Batch Tutorial

D800012X062

© 2006 Fisher-Rosemount Systems, Inc. All rights reserved.

Printed in UK

DeltaV, the DeltaV design, and PlantWeb are marks of one of the Emerson Process Management group of companies. All other marks are property of their respective owners. The contents of this publication are presented for informational purposes only, and while every effort has been made to ensure their accuracy, they are not to be construed as warrantees or guarantees, expressed or implied, regarding the products or services described herein or their use or applicability. All sales are governed by our terms and conditions, which are available on request. We reserve the right to modify or improve the design or specification of such products at any time without notice.

Contents

The DeltaV Batch Subsystem

Welcome to the DeltaV Batch subsystem, the easy-to-use, powerful, and totally integrated batch solution from Emerson Process Management.

In this tutorial, you will be introduced to the basics of using the DeltaV Batch subsystem. You will learn how to create a batch application, including defining the batch equipment, creating and configuring the control logic, and creating recipes.

The tutorial will guide you through some of the common batch configuration activities. For detailed information about the software, refer to the Batch Reference manual or look up specific topics in the DeltaV System Help.

Assumptions

It is assumed that you are a batch control engineer getting ready to configure a DeltaV Batch application. You should be familiar with control theory and have some understanding of batch control concepts, particularly the ISA S88.01 standard. You should have a working knowledge of the equipment, resources, and processes of the batch plant in which a DeltaV Batch application will be running.

You should also be familiar with the basics of using DeltaV software. If you are a new DeltaV user, we recommend that you go through the tutorial *Getting Started with Your DeltaV Digital Automation System* before beginning this tutorial.

It is also assumed that you are familiar with the basics of using Microsoft Windows XP operating system. For more information, you can access the Windows online help by clicking **Start | Help and Support**.

Warning The tutorial exercises are intended for users of a new system, not one that is already controlling a process. Do not perform any tutorial procedures that involve installing or downloading configuration information into an operational system without fully considering the impact of these changes.

If you intend to do the tutorial exercises on your workstation, it must be a configured Professional*PLUS* workstation. You also need to have security access to batch activities (specifically, to configure, download, build recipes, and operate batches) as established in the DeltaV User Manager. To be able to run and troubleshoot the tutorial configuration, you will also need one of the following: a controller with Batch licenses, a workstation with Simulate licenses, or a workstation with Batch licenses. Finally, you will need to enable Batch and Advanced Unit Management functions in the System Preferences (**Start | All Programs | DeltaV | Engineering | System Preferences**).

Documentation Conventions

This tutorial uses the same conventions for using the mouse and for selecting from menus as the DeltaV System tutorial, *Getting Started With Your DeltaV Software*. Since you should now be familiar with the basics of using the DeltaV System, the instructions for many tasks in this tutorial will be abbreviated.

For example, instead of telling you to "Click **Start | All Programs | DeltaV | Engineering | DeltaV Explorer**," the instruction may be simply "Open the DeltaV Explorer." Instead of telling you to "Point to a module in the DeltaV Explorer, right-click and select Properties from the context menu," the instruction may be "Edit the module properties."

If at first it is not obvious what you should do, try pointing to the object and clicking the right mouse button to see if there is something on the context menu that looks relevant to the instruction. In addition, assume that you are to click OK on all dialog boxes unless told to do otherwise.

DeltaV Batch Concepts

DeltaV Batch supports standard batch concepts and follows the ISA S88.01, Namur NE33 Batch, and IEC 61131-3 standards. The ISA standard defines a consistent set of terminology and models to define the control requirements for batch manufacturing plants. The ISA standard and DeltaV Batch are based on procedural and physical models that emphasize good plant design and operation practices. (Refer to DeltaV Batch and S88 and subsequent topics for more information on the ISA S88.01 standard.)

The system features a single, completely integrated database that coordinates all batch configuration activities. Easy-to-use tools are provided for the following:

- Configuring physical and procedural models
- Creating and executing recipes
- Integrating batch actions with the control system
- Generating and storing production information in an object database

DeltaV Batch is a class-based system. Many batch manufacturing plants have multiple production lines made up of the same or very similar equipment. Using classes in developing the process control system greatly simplifies the configuration process. For example, instead of fully defining the control logic for each of 100 identical or nearly identical valves, the control engineer defines the logic for the control module class and then creates 100 instances of the class (that is, control modules). It is then necessary to specify only the I/O references and any configurable parameters that differ for the individual control modules. Changes made to the class are automatically propagated to the instances, except for parameter values that have been changed at the instance level. Classes can be defined for the following: phases, control modules, equipment modules, units, and process cells.

Terminology

A **batch process** is one that leads to the production of finite quantities of material by subjecting specified quantities of input materials to an ordered set of processing activities over a finite period of time using one or more pieces of equipment. The term **batch** is used for the material produced or being produced by a single execution of a batch process.

An **area** is a physical, geographical, or logical grouping of equipment in a process control system. Areas typically represent plant locations or main processing functions. Areas contain process cells. A **process cell** is a logical grouping of unit modules, equipment modules, and control modules required for production of a batch.

At the lowest level in the physical hierarchy, control in DeltaV software is based on reusable control entities called **control modules**. Control modules link algorithms, conditions, alarms, displays, and other characteristics of a particular piece of equipment, such as a motor or valve. A **control module class** defines properties for a group of similar control modules.

When an item is created using a class definition, that item is referred to as an **instance** of the class. Also, the item is said to be **instantiated** from the class.

An **equipment module** in DeltaV batch is usually a module in which several devices (control modules) are involved with control of a single parameter. For example, a Totalizer may contain three control modules to control a coarse inlet valve, a fine inlet valve, and a flow meter, all of which are involved in input of an ingredient to a reactor. An **equipment module class** defines properties for a group of similar equipment modules.

A **unit** is a major piece of equipment that performs a specific task. A unit can contain more than one physical piece of equipment (for example, a tank, its inlet valves and outlet valves, level indicators, etc.), each of which is controlled by a control module or equipment module. A **unit class** defines the properties (unit parameters) for a group of similar units. For example, for mixing tanks, the tank class might have **unit parameters** to specify the material of construction, the tank volume, and whether the tank has the capability to heat, cool, or adjust the pH of the contents.

Alias names are created in the unit class as placeholders for the actual references to control signals, variables, and so on. A **unit module** is an instance of a unit class.

A **phase** is a series of steps that cause one or more equipment- or process-oriented actions, for example, filling a tank or agitating the contents. The **phase logic** defines the states (running, holding, restarting, aborting, and stopping) of the phase and the logic associated with each state. In most batch applications, the phase logic is defined in the phase class. **Phase classes** define common properties for a group of phases.

Typically one or more phase classes are assigned to a unit module. The assigned phase classes are called **unit phases**. In unit modules, **aliases** are resolved, that is, they are assigned unit-specific parameter references.

Another type of module, a **phase logic module**, can be used to define the phase logic for a single unit. In this type of module, the logic is fully defined in the module itself rather than in the phase class. The phase logic module is then assigned to the specific unit on which it will run. In general, batch applications are best configured using unit modules rather than phase logic modules because unit modules are more flexible and efficient. Phase logic modules are not covered in this book.

A **recipe** is a set of information that uniquely identifies the parameters, procedures, and equipment required to manufacture a product. A **recipe hierarchy** is a procedural model that defines the relationship of procedures, unit procedures, and operations. The flow of an individual recipe is defined using a **procedural function chart** (PFC). A PFC is a diagram similar to a sequential function chart (SFC). It uses steps, transitions, and a termination to define the algorithm for a recipe (an operation, unit procedure, or procedure). You can categorize your recipes in DeltaV Explorer using **recipe folders**.

A **procedure** is the highest level of the recipe hierarchy. It is used to control a sequence of unit procedures to perform a function across one or more units. A **unit procedure** is used to control a sequence of operations to perform a certain function on a unit. An **operation** executes a sequence of phases to perform a certain function on a unit. There can be one or more phases within an operation that may execute sequentially or concurrently.

DeltaV Batch Configuration Applications

The DeltaV applications that are the primary tools for configuring and testing a batch system include the DeltaV Explorer, Control Studio, Recipe Studio, and Simulator. The applications used to run batches and collect batch data include Batch Executive, Batch Operator Interface, DeltaV Operate, and Batch Historian.

DeltaV Explorer

DeltaV Explorer is used to define classes (for process cells, units, equipment modules, control modules, and phases) in the Batch Library. It is also used to create areas, process cells, unit modules, equipment modules, and control modules. Phase classes can be created in the DeltaV Explorer and the logic written in Control Studio. Recipes can also be created in DeltaV Explorer and the recipe logic written in Recipe Studio.

Control Studio

In batch applications, Control Studio is used to configure the phase logic. Each phase class contains a pre-configured state transition diagram. The configuration engineer defines the logic for the five active states (running, stopping, aborting, restarting, and holding) and for failure monitoring. This is done by creating a sequential function chart (SFC) for each active state and a function block diagram for failure monitoring. Control Studio is also used create the logic for control modules (typically in the form of function block diagrams or sequential function charts) as well as the logic for equipment modules (typically in the form of command-driven or state-driven algorithms). Several algorithm types are available for defining the logic for control modules and equipment modules.

Recipe Studio

Recipe Studio is used to create and modify the recipes in the batch process, specifically, the procedures, unit procedures, and operations. This is done by creating a PFC (Procedural Function Chart) that consists of steps, transitions, and a termination. Recipe Studio provides graphical configuration tools similar to those in Control Studio.

Recipe Simulator

Recipe Simulator allows you to test recipe configurations before phase implementation. It runs through procedural logic without executing the real phases. You can model and test your equipment configuration, equipment arbitration, phase interactions, and recipes without affecting any real-time processes.

Batch Executive

Batch Executive is the application that executes recipes in DeltaV Batch applications. It manages the procedural logic, coordinates phase communications, and allocates equipment. It downloads phase parameters to the phase logic and uploads report data from the phase logic. It sends recipe and phase status information to the Batch Operator Interface. The Batch Executive also collects and records batch events in recipe journal files that may be viewed at the Batch Operator Interface and used by the Batch Historian to build its database.

Batch Operator Interface

Batch Operator Interface is the graphical interface used by the operator to perform most batch-related functions. Using the Batch Operator Interface, an operator can create and schedule batches for execution, manage batch equipment, modify batch execution online, and view event journal information—all from one interface. The operator can switch back and forth between DeltaV Operate and the Batch Operator Interface.

DeltaV Operate

Operators interact with the process control system through the DeltaV Operate application, the primary interface used to monitor and control the process equipment.

Batch Historian

The Batch Historian provides automated collection of recipe execution data from the DeltaV Batch Executive journal files and process management event data from the DeltaV Event Chronicle. Operators can view the data using the Batch History View application.

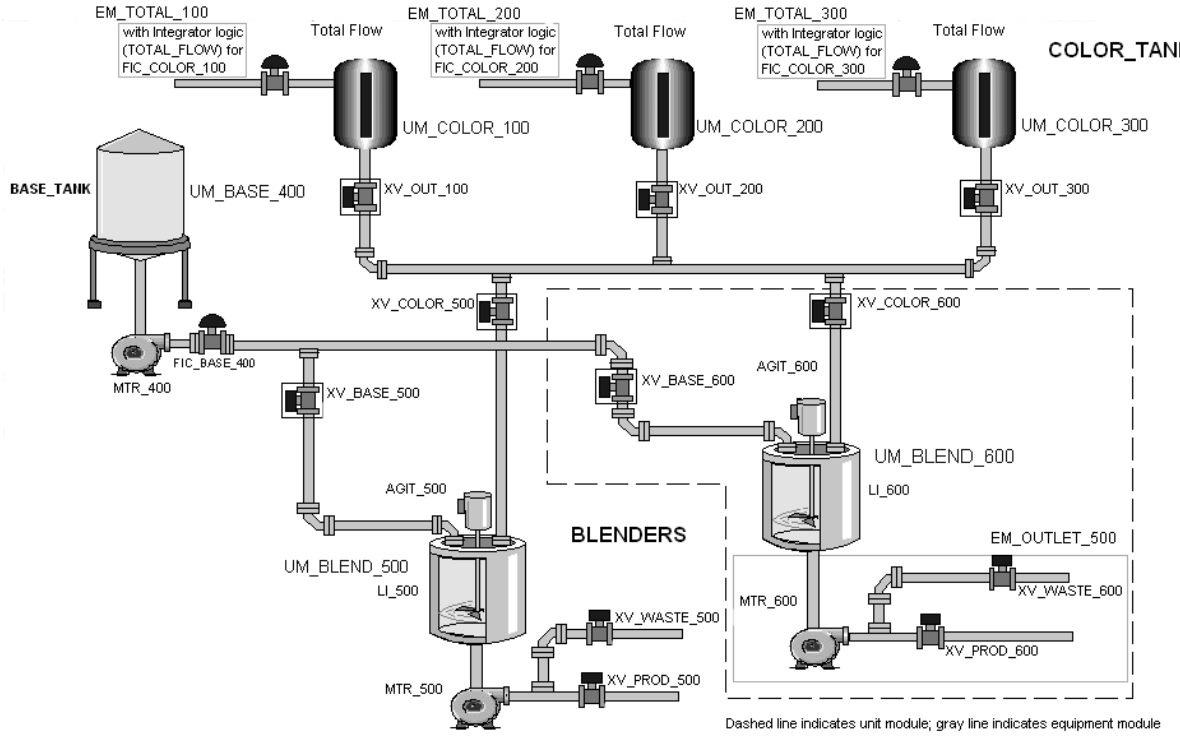
Overview of the Batch Tutorial

In this tutorial, the sample batch application is a paint blending system. Two raw materials are required: the paint base (a pre-mixed blend of liquid, binder, and additives) and the color (one or more of three colors).

The amount of each raw material required for a particular recipe is specified in gallons. After the holding tanks are charged with the required volume of the paint base and color, recipe-specified amounts of the ingredients are transferred to one of two blenders. The materials are then mixed for the period of time specified in the recipe. After blending, a good batch can be transferred to product storage and a bad batch can be transferred to a waste area.

The diagram below shows the equipment for the hypothetical system: a base tank, three color tanks, and two blenders. In the tutorial lessons that follow, we will configure the equipment and the batch control strategy for this application.

Hint You may want to print this topic since this diagram will be referenced frequently.



Tutorial Startup Database

To speed up the configuration tasks and allow you to test and troubleshoot your configuration as you develop it, a tutorial startup database (Startup.fhx) is provided in the DeltaV Samples\BatchTutorial directory. This database contains three control module classes and two phase classes. In addition, there is an .fhx file called Simulation.fhx that contains five simulation modules (such as TANKSIM and RESET_SIM) that allow you to run the paint process and troubleshoot your configuration.

The control module classes and phase classes provided in the startup database, Startup.fhx, are listed in the table below.

Class Name	Class Type	Description	Modules to be Created from the Class
TOTAL_FLOW	control module class	keeps track of the total input to the color tank	TOTAL_FLOW_100 TOTAL_FLOW_200 TOTAL_FLOW_300
FLOW_CONTROL	control module class	controls the inlet valve to the color tank; controls the outlet valve from the base tank	FIC_COLOR_100 FIC_COLOR_200 FIC_COLOR_300 FIC_BASE_400
2STATE_MTR	control module class	controls the outlet pumps for base tank and blenders; controls the agitate motors for the blenders	MTR_400 MTR_500 MTR_600 AGIT_500 AGIT_600
CHG_BASE	phase class	logic for charging the blender with the base ingredient	
DRAIN	phase class	logic for draining the blender	

In addition, the DeltaV Samples\BatchTutorial directory includes a DeltaV Operate graphic (Process.grf), as shown in the overview, that includes the major equipment and control modules for the application. It also contains a batch help graphic (Batch.grf) that allows you to manually run the phases and test the configuration before you create recipes. The BatchTutorial directory also includes another .fhx file, Complete.fhx, that contains the entire database for those who want to review the tutorial configuration without going through all the exercises.

Simulation Control Modules

In addition to control modules for the motors, valves, and so on, there are five modules (in Simulation.fhx) to simulate conditions so that you can run the batch application and manipulate the process. These modules are initially imported into a separate area, Simulation, in the System Configuration | Control Strategies part of the database. Later in the tutorial, when we are ready to run the batch application, we will move the modules from the Simulation area to the EXTERIOR_PAINT | PAINT_BLEND process cell.

The simulation modules, which are imported as part of the Startup.fhx and Complete.fhx files, are listed below.

- TANKSIM - This module simulates the level of the color tanks. It determines whether the inlet and outlet valves for the color tanks are open and keeps track of the tank levels. It also calculates the cumulative discharge from the color tanks to the blenders based on a flow of 100 GPM from the color tank outlet valves.
- LEV500_SIM - This module calculates the level in UM_BLEND_500 by integrating the flow from the base tank and the color tanks into UM_BLEND_500 with the discharge rate from the blender. The level is input to the LI_500 module, the blender level indicator.
- LEV600_SIM - This module is the same as LEV500_SIM, but for UM_BLEND_600.
- RESET_SIM - This module resets the blender tank and color tank levels to 0 so that you can quickly retest the phases without having to first drain the tanks.
- FLOWSIM - This module simulates the flow for all of the FLOW_CONTROL modules. It simply takes the value of the valve output, adds deadtime, and ties it back to be the PV of the loop.

Equipment and Module Naming Conventions

As you can see on the diagram for the process equipment shown in the Overview topic, we have followed some conventions for naming the equipment and related control modules. The unit modules for major pieces of equipment start with a descriptive phrase and are numbered from 100 through 600 (UM_COLOR_100 through UM_BLEND_600). The control modules associated with the process follow a conventional naming strategy where the first part of the tag name indicates the type of module (for example, FIC for flow control and LI for level indicator) and the latter part of the tag name is a number that relates it to the major equipment. For example, AGIT_500 is an agitate motor that will be used by the unit module UM_BLEND_500; and XV_OUT_300 is an outlet valve that will be used by unit module UM_COLOR_300.

When you are getting ready to configure a batch application for your real plant, you will want to develop a naming scheme for your equipment and control modules that will help you quickly identify process areas, equipment, and related control modules. In this tutorial, we use the following conventions for control module names:

- XV - for block valve
- FIC - for flow control
- LI - for level indicator
- MTR - for outlet pump motor
- AGIT - for agitate motor

IMPORTANT When doing the tutorial exercises online, it is important that you use the names provided in the examples. Much of the paint blending application, including the operator graphics, is already configured using those names.

Importing the Tutorial Databases

Warning This tutorial assumes you are using a DeltaV system that is offline, not one that is controlling a process. Do not perform any tutorial procedures that involve importing or downloading configuration information into an operational system without fully considering the impact of the changes.

To run and test your configuration you will need a configured Professional or ProfessionalPLUS workstation. (If your workstation is not configured, refer to the Configuring DeltaV Workstations topic in *Getting Started With Your DeltaV Digital Automation System*.) You will also need one of the following: a controller with Batch licenses, a workstation with Simulate licenses, or a workstation with Batch licenses. If you do not have a controller, you will assign modules to the workstation for simulation purposes.

You will also need to enable Batch and Advanced Unit Management functions in the System Preferences (**Start | DeltaV | Engineering | System Preferences**).

The DeltaV \Samples\BatchTutorial directory contains the following:

- **Startup.fhx** is the startup database that contains some of the control module classes and phase classes. Import this file into a new database if you want to follow along with the tutorial exercises and create the equipment, phase logic, and recipes for the paint application.
- **Simulation.fhx** is the database that contains the simulation control modules. It is to be imported after import of the Startup.fhx file. Note that, upon import, the simulation files are temporarily stored in a separate area named Simulation.
- **Complete.fhx** is the completed database with recipes. Import this file into a new database only if you want to simply view and run the modules and recipes without having to create them.

Note Before attempting to import a tutorial database, check the User Manager Application to make sure you have all keys for all areas (sitewide).

Creating a New Database for the Tutorial

Warning This tutorial assumes you are using a DeltaV system that is offline, not one that is controlling a process. Do not perform any tutorial procedures that involve importing, installing, or downloading configuration information into an operational system without fully considering the impact of the changes.

Note If you are using a workstation that has an existing database, the assumption is that you can safely create a new database and set it to be the active database. Before creating a new database, it is important that you make sure that there are no modules, areas, or recipes assigned to any workstations in the current database. Expand the workstation and select Assigned Modules. Select any assigned modules and click Delete Assignment on the context menu. Also, unassign any areas and recipes that are assigned to the Batch Executive. Finally, delete assignment of any areas to the Alarms and Events subsystem. If you do not delete the module, area, and recipe assignments, when you create a new database using the following procedure, all or portions of the current database will be copied into your new database.

To create a new database and set it to active

1 Review the preceding note for instructions on deleting any module assignments from workstations in the current database.

2 If you are using a controller, decommission the controller you want to use in the new database.

3 Create a new database using the **Create Database** tool in the Database Administrator (**Start | All Programs | DeltaV | Engineering | Database Administrator**). Name the database PaintTutorial (or TutorialComplete if you are importing the completed database) and click **Next.**

4 Select **Create New Database** and click **Next**.

5 Select **Preserve Network Definition** and then select **From the current active database**. (This maintains your current workstation configuration information.)

6 Click **Finish**.

7 Click **Yes** in response to the question about where to create the database.

8 Click **No to all** in response to the question about updating existing objects in the database.

9 After the database has been created, click **OK.**

10 Set the new database to active using the **Set Active Database** tool.

11 Click **Change** and then click **OK** in response to the warning.

12 The new database name appears in the Database name field on the Select New Active Database dialog. Click **OK**.

13 Close the Database Administrator application.

Importing an .fhx File

The following procedure describes how to import the Startup.fhx file, which contains the simulation control modules, three control module classes, and two phase classes. Importing the Complete.fhx file is similar, but involves a few additional considerations, which are detailed in additional steps after the procedure.

To import the Startup.fhx file and Simulation.fhx file

1 Start DeltaV Explorer. The new database will be opened.

2 Import the Startup.fhx file using the File | Import | Standard DeltaV Format command. Browse to the DeltaV\Samples\BatchTutorial directory and select the Startup.fhx file. (If importing Complete.fhx, refer to the additional steps after the procedure.)

> **Note** A message may appear stating that the database already contains an object with the same name and asks if you want to update that object with the one from the import file. Click **No to All** so that the latest database object names are used.
>
> **Note** Some of the imported phases and modules contain references to named sets and other modules that we will create later in the tutorial. Ignore the warnings in the import log that refer to these import errors. Also ignore any warnings about import of the workstation name not being successful.

3 Import the Simulation.fhx file.

4 If you are using a controller, commission the imported controller, CTLRnn, by clicking Decommissioned Controllers, selecting the controller you want to use, and dragging that controller from the right pane to CTLRnn. (If you have I/O cards, choose NO when asked to autosense them.)

5 Download the configuration by selecting Physical Network, clicking the right mouse button and selecting Download | Physical Network from the context menu.

6 If you check the configuration, there might be several messages that refer to some modules not having any blocks, and so on. Click Download anyway.

Additional Steps after Importing Complete.fhx

To be able to run recipes and do some of the testing exercises in the tutorial, you need to do several things after importing the Complete.fhx file and downloading the Physical Network. Because this file contains the completed configuration, you will only be able to do testing and running exercises, not the configuration exercises. Some phases, such as DRAIN and AGITATE, are modified at various points in the tutorial; only the final version of these phases is included in the Complete.fhx file.

Remember to log in as a user who is authorized to perform batch operator activities. You may need to do a View | Refresh after downloading to see all the areas and modules.

1 Assign the modules to the workstation by dragging the EXTERIOR_PAINT area to Assigned Modules under the workstation name.

2 Assign the EXTERIOR_PAINT area to the Alarms and Events subsystem under the workstation using the drag-and-drop method.

3 Enable the Batch Executive on a workstation by opening the Properties dialog for the Batch Executive and selecting Enabled. On the server tab, select Query User as the Restart behavior and UOP as the Hold propagation limit.

4 Assign the EXTERIOR_PAINT area to the Batch Executive.

5 Assign the PRC_PAINT and OP_FINISH recipes (under System Configuration | Recipes) to the Batch Executive using the drag-and-drop method.

6 Download the Physical Network.

7 Start the Batch Executive. Refer to Starting the Batch Executive for more details on how to do this.

8 Copy the process graphics, as described in the next exercise.

Note To be able to charge the blenders with base and color, you may need to check the equipment IDs for your several of your modules (XV_COLOR_500, XV_COLOR_600, COLOR_HEADER, and BASE_HEADER and then edit the logic in the CHG_COLOR, CHG_BASE, and DUMP phases to match the actual equipment IDs assigned to your modules. Refer to Equipment Arbitration and Phase Coordination for more details on how to do this.

Process Graphics

Two graphics files are also provided in the DeltaV\Samples\BatchTutorial directory. The Process graphic (Process.grf) can be opened using DeltaV Operate. This graphic is used to operate the process instrumentation by changing settings, opening and closing valves, and so on. The Batch graphic (Batch.grf) is a diagram that shows all the unit modules with links that let you open the faceplates for a unit module and phase, send batch commands, send and clear requests, and manipulate parameters, such as a requested fill amount. (This is not a typical process graphic, but is provided as a testing and troubleshooting aid.) Later, we will describe in detail how to use the Batch graphic.

After downloading the configuration to the DeltaV system, copy the graphic files to the DeltaV\DVData\Graphics-iFIX\Pic directory.

To copy the graphic files to the DeltaV graphics directory

1 Open Windows Explorer.

2 Locate the graphic files (*.grf) in the DeltaV\Samples\BatchTutorial directory.

3 Right-click and select Copy.

4 Go to the DeltaV\DVData\Graphics-iFIX\Pic, right-click and select Paste.

Overview of the Tutorial Configuration Exercises

Now that you have imported some of the control module classes and phase classes and copied the operator graphics, you are ready to begin configuring the paint blending application. This tutorial contains exercises that lead you through the following main configuration tasks:

- Creation of the additional control module classes
- Creation of the equipment module classes
- Creation of the unit classes
- Configuration of the logic for the module classes
- Configuration of the phase logic
- Creation of the unit modules
- Configuration of the procedural hierarchy--the batch recipes

General Configuration Procedure

Although it is likely that these steps will be done in a different order when configuring an actual batch application, the following is a general outline of the steps needed for configuring a DeltaV Batch system.

1 In the DeltaV Explorer hierarchy, **create the Plant Area** under System Configuration | Control Strategies. Areas let you organize process cells and modules in a way that is meaningful for your application.

2 **Create the process cell class** in the Advanced Definitions Library in DeltaV Explorer, and drag the class into the Plant Area to **create the Process Cell**. The process cell must be defined before unit modules can be assigned to it.

3 **Create the Control Module Classes** for each device and control element. **Configure the logic** in Control Studio**.**

4 **Create the Equipment Module Classes. Create the named sets** for command-driven and state-driven algorithm types. **Add the appropriate control module classes. Configure the control logic** for the equipment module class in Control Studio**.**

5 **Create Unit Classes** and, if needed, define unit parameters. **Add equipment module classes and control module classes** to the unit classes as needed.

6 **Create the Phase Classes** in the Advanced Definitions Library with batch input and report parameters as needed. These parameters are used to download values to the phase logic or report actual process values, respectively.

7 Using Control Studio, **configure the Phase Logic** for the phase classes. The phase logic is written in the form of sequential function charts that step through the actions to be performed during each phase.

8 Using DeltaV Explorer, **assign the Phase Classes** to the unit classes. Note that more than one phase class can be assigned to a single unit class. (For example, a blender unit class might have separate phases assigned for charging ingredients, agitating, heating, and discharging the contents.)

9 **Create the Unit Modules** by dragging a unit class to the appropriate process cell in the DeltaV Explorer. Unit modules are instances of a unit class; they inherit their phase logic from the assigned phase class or classes. The resulting unit module contains instances of the control module and equipment module classes contained in the unit class as well as a list of the module aliases.

10 If necessary, **create additional control modules and equipment modules** that are outside of the unit modules.

11 **Select the Unit Phases** to be included in download of the unit module. Only those phases that are assigned to the unit class and specifically marked as **controller** type will be available when the unit module is run.

12 **Resolve Alias Names** for non-class-based items by specifying the actual parameter reference paths to be used for each alias. For class-based modules, assign device tags to function blocks as needed.

13 **Specify Unit Parameter values** for any unit parameters that you defined for the unit class (UM_BLEND_500, for example, might have a volume of 1000 gallons and be made of stainless steel). Change any other configurable parameters as needed for the individual module instances.

14 **Assign unit modules (and contained control modules and equipment modules) to the controller** (or workstation for simulation**)**.

15 **Configure unit phase type** (Disabled, Controller, or External Phase).

16 **Assign the area to the Batch Executive**.

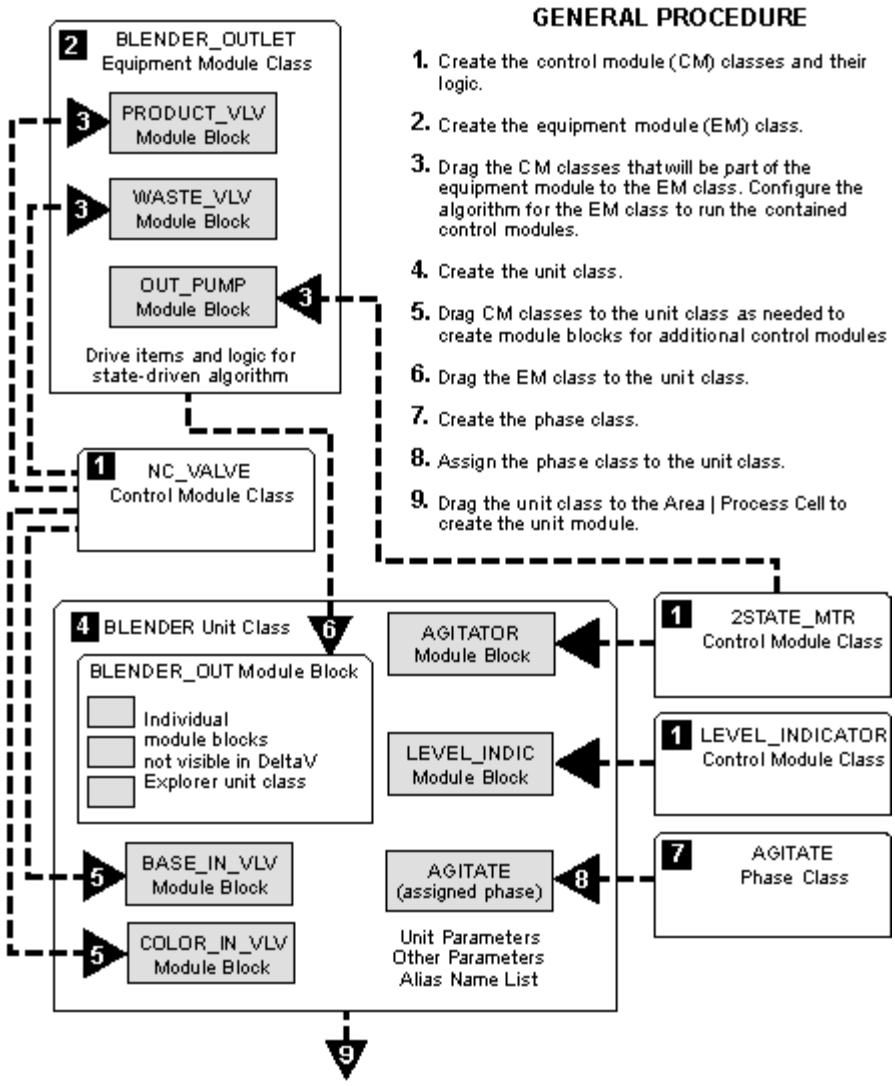
17 **Assign the area to the Alarms and Events subsystem.** (This is necessary to be able to change any parameter setpoints on the current workstation.)

18 **Configure the Batch Recipes**.

19 **Assign the recipes to the Batch Executive.**

Diagram of Module and Phase Classes

The following diagram illustrates the relationships among control module, equipment module, unit, and phase classes.



Parameter Reference Paths and Aliases

When a phase algorithm references a parameter, the general format is:

module/function block/parameter.field

For example, the following could be used to reference the current value of the output parameter of the PID function block in a flow control module, FIC_COLOR_100:

FIC_COLOR_100/PID1/OUT.CV

When the phase logic in a unit module references a module, an alias (placeholder) enclosed in number signs (#) can be used in place of:

- the module name
- the full path
- part of the path, starting from the beginning

For example, an alias such as #FLOW_CONTROL# could be set up to reference any of the following:

- FIC_COLOR_100
- FIC_COLOR_100/PID1
- FIC_COLOR_100/PID1/OUT
- FIC_COLOR_100/PID1/OUT.CV

If the control module is part of an equipment module, the control module name would be preceded by the equipment module name.

When the unit module is created, click on Aliases under the unit module, select the alias, and open its Properties dialog to assign the parameter reference path. You can create additional aliases for non-class-based modules at the unit class level using the New | Alias option on the context menu.

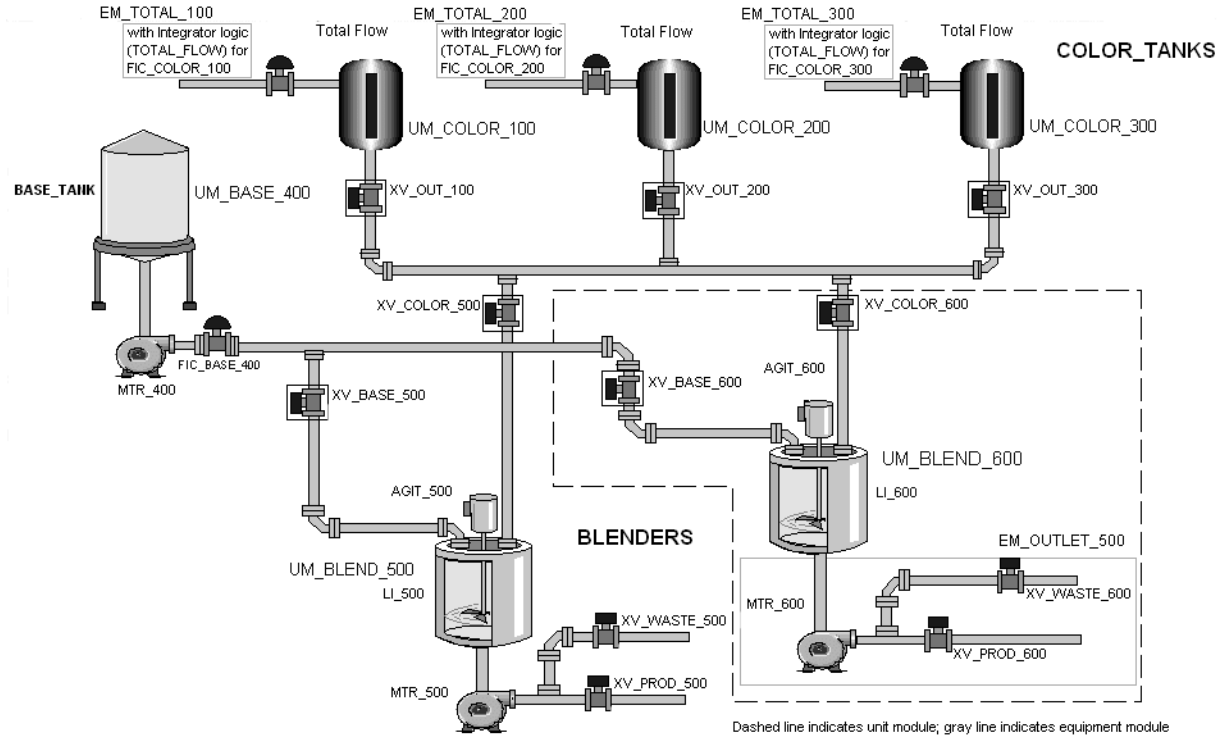
In a class-based system, aliases are automatically created for each module when the module class is added to a unit class. With class-based modules, you can browse to any parameter within the module. This will be demonstrated during phase logic configuration.

Looking Ahead

In the following topics you will set up the framework for your application in the DeltaV Explorer by defining the classes in the Advanced Definitions section of the Library and creating the plant area and process cell.

Equipment Definition

Before defining equipment configuration, you need to have a complete understanding of the physical layout of the plant equipment. The plant's P&ID drawings are a good place to start. Take a look at the layout as shown in the Process graphic.



It is clear that some elements of the process are nearly identical. At a high level, all the equipment pieces for the color tanks are the same, as are the major pieces for the blenders. At the lowest level, the valves used for output from the color tanks are similar to the valves used in other parts of the process, such as input of base and color ingredients to the blenders. Using the DeltaV system's class-based design features will greatly simplify configuration of these elements of the process system.

Class-Based Design

The DeltaV system provides class-based design of common elements that can be used in both batch and continuous projects, consistent with the standards defined for batch processing in ISA S88.01. With class-based modules, users can easily design reusable elements (such as control modules for valves and motors). The class is used to define the properties that are the same or similar for most of the items (such as the control logic, alarm parameters, and default settings). Then individual instances are created from the class. For example, a control module instance for a particular valve can be quickly created from a preconfigured control module class, inheriting all the common parameters, logic, etc., from the class. The control module instance specifies the I/O information that ties it to the physical piece of equipment it controls. If other parameters have been marked as being configurable at the instance level, changes can be made to those parameters as well. However, for the most part, the control module instance will be almost identical to the class from which it was created.

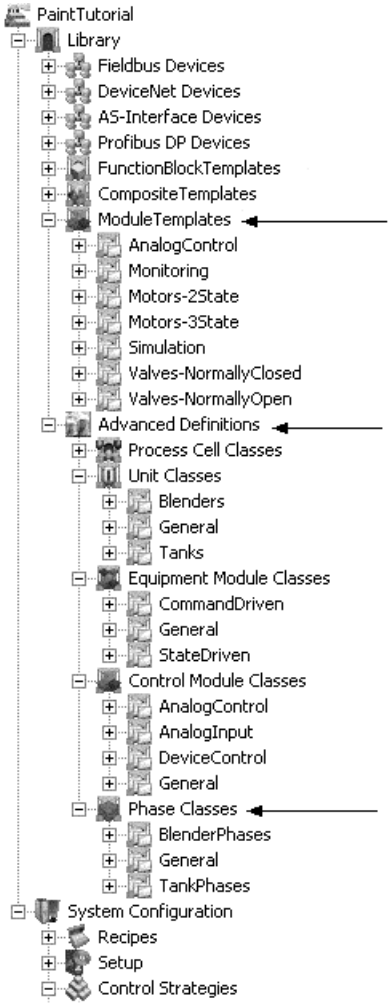
Changes made to a class (such as a control module class) after instances have been created from it are automatically propagated to the instances (the control modules). This is the main difference between creating a module from a predefined library template and creating a module from a class. If you create a module from a library template, there are no links between the module and the template; any changes made to the template have no effect on modules already created from that template.

In some cases, one or more parameters are different on a small number of the instances created from a class. The class designer can mark these parameters at the class level as being configurable at the instance level. There is a checkbox called ';Use default value from library object' on the properties dialog of each configurable parameter at the instance level. By default this checkbox is checked so that a change in the parameter value at the class level (in the library) will propagate to the instance. The user must uncheck this box to break the link to the class. The user can re-check this box to re-establish the link. Only when this box is unchecked can the user modify the value of the configurable parameter.

Key points for creating classes are that few of the instances created from a class should require changes beyond the necessary I/O assignments, and the changes should involve very few parameters. If a subset of a class of valves is different enough from the original class, but the instances in the subset are similar to each other, it may be better to create a new class for that smaller group.

The DeltaV Explorer Hierarchy

The figure below shows the Library section of the hierarchy in the DeltaV Explorer. It includes class categories in the Advanced Definitions section that we will create later in this tutorial.



The Library contains predefined module templates that can be used to create control modules directly. To do this, drag a template into an area or process cell in the System Configuration section of the DeltaV Explorer. This automatically creates a control module that can be modified to tie it to a piece of equipment and to make any other necessary changes in the parameters or logic. A template can also be used as a basis for creating a module class.

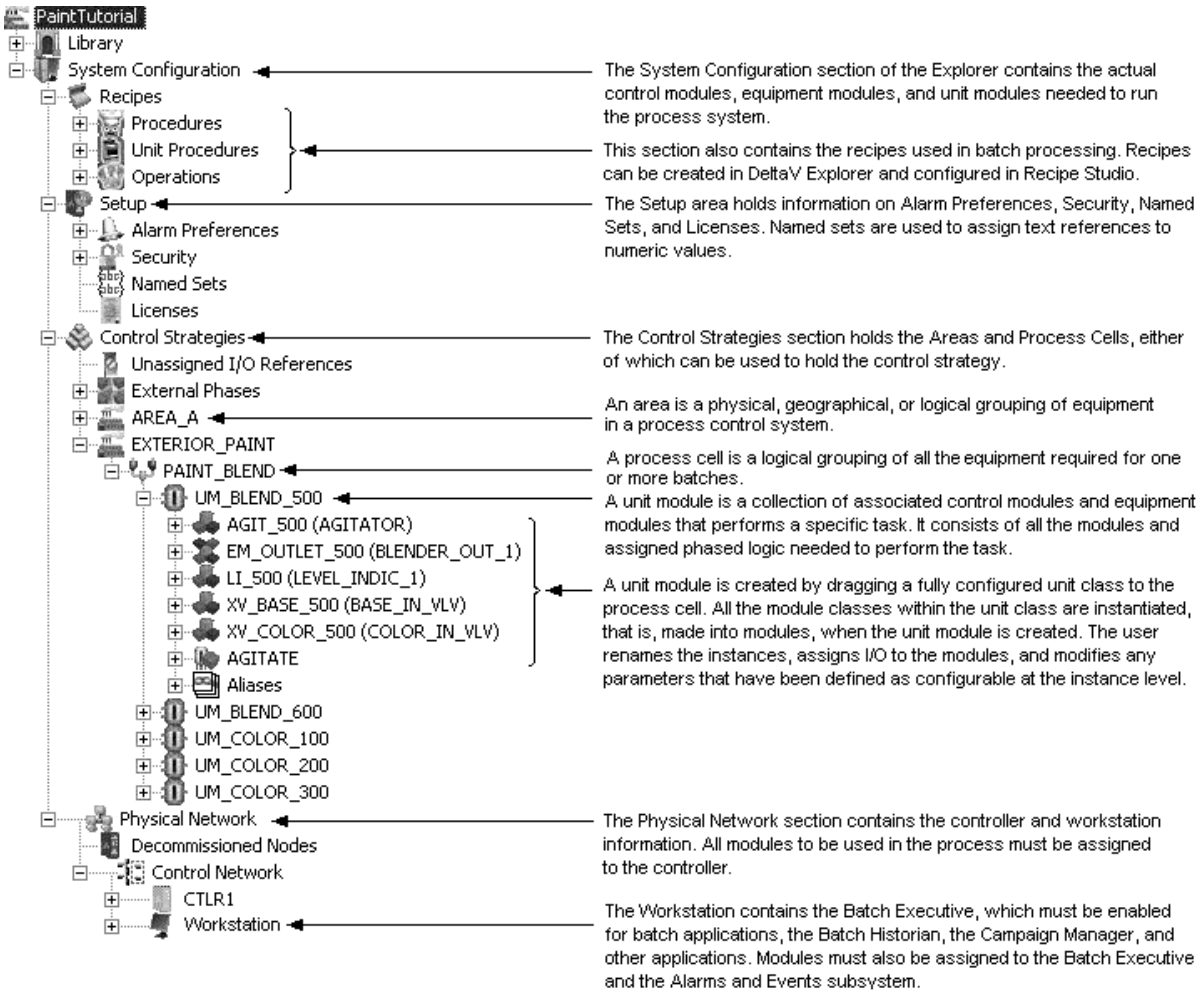
The Advanced Definitions section of the Library is where the user creates the structure for the classes to be used in a class-based control system. The Advanced Definitions section starts with the five types of classes (Process Cell, Unit, Equipment Module, Control Module, and Phase). Users create classes for each of these types as needed. Usually, for organizational purposes, the user further groups the classes into categories. For example, as shown in the hierarchy on the left, categories may be added to organize control module classes into categories which the user may call AnalogControl, AnalogInput, DeviceControl, etc.

The relationships between the types of module classes and phase classes will become clearer as you go through the tutorial. In simple terms, unit classes are at the top of the hierarchy and may contain equipment module classes and control module classes.

Phase classes, which define the logic for the process actions, are assigned to a unit class. For example, a blender may have assigned phases for filling the blender tank, agitating the contents, and discharging the product.

Once the classes are all defined in DeltaV Explorer and configured in Control Studio, creating the actual control strategy is easy. Simply drag the unit class to the Area | Process Cell in the System Configuration section of the Explorer.

The next figure shows the System Configuration section of the hierarchy in the DeltaV Explorer, as it will appear later in the tutorial when we have completed the tutorial configuration.



Defining Equipment in the DeltaV Explorer

Equipment definition in DeltaV Batch is done in the DeltaV Explorer and includes these activities:

- Creating areas in the Control Strategies section
- Creating process cell classes in the Advanced Definitions section
- Creating process cells in the Control Strategies section
- Creating control module classes, equipment module classes, and unit classes in the Advanced Definitions section
- Creating unit modules, equipment modules, and control modules in the plant areas or process cells in the Control Strategies section

Areas and Process Cells

Areas are for holding logical groupings of control strategies. They are not class-based and do not share common properties since areas by themselves do not have properties. However, areas in your batch system are used to contain process cells, which in turn contain all of the equipment for a production of a batch.

Process cells provide a way to organize equipment in a batch system. A process cell is a logical grouping of unit modules, equipment modules, and control modules that control all the equipment required for production of a batch. Process cells are class-based in that process cells within a single class share common properties. Process cell classes are defined in the Advanced Definitions section of the DeltaV Explorer Library. They can be grouped into categories for further organization. In our simple paint system, we will leave the process cell class category as General and create a single process cell class named PRODUCTION.

In the System Configuration section of the DeltaV Explorer we will create an area named EXTERIOR_PAINT that will hold a process cell named PAINT_BLEND, based on the process cell class PRODUCTION.

Creating Areas

To create an area

1 In the DeltaV Explorer, click System Configuration | Control Strategies.

2 Right-click and select New Area from the context menu. A new area, AREA1, is created.

3 Rename the area EXTERIOR_PAINT.

Creating Process Cell Classes

To create a process cell class

1 In the DeltaV Explorer Library, click Advanced Definitions | Process Cell Classes | General.

2 Right-click and select New Process Cell Class. A new class, PCELLCLS1, is created.

3 Rename the new process cell class PRODUCTION.

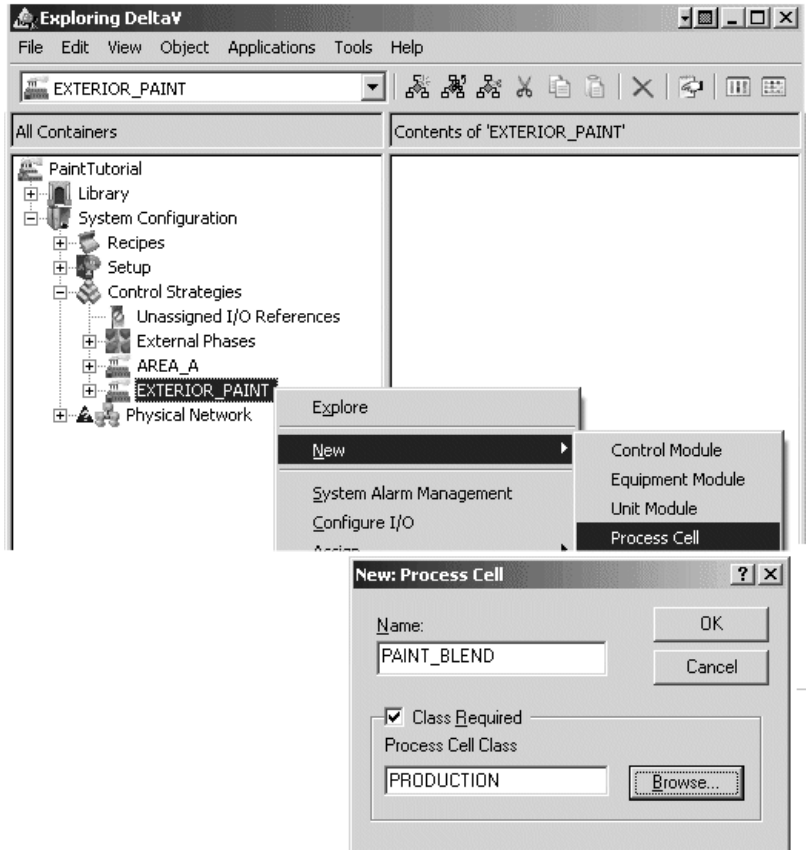
Creating Process Cells

To create a process cell

1 In the DeltaV Explorer, click System Configuration | Control Strategies | EXTERIOR_PAINT.

2 Right-click and select New | Process Cell from the context menu.

3 Name the new process cell PAINT_BLEND. Check the Class Required box and browse for the process cell class PRODUCTION. (In the Browse dialog, double-click General and then double-click PRODUCTION to select it.)



Note An easier way to create a process cell from a class is to drag the PRODUCTION process cell class to the area, EXTERIOR_PAINT, which creates a process cell named PCELL1. You can then rename PCELL1 to PAINT_BLEND. From this point on, we will use the drag-and-drop method when available.

Control Module Classes

Control modules are the basic control elements in a batch system. Each pump, valve, or control loop is controlled by a separate control module. Each module has a unique tag and all the information for a module can be accessed through its tag name. Depending on the type, a control module can be composed of parameters, history collection strategy, alarms, conditions, algorithms, and display elements. For more information on control modules, see Modules - General Information.

Control module classes define the common properties of a group of control modules. You can further group control module classes into categories to help organize the classes for large systems.

Control modules can be created individually from scratch. They can also be created from templates that exist in the DeltaV Library or they can be created as instances of control module classes. In this tutorial, we will create two control module classes (for valves and tank level indicators) and then create control modules from the classes. To speed up the tutorial configuration, other control module class categories and classes have already been imported.

Creating Control Module Class Categories

To create a control module class category

1 In the Advanced Definitions section of the DeltaV Explorer Library, expand Control Module Classes. In addition to the default category, General, it contains two categories, AnalogControl and DeviceControl, that were imported as part of the Startup.fhx file.

2 Right-click Control Module Classes and select New Category.

3 Rename the new category AnalogInput.

Creating Control Module Classes

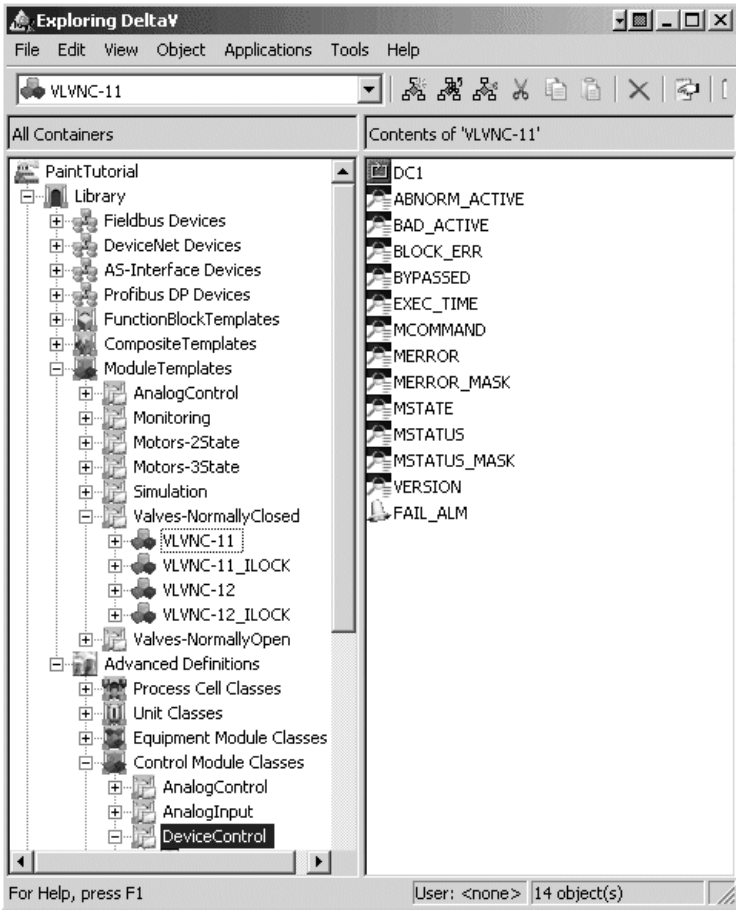
The batch system in the paint example contains a number of similar valves. Rather than create the module class from scratch, we will define a control module class for on-off valves using a preconfigured control module template in the DeltaV Module Template Library as the starting point.

To create a control module class from a library template

1 In the Module Templates section of the Library, open the Valves-NormallyClosed category.

2 Select the module template VLVNC-11 and drag it to the DeviceControl category under Control Module Classes.

3 Rename the new class (VLVNC-11_1) to NC_VALVE (for normally closed valve).



Editing the NC_VALVE Control Module Class

Use Control Studio to make changes to the NC_VALVE control module class. The changes to be made include adding a Deadtime block, an input parameter for the valve setpoint, and an output parameter for the PV. We will also change the mode properties for the DC1 block to allow Cascade as a permitted mode and to change the normal and target modes to Cascade. (The reason for making these changes to the mode properties is that the phases will write to the CAS setpoint. Later, during testing, you will have to put some of the valves in AUTO mode to change the setpoint.) The Deadtime block will be used to simulate the valve opening and closing. The valve setpoint and PV parameters will be used later when the control module is added to an equipment module.

After configuring the logic for the module, we will open the Properties dialog and select Process as the primary control display for this module. Selecting the display at the class level eliminates the need to set the display for all the individual control module instances created from the class. (In a real process system, it may well be that many process graphics are available, but in this tutorial, only one process graphic is used.)

To edit the NC_VALVE control module class

1 Open the NC_VALVE control module class in Control Studio. (In the DeltaV Explorer, right-click the class and select Open with Control Studio.) It contains a single DC function block.

2 Click on the DC1 function block.

3 In the lower left window pane, double-click the MODE parameter.

4 On the MODE Properties dialog, select both Auto and Cascade as permitted modes.

5 Set both the Normal mode and Target modes to Cascade. (The phase logic will be writing to the CAS setpoint, and the block must be in CAS mode to respond to CAS setpoint changes.)

6 Add an input parameter from the Special Items palette, dragging it to the left of the DC block, as shown in the figure below. On the Parameter Properties dialog, name the parameter VALVE_SP, keep the parameter type as Floating point, and check the box marked Allow instance value to be configured.

7 Draw a connection line between VALVE_SP and the CAS_IN_D parameter on the DC block.

8 Show PV_D as an output parameter on the DC block. (To do this, right-click on the DC block, select Show Parameter, browse for the PV_D parameter, and select Output as the type of parameter.)

9 Add an output parameter from the Special Items palette, dragging it to the right of the DC block. On the Parameter Properties dialog, name the parameter PV and keep the parameter type of Floating point.

10 Draw a connection line between PV_D on the DC block and the PV parameter.

11 Add a Deadtime (DT) block from the Analog Control palette, dragging it to the right of the DC block.

12 Draw a connection line from the OUT_D parameter on the DC block to the IN parameter on the DT block.

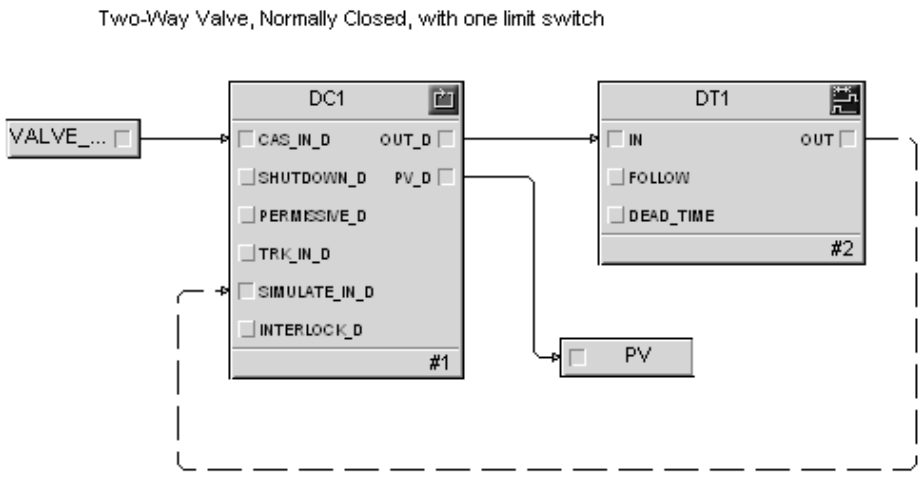
13 Draw a connection line from the OUT parameter on the DT block to the SIMULATE_IN_D parameter on the DC block. (The reason for looping back to the SIMULATE_IN_D parameter is that we will be using modules derived from this class during some simulation exercises.)

14 Open the Properties dialog for the module class by clicking File | Properties or by clicking the Properties button on the button bar.

15 On the Displays tab, browse for and select Process as the primary control display.

16 Save NC_VALVE.

The completed function block diagram looks like the following.



Note Control Studio automatically depicts a feedback wire, such as the line between OUT on the DT block and SIMULATE_IN_D on the DC block, as a dashed line after you save a module or manually set its execution order. To manually set or remove a feedback wire, select the line, right-click and choose Wire | Set as Feedback or Remove as Feedback. Whether the line is dashed or solid has no effect on the logic; it is simply a visual clue.

Making a Parameter Configurable

Sometimes not all instances of a control module class are identical or there are particular parameters that should be handled differently on different modules. To allow for these minor differences, you can define specific parameters as being configurable at the instance level. (Not all parameters have properties that can be modified.) Then, when you create the instances, you can change the configurable parameter values as needed. To do this, you also need to remove the checkmark from the Properties dialog checkbox for the instance parameter, **Use default value from library object**. Removing the checkmark breaks the link with the class. Note that if you later make changes to configurable parameters at the class level, the changes are propagated to all instances except for those that have had the link with the class specifically broken. You can reestablish the link with the class by checking the box (Use default value from library object) again.

Note If you are planning to use the same Primary Control Display for all or most of the control modules created from a class, you can specify the display using the Properties dialog at the class level. This saves the time of specifying the display for each control module. Assigning the faceplate and detailed display at the class level could also save time. This assignment should be done before the control module instances are created since the display names are not propagated to the instances after the control modules are created.

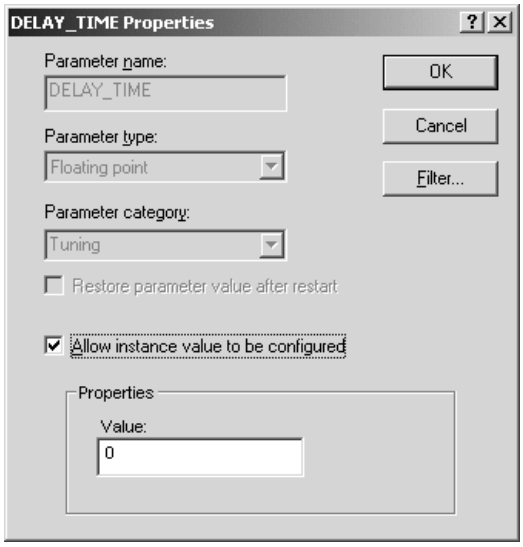
The paint application has a number of normally closed on-off valves with one input and one output parameter. For the purpose of illustration, we will make one of the tuning parameters on the DC block, DELAY_TIME, configurable at the instance level. The DELAY_TIME value for opening the valve is normally 0, but in some instances, it may be desirable to be able to specify a delay before opening the valve.

To specify that a parameter is configurable

1 In Control Studio, select the DC block on the NC_VALVE module class. The parameters for this block are listed in the lower left pane. (If necessary, check the various filter boxes so that all the parameters are visible in the list.)

2 Click the Categorized tab. In the Tuning category, select DELAY_TIME as the parameter you wish to make configurable at the instance level and open the Properties dialog.

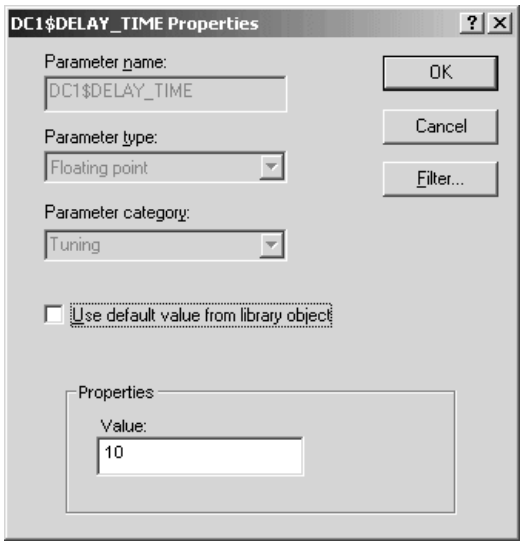
3 Check the box Allow instance value to be configured.
The default value for DELAY_TIME for all instances created from this class is 0, but the value can now be changed for particular instances.



Note that the Properties dialog can also be opened in DeltaV Explorer to make this kind of change.

4 Save the class and close Control Studio.

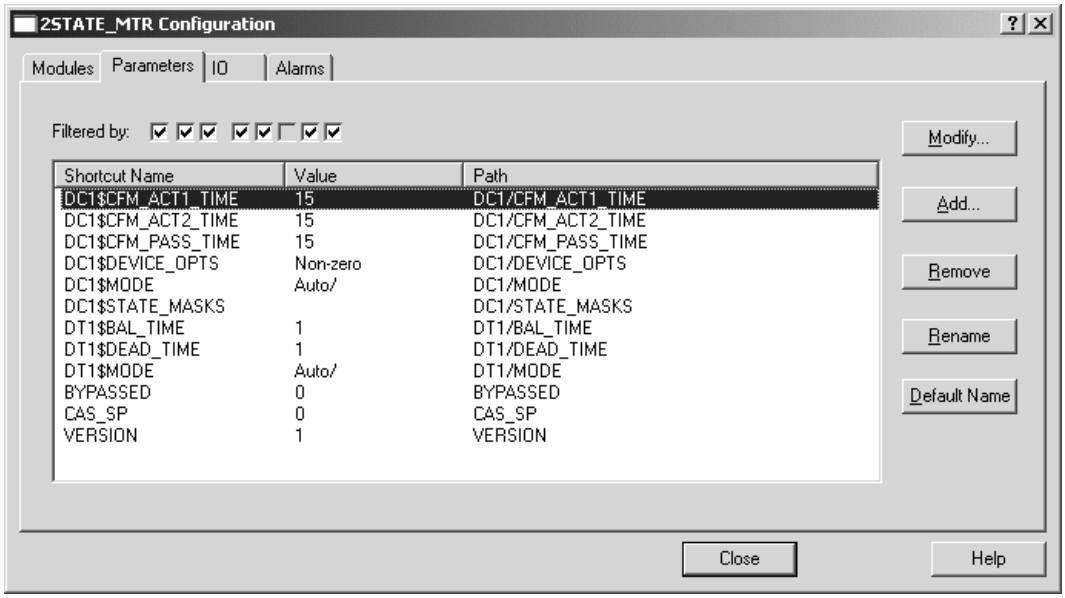
Later, when instances of the valve have been created in the System Configuration section of the DeltaV Explorer, we can select the DC block on any of those valve control modules, open the Properties dialog, delete the check mark from the box Use default value from library object, and then change the parameter value for that instance. Thus, if we wanted to add a delay of 10 seconds to the opening of a particular valve, the Properties dialog for the DELAY_TIME parameter on its DC block would look like the following:



The Configure Dialog in DeltaV Explorer

The Configure dialog in DeltaV Explorer provides a single location for all configurable items related to module classes and class-based modules. The Configure dialog is accessed by right-clicking the module class and selecting Configure from the context menu.

When you are designing a module class, the configure dialog provides a list of all parameters that are configurable by default. You can also add parameters to (or remove parameters from) the list of configurable parameters, change default values at the class level, and rename the parameters for viewing in the System Configuration section of the DeltaV Explorer.



The Configure dialog is also available for editing module instances in the System Configuration section of the DeltaV Explorer. It can be used to change the values of configurable parameters at the instance level, assign I/O device signal tags, and modify alarm attributes.

Modifying the Module Class to Enable Simulation

In this exercise we will show you how to use the Configure dialog in the DeltaV Explorer to add a parameter to the list of configurable parameters and modify its properties. (You can also make this type of change in Control Studio.) To be able to use the simulation capabilities we need to enable the SIMULATE_D parameter on the DC block of the NC_VALVE module class. Therefore, we will add SIMULATE_D to the list of configurable parameters and then enable it.

To enable simulation on the NC_VALVE module class

1 Open DeltaV Explorer.

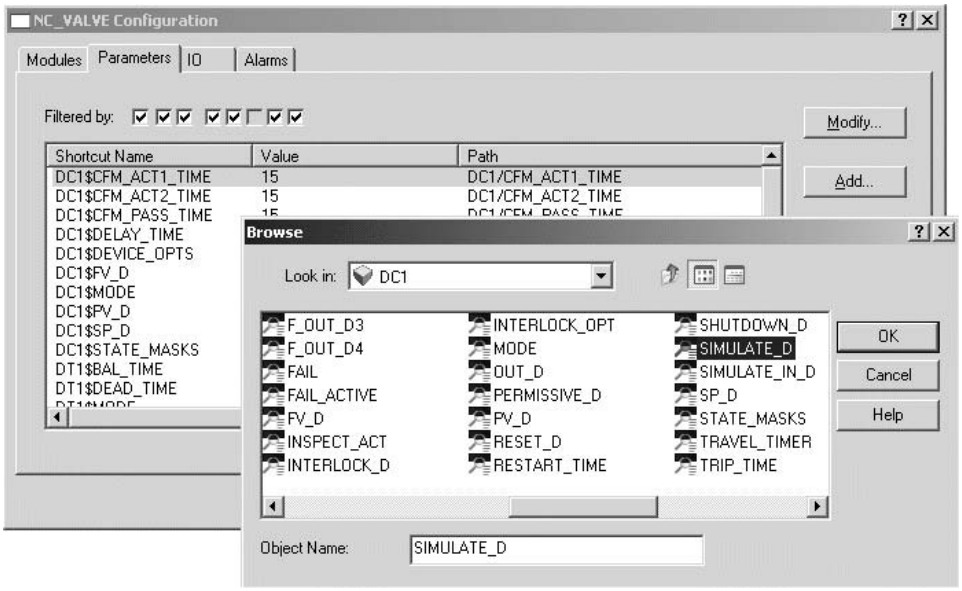
2 Navigate to the NC_VALVE control module class under the DeviceControl category in the Advanced Definitions section.

3 Right-click NC_VALVE and select Configure from the context menu.

4 Select the Parameters tab.

5 Click Add.

6 Browse to DC1/SIMULATE_D and click OK.



7 Select the DC1$SIMULATE_D parameter and click Modify open the Properties dialog (or double-click the parameter name).

8 Select Enabled under Simulate Enabled/Disabled.

9 Click OK on the Properties dialog and close the Configure dialog.

Creating a Control Module Class for Level Indicators

The second control module class we will create is for level indicators that will be used in the blenders. This class will be created in the category named AnalogInput. We will create it from a Library template and then modify several parameters. For example, one parameter we will need to change is the L_TYPE parameter on the AI block. The named state will be changed to Direct (from Indirect) because values will come directly from the simulation.

To create the control module class

1 Select AnalogInput, right-click, and select New Control Module Class.

2 On the New dialog, name the class LEVEL_INDICATOR.

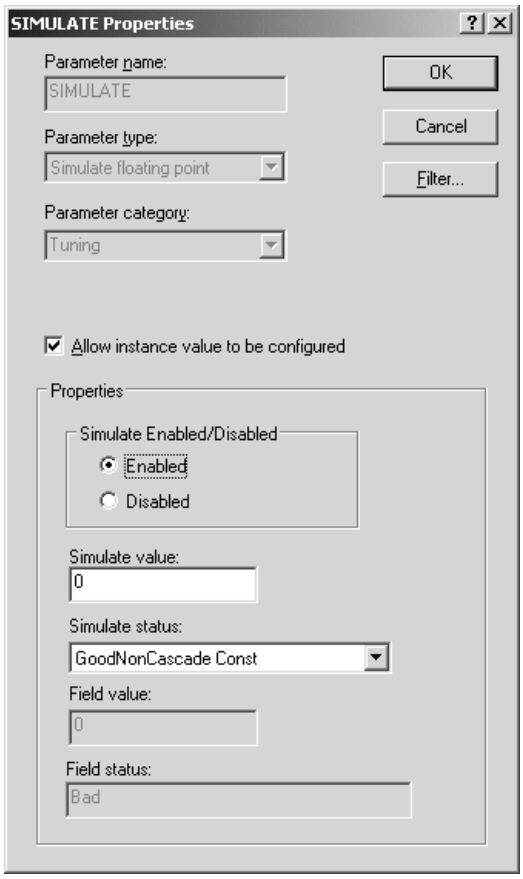
3 Click Start From Existing, browse the module templates to Monitoring | Analog, and click OK to create the class.

To modify the LEVEL_INDICATOR class

1 Open LEVEL_INDICATOR in Control Studio and select the AI block.

2 Select the tuning parameter SIMULATE and open the Properties dialog.

3 Select Enabled, check the box Allow instance value to be configured, and click OK



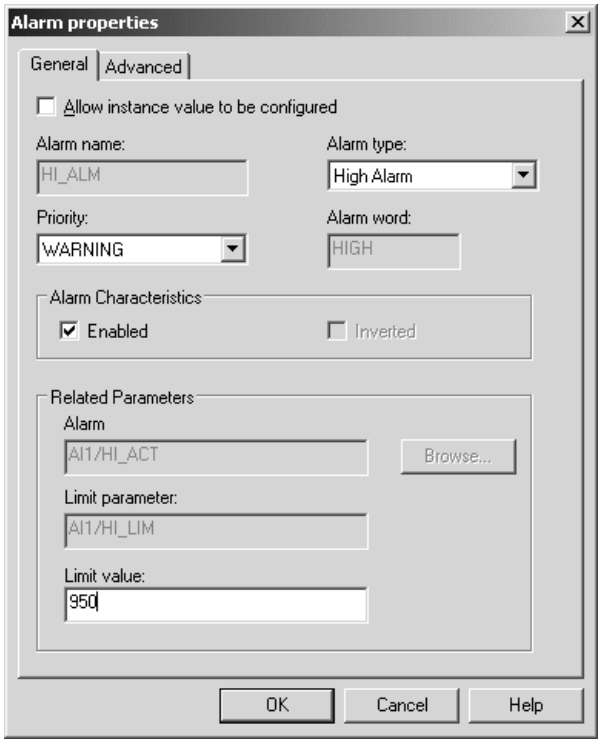
4 Select the I/O parameter OUT_SCALE and open the Properties dialog.

5 Change the 100% scale to 1000 and the Engineering unit to gallons (gal), and check the box Allow instance value to be configured.

6 Select the L_TYPE tuning parameter and change the named state to Direct.

7 Select the HI_ALM parameter in the Alarm Parameters window and open the Properties dialog.

8 Change the Limit value to 950 and click OK.



Open the Properties dialog for the module class (File | Properties) and, on the Displays tab, browse for and select Process as the primary control display.

9 Save the module and close Control Studio.

Equipment Module Classes

Equipment modules provide supervising control for a collection of control modules. There are seven types of algorithms that can be used to define the logic for an equipment module class. The most commonly used are the state-driven algorithm and the command-driven algorithm.

State-driven algorithms are typically used when a single step or value is all that is needed to manipulate the control modules. It is best used for simple equipment manipulation that does not involve complex timing relationships (such as changing the setpoints on a group of valves and motors).

Command-driven algorithms are used when multiple steps or commands are needed to supervise the control modules and there may be timing relationships to be considered (for example, one step must be completed before another can be started). With command-driven algorithms, the logic is written in the form of sequential function charts (SFCs), and the user has full control over how the logic is carried out. A command-driven algorithm selects one of a number of command sub-blocks to run, based on the value written to a command parameter (A_COMMAND). The command parameter is of the type named set, so the values that it can take are defined by a named set. The named set should have states for each of the possible commands, plus a mandatory entry with value 255, which represents an undefined/idle condition.

Several control module classes were imported as part of the Startup.fhx file. Some of these will be incorporated into the following two equipment module classes, which will be created in this chapter:

- TOTALIZER, a command-driven equipment module that will control the amount of input into the color tanks
- BLENDER_OUTLET, a state-driven equipment module that will control output from the blenders

The TOTALIZER equipment module class will be used to create the EM_TOTAL_100 (and 200 and 300) equipment modules that will be part of the unit modules named UM_COLOR_100 (and 200 and 300). The BLENDER_OUTLET equipment module class will be used to create the EM_OUTLET_500 (and 600) equipment modules that will be part of the UM_BLEND_500 (and 600) unit modules. To refresh your memory, look again at the Process picture in the Overview of the Batch Tutorial.

Using the Expression Editor

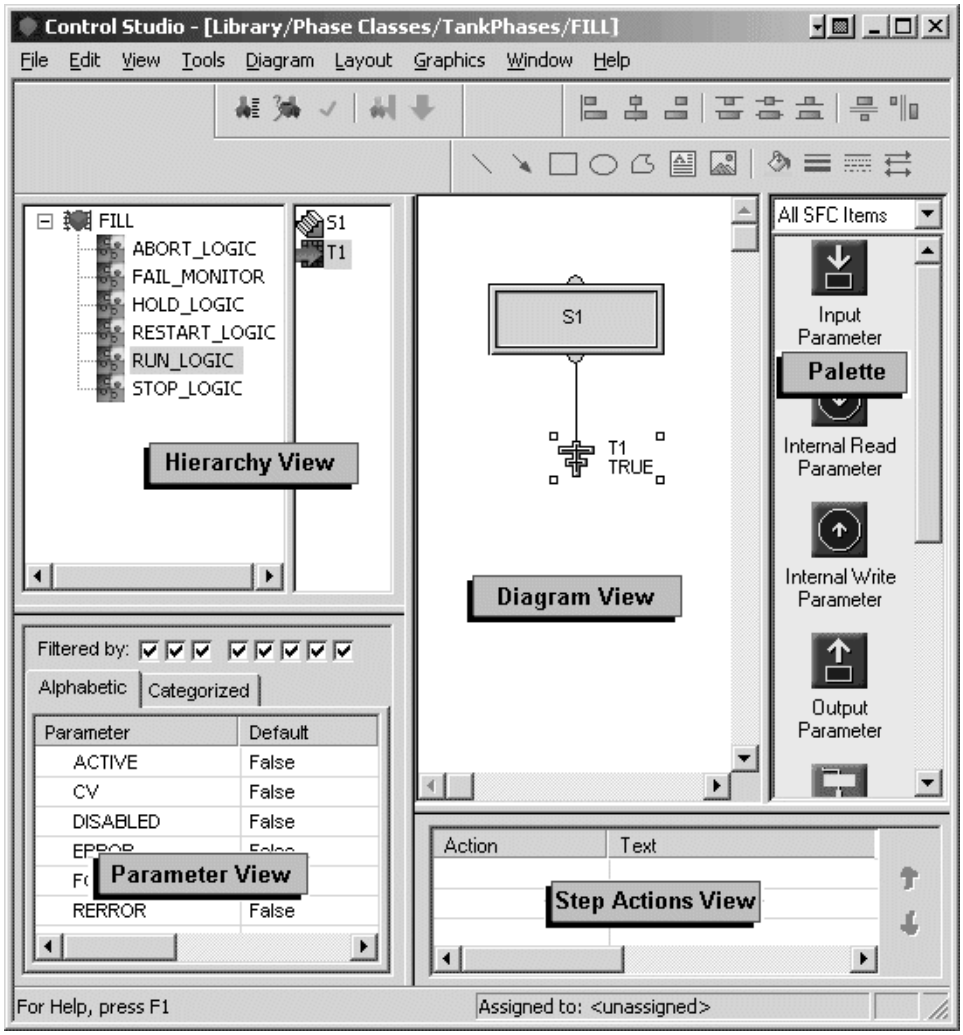
DeltaV software provides an Expression Editor to help you configure the expressions for action statements and transitions that you will use in Sequential Function Charts that make up the logic for command-driven algorithms. The Expression Editor can be accessed from the Properties dialog for either a step action or a transition by clicking the Expression Assistant button. The characters that are used in expressions are summarized in the table below..

Characters	Use	Example
/	Precedes a reference to an internal parameter (that is, one within the current phase class or module). Use the Insert Internal Parameter button to browse for these parameters.	'WAIT/ACTIVE.CV' (WAIT is the name of a step in the SFC where the action is created.)
^/	Precedes a reference to an internal parameter up one block level. Use the Insert Internal Parameter button to browse for these parameters.	'^/REQUEST.CV'
//	Precedes a reference to an external parameter (that is, a parameter within another phase class or module). Use the Insert External Parameter button to browse for these parameters.	'//XV_102/DC1/SP_D'
# #	Used to enclose an alias. Use the Insert Alias button to browse for aliases. References to aliases are external because aliases are defined on unit classes, and therefore need to begin with //. Note: When you want to reference an Alias IGN (Ignore) field from within an expression, you need to use the syntax / aliasname.IGN (with only one slash and no # signs).	'//#BLEND_LEV_CV#'
:	Used to separate a named set from the named set value.	'vlvnc-sp:OPEN'

Characters	Use	Example
:=	Used to assign values. Step actions use this operator. The value of the right operand is assigned to the left operand.	'^/REQUEST.CV' := 3202
=	Used to compare values. Similar operators include >, <, >=, <=, != (not equal to), <> (not equal to). Transitions use these operators.	'^/REQUEST.CV' = 0
+	Used for addition of numeric values or for concatenation of strings.	'LEV.CV' >= 'INIT_LEV.CV' + '^/FILL_AMT.CV'
' '	Single quotes are used to enclose parameters.	
" "	Double quotes are used to enclose strings.	
;	Used to end action statements.	

Refer to the Syntax Rules topic for more information about expressions and operators.

The following figure shows the window panes in Control Studio that show when developing a Sequential Function Chart.



Creating a Command-Driven Equipment Module Class (TOTALIZER)

The TOTALIZER equipment module will have supervising control over the input analog control loop (FLOW_CONTROL) and the flow integrator (TOTAL_FLOW) control modules. (The control module classes for both of these were imported as part of the Startup.fhx file.) The TOTALIZER module determines the amount of the color ingredient to be input into the color tanks. It allows for fast input (with the valve at 75% of the high limit) until 90% of the total amount is reached, and then partly closes the valve (to 25%) until the final 10% is input.

Creating an equipment module class involves these basic steps:

- Create the named set that will be used by the command sub-blocks to manipulate the control modules
- Add the control module classes that will be supervised by the equipment module class logic
- Configure the logic

The following table shows the named set (totalizer) to be used by the TOTALIZER equipment module. It has three states: Totalize_fast, Totalize_accurate, and Reset, as well as the mandatory entry of Idle.

State Name	State Value	Name of command sub-block run when the A_COMMAND parameter is set to the specified value
Totalize_fast	1	COMMAND_00001
Totalize_accurate	2	COMMAND_00002
Reset	99	COMMAND_00099
Idle	255 (Required)	COMMAND_00255

It is preferable to create the named set before creating the equipment module class. When you then create the equipment module class, it is created with the named states as composite blocks. If you do not create the named set before creating the equipment module class, you are asked if you wish to do so when you create the class. If the system creates the named set, it automatically includes the value of 255 with a default name of Undefined. If you create the named set, you need to make sure that you include 255 as one of the state values.

To create the totalizer named set

1 In the DeltaV Explorer, select Named Sets under System Configuration | Setup.

2 Right-click and select New Named Set from the context menu.

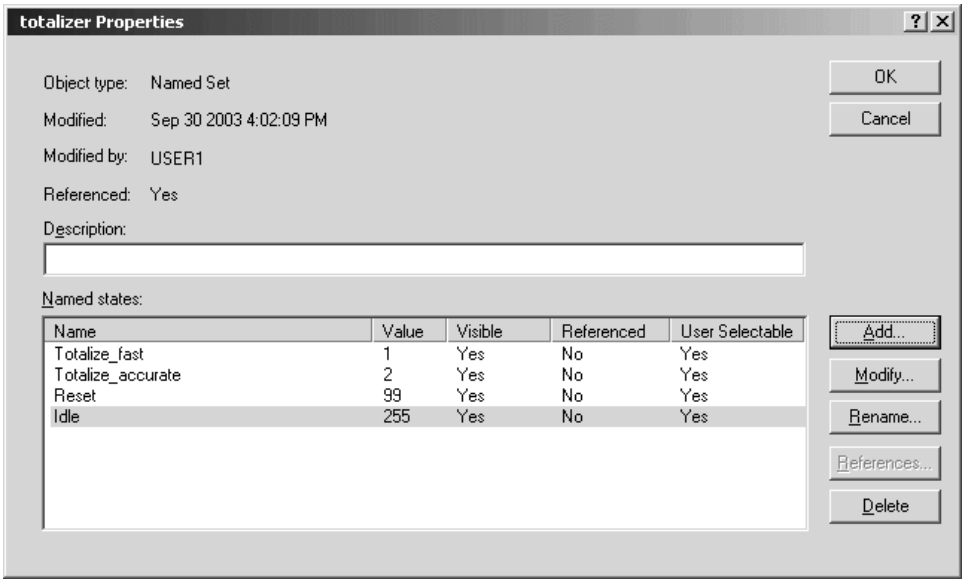
3 Rename NamedSet1 to totalizer.

4 Double-click totalizer to open the Properties dialog.

5 Click Add.

6 In the State Properties dialog, type the name of the state, Totalize_fast, and change the value to 1. Leave the Visible and User selectable check boxes selected.

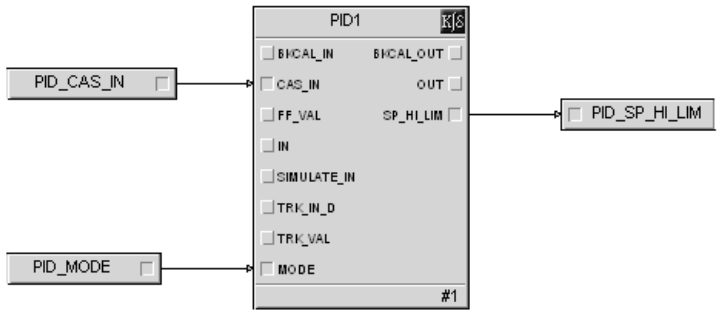
7 Add the additional states (Totalize_accurate, Reset, and Idle) and assign the state values as shown below.)



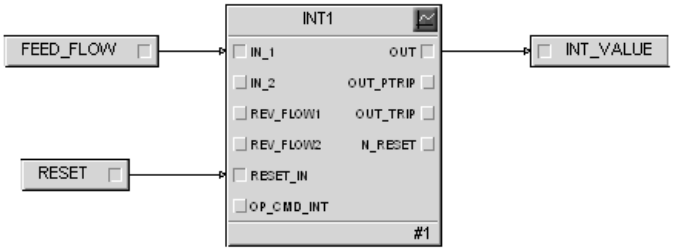
The two control module classes, FLOW_CONTROL and TOTAL_FLOW, were imported as part of the Startup.fhx file. They are both under the AnalogControl category under Advanced Definitions | Control Module Classes.

FLOW_CONTROL is a PID loop that can be adjusted to control the flow into the color tank. TOTAL_FLOW is an integrator that keeps track of the total amount of color added to the tank.

The function block diagram for the FLOW_CONTROL module class is shown below.



The function block diagram for the TOTAL_FLOW module class is shown below.



We will now create the TOTALIZER equipment module class and add the two control modules to the equipment module.

To create the class category and TOTALIZER equipment module class

1 First, create an equipment module class category named CommandDriven under Equipment Module Classes in the Advanced Definitions section of the Library.

2 Right-click CommandDriven and select New Equipment Module Class.

3 On the New dialog, enter TOTALIZER as the Object name.

4 Select Create New and select Command-driven as the algorithm type.

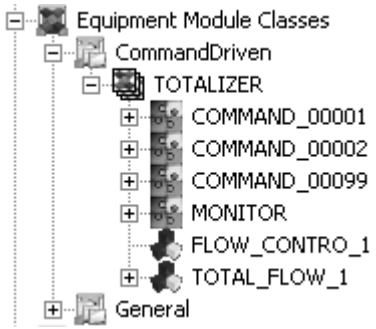
5 Browse for the named set totalizer.

6 Click OK to create the TOTALIZER equipment module class.

7 Drag the FLOW_CONTROL control module class to TOTALIZER.

8 Drag the TOTAL_FLOW control module class to TOTALIZER.

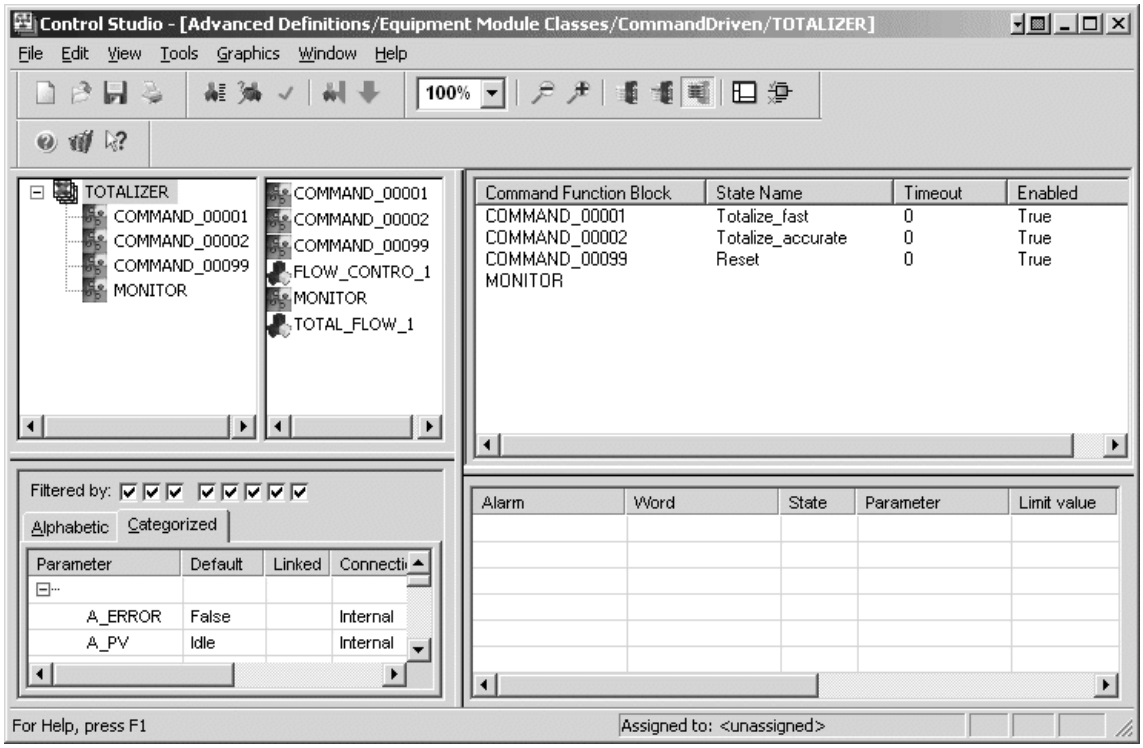
9 Click the + sign next to TOTALIZER to view the contents of the module class.



10 Open the Properties dialog for TOTALIZER. On the Displays tab, browse for and select Process as the primary control display.

Configuring the TOTALIZER Equipment Module Class

Open the TOTALIZER equipment module class in Control Studio. (From the DeltaV Explorer, right-click on TOTALIZER and select Open with Control Studio.) Note that it contains module blocks that represent the FLOW_CONTROL and TOTAL_FLOW control module classes. (The module blocks have been named with _1 at the end to indicate they are copies of the class. The number increments for each new copy.) The equipment module also contains a composite block for MONITOR, which is a default block. In addition, it contains three composite blocks beginning with COMMAND that represent the items in the named set totalizer. Each of these command blocks will be configured using sequential function chart logic.



To configure the logic for COMMAND_00001 (Totalize_fast)

1 Click COMMAND_00001 in the left pane.
The right pane now shows a sequential function chart (SFC) with one step and one transition (termination).

2 Click on the step, S1.

3 In the Action pane in the lower right corner, right-click and select Add to add an action to Step 1.

4 Open the Expression Assistant and click Insert Internal Parameter to configure the following expression to reset the TOTAL_FLOW control module to 0. (Start browsing by clicking the Up One Level button to browse within the entire TOTALIZER module class for the TOTAL_FLOW block.) Enter the assignment symbol (:=) and type in FALSE.

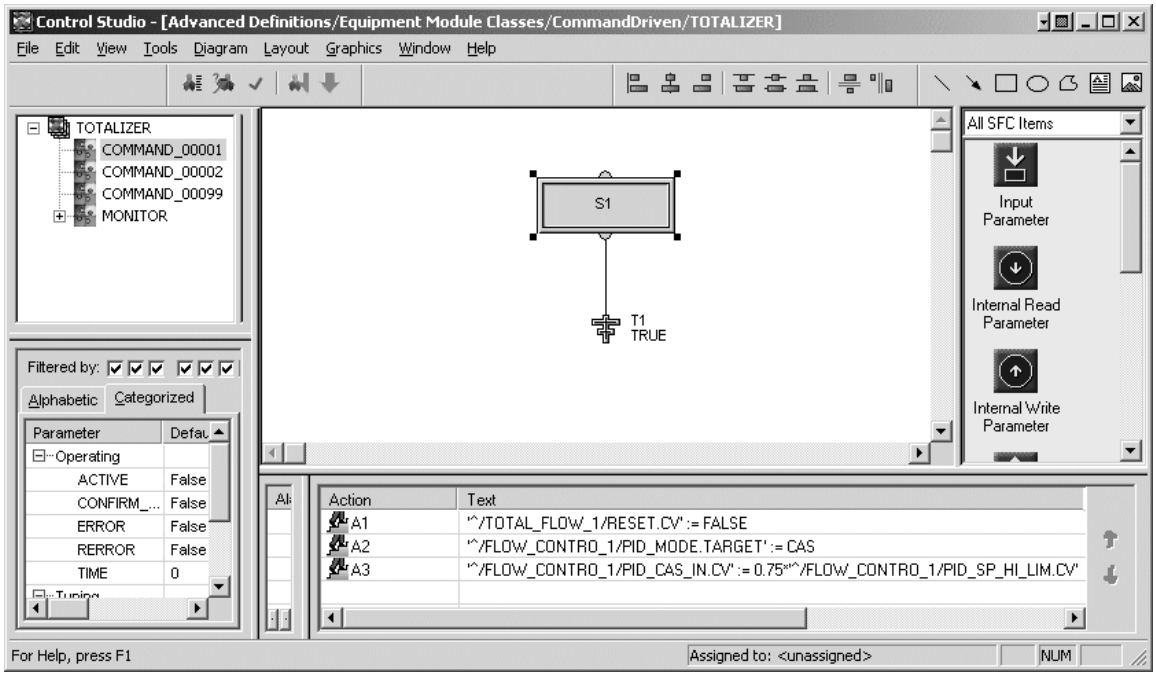
'^/TOTAL_FLOW_1/RESET.CV' := FALSE

5 Add a second action to put the module in CAS mode to be able to write to CAS_IN.

'^/FLOW_CONTRO_1/PID_MODE.TARGET' := CAS

6 Add another action to write the setpoint to the control module. The setpoint will be 75% of the SP high limit. Create the following expression for the action:

'^/FLOW_CONTRO_1/PID_CAS_IN.CV' := 0.75 * '^/FLOW_CONTRO_1/PID_SP_HI_LIM.CV'



The logic for Totalize_accurate is very similar to that for Totalize_fast. The main difference is that in the third action, the flow setpoint for FLOW_CONTRO_1 is assigned .25 so the loop setpoint is 25% of the SP high limit for more accurate filling of the last 10% of the desired input amount. You can configure the logic as in the procedure above or you can simply copy and paste the step action expressions from Command_00001 to Command_00002 and then change the 0.75 to 0.25.

To configure the logic for COMMAND_00002 (Totalize_accurate)

1 Click COMMAND_00002 in the left pane.

2 Click on the step, S1.

3 Add an action and configure the following expression:

'^/TOTAL_FLOW_1/RESET.CV' := FALSE

4 Add another action and configure the following expression:

'^/FLOW_CONTRO_1/PID_MODE.TARGET' := CAS

5 Add another action and create the following expression:

'^/FLOW_CONTRO_1/PID_CAS_IN.CV' := 0.25 * '^/FLOW_CONTRO_1/PID_SP_HI_LIM.CV'

The Reset logic is used to reset the integrator block in the TOTAL_FLOW control module and to put FLOW_CONTROL into manual mode.

To configure the logic for COMMAND_00099 (Reset)

1 Click COMMAND_00099 in the left pane.

2 Click on the step, S1.

3 Add an action and configure the following expression to close the valve:

'^/FLOW_CONTRO_1/PID_CAS_IN.CV' :=0

4 Add an action and configure the following expression to change the mode to manual:

'^/FLOW_CONTRO_1/PID_MODE.TARGET' := MAN

5 Add another action and create the following expression to reset the TOTAL_FLOW integrator block:

'^/TOTAL_FLOW_1/RESET.CV' := TRUE

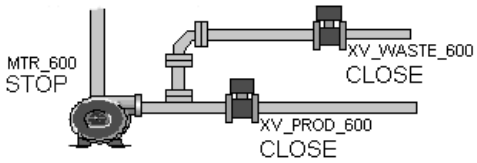
6 Save the TOTALIZER equipment module class.

Creating a State-Driven Equipment Module Class (BLENDER_OUTLET)

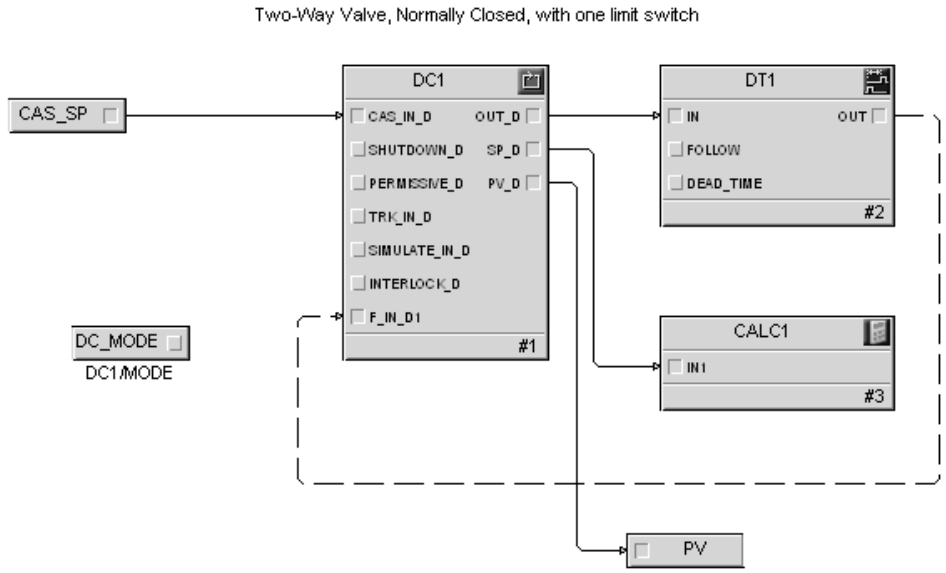
A state-driven algorithm can also be used for the Running logic of an equipment module.

A state-driven algorithm selects one of a number of states, based on the value written to an algorithm setpoint parameter. For each state of the state-driven algorithm, the user defines the parameter setpoints that are to be set for subordinate modules. For each parameter setpoint, the user defines the path to the parameter to be written and, optionally, the path to a PV that can be read back to check that the parameter setpoint has been achieved.

We will now create BLENDER_OUTLET, a state-driven equipment module that will control output from the blenders. Depending on the desired state (Closed, Draining, or Releasing_Product), the equipment module will turn the outlet pump on or off and open or close the appropriate valves. Below is a picture of the control modules for the equipment used in this part of the process system.



NC_VALVE is the control module class that will be used for the two valves. The 2STATE_MTR control module class was imported as part of the Startup.fhx file and will be used for the out pump. The 2STATE_MTR control module class was created from the MTR-11 module template and several blocks and parameters were added to it. The function block diagram for 2STATE_MTR is shown below.



The DT (deadtime) and CALC blocks were added for purposes of simulating a live process. The DC_MODE, CAS_SP, and PV parameters were added in order to read to /write from the DC block.

The following table shows how the two valves and outlet pump are manipulated depending on the selected state. For the Closed state, the valves are closed and the pump is off; for the Draining state, the outlet pump is on and the waste valve is open; for the Releasing_Product state, the outlet pump is on and the product valve is open.

Control Module Class	State		
	Closed	**Draining**	**Releasing Product**
OUT_PUMP (motor)	0	1	1
PRODUCT_VLV (valve)	0	0	1
WASTE_VLV (valve)	0	1	0

Before creating the equipment module class, we will create the named set (outlet_state) that will be referenced by it.

To create the outlet_state named set

1 In the DeltaV Explorer, select Named Sets under System Configuration | Setup.

2 Right-click and select New Named Set from the context menu.

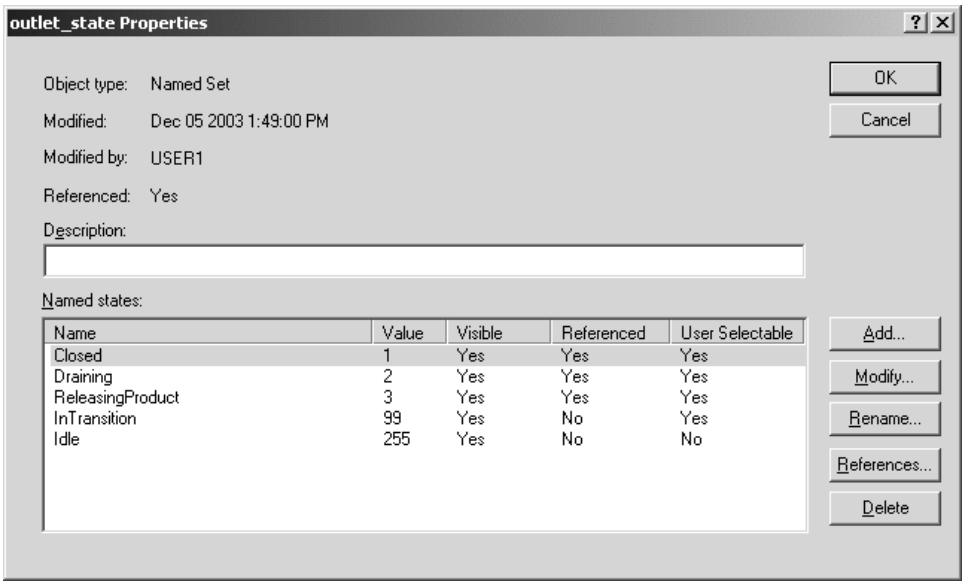
3 Rename NamedSet1 to outlet_state.

4 Double-click outlet_state to open the Properties dialog.

5 Click Add.

6 In the State Properties dialog, type the name of the state, Closed, and change the value to 1. Leave the Visible and User selectable check boxes selected.

7 Add the additional states of Draining (value=2), ReleasingProduct (value=3), InTransition (value=99) and Idle (value=255). When creating the Idle state, remove the checkmark next to User selectable.



Earlier we configured a control module called NC_VALVE. (We started from the VLVNC-11 control module template and added a valve setpoint, a deadtime block, and a PV output parameter.) Now we will add two copies of this control module class to the equipment module class to represent the two outlet valves. The resulting module blocks will be named PRODUCT_VLV and WASTE_VLV. We will also add the 2STATE_MTR control module class to be used as the OUT_PUMP. The 2STATE_MTR control module class was imported as part of the Startup.fhx file.

To create the BLENDER_OUTLET equipment module class

1 First, create an equipment module class category of StateDriven under Equipment Module Classes in the Advanced Definitions section of the Library.

2 Right-click StateDriven and select New Equipment Module Class.

3 On the New Dialog, enter BLENDER_OUTLET as the Object name.

4 Select Create New and select State-driven as the algorithm type.

5 Browse for the named set outlet_state.

6 Click OK to create the equipment module class.

7 Drag the NC_VALVE control module class to BLENDER_OUTLET. Rename the resulting module block (NC_VALVE_1) to PRODUCT_VLV.

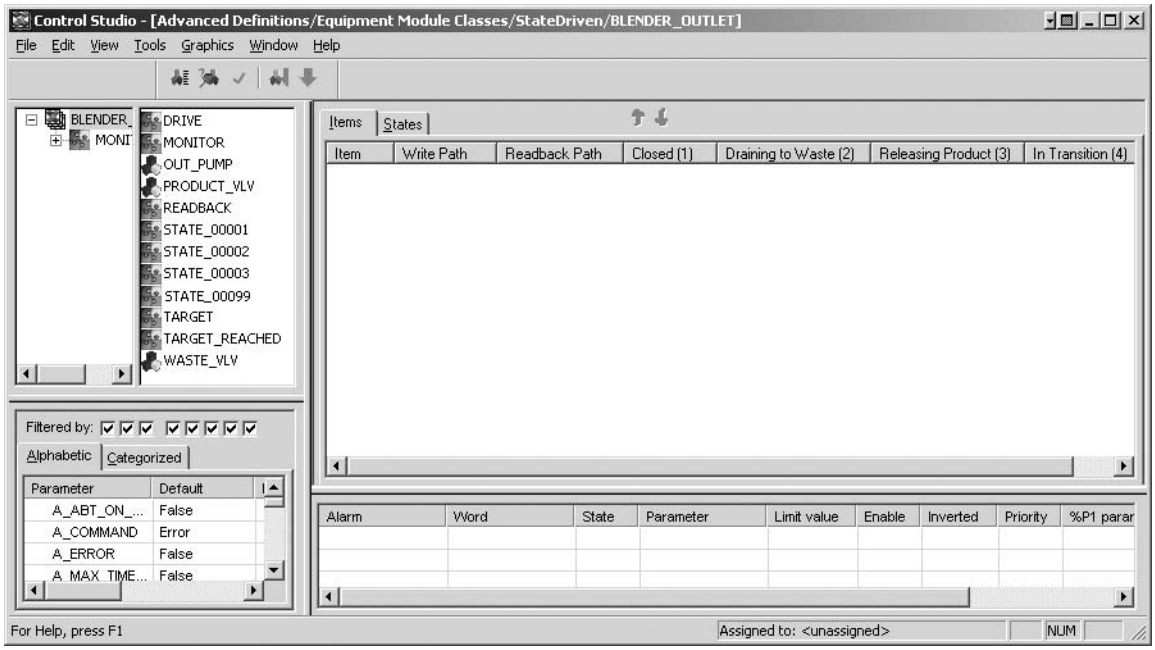
8 Drag the NC_VALVE control module class to BLENDER_OUTLET and rename the new module block WASTE_VLV.

9 Drag the 2STATE_MTR control module class to BLENDER_OUTLET and rename the new module block OUT_PUMP.

10 Open the Properties dialog for BLENDER_OUTLET. On the Displays tab, browse for and select Process as the primary control display.

Configuring the BLENDER_OUTLET Equipment Module Class

Open the BLENDER_OUTLET equipment module class in Control Studio. Note that it contains module blocks that represent OUT_PUMP, PRODUCT_VLV, and WASTE_VLV. The equipment module also contains composite blocks for DRIVE, MONITOR, READBACK, TARGET, and TARGET_REACHED which are default blocks created for all state-driven equipment module classes. In addition, it contains four composite blocks beginning with STATE that represent the items in the named set outlet_state. See the figure below.

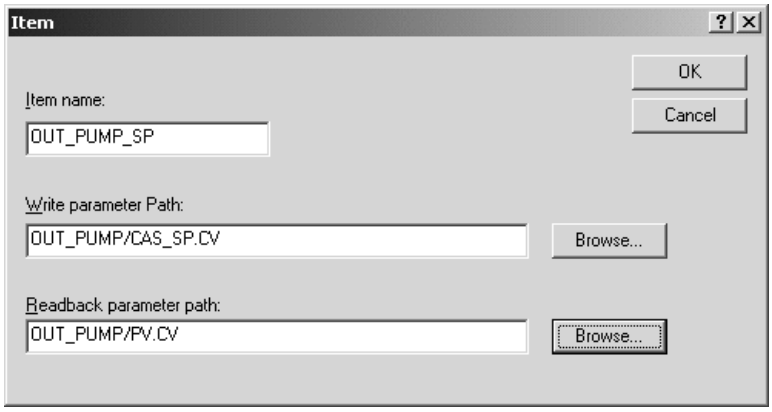


The top right pane is used to configure the logic for the various states in the state-driven algorithm.

To add items to the BLENDER_OUTLET module class

1 Place the cursor in the upper right pane, right-click, and select Add.

2 In the Item dialog, enter OUT_PUMP_SP as the Item Name and browse for the write parameter path and readback parameter path as shown in the following dialog box:



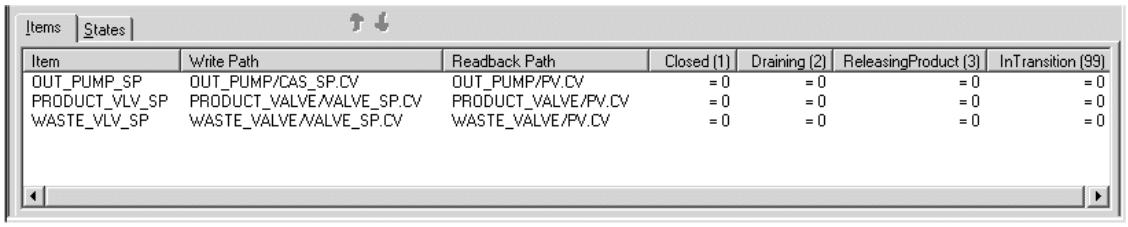
3 Add a second item, PRODUCT_VLV_SP, and set the parameter paths as follows:

Drive: PRODUCT_VLV/VALVE_SP.CV
Readback: PRODUCT_VLV/PV.CV

4 Add a third item, WASTE_VLV_SP, and set the parameter paths as follows:

Drive: WASTE_VLV/VALVE_SP.CV
Readback: WASTE_VLV/PV.CV

The drive item table now looks like this:



Item	Write Path	Readback Path	Closed (1)	Draining (2)	ReleasingProduct (3)	InTransition (99)
OUT_PUMP_SP	OUT_PUMP/CAS_SP.CV	OUT_PUMP/PV.CV	= 0	= 0	= 0	= 0
PRODUCT_VLV_SP	PRODUCT_VALVE/VALVE_SP.CV	PRODUCT_VALVE/PV.CV	= 0	= 0	= 0	= 0
WASTE_VLV_SP	WASTE_VALVE/VALVE_SP.CV	WASTE_VALVE/PV.CV	= 0	= 0	= 0	= 0

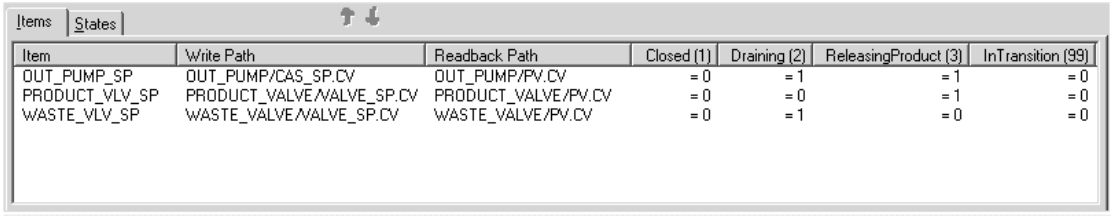
Now we will edit the values in the table to specify the required value of each parameter setpoint for each of the defined states. (A value of 0 will send a closed setpoint to the item and a value of 1 will send an open setpoint to the item.) When the state is Closed, all setpoints should be 0. When the state is Draining, the outlet pump should be on (1) and the waste valve open (1). When the state is Releasing Product, the outlet pump should be on (1) and the product valve open (1).

To edit the parameter setpoints

1 Double-click the value 0 in the Draining column for the OUT_PUMP_SP.

2 On the OUT_PUMP_SP Properties dialog, change the value from 0 to 1.

3 Change other parameter setpoint values so that the table looks like the following, save the file, and close Control Studio.



Item	Write Path	Readback Path	Closed (1)	Draining (2)	ReleasingProduct (3)	InTransition (99)
OUT_PUMP_SP	OUT_PUMP/CAS_SP.CV	OUT_PUMP/PV.CV	= 0	= 1	= 1	= 0
PRODUCT_VLV_SP	PRODUCT_VALVE/VALVE_SP.CV	PRODUCT_VALVE/PV.CV	= 0	= 0	= 1	= 0
WASTE_VLV_SP	WASTE_VALVE/VALVE_SP.CV	WASTE_VALVE/PV.CV	= 0	= 1	= 0	= 0

We now have all the basic control module classes and equipment module classes that will be used for this application. Now we can create the remainder of the equipment hierarchy. This includes creating the unit classes in the Advanced Definitions section of the library and the unit modules in the System Configuration section. After that we will create the phase logic that will be assigned to the unit classes.

Unit Classes and Unit Modules

A group of units that share common properties are said to belong to the same **unit class**. Unit class categories help you to further organize your unit classes. We will create two categories of unit classes, Tanks for the ingredient tanks (both base and color) and Blenders for the blenders.

Creating Unit Class Categories and Unit Classes

To create unit class categories and unit classes

1 In the DeltaV Explorer, select Library | Advanced Definitions | Unit Classes.

2 Create a new category and name it Tanks.

3 While pointing to Tanks, right-click and select New Unit Class.

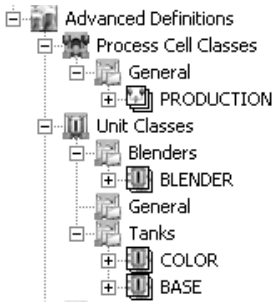
4 Name the new unit class COLOR. Click OK.

5 Under the Tanks category, create another unit class named BASE.

6 Create a unit class category named Blenders.

7 Within the Blenders category, create a unit class named BLENDER.

When expanded, this part of the Advanced Definitions section of the Library should look like this:

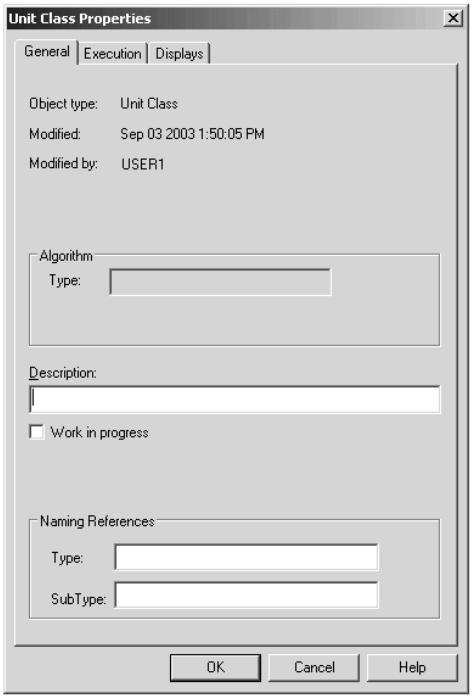


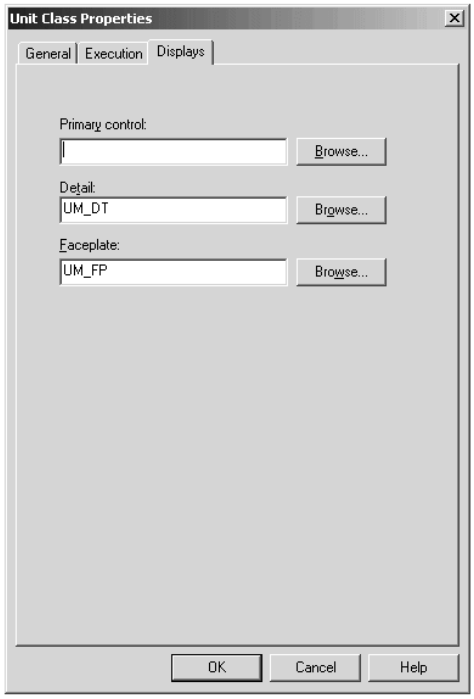
Unit Class Properties

When you create a unit class, you can specify properties for the class. Open the Unit Properties dialog for the COLOR unit class by right-clicking COLOR and select Properties from the context menu.

On the General page of the Unit Properties dialog, there is a field for entering a description of the unit class and marking it as Work in progress. There are also fields for creating searchable information (types and subtypes), which can be useful when exporting unit modules to a user-defined format. On the Displays page, you can associate unit modules created from this class with a picture file (such as Process.grf). Default displays are defined for unit module detail and faceplate displays. On the Execution page, you can define how frequently the unit module is scanned.

Don't make any changes on these dialogs at this time. Click Cancel to close the dialog box.





Creating Unit Parameters

When you create unit modules for an application, you can define specific values for some types of unit class information (unit parameters and aliases).

The following table shows the unit parameters we will create for the BLENDER and COLOR unit classes..

Unit Class	Parameter Name	Parameter Type	Values
BLENDER	CAPACITY	Floating Point	1000 (default)
	CLEAN	Boolean	true (default)
COLOR	CAPACITY	Floating Point	500 (default)

You can make use of unit parameters in your control logic and in your recipes. For instance, you could use the CLEAN parameter to identify a blender that is clean and ready for use. Then, in a recipe, you could use the CLEAN parameter to select a clean tank to use in a batch. After being used for blending, the blender could be marked by the control logic as not clean (that is, CLEAN=false). We won't actually be using the unit parameters in this way in this tutorial, but you should be aware of their potential.

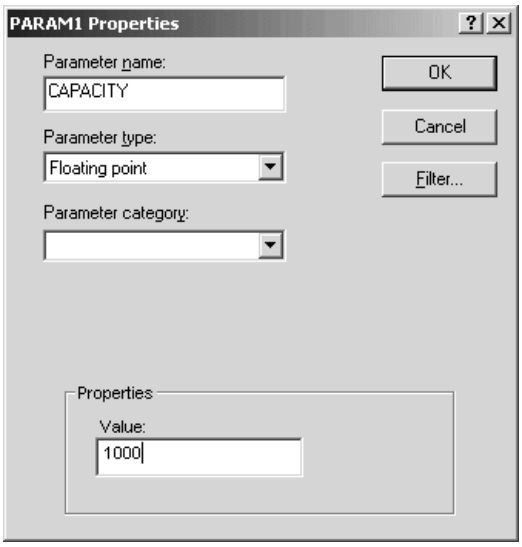
To create a unit parameter definition

1 In the Library section of the DeltaV Explorer, select the BLENDER unit class.

> **Note** Every new unit class has unit parameters named FREMEM and NUMPHASES. The FREMEM value is the amount of the free memory available in a node, and the NUMPHASES parameter contains the number of phases currently loaded on the unit.

2 Right-click and select New | Unit Parameter Definition from the context menu.

3 In the PARAM1 properties dialog box, enter the parameter name (CAPACITY), parameter type (Floating point), and default value (1000). Leave the Parameter category field blank.



4 Click OK to create the unit parameter.

5 Repeat the steps to create the additional parameters listed in the table.

Creating Alias Names in the Unit Classes

Alias names are automatically created for any control module classes/equipment module classes when they are added to a unit class. **Alias names** are placeholders for the actual parameter reference path strings used in the phase logic defined in a phase class. Using aliases makes the phase logic more flexible by allowing phases to be used by multiple units/equipment.

You can create alias names directly in the DeltaV Explorer and then bind the alias by browsing to the parameter reference path. It is also possible to create an alias name and then drag a control module class or equipment module class to the alias name. This creates in the unit class a module block for the equipment or control module class with the alias name.

Alias names can be used for all or a portion of the parameter reference path (module/function block/parameter.field). An alias can also be used to point to a unit parameter on a unit module. Examples of valid alias names and their reference paths are:

#AGITATOR# = AGIT_500

#INLET_VLV# = XV_BASE_500/DC1/SP_D.CV

To create an alias name

1 In the DeltaV Explorer, under Library | Advanced Definitions | Unit Classes | Blenders, select BLENDER and right-click.

2 Select New | Alias. A new alias, ALIAS1, appears in an edit box in the right pane, ready for you to rename.

3 Rename ALIAS1 to TANKCAPACITY.

Later, when the unit module UM_BLEND_500 is created, the path for the TANKCAPACITY alias can be bound by browsing to the following:

UM_BLEND_500/CAPACITY.CV

Alias names can also be derived automatically from the phase logic. (That is, if you make reference to a new alias in the phase logic, it will automatically be added to the unit class and to the Alias Resolution Table when the unit module is created.) An advantage to identifying the aliases when you create the unit class is that you can browse for them when you create the phase logic.

Adding Module Classes to Unit Classes

At this point we have all the control module classes and equipment module classes we need for the application. In earlier exercises, we created unit classes (COLOR, BASE, and BLENDER) to hold the control modules and equipment modules for the color tank units, base unit, and blender units. Now we will add the required module classes to these unit classes. The easiest way to do this is to drag-and-drop the control module class (or equipment module class) to the unit class and then rename the resulting module blocks to more meaningful names. Note that when you rename the module block, the associated alias is also renamed.

To add module classes to unit classes

1 Drag-and-drop the following module classes to the BLENDER unit class and rename them as indicated:

2STATE_MTR (Rename to AGITATOR)
NC_VALVE (Rename to COLOR_IN_VLV)
NC_VALVE (Rename to BASE_IN_VLV)
LEVEL_INDICATOR (Rename to BLENDER_LEVEL)
BLENDER_OUTLET (Leave the name as BLENDER_OUT_1)

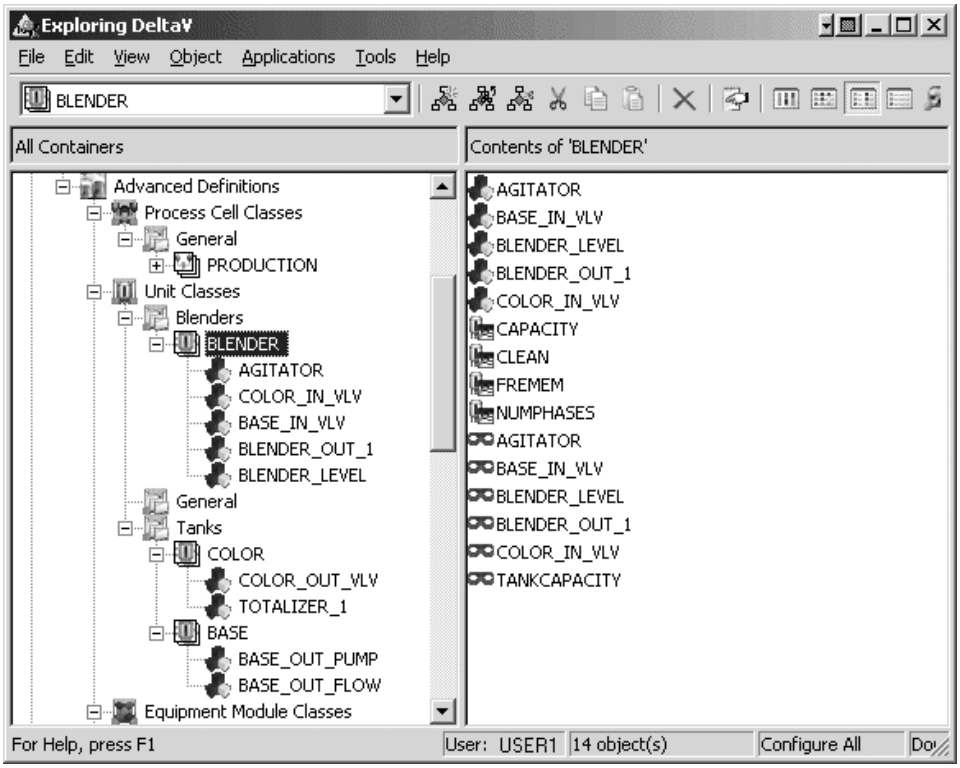
2 Drag-and-drop the following module classes to the BASE unit class and rename them as indicated:

2STATE_MTR (Rename to BASE_OUT_PUMP)
FLOW_CONTROL (Rename to BASE_OUT_FLOW)

3 Drag-and-drop the following module classes to the COLOR unit class:

NC_VALVE (Rename to COLOR_OUT_VLV)
TOTALIZER (Leave the name as TOTALIZER_1)

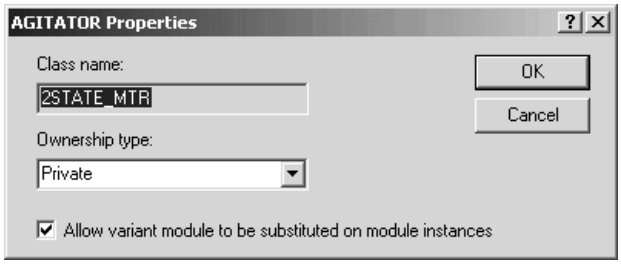
4 Click on the BLENDER unit class. Note that in addition to module blocks that represent the control module classes and equipment module class, the unit class also contains aliases (marked by a "mask" icon) that have the same names as the module blocks.



Ownership Type for Module Blocks

When you drag-and-drop a module class (such as control module class or equipment module class) to another module class (such as an equipment module class or unit class), the result is that a module block is created under the container class. (For example, in the previous illustration, the BLENDER unit class is a container that holds module blocks representing several control module classes as well as a module block for the equipment module class BLENDER_OUTLET. When the unit class BLENDER is instantiated, that is, when it is dragged to the process cell to become a unit module, all those contained module blocks will either be converted into actual control modules/ equipment modules or they will need to be "bound to" modules.

If you open the Properties dialog for one of the module blocks, such as AGITATOR in the BLENDER unit class, you will see that there is a field for specifying the type of ownership--either Private or Shared.



Private is the default and means that when you create an instance of the container module, any contained module blocks are converted to actual modules. Selecting Shared creates an unbound module block. At some point, the unbound module blocks must be bound to an actual module instance. Sharing is typically done when there is one control module, such as a water inlet, that is used by a number of different unit modules. You don't want to create multiple copies of the control module; you simply want to "bind" the contained module block to the actual control module.

Variant Module Substitution

There is a check box on the Properties dialog for contained module blocks for specifying whether or not to allow a variant module to be substituted on module instances. (By default the box is checked.) Sometimes individual units are not identical in every way to the unit class from which they are derived. For example, some units may have a slightly different type of outlet valve that nonetheless performs the same function. To accommodate these differences, you can use variant module substitution, which lets you substitute any module that has the same top-level interface parameters as defined by the module class. (The substitute module can have additional parameters, but must have at least all the parameters of the original module). The module substituted can be class-based or non-class-based. To do the substitution, simply delete the module instance in the unit module and replace it with a module of the same class as the original by editing the module block properties and defining the bound module path. When you delete the module block instance, the block name is retained and marked by <unbound>. Copy and paste to the area or process cell a copy of the module you wish to substitute. Open the Properties dialog for the unbound module block and, under Bound module name, browse to the name of the module you want to use.

Creating Unit Modules

Unit modules are the highest level control entities in the physical hierarchy in a DeltaV batch application. Each unit module represents a piece or group of equipment used to carry out part of a batch process; a unit module executes phase logic to carry out that process. A unit module contains module blocks representing equipment modules and control modules that identify the actual equipment to be manipulated when that unit module runs.

You can create unit modules by dragging a unit module class to the Process Cell (or Area) in the System Configuration | Control Strategies section of the DeltaV Explorer. When the unit module is created, all the contained equipment module classes and control module classes are instantiated.

If you are not using class-based modules, the control modules associated with the unit module can be dragged and dropped into the unit module.

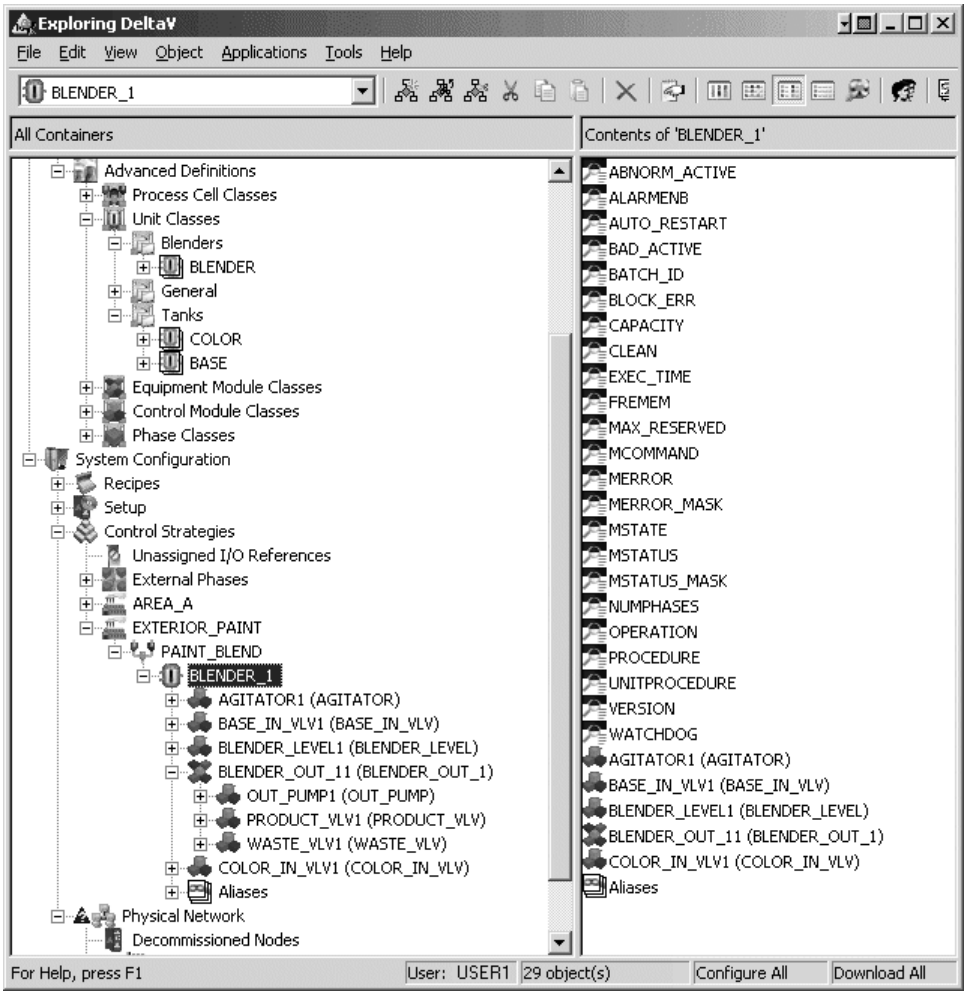
Our sample paint application will have unit modules for one base tank (UM_BASE_400), three color tanks (UM_COLOR 100, UM_COLOR_200, and UM_COLOR_300), and two blenders (UM_BLENDER_500 and UM_BLENDER_600). After a unit module is created, we will rename each of the contained control module and equipment module instances to make the names more meaningful for the individual unit. Note that, in the DeltaV Explorer hierarchy, the name of the module block from which the module instance was created appears in parentheses after the module name to help in identifying the source. (The module block properties dialog can be opened to see the name of the module class.)

To create the BLENDER unit modules

1 Drag the BLENDER unit class to the PAINT_BLEND process cell under EXTERIOR_PAINT area in the System Configuration | Control Strategies section of the DeltaV Explorer. A unit module, BLENDER_1, is created.

2 Expand BLENDER_1 and the contained equipment module, BLENDER_OUT_11, to see the contents. The unit module contains module blocks for four control modules (AGITATOR1, BASE_IN_VLV1,

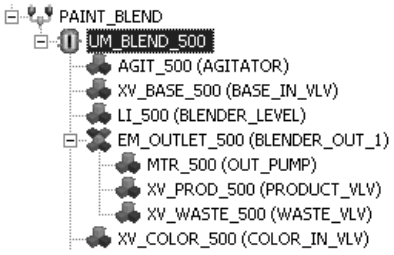
BLENDER_LEVEL1, and COLOR_IN_VLV1); the equipment module BLENDER_OUT_11 contains control modules OUT_PUMP1, PRODUCT_VLV1, and WASTE_VLV1.



3 Rename BLENDER_1 to UM_BLEND_500.

4 Rename AGITATOR1 to AGIT_500.

5 Rename the remaining modules as shown below:

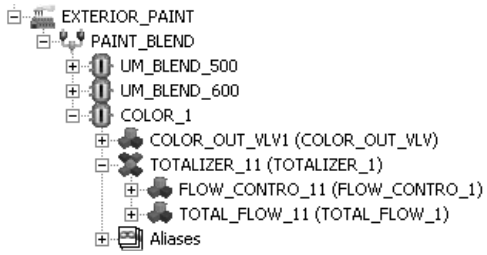


6 To create the UM_BLENDER_600 unit module, drag another copy of the BLENDER unit class to the PAINT_BLEND process cell.

7 Rename the resulting module instances as in steps 3 through 5, but substitute 600 for 500 in the module names.

To create the COLOR unit modules

1 Drag the COLOR unit class to the PAINT_BLEND process cell. Expand the COLOR_1 unit module and the contained equipment module, TOTALIZER_11 to see all the module instances contained within the unit module.

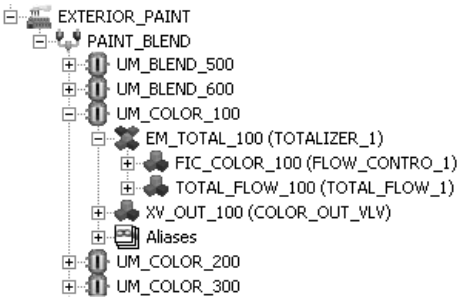


2 Rename COLOR_1 to UM_COLOR_100.

3 Rename the remaining modules under UM_COLOR_100 as shown in the figure below.

4 To create the UM_COLOR_200 and UM_COLOR_300 unit modules, drag two more copies of the COLOR unit class to the PAINT_BLEND process cell.

5 Rename the resulting module instances as in the figure below, but use 200 and 300 (for UM_COLOR_200 and UM_COLOR_300, respectively), in all the module names.



To create the BASE unit module

1 Drag the BASE unit class to the PAINT_BLEND process cell. Expand the unit module to see its contents.

2 Rename BASE_1 to UM_BASE_400.

3 Rename BASE_OUT_FLOW1 to FIC_BASE_400.

4 Rename BASE_OUT_PUMP1 to MTR_400.

Specifying Unit Parameter Values

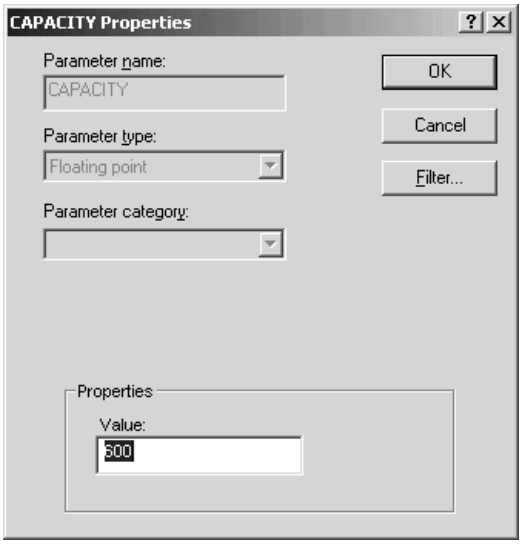
When we defined the COLOR unit class, we set up a unit parameter to define the capacity of the color tanks and gave it a default value of 500. Now that we have individual color unit modules, we can specify a different value for each tank's capacity by editing the unit parameter properties for the unit module.

To specify the unit parameter value for UM_COLOR_100

1 Select UM_COLOR_100.

2 In the right pane, double-click the unit parameter CAPACITY to open the Properties dialog.

3 Change the Value to 600.



Binding Aliases

Earlier we created an alias, TANKCAPACITY, that can be used to reference the unit parameter CAPACITY. Now we need to bind the alias to the unit parameter in both the Blender unit modules.

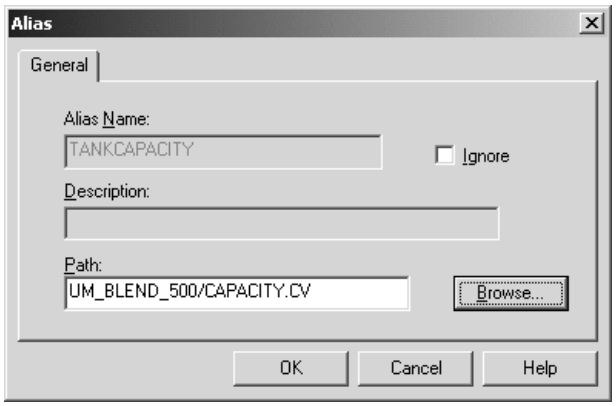
To bind the alias to the unit parameter

1 Under UM_BLEND_500, click on Aliases.

2 Click View | Details to see the parameter reference paths to which the aliases are bound. Note that TANKCAPACITY is not bound.

3 Double-click TANKCAPACITY to open the Alias dialog.

4 Browse to the parameter reference path as show below:



5 Repeat the process for UM_BLEND_600.

Note If an alias is not going to be bound to a parameter reference path, it is important to open the property dialog for the alias and select the Ignore box. See the topic Defining Aliases in the Batch Reference manual for more information about the Ignore field.

Finishing the Unit Modules

There are a few more things we need to do to finish the unit modules for the color tanks. For each unit module, we will perform the following tasks:

- Change the phase type to Controller for phases assigned to the unit module. (This will be demonstrated in a later chapter, after we create phases and assign them to units.)
- Assign the unit module to a node (usually the controller, but, for simulation, it may be the workstation). The easiest way to assign unit modules to a node is to drag the entire area to the node. To assign individual unit modules, there are several different methods that can be used. In the DeltaV Explorer, assignment can be done on the Tools tab of the Unit Module Properties dialog. It can also be done using the context menu (Assign | Unit and All Contained Modules to Node) or using the drag-and-drop method. In Control Studio, the assignment can be made from the File menu or using the Assign to Node button.
- Assign the area (EXTERIOR_PAINT) to the Alarms And Events Subsystem under the workstation. This is necessary because the batch operator is only allowed to operate modules that are assigned to the Alarms and

Events subsystem for the particular operator station. This prevents an operator from inadvertently causing an alarm condition on a module that cannot be viewed on the operator's station.

- Download the controller or workstation.

To finish the unit modules

1 Drag the EXTERIOR_PAINT area to the controller or to Assigned Modules under the Workstation in the Control Network. All the contained modules will automatically be assigned to the node.

2 Drag the EXTERIOR_PAINT area to the Alarms and Events Subsystem under the Workstation. A message appears to remind you to download the station's Setup Data. Click Yes to confirm that you want to assign the area.

3 Download the node. (You may need to download the controller, the workstation, the Control Network, or the Physical Network, depending on the download status of the network elements.)

At this point all the equipment entities have been defined. Now we will move on to configuring the phase logic that will be used in the process.

Phase Classes

A phase is a series of steps that cause one or more equipment- or process-oriented actions, for example, filling a tank or agitating the contents. The phase logic defines the states of the phase (running, holding, restarting, aborting, and stopping) and the logic associated with each state. In most batch applications, the phase logic is defined in the phase class.

In this chapter you will learn about creating phase classes and their associated batch input and report parameters. You will also be introduced to state transition diagrams and how they are used to define the phase logic for the different states of each phase. In future chapters you will do the actual configuration and testing of the phase logic.

Phase Classes and Batch Parameters

Phase classes define common properties for one or more phases. Phase classes are defined in the Library section of the DeltaV Explorer. Phase classes can be grouped into phase class categories for organizational purposes. For this tutorial, you will create two phase class categories, BlenderPhases and TankPhases.

You will also create batch input parameters and batch report parameters for each of the phase classes.

Batch input parameters specify values that are downloaded to the phase logic in the controller when the phase starts. The phase logic uses these values to control the unit in which the phase runs. Values for batch input parameters can be specified in a recipe, included in recipe formulas, or can be requested from the operator at runtime.

Batch report parameters are typically used to capture and report actual process values or batch values used by the phase. From the phase logic in the controller, batch report values are uploaded to the DeltaV Batch Executive when the phase completes and can be used for batch history purposes. They can also be used in building recipes. For example, the value of a report parameter could be used to determine what to do next in the recipe.

Creating Phase Class Categories

The BlenderPhases category was created as part of the import of two phase classes, CHG_BASE and DRAIN, which were part of the Startup.fhx file. Now we will create another phase class category, TankPhases, to hold the phase classes associated with the color and base tanks.

To create a phase class category

1 In the DeltaV Explorer, select Library | Advanced Definitions | Phase Classes.

2 Right-click and select New Category from the context menu.

3 Rename Category1 to TankPhases.

Creating Phase Classes and Parameters

Now we are going to create phase classes and their associated batch input and report parameters. After that, we will create the logic for the phase class. The input parameters will be used later when we create recipes. One of the report parameters will be used later to log a value to the Event Chronicle.

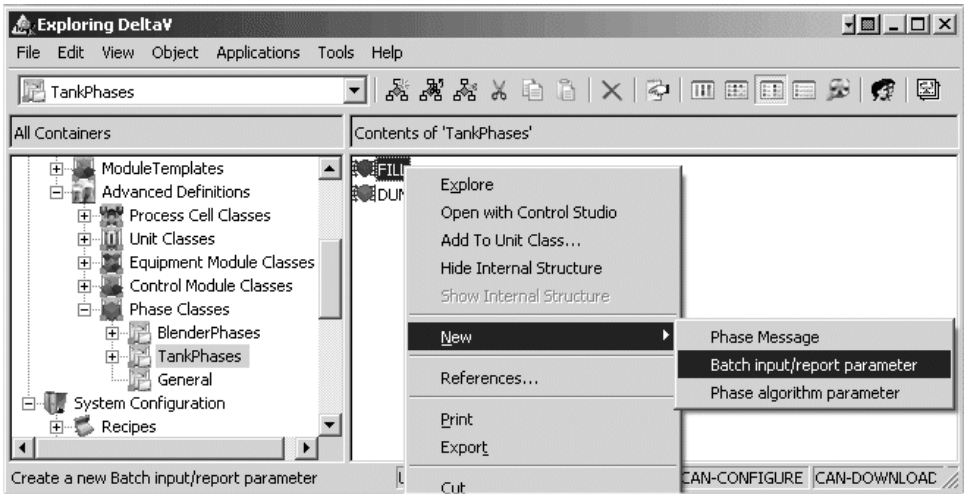
To create a phase class and batch input/report parameters

1 In the DeltaV Explorer, select the TankPhases phase category.

2 Right-click and select New Phase Class. Name it FILL.

3 Create another phase class and name it DUMP.

4 Select the phase class FILL, right-click and select New | Batch input/report parameter from the context menu.



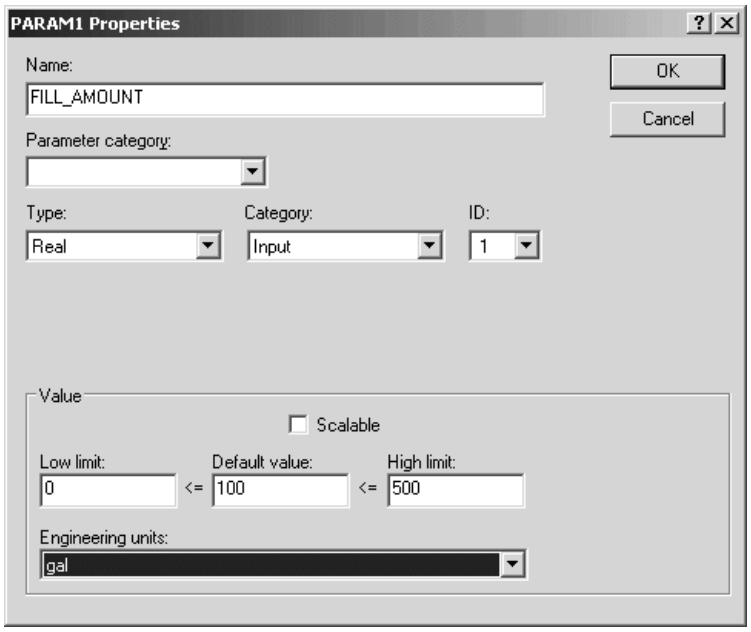
A Parameter Properties dialog box opens.

5 In the Parameter Properties dialog box, enter the name FILL_AMOUNT and select the following choices from the drop-down boxes:

- Type: Real
- Category: Input

(The ID field is used as an identifier that can be used to specify this parameter in phase requests. Phase requests are discussed later in this tutorial.)

6 In the value boxes, leave 0 as the Low Limit, enter 100 as the Default Value and 500 as the High Limit, and select gal for Engineering Units. (For Engineering units, type gal.)



Note The option Scalable can be used to indicate that, if the batch recipe is scaled (for example, by 50 percent), this is one of the parameters that will be scaled by the given percentage. For example, you may want to be able to scale the amounts of the input ingredients by the same percentage, but you may not want to scale the mixing time by the same percentage. Therefore, you would not check scalable for the mixing time parameter.

7 Click OK.

Creating Phase Classes and Parameters for Blender Phases

The following table contains information for the phase classes and batch input/report parameters for the BlenderPhases category. We will create the first two phase classes, AGITATE and CHG_COLOR. The other two phase classes, CHG_BASE and DRAIN, were imported as part of the Startup.fhx file.

Phase Class	Parameter Name	Type	Category	Def. Value	Range (Low-High)	Eng. Units
AGITATE	AGITATE_TIME	Integer	Input	30	0-300	sec
	ACTUAL_TIME	Integer	Report			sec
CHG_COLOR	CHARGE_AMT	Real	Input	15	0-100	gal
	ACTUAL_CHARGE	Real	Report			gal
CHG_BASE	CHARGE_AMT	Real	Input	150	0-1000	gal
	ACTUAL_CHARGE	Real	Report			gal
DRAIN	DRAIN_LEVEL	Real	Input	5	0-1000	gal
	ACTUAL_LEVEL	Real	Report			gal

Batch input parameters are denoted by a green parameter icon in the DeltaV Explorer, whereas batch report parameters have a yellow parameter icon.

To create the AGITATE and CHG_COLOR phase classes

Perform the following steps according to the information in the above table:

1 Create the phase class under the BlenderPhases category.

2 Create the batch input/report parameter.

3 In the Parameter Properties dialog, enter the parameter name.

4 Select the type and category (either input or report).

5 For batch input parameters, enter the range (Low and High Limit Values), the Default Value, and the Engineering Units.

6 For batch report parameters, enter the Engineering Units.

State Transition Diagrams

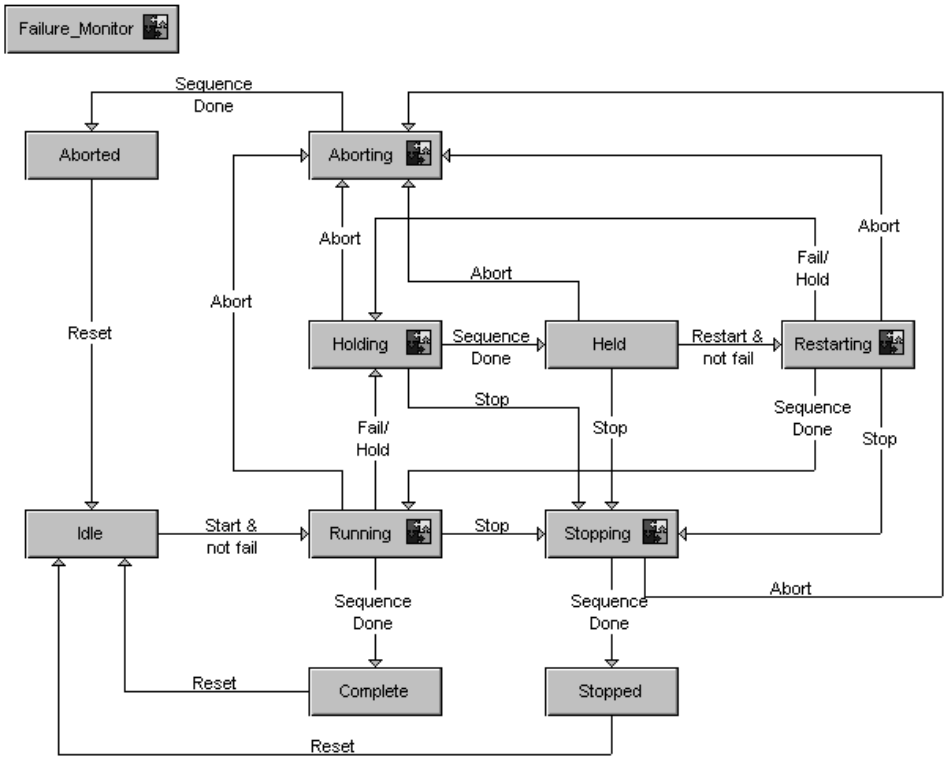
Before doing the exercises for creating the phase logic, read the next few topics to gain a better understanding of what comprises the phase logic. The phase logic defines the states of a phase (running, stopping, holding, and so on) and the logic associated with each state. The phase logic executes in the DeltaV controller. (In simulation mode, it can execute in the workstation.)

When you open a phase class in Control Studio, you will see that it contains a preconfigured State Transition Diagram that governs the transitions between the phase states. (The overall flow of this State Transition Diagram cannot be changed.) This diagram shows the active states (when the phase is doing something, such as executing its normal control actions or an orderly shutdown), the static states (when the phase is waiting for a command, such as Start or Reset), and the paths between states. The phase logic also contains a Failure Monitor block.

The configuration engineer uses Control Studio to write a sequential function chart (SFC) for each active state (Running, Holding, Stopping, Restarting, and Aborting). The active states are shown in the State Transition Diagram as embedded composite blocks, marked by  . The Failure Monitor is configured in Control Studio using a Function Block Diagram. The static states (Idle, Held, Complete, Stopped, and Aborted) cannot be modified and do not contain any logic.

The paths on the diagram follow a defined sequence for going from one state to another. The words along the paths are either Sequence Done (meaning an active phase has completed and is going to the following static state) or they are specific commands that can be used in the phase logic to transition from the current state to the next. For example, the Stop command can be used to go from Running (or Holding, Held, or Restarting) to Stopping.

Note In this tutorial we show examples of the logic for the different states, but we don't fully configure every state for every phase. However, it is important that in your real batch system you carefully consider which of the active states you will use and configure the logic for those states as well as for the Failure_Monitor. If you don't configure logic for an active state, it defaults to true (Sequence Done) and the phase goes to the next state along the path.



Phase States

The following table describes the states that appear on the State Transition Diagram. States ending in "ing" and marked by an asterisk are the **active** states..

State	Description
Idle	The phase is ready for a Start command.
Running*	Normal sequence is in progress.
Complete	Normal sequence has completed.
Stopping*	Permanent (but not emergency) shutdown is in progress.
Stopped	Permanent (but not emergency) shutdown has completed.
Aborting*	Emergency shutdown is in progress.
Aborted	Emergency shutdown has completed.
Holding*	Temporary hold or failure handling sequence is in progress.
Held	Temporary hold or failure handling sequence has completed; the phase is ready for a Restart command.
Restarting*	Sequence for recovery from hold or failure handling is in progress.

Phase Commands

Normally, when the sequential function chart for an active state completes without error, the phase continues along the defined path to the next state. However, there may be times when you want to force a transition to a particular state. (It must still be along one of the predefined paths.) In such cases, you can use a phase command in your SFC logic.

The main phase commands that are used to transition from one state to another are listed in the table below. These commands can be used in the phase logic. There are equivalent batch commands that an operator can issue to control a batch from the Batch Operator Interface. (Additional batch commands, such as Pause, Resume, and Clear All Failures, are described in the *Batch Reference* manual in the Batch Commands topic.)

Command	Description
Abort	Invokes the aborting logic. Abort is an emergency stop.
Hold	Used for a phase in the running or restarting state. Hold invokes the holding logic to temporarily stop the phase and proceed to the held state, from which the Restart command can be used.
Reset	Transitions the phase from an aborted, complete, or stopped state to the idle state.
Restart	Used for a phase in the held state to invoke the restarting logic.
Start	From idle, the Start command invokes the running logic.
Stop	Invokes the stopping logic.

Common Phase Logic Parameters

The phase logic starts with a number of default phase parameters, such as BATCH_ID and OWNER. In addition, a unit phase inherits any phase parameters that were created in the phase class. (For example, any unit phase created from the phase class FILL will have the additional phase parameter FILL_AMOUNT.) The default phase parameters are defined in the *Batch Reference* manual in the Phase Logic Parameters topic. The following table shows some of the commonly referenced parameters, a few of which are mentioned in this tutorial.

Parameter	Description
AUTO_RESTART	Set by the Batch Executive to instruct all loaded phases whether or not to auto-restart from a holding condition.
BATCH_ID	Contains the unique batch ID for the phase at run-time.
BCOMMAND	Commands the phase logic to go to a state, as defined by the state transition diagram. This parameter is to be used exclusively by the DeltaV Batch Executive, while the phase owner is DeltaV Batch. **Caution** Do not write to this parameter directly as it can cause the Batch Executive to lose control of the phase.
BSTATUS	Shows the state of the phase.

Parameter	Description
FAIL_INDEX	Shows and defines the failure of a phase, where 0 = No Failure and any non-zero integer = Failure. A translation string for a positive integer less than 255 can be created in the phase_failures named set.
INITIAL_STATE	Defines the state that the SFC is in when downloaded (Sequence Active or Sequence Idle). The default is Sequence Active. This parameter is available in the composite block.
OWNER	Identifies the control mode, DeltaV Batch or External. When the batch control is External, the parameter controlling the phase transitions is XCOMMAND (instead of BCOMMAND), which allows the user to control the phase.
PROMPT_BOOL	Stores the operator response to a prompt requiring a Yes/No response.
PROMPT_INT	Stores the operator response to a prompt requiring an integer response.
PROMPT_FLOAT	Stores the operator response to a prompt requiring a floating point response.
PROMPT_STRING	Stores the operator response to a prompt requiring a string response.
REQDATA1…5	Used to pass request data between the phase logic and the DeltaV Batch Executive.
REQUEST	A request to the DeltaV Batch Executive from the phase logic.
STATE_TIME	Shows the time (in seconds) the module has been in the current state.
TIME	The amount of time the SFC has been running. This is the accumulated time that has passed since the sequence was started.
UNIT	Shows the ID of the unit.
WATCHDOG	Used to verify communication with the DeltaV Batch Executive. The Batch Executive writes to the Watchdog parameter when the phase state is not Idle and the owner is DeltaV Batch. The phase logic checks the Watchdog parameter every time it executes.
XCOMMAND	Commands the phase to go to a particular state, as defined by the state transition diagram. This parameter is used when running the batch externally. See OWNER parameter.

Phase Logic Configuration

In DeltaV Batch, the phase logic is a state transition diagram made up of embedded composites that contain the logic that governs the transition between the states of a phase. The state transition logic is written using Sequential Function Charts (SFCs). Refer to the Using the Expression Editor topic for information on writing expressions.

The phase logic incorporates the use of phase logic parameters. Refer to the Common Phase Logic Parameters topic for specific information on the parameters.

Following are the general steps for creating the phase logic for one of the active states (marked by a composite block) in the phase's state transition diagram:

1 Open the phase class in Control Studio.

2 Drill down into the active state's composite block. (This can be done in several ways: Select the composite block in the left pane; double-click the state in the diagram; or right-click the state in the diagram and select Drill Down from the context menu.)

3 Use steps (and associated actions), transitions, and terminations to create the SFC.

4 Continue until all logic is defined for each active state.

Once defined, download the phase.

Configuring the Running Logic for the FILL Phase - Overview

logic starts with a step to ensure the outlet valve is closed. It then sets the TOTALIZER equipment module to totalize_fast, waits for the fill amount to be 90%, and then sets it to totalize_accurate. It resets TOTALIZER when the desired fill level is reached.

Following is an overview for configuring the running logic. The next six procedures in this chapter step through the tasks outlined below.

- Open the phase class FILL in Control Studio and open the Running composite block for editing.
- Modify the first step in the Running logic's sequential function chart (SFC) to add an action to drive the setpoint of the outlet valve to CLOSE. Use the named set vlvnc-sp, which stands for valve normally closed - setpoint.
- Create a transition to wait for the outlet valve PV to be Closed. Use the named set vlvnc-pv.
- Add a step to set the TOTALIZER equipment module to Totalize_fast.
- Create a transition to wait for the total flow to reach 90% of the desired amount (defined by the batch input parameter FILL_AMOUNT).
- Add a step to set the TOTALIZER equipment module to Totalize_accurate.
- Create a transition to wait for the level to reach the value defined by FILL_AMOUNT.
- Add a step to set the TOTALIZER to Reset.
- Verify the construction of the SFC.

Opening the FILL Running Logic

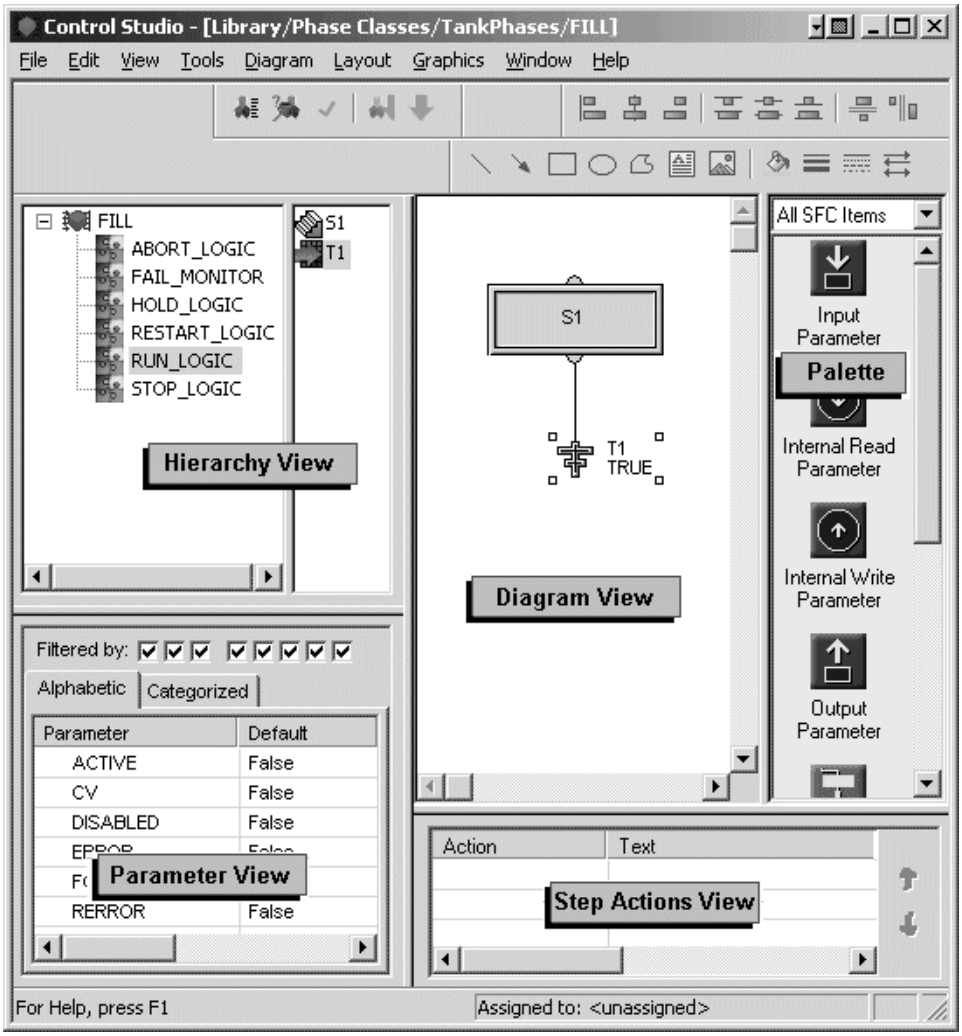
To open FILL and select the Running composite block

1 In DeltaV Explorer, select Library | Advanced Definitions | Phase Classes | TankPhases | FILL.

2 Right-click and select Open with Control Studio from the context menu. Control Studio opens with the standard State Transition Diagram in the diagram window, ready for editing.

3 Double-click Running. (Other options are to right-click Running and select Drill Down from the context menu or click RUN_LOGIC in the hierarchy view).

A sequential function chart with one step and one termination opens.



Adding an Action to the FILL Running Logic

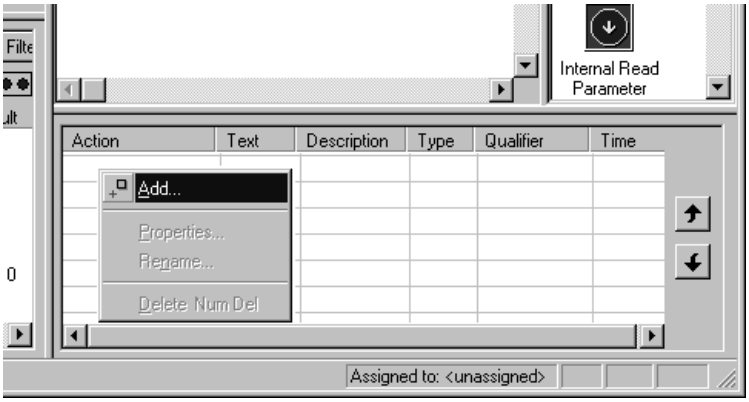
To modify and add an action to the first step in the SFC

1 Double-click the step.

2 In the S1 Properties box, enter the step description: Close outlet.

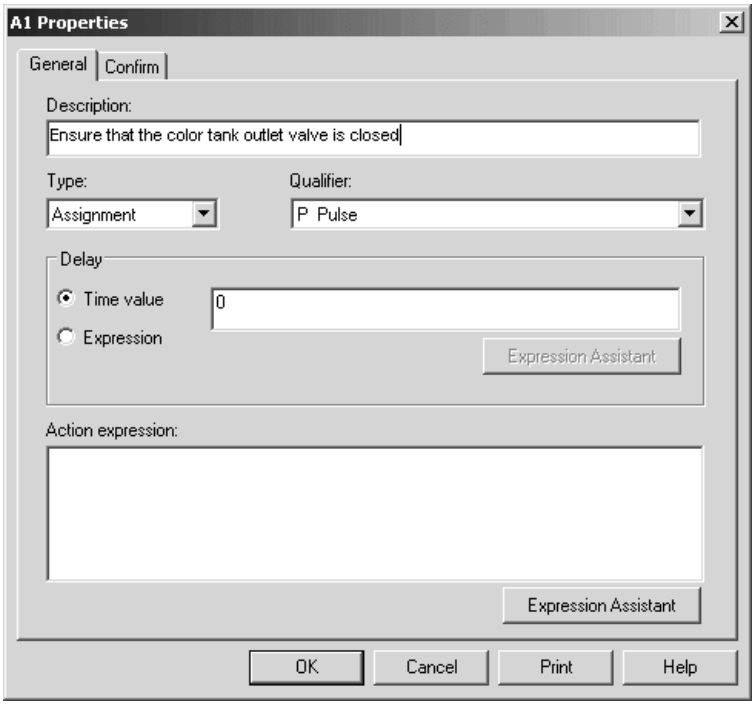
3 Rename the step to CLOSE_OUTLET. (Click S1 in the step box to rename it.)

4 With the CLOSE_OUTLET step box selected, point to the Step Actions View and right-click.

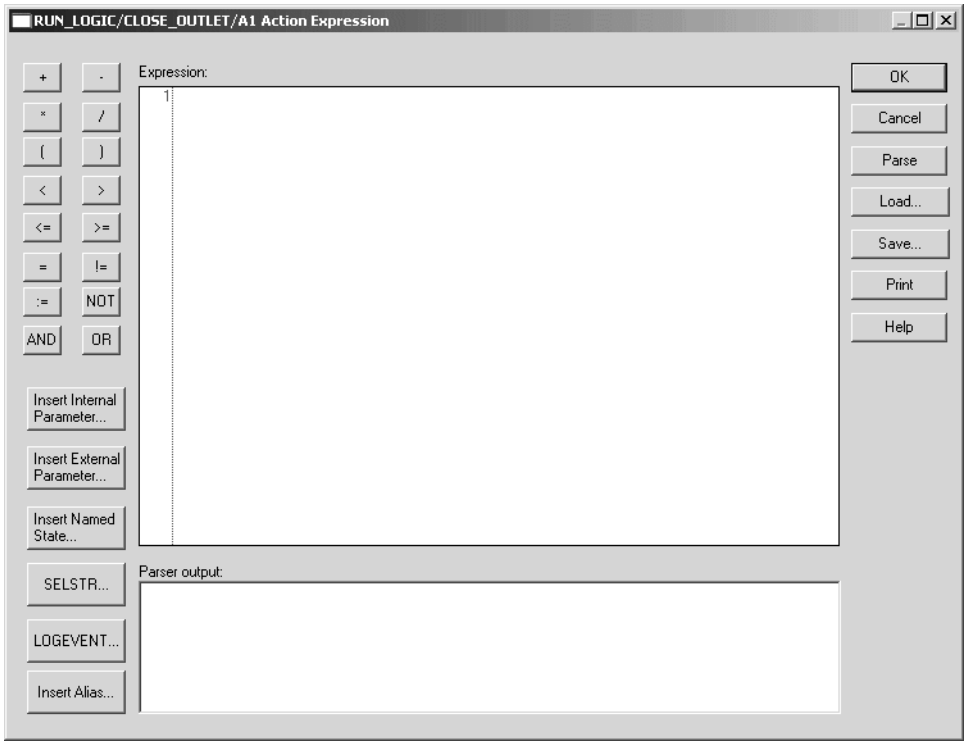


5 Select Add to add an action. A Properties dialog box with two tabs opens.

6 On the General page, enter the description (optional) as **Ensure that the color tank outlet valve is closed**, leave the Type as Assignment, and leave the Qualifier as Pulse.



7 Click Expression Assistant to open the Expression Editor.



8 Click Insert Alias. Browse under Tanks | COLOR for the desired alias. Note that COLOR_OUT_VLV may appear in the browse list as both a module block (COLOR_OUT_VLV) and a simple alias (COLOR_OUT_VLVLV). Double-clicking the module block lets you browse to the module parameter for VALVE_SP, whereas selecting the simple alias allows no further browsing. The first part of the expression should be defined as shown in the following:

```
'//#COLOR_OUT_VLV#/VALVE_SP'
```

9 Next, click the Assignment button := or type := .

10 Then, click Insert Named State. In the Browse dialog box, select the valve setpoint named state (vlvnc-sp) and the state of CLOSE. The completed expression now appears in the window of the Expression Editor:

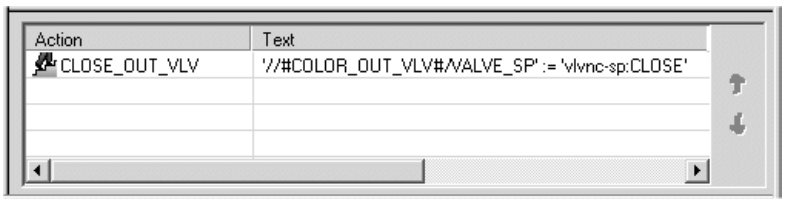
```
'//#COLOR_OUT_VLV#/VALVE_SP' := 'vlvnc-sp:CLOSE'
```

11 Click Parse to check the expression.

12 If the parsing completes without errors, click OK. Otherwise, correct the expression as needed. The expression now appears in the action Properties box.

13 Click OK. The action text appears in the Step Actions View.

14 Rename the step action, A1, to CLOSE_OUT_VLV.



Adding a Transition to the FILL Running Logic

To add a transition to the SFC

1 Move the termination icon to the bottom of the diagram and delete the connecting line.

2 Drag a transition icon from the palette to the spot where the termination was.

3 Rename the transition to OUTLET_CLOSED (to wait until the outlet valve is closed).

4 Double-click the transition. The Properties dialog box opens.

5 Open the Expression Assistant. Delete the default condition of False, select Insert Alias, and browse in Tanks | COLOR for the alias COLOR_OUT_VLV/PV.

6 Type an equals sign (=) or use the button provided.

7 To construct the right side of the transition condition, select Insert Named State, and browse for vlvnc-pv:CLOSED. The expression now reads:

```
'//#COLOR_OUT_VLV#/PV' = 'vlvnc-pv:CLOSED'
```

8 Click Parse to check the expression and correct any errors.

9 Click OK on the Expression Editor and Properties dialog boxes.

10 Draw a line from the first step to the transition. The cursor automatically turns into an asterisk when it is in the correct position for drawing lines. (Hint: To see the name and expression for the transition on the diagram, select Tools | Diagram Preferences from the menu bar and select Name and Condition under the Transitions section of the dialog box.)

Adding a Step to the FILL Running Logic

To add a second step to the SFC

1 Drag a step from the palette to a point below the transition.

2 Change the step name to START_FAST.

3 Add an action to set the TOTALIZER state to Totalize_fast and modify it as follows:

- Check that the Type is Assignment and the Qualifier is Pulse.
- Use the Expression Editor to create the following expression. (TOTALIZER_1 is an equipment module class under the COLOR unit class in the Tanks category. A_COMMAND is a parameter that sets the behavior to the state specified in the named set in the right side of the expression.) Parse and correct if necessary.

```
'//#TOTALIZER_1#/A_COMMAND' := 'totalizer:Totalize_fast'
```

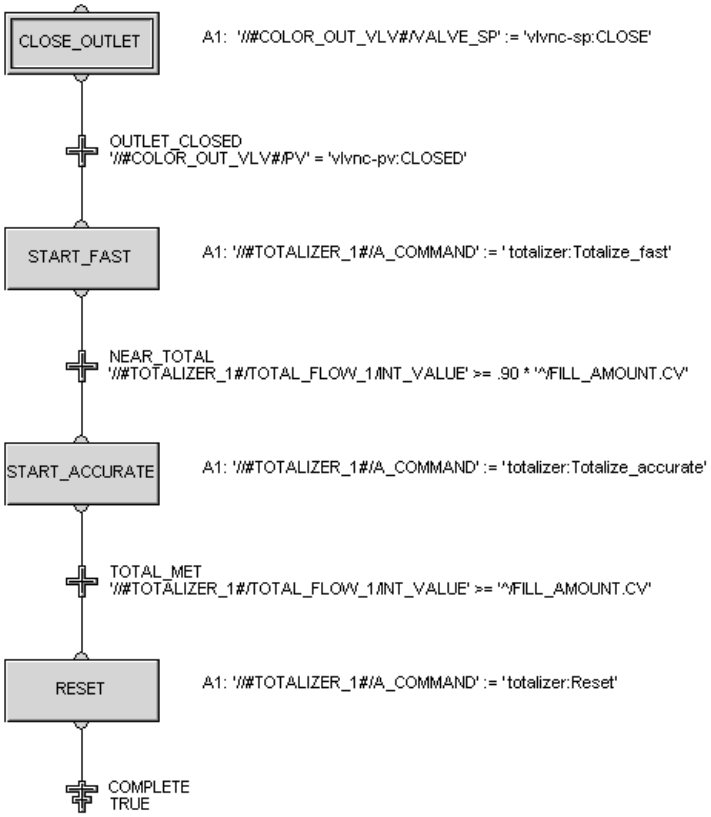
4 Draw a line from the transition to the step.

Creating the Remaining Logic for the FILL Phase

Complete the logic for the FILL phase using information from the table below.

Step/Transition Name	Action or Condition Text
Transition: NEAR_TOTAL	'//#TOTALIZER_1#/TOTAL_FLOW_1/INT_VALUE'>= 0.90 * '^/FILL_AMOUNT.CV'
Step: START_ACCURATE	'//#TOTALIZER_1#/A_COMMAND' := 'totalizer:Totalize_accurate'
Transition: TOTAL_MET	'//#TOTALIZER_1#/TOTAL_FLOW_1/INT_VALUE' >= '^FILL_AMOUNT.CV'
Step: RESET	'//#TOTALIZER_1#/A_COMMAND' := 'totalizer:Reset'
Termination: COMPLETE	TRUE

The completed SFC should look like the following, except that the step actions will not show on the diagram. (We added the step actions to the diagram using the Comment tool on the graphics toolbar.)



Verifying the SFC

The DeltaV system provides a tool to help you check the completeness (not the logic) of your sequential function charts. It checks the syntax of the step actions and transition conditions and verifies the existence of any modules or parameters referenced in the logic. It also checks to see if all the steps and transitions have been connected in a legal form.

To check your SFC

1 Select Diagram | Check SFC or click the checkmark icon on the toolbar.

If the SFC passes the syntax checks, the following message appears.

2 If you get an error message, go back and fix the SFC as needed.

Returning to the State Transition Diagram

You have completed the running logic for the FILL phase logic. To return to the state transition diagram, you can do either of the following:

- Point anywhere on the SFC, right-click and select Back Out from the context menu.
- Click the phase class name, FILL , in the hierarchy view.

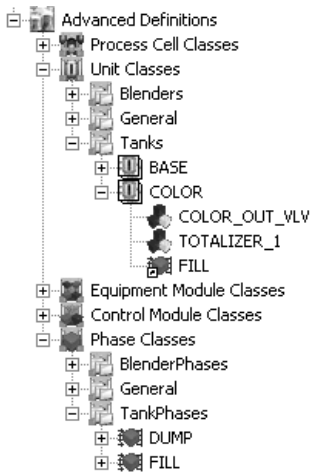
Save the FILL phase class and close Control Studio.

Assigning Phase Classes to a Unit Class

Now that the FILL phase logic has been defined, we can assign the phase class to a unit class. The phase assignment is automatically propagated to any unit modules created from the unit class.

To assign a phase class to a unit class

1 In the DeltaV Explorer, drag the phase class FILL to the unit class COLOR.



2 Assign the phase class DUMP by dragging it to COLOR. (We will configure the logic for the DUMP phase in a later exercise.)

Note the difference in the icons used for phase class  and assigned phase class  .

Additional Steps for the FILL Unit Phases

The color unit modules (UM_COLOR_100, 200, and 300) in the System Configuration section of the DeltaV Explorer now have the FILL and DUMP phases assigned to them. Within the unit modules, the assigned phase classes are called unit phases. There are two things that need to be done at this point:

- Change the phase type for the unit phase to controller
- Download the node

To change the phase type and download the controller

1 Select the FILL unit phase under UM_ COLOR_100, open the Unit Phase Properties dialog, and choose Controller for the phase type. Do the same for the FILL unit phases under UM_ COLOR_200 and UM_COLOR_300.

2 Download the node.

Additional Step for the COLOR Unit Modules

The TOTAL_FLOW control module class (part of the TOTALIZER equipment module class) has a parameter, FEED_FLOW, that is used to get the actual flow value (in gallons per minute) to be used in the calculation of the total amount of feed. This parameter needs to be modified for each of the color unit modules to set the parameter path.

To modify the FEED_FLOW parameter

1 In DeltaV Explorer, select TOTAL_FLOW_100 under the unit module UM_COLOR_100.

2 Right-click the FEED_FLOW parameter in the right pane and open the Properties dialog.

3 Browse for the following External parameter path:

`UM_COLOR_100/EM_TOTAL_100/FIC_COLOR_100/PID1/PV`

4 Repeat the above steps for the FEED_FLOW parameters in UM_COLOR_200 and UM_COLOR_300, substituting 200 (or 300) where appropriate in the names.

Moving the Simulation Modules to the Process Cell

In the next exercises, we will be testing the FILL phase. Before we do this we need to move the simulation modules that we imported as part of the Startup.fhx file to the PAINT_BLEND process cell so that they will be available for our testing and simulation activities.

To move the simulation modules to the process cell

1 Click SIMULATION in the left pane of the DeltaV Explorer (under System Configuration | Control Strategies).

2 In the right pane, select all the simulation modules.

3 Drag the simulation modules to the PAINT_BLEND process cell.

4 Assign the simulation modules to the workstation or controller.

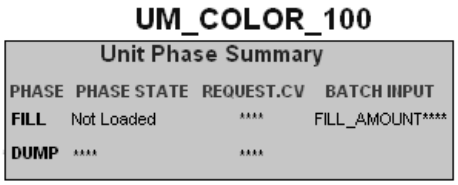
5 Download.

Testing UM_COLOR_100 Using the Batch Graphic

Now you are ready to test the FILL unit phase for UM_COLOR_100. You will start by launching DeltaV Operate in run mode and opening the graphic named Batch, which shows the unit modules you will create for this tutorial. This is not a graphic that you would normally create for an application. We have provided it as a testing tool.

Each box on this graphic represents a unit module with its list of assigned phases. Each box also has data links for request and batch input parameters.

For example, the box representing the UM_COLOR_100 unit module looks like this:



One reason for having the Batch graphic is that it has been set up with data links that allow us to open unit module faceplates and directly manipulate the phases. To open the faceplate for a unit module, click the unit module name. A phase faceplate can then be opened from the unit module faceplate by loading the phase and clicking the faceplate button beside the phase name. At this point in our configuration effort, the Batch graphic provides this direct access to the unit modules and unit phases and is therefore a useful tool for testing and troubleshooting.

After you open the Batch graphic, you will open the faceplate for the UM_COLOR_100 unit module, load the FILL phase, open its faceplate, and then go to the Process graphic to watch what happens when you manipulate the phase states. When you start the FILL phase, the valve state changes to OPEN, and the tank begins to fill, as shown by the fill level. (A Total Flow value appears above the tank.) Then, you will change the FILL_AMOUNT parameter. When the tank fill level is reached, the phase goes to COMPLETE. At that point, you will reset the phase and unload it.

Changing the Phase Owner

When you manually load a phase from the unit module faceplate, the Phase Owner is automatically changed from DeltaV Batch to External. This takes control of the phase away from DeltaV Batch and gives it to an external owner, which, in this case, is you. (You need to have the Restricted Control key to be able to do this. Keys are assigned in the User Manager.)

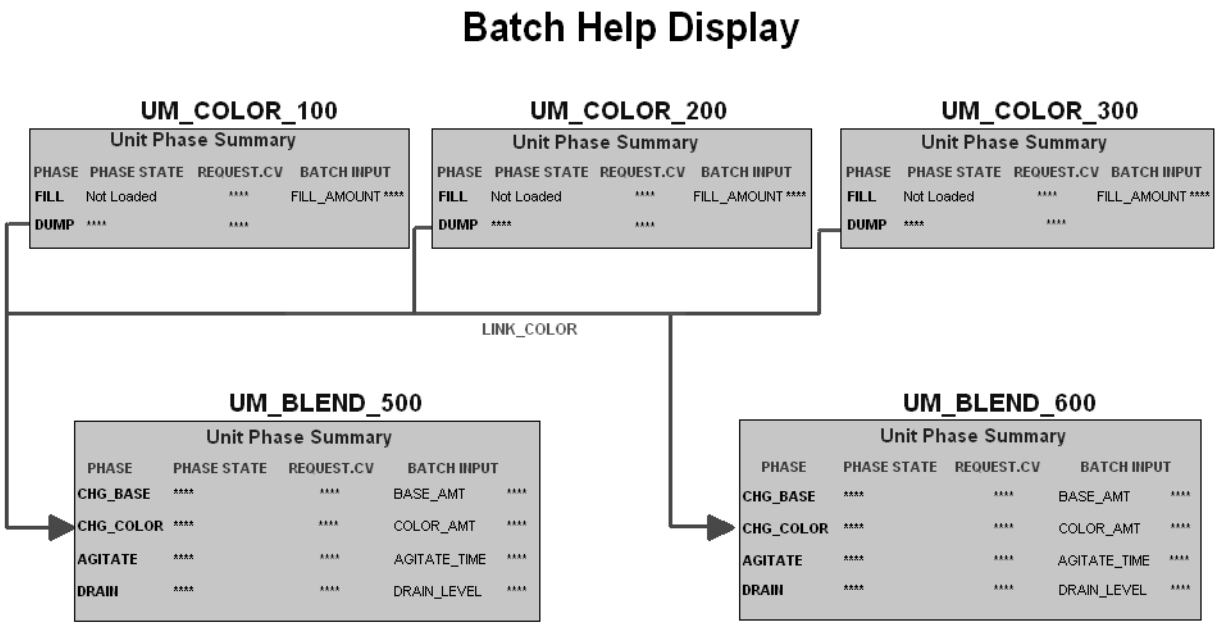
Note After manipulating phases using external control, you will need to unload them so they will return to DeltaV Batch control. This is important because later, when you run recipes that use these unit modules, the modules must be under DeltaV Batch control. To be able to unload a phase, it must be in the Idle state, so remember to issue a Reset command after the phase completes.

Opening the Batch Graphic

To open the Batch graphic and module/phase faceplates

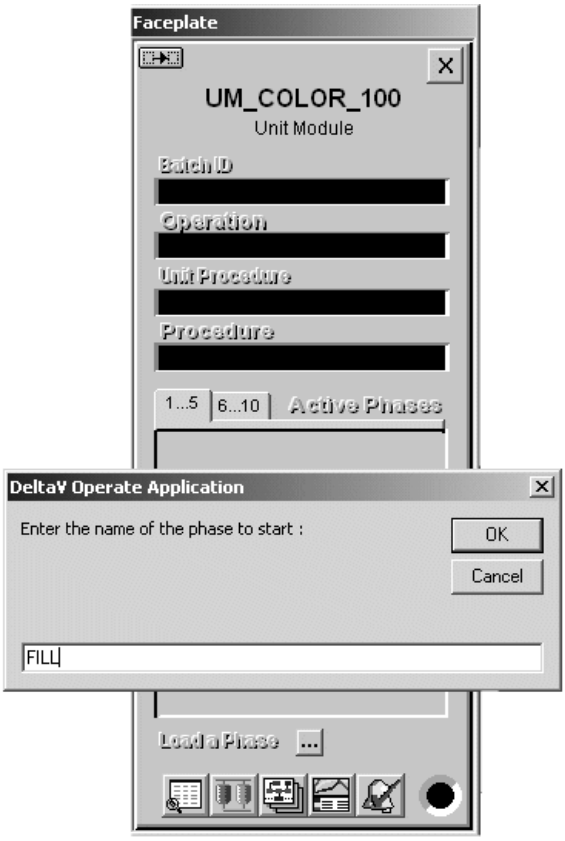
1 Start DeltaV Operate (**Start | DeltaV | Operator | DeltaV Operate**).

2 Open the Batch graphic by clicking the Open folder button  and selecting Batch from the list of graphics available. (Click Skip All for any messages warning about parameters not being configured, since we haven't configured many of the unit modules referenced in this graphic.) The Batch graphic opens.



3 Click the name UM_COLOR_100 to select the unit module and open the faceplate.

4 Click the button next to Load a Phase at the bottom of the faceplate. Enter the phase name FILL (all uppercase) in the dialog box.

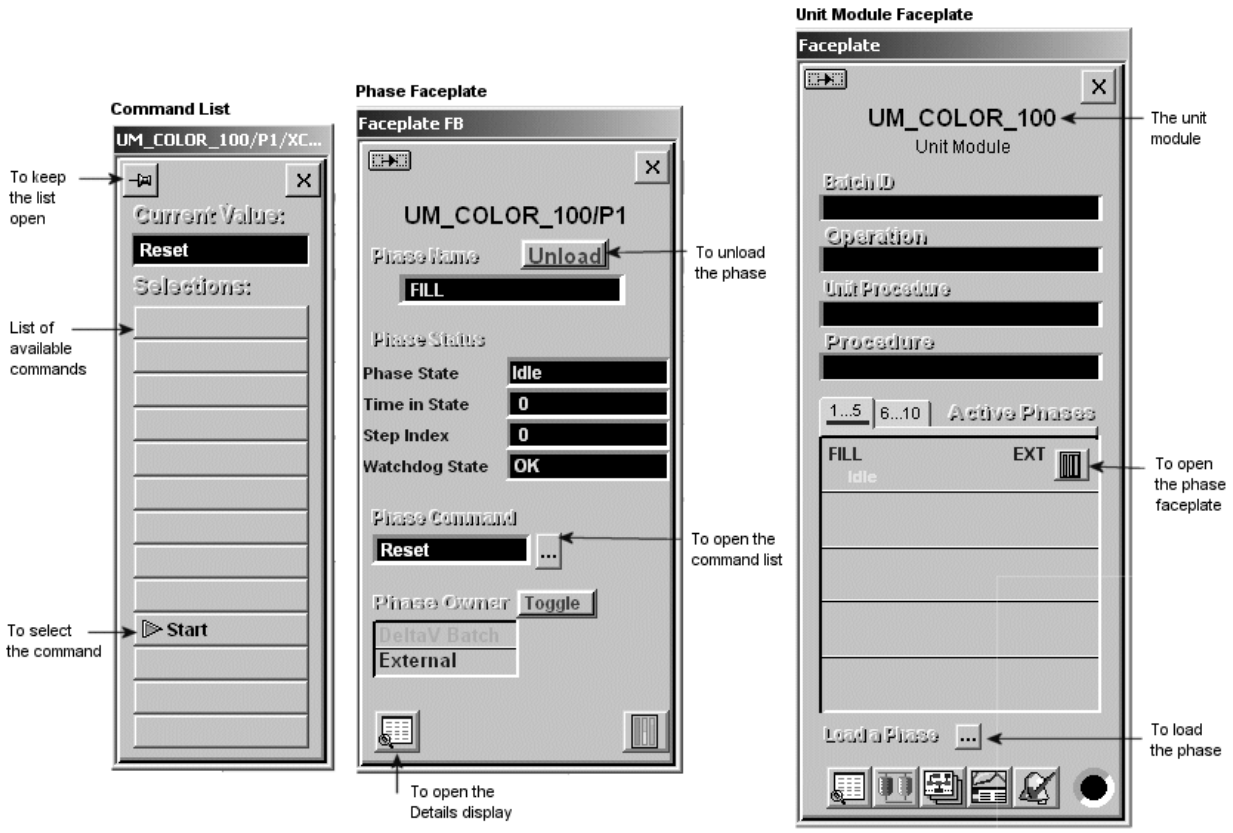


5 When the phase has been loaded, it is listed under Active Phases on the unit module faceplate.

6 Click the faceplate icon next to the phase name FILL to open the phase faceplate.
The phase is listed as UM_COLOR_100/P1 on the faceplate title, meaning it is the first active phase loaded.
Note that the phase owner is listed as External.

7 Click the button next to Phase Command to open the list of phase commands that are valid for the current state.
Click the pushpin button at the top left of the command list to keep the list open.

Note The pushpin button appears when multi-faceplate capability has been enabled in usersettings.grf. When multi-monitor capability has been enabled, a display selector appears to the right of the pushpin.

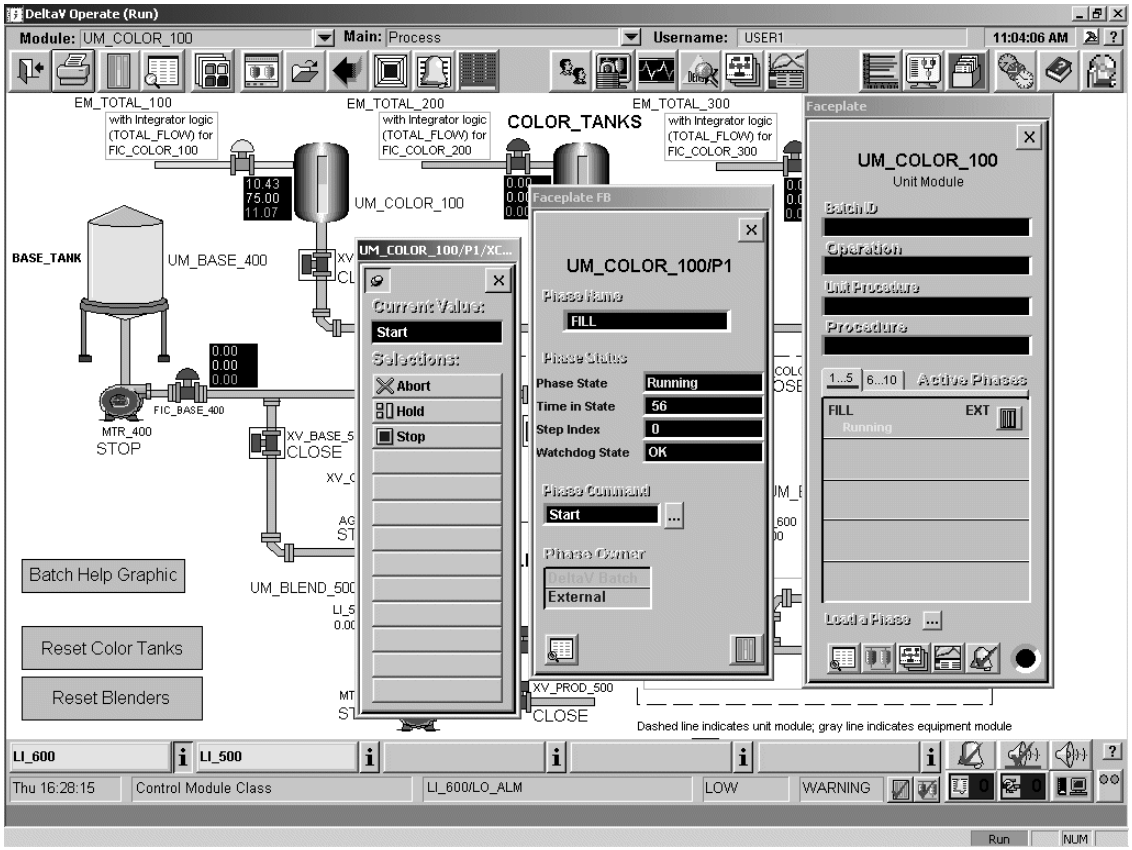


8 Click the Process Graphic button at the top of the graphic to replace the Batch picture with the Process picture. Click Skip All on any warning messages.

Starting and Testing the Phase

To start and test the FILL phase

1 Click Start on the command list to send a Start command to the phase. The inlet valve on the Process graphic changes to open and the tank data link shows the tank filling. The tank will fill to approximately 100 gallons, the amount specified by the batch input parameter FILL_AMOUNT in the FILL phase.



2 Click the FILL_AMOUNT parameter on the Batch picture and change the value to 75 in the Data Entry box.

Resetting the Phase

To reset FILL

1 When the tank finishes filling, the phase automatically goes to Complete.

2 Send a Reset command to place the phase in the Idle state.

3 Unload the phase by clicking the Unload button on the UM_COLOR_100/P1 faceplate.

4 Click the Reset Color Tanks button on the Process graphic to reset the tank.

5 Minimize DeltaV Operate.

You might want to repeat the exercises for testing the FILL phase until you are comfortable with the procedures. Later, in the Verifying Phase Coordination topic, you will use the Batch picture for stepping through equipment arbitration and phase messaging.

AGITATE Phase Logic Configuration

In this chapter you will configure the running, holding, and restarting phase logic for the AGITATE phase class. In the next chapter, you will change the AGITATE phase logic to incorporate failure monitoring.

However, before we start configuring the phase logic for the AGITATE phase, we will explain a feature called pulse action confirmation. Adding a confirm expression to an action lets you simplify your logic when a step contains a number of actions that need to be confirmed before proceeding to the next step.

Adding a Confirm to an Action

You can add a confirm expression to an action that has a pulse qualifier. For instance, you may want to confirm that a valve has closed before going on to the next action or the next step. You can configure the action with a confirm expression, rather than write the expression in a delay condition or transition condition.

After a step (for example, CLOSE_VLVS) that has one or more actions with confirms, you would add a transition or termination with an expression such as:

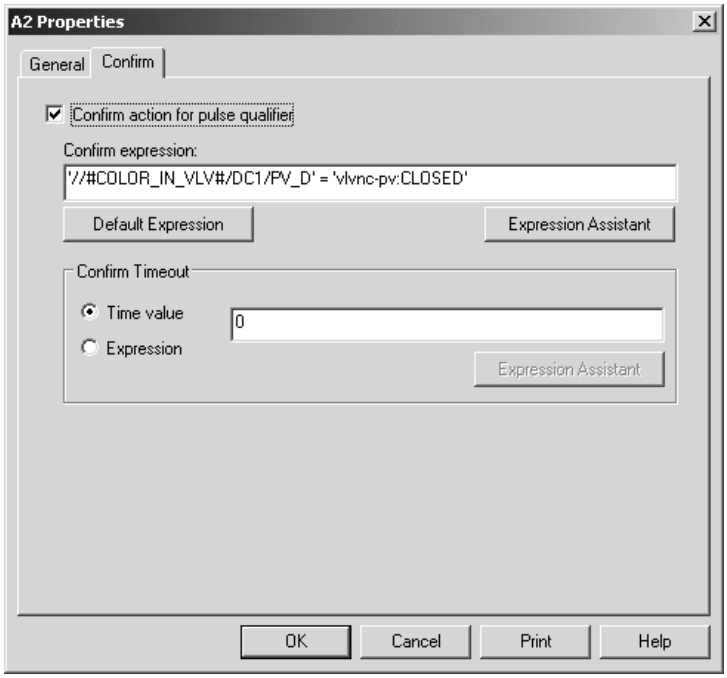
```
('CLOSE_VLVS/PENDING_CONFIRMS.CV' = 0) AND('CLOSE_VLVS/CONFIRM_FAIL.CV' =
FALSE)
```

If there are no pending confirms and no failed confirms, then the step action or actions have completed successfully. If the expression evaluates to TRUE, the SFC will proceed to the next step.

To add a confirm expression, select the action and go to the action properties dialog Confirm tab. Check the box for Confirm action for pulse qualifier and use the Expression Assistant to write the expression. Or, click the Default Expression button to use the default confirm expression.

As an example, the confirm expression for a step action to close an inlet valve could be written as:

```
'//#COLOR_IN_VLV#/DC1/PV_D' = 'vlvnc-pv:CLOSED'
```



A confirm requires either a Timeout value or a Timeout expression. You could either set the Confirm Timeout to a Time value of 5 (meaning that if the valve does not close within 5 seconds, the confirm times out; a timeout value of 0 means there is no time limit on the confirm timeout) or set the Confirm Timeout expression to:

```
'//#COLOR_IN_VLV#/DC1/FAIL' = 1
```

meaning that if the inlet valve module fails before the valve is closed, the confirm times out.

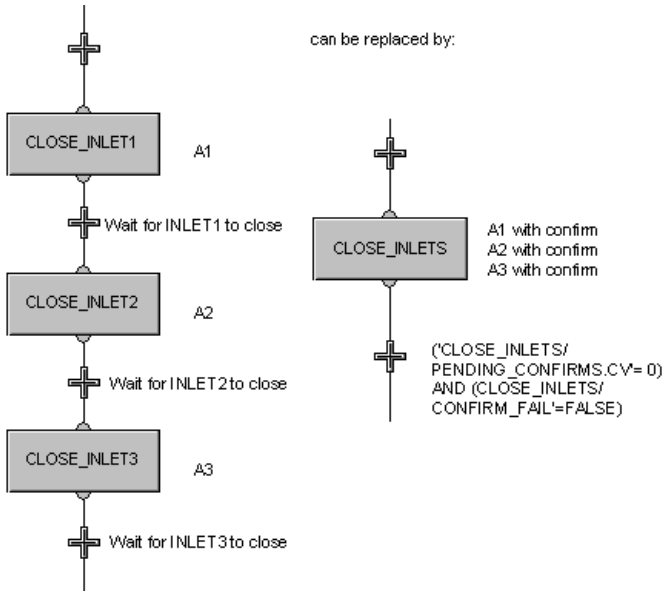
When a step becomes active, the PENDING_CONFIRMS parameter for the step is set equal to the number of actions with confirms in the step. In our example, we have one action with one confirm; therefore, PENDING_CONFIRMS is set to 1. When a confirm either completes or times out, the PENDING_CONFIRMS is decreased by one. When all confirm actions within the step have either completed or timed out, PENDING_CONFIRMS is set to 0.

If a confirm condition times out (the Confirm Timeout time expires or the Confirm Timeout expression becomes TRUE before the Confirm expression becomes TRUE), the FAILED_CONFIRMS parameter is incremented by 1 and the CONFIRM_FAIL parameter is set to TRUE.

The CONFIRM_FAIL parameter is available at the action level, the step level, and the state logic level. It can be used in the failure monitoring logic to send the phase to the holding state, send a message to the operator, or take other appropriate action.

In real world batch applications, adding confirms to actions can greatly simplify configuration of the logic, especially when there are numerous actions that have to be completed and confirmed before going on to the next step.

For example, if there were three inlet valves that had to be closed, the actions could be written in separate steps with confirming transition conditions after each, or the three actions could be written in one step followed by a complex transition statement. More concisely, all three actions could be written in one step with a confirm for each action. Adding the confirms to the actions would simplify the logic, as shown in the following diagram.



Configuring the AGITATE Running Logic - Overview

The phase class AGITATE was created earlier in the BlenderPhases category of the Advanced Definitions Library. (The AGITATE phase class was created with a batch input parameter, AGITATE_TIME, and a batch report parameter, ACTUAL_TIME.) Now you will configure the running logic for this phase class using Control Studio. The next topic shows the sequential function chart for the running logic. The illustration shows the steps, step actions, transitions, and conditions.

Note In an actual Sequential Function Chart (SFC), the actions are not displayed next to the steps. We added these to the figure using the Comment tool. However, you can display conditions next to transitions by selecting Tools | Diagram Preferences and selecting Condition under the Transitions heading.

The general flow of the running logic is outlined below. (Aliases are enclosed in # signs.)

1 Set the blender outlet equipment module, #BLENDER_OUT_1#, to Closed.

2 Close the color inlet valve, #COLOR_IN_VLV#.

3 Close the base inlet valve, #BASE_IN_VLV#.

4 Confirm that the above devices have transitioned to the required states.

5 Start the agitate motor and wait until #AGITATOR# is running. (You will add a new phase parameter, AGIT_FLAG, to the running logic to keep track of whether or not the motor is supposed to be running. This parameter will be important when the restart logic is created.)

6 Wait for the TIME parameter on the WAIT step to be greater than the batch input parameter AGITATE_TIME.

7 Stop #AGITATOR#.

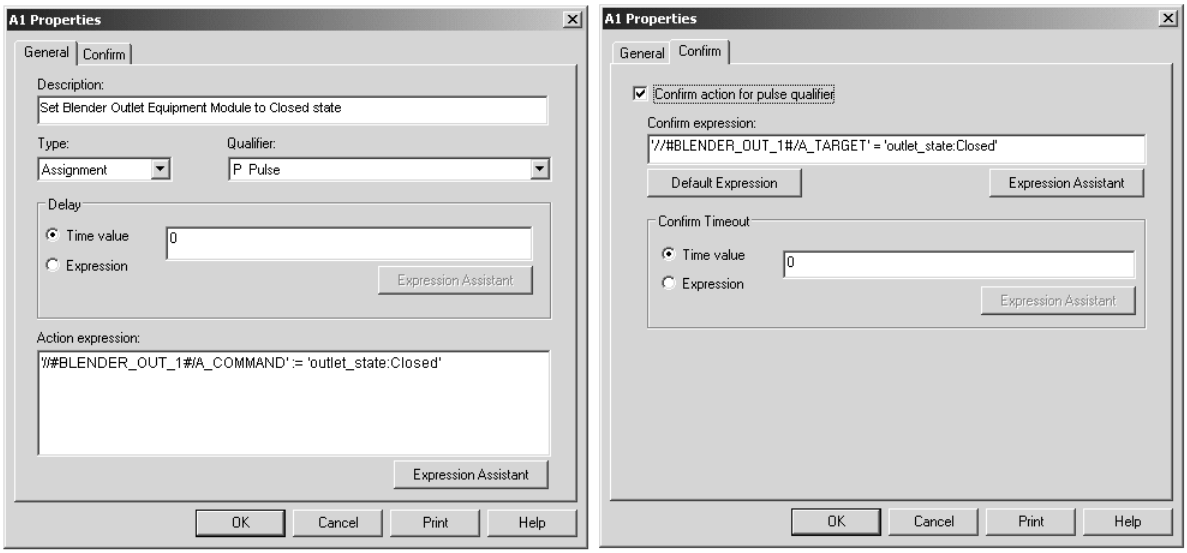
8 Confirm that #AGITATOR# has stopped.

The AGITATE Running Logic - Details

The first step in the SFC for the AGITATE phase running logic includes a number of actions. If we confirmed them all in a single transition statement, it would look like this:

```
('//#BLENDER_OUT_1#/A_TARGET' = '$outlet_state:Closed'
AND '//#COLOR_IN_VLV#/DC1/PV_D' = 'vlvnc-pv:CLOSED'
AND '//#BASE_IN_VLV#/DC1/PV_D' = 'vlvnc-pv:CLOSED')
```

To simplify the logic, each action will have a confirm expression. For example, here are the two pages of the Properties dialog for the first step action, which sets the BLENDER_OUTLET equipment module to a closed state.



Note If you imported the Complete.fhx database, the logic for the AGITATE phase class will not match exactly the figures shown in the next few topics. The phase classes in the database include changes that will be made later in this tutorial to allow us to monitor failure conditions.

Before you configure the running logic, you will add the new phase algorithm parameter, AGIT_FLAG, so that you will be able to browse for it when you create the running logic.

Adding a Parameter to the Running Logic

Normally, you will want to add phase algorithm parameters at the phase class level. However, you can create a parameter for an active state such as the running logic. For the AGITATE running logic you will create a phase algorithm parameter (AGIT_FLAG) that will be used to keep track of whether or not the agitate motor is supposed to be running.

To add a parameter to the AGITATE running logic

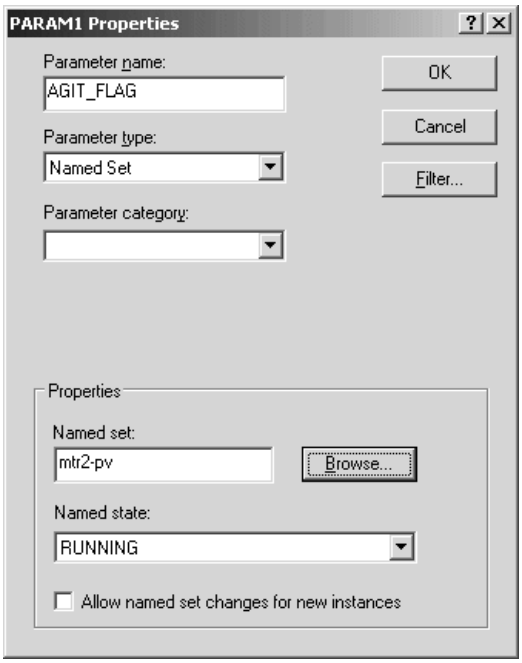
1 Open the AGITATE phase class in Control Studio.

2 Drill down into the running logic block.

3 With nothing selected in the diagram view, point to the parameter view in the lower left corner and right-click.

4 Select Add | Phase algorithm parameter. A Parameter Properties dialog box opens.

5 Name the parameter AGIT_FLAG, specify a parameter type of named set, browse for the named set mtr2-pv, and select a state of RUNNING.



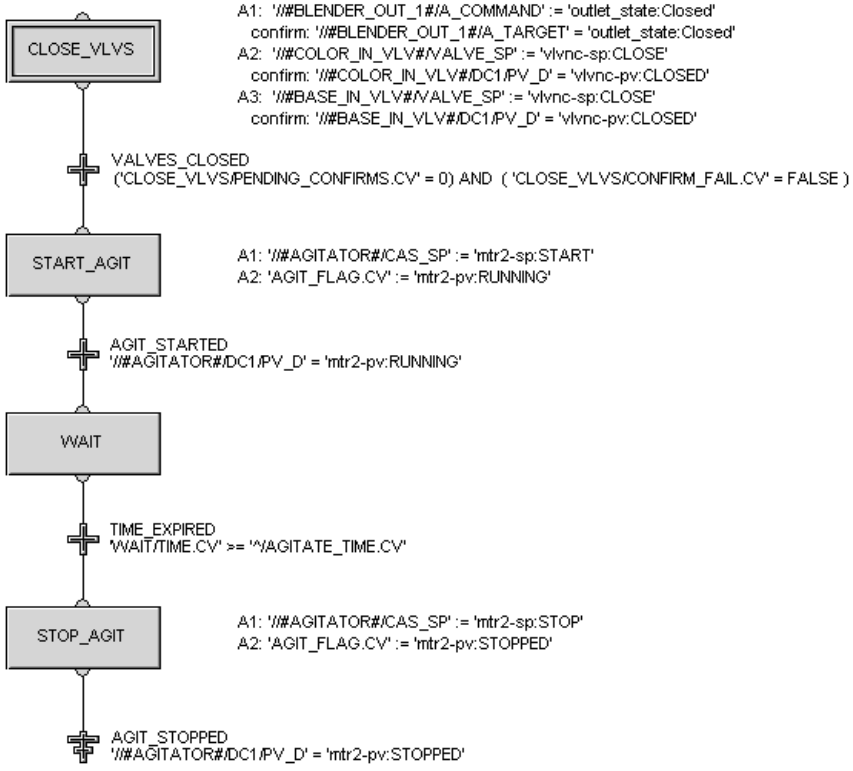
6 Click OK to save this phase algorithm parameter.

7 Click File | Save to save the phase. You can now browse for AGIT_FLAG when you create the step actions, two of which will reference this parameter.

Configuring the AGITATE Running Logic

In Control Studio, configure the running logic for the AGITATE phase class by adding the steps, actions, and conditions as shown in the SFC below. For the actions in the first step, remember to configure corresponding confirm expressions. Also, remember when you are inserting an alias, select the module block rather than the simple alias; the module block allows you to browse to embedded parameters.

In the figure that follows, the transition conditions are shown next to the transitions. The step actions are also shown in the figure but don't actually appear on the diagram that you will create. (The step actions will appear in the action view when a step is selected.)



After creating the running logic for AGITATE, save the phase class.

Assigning the AGITATE Phase to the BLENDER Unit Class

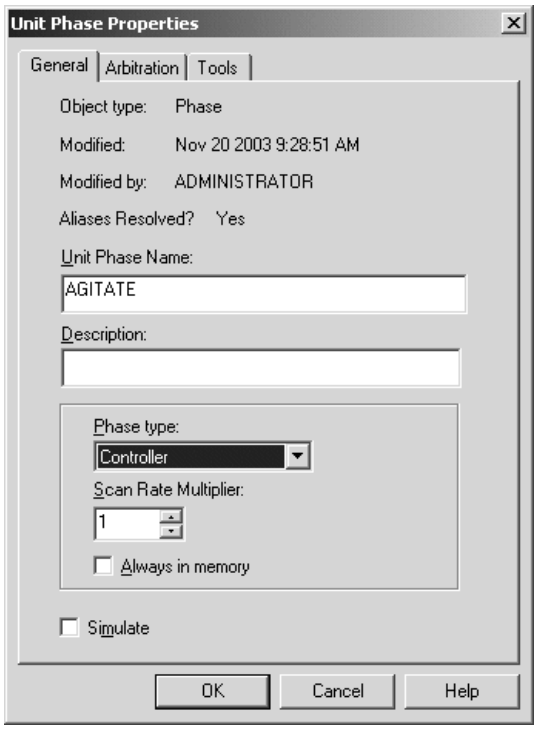
Now that we have the AGITATE phase running logic, we will assign the phase to the unit module class for the blenders. The assignment will automatically be propagated to the BLENDER unit modules we created earlier in the process cell. The assigned phases are called unit phases and are marked by the icon .

To assign AGITATE to the BLENDER unit class

1 In the DeltaV Explorer, assign the phase class AGITATE (under BlenderPhases) to the BLENDER unit class using the drag-and-drop method.

2 Expand the UM_BLEND_500 and UM_BLEND_600 unit modules in the PAINT_BLEND process cell to confirm that the AGITATE phase has been assigned to the unit modules.

3 Select the AGITATE unit phase under UM_BLEND_500 and choose Controller for the phase type on the Unit Phase Properties dialog. Do the same for UM_BLEND_600.



4 Download the node. Do not accept the online values for uploading. Click Select None, close the Upload dialog box, and then proceed with the download.

Testing the AGITATE Running Logic

Verify the running logic by modifying the AGITATE_TIME parameter and then manually running the AGITATE phase for UM_BLEND_500.

To test the AGITATE running logic

1 Open the Batch display in DeltaV Operate.

2 Click UM_BLEND_500 to open the unit module faceplate.

3 Load the AGITATE phase.

4 Click the AGITATE_TIME parameter data link and enter a value of 20.

5 Open the faceplate for the AGITATE phase and the command list.

6 Send a Start command.

7 Go to the Process display.

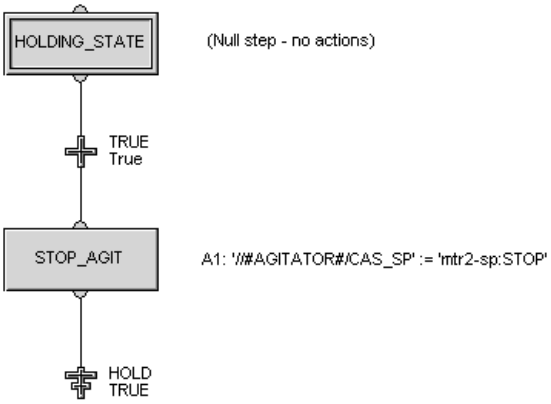
8 When the phase goes to complete, send a Reset command to return the phase to the Idle state.

9 Unload the phase.

Note After testing, the phase must be reset to the Idle state before it can be unloaded. It must then be unloaded so that it will be available later for Batch Executive control.

Configuring the AGITATE Holding Logic

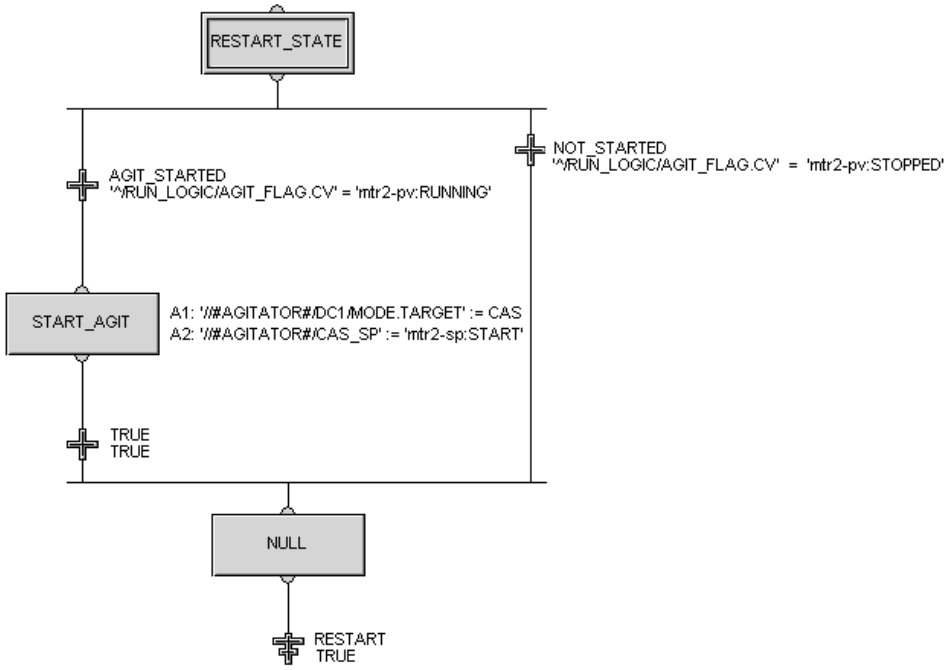
The holding logic for the AGITATE phase is shown in the following diagram. When the AGITATE phase is in the Holding state, we want to stop the agitator. Create the holding logic in Control Studio by adding the steps and transitions as shown and then save the phase.



Configuring the AGITATE Restarting Logic

The restarting logic can be reached only from the held state. The restarting logic, as shown below, has two branches: one for if the motor is supposed to be running and one for if it is not supposed to be running. (The AGIT_FLAG parameter, created earlier, is used to determine whether the motor should be restarted before returning to the running state.) If the agitator was running when the phase went to the Holding state, we will start the agitator.

Create the restarting logic. Save and download UM_BLEND_500. (Do not upload any on-line parameter changes.) After configuring this phase logic, verify the holding and restarting logic, as described in the next topic.



Verifying the AGITATE Holding and Restarting Logic

To verify the holding and restarting logic

1 Download UM_BLEND_500. If asked, do not accept the online values for uploading. Click Select None and then proceed with the download.

2 Go to the Batch picture in DeltaV Operate.

3 Open the UM_BLEND_500 unit module faceplate.

4 Load the AGITATE phase and open the phase faceplate and command list.

5 Go to the Process picture.

6 Send a Start command and wait for AGIT_500 to begin agitating (phase shows Running).

7 Send a Hold command to the AGITATE phase.

8 Verify that the phase goes to the Held state and that AGIT_500 stops.

9 Send a Restart command to the AGITATE phase.

10 Verify that the phase transitions back to the Running state and that AGIT_500 resumes agitating.

11 When the phase completes, reset the phase to Idle and unload it.

Looking Ahead

In the next set of topics we will take a look at failure monitoring for the AGITATE phase. We will need to decide what conditions constitute a failure, how we will monitor those conditions, and what action should be taken if a failure occurs.

Failure Monitoring

Failure monitoring is a key part of any process system. The purpose of the Fail_Monitor composite in a phase is to alert the phase to the presence of a failure condition so that steps can be taken to put the process in a safe state if necessary. The failure monitor in a unit phase is active every time the module is scanned at its configured scan rate.

When a failure condition is detected, the FAIL_INDEX parameter is set, and the phase automatically goes to the holding state, from which it can go to the aborting, held, or stopping state according to the preconfigured State Transition Diagram. (Recall that FAIL_INDEX is one of the default phase logic parameters. A non-zero value indicates a failure.) The configured control logic determines how the phase proceeds. The operator may also intercede if necessary.

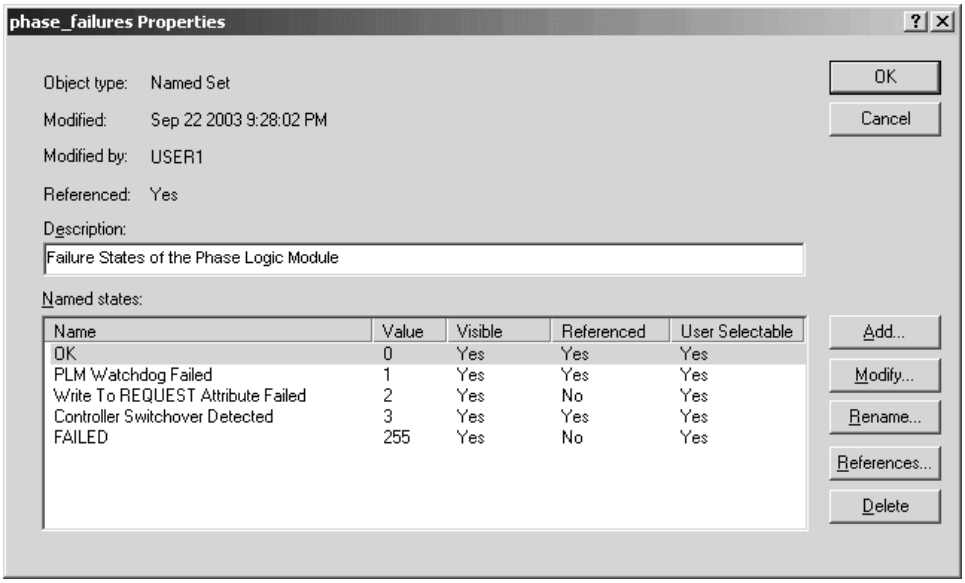
In this chapter, you will incorporate failure monitoring into the AGITATE phase class. Within the Fail_Monitor block in the phase class you will create the logic to evaluate if any user-defined failure conditions are true.

Following are the main tasks you will accomplish in this chapter:

- Add the possible failure states to the named set phase_failures. (Named sets are in the System Configuration | Setup section of the DeltaV Explorer.)
- Configure the Fail_Monitor logic in the AGITATE phase class to monitor for an active failure condition.
- Modify the running logic of the AGITATE phase class to specify which failure conditions should be monitored. (Not all failure conditions need to be monitored all the time.)
- Modify the holding and restarting logic of the AGITATE phase class to include actions that will bring the process to a safe state as well as prepare it for a restart.
- Test the AGITATE failure logic by enabling the monitoring of the phase conditions and individually tripping the conditions.

The Phase_Failures Named Set

The failure monitor for any phase is designed to use the phase_failures named set as a way to display information to the operator when a failure occurs. The different named states in the set describe the conditions that can cause the phase to fail. The phase_failures named set is a standard DeltaV named set, phase_failures; it is predefined with five default states, as shown in the following figure:



Modifying the Phase_Failures Set

For the paint application, we want to add the following failure states for the AGITATE phase:

- Blender inlet valve (from either the base tank or color tank) is open.
- Agitator is stopped (after starting).
- Blender outlet is not closed.

Of course, there could be a number of additional conditions that we would want to add to the list of possible failures, but these will serve to illustrate the concept.

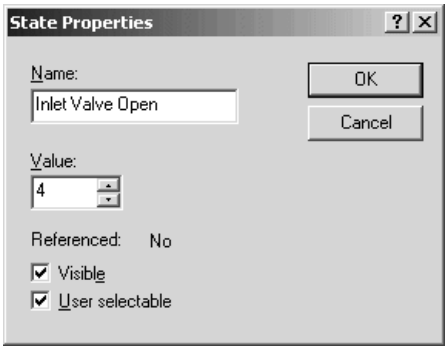
To modify the phase_failures set

1 In the DeltaV Explorer, select Named Sets under System Configuration | Setup.

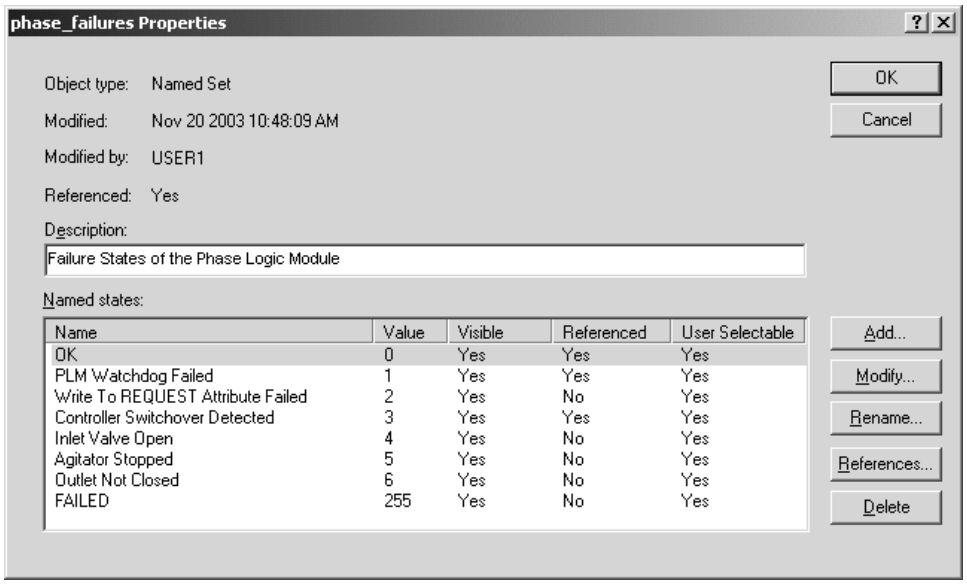
2 Double-click the phase_failures set to open the Properties dialog.

3 Click Add.

4 In the State Properties dialog, type the name of the new failure state, Inlet Valve Open, and accept the assigned value of 4. Make sure that the Visible and User selectable check boxes are selected.



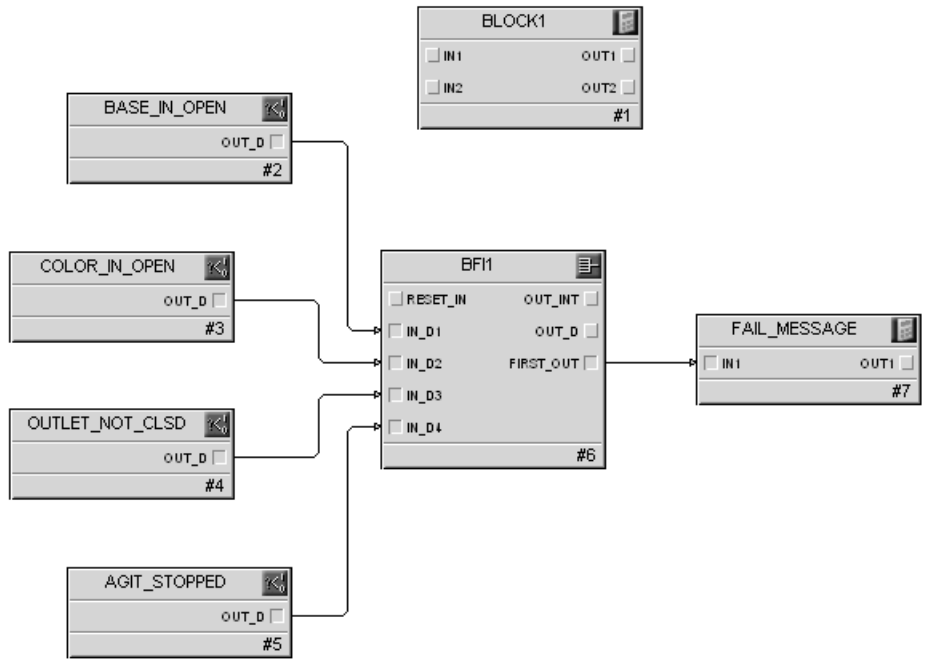
5 Add the failure states Agitator Stopped and Outlet Not Closed. The phase_failures set should now look like the following:



6 Click OK to save the named set.

Creating the Logic to Monitor Phase Conditions - Overview

The diagram below shows the initial Fail_Monitor block, BLOCK1, and the blocks we will add to monitor the failure conditions.



Each of the four condition blocks is used to define the expression for a failure condition. If the condition becomes TRUE, then the OUT_D parameter of that block is set to TRUE as well. By wiring all of the OUT_D parameters into the Boolean Fan In block, we can watch for any failure condition to become active. The Boolean Fan In block makes three decisions that are extremely useful in monitoring for failure conditions. It tells us if any condition is active (OUT_D), it identifies which conditions are active (OUT_INT), and it identifies which condition became active first (FIRST_OUT).

Using the FIRST_OUT parameter and the phase_failures named set, we can display the appropriate message to the operator to describe why the phase has failed. This is done by determining which message to display (in the FAIL_MESSAGE calc/logic block) and writing the correct message to the FAIL_INDEX parameter (in the BLOCK1 calc/logic block).

The FAIL_INDEX parameter is the heart of the failure monitoring system. All phases are designed such that if the FAIL_INDEX parameter for that phase is anything other than 0, then the phase automatically transitions from the running logic to the holding logic. Additionally, the value of the FAIL_INDEX parameter is displayed both on the DeltaV Batch Operator Interface as well as the detail display on the unit phase faceplate as the Failure State message.

We will add four condition blocks that feed into a Boolean Fan In (BFI) block. When a condition is tripped, the following things happen:

- The failure is detected and sent to the BFI block. The value of the FIRST_OUT parameter of the BFI is sent to the Calc/Logic block named FAIL_MESSAGE.
- The FAIL_MESSAGE block converts the FIRST_OUT value to one of the failures defined in the phase_failures named set.
- The FAIL_MESSAGE parameter OUT1 is used to set the FAIL_INDEX parameter in the phase.
- When the FAIL_INDEX parameter is non-zero, the phase goes to the holding state and the appropriate phase_failures named state string is displayed in the DeltaV Batch Operator Interface (as well as in the details display on the unit phase faceplate).

Creating the Logic to Monitor Phase Conditions - Details

To create the failure monitor logic

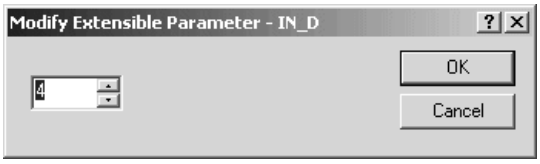
1 Open the AGITATE phase class in Control Studio.

2 Double-click the Fail_Monitor to open it. It opens with one CALC block, named BLOCK1.

3 Add four Condition blocks, one Boolean Fan In block, and one Calc/Logic block to the diagram, as shown in the previous topic. (Look on the Logical and Analog Control palettes for these items.)

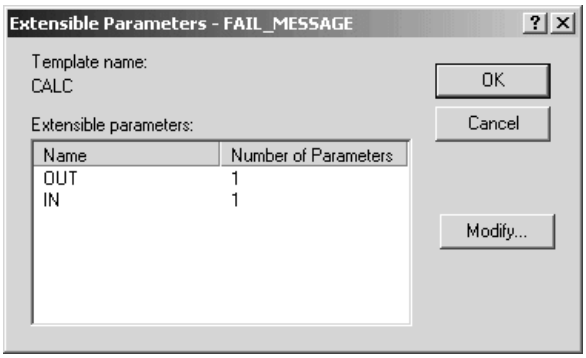
4 Rename the blocks as shown on the diagram. (You can remove the display of the block template names by selecting Tools | Diagram Preferences and removing the check mark next to Show | Template Names.)

5 To change the number of inputs to the BFI1 block, select the block, select Extensible Parameters from the context menu, and change the number of connectors to 4.

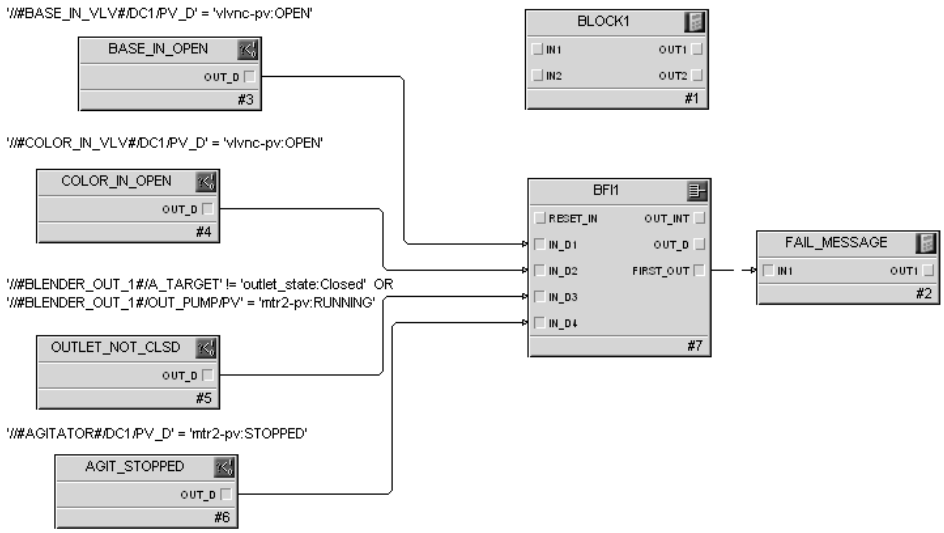


Hint If you would like more information about the BFI block, right-click the block and select What's this? from the menu. For a more detailed explanation, click the Books Online icon.

6 To change the number of outputs and inputs on the FAIL_MESSAGE block to 1 each, select Extensible Parameters from the context menu, double-click the OUT (then the IN parameter), and change the number of connectors to 1.)



7 Connect the blocks as shown in the following figure.



8 For each condition block, select Expression from the context menu and modify the expression to check for the failure conditions. (We have shown the expressions on the diagram as a configuration aid.)

9 Set the DISABLE parameter for each condition block equal to 1. (This initially disables the conditions so that the expressions are not monitored. During execution of the phase, we will selectively enable any condition that we want to monitor.) To disable a condition, display all parameters in the Parameter View by selecting all the filters (all boxes selected). Click one of the Condition blocks (for example, BASE_IN_OPEN). Double-click the Disable parameter and change the value to 1. Continue for all the Condition blocks.

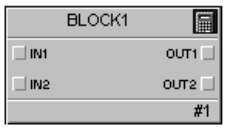
10 Create the following expression for the FAIL_MESSAGE calculation block:

```
FAIL_MOD  :=  IN1;
OUT1 :=  0;
IF (FAIL_MOD = 1) THEN
   OUT1  :=  'phase_failures:Inlet Valve Open';
ELSE
   IF (FAIL_MOD = 2) THEN
      OUT1  :=  'phase_failures:Inlet Valve Open';
    ELSE
       IF (FAIL_MOD = 4) THEN
          OUT1  := 'phase_failures:Outlet Not Closed';
       ELSE
          IF (FAIL_MOD = 8) THEN
             OUT1  :=  'phase_failures:Agitator Stopped';
           ENDIF;
        ENDIF;
     ENDIF;
 ENDIF
```

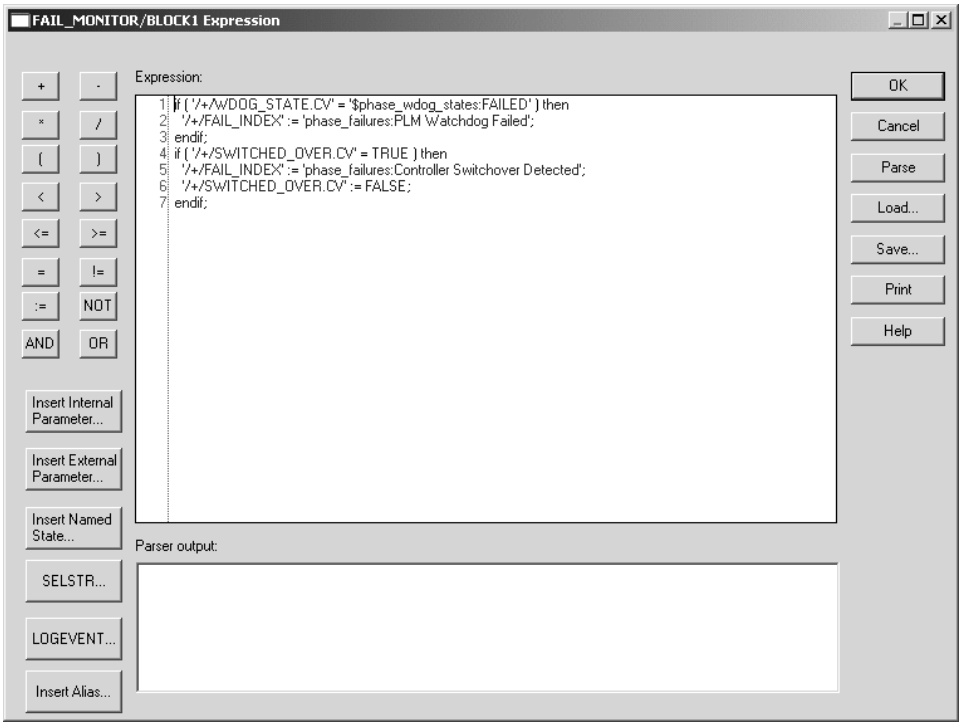
This expression determines which failure message should be displayed to the operator based on the first failure condition that was detected by the Boolean Fan In block. The failure message will later be written to the phase's FAIL_INDEX parameter.

Modifying BLOCK1 of the Failure Monitor

The Failure Monitor started out as a single Calc/Logic block named BLOCK1. Click this block and select Expression from the context menu.



The initial failure logic for BLOCK1 in the AGITATE phase is shown in the following figure. (This is the default failure logic for all phase classes.)



The expression checks the state of the Watchdog parameter and, if it is Failed, sets the FAIL_INDEX. It also checks to see if the primary controller has switched to the backup. If neither condition is detected, the FAIL_INDEX remains 0 (that is, no failure).

Recall that when the FAIL_INDEX changes from 0 to any other value, the phase abandons the running logic and begins to execute the holding logic.

Now we will modify the expression by adding an IF/THEN statement to check the OUT_D parameter of the BFI block. If OUT_D is true (1), then a failure condition has been detected and we will need to copy FAIL_MESSAGE/OUT1 to the FAIL_INDEX parameter. (The value of FAIL_MESSAGE/OUT1 is based on the first failure condition that is detected by the BFI block.)

We will also add a check to each IF/THEN statement to see if a failure has already been detected. As long as the FAIL_INDEX is 0, we can check for any new failure condition. Adding this logic will protect against overwriting an active failure condition message.

To modify the expression for the Monitor block

1 Select BLOCK1 and change the name to MONITOR.

2 Right-click and select Expression from the context menu.

3 Modify the text of the expression, using the Insert Internal Parameter and Insert Named State buttons as necessary to match the logic shown below:

```
IF (( '/+/WDOG_STATE.CV' = '$phase_wdog_states:FAILED' )  AND
     ('/+/FAIL_INDEX.CV' = 0)) THEN
      '/+/FAIL_INDEX' := 'phase_failures:PLM Watchdog Failed';
ELSE;
  IF (( '/+/SWITCHED_OVER.CV' = TRUE )  AND
       ('/+/FAIL_INDEX.CV' = 0)) THEN
        '/+/FAIL_INDEX' := 'phase_failures:Controller Switchover Detected';
        '/+/SWITCHED_OVER.CV' := FALSE;
  ELSE;
     IF (('^/BFI1/OUT_D.CV' = 1)  AND ( '/+/FAIL_INDEX.CV' = 0 )) THEN
         '/+/FAIL_INDEX' := '^/FAIL_MESSAGE/OUT1.CV';
     ENDIF;
  ENDIF;
ENDIF;
```

4 Click Parse to check the expression and make any corrections necessary.

5 Save the AGITATE phase class.

6 Download the controller or workstation so that changes to the AGITATE phase logic as well as changes to the phase_failures named set are sent to the node. Do not accept the online values for uploading.

Testing the Phase Failure Conditions

Now we will verify that when a condition is tripped, it is detected by the Fail Monitor. To create a failure condition, we will open the XV_BASE_500 valve while the agitator is running. To do this, we will first put the XV_BASE_500 module in AUTO mode to allow us to open the valve.

To verify the AGITATE phase failure conditions

1 Open the Batch picture in DeltaV Operate.

2 Open the faceplate for the unit module UM_BLEND_500, load the AGITATE phase, open the AGITATE faceplate, and open the command list.

3 In Control Studio, open the UM_BLEND_500 unit module and click View | Online.

4 Drill down into the AGITATE | FAIL_MONITOR block.

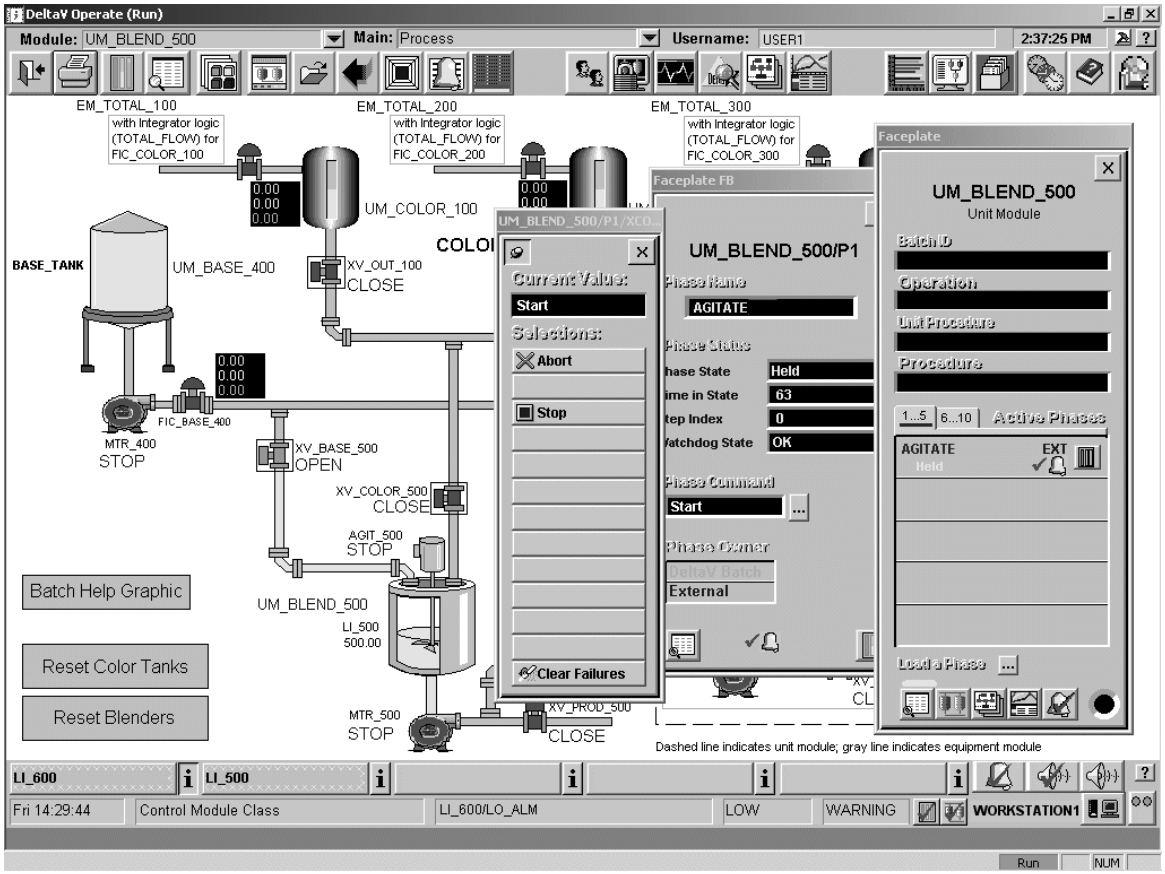
5 Enable monitoring for BASE_IN_OPEN by changing its DISABLE parameter to 0.

6 Go to the Process picture in DeltaV Operate.

7 Click the tag XV_BASE_500 to open the faceplate for the module. Change the mode to AUTO.

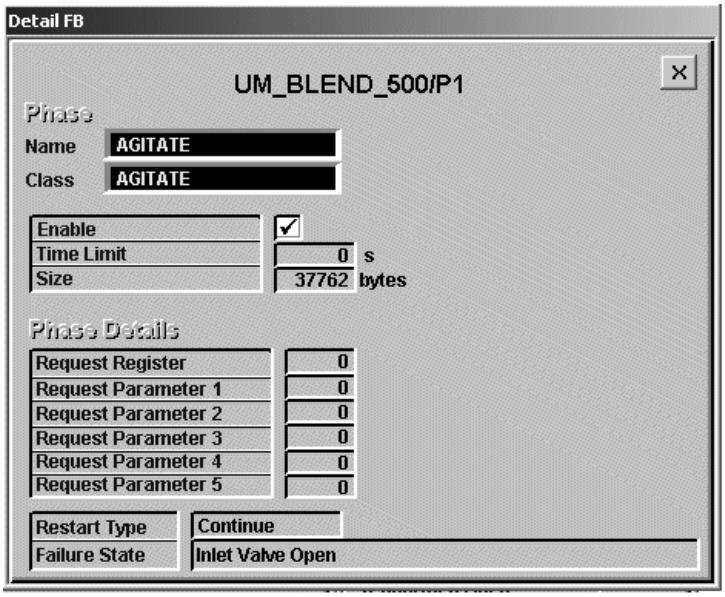
8 Start the AGITATE phase.

9 After AGITATE has started, open XV_BASE_500 by clicking the word CLOSE next to the XV_BASE_500 valve icon and clicking the OPEN button on the dialog box.

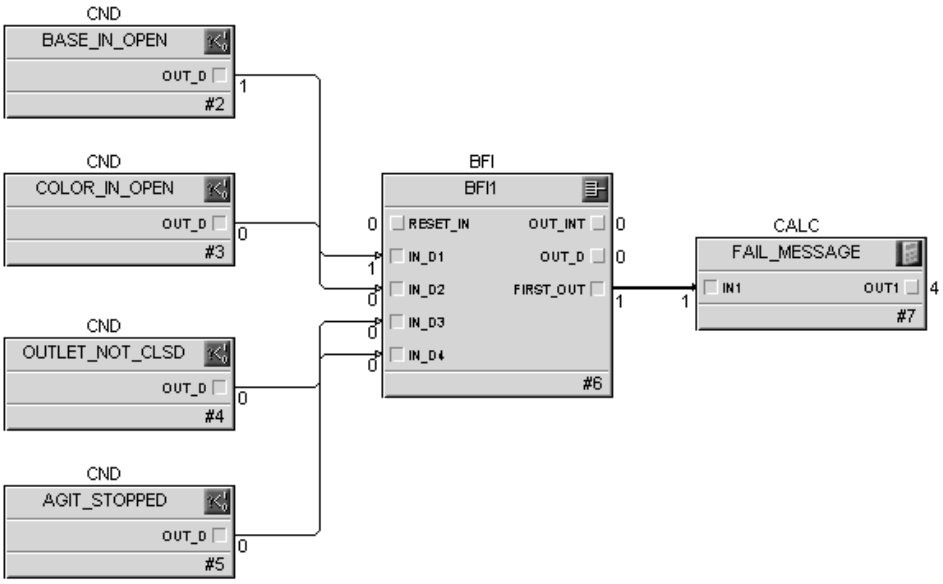


The AGITATE phase should go through the holding state to the held state.

10 On the faceplate for UM_BLEND_500/P1, click the Details button  at the bottom of the faceplate to bring up the detail display. The Failure State at the bottom of the display should read **Inlet Valve Open**.



11 Return to Control Studio. The OUT_D parameter on the BASE_IN_OPEN block temporarily changes to 1 and is sent to the BFI block's IN_D1 parameter. This value passes through the BFI to the FAIL_MESSAGE block using the FIRST_OUT parameter. The OUT1 parameter on the FAIL_MESSAGE block changes to 4, which reflects the numeric value associated with the named state Inlet Valve Open in the phase_failures named set.



12 To test another condition, you need to reset XV_BASE_500 to Closed in DeltaV Operate. Clear the failure by clicking Clear Failures at the bottom of the command list. Then, click the STOP and RESET commands. Remember to enable failure monitoring for the condition block you want to test by setting its DISABLE parameter to 0.

Note When you test the condition, AGIT_STOPPED, the phase should fail immediately because the agitator will not be running when the phase starts.

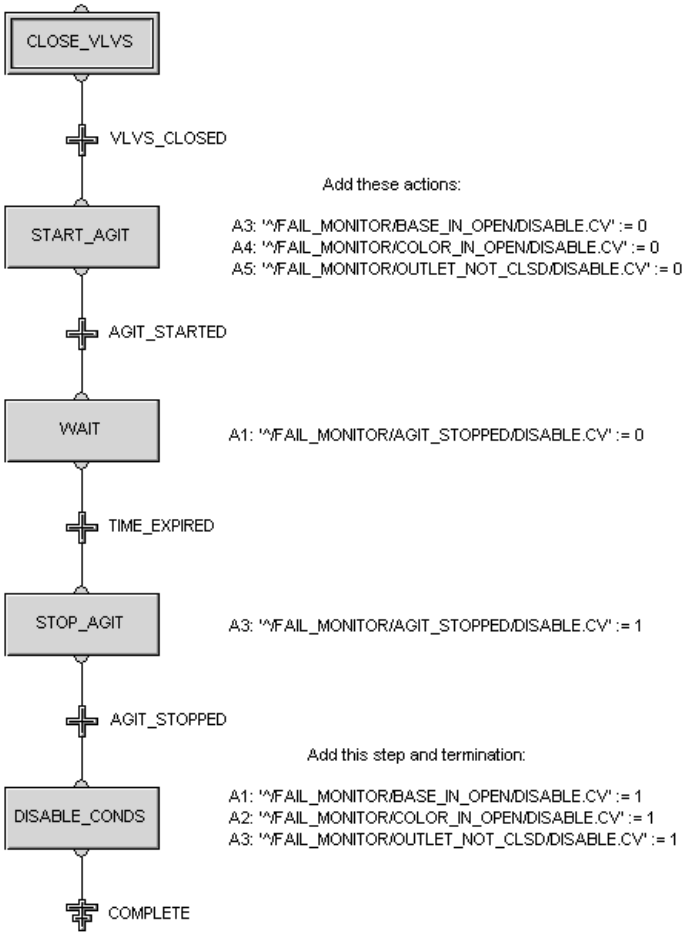
13 When you are finished testing, return the mode for all the modules to CAS, unload the AGITATE phase, and close the faceplates.

Note If you forget to unload a phase, it will remain in external (manual) mode. Later, when you try to run recipes, you will get errors about the phase being under external control.

Modifying the AGITATE Phase Running Logic

Now that you have configured the failure monitoring logic, you can modify the running, holding, and restarting logic of the AGITATE phase class to incorporate monitoring of the phase conditions. Save the phase class after modifying each of the logic states.

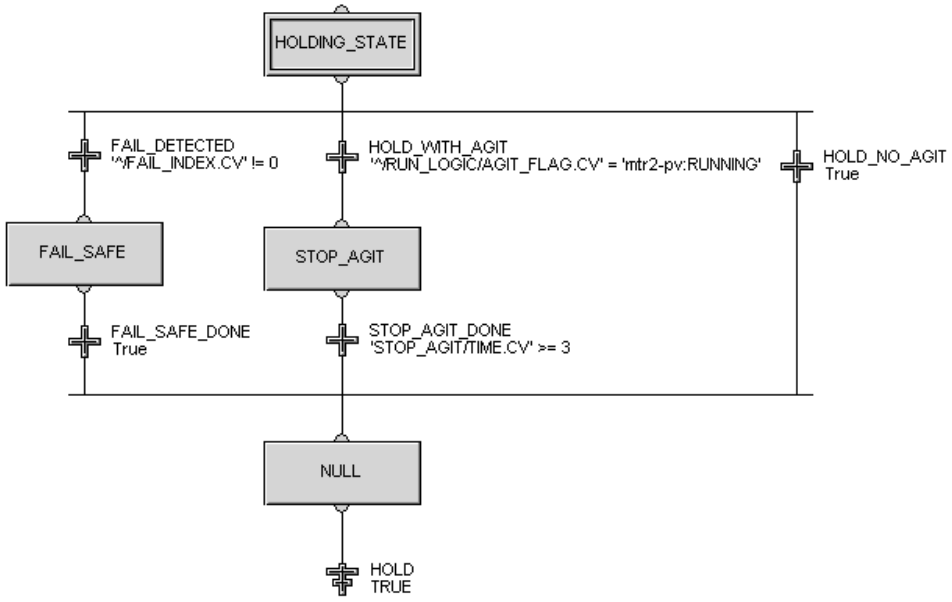
The logic diagram for the running logic is shown below. Modify the running logic by adding the last step (DISABLE_PH_COND) and termination (COMPLETE) and by adding actions to the other steps, as shown in the following figure. Notice that we don't enable monitoring of the agitate motor until after agitation has started. Otherwise, the Fail_Monitor would start out with a true failure condition.



Modifying the AGITATE Phase Holding Logic

The revised holding logic for the AGITATE phase class is shown in the following SFC diagram. Modify the holding logic to branch based on the value of the FAIL_INDEX parameter. If a failure is detected, go to the FAIL_SAFE actions, which include closing all the valves and stopping the outlet pump and agitate motor. If a fail is not detected, branch based on the value of the AGIT_FLAG parameter. (If the AGIT_FLAG is set to Running, then stop the agitate motor.)

Add the steps and actions as shown in the table in the next topic. Add the transitions as shown in the diagram below.



Changes to the AGITATE Holding Logic

The FAIL_SAFE step includes several actions to put the valves and motor in CAS mode before closing. When configuring these actions in the expression editor, you need to type in ".TARGET" as part of the expression.

Steps	Action Text	Qualifier
HOLDING_STATE	A1: ' ^FAIL_MONITOR/AGIT_STOPPED/DISABLE.CV' := 1	
	A2: ' ^FAIL_MONITOR/BASE_IN_OPEN /DISABLE.CV' := 1	
	A3: ' ^FAIL_MONITOR/COLOR_IN_OPEN /DISABLE.CV' := 1	
	A4: ' ^FAIL_MONITOR/OUTLET_NOT_CLSD /DISABLE.CV' := 1	
FAIL_SAFE	A1: '//#BLENDER_OUT_1#/A_COMMAND' := 'outlet_state:Closed'	
	A2: '//#COLOR_IN_VLV#/DC1/MODE.TARGET' := CAS	
	A3: '//#COLOR_IN_VLV#/VALVE_SP' := 'vlvnc-sp: CLOSE'	
	A4: '//#BASE_IN_VLV#/DC1/MODE.TARGET' := CAS	
	A5: '//#BASE_IN_VLV#/VALVE_SP' := 'vlvnc-sp: CLOSE'	
	A6: '//#BLENDER_OUT_1#/OUT_PUMP/DC_MODE' := CAS	
	A7: '//#BLENDER_OUT_1#/OUT_PUMP/CAS_SP' := 'mtr2-sp:STOP'	
	A8 '//#AGITATOR#/CAS_SP' := 'mtr2-sp : STOP'	
STOP_AGIT	A1: '//#AGITATOR#/CAS_SP' := 'mtr2-sp : STOP'	Delay=2

Modifying the AGITATE Phase Restarting Logic

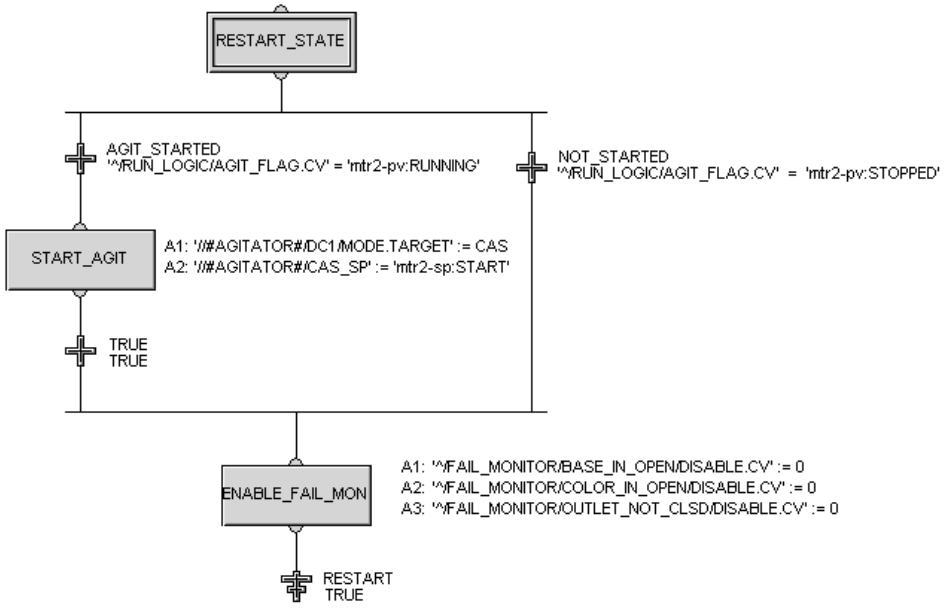
The revised restarting logic is shown in the following figure.

To modify the restarting logic

1 Change the restarting logic according to the figure below.

2 Save the AGITATE phase.

3 Download. Do not select any on-line changes.



Verifying Failure Monitoring for AGITATE

To verify the failure monitoring for AGITATE, follow the steps below. Since the AGITATE_TIME parameter is set to only 30 seconds, you may want to increase this a little to allow yourself more time to trip the failure condition before the phase goes to complete.

1 In Control Studio, open the UM_BLEND_500 unit module and click View | Online. Drill down into the AGITATE | FAIL_MONITOR block.

2 Open the Batch picture in DeltaV Operate.

3 Load and start the AGITATE phase for the unit module UM_BLEND_500.

4 Trip a failure condition by opening XV_BASE_500 (in the Process picture). (XV_BASE_500 must be in Auto mode.)

5 In Control Studio, verify that the FAIL_INDEX parameter is set and that AGITATE goes to Held. (XV_BASE_500 closes as part of the Hold logic FAIL_SAFE instructions.)

6 Clear the failure by clicking the Clear Failures button on the command list.

7 Send a Restart command.

8 Verify that the AGITATE phase transitions from the Restarting state to the Running state.

9 When the AGITATE phase goes to Complete, send a Reset command.

10 You can verify other failure conditions using the same general procedure. When you are finished, reset the phase to Idle.

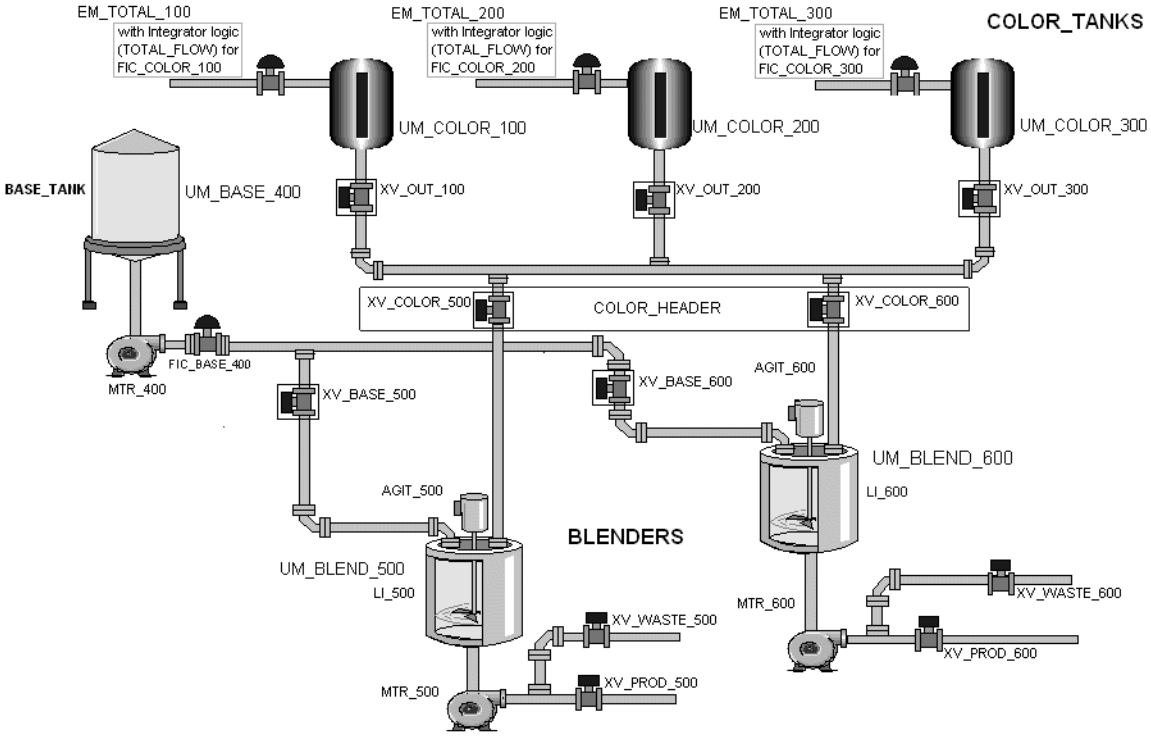
11 Unload the AGITATE phase.

Equipment Arbitration and Phase Coordination

All equipment configured in the DeltaV Explorer is assigned a unique equipment ID. The DeltaV system uses this ID to allocate equipment and service any requests for ownership. The term **arbitration** refers to how something with an equipment ID (usually an equipment unit) is owned and released through different stages of a recipe. You can also assign equipment IDs to control modules and unit modules to allow them to participate in arbitration.

In the paint tutorial example, both blenders are filled from the same color tanks, but only one should be filled at a time so that two colors are not in the same pipe. The modules for the inlet valves, XV_COLOR_500 and XV_COLOR_600, will be tied together so that only one can be used at a time. To do this, you will create a control module named COLOR_HEADER that will list the two modules as Equipment Needed. Only the phase that has acquired the COLOR_HEADER resource will be able to open and close the valves XV_COLOR_500 and XV_COLOR_600.

Before creating the COLOR_HEADER module, you will modify both XV_COLOR_500 and XV_COLOR_600 to allow them to participate in equipment arbitration. This makes it possible to add them to the Equipment Needed list for the COLOR_HEADER module.



After you have set up the modules for equipment arbitration, you will create the phase coordination through a phase class for charging color into UM_BLEND_500. The phase logic will:

- Request (through a phase REQUEST) acquisition of the COLOR_HEADER resource
- Open #COLOR_IN_VLV#, the alias used to identify valve XV_COLOR_500 or XV_COLOR_600
- Release the COLOR_HEADER (through another REQUEST message) when the blender is finished filling

From the time the COLOR_HEADER is acquired until it is released, the second color input valve can only be manipulated by the owner of the COLOR_HEADER, which in this case is the phase logic running on

UM_BLEND_500. (An exception is that the valve may also be opened or closed through direct intervention by an operator with sufficient privileges).

You will learn more about REQUEST parameters later, in the Coordinating the Dump and Charge Phases topic.

Modifying Modules to Allow Equipment Arbitration

First, you will modify the properties of the control modules (XV_COLOR_500 and XV_COLOR_600) that control the color inlet valves to allow them to participate in equipment arbitration.

To modify the module properties

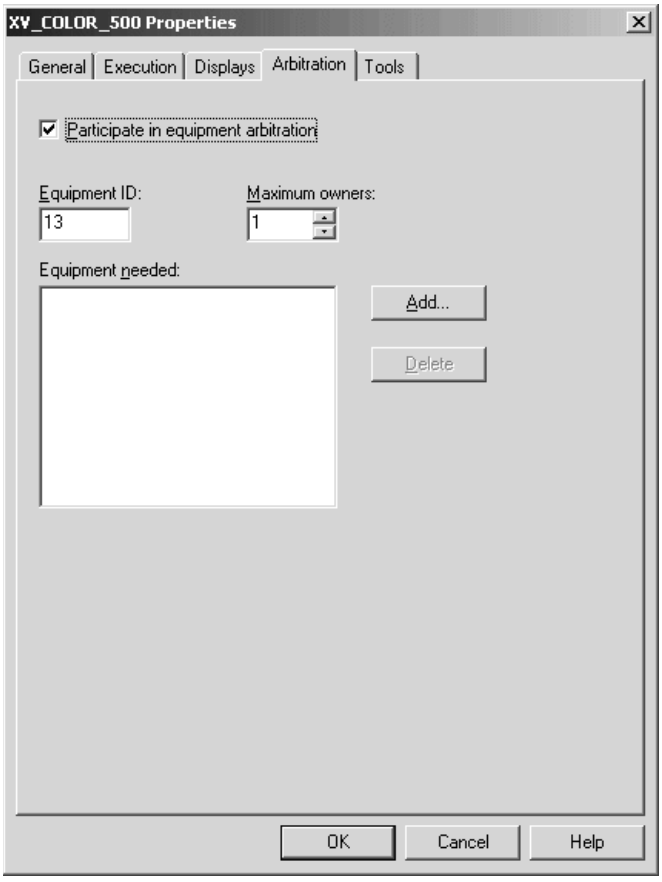
1 In the DeltaV Explorer, select the module XV_COLOR_500 under System Configuration | Control Strategies | EXTERIOR_PAINT | PAINT_BLEND | UM_BLEND_500.

2 Right-click and select Properties from the context menu.

3 On the Arbitration tab of the Properties dialog box, select the check box Participate in equipment arbitration. The Equipment ID is automatically assigned the next available number.

> **Note** The Equipment IDs in the examples may not match those in your system. Whenever they are referenced in the phase logic, you need to substitute the ID numbers used in your system.

4 Leave the Maximum Owners value at 1 and click OK.



5 Repeat this procedure for module XV_COLOR_600 (in UM_BLEND_600).

Creating the COLOR_HEADER Module

Now, you will create a control module called COLOR_HEADER in the PAINT_BLEND process cell and modify its properties to add XV_COLOR_500 and XV_COLOR_600 to its Equipment Needed list.

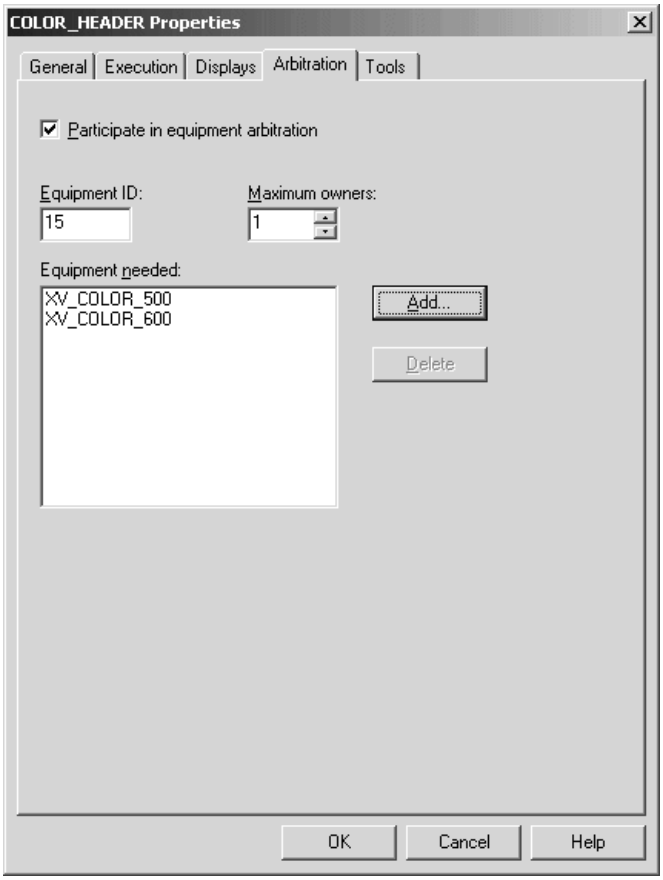
To create the COLOR_HEADER control module

1 In DeltaV Explorer, right-click the PAINT_BLEND process cell and select New |Control Module from the context menu.

2 Name the new control module COLOR_HEADER. (Leave the Algorithm type as Function Block Diagram.)

3 Open the COLOR_HEADER Properties dialog.

4 On the Arbitration tab of the Properties dialog box, select the check box Participate in equipment arbitration. (Note the Equipment ID, which is 15 in the example below. You may have a different number, which you will need to substitute for 15. This ID will be used later in requesting the resource.)



5 In the Equipment Needed list, add XV_COLOR_500 by clicking Add and browsing in EXTERIOR_PAINT | PAINT_BLEND | UM_BLEND_500 for the control module.

6 Add XV_COLOR_600 to the Equipment Needed list.

7 On the Tools tab, assign the module to the node.

8 Click OK to close the Properties dialog.

9 Download.

Coordinating the Dump and Charge Color Phases

To transfer color from UM_COLOR_100 to UM_BLEND_500, the Batch Executive coordinates communications between unit phases (DUMP and CHG_COLOR) running on the unit modules (UM_COLOR_100 and UM_BLEND_500, respectively). It does this by means of REQUEST parameters that are used in the running logic of the DUMP and CHG_COLOR phase classes.

The phase request codes we will use in the example are listed in the table below. (For a complete list of the phase request codes used in DeltaV Batch, refer to the Phase Request Codes topic in the *Batch Reference* manual.) As you can see, 40nn and 42nn are paired; nn is the resource's equipment ID and will correspond to the number assigned earlier to the COLOR_HEADER module. The codes 52nn and 55nn are also paired, with nn being a user-defined message ID.

Request Parameter	Request Type	where nn is
40nn	Acquire a single resource	the equipment ID of the resource
42nn	Release resource	the equipment ID of the resource
52nn	Send message to corresponding 55nn message	message ID
55nn	Wait for a corresponding 52nn message	message ID

Coordinating the Dump and Charge Phases - Diagram

The dashed lines in the following diagram show how messages from the DUMP and CHG_COLOR phases are coordinated by the Batch Executive. The request messages are sent from the phases to the Batch Executive (using their respective REQUEST parameters) at various places in the phase logic. The requests are a means of acquiring resources or coordinating activities between one or more phases (in this case, the DUMP and CHG_COLOR phases). For phase coordination there are two types of requests we will use: Send message to another phase (52nn) and Wait for a message (55nn).

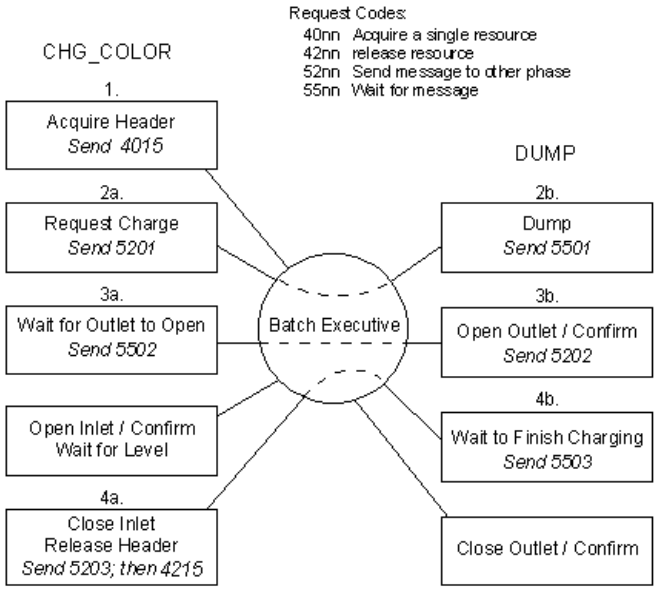
Once a request is received, the Batch Executive attempts to carry out the request. For any Wait message request that is received, there must be a corresponding Send message request before the Batch Executive can respond. For example, if one phase sends a request message indicating that it is waiting to transfer material (Wait message), then the Batch Executive will look for another phase to send a request message indicating that it is ready to transfer material (Send message).

After the Batch Executive receives both requests, it clears the REQUEST parameters for both phases by setting them to a value of 0. This indicates to the phases that their respective requests have been successfully processed.

The other type of request message we will use in this example is the Acquire a single resource request. For this type of request, the Batch Executive checks to see if the resource being requested is available. If it is, it acquires the resource for the phase. If the resource is not available, the Batch Executive will wait until the resource becomes available and then acquire it. Once it acquires the resource it clears the phase's REQUEST parameter.

Later, when we manually run through this example, you will act as the Batch Executive and set the values to 0 to acknowledge the requests.

Note In this example, the equipment ID for the COLOR_HEADER is 15. Your equipment ID may be different. Substitute your ID wherever 15 is used in this example.



Phase Coordination between Charge and Dump Phases

Configuring the Dump and Charge Phase Classes

The running logic SFCs for DUMP and CHG_COLOR are shown in the following diagrams. If you are viewing this book online, you might want to print these pages. Then you can put the pages next to each other and see how the phases communicate at different steps. The coordinated messages will have the same message IDs (the last two digits): a 5201 pairs with a 5501; a 5202 pairs with a 5502; and so on. In addition, note that the request to acquire the color header (4015, where 15 is the equipment ID) is later followed by a request to release the resource when it is no longer needed (4215). To make it easier to configure, we have shown the text for the actions and the transition expressions on the illustration.

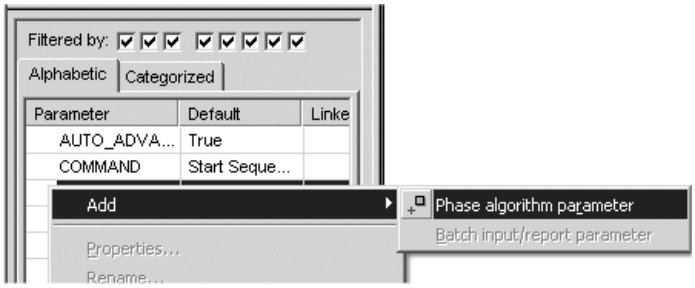
Note For the sake of brevity, in many of the examples in this tutorial, we only configure the running logic. However, in a real system, you need to carefully consider which active states you will use and then configure the logic for those active states, as well as for the Failure_Monitor.

Before configuring CHG_COLOR, you will add a parameter to the running logic to record the initial level in the blender.

To add a parameter to record the blender's initial level

1 Open the CHG_COLOR phase class in Control Studio and drill down into the running logic.

2 Right-click in the parameter view and select Add | Phase algorithm parameter.



3 Name the parameter INIT_LEVEL.

4 Leave the type as floating point and the value as 0.

5 Save the phase class now to be able to browse for the new parameter when creating the running logic.

Now you can configure the running logic for CHG_COLOR and DUMP, as shown in the illustrations that follow the procedure.

To configure the Charge and Dump phase classes

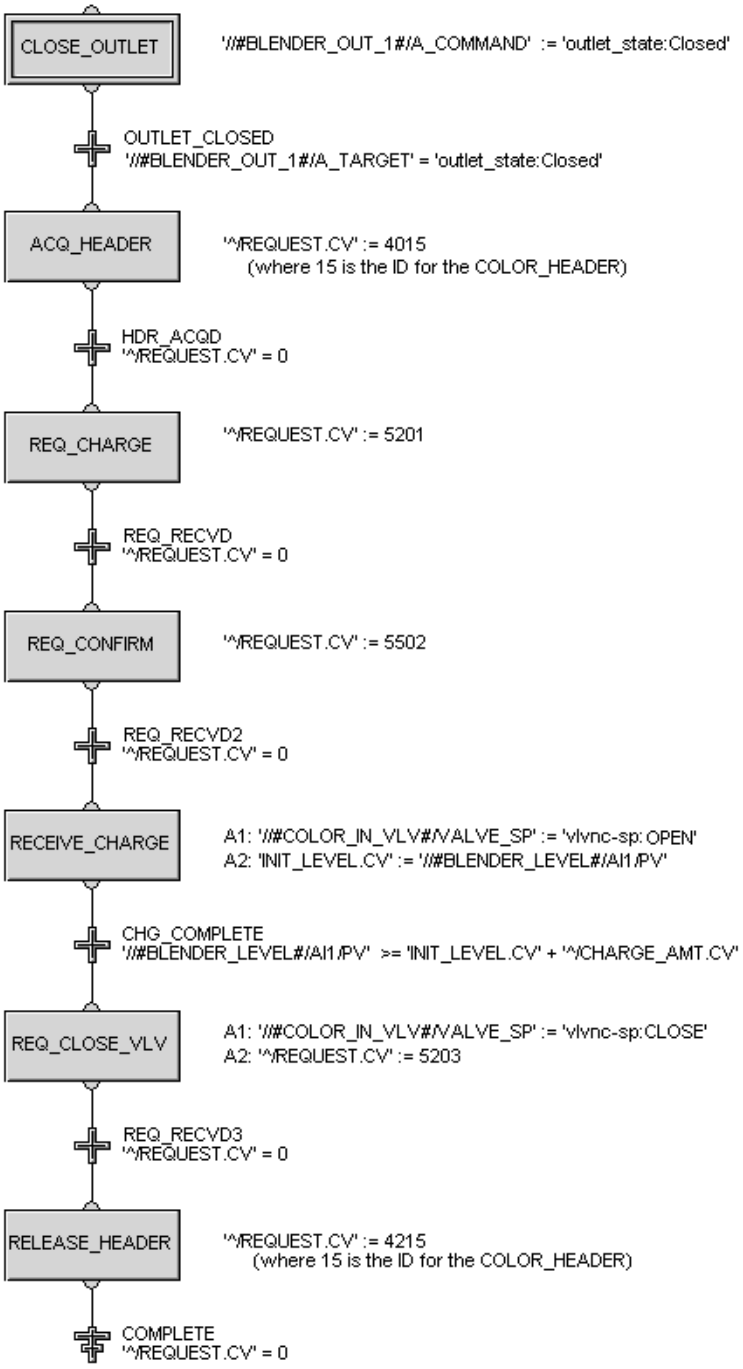
1 Create the SFC logic diagrams for the CHG_COLOR and DUMP phase class running logic as shown in the following diagrams.

2 In the DeltaV Explorer, assign the CHG_COLOR phase class to the BLENDER unit class. DUMP was previously assigned to the COLOR unit class.

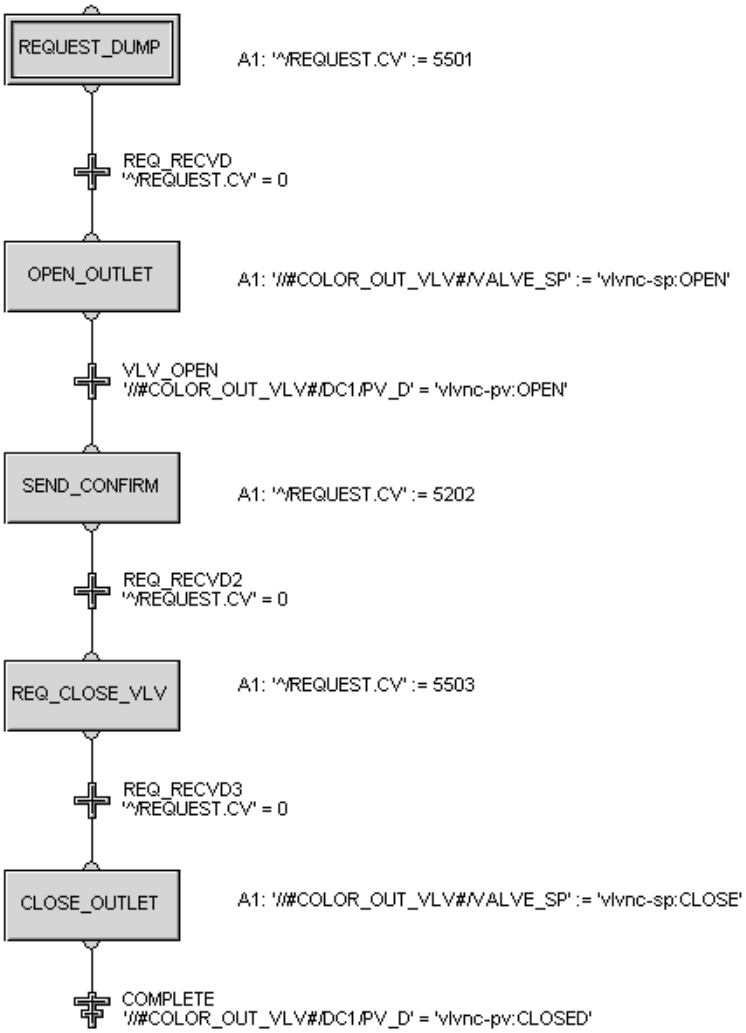
3 Click View | Refresh (or F5 on the keyboard) and verify that the new unit phases (that is, CHG_COLOR in both UM_BLEND_500 and UM_BLEND_600) have been created on the appropriate unit modules.

4 Select the new unit phases (both CHG_COLOR and DUMP) and mark them to be included in the download by selecting Controller as the Phase type on the Properties dialog.

CHG_COLOR Running Logic



DUMP Running Logic



Verifying the Phase Coordination - Overview

To verify the coordination of request messages between the dump and charge color phases, you will act as the Batch Executive (that is, you will manually acknowledge and clear the phase requests). During the next exercise, you will also verify the following:

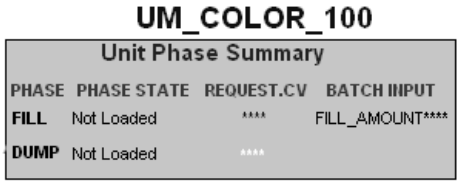
- The messages are sent and received.
- The COLOR_HEADER module is acquired and released.
- The correct amount of product is transferred from UM_COLOR_100 to UM_BLEND_500.

The way you will do these tasks is to open the Batch graphic in DeltaV Operate, start each phase, wait for the phase request codes to appear in the REQUEST parameter data link, and clear the requests by changing the data link value to 0. (In practice, you would never write 0 to a REQUEST parameter; this is something only the Batch Executive should do.)

You can keep the Batch graphic open in DeltaV Operate or you can switch back and forth from the Batch graphic to the Process graphic if you want to see what happens at each step in the phase communication.

Verifying the Phase Coordination - Details

Details for verifying the phase coordination follow. As a brief reminder, each box on the Batch graphic represents a unit module and has data links that let you access the unit module directly. For example, the box representing the unit module UM_COLOR_100 looks like this:



To start a phase, click the unit module name to open the unit module faceplate, load the phase, open the phase faceplate, and open the command list. If you need additional reminders of how to use the Batch graphic, refer to the Testing UM_COLOR_100 Using the Batch Graphic - Overview topic. In that exercise, you filled the first color tank to approximately 100 gallons. Now you will coordinate the phases to dump the color from the color tank and charge the blender.

To verify the phase coordination

1 Download.

2 Launch DeltaV Operate and open the Batch graphic.

3 Open the faceplate for unit module UM_COLOR_100.

4 Load the FILL phase, open its faceplate, and start the FILL phase.

5 The color tank fills to approximately 100 gallons. When complete, send a Reset command and unload the FILL phase.

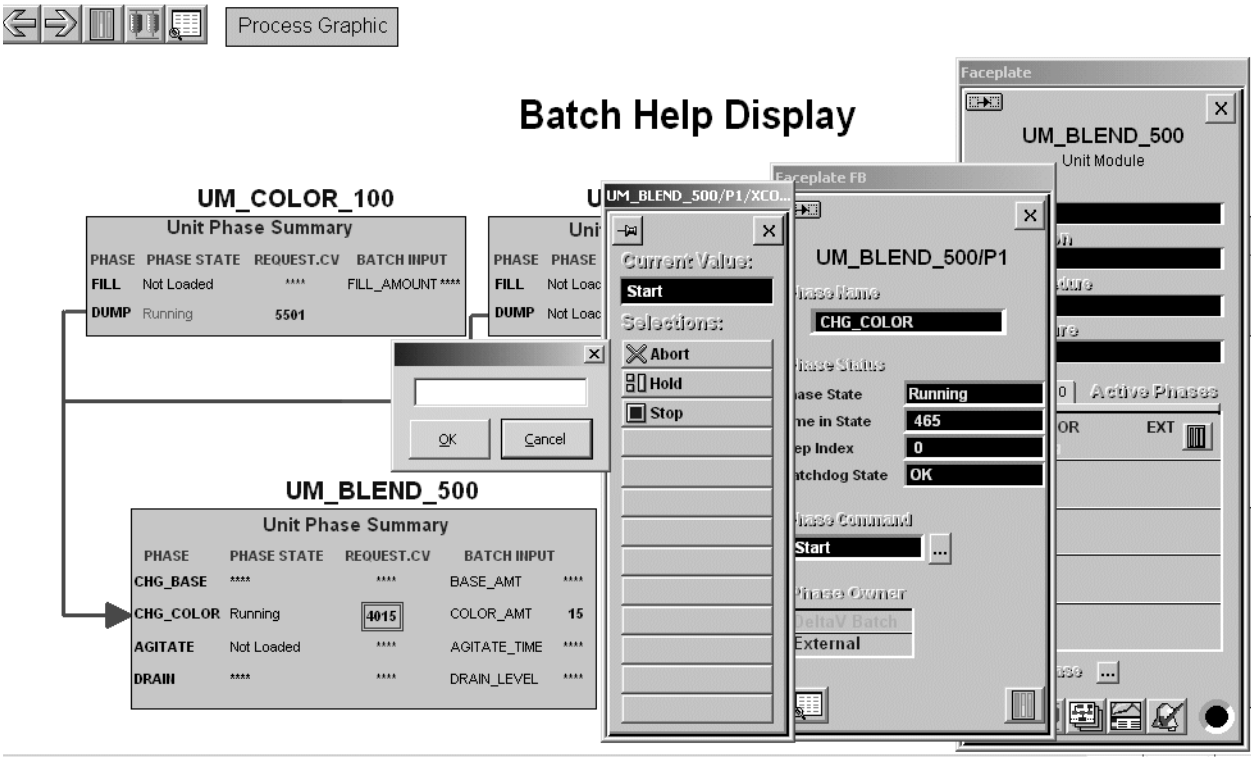
6 Load the DUMP phase for UM_COLOR_100 and open its faceplate.

7 Open the command list and click Start.

The DUMP phase state changes to Running. The REQUEST parameter changes to 5501, indicating it is waiting for the first message.

8 Load and start the CHG_COLOR phase in unit module UM_BLEND_500.

The CHG_COLOR state changes to Running and waits with a 40nn (such as 4015) request to acquire the color header resource.



9 Clear the 40nn request in CHG_COLOR by clicking the number and entering 0 in the Data Entry dialog box. This gives control of the two valves in the color header to the CHG_COLOR phase.

A new request, 5201, appears. This request synchronizes with the 5501 request in the DUMP phase.

10 Clear the 5201 request and the 5501 request.

Two new requests, 5202 and 5502, are written to DUMP and CHG_COLOR, respectively. The outlet valve XV_OUT_100 from the color tank opens.

11 Clear 5202 and 5502.

Request 5503 is written to the DUMP phase. Inlet valve XV_COLOR_500 opens, and UM_BLEND_500 starts filling.

12 Open the Process graphic so you can see UM_BLEND_500 fill. (The default charge amount, 15 gallons, was specified earlier, in the phase class definition.) When the tank stops filling, return to the Batch graphic.

When the tank charge is complete, Request 5203 (to close the inlet valve XV_COLOR_500) is written to the CHG_COLOR phase.

13 Clear 5503.

On the faceplate for DUMP, the state goes to Complete, and outlet valve XV_OUT_100 closes.

14 Clear 5203.

Valve XV_COLOR_500 closes. Request 42nn is written to the CHG_COLOR phase to request release of the color header.

15 Clear 42nn.

On the faceplate for CHG_COLOR, the state goes to Complete.

16 Send a Reset command to CHG_COLOR and unload the phase.

17 Send a Reset command to DUMP and unload the phase.

18 Close any open faceplates and close DeltaV Operate.

Hint If you want to rerun the modules a few times, you need to be able to drain the blender. Since we don't have a drain phase yet, you can click the RESET BLENDERS button on the Process graphic. This resets the blender tank level to 0.

The CHG_BASE and DRAIN Phase Classes

Two additional phase classes (CHG_BASE and DRAIN) were imported as part of the Startup.fhx file. The diagrams for the running logic for these phase classes are included in the following topics. CHG_BASE requires creation of a BASE_HEADER module similar to the COLOR_HEADER module created earlier. When CHG_BASE is assigned to the BLENDER unit class, two aliases, BASE_OUT_FLOW and BASE_OUT_PUMP, are automatically generated. These aliases must be resolved in the UM_BLEND_500 and UM_BLEND_600 unit modules.

To create the BASE_HEADER module and complete the configuration

1 Modify the XV_BASE_500 and XV_BASE_600 modules to allow them to participate in equipment arbitration.

2 Create a BASE_HEADER module similar to the COLOR_HEADER module that you created earlier. (Refer to the Creating the COLOR_HEADER Module topic if you need details for doing this.)

3 Modify BASE_HEADER to allow it to participate in equipment arbitration, and add the XV_BASE_500 and XV_BASE_600 to its equipment needed list. Note the Equipment ID of the BASE_HEADER module to use in your CHG_BASE requests to acquire and release the resource. (In the example, we use 23 for the BASE_HEADER equipment ID. If necessary, change the phase logic to use the actual ID numbers in your database.)

4 Assign BASE_HEADER to the controller or workstation.

5 Assign the CHG_BASE and DRAIN phase classes to the unit class BLENDER and make the phase type for the corresponding unit phases Controller in both UM_BLEND_500 and UM_BLEND_600.

6 Click Aliases under UM_BLEND_500 and click View | Details.

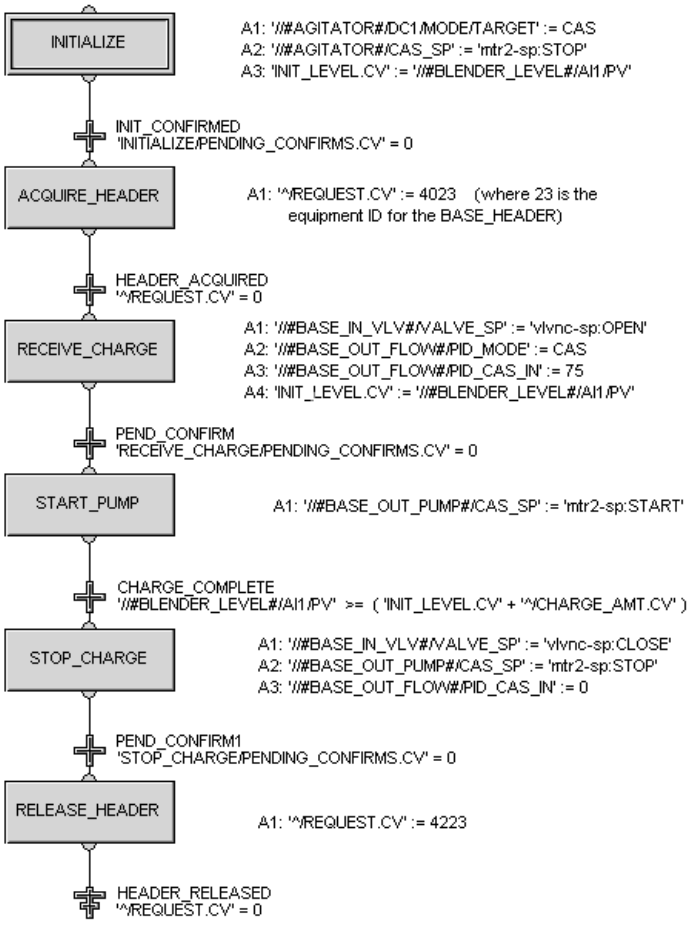
7 Double-click the alias BASE_OUT_FLOW and browse to UM_BASE_400 | FIC_BASE_400.

8 Double-click the alias BASE_OUT_PUMP and browse to UM_BASE_400 | MTR_400.

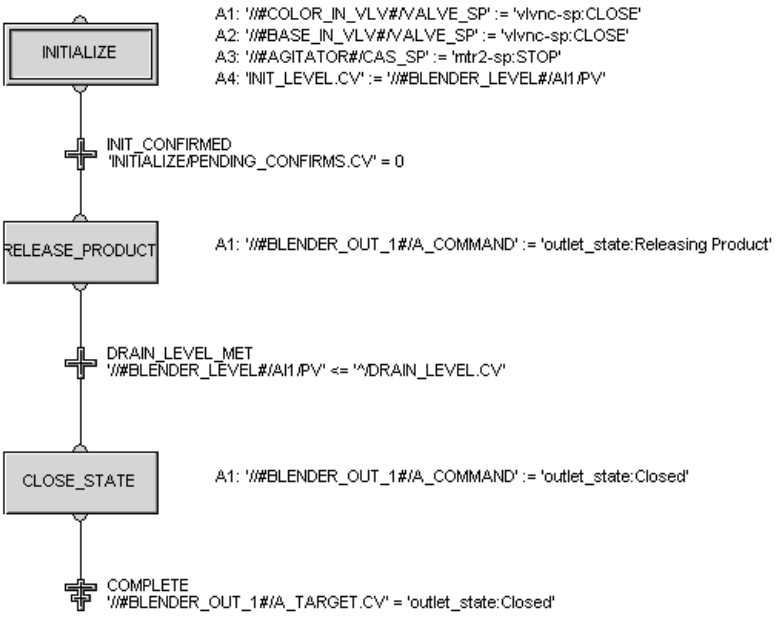
9 Download the controller or workstation and test the modules using the Batch graphic. Remember to manually clear any phase requests that appear in the CHG_BASE phase by setting them to 0.

10 After testing, reset all phases to Idle and unload them.

CHG_BASE Running Logic



DRAIN Running Logic



Unit Module Summary

The following table lists the unit modules and the assigned phase classes you should now have in the paint application. You may want to check in the DeltaV Explorer to make sure all these assignments have been made and that the unit phases have been marked for inclusion in the download (that is, the phase type is controller).

Unit Module	Unit Phases	Module Class	Modules
UM_COLOR_100	DUMP FILL	TOTALIZER_1 FLOW_CONTROL TOTAL_FLOW COLOR_OUT_VLV	EM_TOTAL_100 FIC_COLOR_100 TOTAL_FLOW_100 XV_OUT_100
UM_COLOR_200	DUMP FILL	TOTALIZER_1 FLOW_CONTROL TOTAL_FLOW COLOR_OUT_VLV	EM_TOTAL_200 FIC_COLOR_200 TOTAL_FLOW_200 XV_OUT_200
UM_COLOR_300	DUMP FILL	TOTALIZER_1 FLOW_CONTROL TOTAL_FLOW COLOR_OUT_VLV	EM_TOTAL_300 FIC_COLOR_300 TOTAL_FLOW_300 XV_OUT_300
UM_BLEND_500	AGITATE CHG_BASE CHG_COLOR DRAIN	AGITATOR BASE_IN_VLV BLENDER_LEVEL COLOR_IN_VLV BLENDER_OUT_1 OUT_PUMP PRODUCT_VLV WASTE_VLV	AGIT_500 XV_BASE_500 LI_500 XV_COLOR_500 EM_OUTLET_500 MTR_500 XV_PROD_500 XV_WASTE_500
UM_BLEND_600	AGITATE CHG_BASE CHG_COLOR DRAIN	AGITATOR BASE_IN_VLV BLENDER_LEVEL COLOR_IN_VLV BLENDER_OUT_1 OUT_PUMP PRODUCT_VLV WASTE_VLV	AGIT_600 XV_BASE_600 LI_600 XV_COLOR_600 EM_OUTLET_600 MTR_600 XV_PROD_600 XV_WASTE_600
UM_BASE_400		BASE_OUT_FLOW BASE_OUT_PUMP	FIC_BASE_400 MTR_400

You can test the phases using the Batch and Process graphics. Reset all phases to Idle and unload them when you are done.

Looking Ahead

Now you have all the phase logic needed to operate the various units--but everything has to be done manually by loading and starting the unit phases. You could write an SFC to sequence the phases, but that approach has obvious limitations. This is where the batch recipes come into the picture. Recipes let you fully automate your batch application and give you great flexibility at the same time. With recipes and formulas, you have control over which equipment is used, what ingredients go into the product, the quantities of ingredients, and virtually every other parameter in your batch process.

Batch Recipe Creation

A recipe defines a sequence of steps that is started and stopped in a specific order. Recipes enable a batch plant to use the same process equipment to produce many different products. A recipe can be as simple as a single operation or as complex as a series of unit procedures.

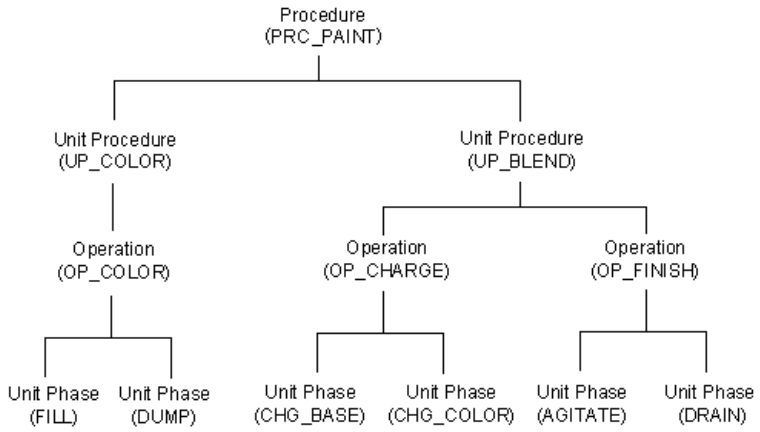
In the DeltaV Batch subsystem, recipes are defined following the ISA S88.01 procedural hierarchy, as described in the Procedural Control Model topic in the *Batch Reference* manual:

- **Procedure** - at the top of the hierarchy; controls a sequence of unit procedures to make a batch
- **Unit procedures** - at the next highest level; control a sequence of operations on a single unit
- **Operations** - at the lowest level; control a sequence of phases to perform a certain function on a single unit

To develop flexible and reusable recipes, consider the following:

- Build a library of generic operations that can be reused in multiple recipes.
- Create recipes that can run on multiple unit modules.
- Know your plant's layout and how it will affect the recipe design. For example, the physical configuration of the equipment in the plant determines which equipment units (and, therefore, unit modules) are available for a recipe.

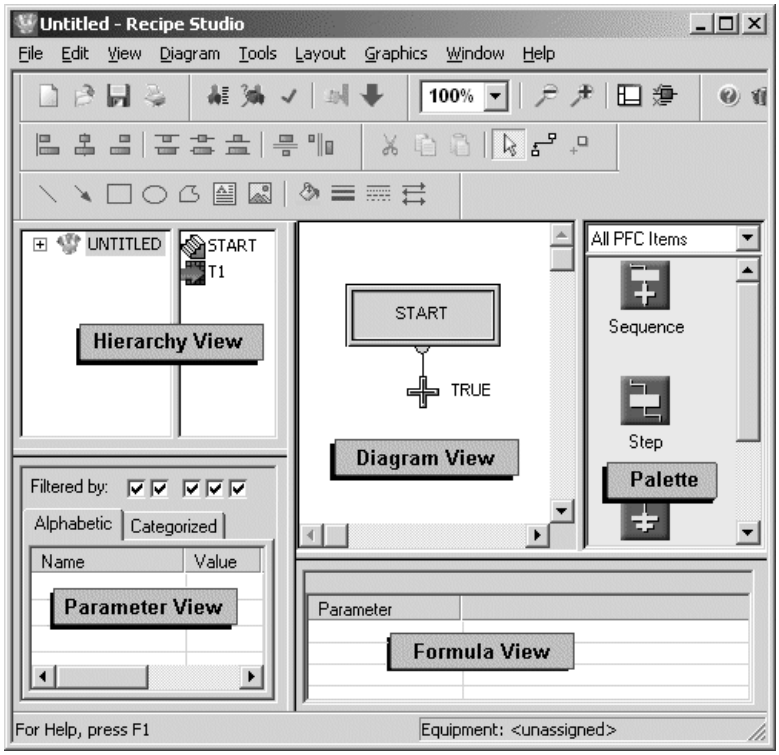
The following diagram shows the recipes we will create for the paint application. We have used a convention of prefixing the names of procedures, unit procedures, and operations with PRC, UP, and OP, respectively, but that isn't a requirement. Recipe names may be up to 16 characters (letters, numbers, and underscores) and start with a letter.



As shown in the diagram, operations (such as OP_FINISH) execute one or more unit phases (such as AGITATE and DRAIN). In turn, unit procedures execute one or more operations, and procedures execute one or more unit procedures. Class-based recipes use unit classes to define the equipment the recipe should run on; any unit in the defined unit class can be chosen at run time. For example, if OP_FINISH is configured as a class-based operation, then the AGITATE and DRAIN phases may run on either UM_BLEND_500 or UM_BLEND_600 during the operation. The equipment is assigned (bound) at batch creation time by selecting the unit modules to be used for that batch.

The Recipe Studio Application

Recipe Studio, the application used to configure recipes, is similar in look and feel to Control Studio, the application used to configure control module and phase class logic diagrams. The steps and transitions in a recipe make up a procedural function chart rather than a sequential function chart. At the operation level, the steps are phases; at the unit procedure level, the steps are operations; and at the procedure level, the steps are unit procedures.



Procedural Function Charts

A **procedural function chart** (PFC) is a graphical representation of the recipe's logic. You construct the PFC using steps, transitions, and a termination from the Recipe Studio palette. As soon as a transition is true, the previous step is stopped if it is not already complete. Typically a transition condition is a pause to wait until the preceding step completes. However, it is also valid to create a transition such as LEVEL > 50 to stop a FILL step when the tank level is greater than 50.

Creating Transitions Automatically

There is an option in Recipe Studio to automatically write a simple transition condition that checks for completion of the preceding step. To set this capability as a default for a recipe, click Tools | Default Transition Conditions. For an individual transition, you can right-click and select Modify and click the Default Condition button.

In an operation with default transition writing enabled, when you draw the connecting line from a step to a transition, the transition condition is automatically written as follows:

```
phaseclass:n/BSTATUS=$phase_state:Complete
```

The term phaseclass:n is the instance of the particular phase class being used in the operation. The parameter BSTATUS is the phase logic parameter that indicates the current status of the phase. The name $phase_state is a standard named set in the DeltaV system.

Hint To see the text of the transition condition on the diagram instead of the name or description, click Tools | Diagram Preferences. In the Show Diagram section, under Transitions, select Condition.

Creating an Operation (OP_FINISH)

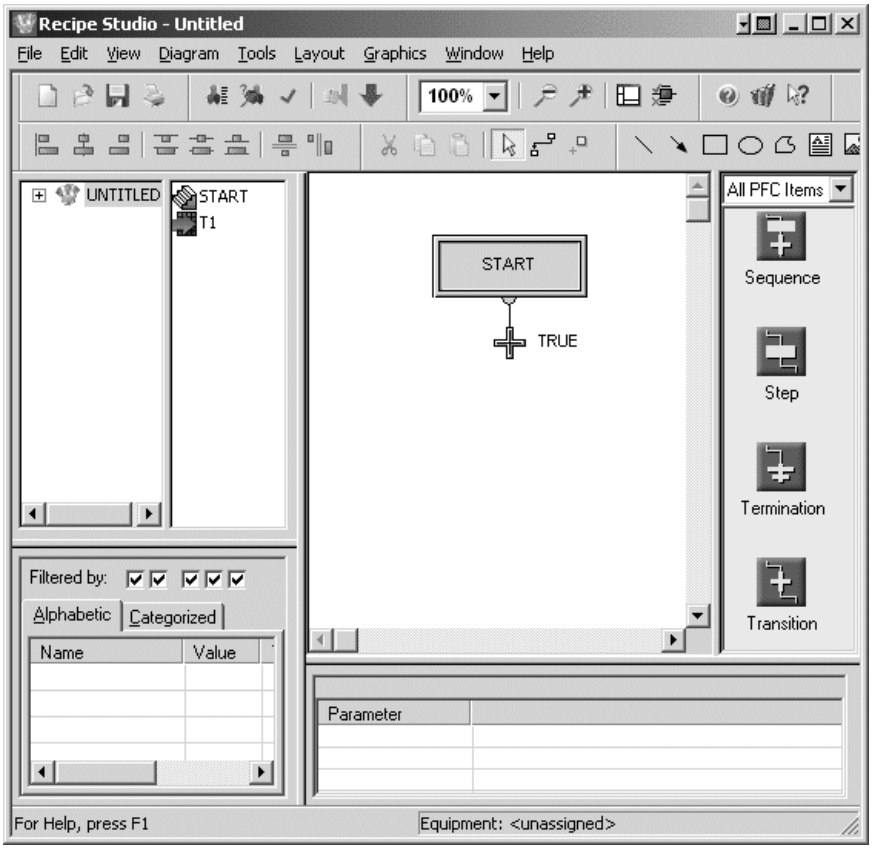
The first operation you will create is one to agitate and drain the contents of a blender.

You can create a recipe in DeltaV Explorer and then open it in Recipe Studio to configure it. You can also create a recipe directly in Recipe Studio, configure it, and save it.

To create the operation OP_FINISH in Recipe Studio

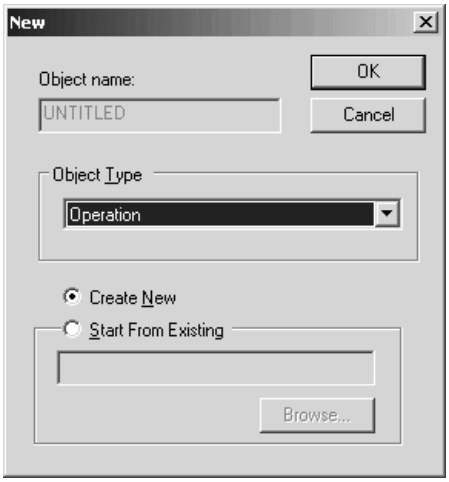
1 Click **Start | DeltaV | Engineering | Recipe Studio**.

Recipe Studio opens with a single step and transition in the diagram view.



2 Click File | New.

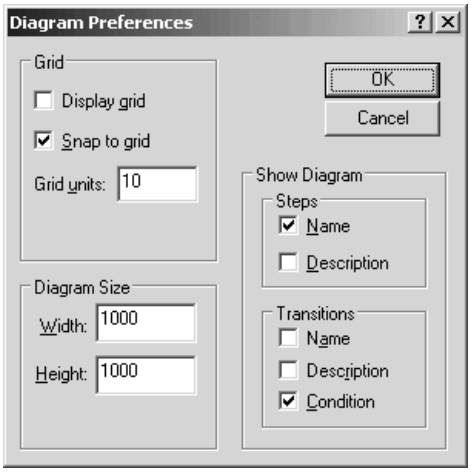
A New dialog box opens.



3 Accept Operation as the Object Type and click OK.

4 Click Tools. Make sure Default Transition Conditions is selected so that transition conditions will be written automatically.

5 Click Tools | Diagram Preferences and select Condition under Transitions so that you can see the conditions on the diagram.



Configuring an Operation (OP_FINISH)

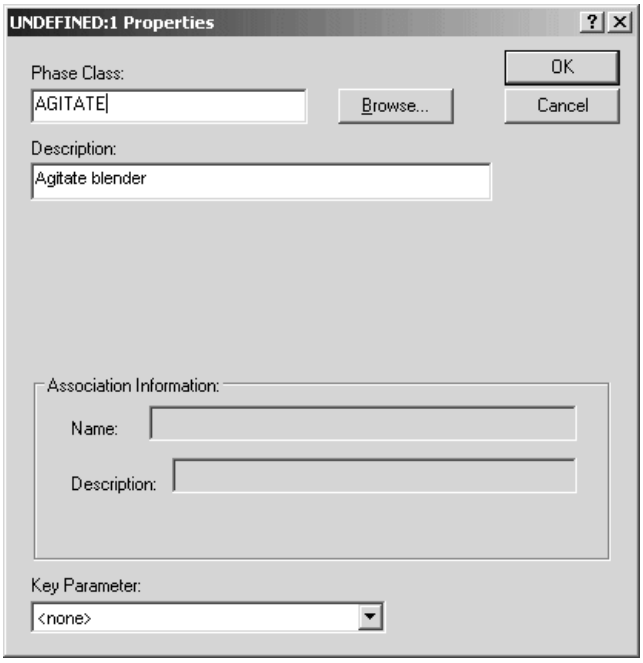
To configure OP_FINISH

1 Leave the first step and transition as they are on the screen.

2 Add a new step by dragging a step icon from the palette to a spot below the first transition.

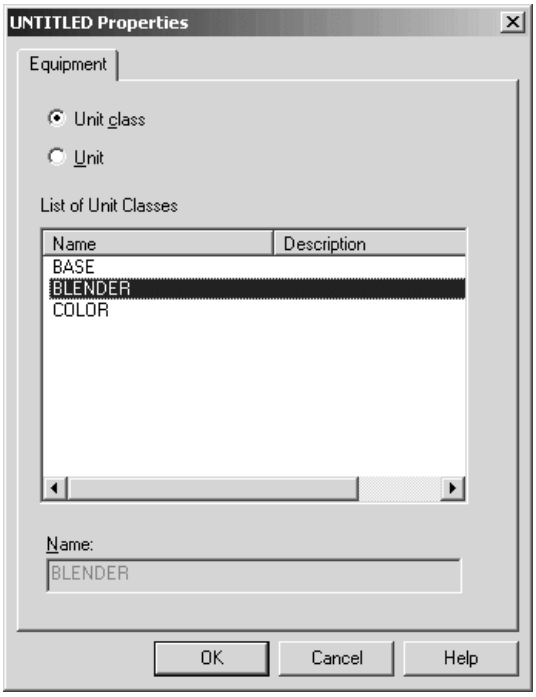
3 Select Properties from the context menu to open the Properties dialog (or double-click the step).

4 Browse for the phase class AGITATE (in the BlenderPhases category). Enter the description as Agitate blender.



The Key Parameter field allows you to select one parameter that will display on the step when viewing the recipe in the Batch Operator Interface. For this tutorial, leave it as the default <none>.

5 Click OK. Another dialog box opens for you to select whether the recipe is to be class-based or unit specific. This only occurs on the first step in the recipe and sets the class or unit for the current recipe. Select Unit class and select the class BLENDER.



6 Draw a line from the previous transition to the step.

7 Add a transition and draw a line from the previous step to the transition. The following transition text is automatically created:

```
'AGITATE:1/BSTATUS' = '$phase_state:Complete'
```

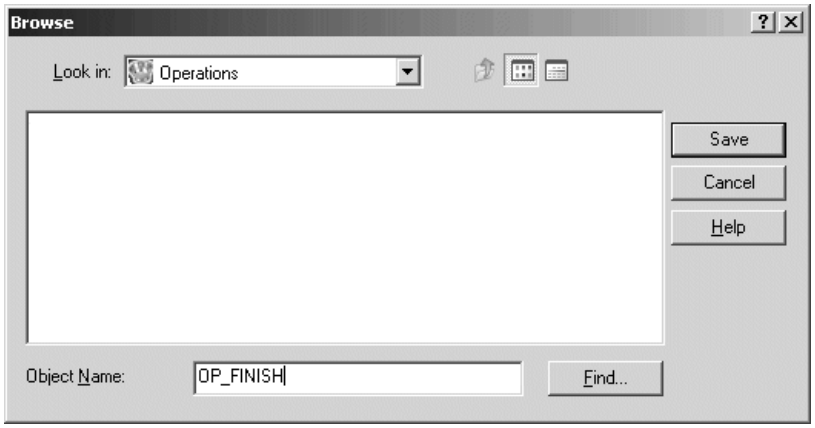
8 Add a step and modify it to use the phase class DRAIN. Enter the description as Drain blender.

9 Draw a line from the previous transition to the step.

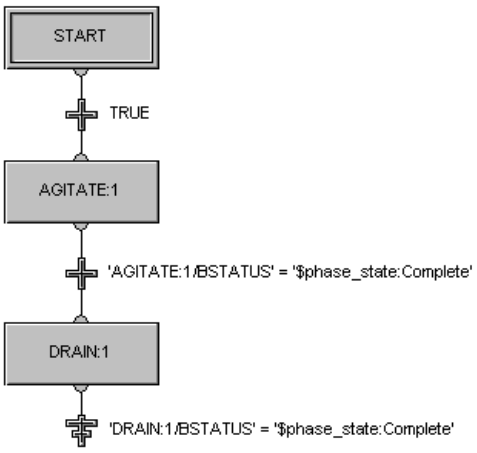
10 Drag a termination below the step.

11 Draw a line from the step to the termination.

12 Click File | Save and name the operation OP_FINISH.



The completed procedural function chart looks like the following:



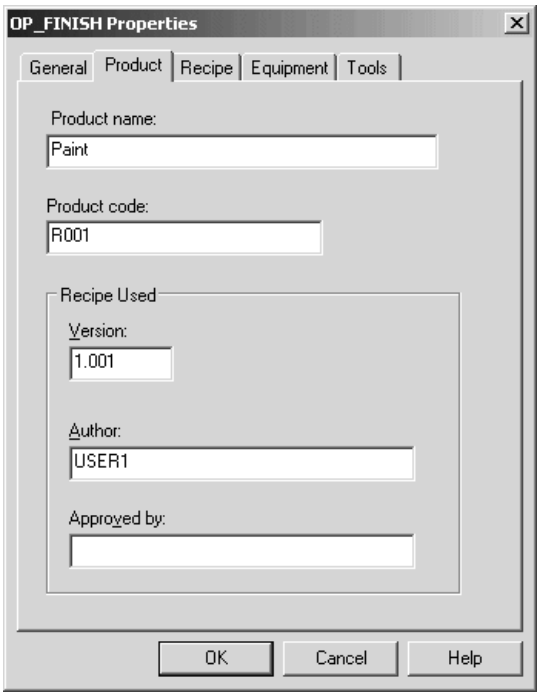
Completing the Recipe Properties

There are several fields you might want to complete to document the recipe. These fields are found on the Product tab of the recipe's Properties dialog. We recommend that, at a minimum, you enter a Version identifier and Author. The Version identifier can be used to manually track when changes have been made to the recipe or the formula. The Version field and Author are logged in the batch history file when a recipe is run.

To complete the recipe properties

1 Click File | Properties to open the Properties dialog.

2 On the Product tab, enter a Product name (Paint) and Product code (any string).



3 Enter 1.001 (or any string) in the Version field and your name in the Author field.

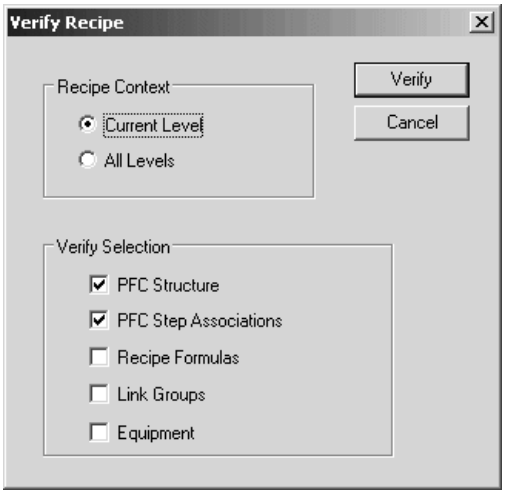
4 Click OK.

Verifying and Saving the Recipe

To finish the recipe, you should verify and then save it.

To verify and save the recipe

1 Select File | Verify Recipe or click the Verify Recipe button  on the toolbar. The Verify Recipe dialog appears.



2 Keep the default settings and click Verify. If any errors are found, correct them.

3 Save the recipe.

Assigning Recipes to a Batch Executive

Each Batch Executive contains an Assigned Recipes object. When you assign recipes and recipe folders to the Batch Executive, you release the recipe to production, making it available to the operator to create and run the recipe. Individual recipes and recipe folders can be assigned to run on multiple Batch Executives; however, all the units needed for a particular recipe must reside on one Batch Executive.

Assigning a recipe to a Batch Executive allows it to be displayed in the Batch List in the DeltaV Batch Operator Interface. You normally assign to a Batch Executive only high-level recipes (procedures and perhaps unit procedures) that you want the operator to be able to select and run individually. We will also assign an operations-level recipe (OP_FINISH) so that we can run it individually to demonstrate some features in later sections. We will talk more about the Batch Executive in the next chapter. (Although the Batch Executive has not yet been enabled, assign the recipe anyway.)

To assign the recipe to a Batch Executive

1 In the DeltaV Explorer, drill down to the Assigned Recipes object under the Batch Executive.

2 Drag OP_FINISH from the Recipes | Operations folder to Assigned Recipes. (You can also drag the recipe to the Batch Executive itself and the system will automatically assign it to the Assigned Recipes folder.)

3 Download the workstation.

Creating Additional Operations

Use the following table to help you create additional operations needed to run batches in the paint application. Remember to fill in the Version (any string) and Author on the Product tab of the Properties dialog.

Operation Name	Phase Classes	Unit Class
OP_CHARGE	CHG_BASE	BLENDER
	CHG_COLOR	
OP_COLOR	FILL	COLOR
	DUMP	

Save each recipe after configuring it.

Creating Unit Procedures

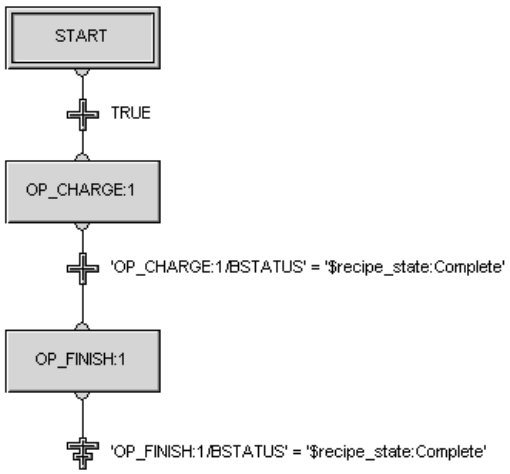
Unit procedures are the second level in the procedural hierarchy. The steps in a unit procedure represent operations. All operations in a unit procedure must run on the same unit or unit class.

Creating unit procedures is very similar to creating operations. The first unit procedure you will create is UP_BLEND, which will be run on the BLENDER unit class. This unit procedure will run OP_CHARGE (for charging the base and color into a blender) and OP_FINISH (for agitating the contents of and draining a blender). Use Default Transition Conditions from the Tools menu when creating these recipes.

To create unit procedures (UP_BLEND and UP_COLOR)

1 In Recipe Studio, click File | New and select Unit Procedure as the recipe object type.

2 Create the unit procedure as shown in the following diagram.



3 Select BLENDER as the unit class.

4 Finish the unit procedure by filling in the Properties dialog.

5 Save the recipe as UP_BLEND.

6 Create a unit procedure named UP_COLOR. This unit procedure contains only one operation, OP_COLOR. Use COLOR for the unit class.

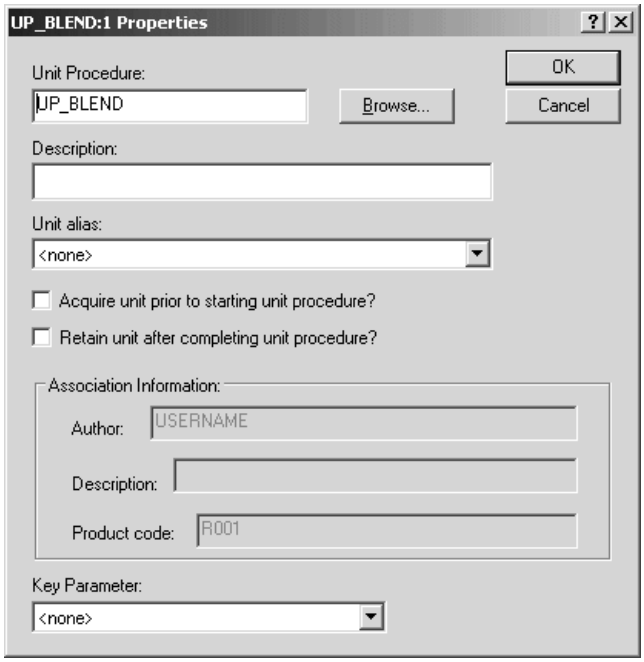
Creating Procedures

Procedures are the highest level in the recipe hierarchy. A procedure includes unit procedures in the steps of its procedural function chart. The unit procedures in a procedure may (and usually do) run on more than one unit or unit class.

The Properties dialog for a unit procedure step within a procedure includes an additional check box option: **Acquire unit prior to starting unit procedure?** A procedure may control several units simultaneously; this option lets you decide when a branch of the procedure can be started. (If the check box is checked for any unit procedure in a parallel branch, all of the required units for that unit procedure must be acquired before any unit procedures in that branch will be started.)

Additionally, selecting **Retain unit after completing unit procedure** means that the procedure acquires the unit, and the unit is not released until the completion of another unit procedure for which this selection is clear (that is, the Retain unit after completing unit procedure check box is **not** selected). If this check box is clear, the unit is released as soon as the unit procedure completes.

We will leave both options unselected for the purpose of this tutorial.

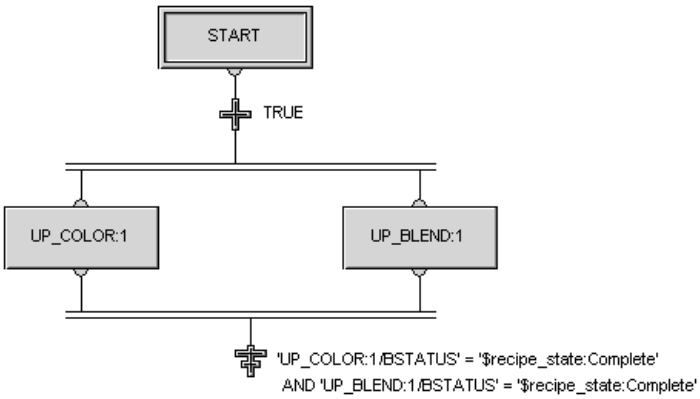


Procedures are created in the same way as unit procedures and operations.

To create a procedure

1 In Recipe Studio, click File | New and select Procedure as the recipe object type.

2 Create the procedure as shown in the following diagram.



3 On the Properties dialogs for UP_COLOR and UP_BLEND, do **not** select the check box Acquire unit prior to starting unit procedure. This allows each unit procedure to be started individually as soon as the unit it needs is available.

4 Click File | Properties and fill in the other fields on the Properties tab.

5 Save the procedure as PRC_PAINT.

6 In DeltaV Explorer, assign the procedure recipe to the Batch Executive and download.

Unit Aliasing

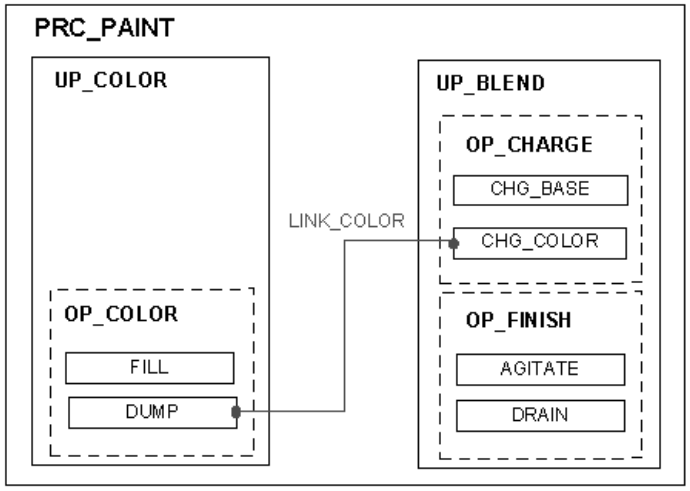
You can configure a unit alias for a step in a procedure. Because our example in this tutorial is a simple recipe, we will not be using unit aliasing. To use unit aliases, you first define any unit aliases to be used in the procedure in either Recipe Studio or DeltaV Explorer. (In Recipe Studio, select Unit Aliases from the File menu; in DeltaV Explorer, right-click the procedure and select Unit Aliases from the context menu.) Then, when you add a step to the procedure, you specify on the Properties dialog the unit alias that it will use. (See the Properties dialog in the previous topic, Creating Procedures.) The unit alias can be for a specific unit or for a unit class.

If we were going to create a recipe that required multiple colors to be charged to one blender, unit aliasing would be useful. We could have one unit procedure that charges color and use unit aliasing to specify that the tank unit class would be used. If we didn't use unit aliasing, we would need a separate unit procedure for the charging of color from each color tank.

For more information on unit aliasing, refer to the topic Using Unit Aliases in Recipes.

Link Groups

Link groups facilitate communication between and synchronization of phases during execution of the batch. The two unit procedures referenced by the procedure PRC_PAINT are started at the same time. However, two phases in different unit procedures need to be synchronized, as shown in the following figure.



Dumping color from the color tank needs to be synchronized with charging color to the blender. Phases that require synchronization must belong to a link group so that the Batch Executive knows which phases are allowed to communicate with each other. Potentially, request messages could be sent to the Batch Executive from numerous phases in different recipes; the link groups keep the Batch Executive from trying to match requests that are not intended for each other.

In addition, a phase that needs to communicate with another phase during production of a batch must let the Batch Executive know that it is to have a phase partner. We will, therefore, increase the number of phase partners (from 0 to 1) for the two phase classes that need to be in link groups so that those phases can be set up to communicate.

Adding Phase Partners

Phase partners are phases that communicate with other phases during batch execution. The number of phase partners that a phase can have is defined for the phase class in the DeltaV Explorer. Based on the number of partners you specify, the recipe author can build a phase link group, which lists the phases that can communicate with each other in the recipe.

Note Any phase used in a link group must be based on a phase class with a number of partners greater than zero (0); otherwise, the recipe will not be loaded.

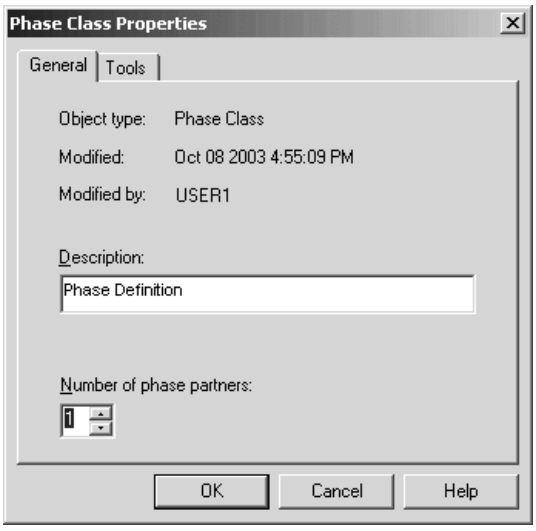
Typically, a phase link group contains phases that must be synchronized. Earlier in the tutorial, in the Coordinating the Dump and Charge Color Phases topic, you synchronized transferring the color out of the color tank and into the blender. The synchronization was programmed into the phase logic by adding REQUEST codes to send messages from one phase to the other. Now, you will add phase partners to allow these phases to establish communication during recipe execution.

To add phase partners

1 Go to DeltaV Explorer and expand Library | Advanced Definitions | Phase Classes.

2 Select the phase class DUMP (under TankPhases), right-click, and select Properties from the context menu.

3 Increase the number of phase partners to 1.



4 Select the phase class CHG_COLOR (under BlenderPhases) and change the number of phase partners to 1.

Creating a Link Group

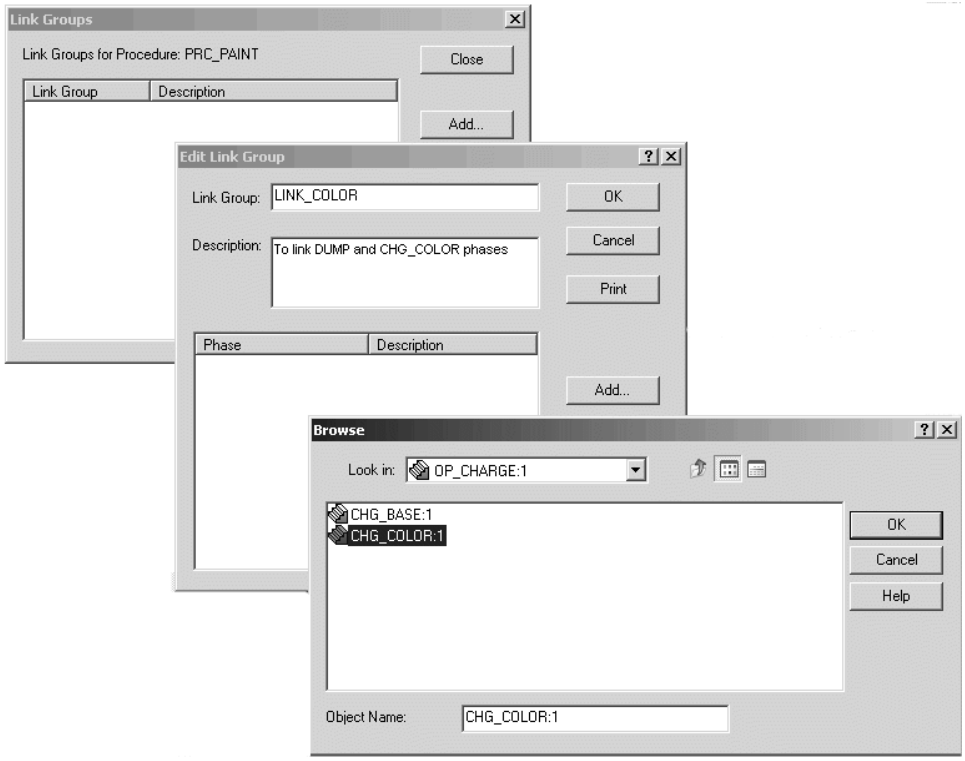
In this exercise, you will create a link group called LINK_COLOR. The link group will allow phase messaging between DUMP and CHG_COLOR.

To add the link group LINK_COLOR

1 In Recipe Studio, with PRC_PAINT open, select Edit | Link Groups and click Add.

2 Name the link group LINK_COLOR and enter a description.

3 Click Add in the dialog box.



4 Browse for the CHG_COLOR phase (under UP_BLEND | OP_CHARGE) and click OK.

5 Add DUMP to LINK_COLOR. (Browse under UP_COLOR | OP_COLOR.)

6 Close the link group and save PRC_PAINT.

We are almost ready to run batches in the Batch Operator Interface. However, before we can do that, we need to enable the workstation to run batch, initialize the DeltaV Batch Executive, assign the area to the Batch Executive, and download the workstation.

The DeltaV Batch Executive

The DeltaV Batch Executive consists of the Batch Manager service, the Batch Runtime Server, and the individual batch runner applications. One instance of a batch runner is created for every running batch. Each batch runner is responsible for taking the batch through all its states and for communicating to the Batch Manager for equipment arbitration and any other inter-runner or client-based communications. The Batch Manager is responsible for managing the batch runners. It also sends information from the runners to clients (such as the Batch Operator Interface and Campaign Manager Server) that require run-time data. For more information, refer to the Batch Executive Architecture topic in the Batch Reference book.

In the tutorial, we will configure the Batch Executive on the same workstation on which we are doing configuration and operation. However, optimal performance of the Batch Executive is achieved by installing the Batch Executive on its own Application Station. Consult your Emerson representative before installing a Batch Executive on a workstation other than an Application Station.

Note You cannot run the Batch Executive, Recipe Simulator, or Batch Operator Interface applications without first enabling Batch on the node (on the Systems Preferences dialog) and then enabling the Batch Executive on the workstation (on the Batch Executive Properties dialog in the DeltaV Explorer).

Options you need to configure when enabling the Batch Executive are:

Remove files older than... days - sets the maximum number of days for which batch event records are stored in the .evt file

Restart Behavior - sets the method for restarting the Batch Executive. Choices are:

- **Query User** - allows the user to choose between Warm & Cold Restarts or Cancel the operation
- **Warm Restart** - returns the Batch Executive to its last known state. Control recipes, manually controlled phases, and arbitration functions are returned to the state that existed prior to termination of the Batch Executive. The warm boot method is only used to recover batches and data after an unexpected loss of the Batch Executive.
- **Cold Restart** - starts Batch Executive with an empty batch list and with no resources allocated to the operator.

Hold propagation limit - sets the level to which a Hold propagates. The choices are Phase, UOP (operation), UPC (unit procedure), and BPC (procedure).

Setting up a Workstation to Run Batch Executive

To set up a workstation to run Batch Executive

1 In DeltaV Explorer, select System Configuration | Physical Network | Control Network, and click the + next to the workstation name.

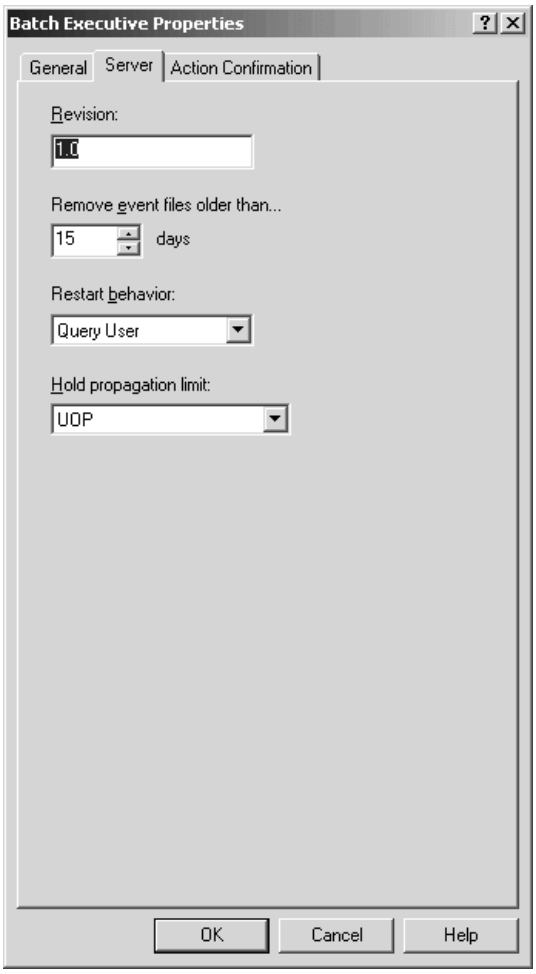
2 Select Batch Executive, right-click, and select Properties from the context menu.

3 On the General tab of the Properties dialog, click Enabled to enable this workstation for batch. (You will be reminded that you will need to download the workstation.)

4 Fill in the fields on the Server tab as follows:

- Revision - leave as is
- Maximum event file age - 15
- Restart behavior - Query User
- Hold propagation limit - UOP (operation level)

5 Click OK.



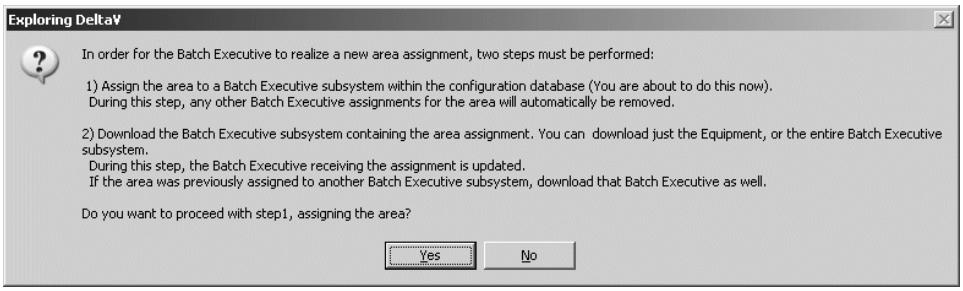
6 Download the workstation.

Assigning Areas to a Batch Executive

Each Batch Executive contains an Assigned Areas object. When you assign an area to a Batch Executive subsystem, you also assign the equipment contained in that area to that Batch Executive. A Batch Executive can control only the equipment assigned to it. A recipe will not run if all the required units are not found on the same Batch Executive as the recipe. An area can be assigned to only one Batch Executive subsystem at a time.

To assign an area to a Batch Executive

1 In the DeltaV Explorer, drag the area EXTERIOR_PAINT (under Control Strategies) to Assigned Areas under the Batch Executive. The following message is displayed:



2 Click Yes to continue with the area assignment.

3 Click Assigned Recipes under the Batch Executive to make sure that the recipes have been assigned to the Batch Executive. If recipes have not been assigned, drag OP_FINISH and PRC_PAINT to the Batch Executive.

4 Download the Workstation.

You are now ready to start the Batch Executive, as described below.

Starting the Batch Executive

The Batch Executive must be started and the recipe phases verified before any batches can be run..

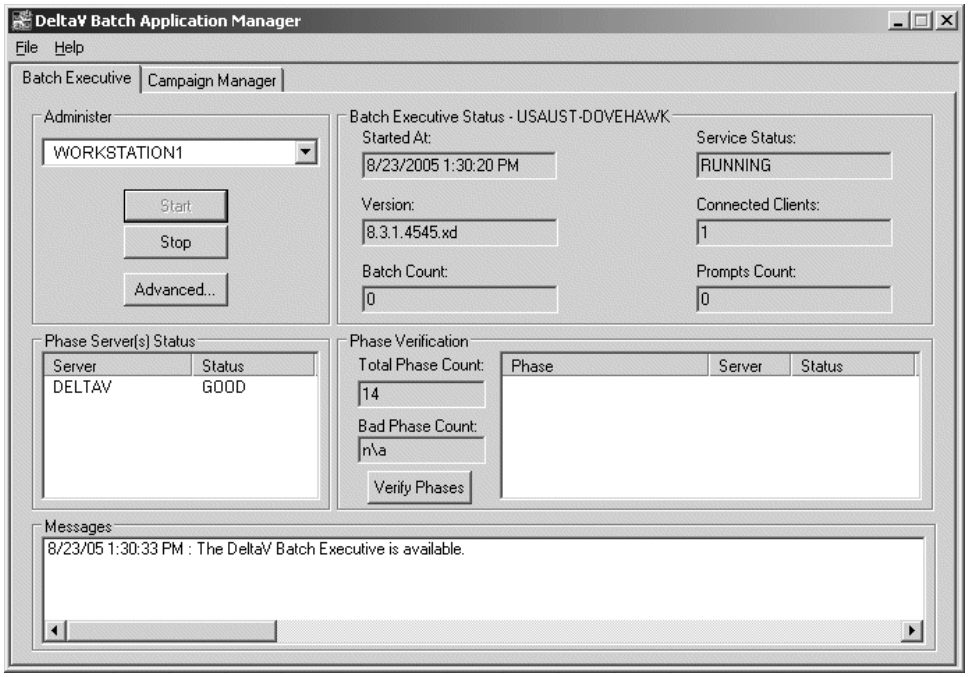
To start the Batch Executive

1 Click **Start | DeltaV | Batch | Application Manager**. A dialog box opens.

2 Under the Administer section of the Batch Executive tab, click the Start button.

3 Select Cold on the Choose Boot Method dialog. You are given this choice because we selected User Query as the restart method in the Batch Executive Properties dialog.

4 Wait for a few seconds until the Batch Executive is running and the phase server status is good.



5 Click Verify Phases.

The Batch Executive checks that all items specified in the batch configuration (phases, phase parameters, and so on) are available. There should be no Bad Phases. (If there are bad phases, first try clicking Verify Phases again. If there are still errors, go to the DeltaV Batch log file to find the errors. In Windows Explorer, go to DeltaV\DVData\Batch\Logs). If there are bad phases, you need to correct them in your configuration. Another thing to check (on the unit phase Properties dialog) is that the unit phases all have an assigned phase type of Controller. (Unit phases are the assigned phases in each blender and color unit module in the PAINT_BLEND process cell.)

6 Verify the number of Bad Phases is 0 before proceeding.

7 Click File | Exit.

The Batch Executive will continue running in the background. If you need to stop it, open the Application Manager dialog (from the **Start | DeltaV | Batch** menu) and click Stop.

The Batch Operator Interface

The DeltaV Batch system has two main operator interfaces:

- Batch Operator Interface - used primarily to manage batch recipes
- DeltaV Operate - used primarily to monitor the process equipment

You should already be familiar with DeltaV Operate. In the following topics, you will learn about the Batch Operator Interface and how to use it along with DeltaV Operate to manage the batch production process.

The Batch Operator Interface provides different views of the batch process through a number of operator displays, which are described in the *Batch Reference* manual in The Batch Operator Interface Screens topic. These displays allow users to perform the following tasks:

- Create batches from the list of recipes assigned to a Batch Executive
- Control and monitor recipe-based batches through the use of START, HOLD, RESTART, STOP, and ABORT commands
- Provide advanced batch control (change the mode of a batch, bind equipment to a recipe, scale a recipe, or run a segment of a recipe)
- Acquire and release batch resources and monitor resource arbitration
- Control and monitor individual phases
- Respond to prompts for parameter input
- View event logs of current or past batches
- Monitor, acknowledge, and clear batch-related alarms

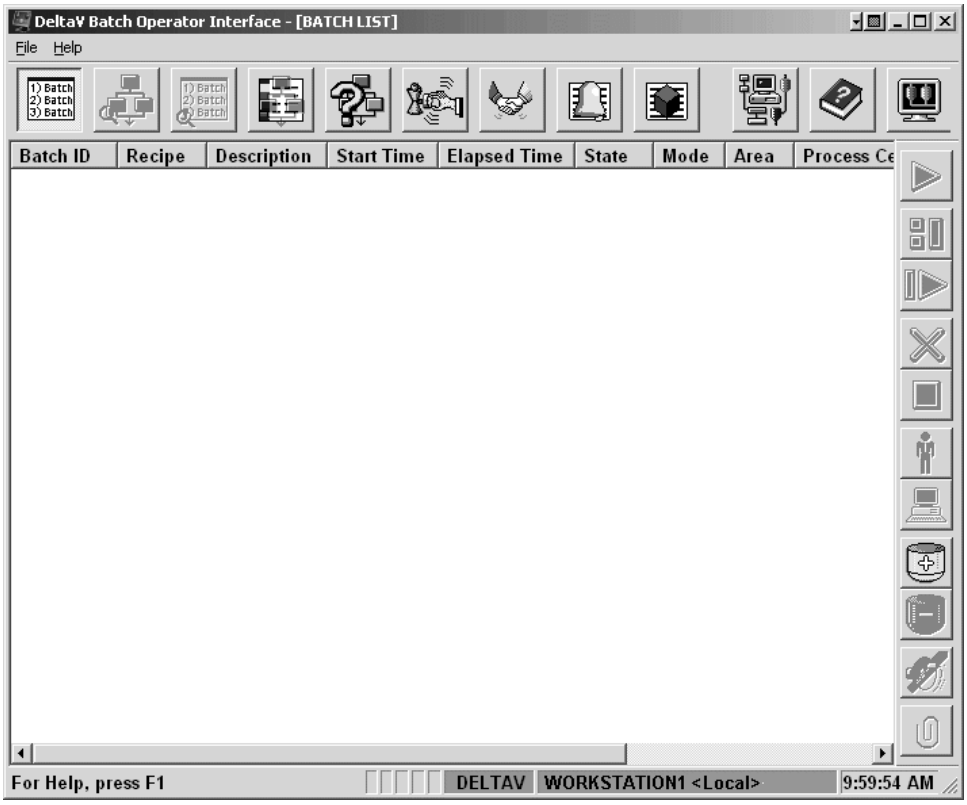
The Procedure as PFC display in the Batch Operator Interface allows the operator to view each step of the batch as it progresses. As a recipe executes, each phase transitions according to the state transition diagram. In the simplest case (when there are no failures or operator intervention), a phase goes from Idle to Running to Complete. Using the Batch Operator Interface, an operator can create a batch, start it, watch the progress, interrupt if necessary, and restart or abort the batch.

The batch state is displayed on most operator screens because it is important for the operator to be aware of the current state. The commands available for the operator to issue are dependent on the current state.

The Batch Executive must be running before you can use the Batch Operator Interface. See the DeltaV Batch Executive topic if you need to enable and start the Batch Executive.

Using the Batch Operator Interface

The default Batch Operator Interface screen is shown below. This is also called the Batch List screen. Any batch that has been created is listed on this screen until it is specifically removed. You can start batches from this screen and view the batch progress. You also have the option of going to a number of other screens within the Batch Operator Interface or directly to DeltaV Operate using the buttons on the toolbar.



The Batch Operator Interface Toolbar

The Batch Operator Interface toolbar along the top of the screen is a group of icons used mainly to navigate among the various Batch Operator Interface screens. To move from one window to another, simply click an icon on the toolbar. Tool tips help you easily identify each icon; when you pause the pointer over an icon, the name pops up. The following table describes some of the tools you will use in this tutorial. Refer to The Toolbar topic for a description of the complete toolbar.

Click this icon …	To …
	Switch to the Batch List screen. This screen opens by default each time you launch the Batch Operator Interface.
	Open the PFC View screen if there is a batch currently selected in the Batch List. If there are no batches selected in the Batch List, this icon is inactive.
	Open the Batch Event Journal screen.
	Open DeltaV Operate.
	Open the Batch Alarm Summary.

The Batch Operator Interface Button Bar

Many of the Batch Operator Interface screens have a set of command buttons on the right side of the screen that are used to issue commands to the current batch. When you pause the cursor over a button, the name of the button displays.

The buttons are not always active or available. The buttons and their availability change, depending on the function of a particular screen and the current state of the batch. For example, to avoid the inadvertent removal of an active batch, the Remove Batch button is available only if the Batch List has batches in the Stopped, Aborted, Complete, or Ready states.

The table below shows some of the key buttons and their purpose. The full set of buttons is described in the *Batch Reference* manual in The Command Button Bar topic.

Click this icon …	To …
	Start the batch.
	Hold the batch.
	Restart the batch.
	Abort the batch.
	Stop the batch or step.
	Put the batch in Manual mode.
	Put the batch in Auto mode.
	Add a batch.
	Remove a batch.
	Clear failures for the batch.
	Select equipment. **Note** Batch must be in MANUAL mode before this button is active.

Adding and Starting a Batch

Recipes built referencing a unit class must have the equipment bound to the step before that step will become active. You can select the equipment in the following ways:

- On the Batch Creation dialog, in the Unit Binding section, select the units from the drop-down list.
- If you have not selected the equipment before starting the batch, the Unacknowledged Prompts button will flash on the toolbar when the equipment is needed by the phase. When you click the button, a dialog box opens for you to select the equipment to be used.

The OP_FINISH recipe is class-based and will run on one of the units in the BLENDER class.

To be able to select equipment for a step in the Batch Operator Interface, the operator (or the user who is currently logged in) must have the BATCH_ADD key for the area containing the equipment that is required by the recipe. That area (containing the equipment) must be assigned to the same Batch Executive as the recipe. Therefore, if the equipment list is blank, either the unit class referenced by the recipe is not in the area assigned to the Batch Executive, or the user does not have the BATCH_ADD privileges for that area. (Function lock assignments are listed in the System Configuration | Setup | Security area of the DeltaV Explorer. User privileges are assigned in the DeltaV User Manager.)

To start the Batch Operator Interface and run the batch

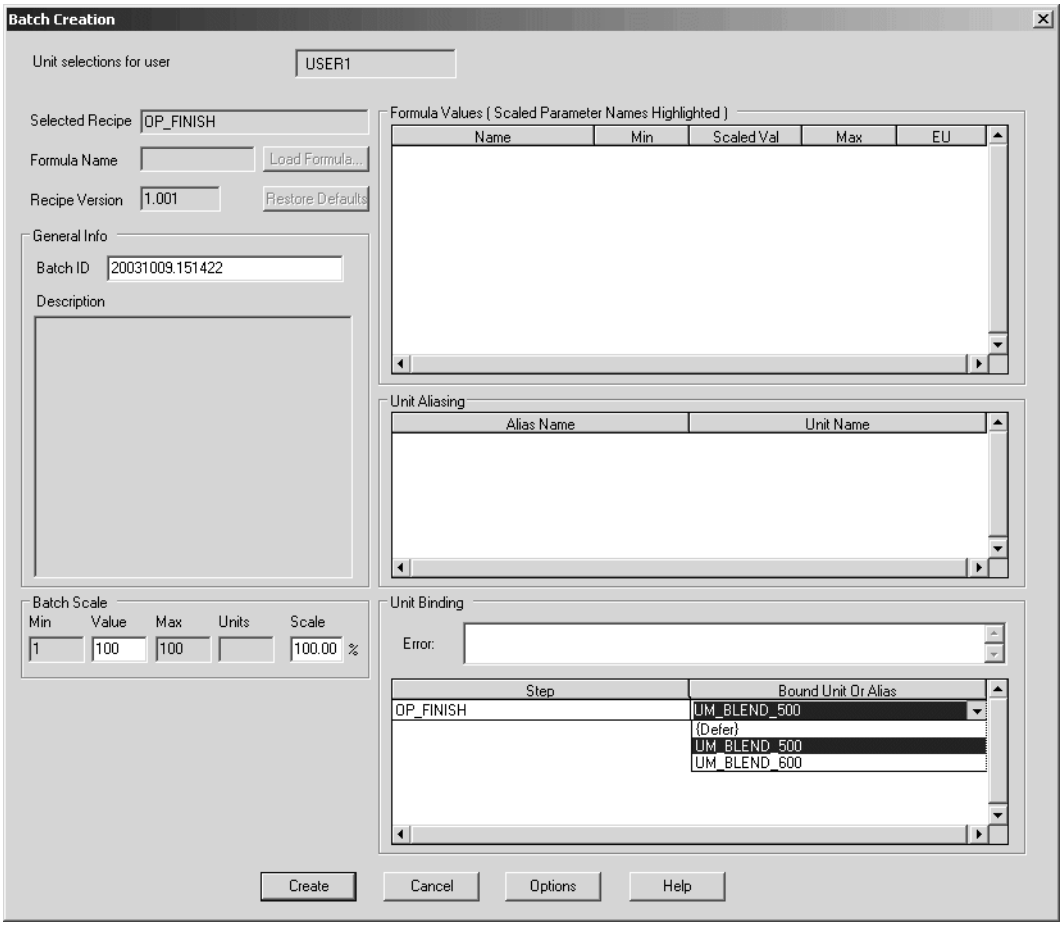
1 Launch the Batch Operator Interface by clicking **Start | DeltaV | Operator | Batch Operator Interface**.

2 If necessary, select the workstation for the desired Batch Executive. The Batch List screen opens.

3 Click the Add Batch button  .

4 Select OP_FINISH from the list of available recipes. (Only those recipes specifically assigned to a Batch Executive will appear in this list.)

The Batch Creation dialog opens.



5 The default Batch ID is the date and time. You can leave it as is or edit the ID by typing an appropriate Batch ID (alphanumeric). Since this is a class-based recipe, the Bound Unit has a drop-down list from which you can select the blender you want to use. Select UM_BLEND_500.

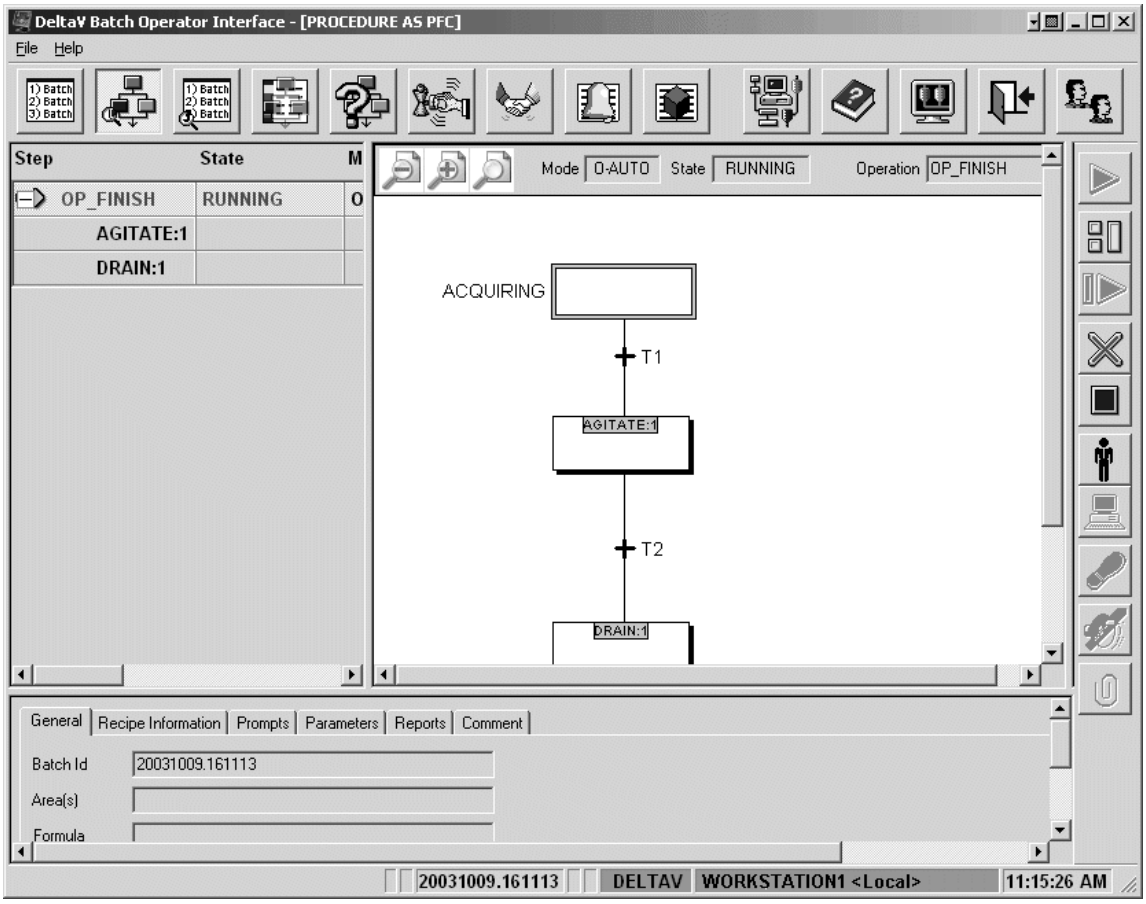
6 Click Create to add this batch to the batch list.

7 Select the pending operation you just created and click the Start Batch button  .

8 Click OK when asked if you want to start the batch.

9 Verify from the Batch List that the batch state indicates RUNNING.

10 Click the Procedure as PFC button  on the toolbar to see how the batch is progressing.



11 Click the Phase Control button  to see the information on this view.

12 On the Phase Control screen, click Display Process Cells, then Display Units and then click on the unit UM_BLEND_500 to view the details on this screen.

13 Return to the Batch List display by clicking the first button on the toolbar and confirm that the operation goes to Complete.

Hint If you encounter a failure, make sure that none of the failure conditions have been tripped. Fix the condition that caused the failure, clear the failure, and restart the batch.

On Your Own

If you have created or imported all the unit modules and recipes, you can now experiment on your own. You may want to do the following:

- Add and start the batch PRC_PAINT using different combinations of color tanks and blenders to see how the batch progresses and how equipment is allocated during the running of the batch.
- Trip and clear error conditions for the AGITATE phase.
- Hold and restart the batches at various points.
- Switch back and forth between different views in the Batch Operator Interface and go to different pictures in DeltaV Operate to watch the batch run and to see the types of information available on the different screens. You can resize the windows so both are visible at one time.
- Add a batch, leaving the unit binding as Defer. Then watch for the Unacknowledged Prompts button to flash. Specify equipment from the prompt.

When you are done, make sure all the unit modules are reset and under DeltaV Batch control and that all error conditions are cleared.

Formulas and Deferred Parameters

A recipe **formula** is made up of a set of variables used to control process values such as time, temperature, and quantities of ingredients. By including a formula in a recipe, you can use the same recipe to make variations of a product. This is especially useful if the same sequence of events will be used every time but the variables used by the logic will be different. In our example, we go through the same steps every time we want to make a batch of paint: add base, add color, mix, and drain. However, we can make different colors or shades by altering which color is added and how much base and color are used. Formulas can override the default input parameter values in a recipe and let you specify any value you need depending on the batch to be produced.

When setting up formulas for your recipes you can:

1 Create recipe parameters at the operation, unit procedure, or procedure level. (Create the parameter at the lowest level at which it is used.) The type of value at the different levels must match. That is, if the phase input parameter to be deferred is an integer, then the recipe input parameter must also be an integer.

2 Modify individual phase or recipe parameters to defer to a higher level in the recipe to obtain a value for the parameter. (The higher level will have formulas that provide values for the parameters.)

3 Define formulas for different combinations of values to be used by deferred parameters. Parameter values in formulas can be easily selected and modified at batch creation time. Formulas can be created at any recipe level that gets assigned to a Batch Executive.

Deferring Parameters

A deferred parameter is a process input value that is passed up one recipe level. By deferring a parameter, you instruct a recipe to retrieve the parameter's value from a formula. By creating a chain of parameters, you input the desired values at a high level (typically the procedure level) and pass the values all the way down to the phase. This approach lets you use generic operations that are shared by multiple recipes.

For the paint application, we want to be able to vary the values for three phase-level batch input parameters by creating formulas at the procedure level. The parameters we want to vary are as follows:

- the amount of base to be charged (parameter CHARGE_AMT in the CHG_BASE phase)
- the amount of color to be charged (parameter CHARGE_AMT in the CHG_COLOR phase)
- the agitation time (parameter AGITATE_TIME in the AGITATE phase)

To be able to specify the values for these parameters at the procedure level, we need to create equivalent parameters at the lower recipe levels (operation, and unit procedure) and defer them up to the top level. (Refer to the Batch Recipe Creation topic for a reminder of the recipe hierarchy in the paint application.)

We will start by creating an operation level parameter for the amount of base to be charged. The lowest recipe level at which this parameter is used is OP_CHARGE. (OP_CHARGE contains the phases CHG_BASE as well as CHG_COLOR.)

Creating Deferred Parameters

There are two ways to create a recipe parameter and defer the lower level parameter to the new higher level parameter:

- Create the higher level parameter directly at its recipe level and then modify the lower level parameter to defer to the higher level.

  or

- Defer the lower level parameter and let the system automatically create the higher level parameter.

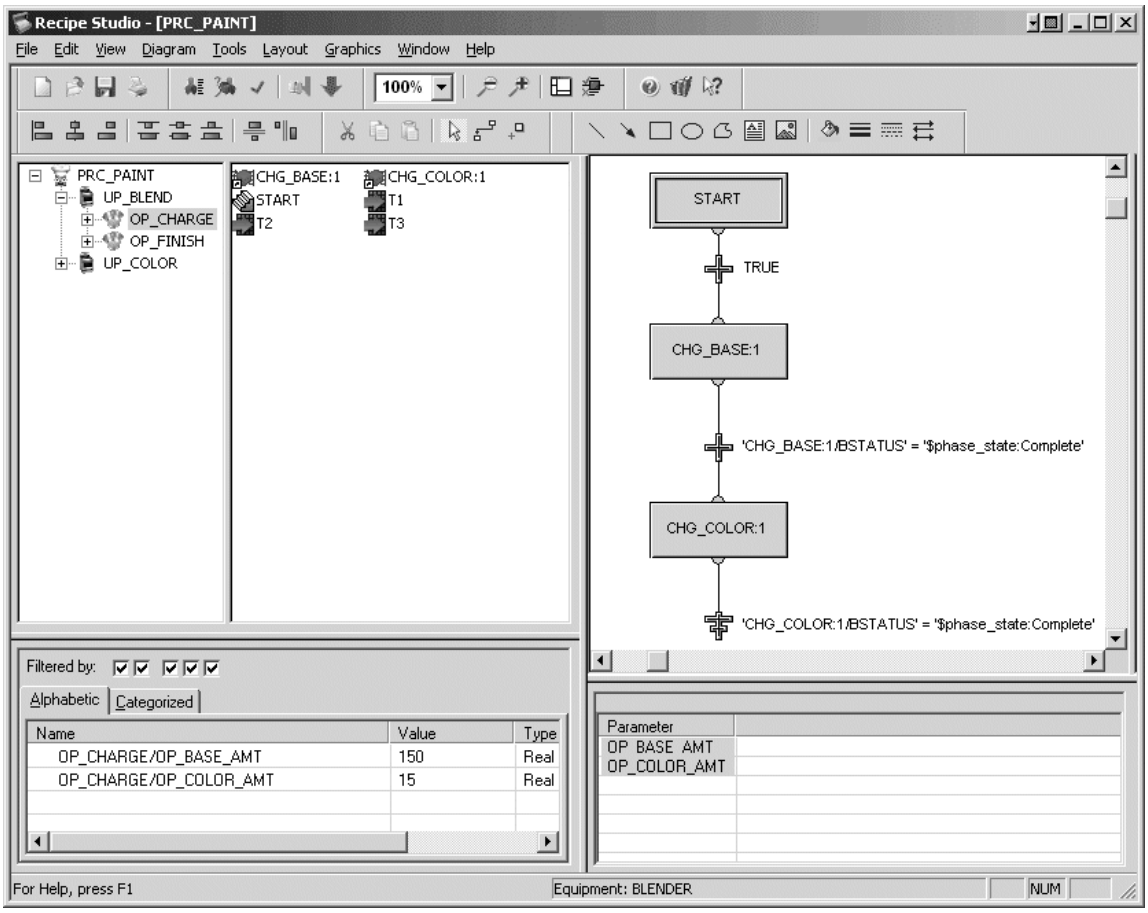
The second method, which is easier, is used in the following steps.

Note Recipe level parameters can have the same names or different names as parameters at the phase level. Both naming conventions are used in the following examples. For illustration purposes, the charge amount parameters will be given a new name at each new level (prefaced by OP, UP, or PRC). However, the agitate parameter, named AGITATE_TIME at the phase level, will retain the same name in all three recipe level parameters.

To defer a phase parameter and create an operation level parameter

1 Open the procedure PRC_PAINT in Recipe Studio.

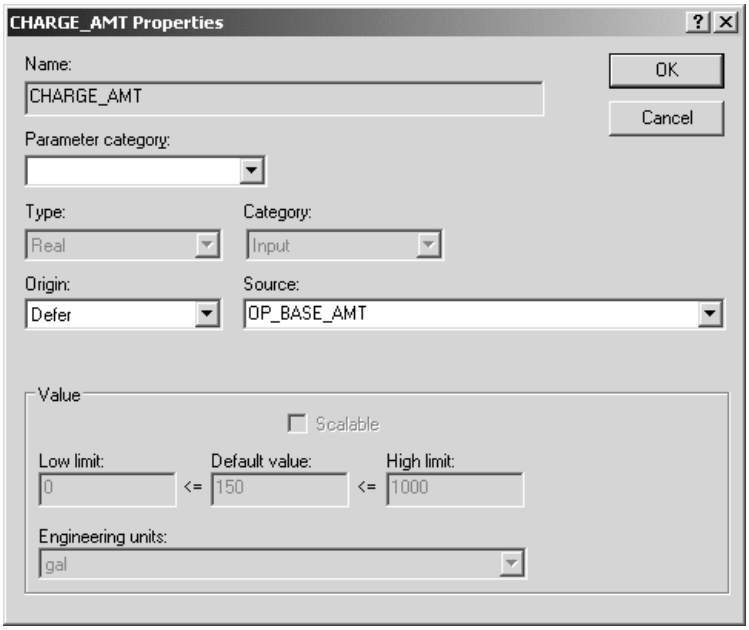
2 In the hierarchy view in the upper left corner, expand PRC_PAINT and UP_BLEND and select the operation OP_CHARGE.



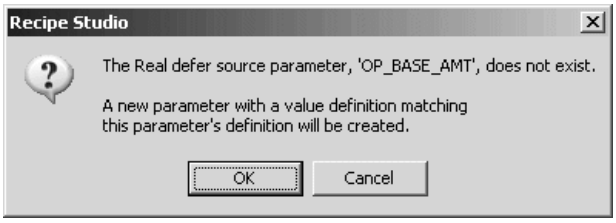
3 In the Diagram view, click the step named CHG_BASE:1 to select the phase.

4 In the Parameter view, in the lower left corner, right-click the phase parameter CHG_BASE:1/CHARGE_AMT and select Properties from the context menu. The parameter Properties dialog box opens.

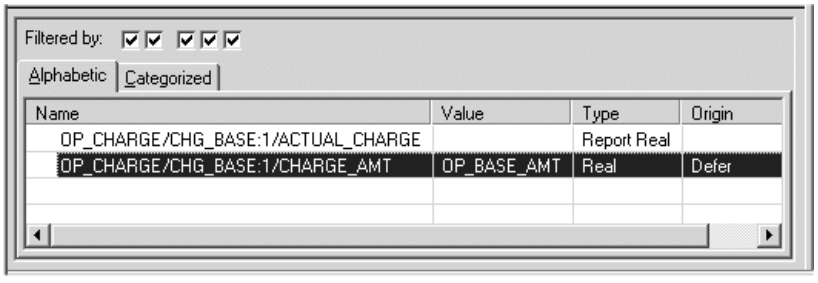
5 Change the Origin to Defer and type OP_BASE_AMT as the operation-level parameter to be used as the source for the phase parameter value.



6 Click OK when the system displays the following message about creating the parameter. Since you haven't created the parameter directly, the system will do it for you.



7 Note that in the Parameter view, with the CHG_BASE phase selected, the Origin field for the CHARGE_AMT parameter now says Defer and the Value field specifies OP_BASE_AMT.



8 To create a parameter for the charge amount for color, follow steps 3 through 7 above, except click the CHG_COLOR step and name the new parameter source for CHARGE_AMT to OP_COLOR_AMT.

9 To create a parameter for the agitate time, follow steps 2 through 7 above, except select the OP_FINISH operation in the hierarchy view, click the AGITATE step, modify the AGITATE_TIME parameter, and name the new parameter source AGITATE_TIME.

Now that we have the three process input parameters at the operation level, we need to create them at the unit procedure level and then the procedure level. We will do this the same way, by deferring the lower level parameters to the next higher level, which automatically creates the parameters at the higher level.

To create the parameters at the unit procedure and procedure levels

1 In the hierarchy view, in the upper left corner, select the unit procedure UP_BLEND.

2 In the diagram view, click the OP_CHARGE step.

Note You may need to press <F5> to refresh the screen to see the parameters.

---

3 In the parameter view, double-click the OP_BASE_AMT parameter to open the Properties dialog.

4 Change the Origin to Defer and the Source to UP_BASE_AMT.

5 Modify the OP_COLOR_AMT parameter to defer the source to UP_COLOR_AMT.

6 Click the OP_FINISH step.

7 Modify the AGITATE_TIME parameter to defer the source to AGITATE_TIME.

8 In the hierarchy view, select the procedure PRC_PAINT.

9 To create the procedure level parameters, click the UP_BLEND step.

10 Modify the UP_BASE_AMT parameter to defer the source to PRC_BASE_AMT.

11 Modify the UP_COLOR_AMT parameter to defer the source to PRC_COLOR_AMT.

12 Modify the AGITATE_TIME parameter to defer the source to AGITATE_TIME.

13 Save PRC_PAINT.

14 Download.

Defining Formulas

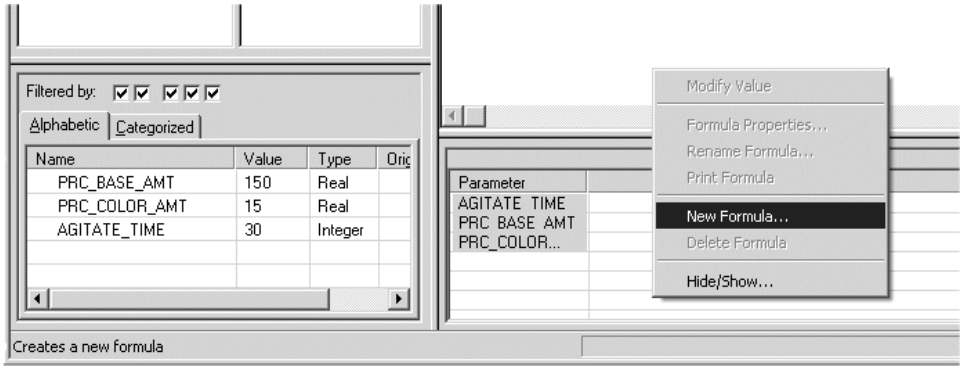
Typically, when you have a number of process input parameters for which you want to be able to specify values in a particular batch, you set up a recipe formula. For instance, if you want to have different values for small, medium, and large batches, you would set up three recipe formulas that specify the typical parameter values for each batch size. Then, when you create the batch, you simply pick the basic formula that you want to use and, if necessary, modify any or all parameter values. You can create a formula at any recipe level that gets assigned to a Batch Executive.

As a simple example, we will create three formulas for the procedure PRC_PAINT. The formulas will specify the parameter values for light, medium, and dark shades of paint. The light shades are most popular and are usually produced in larger batches. The formulas, parameters, and values are shown in the following table. (Values enclosed in angle brackets are the default values originally established for the batch input parameters.))

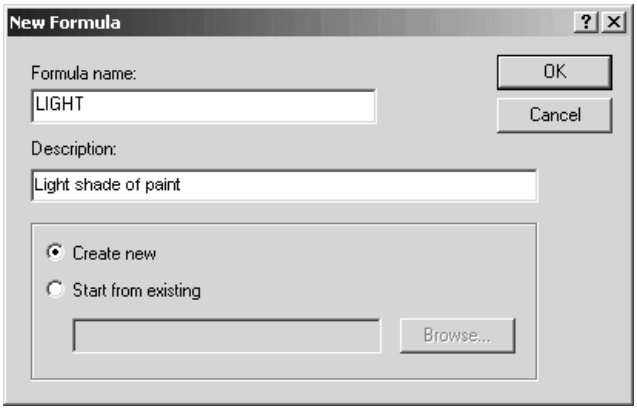
Parameter	Light	Medium	Dark
PRC_BASE_AMT	180	<150>	130
PRC_COLOR_AMT	<15>	<15>	20
AGITATE_TIME	50	<30>	40

To create a formula for PRC_PAINT

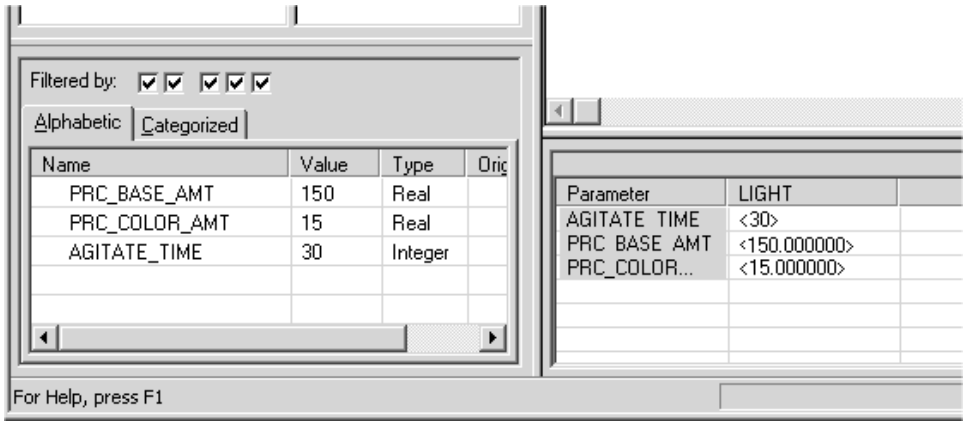
1 With nothing selected in the diagram view, right-click in the Formula view in the lower right corner, and select New Formula from the context menu.



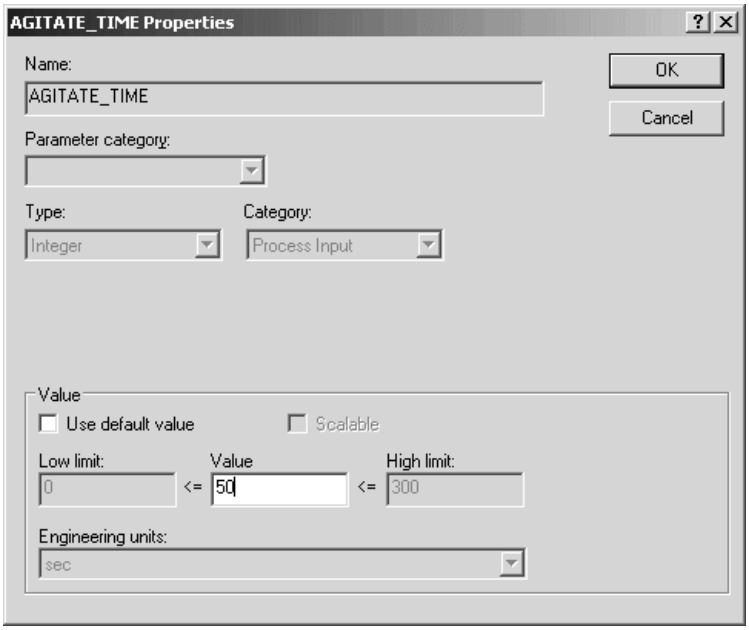
2 In the New Formula box, enter the name for the formula: LIGHT.



3 The formula named LIGHT is created in the formula view, with values set as the default values originally set in the definition of the phase input parameters for the three parameters. The default values are shown in angle brackets, < >. (You may need to press <F5> to refresh the screen to see the parameters.)



4 Right-click the value for AGITATE_TIME and select Modify Value from the context menu (or double-click to open the Properties dialog). Clear the Use default value check box and change the value to 50. Click OK.



Note that you can easily reset the value to the default by selecting the Use default value check box.

5 Change the remaining two parameter properties for the LIGHT formula, as shown in the following table. Then, add two more formulas--one for a MEDIUM batch (using the default values) and one for a DARK batch. Then save PRC_PAINT..

Parameter	Light	Medium	Dark
AGITATE_TIME	50	<30>	40
PRC_BASE_AMT	180	<150>	130
PRC_COLOR_AMT	<15>	<15>	20

Making a Formula Available to an Operator to Load

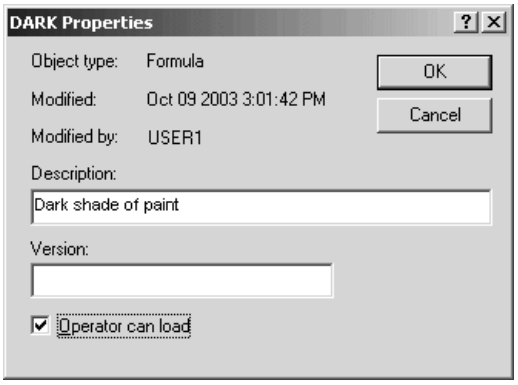
Before you can use the formula in a batch, you need to make it available to an operator to load. To open the Properties dialog for the formula, use one of the following two methods:

- Select one of the values in the formula, right-click, and select Formula Properties from the context menu.
- Select the formula in the DeltaV Explorer, right click, and select Properties from the context menu.

To make the formula available to an operator to load

1 Open the Properties dialog for the DARK formula.

2 Select the Operator can load check box and click OK.



3 Repeat steps 1 and 2 for the LIGHT and MEDIUM formulas.

4 Save and download the recipe.

Note Whenever you make changes to a recipe, you need to save and download it before you can use it.

Testing Recipe Formulas

Now that you have created three recipe formulas, you can go to Batch Operator Interface to see how they can be used in a batch.

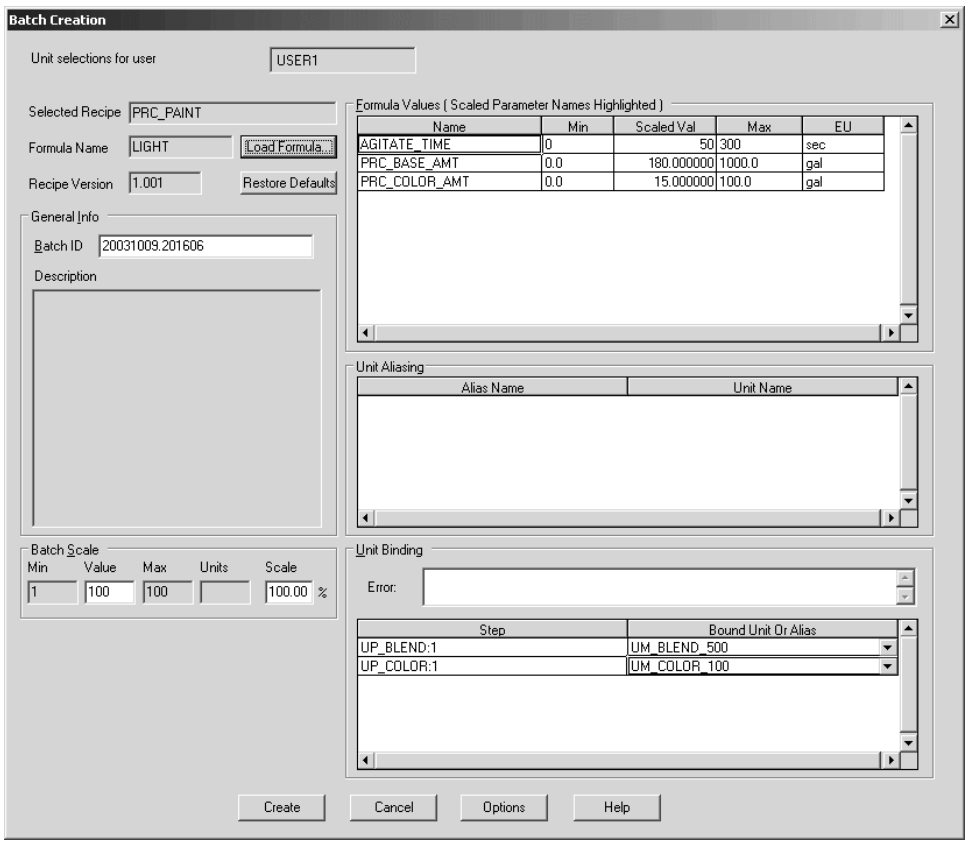
To test the recipe formulas

1 Open Batch Operator Interface. (Remember to start the Batch Executive first if it isn't already running in the background.)

2 Click the Add Batch button and select PRC_PAINT from the list of recipes that have been assigned to a Batch Executive.

3 In the Unit Binding section, select UM_BLEND_500 for UP_BLEND and UM_COLOR_100 for UP_COLOR.

4 In the Batch Creation dialog, click Load Formula and select the LIGHT formula.



The formula values for the LIGHT formula are listed.

5 You can accept the values you originally entered for this formula, or you can modify any value by double-clicking and editing it. If you modify values, notice that the Formula Name changes from LIGHT to CUSTOM FORMULA.

6 Click Create to add the batch to the batch list.

7 Select the batch on the batch list and click the Start Batch button.

8 When the batch has completed, close Batch Operator Interface.

Operator Prompts and Dynamic References

Many batch processes need to allow for interaction with an operator at certain places in the recipe. An operator prompt is a type of message to the operator that requires a response before the phase will continue. After receiving and evaluating the response, the phase will continue based on the configured logic.

Phase messages for the operator are displayed on the Batch Operator Interface in several places. All active messages appear on the Unacknowledged Prompts screen until the operator acknowledges them. Messages can be viewed from the Active Phase Summary screen, the Phase Control screen, the Procedure as PFC screen and the Procedure as Table screen; they can also be displayed in DeltaV Operate. When a phase is executed manually, messages will not appear on the Procedure as PFC or Procedure as Table screens since these views are not active for manual phases. DeltaV Operate graphics can also be configured to display batch operator prompts. This enables the operator to acknowledge prompts while still maintaining a view of the process equipment.

The following steps are used to create the prompt, receive the response, and evaluate the input to determine the course of action:

1 Create the phase message in the DeltaV Explorer as part of the phase class definition. For example, the message "Send this batch to product or waste?" could be created as the first phase message with a message ID of 1, and the second message "How many gallons would you like to release?" could be created with a message ID of 2.

2 In the run logic of the phase, create an operator prompt (using the REQUEST phase parameter) and specify the ID of the phase message that should be displayed to the operator. The phase request codes for operator prompts identify the type of response you are expecting (32*nn* for integer, 33*nn* for floating point, 34*nn* for Boolean, or 35*nn* for string-where *nn* in each case is the message ID). The operator responses will be written to the appropriate parameters PROMPT_INT, PROMPT_FLOAT, PROMPT_BOOL, and PROMPT_STRING and can be used in transition conditions to determine how to proceed in the logic. These parameters hold the value temporarily; for example, a new PROMPT_BOOL response will overwrite the existing one. Therefore, if you are going to send multiple prompts to the operator, you may want to save certain responses in temporary variables to use them later in the logic.

3 Configure the phase logic to continue based on the operator response.

Using a dynamic reference lets you change the value of a phase algorithm parameter during execution of the phase. The information may come from an operator entry, a recipe parameter passed from the Batch Executive, or a run-time value of a control variable. For instance, you could configure the phase to open a valve, which is referenced using a dynamic reference parameter. Then you can specify the valve you want to use at run-time by prompting the operator or by reading a value passed as part of the recipe.

Overview of Changes to Paint Application

In the paint application we will now add an operator prompt to ask the operator if the batch should be sent to Product storage or to Waste. If the response is Product, the operator will be asked how many gallons; if Waste, the entire contents will be sent to Waste.

In the example, we will request operator responses in two formats: Boolean and floating point.

Following is an outline of the steps we will follow to implement these changes. Subsequent topics give more details.

1 In the DeltaV Explorer, create the two phase messages that will be used in the DRAIN phase logic.

2 In Control Studio, modify the DRAIN running logic to add a request to send a Boolean operator prompt to ask whether the batch should be sent to Product or to Waste.

3 Add transitions to branch depending on whether the response is Product (Yes) or Waste (No).

4 If the response is Waste, set the BLENDER_OUTLET command to drain to waste. If the response is Product, ask how many gallons are to be released to product, then set the BLENDER_OUTLET command to release to product.

5 Add a step to record a LOGEVENT that can be viewed as an event record in the Process History View application.

Creating Phase Messages

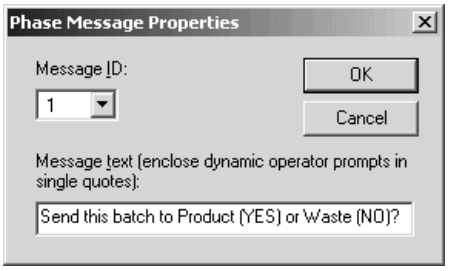
The first step in defining an operator prompt is to create the actual message that will appear on the Batch Operator Interface screen.

To create the phase messages

1 In DeltaV Explorer, select the DRAIN phase class.

2 Right-click and select New | Phase Message from the context menu.

3 In the Phase Message Properties box enter the message text as shown and click OK.



Note that the message ID is automatically assigned number 1.

4 Create message 2 using the following message text:

How many gallons would you like to release?

Adding Operator Prompts

Operator prompts are created within the phase logic at the phase class level. When you are ready to insert an operator prompt into the phase logic, you need to decide what type of message (Integer, Floating Point, Boolean, or String) the prompt needs to be. The type of message you send will determine not only the format of the response you expect to receive but also the code you will use to send the message. Other phase request codes are listed in the Batch Reference in the Phase Request Codes topic.

Following are the codes used in phase logic requests to send operator messages, including prompts, which require a response.

Type of Message	Request Code (nn is message ID)
Send message (no response required)	30nn
Clear message	3100 (REQDATA1=ID of message)
Send operator prompt, Integer	32nn
Send operator prompt, Floating Point	33nn
Send operator prompt, Boolean	34nn
Send operator prompt, String	35nn

The Unacknowledged Prompts button  (in the button bar at the top of every operator screen in the Batch Operator Interface) flashes to alert the operator when a phase is sending a message to the operator. The operator clicks this button to display the Unacknowledged Prompts screen. Double-clicking the message line opens a dialog box in which the operator can enter a response.

Note Remember that there are several other places in the Batch Operator Interface where the operator can view active messages or prompts and that DeltaV Operate graphics can also be configured to display such messages and prompts. Any of these locations will allow the operator to respond to prompts as well.

We will add steps and transitions to ask the operator the question about where to send the batch. The first step will be inserted into the existing running logic after the first transition, before the CLOSE_STATE step.

To add an operator prompt

1 In Control Studio, open the DRAIN phase class and drill down into the running logic.

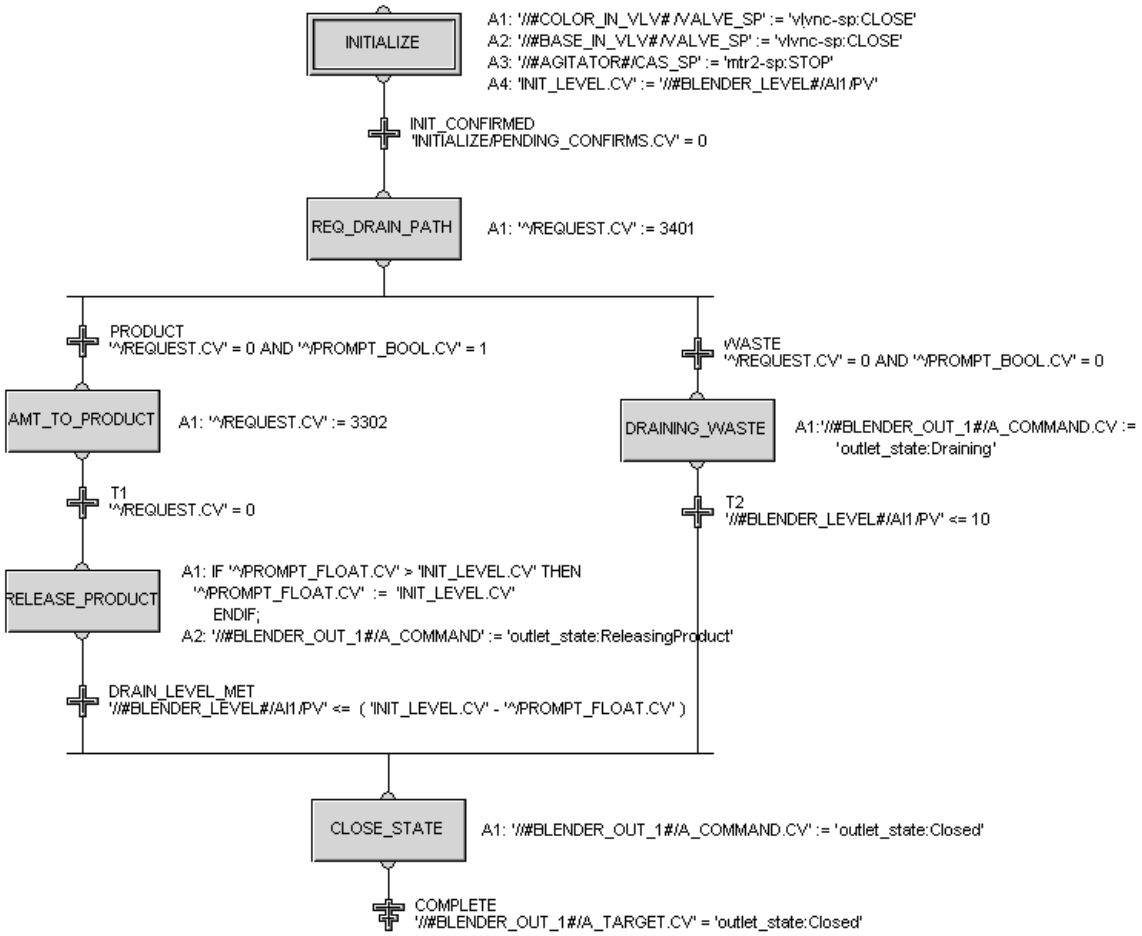
2 Delete the line after the INIT_CONFIRMED condition and move the last step and transition down enough to allow room for modifying the logic as shown in the following diagram.

3 Drag and drop steps and transitions from the palette to the diagram to create the logic shown in the figure below.

4 Connect the steps and transitions as shown.

5 Configure the expressions for the step actions and transitions using the Expression Editor.

6 Save the DRAIN phase class and download the phase.



The phase logic is now branched to do different actions depending on the responses to the operator prompts.

In the expression in the REQ_DRAIN_PATH step ('^/REQUEST.CV' := 3401), 34 is the code for sending a Boolean (yes/no) prompt and 01 is the ID of the message to be displayed: Send this batch to Product (YES) or Waste (NO). The operator's response is stored in the PROMPT_BOOL parameter.

The expression for the NO branch is:

```
'^/REQUEST.CV' = 0 AND '^/PROMPT_BOOL.CV' = 0
```

This will check to make sure that the operator has responded to the prompt (REQUEST.CV = 0). If the response is NO (PROMPT_BOOL = 0), then the logic will continue through this branch.

The expression for the YES branch is:

```
'^/REQUEST.CV' = 0 AND '^/PROMPT_BOOL.CV' = 1
```

Again, this checks to make sure that the operator has responded to the prompt (REQUEST.CV = 0). If the response is YES (PROMPT_BOOL = 1), then the logic will continue through this branch.

Adding a LOGEVENT Record

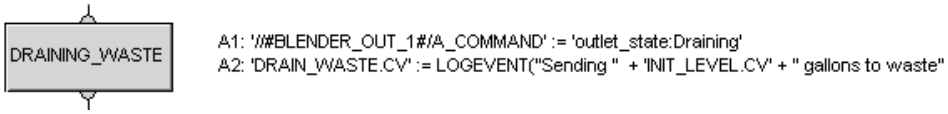
The DeltaV Process History application displays real-time and historical data for all modules selected for trending. Module trends are plotted on a graph, and events are displayed in a tabular (grid) format. All event records contain fields that are displayed in columns, including Date/Time, Event Type, Category, Area, Node, Module, and two Description fields. You can use the LOGEVENT button to add an action to the phase logic to record a message in the Desc2 field of an event record.

Note You will need to add a new phase algorithm parameter to the run logic of the DRAIN phase class. Name the parameter DRAIN_WASTE and make the type STRING. Leave the value blank (null).

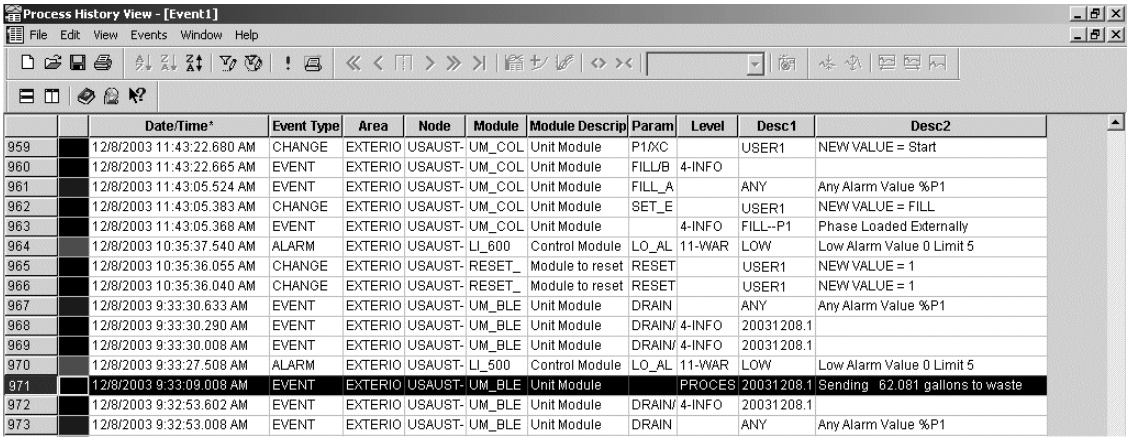
Add the following action to the DRAINING_WASTE step:

```
'DRAIN_WASTE.CV' := LOGEVENT ("Sending " + 'INIT_LEVEL.CV' + "gallons to
waste")
```

The DRAINING_WASTE step now has two actions, with the Logevent being the second action.



This action adds an event record, as shown in the following figure.



		Date/Time*	Event Type	Area	Node	Module	Module Descrip	Param	Level	Desc1	Desc2
959		12/8/2003 11:43:22.680 AM	CHANGE	EXTERIO	USAUST-	UM_COL	Unit Module	P1/XC		USER1	NEW VALUE = Start
960		12/8/2003 11:43:22.665 AM	EVENT	EXTERIO	USAUST-	UM_COL	Unit Module	FILL/B	4-INFO		
961		12/8/2003 11:43:05.524 AM	EVENT	EXTERIO	USAUST-	UM_COL	Unit Module	FILL_A		ANY	Any Alarm Value %P1
962		12/8/2003 11:43:05.383 AM	CHANGE	EXTERIO	USAUST-	UM_COL	Unit Module	SET_E		USER1	NEW VALUE = FILL
963		12/8/2003 11:43:05.368 AM	EVENT	EXTERIO	USAUST-	UM_COL	Unit Module		4-INFO	FILL--P1	Phase Loaded Externally
964		12/8/2003 10:35:37.540 AM	ALARM	EXTERIO	USAUST-	LI_600	Control Module	LO_AL	11-WAR	LOW	Low Alarm Value 0 Limit 5
965		12/8/2003 10:35:36.055 AM	CHANGE	EXTERIO	USAUST-	RESET_	Module to reset	RESET		USER1	NEW VALUE = 1
966		12/8/2003 10:35:36.040 AM	CHANGE	EXTERIO	USAUST-	RESET_	Module to reset	RESET		USER1	NEW VALUE = 1
967		12/8/2003 9:33:30.633 AM	EVENT	EXTERIO	USAUST-	UM_BLE	Unit Module	DRAIN		ANY	Any Alarm Value %P1
968		12/8/2003 9:33:30.290 AM	EVENT	EXTERIO	USAUST-	UM_BLE	Unit Module	DRAIN/	4-INFO	20031208.1	
969		12/8/2003 9:33:30.008 AM	EVENT	EXTERIO	USAUST-	UM_BLE	Unit Module	DRAIN/	4-INFO	20031208.1	
970		12/8/2003 9:33:27.508 AM	ALARM	EXTERIO	USAUST-	LI_500	Control Module	LO_AL	11-WAR	LOW	Low Alarm Value 0 Limit 5
971		12/8/2003 9:33:09.008 AM	EVENT	EXTERIO	USAUST-	UM_BLE	Unit Module		PROCES	20031208.1	Sending 62.081 gallons to waste
972		12/8/2003 9:32:53.602 AM	EVENT	EXTERIO	USAUST-	UM_BLE	Unit Module	DRAIN/	4-INFO	20031208.1	
973		12/8/2003 9:32:53.008 AM	EVENT	EXTERIO	USAUST-	UM_BLE	Unit Module	DRAIN		ANY	Any Alarm Value %P1

Testing the Revised DRAIN Phase

To test the operator prompt, you will need to open the Batch Operator Interface. You might also want to open DeltaV Operate to see if all the valves are closed and that the process is otherwise in a state of readiness. With both operator interfaces open, you can switch back and forth to run the batch recipes and watch the batch progress on the Process picture.

You can resize the operator interface screens so you can see the important parts of both at one time. You might even want to open the unit module UM_BLEND_500 in Control Studio. Select the DRAIN running logic and put it in

online mode so you can see the phase progress through the steps of the Sequential Function Chart. (This is useful if you encounter any errors that you need to fix.)

Note In the following procedure, the Batch Executive is stopped before downloading. For more information on the types of configuration changes that require stopping and restarting the Batch Executive, see the Effects of Configuration Changes topic in the Batch Reference book.

To test the revised DRAIN phase

1 Open the DeltaV Explorer and navigate to the Assigned Recipes object under the Batch Executive.

2 Make sure that OP_FINISH has been assigned to this Batch Executive so we can choose it from the available list of recipes in the Batch Operator Interface.

3 Stop the Batch Executive (if running).

4 Download the controller or workstation. This will download all changes to the phases.

5 Start the Batch Executive and verify phases.

6 Open DeltaV Operate to the Process picture.

7 If there is not any level in the blender, you may need to run the CHG_BASE phase.

8 Make sure all the valves are closed.

9 Open the Batch Operator Interface.

10 Click Add Batch and select OP_FINISH from the list of recipes assigned to the Batch Executive. Bind to the Blender UM_BLEND_500.

11 Select the batch and click the Start button.

12 When the DRAIN phase starts, the Unacknowledged Operator Prompts button in the button bar starts to flash on and off.

13 Click the Unacknowledged Operator Prompts button. A new screen opens with a list of unacknowledged operator messages (which should only contain one message).

14 Double-click the message line to open a dialog box to enter a response to the message **Send this batch to Product (YES) or Waste (NO)?**.

15 Select Yes.

16 Another prompt appears to ask the number of gallons to drain.

17 Type in the number of gallons you want to drain. Make sure that this number is less than or equal to the level in the blender.

18 At this point, you may want to go to the Process picture to watch the draining of the blender.

19 If the blender is not empty, you can run OP_FINISH again and send the rest of the batch to waste to verify that portion of your logic.

Conclusion

The exercises in this tutorial demonstrate some of the basic techniques for using the DeltaV Batch subsystem to create a simple batch application. If you have done the exercises online, you now have some hands-on experience in these areas:

- Configuring batch equipment
- Configuring phase logic
- Creating and configuring unit modules
- Arbitrating equipment so recipes can acquire and release necessary resources
- Coordinating phases that need to be synchronized
- Creating recipes and recipe formulas
- Running batches from the batch operator's point of view

You can use the paint application as a basis for further learning on your own. Try rewriting parts of the phase logic to increase your proficiency. Experiment with other DeltaV Batch subsystem features to get a better grasp of the power and versatility of this system. To learn more about the DeltaV Batch subsystem, refer to the Batch Reference manual or access specific topics in the DeltaV system online help.

Index

D

E

F

Q

R

S

T

U

V

W