE_TOTAL := 0
     CHARGE_SI := 102
ELSIF (CHARGE_SI = 102)
     (* Perform Step 102 - Open the valves *)
     VLV01 := 1
     VLV02 := 1
ELSIF (CHARGE_SI = 103)
     (* Perform Step 103 - Start the Pump*)
     PMP01 := 1
ELSIF (CHARGE_SI = 104)
     (* Perform Step 104 - Wait for Target to be reached *)
     IF (CHARGE_TOTAL >= CHARGE_TARGET)
          CHARGE_SI := 105
     ENDIF
ELSIF (CHARGE_SI = 105)
     (* Perform Step 105 - Stop the pump *)
     PMP01 := 0
ELSIF (CHARGE_SI = 106)
     (* Perform Step 106 - Close the valves *)
     VLV01 := 0
     VLV02 := 0
ELSIF (CHARGE_SI = 1001)
     (* Perform Step 1001 - Release the pump *)
     CHARGE_RQ := 4211 (* Pump's Resource ID is 11 *)
     CHARGE_SI := 1002
ELSIF (CHARGE_SI = 1002)
     (* Perform Step 1002 - Wait for release *)
     IF (CHARGE_RQ = 0)
          CHARGE_SI := 1003
     ENDIF
ELSIF (CHARGE_SI = 1003)
     (* Perform Step 1003 - Upload actual amount *)
     CHARGE_R01 := CHARGE_TOTAL
     CHARGE_RQ := 2000
     CHARGE_SI := 1004
ELSIF (CHARGE_SI = 1004)
     (* Perform Step 1004 - Wait for upload *)
     IF (CHARGE_RQ = 0)
```

```
                CHARGE_SI := 1005
            ENDIF
        ELSIF (CHARGE_SI = 1005)
            (* Perform Step 1005 - Reset the phase *)
            CHARGE_TARGET := 0
            CHARGE_RC := 1
        ENDIF
    ENDIF
```

NOTE: *Batch Execution automatically acquires and releases resources. However, if you want to make a resource available prior to the completion of a phase, you can program requests to acquire and release the resource. Otherwise, the resource is unavailable until the phase completes and is reset by Batch Execution. Refer to the Acquiring Resources section for additional information.*

Writing Phase Logic

After you have identified the phases in your process, you can begin writing the phase logic. Designing and programming phase logic is very specific to the target process controller. However, in order for Batch Execution to communicate with the phase logic in the process controller, there are specific items that must be accounted for in your phase logic.

When writing phase logic, you should first ask yourself the following questions:

- What types of requests does this phase require? For example, does this phase need to make a request to download phase parameters? If so, how many phase parameters are required?
- Does this phase execute on multiple units? If so, what logic must be written generically?
- What phase memory variables need to be set to communicate with the PLI?

Using Phase Templates

As you design phases, you may discover that it is easier to create a phase template than to design each individual phase from the ground up. Many elements in the structure of the phase are similar throughout different types of phase logic. For example, in the following samples of code written in structured text format, you can see many similarities between the Aborting logic and the Stopping logic.

Sample of Aborting Logic (as structured text)

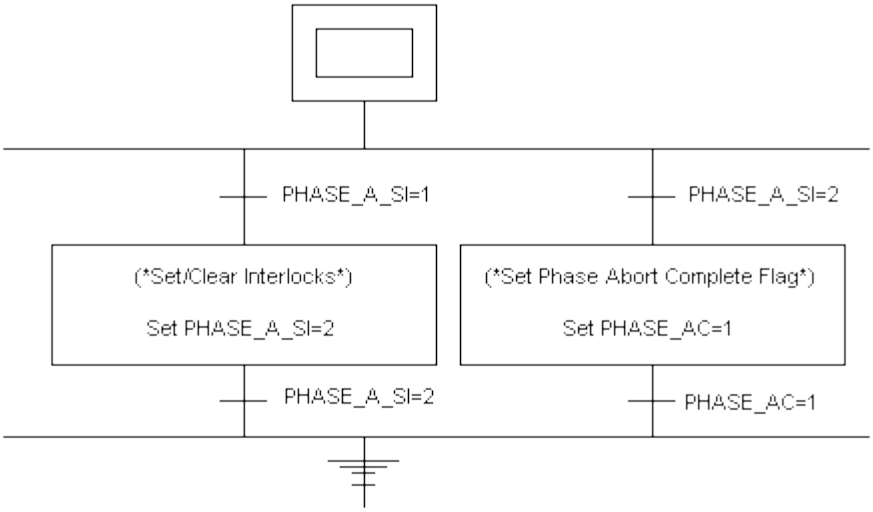
```
IF CHARGE_A <> 0 THEN
    IF(CHARGE_A_SI = 1) THEN
        (*SET/CLEAR INTERLOCKS*)
        CHARGE_A_SI := 2;
    ELSIF(CHARGE_A_SI := 2) THEN
        (*SET THE PHASE ABORT COMPLETE FLAG*)
        CHARGE_AC := 1
    END_IF;
END_IF;
```

Sample of Stopping Logic (as structured text)

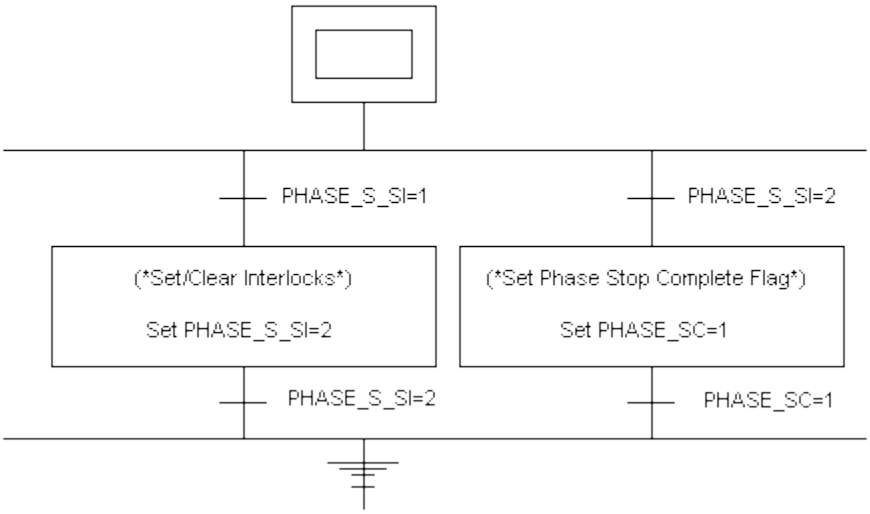
```
ELSIF CHARGE_S <> 0 THEN
     IF (CHARGE_S_SI = 1) THEN
          (*SET/CLEAR INTERLOCKS*)
          CHARGE_S_SI := 2;
     ELSIF(CHARGE_S_SI = 2) THEN
          (*SET THE STOP COMPLETE FLAG*)
          CHARGE_SC := 1
     END_IF;
END_IF;
```

Creating Phase Templates with Sequential Function Charts

You can represent phase logic by using SFCs, as shown in the following two figures. Whether you design your phases using structured text or SFCs, by reusing segments of your design as a template you can help to streamline the phase design process.



Sample of Aborting Logic in SFC Format



Sample of Stopping Logic in SFC Format

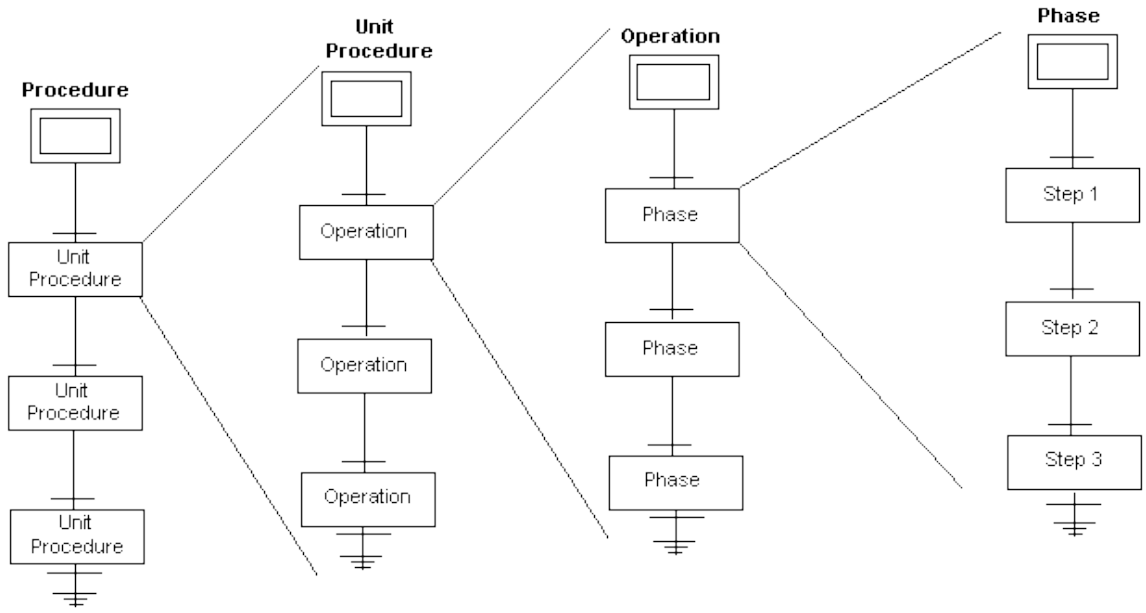
Designing Phase Steps

Much of the work involved with writing phase logic centers around defining a sequence of steps that the phase must do to accomplish a task. These steps are grouped into two categories:

Control steps – to interact with the devices associated with the phase. For example, opening a valve.

Administrative steps – can request information from Batch Execution, can set internal variables, and can handle interlocks. For example, a phase can request the Batch Execution Server to download parameters.

In phase design it is important to consider the individual steps that constitute the entire phase. Phases may contain one or more steps that can execute sequentially or concurrently. SFCs can be an effective way of designing and visualizing the steps within your phases, as shown in the following figure.



Designing Phase Steps

Implementing a Phase Step Numbering Scheme

You should implement a numbering scheme for the steps in your phases, so that you can identify the steps from phase to phase. For example, you can use:

- 1 to 100 for the steps in a phase that perform the initial administrative work. This can include downloading requests or initialization of the phase.
- 101 to 1000 for the actual work of the phase. These steps can include opening a valve or starting a motor.
- 1001 to 1100 for the steps that perform the final administrative work in the phase, such as uploading reports or resetting the phase.

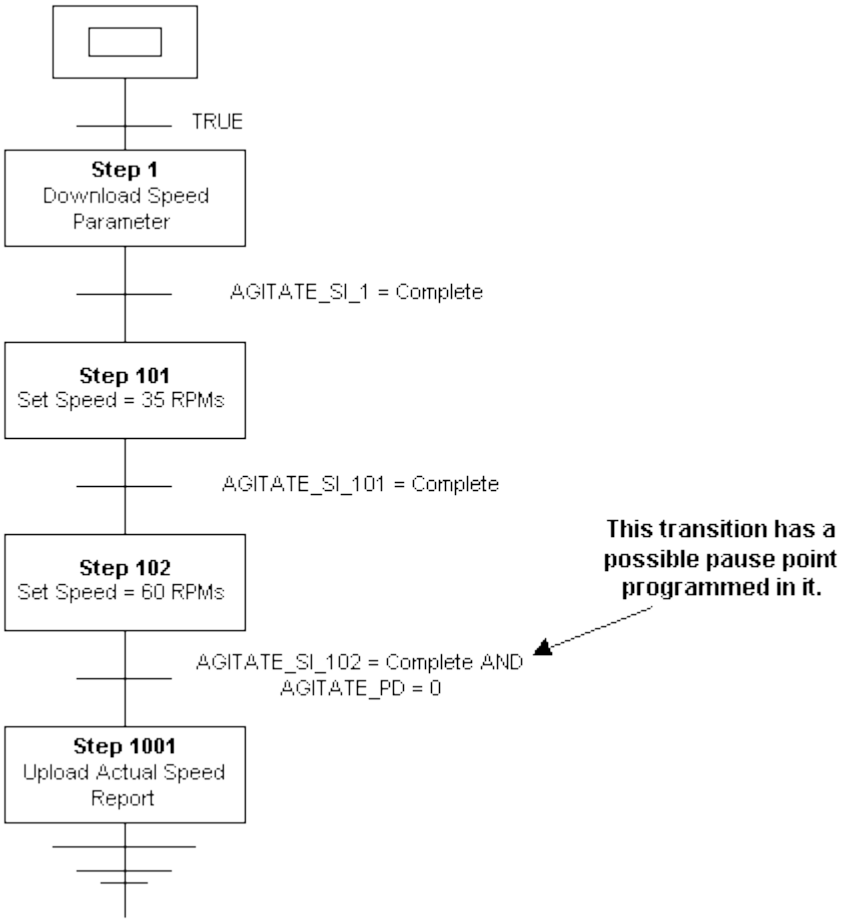
Programming Pauses

When designing phases, you need to program safe pause points into your transition logic. Programming pause logic requires two steps:

1. Recognize the Pause (P) flag and set the Paused (PD) flag.
2. At the transition, test for the Paused (PD) flag.

There may be steps or a series of steps in a phase where pausing cannot be tolerated. For example, a batch may use an extremely viscous substance in its recipe. During the production process, an AGITATE phase, illustrated in the following figure is used to mix the substance. The steps in the AGITATE phase are:

1. Download the speed parameters for the phase.
2. Agitate at a rate of 35 RPMs.
3. Agitate at a rate of 60 RPMs.
4. Upload the actual speed value.



Programming a Pause Point

Due to the viscosity of the substance, pausing between the two agitate steps is impossible. The phase needs to execute Step 101 at 35 RPMs to eventually transition to Step 102, which agitates at 60 RPMs.

However, in this particular scenario, you can program a pause point into the transition after Step 102, before the speed value is uploaded. Write your logic to address this by indicating that at this transition, if the Pause flag is set, put the phase into the Paused state.

Understanding the Step Index and Step Buffer

Batch Execution uses two variables to track the progress of the steps in a phase: the Step Index and Step Buffer.

Step Index – indicates the active phase's currently executing step.

Step Buffer – stores the Step Index value when a phase changes state.

When you implement a consistent phase step numbering scheme, as described in the section, Implementing a Phase Step Numbering Scheme, you can use the Step Index as a quick method to evaluate what type of phase step is executing.

For example, suppose you use the numbers from 1 to 100 for administrative phase steps. An operator monitoring the step index from the Client can determine that a phase with a current Step Index of 27 is performing an administrative task and not executing any of the actual work steps of the phase.

Using the Step Index in Restarting Logic

The step index is an important variable to use when programming restarting logic. Let's look at two possible scenarios:

Example 1: Restarting at the Same Step

A process is underway on a plant floor that uses several units, including MIXER3. During the AGITATE phase, the motor on MIXER3 burns out, causing the process to fail. At this point in the process:

1. The PLI places the phase into the HOLDING state.
2. The PLI copies the step index (SI) into the step buffer (SB).

A mechanic replaces the damaged motor, and the operator issues a restart command to the batch. The following steps occur:

1. The PLI places the phase into the RESTARTING state.
2. The AGITATE phase processes its restarting logic.
3. The phase sets the Restarting Complete flag (RC).
4. The PLI copies the step buffer (SB) to the step index (SI).
5. The Running flag (R) is set.
6. The phase resumes from the same step that was executing when the failure occurred.

Example 2: Restarting at a Different Step

In this second scenario, the motor on MIXER3 burns out again during the AGITATE phase. Specifically, the step index of the phase is 1025, which uploads final report values.

This specific mixture must be agitated at a specific speed for a certain length of time. If for any reason the process is interrupted, you must re-agitate the mixture for the required time and at the required speed.

At this point in the example, the steps are identical to the earlier example, specifically:

1. The PLI places the phase into the HOLDING state.
2. The PLI copies the step index (SI) into the step buffer (SB).

A mechanic replaces the damaged motor and the operator issues a restart command to the batch. The following steps occur:

1. The PLI places the phase into the RESTARTING state.
2. The AGITATE phase processes its restarting logic.
3. The AGITATE phase logic resets the step buffer to a number that allows the important steps of the phase to execute on restart.

   NOTE: *The logic to change the step buffer can also occur in the Holding logic.*

   An example of this restarting code is:

```
IF (PHASE_SB < 100) THEN
     PHASE_SB := 1;
ELSIF(PHASE_SB > 100) AND (PHASE_SB < 1000) THEN
     PHASE_SB := 101;
ELSIF(PHASE_SB > 1000) AND (PHASE_SB < 1100) THEN
     PHASE_SB := 101;
END_IF;
```

4. The Restarting Complete flag (RC) is set.
5. The PLI takes the changed step buffer (SB) and copies it to the step index (SI).
6. The Running flag (R) is set.
7. The phase resumes from the step defined in the restarting logic.

Transfer of Control

Transfer of control occurs when the following conditions exist:

1. Two consecutive phases in an operation are communicating with the same equipment phase.
2. The transition does not contain "Phase.State = Complete" as part of the expression.

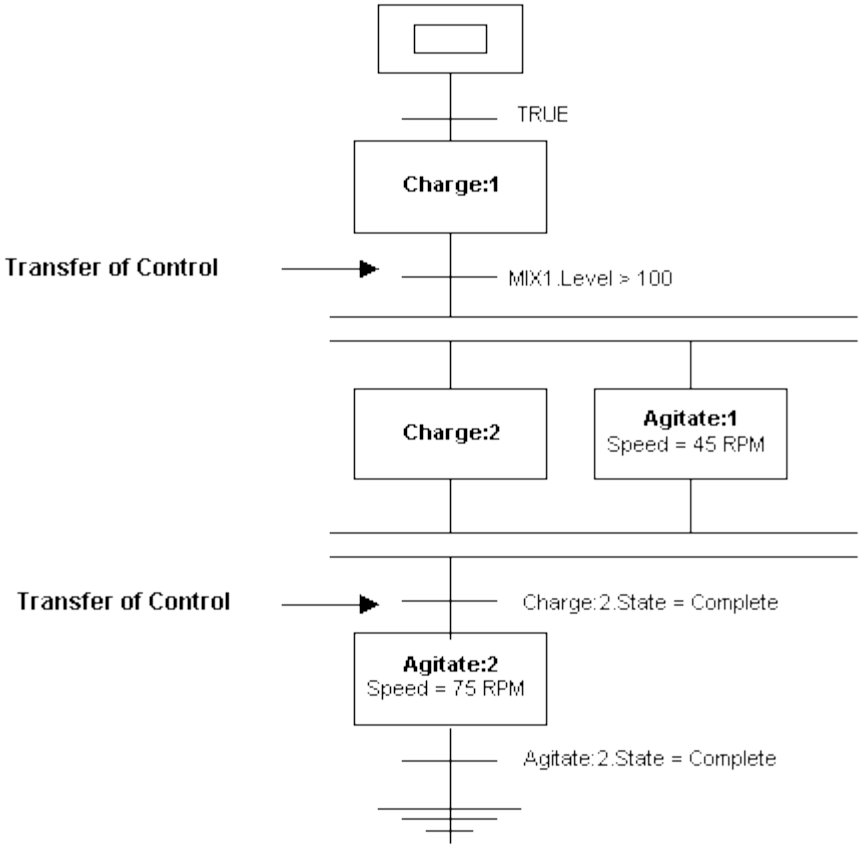
Downloading New Parameters

When a transfer of control occurs:

1. The Batch Execution Server sets the command register to 70 (PHASE_VC = 70).
2. The PLI reads the 70 command and sets the download request flag (PHASE_DRQ = 1).
3. The phase sends a download request command to receive the new parameter values (RQ = 1000).

The following figure shows an operation in which two transfers of control occur.

- The first transfer of control occurs between the two Charge phases. The transfer of control does not affect the Charge phases, so the phases ignore PHASE_DRQ.
- The second transfer of control occurs between the Agitate phases. This transfer of control includes a change in the speed of the Agitate phases, so Agitate:2 sends a download request command and receives the new parameter value.



Transfer of Control

Programming Project-Specific Phase Logic

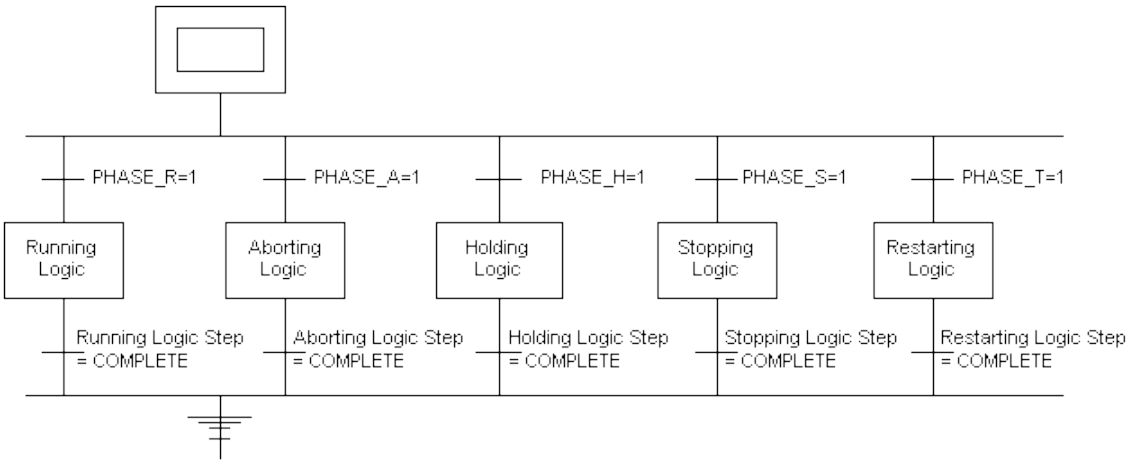
The sections that follow describe how to program your project-specific phase logic for Batch Execution. The *project-specific phase logic* contains the instructions to sequence the individual device drivers connected to the physical devices. It is the code that contains the control steps, such as moving a valve, starting a pump, or resetting a totalizer.

Design the control logic based on your specific processing needs. However, in order for a phase to transition from one state to the next, program the following into the phase logic:

1. Check the status of the designated variable in the state transition logic. For example, the Running logic must read the Running variable (PHASE_R) to determine if the Running logic is active.

2. Set the PHASE_R register to high if the Running logic is active.
3. Set the completion variable to indicate to the state transition logic that the module is complete. For example, when the Running logic for a phase completes, it must set the Running Complete (PHASE_RC) memory variable.

The SFC illustration in the following figure demonstrates the design of a typical phase, including the location of the project-specific phase logic. For additional examples of the project-specific logic in both a structured text and an SFC format, refer to the following sections.



Project-Specific Phase Logic

You can also design the project-specific phase logic using structured text, as shown in the following example:

Project-Specific Phase Logic (as structured text)

```
IF(PHASE_R = 1)
     (*RUNNING LOGIC*);
ELSIF(PHASE_A = 1)
     (*ABORTING LOGIC*);
ELSIF(PHASE_H = 1)
     (*HOLDING LOGIC*);
ELSIF(PHASE_S = 1)
     (*STOPPING LOGIC*);
ELSIF(PHASE_T = 1)
     (*RESTARTING LOGIC*);
END_IF;
```

Understanding Project-Specific Phase Logic

The following table explains the five modules of project-specific phase logic and the recommended naming convention for the variables used to designate the logic. (PHASE represents a unique character identifier for the phase.)

Project-Specific Phase Logic		
Phase Logic	**Description**	**Naming Convention**
Running	Normal operating sequence of a phase.	PHASE_R
Holding	Temporarily suspends the operation of a phase. The phase transitions to the Held state. For example, if a valve malfunctions, the user may want to hold the operation of a phase while the valve is repaired. After repairs are complete, the operator can issue a RESTART command to begin the Restarting logic and transition the phase to the Running state.	PHASE_H
Aborting	Abnormally terminates a phase. The aborting logic is typically reserved to terminate the operation of a phase when an unsafe condition develops. For example, if a pump begins pumping an ingredient onto the plant floor, the operator can issue an ABORT command to terminate the phase. The ABORT command causes the phase to transition through the Aborting state to the Aborted state. The ABORT command normally does not interrupt parameter uploads.	PHASE_A
Stopping	Terminates the phase before normal transition to the Complete state. For example, an operator may decide to transfer 150 gallons of an ingredient to a mixer rather than the 200 gallons specified in the running recipe. The operator can issue a STOP command to terminate the phase before the recipe transfers 200 gallons into the mixer.	PHASE_S
Restarting	Starts the operation of the phase from the Held state.	PHASE_T

Completion Variables

In order to indicate to the state transition logic in the Phase Logic Interface (PLI) that a module of code has completed, you must allocate the completion variables listed in the following table.

Project-Specific Logic Completion Variables	
Recommended Variable Name...	**Indicates...**
PHASE_RC	Running logic has completed.
PHASE_HC	Holding logic has completed.

Project-Specific Logic Completion Variables	
Recommended Variable Name...	**Indicates...**
PHASE_AC	Aborting logic has completed.
PHASE_SC	Stopping logic has completed.
PHASE_TC	Restarting logic has completed.

Programming Running Logic

The Running logic is the module of code that controls the phase during the Running state.

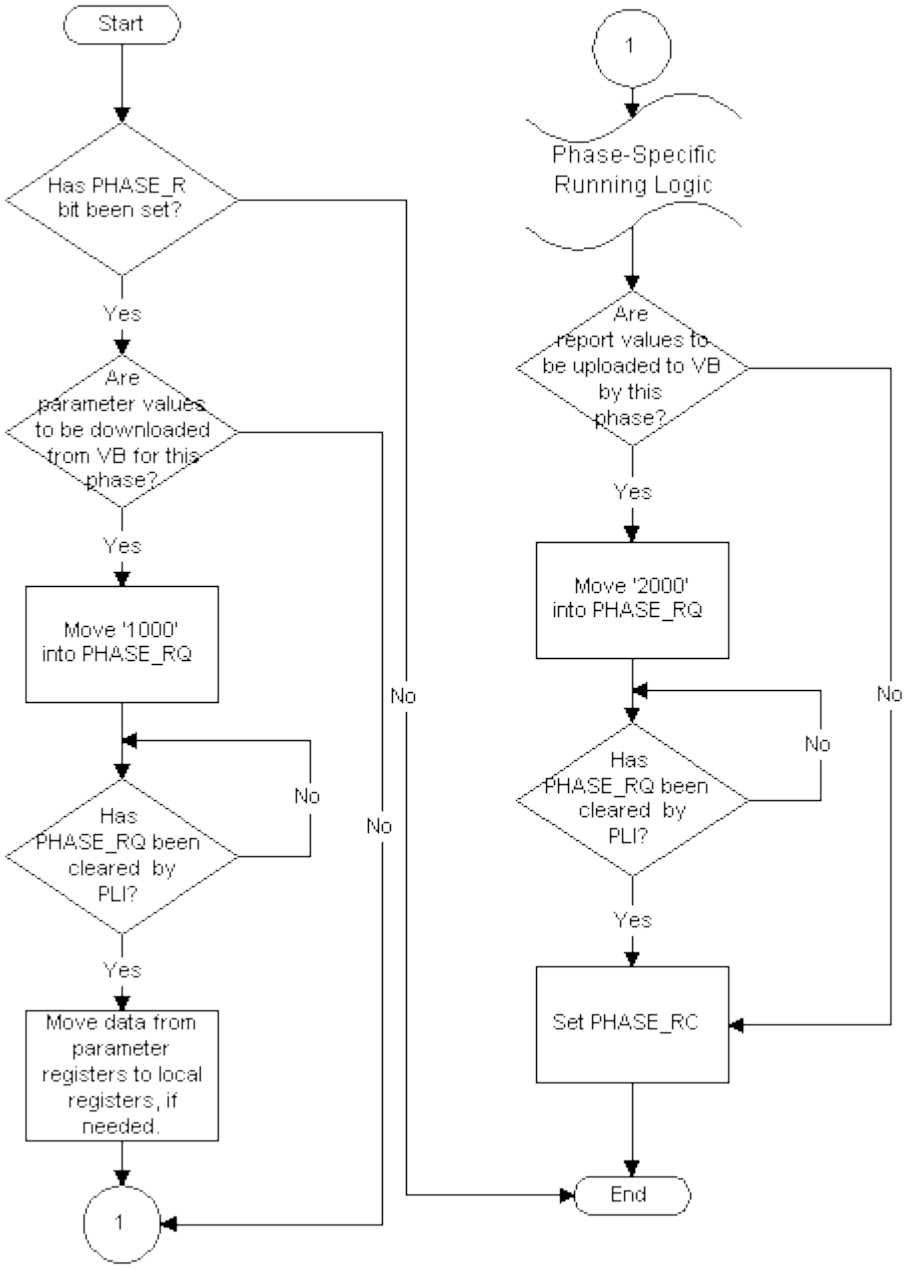
Phase_RQ

The Running logic diagram contains the variable PHASE_RQ. This represents a request function, which is a request from the phase logic to the Batch Execution Server to perform a specific action. In this particular diagram, the first occurrence of PHASE_RQ makes a request to the Batch Execution Server to download parameter values. Later, PHASE_RQ makes a request to upload report values to the Batch Execution Server. For additional information on phase requests, refer to the Programming Requests section.

Running Phase-Specific Logic

The section of the diagram labeled *Running Phase Specific Logic* is where you insert the code that controls the phase's action. For example, if you program the phase to open a valve, the code that instructs the valve to open is located at this point in the logic.

The following figure illustrates the Running logic.

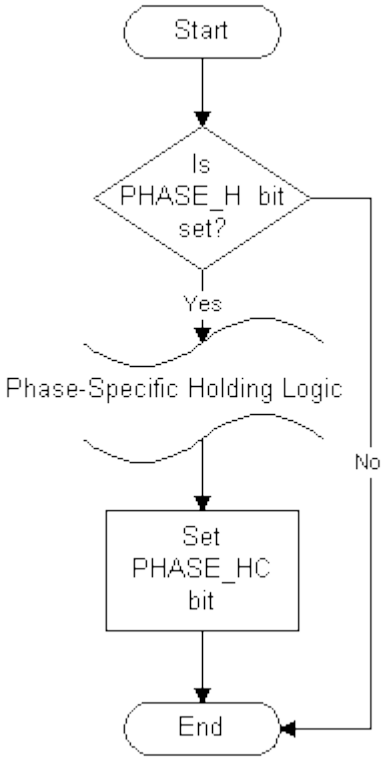


Sample Running Logic

Programming Holding Logic

The Holding logic is the module of code that controls the phase during the Holding state. The following figure illustrates the holding logic.

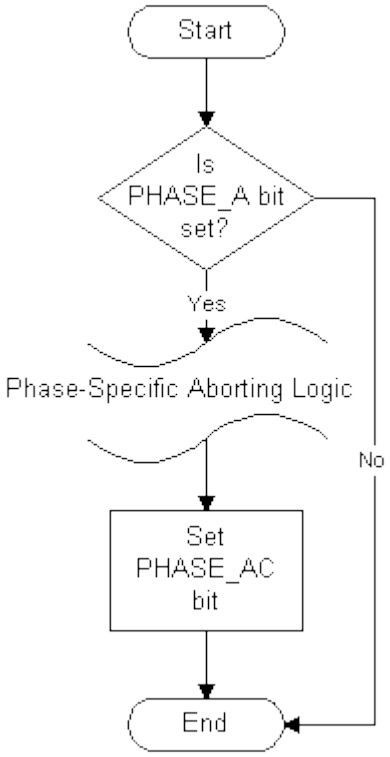
NOTE: *Generally, instructions for dealing with Failures are also programmed into the Holding logic.*



Sample Holding Logic

Programming Aborting Logic

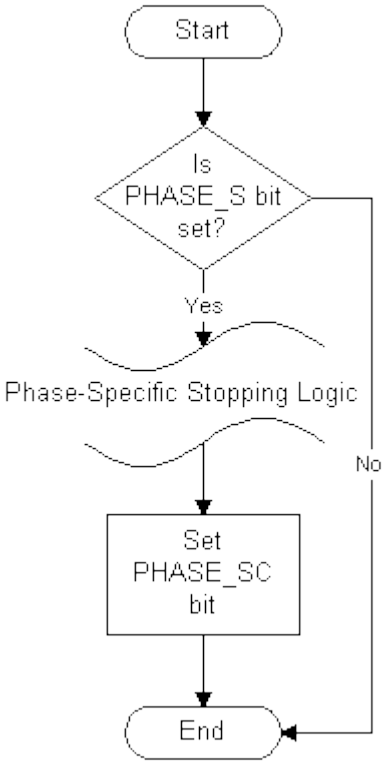
The Aborting logic is the module of code that controls the phase during the Aborting state. The following figure illustrates the aborting logic.



Sample Aborting Logic

Programming Stopping Logic

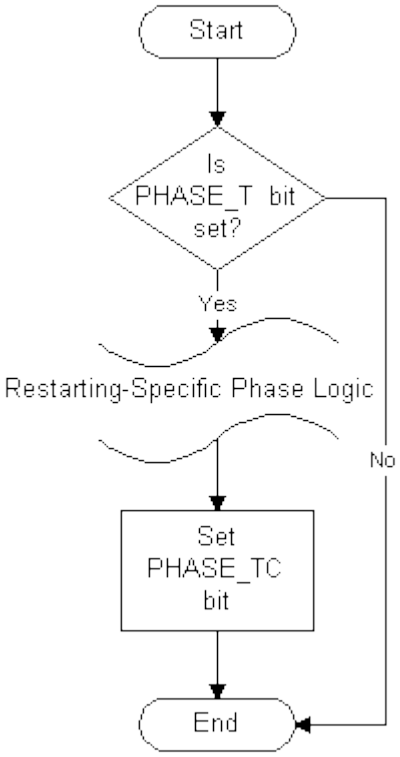
The Stopping logic is the module of code that controls the phase during the Stopping state. The following figure illustrates the stopping logic.



Sample Stopping Logic

Programming Restarting Logic

The Restarting logic is the module of code that controls the phase during the Restarting state. The following figure illustrates the restarting logic.



Sample Restarting Logic

Programming Requests

The Batch Execution Server is event-driven. Requests made by phases are one type of event. The Batch Execution Server responds to requests from phases.

To request information from the Batch Execution Server, the phase logic must set the Request (PHASE_RQ) and Request Data (PHASE_Q*nn*) registers to the appropriate values. Therefore, you need to know what types of requests a phase needs to perform and program the appropriate request logic into the phase logic.

Batch Execution provides a series of request functions that enable the phase logic to request the Batch Execution Server to perform specific actions, including:

- Downloading phase parameter values.
- Sending messages to the operator.
- Making requests to acquire resources.

- Making requests to release resources.
- Uploading report values.
- Sending messages to other phases.
- Canceling messages to other phases.
- Waiting for a message from another phase.
- Aborting a pending request.
- Downloading identification parameters.

Requests are programmed in the project-specific phase logic. For example, you may have an Agitate phase that requires a value to set the mixer speed. In this case, you would program a Download Parameter Request in the Agitate phase logic to download the mixer speed parameter value.

Understanding the Request Variables

The Request and the Request Data memory variables in the phase logic:

- Specify the type of request.
- Clarify the request instructions.

These variables, as shown in the following table, allow the phase logic and the Batch Execution Server to interact. When the Request variables are set to specific values, the Batch Execution Server performs the specified request.

Request Variables	
Data stored in this variable...	**Is used to...**
PHASE_RQ	Specify the type of request that the Batch Execution Server is to perform.
PHASE_Q01 PHASE_Q02 PHASE_Q03 PHASE_Q04 PHASE_Q05	Provide additional data to complete the initial request instructions.

Writing Request Logic

When writing your phase logic, use the sequence listed below to program requests:

1. Prepare for the request (set reports, for example).
2. Set any request parameters, if needed.

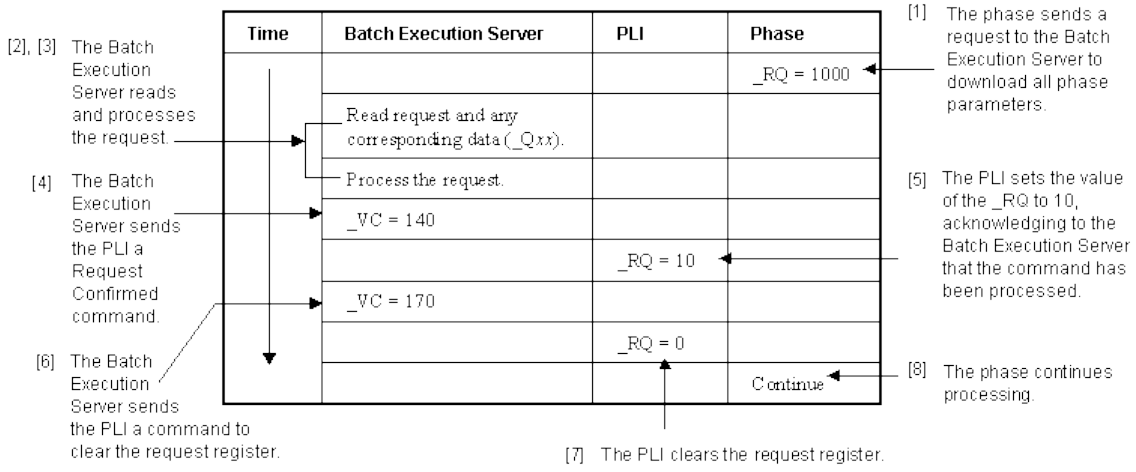
3. Set the request.
4. Wait for the request to be zero.

Processing Requests

Once a request is sent, a handshaking protocol occurs. This method of communication, illustrated in the following figure, ensures that the requests are read, processed, and completed. The following steps occur during the processing of all requests:

1. The phase sends a request to the Batch Execution Server.
2. The Batch Execution Server reads and processes the request.
3. The Batch Execution Server sends the PLI a Request Confirmed command.
4. The PLI sets the value of _RQ to 10, acknowledging to the Batch Execution Server that the command has been processed.
5. The Batch Execution Server sends the PLI a command to clear the request register.
6. The PLI clears the request register.
7. The phase continues processing.

Example: Downloading Parameters



Processing Requests

Configuring Parameter and Report Registers

In addition to setting the Request and Request Data memory variables, your phase logic may also contain:

- Phase parameter registers to store phase parameter values that are downloaded from the Batch Execution Server.
- Report parameter registers to store report values that are uploaded to the Batch Execution Server.

You must make sure you have set up a sufficient number of registers to store the parameter and report values.

Understanding Phase Parameters

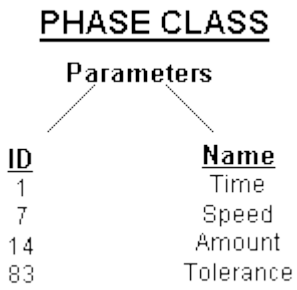
Batch Execution uses three types of phase parameters:

- Phase Class parameters
- Phase Instance parameters
- Process Controller phase parameters

You configure each type of parameter in a different location in Batch Execution, as described in the following sections.

Phase Class Parameters

You configure a phase parameter for a phase class by specifying an ID and a name in the Equipment Editor, as illustrated in the following figure.



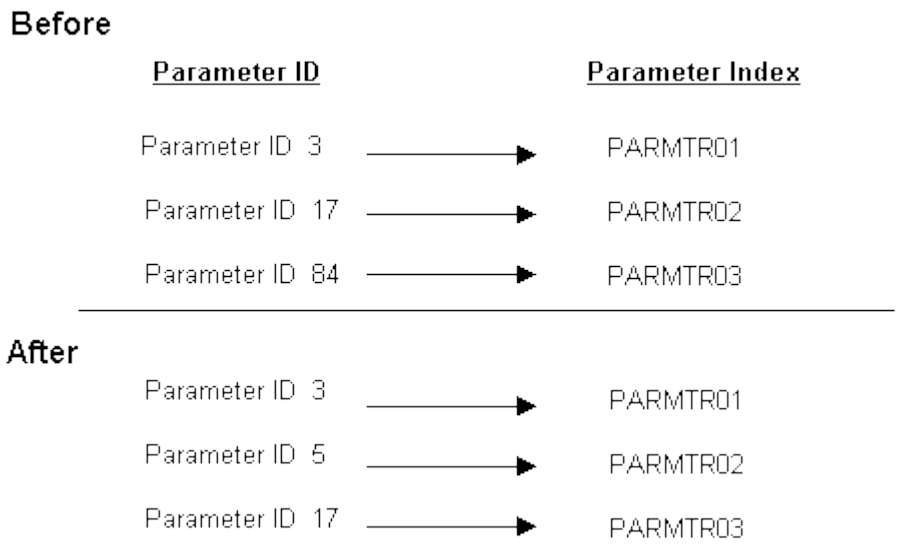
Phase Class Parameters

Phase Instance Parameters

You configure a phase instance parameter in a phase by associating a parameter index with an I/O address. The parameter index number is generated by Batch Execution and is related to the parameter ID. The index numbers start at 1, with parameter index 1 assigned to the lowest numbered parameter ID.

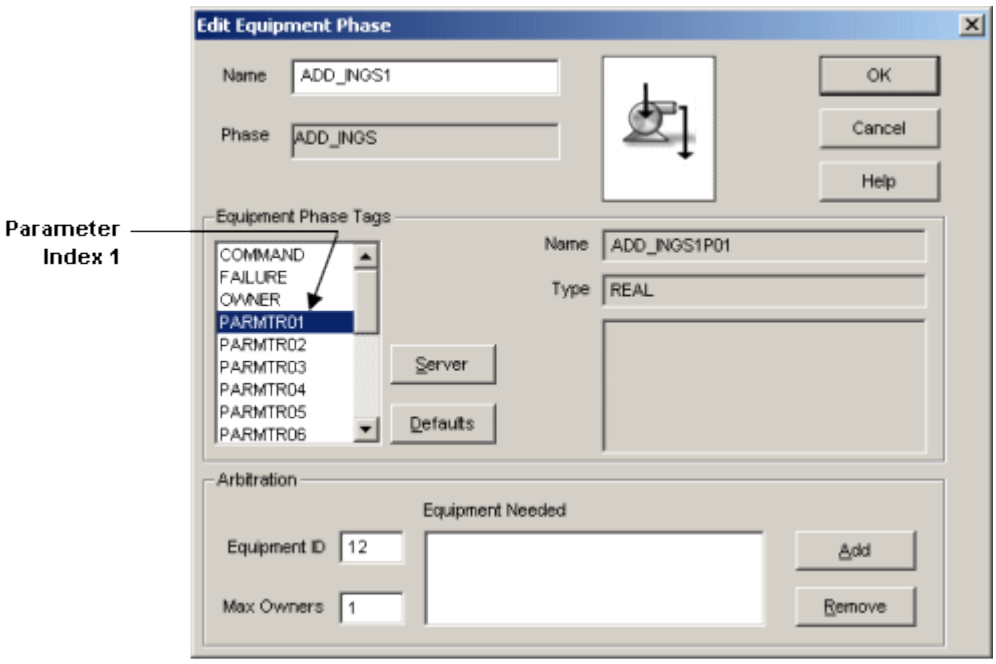
Example

Assume you have parameter IDs of 3, 17, and 84, their corresponding parameter index numbers are 1, 2, and 3, as shown in the following figure. If you renumber parameter 84 to parameter 5, the index numbers assigned to these IDs change. Parameter ID 3 retains the parameter index of 1, the newly renumbered parameter 5 now has the parameter index of 2, and parameter 17 is assigned the parameter index of 3.



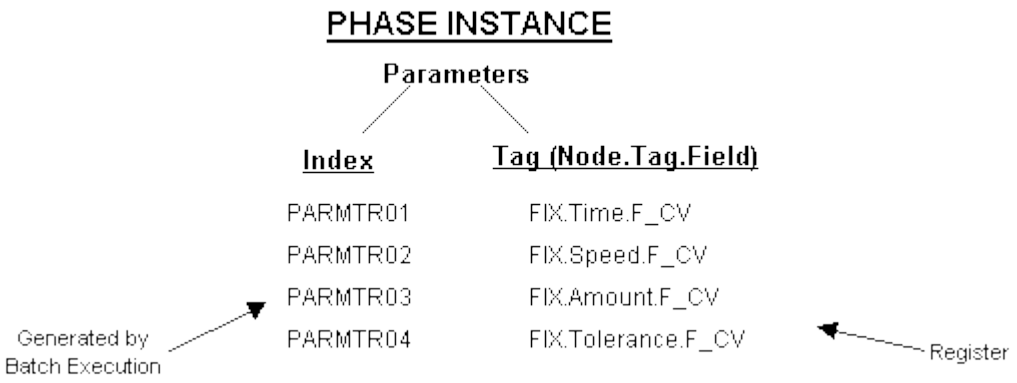
Relationship Between Parameter ID and Parameter Index

The parameter index of each phase parameter is located in the Equipment Editor's Edit Equipment Phase dialog box, as shown in the following figure.



Edit Equipment Phase Dialog Box

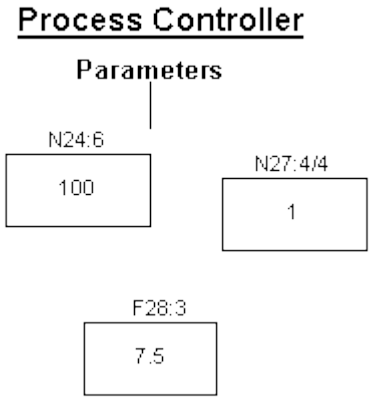
You can use the parameter index and an I/O address to configure a phase instance parameter, as shown in the following figure.



Phase Instance Parameters

Process Controller Phase Parameters

You can configure phase parameters in a process controller by allocating memory registers, as shown in the following figure. These process controller parameters can reside in different files or different process controllers.



Process Controller Phase Parameters

Downloading Parameters

During batch execution, a phase can send a request to the Batch Execution Server to download phase parameter values to the phase in the process controller. You can define download requests to:

- Download all phase parameters.
- Download a range of parameter values.
- Download a single phase parameter.

Understanding the Download Process

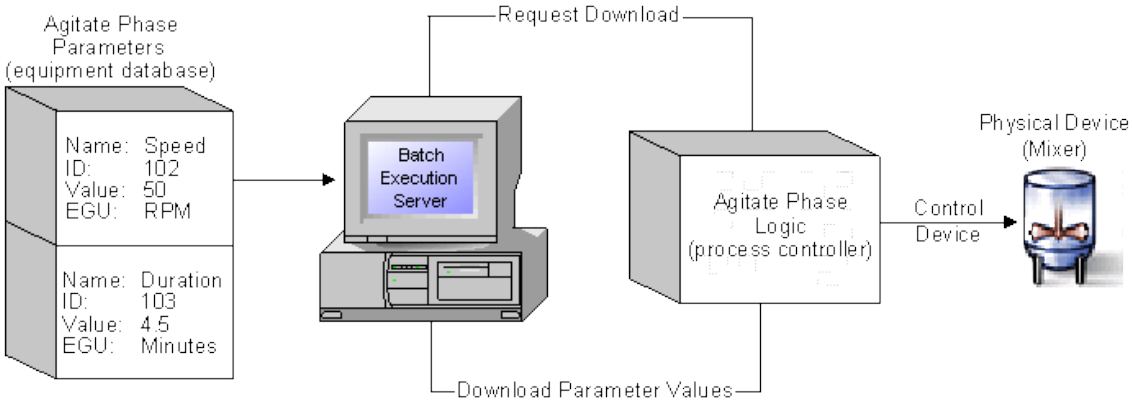
The download process varies slightly depending on the type of download request. In general, a download request specifies the:

- Number of values to download.
- Phase parameter IDs associated with the parameter values.
- Destination location (parameter index) in the process controller to store the parameter values.

NOTE: *The parameter IDs need not increase sequentially.*

Example: Download Parameter Request

Typically, a phase makes a download parameter request when the phase requires parameter values to execute. As illustrated in the following figure, an Agitate phase may require speed and duration values. The Agitate phase class, defined in the Equipment Editor, contains these parameters. During batch production, when the Agitate phase executes, it requests the Batch Execution Server to download the batch-specific parameter values.



Download Request Process

Understanding Phase Class Parameters

You configure phase parameter information including the name, ID, data type, and default value for an equipment phase class in the Equipment Editor. Requests to download a phase parameter value refer to the phase parameter using the parameter ID.

Locating Parameter IDs

If the Batch Execution Server cannot find a specified phase parameter ID, the ID is incremented by one until all the specified phase parameters are found. The specified phase parameters are then downloaded. For example, there may be four phase parameters with parameter IDs of 1, 2, 8, and 10. If the request specifies to download three phase parameters starting at the parameter ID 1, the Batch Execution Server finds phase parameter ID 1, then phase parameter ID 2. When no phase parameter with the ID of 3 is found, the Batch Execution Server increments by one until phase parameter ID 8 is located. The three phase parameters, 1, 2, and 8 are downloaded.

This process repeats until the originally requested ID is incremented by 1000. If the requested number of phase parameters is not located after the parameter ID is incremented by 1000, the Hold logic for the phase step is executed.

Obtaining Parameter Values

Phase parameter values are specified when a recipe is built, when the batch is first started, or during batch execution by the operator. If a parameter value is not specified when the recipe is built and the Batch Execution Server receives a request from a phase to download a value, the Batch Execution Server forwards this request to the Batch Execution Client and prompts the operator to supply the parameter value.

For More Information

For information on configuring equipment phase class parameters, refer to the Equipment Configuration Manual. For information on defining parameter values, refer to the Recipe Development Manual.

Syntax: Download Request

The following table lists each type of Download Phase Parameter request and the request variable values that must be specified.

Download Phase Parameter Requests		
To Download...	**Use This Request...**	**Where...**
All phase parameter values, starting at parameter index 1.	PHASE_RQ = 1000	(No further clarification is required.)
A range of parameter values.	PHASE_RQ = 11*nn* PHASE_Q01 = *ID* PHASE_Q02 = *index*	*nn* is the number of parameters to download, ranging from 1 to 99. *ID* is the parameter ID of the first phase parameter to download. *index* is the parameter index in which to store the first parameter value.

Download Phase Parameter Requests		
To Download...	**Use This Request...**	**Where...**
A range of 100 or more parameter values.	PHASE_Q01 = *ID* PHASE_Q02 = *index* PHASE_Q03 = *nnn* PHASE_RQ = 1100	*ID* is the parameter ID of the first phase parameter to download. *index* is the parameter index in which to store the first parameter value. *nnn* is the number of parameters to download.
A single parameter value, starting at the parameter ID.	PHASE_RQ = 12*nn*	*nn* is the parameter ID.
A single parameter value with an ID greater than 99 and starting at the parameter ID.	PHASE_RQ = 1200 PHASE_Q01 = *ID*	*ID* is the parameter ID.
Single parameter value stored in a specific parameter index.	PHASE_RQ = 13*nn* PHASE_Q01 = *index*	*nn* is the parameter ID. *index* is the parameter index in which to store the parameter value.
Single parameter value with an ID greater than 99 and stored in a specific parameter index.	PHASE_RQ = 1300 PHASE_Q01 = *index* PHASE_Q02 = *ID*	*index is the parameter index in which to store the parameter value.* *ID* is the parameter ID.

Examples: Download Requests

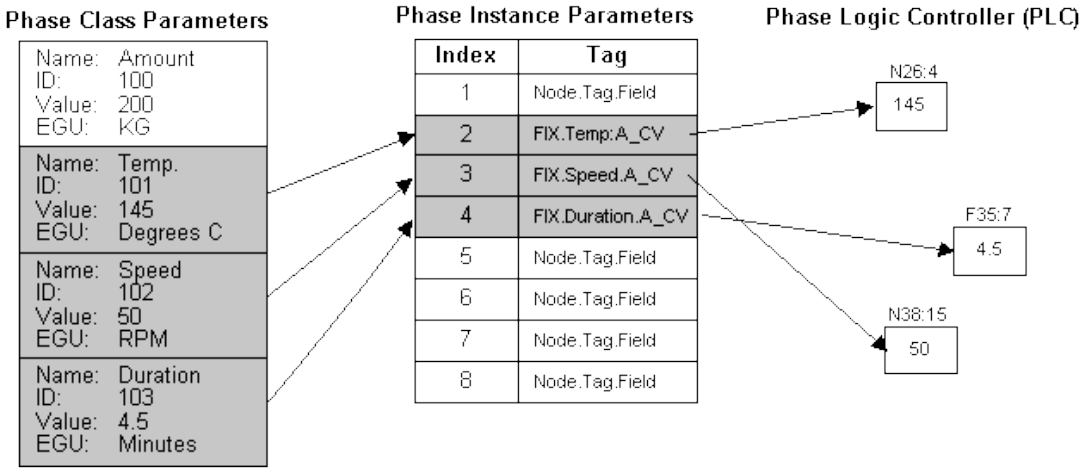
The Download Phase Parameter requests listed below instruct the Batch Execution Server to download three parameters, starting with parameter ID 101. When obtained, the parameter values are stored in the parameter index defined for the phase in the process controller, starting at parameter index 2.

```
PHASE_RQ = 1103

PHASE_Q01 = 101

PHASE_Q02 = 2
```

The following figure illustrates this request.



Downloading a Range of Phase Parameters

Uploading Report Values

During batch execution, a phase can request the Batch Execution Server to upload report values. This process is very similar to downloading parameters. A phase can:

- Upload all report parameters (most common request).
- Upload a range of report parameters.
- Upload a single report parameter.

Batch Event Journal Entries

The uploaded report values are combined with the following information and then written as journal entries into the batch event journal:

- Batch ID
- Recipe Name
- Process Cell
- Time/Date
- Unit ID
- Area
- Phase Name
- Engineering Units
- Report Description

Understanding the Upload Process

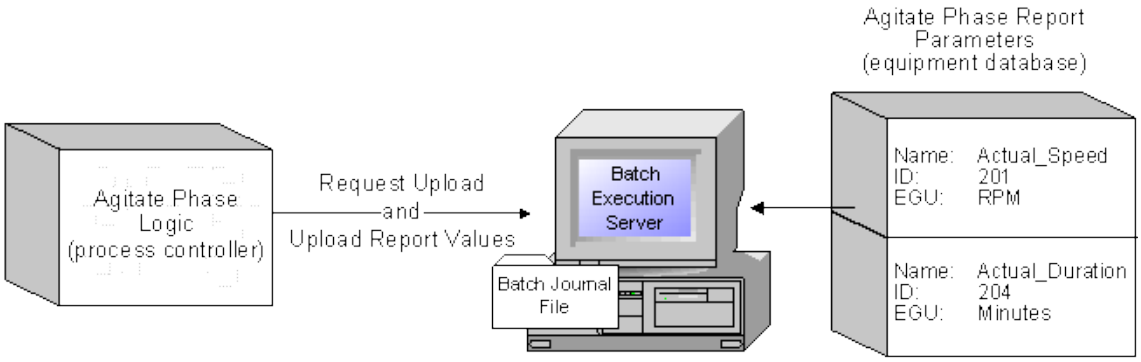
The upload process varies depending on the type of upload request. In general, the request specifies the:

- Number of values to upload.
- Report parameter IDs associated with the parameter values.
- Source location (parameter index) in the process controller to retrieve the report parameter values.

NOTE: *The report IDs need not increase sequentially.*

Example: Upload Report Parameter Request

Typically, an Upload Report Parameter request is made by a phase to report actual values. As illustrated in the following figure, an Agitate phase may request to upload the actual speed and duration values. The Agitate phase class, defined in the Equipment Editor, contains these report parameters.



Upload Requests Process

Understanding Report Parameters

Report parameter information including the name, ID, data type, and engineering units are configured for an equipment phase class in the Equipment Editor. Requests to upload a report value refer to the report parameter using a report ID.

Locating Report IDs

If the Batch Execution Server cannot find a specified report parameter ID, the ID is incremented by one until all the specified report parameters are found. The specified report parameters are then uploaded. For example, there may be four report parameters with parameter IDs of 1, 2, 8, and 10. If the request specifies to upload three report parameters starting at the report parameter ID 1, the Batch Execution Server finds report parameter ID 1 and then report parameter ID 2. When no report parameter with the ID of 3 is found, the Batch Execution Server increments by one until report parameter ID 8 is located. The three report parameters, IDs 1, 2, and 8, are uploaded.

This process is repeated until the originally requested ID has been incremented by 1000. If the requested number of report parameters is not located after the report ID is incremented by 1000, the Hold logic for the phase step is executed.

For More Information

For information on configuring report parameters, refer to the Equipment Configuration Manual.

Syntax: Upload Report Parameter Request

The following table lists each type of Upload Report Parameter request and the request variable values that must be specified.

Upload Report Parameter Requests		
To Upload...	**Use This Request...**	**Where...**
All report values, starting from a value of 1.	PHASE_RQ = 2000	(No further clarification is required.)
A range of report values.	PHASE_RQ = 21*nn* PHASE_Q01 = *report_id* PHASE_Q02 = *index*	*nn* is the number of report values to upload, ranging from 1 to 99. *report_id* is the report ID in the phase class definition of the first report value to upload. *index* is the report identifier in the equipment phase instance from which to retrieve the first report value.
A range of 100 or more report values.	PHASE_Q01 = *report_id* PHASE_Q02 = *index* PHASE_Q03 = *nnn* PHASE_RQ = 2100	*report_id* is the report ID in the phase class definition of the first report value to upload. *index* is the report identifier in the equipment phase instance from which to retrieve the first report value. *nnn* is the number of report parameters to upload.
A single report value, retrieved from report ID number 1.	PHASE_RQ = 22*nn*	*nn* is the report ID.
A single report value with an ID greater than 99 and retrieved from report ID number 1.	PHASE_RQ = 2200 PHASE_Q01 = *report_id*	*report_id* is the report ID in the phase class definition.

Upload Report Parameter Requests		
To Upload...	**Use This Request...**	**Where...**
Single report value retrieved from a specific identifier.	PHASE_RQ = 23*nn* PHASE_Q01 = *index*	*nn* is the report parameter ID. *index* is the report identifier in the equipment phase instance from which to retrieve the first report value.
Single report value with an ID greater than 99 and retrieved from a specific identifier.	PHASE_RQ = 2300 PHASE_Q01 = *index* PHASE_Q02 = *report_id*	*index* is the report identifier in the equipment phase instance from which to retrieve the first report value. *report_id* is the report ID in the phase class definition of the report value to upload.

Example: Uploading a Range of Report Parameters

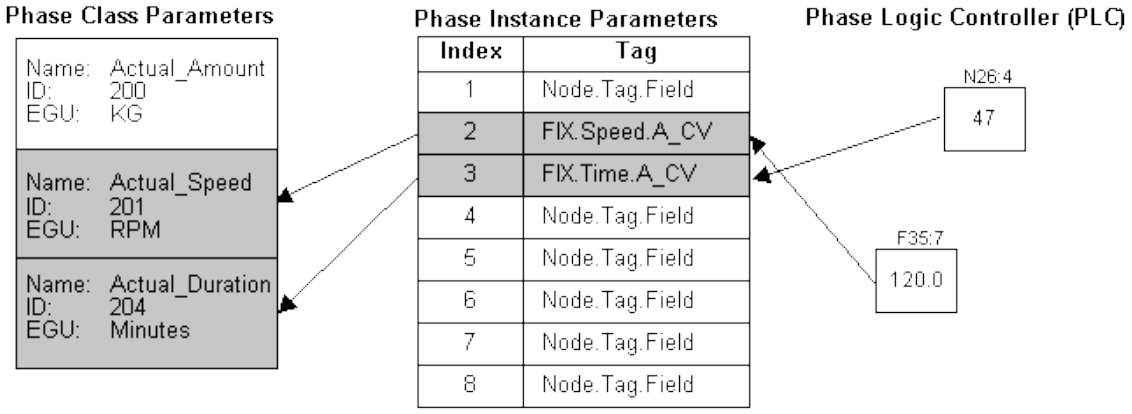
The Upload Report Parameter request listed below instructs the Batch Execution Server to upload two report parameters, starting with report ID 201. The report parameter values are retrieved from the report tags defined for the phase in the process controller, starting at parameter index 2.

```
PHASE_RQ = 2102

PHASE_Q01 = 201

PHASE_Q02 = 2
```

The following figure illustrates this request.



Uploading a Range of Report Parameters

Sending Messages to the Operator

During batch execution, a phase can request the Batch Execution Server to send a pre-configured operator message to the Batch Execution Client. This message is:

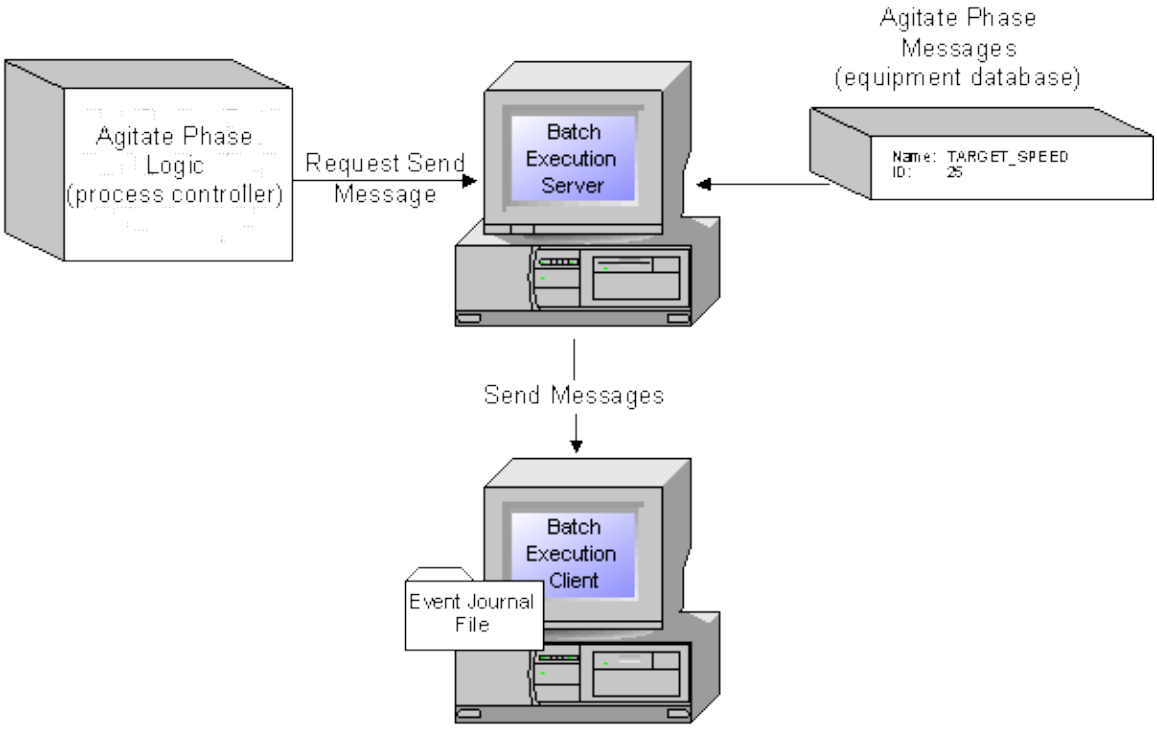
- Displayed to the operator, if the phase is executed manually by the operator.
- Sent to the batch event journal. Every batch produces a batch event journal file. This file is in ASCII format and is displayed in the Batch Execution Client.

Understanding the Send Message Process

The Send Message request specifies the message ID to send to the Batch Execution Client.

Example: Send Message Request

The following figure illustrates a typical Send Message request. The Agitate phase requests the Batch Execution Server to send the TARGET_SPEED message when the phase reaches the programmed target speed.



Send Message Request Process

Understanding Operator Messages

You configure operator message information for an equipment phase class in the Equipment Editor. The message information includes the message name and the message ID. The message name is the message text that is sent to the event journal file or displayed to the operator. Requests to send an operator message refer to the message using the message ID.

Syntax: Send Operator Message Request

The following table lists each type of Send Message request and the request variable values that must be specified.

Send Operator Message Requests		
To Send...	**Use This Request...**	**Where...**
A message with an ID from 1 to 99.	PHASE_RQ = 30*nn*	*nn* is the ID of the message to send.
A message with an ID greater than 99.	PHASE_RQ = 3000 PHASE_Q01 = *nnn*	*nnn* is the ID of the message to send.

Example: Send Message Request

The instructions below request the Batch Execution Server to send the message ID 25.

```
PHASE_RQ = 3025
```

The instructions below request the Batch Execution Server to send the message ID 125.

```
PHASE_RQ = 3000
PHASE_Q01 = 125
```

Acquiring Resources

During batch execution, a phase can request the Batch Execution Server to acquire a pre-configured resource, reserving the resource for the phase. A phase can:

- Acquire a single resource.
- Acquire multiple resources, up to five.

Understanding the Acquire Resource Process

Each resource (for example, a unit or an equipment module) is assigned an equipment ID during configuration in the Equipment Editor. Requests to acquire a resource refer to the resource using the equipment ID.

During configuration, you can assign required resources to a phase. For example, an Agitate phase that executes on a mixing fork may also require additional equipment to execute. Upon execution, the Agitate phase acquires all required resources.

When a phase begins execution, it acquires all required resources before the phase transitions from the Idle state to the Running state. When the phase completes (either normally or abnormally), all acquired resources are automatically released.

NOTE: *Most arbitration occurs automatically. This request is generally used to acquire additional resources, or to gain additional control over when the acquisition or release of resources occurs.*

Syntax: Acquire Resource Request

The following table lists each type of Acquire Resource request and the request variable values that must be specified.

Acquire Resource Requests		
To Acquire...	**Use This Request...**	**Where...**
A single resource with an equipment ID ranging from 1 to 99.	PHASE_RQ = 40*nn*	*nn* is the equipment ID to acquire.
A single resource with an ID greater than 99.	PHASE_RQ = 4000 PHASE_Q01 = *nnn*	*nnn* is the equipment ID to acquire.
Multiple resources.	PHASE_RQ = 41*nn* PHASE_Q01 = *nnn* PHASE_Q02 = *nnn* PHASE_Q03 = *nnn* PHASE_Q04 = *nnn* PHASE_Q05 = *nnn*	*nn* is the number of resources to acquire, up to five. *nnn* is the equipment ID to acquire.

Releasing Resources

During batch execution, a phase can issue a request to the Batch Execution Server to release a resource, making the resource available to other phases. A phase can:

- Release a single resource.
- Release multiple resources, up to five.
- Release all currently acquired resources.

Understanding the Release Resource Process

When a phase begins execution, it must acquire all its required resources before the phase can transition to the Running state. This is done using the Acquire Resource request, described in the Acquiring Resources section. When a phase completes, it automatically releases its acquired resources.

The Release Resource request allows a phase to release previously acquired resources prior to the completion of a phase, making the resource available to other phases. Requests to release a resource refer to the resource using the equipment ID, which is assigned during configuration in the Equipment Editor.

Syntax: Release Resource Request

The following table lists each type of Release Resource request and the request variable values that must be specified.

Release Resource Requests		
To Release...	**Use This Request...**	**Where...**
A single resource with an equipment ID ranging from 1 to 99.	PHASE_RQ = 42*nn*	*nn* is the equipment ID to release.
A single resource with an ID greater than 99.	PHASE_RQ = 4200 PHASE_Q01 = *nnn*	*nnn* is the equipment ID to release.
Multiple resources.	PHASE_RQ = 43*nn* PHASE_Q01 = *nnn* PHASE_Q02 = *nnn* PHASE_Q03 = *nnn* PHASE_Q04 = *nnn* PHASE_Q05 = *nnn*	*nn* is the number of resources to release, up to five. *nnn* is the equipment ID to release.
All currently acquired resources.	PHASE_RQ = 4400	(No further clarification is required.)

Sending and Waiting for Phase Messages

Phases that are members of a synchronization group can communicate with each other. A phase can:

- Send a message to another phase.
- Send a message and wait for confirmation of the responses from all receivers.
- Send a message and wait for confirmation of the responses from one specific receiver.
- Notify the Batch Execution Server that the phase is prepared to receive a message or a set of messages.

You can use this communication for the following purposes:

Synchronization – ensures that multiple phases are in exactly the proper state before they proceed.

Permissive – ensures that one phase in the synchronization group has passed a certain point before other phases can proceed.

Data Transfer – moves data from one phase to another. Data is transferred by sending and receiving message values, which are stored in one or more Request Data variables.

Understanding Synchronization Groups

In order for phases to communicate, they must be part of the same synchronization group. To create a synchronization group, the following must be configured in Batch Execution:

- During equipment configuration, specify the number of *phase partners* for a phase. Phase partners identify the number of phases with which the phase can communicate.
- During recipe development create a *phase link group*, which lists the group of phases that can communicate.

For More Information

For more information on phase partners, refer to Equipment Configuration Manual. For more information on phase link groups, refer to the Recipe Development Manual.

Understanding the Send and Receive Message Process

The Send Message request typically works with the Receive Message function. A Send Message Wait request and a Receive Message Wait request pair can complete only if the message IDs for each of the requests are identical. This prevents messages from being routed to the improper request.

Send Message Process

Upon receiving the Send Message request, the Batch Execution Server:

1. Stores the message in the message queue.
2. Responds to any outstanding or incoming Receive Message Wait requests from other phases within the synchronization group.

Receive Message Process

When a phase receives a Receive Message Wait request, the Batch Execution Server:

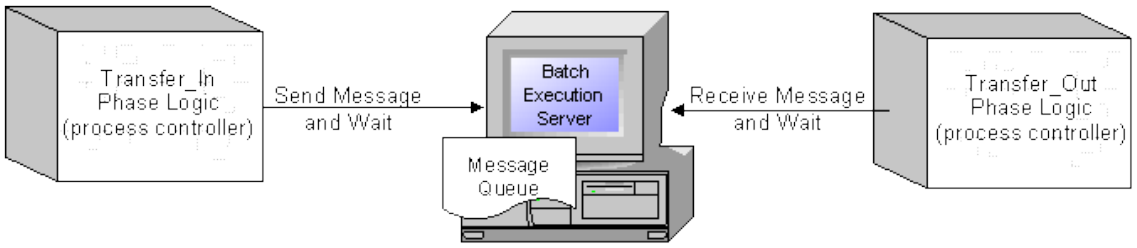
1. Scans the synchronization group for pending messages of matching message IDs.
2. Completes the message transfer by storing the message values in the Request Data memory variables.
3. Removes the message from the message queue.
4. Clears the request code in the phase logic.

Example: Send Message and Receive Message Requests

Typically, you use a Send Message request and a Receive Message request to synchronize two phases. The Send Message request functions as the Master and the Receive Message request functions as the Slave during phase synchronization.

As illustrated in the following figure, a Transfer_In phase and a Transfer_Out phase synchronize the transfer of materials from one unit to another. Before the material is transferred:

1. The Transfer_In phase issues a Send Message request, which the Batch Execution Server stores in the message queue.
2. The Transfer_Out phase issues a receive message request to the Batch Execution Server.
3. The Batch Execution Server looks for the matching message ID in the message queue.
4. The Batch Execution Server transfers the message values.
5. The Transfer_In and the Transfer_Out phases continue and the material is transferred.



Send and Receive Message Process

Syntax: Send Message Request

The following table lists each type of Send Message request and the request variable values that must be specified.

Send Message Requests		
To Send...	**Use This Request...**	**Where...**
A message to a phase.	PHASE_RQ = 50*nn* PHASE_Q01 = *nnn* PHASE_Q02 = *val1* PHASE_Q03 = *val2* PHASE_Q04 = *val3* PHASE_Q05 = *val4*	*nn* is the message ID to send. *nnn* is the number of phases to receive the message. *val1-val4* are message values.
A message to a phase with a message ID greater than 99.	PHASE_RQ = 5000 PHASE_Q01 = *nnn* PHASE_Q02 = *nn* PHASE_Q03 = *val1* PHASE_Q04 = *val2* PHASE_Q05 = *val3*	*nnn* is the message ID to send. *nn* is the number of phases to receive the message. *val1-val3* are message values.

Syntax: Send Message and Wait Request

The following table lists each type of Send Message and Wait request and the request variable values that must be specified.

Send Message and Wait Requests		
To Send...	**Use This Request...**	**Where...**
A message to a phase and wait for a response from all receivers.	PHASE_RQ = 51*nn* PHASE_Q01 = *nnn* PHASE_Q02 = *val1* PHASE_Q03 = *val2* PHASE_Q04 = *val3* PHASE_Q05 = *val4*	*nn* is the message ID to send. *nnn* is the number of phases to receive the message. *val1- val4* are message values.
A message to phase with a message ID greater than 99 and wait for a response from all receivers.	PHASE_RQ = 5100 PHASE_Q01 = *nnn* PHASE_Q02 = *nn* PHASE_Q03 = *val1* PHASE_Q04 = *val2* PHASE_Q05 = *val3*	*nnn* is the message ID to send. *nn* is the number of phases to receive the message. *val1-val3* are message values.

Send Message and Wait Requests		
To Send...	**Use This Request...**	**Where...**
A message to a phase and wait for a response from one receiver.	PHASE_RQ = 52*nn* PHASE_Q01 = *val1* PHASE_Q02 = *val2* PHASE_Q03 = *val3* PHASE_Q04 = *val4* PHASE_Q05 = *val5*	*nn* is the message ID to send. *val1*-*val5* are message values.
A message to a phase with an ID greater than 99 and wait for a response from one receiver.	PHASE_RQ = 5200 PHASE_Q01 = *nnn* PHASE_Q02 = *val1* PHASE_Q03 = *val2* PHASE_Q04 = *val3* PHASE_Q05 = *val4*	*nnn* is the message ID to send. *val1*-*val4* are message values.

Example: Sending and Receiving Messages

The following table contains send and receive instructions, which are paired to synchronize and transfer data between a Transfer_Out and a Transfer_In phase. In this example, a value must be passed to the Transfer_In phase to indicate the flow rate at which the Transfer_Out phase will transfer material.

Sending and Receiving Messages	
Transfer_Out Phase (send message and wait)	**Transfer_In Phase (receive message and wait)**
PHASE_RQ = 5225 PHASE_Q01 = 80	PHASE_RQ = 5525

Upon receiving the Send Message request from the Transfer_Out phase, the Batch Execution Server stores message ID 25 in the message queue. Upon receiving the receive message request from the Transfer_In phase, the Batch Execution Server searches within the phase link group for a match to message ID 25 in the message queue. When the message ID is located, a value of 80 is transferred to the PHASE_Q01 register of the Transfer_In phase, the Request register is cleared, and both phases are allowed to continue executing.

For more information on phase link groups, refer to the Recipe Development Manual.

Canceling Messages to Other Phases

During batch execution, a phase can request the Batch Execution Server to cancel a Send Message request. The phase that sent the message is the only phase that can cancel the message. A phase can:

- Cancel a specific message that was sent by the phase.
- Cancel all messages that were sent by the phase.

Understanding the Cancel Message Process

When the Batch Execution Server receives a Cancel Message request, the Batch Execution Server:

1. Removes the message from the message queue.
2. Clears the request from the phase logic.

Syntax: Cancel Message Request

The following table lists each type of Cancel Message request and the request variable values that must be specified.

Cancel Message Requests		
To Cancel...	**Use This Request...**	**Where...**
A specific message that was sent to a phase.	PHASE_RQ = 53*nn*	*nn* is the message ID to cancel.
A specific message to phase with a message ID greater than 99.	PHASE_RQ = 5300 PHASE_Q01 = *nnn*	*nnn* is the message ID to cancel.
Cancel all messages.	PHASE_RQ = 5400	(No further clarification is necessary.)

Syntax: Receive Message and Wait Request

The following table lists each type of Wait Message request and the request variable values that must be specified.

Wait Message Requests		
To Wait For...	**Use This Request...**	**Where...**
A message from a phase.	PHASE_RQ = 55*nn* PHASE_Q01 = *val1* PHASE_Q02 = *val2* PHASE_Q03 = *val3* PHASE_Q04 = *val4* PHASE_Q05 = *val5*	*nn* is the incoming message ID for which to wait. *val1*-*val5* stores message values, which were sent from the sending phase.

Wait Message Requests		
To Wait For...	**Use This Request...**	**Where...**
A message from a phase with a message ID greater than 99.	PHASE_RQ = 5500 PHASE_Q01 = *nnn* PHASE_Q02 = *val1* PHASE_Q03 = *val2* PHASE_Q04 = *val3* PHASE_Q05 = *val4*	*nnn* is the incoming message ID to wait for. *val1-val4* stores message values, which were sent from the sending phase.

Aborting Requests

If you want to cancel a pending request, you can send a request to the Batch Execution Server instructing it to abort the outstanding request.

Syntax: Aborting Requests

The syntax to abort a request is PHASE_RQ = 6000.

Downloading Identification Parameters

During batch execution, a phase can send a request to the Batch Execution Server to download identification parameter values to the phase in the process controller. You can define download requests to:

- Download the user-defined Batch ID.
- Download the Batch Execution Serial Number.
- Download the Phase ID.
- Download the Batch Execution node name.
- Download the fully qualified phase path.

Understanding Identification Parameters

The following list describes the information that is returned when you download these identification parameters:

User-Defined Batch ID – the name assigned to the batch by the user when the batch is added to the Batch List. This name is usually in a string format.

Unless the user always uses an integer for batch identification, define the equipment phase tag as a string to download this information. You can define the phase tag's data type in the Edit Phase Parameter dialog box in the Equipment Editor.

Batch Execution Serial Number – the unique identification number assigned to the batch by Batch Execution.

Phase ID – the equipment ID assigned to the equipment phase in the Equipment Editor. This value is defined in the Edit Equipment Phase dialog box.

Batch Execution Node Name – the unique node name assigned to the Batch Execution server.

Phase Path – the path to the phase parameter. For example, for the demo project, the phase path might look something like this: "MAKE_TOOTHPASTE\BASE:1\MAKE_BASE:1\ADD_INGS:1"

Syntax: Download Identification Parameters

To download identification parameters, you need to supply the parameter index as part of the request syntax. This index number is generated by Batch Execution and is related to the parameter ID. The index numbers start at 1, with parameter index 1 assigned to the lowest numbered parameter ID. For additional information on the parameter index, refer to the Phase Instance Parameters section.

The following table lists each type of Download Identification Parameters request and the request value that must be specified.

Download Identification Parameters Requests			
To Download...	**Use This Request...**	**Where...**	**And, the Parameter Specified by the Index is of Type...**
The user-defined Batch ID.	PHASE_RQ = 71*nn*	*nn* is the parameter index.	String
The Batch Execution Serial Number.	PHASE_RQ = 72*nn*	*nn* is the parameter index.	Integer
The Phase ID.	PHASE_RQ = 73*nn*	*nn* is the parameter index.	Integer

Download Identification Parameters Requests			
To Download...	**Use This Request...**	**Where...**	**And, the Parameter Specified by the Index is of Type...**
The Batch Execution node name.	PHASE_RQ = 74*nn*	*nn* is the parameter index.	String
The fully qualified phase path.	PHASE_RQ = 75*nn*	*nn* is the parameter index.	String

NOTES:

- *If the parameter specified by the index is a String data type, use caution if the corresponding controller or control system (such as iFIX) assumes a certain data type or string length limit. An issue could occur if the PLC has not allocated enough memory for the String that is about to be downloaded. For example, for 7500 request (phase path), depending on the recipe/phase names, the phase path could get VERY LARGE. For the demo project, for instance, the phase path might look something like this: "MAKE_TOOTHPASTE\BASE:1\MAKE_BASE:1\ADD_INGS:1" This path is 45 characters. The PLC programmer should not assume only 45 characters.*

- *If the parameter specified by the index is an Integer data type, use caution with the limits defined for your parameters in the Batch Equipment Editor. For example, if you want to download the batch serial number of a batch to the location specified by the second parameter, the phase would issue a 7202 request. Since the batch serial number can be quite large, the second parameter's high limit (which defaults to 100) might not be set sufficiently high enough to handle the batch serial number. To change the high limit, in the Batch Equipment Editor, select the phase and open the Equipment Phase Class dialog box. Click the Parameter tab, select the parameter, and click Edit to display the Edit Phase Parameter dialog box. From this dialog box, you can change the parameter's High limit to something more applicable (such as 65535).*

Quick References

This section provides several tables that allow you to quickly get information on the following:

- Batch Execution Requests
- Batch Execution Memory Variables
- iFIX Database Tags

Batch Execution Requests

This section 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.

Requests Quick-Reference	
Use this request...	**To...**
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 Batch ID.
PHASE_RQ = 7200	Download the Batch Execution serial number.
PHASE_RQ = 7300	Download the phase ID.

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

Batch Execution Memory Variables

The following table lists the common memory variables used to program your phase logic. These variables are used in the sample ladder logic available in the PLI Development Manual. The recommended naming convention for the variables begins with a unique phase name followed by an underscore and a two or three character extension to indicate the variable type.

For more information on the purpose of each variable refer to the PLI Development Manual.

Address Descriptions	
Tag Name	**Description**
PHASE_F	Phase Failure
PHASE_RQ	Request. Stores the request value from the phase logic to Batch Execution.
PHASE_SI	Step Index
PHASE_ST	State
PHASE_UN	Unit Number
PHASE_VC	Batch Execution Command
PHASE_W	Phase Ownership Flag
PHASE_PD	Phase Paused Flag
PHASE_P	Phase Pausing Flag

Address Descriptions	
Tag Name	**Description**
PHASE_SS	Phase Single Step Flag
PHASE_AD	Phase Aborted
PHASE_AG	Phase Aborting
PHASE_Q*xx*	Phase Qualifiers
PHASE_P*xx*	Floating Point Phase Parameters
PHASE_P*xx*	Integer Phase Parameters
PHASE_R*xx*	Floating Point Report Parameters
PHASE_R*xx*	Integer Phase Parameters

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.

Recommended iFIX Tag Types for Equipment Phase Tags		
Equipment Module Tag	**Equivalent PLI Register**	**Recommended iFIX Tag Type**
COMMAND	Command Register (PHASE_VC)	Analog Input (AI)*
FAILURE	Failure Register (PHASE_F)	Analog Input (AI)*
OWNER	Owner Register (PHASE_W)	Digital Input (DI)
PARMTR0*n*	Parameter Value Register (PHASEP0*n*)	Analog Input (AI)*
REQUEST	Request Register (PHASE_RQ)	Analog Input (AI)*

Recommended iFIX Tag Types for Equipment Phase Tags		
Equipment Module Tag	**Equivalent PLI Register**	**Recommended iFIX Tag Type**
REQUST0*n*	Request Data Register (PHASEQ0*n*)	Analog Input (AI)*
REPORT0*n*	Report Value Register (PHASER0*n*)	Analog Input (AI)*
PAUSE	Pause Register (PHASE_P)	Digital Input (DI)
PAUSED	Paused Register (PHASE_PD)	Digital Input (DI)
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

Index

D

E

H

I

M

O

P

R

S