



# FactoryTalk Batch PC-Based Phase Programming Reference Manual

Publication# BATCH-RM006B-EN-D - May 2022

Supersedes Publication# BATCH-RM006A-EN-D - October 2020



## Important User Information

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

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

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

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

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

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

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

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



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



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

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

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



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



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



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

Rockwell Automation recognizes that some of the terms that are currently used in our industry and in this publication are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

|  |  |    |
|--|--|----|
| <b>Preface</b>                         | About this manual.....   | 5  |
|  | Legal Notices.....   | 5  |
|  | Additional resources.....  | 6  |
|  | <b>Chapter 1</b>   |    |
| <b>PC-based phases</b>                 | PC-based phase overview.....   | 9  |
|  | PC-based phases.....   | 9  |
|  | PC-based phase execution .....                                       | 11 |
|  | <b>Chapter 2</b>   |    |
| <b>Create a PC-based phase</b>         | Add the Batch Phase Execution control to a Visual Basic project..... | 13 |
|  | Place the Batch Phase Execution control on a form.....               | 13 |
|  | PC-based phase properties .....                                      | 14 |
|  | Set custom properties in the Property Pages dialog box.....          | 14 |
|  | Program the PC-based phase.....                                      | 15 |
|  | Methods.....   | 15 |
|  | Events.....  | 16 |
|  | Event example using OnRestart event .....                            | 16 |
|  | Create the executable .....  | 17 |
|  | Add the phase execution control to a Visual C++ project.....         | 17 |
|  | Create a new project.....  | 17 |
|  | Add the control to the project .....                                 | 17 |
|  | Add the control to the dialog .....                                  | 18 |
|  | Access control properties from Visual C++.....                       | 18 |
|  | Configure the BaseTagName property .....                             | 18 |
|  | Add control events.....  | 18 |
|  | Add a member variable derived from the Phase Control Class.....      | 19 |
|  | Add phase logic to event handlers .....                              | 19 |
|  | General usage signature requests .....                               | 19 |
|  | Parameter and report parameter limits.....                           | 19 |
|  | Distribute PC-based phase applications .....                         | 20 |
|  | <b>Chapter 3</b>   |    |
| <b>Use a PC-based phase</b>            | Update the area model.....   | 21 |
|  | Change the default tag separators for a PC-based phase .....         | 22 |
|  | Set tag items to default names .....                                 | 23 |
|  | Add the new phase to a recipe.....                                   | 24 |
|  | Run a recipe with a PC-based phase.....                              | 24 |
|  | <b>Appendix A</b>  |    |
| <b>Properties, methods, and events</b> | Properties.....  | 25 |

Methods.....29  
Events ..... 31

**Appendix B**

**Application notes and example phase**

Application notes..... 33  
PC-based phase example ..... 34  
Create a new PC-based phase..... 36  
    Add a new data server definition to an area model ..... 37  
    Use limit tags on a PC-based phase..... 37  
    Add limit tags to a PC-based phase .....38

**Index**

### About this manual

This manual provides usage instructions for the FactoryTalk Batch PC-Based Phase programming. It is one of a set of related manuals that describe installing, programming, and operating the FactoryTalk Batch system.

To review FactoryTalk Batch release notes and latest information regarding product compatibility refer to the [Product Compatibility and Download Center \(PCDC\)](#).

### Legal Notices

Rockwell Automation publishes legal notices, such as privacy policies, license agreements, trademark disclosures, and other terms and conditions on the [Legal Notices](#) page of the Rockwell Automation website.

### End User License Agreement (EULA)

You can view the Rockwell Automation End User License Agreement (EULA) by opening the license.rtf file located in your product's install folder.

The default location of this file is:

C:\Program Files (x86)\Common Files\Rockwell\license.rtf

### Open Source Software Licenses

The software included in this product contains copyrighted software that is licensed under one or more open-source licenses.

You can view a full list of all open-source software used in this product and their corresponding licenses by opening the oss\_license.txt file located your product's OPENSOURCE folder on your hard drive. This file is divided into these sections:

- **Components**  
Includes the name of the open-source component, its version number, and the type of license.
- **Copyright Text**  
Includes the name of the open-source component, its version number, and the copyright declaration.
- **Licenses**  
Includes the name of the license, the list of open-source components citing the license, and the terms of the license.

The default location of this file is:

C:\Program Files (x86)\Common Files\Rockwell\Help\<product name>\Release Notes\OPENSOURCE\oss\_licenses.txt

You may obtain Corresponding Source code for open-source packages included in this product from their respective project web site(s).

Alternatively, you may obtain complete Corresponding Source code by contacting Rockwell Automation via the **Contact** form on the Rockwell Automation website:

<http://www.rockwellautomation.com/global/about-us/contact/contact.page>.

Please include "Open Source" as part of the request text.

## Additional resources

Following is a comprehensive list of documentation for the FactoryTalk® Batch products from Rockwell Automation.

### Installation, Quick Start, and Getting Results Guides

| Resource  | Description  |
|---|--|
| FactoryTalk Batch Components Installation and Upgrade Guide (BATCH-IN002) | Provides information and procedures for FactoryTalk Batch system installation. Includes information for FactoryTalk Batch Material Manager, FactoryTalk Event Archiver, and associated FactoryTalk Batch Client and Server components. |
| FactoryTalk Batch View Quick Start Guide (FTBVS-QS001)                    | Provides information about using FactoryTalk Batch View to create, view, and command control recipes, acknowledge prompts and signatures, view equipment phases and diagnostic information, and view profile information.              |
| FactoryTalk Batch View HMI Controls Quick Start Guide (BATCH-QS001D)      | Provides a general overview of FactoryTalk Batch View HMI Controls.  |
| FactoryTalk Batch eProcedure® Getting Results Guide (BWEPRO-GR01)         | Explains the basics of FactoryTalk Batch eProcedure.   |
| FactoryTalk Batch Getting Results Guide (BATCH-GR01)                      | Introduces the basics of automated batch manufacturing and the FactoryTalk Batch product components.   |
| FactoryTalk Batch Material Manager Getting Results Guide (BWMTR-GR01)     | Introduces the basics of FactoryTalk Batch Material Manager.   |

### User Guides

| Resource  | Description  |
|---|--|
| FactoryTalk Batch Material Editor User Guide (BWMTR-UM001)  | Provides access to information and procedural instructions required to configure materials and the containers to hold them. The material data is stored in the material database, which is used to create material-based recipes. This information is intended as a reference for formulators. |
| FactoryTalk Batch Equipment Editor User Guide (BATCH-UM004) | Provides information on creating and maintaining an equipment database (area model). The area model is stored in a file with a .cfg file extension and is available to all other FactoryTalk Batch programs, including the Recipe Editor, View, and Phase Simulator.                           |

| Resource  | Description  |
|---|--|
| FactoryTalk Batch PhaseManager™ User Guide (BATCX-UM011)      | Describes the integration of the FactoryTalk Batch software with the Studio 5000 Logix Designer® application and the Logix 5000™ family of controllers. The integration simplifies the configuration and maintenance of the FactoryTalk Batch automation system, provides better communication between the FactoryTalk Batch Server and the Logix 5000 controller, and significantly reduces the programming effort required to develop the phase logic code that resides in your Logix 5000 controller. |
| FactoryTalk Batch Recipe Editor User Guide (BATCX-UM006)      | Provides instructions on using FactoryTalk Batch Recipe Editor to create and configure master recipes for use in batch automation. The interface is based on IEC 61131-3 sequential function charts to organize recipes graphically into procedures, unit procedures, operations, and phases. Build recipes using either the SFC format or a table-based format.   |
| FactoryTalk Batch View HMI Controls User Manual (FTBVS-UM003) | Provides details about using FactoryTalk Batch View HMI Controls to monitor and interact with the production process within a FactoryTalk View SE Display Client.  |
| FactoryTalk Batch View User Manual (FTBVS-UM002)              | Provides information and procedural instructions for using FactoryTalk Batch View in a modern and intuitive portal into a comprehensive batching solution for effective operations, leveraging its own web server using HTML5 technology to provide connectivity into a FactoryTalk Batch Server.  |
| FactoryTalk Event Archiver User Guide (BATCX-UM012)           | Provides information and instructions specific to the FactoryTalk Event Archiver. Intended for use by system administrators and production supervisors.  |

## Administrator Guides

| Resource  | Description  |
|---|--|
| FactoryTalk Batch Administrator Guide (BATCX-UM003)                   | Provides instructions for configuring security and services, and implementation and use of components not typically accessed or used by batch operators, such as the FactoryTalk Batch Server.   |
| FactoryTalk Batch eProcedure Administrator Guide (BWEPRO-UM011)       | Provides procedures specific to FactoryTalk Batch eProcedure, such as implementing security and configuring the user-defined area Active Server Page. Included are instructions for tasks specific to FactoryTalk Batch, such as configuring security and services to support FactoryTalk Batch eProcedure. Provides instructions on the implementation and use of components not typically accessed or used by batch operators, such as the FactoryTalk Batch Server. |
| FactoryTalk Batch Material Manager Administrator Guide (BWEPRO-UM011) | Provides information and instructions specific to FactoryTalk Batch Material Manager. Intended for use by system administrators and database administrators.   |

## Reference Guides

| Resource  | Description  |
|---|--|
| FactoryTalk Batch Material Server API Technical Reference (BWMTR-RM001) | Provides access to information regarding the interface between the FactoryTalk Batch Material Server and the FactoryTalk Batch Material Editor and FactoryTalk Batch. It is intended to be used as a reference information by custom interface developers. |
| FactoryTalk Batch PCD Programming Reference Manual (BATCX-RM004)        | Provides information and instructions about the FactoryTalk Batch PCD interface design. It is intended to be used as a reference guide for PCD programmers.  |

| <b>Resource</b>  | <b>Description</b>  |
|--|---|
| FactoryTalk Batch Server API Reference Manual (BATCH-RM003)                          | Provides information regarding the interface between the FactoryTalk Batch Server and FactoryTalk Batch View – the Server Application Programming Interface (API). It is intended to be used as a reference guide by custom interface developers. |
| FactoryTalk Batch System Files Reference Manual (BATCH-RM005)                        | Provides the technical information for configuration and maintenance of a FactoryTalk Batch system. It can be used as a reference information for implementation engineers and system administrators.   |
| FactoryTalk Batch eProcedure Instruction File Design Reference Manual (BWEPRO-RM001) | Includes information about the building of instruction files for use in equipment database creation and recipe development. This information is intended to be used as a reference by instruction file authors.                                   |

View or download publications at <http://www.rockwellautomation.com/literature>. To order paper copies of technical documentation, contact your local Allen-Bradley® distributor or sales representative.

## PC-based phases

This guide was developed to provide information and procedures required to create and use PC-based phases, and is intended to be used as a reference guide.

---

**IMPORTANT** DDE server is no longer supported. When upgrading from RSBatch 4.X or an earlier version, set all tags to use an OPC server. Also, due to a functional address change in batchphs.ocx, the tag separator has changed from an underscore (\_) to a period (.). Change the FactoryTalk Batch Equipment Editor tag separator to a period by editing the batcheqp.ini file.

---

### PC-based phase overview

The PC-Based Phase option is a control that allows you to design phases for use in FactoryTalk Batch recipes. Use these phases for a variety of tasks, such as operator prompts, timers and calculations. A PC-based phase communicates with the FactoryTalk Batch Server, and does not require a process-connected device. Use in place of phase logic to decrease the amount of memory required within a process-connected device.

PC-based phases use the **Batch Phase Execution** control, which encapsulates the FactoryTalk Batch phase logic interface and the OPC interface. The **Batch Phase Execution** control functions as a data server for PC-based phases and supports the full functionality of a FactoryTalk Batch phase. It is an SP88-aware component and is used to provide Level 2 control of the FactoryTalk Batch control system. The **Batch Phase Execution** control is a persistent component that supports automatic checkpointing. It can be accessed directly from within Visual Basic, Access, or any other Microsoft-compatible control container.



Tip: The Batch Phase Execution control is part of a group of controls found in the BATCHPHS.OCX file. When distributing your application, you may install the BATCHPHS.OCX file in any directory, as long as the control is properly registered.

### PC-based phases

A PC-based phase allows you to program a phase outside of a process-connected device (PCD). You can program the phase to meet your own unique requirements from within your Microsoft-compatible control container. The PC-based phase works the same way as any phase defined in a PCD, communicating with the FactoryTalk Batch Server through the **Batch Phase Execution** control. This saves your system's memory for processes requiring interaction with hardware.

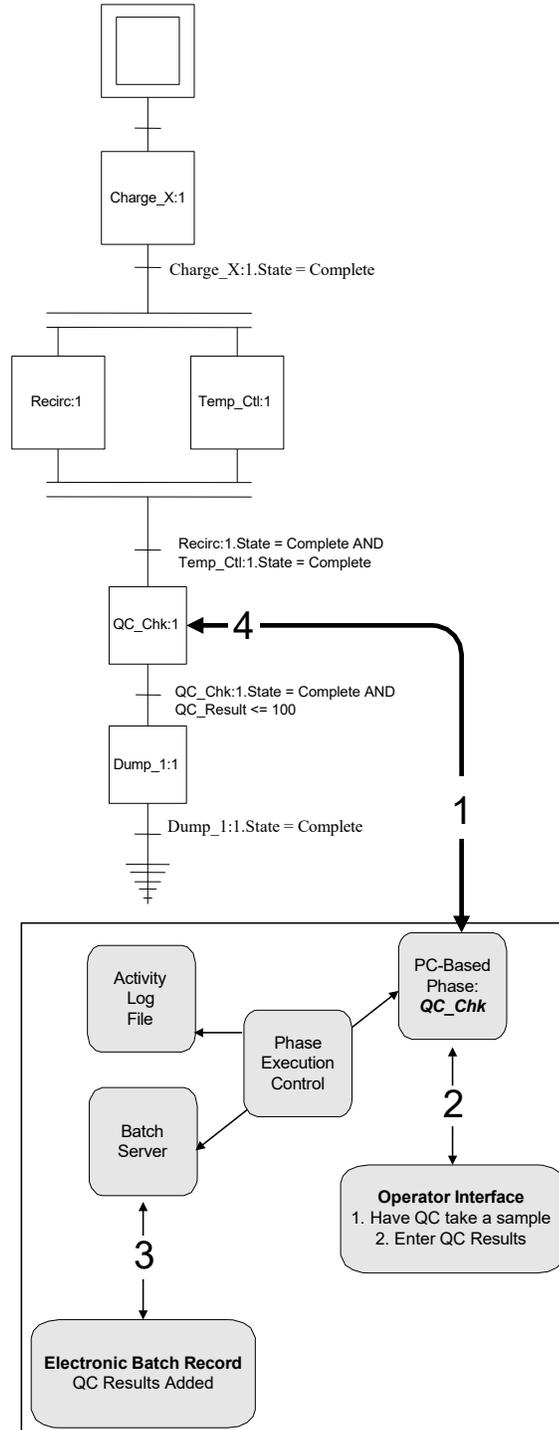
After a PC-based phase is created, the FactoryTalk Batch area model must be updated to include the new phase and the appropriate equipment modules. Because the **Batch Phase Execution** control behaves as a data server, a new

data server definition must be created. The data server definition is created using FactoryTalk Batch Equipment Editor, and defined using the **Batch Phase Execution** control's communication parameters. The phase can then be added to your recipes. The PC-based phase must remain active during recipe execution, allowing the phase to communicate with the FactoryTalk Batch Server through the **Batch Phase Execution** control.

An activity log file is created for each PC-based phase when it runs. This log file, **Singleton.log**, is written to the *WINDOWS\system32* directory.

## PC-based phase execution

The following is an example of a PC-based phase executing within a recipe. The PC-based phase is created to prompt the operator to take a sample to the **Quality** control and then enter the results. The phase communicates with the FactoryTalk Batch Server by way of the **Batch Phase Execution** control. The phase can direct the server to prompt the operator, wait for results and have the server store the result in the electronic batch record. The recipe only proceeds if the result is within the appropriate range.



1. As the batch is processed, the **QC\_Chk** phase is executed.
2. The **QC\_Chk** phase displays a screen to the operator, requesting that the operator have QC take an in-process sample. The **QC\_Chk** phase does not continue processing until the operator enters the QC result.
3. When the operator enters the result, the **QC\_Chk** phase sends the report value to the FactoryTalk Batch Server by way of the **Batch Phase Execution** control. The **Batch Phase Execution** control stores the value in the electronic batch record. While the phase is executing, an activity log file is updated with process events.
4. Control is returned to the recipe, and it continues processing the balance of the transitions and steps.

## Create a PC-based phase

A PC-based phase is created using the **Batch Phase Execution** control and an OPC container application. The control is added to the container application, placed in the appropriate medium (form, spreadsheet, document, and so on), and its properties, methods, and events defined.

The instructions included in this section relate to creating a PC-based phase within Visual Basic. If you choose to implement a PC-based phase in an application other than Visual Basic, on the Control Toolbox, click **More Controls** to retrieve the **Batch Phase Execution** control.

### Add the Batch Phase Execution control to a Visual Basic project

The **Batch Phase Execution** control must be added to the Visual Basic toolbox in order to make it available to your project. The instructions included here define the steps for manually adding the control to the toolbox after starting a new project. However, if you choose, you may configure the control to load automatically.

#### To add the Batch Phase Execution control to a Visual Basic project

1. Start Visual Basic.
2. From the **Projects** menu, select **Components**.
3. On the **Components** dialog box, select the **Controls** tab. The available Custom Controls are displayed in a list.
4. Scrolling through the list, locate and select the **Batch Phase Execution** control.
5. Click **OK** to close the dialog box. The **Batch Phase Execution** control is available on the Visual Basic toolbox. It is marked with a picture of a valve.

### Place the Batch Phase Execution control on a form

Use Visual Basic to place the **Batch Phase Execution** control on a form.

#### To place the Batch Phase Execution control on a form:

1. Select the **Batch Phase Execution** control in the Visual Basic toolbox.
2. Position the cursor on the form where you want the control.
3. Click and drag to draw the outline of the control on the form.
4. Release the outline to place the new **Batch Phase Execution** control at the location specified. The PC-based phase can now be programmed.

## PC-based phase properties

To enable communication between a PC-based phase and the FactoryTalk Batch Server, properties can be modified in two ways:

- With the phase selected, use the **Properties** window to locate and set the appropriate properties, or
- Use the custom **Property Pages** dialog box.

All properties specific to the **Batch Phase Execution** control that can be configured at design time are included in this dialog box.

## Set custom properties in the Property Pages dialog box

Use the **Property Pages** dialog box to set custom properties to enable communication between a PC-based phase and the FactoryTalk Batch Server.

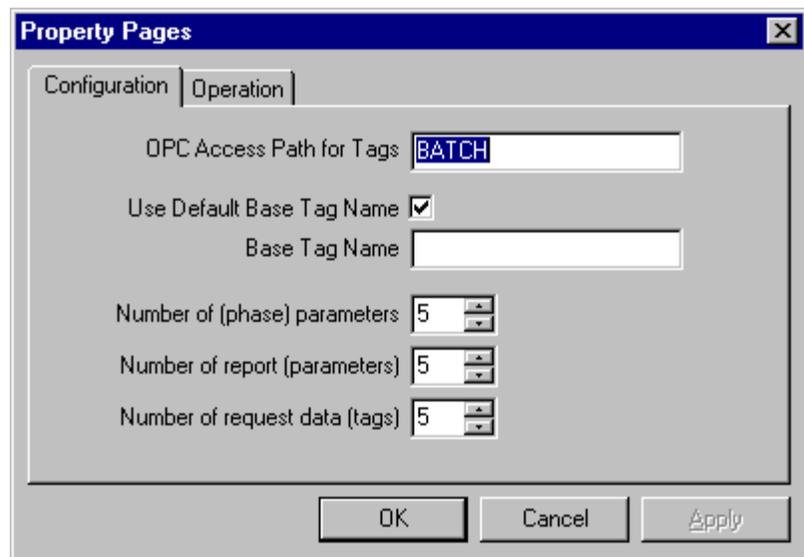
---

**IMPORTANT** If you attempt to configure a property at design time with an invalid entry, you are notified of the error by a message box, which includes a range of valid entries. If you attempt to configure a property at design time that does not support configuration at design time, a message box opens. If you attempt to configure a property at runtime that does not support modification at runtime, an entry is written to the activity log file.

---

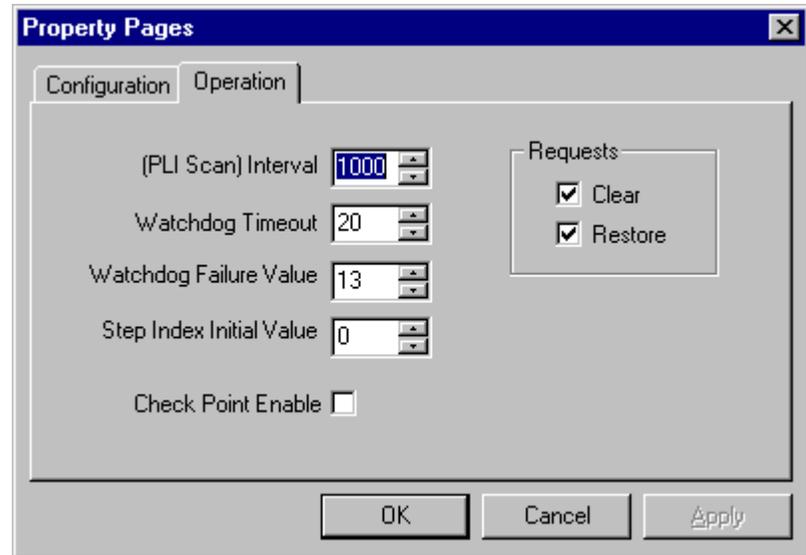
### To set custom properties in the Property Pages dialog box

1. Right-click the **Batch Phase Execution** control and select **Properties**. The **Property Pages** dialog box opens with the **Configuration** tab selected.



2. Enter the values for the custom configuration properties.
3. Select **Apply**.

4. Select the **Operation** tab and enter the values for the custom operation properties.



5. Select **Apply**, then **OK**.

## Program the PC-based phase

The PC-based phase requires specific methods and events to function properly. These methods and events are predefined in the **Batch Phase Execution** control, and must be programmed for the specific PC-based phase being created. The following describes each of the required methods, along with example code for the **OnRestart** event.

## Methods

The **Initialize** and **TerminateState** methods are required for the PC-based phase to function properly.

### Initialize

You must invoke the **Initialize** method in order to create OPC tags and establish the PLI's scan interval. This is done by calling the **Initialize** method, and is usually performed in the **Form\_Load()** event. The following is an example of calling the **Initialize** method:

```
Private Sub Form_Load()
    Charge.Initialize `Initialize the control
End Sub
```

### TerminateState

The **TerminateState** method is used to terminate the active state. You must call this method if you wish the phase to complete its current state. The five active states are RUNNING, HOLDING, STOPPING, RESTARTING, and

ABORTING. For example, calling **TerminateState** while **RUNNING** will transition the phase to **COMPLETE**. The following is an example of calling the **TerminateState** method: `Const UploadAllReports = 2000`

```
Select Case Agitate.StepIndex
    Case 4:
        \ ensure that previous request is complete
        If Agitate.Request = 0 Then
            Agitate.Report(1) = txtAgitateElapsedTime.Text
            If Not Agitate.Owner Then
                \ upload all report parameters
                Agitate.Request = UploadAllReports
            End If
            Agitate.StepIndex = 5
        End If
    Exit Sub
    Case 5:
        \ ensure that report upload is complete
        If Agitate.Request = 0 Then
            Agitate.TerminateState \Terminates the active
            state
        End If
    Exit Sub
End Select
```

## Events

The phase programmer must follow certain conventions when using Visual Basic and the **Batch Phase Execution** control:

- NEVER use modal dialog boxes or other user interface that causes the program logic to wait at some point. This can prevent the PLI from responding to state changes.
- NEVER use looping logic that waits at some point. This can prevent the PLI from responding to state changes. The PLI runs every X millisecond as specified by the **Interval** property. Use **Select Case** or **If Then Else** logic combined with changes to the **StepIndex** property to advance from section to section within your code.
- The phase programmer is responsible for transitioning the phase from an active state (**RUNNING**, **STOPPING**, and so on) to a finished state (**COMPLETE**, **STOPPED**, and so on) using the **TerminateState** method.

## Event example using OnRestart event

The PLI uses the **StepIndexBuffer** property to store the Step Index when the phase is **HELD**. The Step Index is then restored (by the **TerminateState** method) when the phase transitions from **RESTARTING** to **RUNNING**. Typically, the **StepIndexBuffer** would be checked to determine where the phase was **HELD**. The **RESTART** logic would then determine if it's safe for the phase to **RESTART** at that step, or if it should **RESTART** at an earlier step. The **OnRUN** event logic should account for all possible restart conditions.

An example of the conventions that should be used to program the events follows:

```
Private Sub Agitate_OnRestart
    ` Restart the phase at the beginning if it was HELD at
    ` step #2. Otherwise, it's OK to restart where we left
    off.
    If Agitate.StepIndexBuffer = 2 Then
        Agitate.StepIndexBuffer = 1
    End If
    Agitate.TerminateState
        ` transition to RUNNING
End Sub
```

## Create the executable

When you have defined the **Batch Phase Execution** control's properties, methods, and events, you must create the executable that runs during recipe execution. Refer to the Microsoft Visual Basic documentation for specific instructions on creating the executable file.

## Add the phase execution control to a Visual C++ project

We suggest that you contact Rockwell Product Support to determine which version of the Microsoft Foundation Classes (MFC) is currently used by the FactoryTalk Batch control product.

Any phases written with Microsoft Visual C++ should use the same version.

## Create a new project

Use Microsoft Visual C++ to add the **Batch Phase Execution** control to a project.

### To create a new project

1. Start the MFC AppWizard executable.
2. Select the **Dialog Based** option.
3. Ensure that support for ActiveX Controls is selected.
4. Use MFC as a shared DLL.

## Add the control to the project

Use MFC to add the control to the project.

### To add the control to the project

1. In MFC, select **Project > Add to Project**.
2. Select the **Components and Controls** option.
3. Open the **Registered ActiveX Controls** folder.
4. Select the **Batch Phase Execution** control.

**Add the control to the dialog** Use MFC to add the control to the dialog.

### To add the control to the dialog

1. In MFC, open the **Project** dialog (not the **About** box, if there is one).
2. Add the **Batch Phase Execution** control to the dialog from the **Control** toolbar.



Tip: Add one instance of the control for each phase. The resource ID assigned here is used to create event handlers (the framework for phase logic). The resource ID is associated with a member variable that allows access to the phase properties from within the source code.

**Access control properties from Visual C++**

Use MFC to access the control properties from Visual C++.

### To access control properties from Visual C++

1. In MFC, right-click the **Batch Phase Execution** control's icon.
2. Under the **Operation** and **Configuration** tabs, find the properties that require configuration at design time. You can view all properties by using the **All** tab.

**Configure the BaseTagName property**

Visual C++ does not allow you to configure the **Name** property of the control. Configure the **BaseTagName** property to allow the phase to function properly.



Tip: Even though DDE server is no longer supported, the use of DDE in a property name has no significance and is not associated with the server.

### To configure the BaseTagName property

1. Set the **UseDefaultDDETagName** property to **FALSE**.
2. Set the **DDEBaseTagName** property to the name of the phase. This becomes the base for all OPC tag names (example: BASENAME\_ST for phase state).

**Add control events**

Use the resource ID of the phase for each event to add control events.

### To add control events

1. Right-click the **Batch Phase Execution** control's icon.
2. Select the event.
3. Double-click the event.

4. Edit the member function name as desired and click **OK**.

## Add a member variable derived from the Phase Control Class

Use MFC to add a member variable derived from the **Phase Control Class**.



Tip: The category is Control, and the variable type is the Phase Control Class. Use the appropriate resource ID as assigned to the instance of the control in the dialog.

### To add a member variable derived from the Phase Control Class

1. Using the Class Wizard, add a member variable to the project's dialog class.
2. Repeat for each phase (each instance of the control).

## Add phase logic to event handlers

Reference the **Batch Phase Execution** control's properties and methods using the member variable names.

## General usage signature requests

Use a Phase Logic Request to generate a signature request for a PC-based phase.

Before you can successfully use the General Signature Request, electronic signatures must be configured in the area model. Signature templates define the signature signoff information required to complete a signature request generated against a phase. Each signature can require up to three signoffs, which consist of security permissions and optional comments.

## Parameter and report parameter limits

The **Batch Phase Execution** control can be configured to obtain the **Parameter** and **Report Parameter Limits** defined on a PC-based phase in the area model.

The PC-based phase, like any other phase, can be assigned a **Verification** method and associated with verification policies in the area model. Adding additional phase parameter tags and code to the **Batch Phase Execution** control application allows the parameter values to be downloaded for use in the control. Assigning verification policies to a PC-based phase can generate signature requests when parameter or report parameter limit values are too high or too low.

If a PC-based phase has tag locking enabled, and limit tags enabled on the equipment module, additional limit tags are created. For example, if the phase parameter has a **Verification** method equal to **High/Low**, three additional tags are created for the equipment module as shown here:

- *Phasename.P01M* - tag for the **Verification** method
- *Phasename.P01H* - tag for the High Limit
- *Phasename.P01L* - tag for the Low Limit



Tip: Parameter limit tags and Report limit tags are created with a tag delimiter that must be removed when used with a PC-based phase.

## Distribute PC-based phase applications

Please read the license agreement that was shipped with this software. You are bound by the licensing restrictions contained in the agreement. You can use all of the files accompanying this product for development use and redistribute the runtime version of the software in accordance with the terms of the license agreement.



Tip: The FactoryTalk Batch Server is required for the Batch Phase Execution control to function and must be installed on the remote computer.

## Use a PC-based phase

After creating the PC-based phase executable, you must perform the following tasks:

- Update the area model in FactoryTalk Batch Equipment Editor
- Add the phase to your recipes using FactoryTalk Batch Recipe Editor

The following table indicates the relationship between the PC-based phase property and the area model. You must adhere to the rules shown here to allow for proper communication.

| PC-based phase property  | Area model   |
|--|--|
| The OPC's Watchdog Item Access Path  | PC-based phase's OPC Access Path for Tags, as defined in Visual Basic, or other container, or the Topic property (changing one also changes the other) |
| PC-based phase Name property, as defined in Visual Basic, or other container | Equipment Module's Tag name, as defined in FactoryTalk Batch Equipment Editor  |

## Update the area model

You must add a new data server definition to your area model, as well as add the new phase and associated equipment module.

### To update the area model

1. Open the area model in which the new phase will be used.
2. From the **Edit** menu, select **Data Server**, or select the **Edit Data Servers** toolbar.
3. On the **Data Servers** dialog box, select **Add**.
4. On the **Add Data Server** dialog box, enter the new data server definition name in the **Name** box.
5. From the **Type** list, select **PC Phase OPC**.

The default values for the data server type are displayed. The **Watchdog Item Name** must be **WATCHDOG**. Any string may be used for the **Watchdog Item Access Path**, but the same string must be used in the PC-based phase's **Topic** property.

6. Enter the following information for the new data server definition and then click **OK** to return to the **Data Servers** dialog box:
  - Watchdog Item Access Path: **Phase** (*user-definable*)
  - Watchdog Item Name: *watchdog*

7. Create a phase to be used as the PC-based phase. Assign the required number of parameter, report, and request tags. Generally, these would be equal to the number assigned to the PC-based phase's properties.
8. Create an instance of this phase in the appropriate units. Be sure to use the same name for the equipment module as the **Name** property defined for the PC-based phase.
9. Assign the new data server definition to each of the tags in the newly defined equipment module. Do not change the **Item** names. These should follow standard conventions and start with the name assigned to the instance of the control in the container application.
10. Save the area model with the new changes.

## Change the default tag separators for a PC-based phase

Change the default tag separators and update the area model for a PC-based phase.



Tip: The tag separator used in the FactoryTalk Batch Equipment Editor must be removed for all PC-based phase parameter, report, and request data tags.

By default, the tags are set up as:

- [Phasename],P01 for parameters
- [Phasename],R01 for report parameters
- [Phasename],Q01 for request data parameters

The period (.) tag separator must be removed to use these tags.

The tags should be:

- [Phasename]P01 for parameters
- [Phasename]R01 for report parameters
- [Phasename]Q01 for request data parameters

The ten base tags need a period; the upload, download, and request tags do not require a period as the tag separator.

### To change the default tag separators for a PC-based phase

1. Edit the **batcheqp.ini** file and change the default FactoryTalk Batch Equipment Editor tag separator to a period.
2. Save the **batcheqp.ini** file.
3. In FactoryTalk Batch Equipment Editor, double-click a Visual Basic equipment module instance within a unit to display the **Edit Equipment Module** dialog box.

---

**IMPORTANT** This is an irreversible action. When you select Defaults, all applicable tag items are automatically changed.

---

4. On the **Edit Equipment Module** dialog box, select the **Tags** tab and click **Defaults**.

A message displays stating that all tag items have been set to their default names.

- Click **OK**. This places a "." into each tag before the separator. To verify, double-click a tag address to open the **Edit Equipment Module Tag** dialog box.

Notice that the **Item Name** reference of CALCULATE\_WEIGHT1P01 is now CALCULATE\_WEIGHT1.P01.

All other tags should have a "." separator in the **Item Name** tag addressing, such as:

- CALCULATE\_WEIGHT1.F
- CALCULATE\_WEIGHT1.W
- CALCULATE\_WEIGHT1.OC
- CALCULATE\_WEIGHT1.RQ



Tip: Parameter, report, and data request tags do not have a tag separator, as shown here:

- CALCULATE\_WEIGHT1P01
- CALCULATE\_WEIGHT1R01
- CALCULATE\_WEIGHT1Q01

---

**IMPORTANT** Another way to change addressing is to export and import the area model, but this is not recommended because it can introduce errors to the area model.

---

## Set tag items to default names

Set all of the tag items in an area model to default names.

### To set tag items to default names

---

**IMPORTANT** Use Set Default Tag Items with caution. For example, if you have removed the tag separators from the tag items in your area model, setting the default tag items puts the default tag separator back in.

---

- From the **Edit** menu, select **Set Default Tag Items**.
- Verify that you want to set all of the tags to their default names.
- Select **OK** to continue. A message dialog box opens when the process is complete.
- Select **OK** to return to the **Design View** area.

## Add the new phase to a recipe

Use FactoryTalk Batch Recipe Editor to add the new phase to a recipe.

### To add the new phase to a recipe

1. Open FactoryTalk Batch Recipe Editor.
2. Open the recipe where you want to add the new phase.
3. Add the phase and associated transition to the recipe.
4. Verify and save the recipe.

## Run a recipe with a PC-based phase

The PC-based phase executable must be active when the recipe is processed.

### To run a recipe with a PC-based phase

1. Start the PC-based phase executable.
2. Start the FactoryTalk Batch Server.
3. Open the FactoryTalk Batch View or HMI application.
4. Create a batch from the recipe containing the PC-based phase.
5. Run the batch. The recipe phase interacts with the PC-based phase and performs its pre-programmed functions.

The icon that represents the phase changes color based on the state of the phase. These color changes match the colors used in the FactoryTalk Batch View.

## Properties, methods, and events

A PC-based phase is created using the **Batch Phase Execution** control and an OPC container application. The control must be added to the container application, placed in the appropriate medium (form, spreadsheet, document, and so on), and its properties, methods, and events defined. This section defines those properties, methods and events.

### Properties

The following tables provide a summary of the properties associated with the **Batch Phase Execution** control and a description of each property.

#### Property summary

| Property           | Indexed | Type    | Read-only | Can be changed at: |                            | Tag          |
|--------------------|---------|---------|-----------|--------------------|----------------------------|--------------|
|                    |         |         |           | Design time?       | Runtime?                   |              |
| CheckPointEnable   | No      | Boolean | No        | Yes                | Yes                        | None         |
| ClearRequest       | No      | Boolean | No        | Yes                | Yes                        | None         |
| Command            | No      | Long    | —         | No                 | No                         | Phasename_OC |
| DDEBaseTagName     | No      | String  | No        | Yes                | Only before initialization | None         |
| DownloadRequest    | No      | Boolean | Yes       | No                 | No                         | None         |
| ExternalCommand    | No      | Long    | No        | No                 |                            | Phasename_EC |
| Failure            | No      | Long    | No        | No                 |                            | Phasename_F  |
| Interval           | No      | Long    | No        | Yes                | Only before initialization | None         |
| LogFileName        | No      | Text    | No        | Yes                | Yes                        | None         |
| Name               | No      | Text    | No        | Yes                | No                         | None         |
| NumberOfParameters | No      | Long    | No        | Yes                | Only before initialization | None         |
| NumberOfReports    | No      | Long    | No        | Yes                | Only before initialization | None         |
| NumberOfReqData    | No      | Long    | No        | Yes                | Only before initialization | None         |
| Owner              | No      | Boolean | No        | No                 | Only after initialization  | Phasename_W  |
| Parameter(N)       | Yes     | Text    | No        | No                 | Only after initialization  | PhasenamePnn |
| Pause              | No      | Boolean | Yes       | No                 | No                         | Phasename_P  |
| Paused             | No      | Boolean | No        | No                 | Only after initialization  | Phasename_PD |
| Report(N)          | Yes     | Text    | No        | No                 | Only after initialization  | PhasenameRnn |
| Request            | No      | Long    | No        | No                 | Only after initialization  | Phasename_RQ |

| Property              | Indexed | Type    | Read-only | Can be changed at: |                            | Tag           |
|-----------------------|---------|---------|-----------|--------------------|----------------------------|---------------|
|                       |         |         |           | Design time?       | Runtime?                   |               |
| RequestData(N)        | Yes     | Text    | No        | No                 | Only after initialization  | Phasename_Qnn |
| RestoreRequest        | No      | Boolean | No        | Yes                | Yes                        | None          |
| SingleStep            | No      | Boolean | Yes       | No                 | No                         | Phasename_SS  |
| Status                | No      | Long    | Yes       | No                 | No                         | Phasename_ST  |
| StepIndex             | No      | Long    | No        | No                 | Only after initialization  | Phasename_SI  |
| StepIndexBuffer       | No      | Long    | No        | No                 | Yes                        | None          |
| StepIndexInitialValue | No      | Long    | No        | Yes                | Yes                        | None          |
| Topic                 | No      | Text    | No        | Yes                | Only before initialization | None          |
| Unit                  | No      | Long    | Yes       | No                 | No                         | Phasename_UN  |
| UseDefaultDDETag-Name | No      | Boolean | No        | Yes                | Only before initialization | None          |
| WatchdogFailureValue  | No      | Long    | No        | Yes                | Only before initialization | None          |
| WatchdogStatus        | No      | Boolean | Yes       | No                 | No                         | None          |
| WatchdogTimeout       | No      | Long    | No        | Yes                | Only before initialization | None          |

### Property description

| Property         | Description   |
|------------------|---|
| CheckPointEnable | User configurable flag. When set to TRUE, a checkpoint file is updated as needed with all pertinent runtime parameters. The checkpoint file is an internal reference file that is not user-modifiable. The file is updated after each scan if a property change is seen. This file is then loaded when the phase application is started, or when the CheckPoint method is called. When set to FALSE, a checkpoint file is not maintained.<br>The default value is TRUE. |
| ClearRequest     | User configurable flag. When set to TRUE, any pending request by the phase logic is canceled when a change of state is initiated. This property should be used in conjunction with the RestoreRequest property. The PLI will not allow ClearRequest to be FALSE and RestoreRequest to be TRUE (the request would be issued twice in this case).<br>The default value is TRUE.   |
| Command          | Contains the command information as received from the FactoryTalk Batch Server. The phase responds to this command register when the phase is in program control (owned by the FactoryTalk Batch Server). This parameter is reset after each PLI scan, so its contents are NOT indicative of the last command sent by the FactoryTalk Batch Server.   |
| DDEBaseTagName   | Used as the base for tag names when the UseDefaultDDETagName property is set = FALSE and DDEBaseTagName property is not null. For example, if this property is set = BOO, and the UseDefaultDDETagName property is FALSE, then the tag for the phase state is BOO_ST. If this property is empty, the Name property is used as the tag name base regardless of how the UseDefaultDDETagName property is set.   |
| DownloadRequest  | Flag set by the PLI if the FactoryTalk Batch Server has sent a DOWNLOAD command to the phase. The Server sends a DOWNLOAD command when there is a Transfer of Control condition between two consecutive equipment modules (phase instances) based on the same phase. An example would be two consecutive agitation phases where a change in agitation speed is desired when a tank level has reached a certain point.<br>The default value is FALSE.                    |

|                     |  |
|---------------------|--|
| ExternalCommand     | Contains the command information as received from an external source (an HMI, for example). The phase responds to this command register when the phase is in external control (NOT owned by the FactoryTalk Batch Server). The default value is 0.   |
| Property            | Description  |
| Failure             | The Failure property is set within the phase logic as desired when there is an error condition that should cause the phase to be HELD. When this property is set to any value greater than zero, the PLI will execute the HOLDING sequence. The PLI will also set this property to the WatchdogFailureValue when the watchdog status is bad and will HOLD the phase. The valid range is 0 to 32767. The default value is 0.  |
| Interval            | Determines the execution interval (in milliseconds) of the PLI and phase logic. (This value should be the number of seconds multiplied by 1000.) The default value is 1000.  |
| LogFileName         | File name prefix (.LOG is added by the phase execution control) for the activity log file. The default value is BATCHPHS.  |
| Name                | Name of the equipment module. This property must match the name given to the equipment module in FactoryTalk Batch Equipment Editor. Parenthesis created for control arrays are ignored. The default value is PhaseI.  |
| NumberOfParameters  | Indicates the number of parameter tags that will be available to the phase. Generally, this value will be equivalent to the number of parameter tags assigned within FactoryTalk Batch Equipment Editor. The valid range is 0 to 999. The default value is 5.  |
| NumberOfReports     | Indicates the number of report tags that will be available to the phase. Generally, this value will be equivalent to the number of report tags assigned within FactoryTalk Batch Equipment Editor. The valid range is 0 to 999. The default value is 5.  |
| NumberOfRequestData | Indicates the number of request data tags that will be available to the phase. Generally, this value will be equivalent to the number of request data tags assigned within FactoryTalk Batch Equipment Editor. The valid range is 0 to 10. The default value is 5.   |
| Owner               | Determines phase ownership. When set to TRUE, the phase is in External control and will respond to the contents of the ExternalCommand property. When set to FALSE, the phase is owned by the FactoryTalk Batch Server and will respond to the Command property. The default value is FALSE.   |
| Parameter(N)        | Phase parameters, where the total number (N) is specified in the NumberOfParameters property. These are downloaded from the FactoryTalk Batch Server by placing the appropriate values into the Request property. A NULL string is returned if Initialize has not yet been called; Invalid index is returned when an illegal (out of range) index is used. (Refer to the <i>PCD Programmer Technical Reference Guide</i> for details.)   |
| Pause               | The Pause property (pausNG) is set by the PLI when the Pause command is received, or when single-step mode is activated. (Refer to the <i>FactoryTalk Batch Server API Communications Language Reference Guide</i> for further details). The default value is FALSE.   |
| Paused              | Set within the phase logic as desired to place the phase in the Paused state. A Resume command must be issued to continue phase execution when the Paused (pausED) property is set. (Refer to the <i>PCD Programmer Technical Reference Guide</i> for details). The default value is FALSE.  |
| Report(N)           | Report parameters, where the total number (N) is specified in the NumberOfReports property. Report parameters are set within the phase logic to upload report values to the FactoryTalk Batch Server for archiving. The phase logic must store report values to this property and then store an Upload Request to the Request property. A NULL string is returned if Initialize has not yet been called; Invalid index is returned when an illegal (out of range) index is used. |

| Request               | Used to make requests to the FactoryTalk Batch Server to accomplish a wide range of tasks such as downloading and uploading parameter and report values. Phase logic stores a predefined request value to this property to initiate action by the Server. This property value must be equal to zero before storing a valid request into this property. This property will return to zero when the Server has completed the requested action. (Refer to the <i>PCD Programmer Technical Reference Guide</i> for details.) The valid range is 1000 to 2147483648. The default value is 0.                                    |
|-----------------------|--|
| Property              | Description  |
| RequestData(N)        | Request data parameters, where the total number (N) is specified in the NumberOfRequestData property. The RequestData property performs two functions: <ol style="list-style-type: none"> <li>1. Supplies additional arguments for a Request that is being made using the Request property.</li> <li>2. Transfers data values from a sending Transfer phase to a receiving Transfer phase. A NULL string is returned if Initialize has not yet been called; Invalid index is returned when an illegal (out of range) index is used. (Refer to the <i>PCD Programmer Technical Reference Guide</i> for details.)</li> </ol> |
| RestoreRequest        | User configurable flag. When the phase transitions from RESTARTING to RUNNING, this property (when TRUE) will cause the request that was pending at the time the phase was HELD to be restored and executed. This property should be used in conjunction with the ClearRequest property. The PLI will not allow ClearRequest to be FALSE and RestoreRequest to be TRUE (the request would be issued twice in this case). The default value is TRUE.  |
| SingleStep            | Flag toggled by the PLI when it receives a SingleStep command. When single-stepping is turned ON, the Pause (pausING) property is set to TRUE if the phase has been in the free-running state. This combination is referred to as the single-step pausing state. When single-stepping is turned OFF, both the Pause (pausING) and Paused (pausED) properties are cleared (set to FALSE) and the phase return to the free-running state. (Refer to the <i>PCD Programmer Technical Reference Guide</i> for details). The default value is FALSE.  |
| Status                | Contains a numerical value that represents the current phase state. The following are the possible values with the corresponding state: <ul style="list-style-type: none"> <li>• 10 = ABORTING</li> <li>• 20 = HOLDING</li> <li>• 30 = STOPPING</li> <li>• 40 = RESTARTING</li> <li>• 50 = RUNNING</li> <li>• 60 = HELD</li> <li>• 70 = COMPLETE</li> <li>• 80 = STOPPED</li> <li>• 90 = ABORTED</li> <li>• 100 = IDLE</li> </ul> The default value is 100.  |
| StepIndex             | Set equal to the StepIndexInitialValue property by the PLI when the phase is STARTed or RESET. The phase logic may use this property to control program flow. The default value is 1.  |
| StepIndexBuffer       | Used by the PLI to save the StepIndex property when the phase is HELD. The StepIndex is restored from the StepIndexBuffer when the phase is RESTARTED. The default value is 0.   |
| StepIndexInitialValue | Default value of the step index. The PLI initializes the StepIndex property to the StepIndexInitialValue value when the START or RESET command is received. The default value is 1.  |

|                      |   |
|----------------------|---|
| Topic                | <p>This is the Watchdog Item Access Path as defined in a OPC server within the area model. You may define multiple OPC servers to be used with PC-based phases. They do not need to use the same topic. You may also use multiple instances of a control within one project and refer to different OPC servers. If the Topic property is set at runtime, it must be set before the Initialize method is called. The default value is BATCH.</p> |
| Unit                 | <p>Contains the UnitID of the unit that controls the phase. The default value is 0.</p>   |
| UseDefaultDDETagName | <p>User configurable flag. When set to TRUE, or when the DDEBaseTagName property is not null, the content of the Name property is used as the base for tag names. When set to FALSE, and the DDEBaseTagName property is not null, then the content of the DDEBaseTagName property is used.</p>  |

| Property             | Description  |
|----------------------|--|
| WatchdogFailureValue | This is the value that the Failure register is set to when the WatchdogStatus is bad. The value used here should then be defined within the phase_failure enumeration set in the area model. This property can be set to 0, which will turn off the hold on watchdog failure logic; the phase will not be HELD when the WatchdogStatus goes bad. The valid range is 0 to 2147483648. The default value is 13.  |
| WatchdogStatus       | Current status of the watchdog between the FactoryTalk Batch Server and the phase. A zero (FALSE) indicates that the status is bad and communications are down. The PLI will set the Failure property to 13 when the WatchdogStatus is bad. The default value is FALSE.  |
| WatchdogTimeout      | Interval, in seconds, between watchdog checks by the PLI. If the WatchdogTimeout property is set at runtime, it must be set before the Initialize method is called. The valid range is 5 to 1800 seconds. The default value is 20. The timeout period is configurable in the Batchsvr.ini file, entered in milliseconds. The default settings are in the [XMAN] section: WatchdogPeriod=10000 and WatchdogCycles=5. The default time period for the FactoryTalk Batch is ten (10) seconds. The PCD timer should be set to a value slightly larger than the product of the WatchdogPeriod and the WatchdogCycles. |

## Methods

The following section provides a summary of each method associated with the **Batch Phase Execution** control and a description of how each method is used. There are no parameters associated with any of these methods.

### Method summary

| Method         | Return value                      | Description   |
|----------------|-----------------------------------|---|
| CheckPoint     | True = success<br>False = failure | Called to force an update of the checkpoint data file.  |
| Initialize     | True = success<br>False = failure | Used to initialize the control. The control will not function properly if this method is not invoked. |
| LogTagNames    | True                              | Used to write all tag names to the activity log file.   |
| TerminateState | None                              | Your application must call this method to complete the transition of an Active state.                 |
| CommandAbort   | True = success<br>False = failure | Used to transition the phase to ABORTING.   |
| CommandHold    | True = success<br>False = failure | Used to transition the phase to HOLDING.  |
| CommandReset   | True = success<br>False = failure | Used to transition the phase to IDLE.   |
| CommandRestart | True = success<br>False = failure | Used to transition the phase to RESTARTING.   |
| CommandStart   | True = success<br>False = failure | Used to transition the phase to RUNNING.  |
| CommandStop    | True = success<br>False = failure | Used to transition the phase to STOPPING.   |

### Method description

- Checkpoint:** This method is invoked to update the checkpoint data file. Typically, this is not needed since the checkpoints are saved as needed

at the end of every PLI scan. It should only be called AFTER the **Initialize** method has been called and executed successfully.

- **Initialize:** This method performs all initialization and MUST be called before the phase will execute properly. It only needs to be called once and in fact, will not execute again if called more than once. You should check the return value and not allow the application to continue if this method fails. **Initialize** does the following:
  - Loads checkpoint data if required (fires **OnCheckPointFail** if the checkpoint read fails).
  - Creates and initializes all tags.
  - Creates the watchdog.
  - Creates a timer based on the **Interval** property.
  - Fires the **OnPowerUp** event.
- **LogTagNames:** This method writes all tag names to the activity log file. It is used as a tool to view the current tag names. It should only be called AFTER the **Initialize** method has created and initialized the tags. It will always return a value of TRUE.
- **TerminateState:** This method is used to terminate an active state and transition the phase to the next desired state. In other words, it will cause the phase to change state as shown in the following table:

| Current state | Transitioned state upon call of <b>TerminateState</b> |
|---------------|---|
| ABORTING      | ABORTED   |
| HOLDING       | HELD  |
| STOPPING      | STOPPED   |
| RESTARTING    | RUNNING   |
| RUNNING       | COMPLETE  |

The phase programmer must use **TerminateState** within the phase logic to transition the phase state at the appropriate time. Refer to the *PCD Programmer Technical Reference Guide* for complete instructions regarding state transition logic.

- **Transition command methods:** The Transition Command methods allow the phase programmer to change the state of the phase from within the source code. The state transition rules must be followed in order for the PLI within the **Batch Phase Execution** control to honor the command. The following table shows the command name, the state to which the phase will transition, and the state(s) that the phase must be in for the command to be honored.

| Method         | Phase transitions to: | Phase state must be:                            |
|----------------|-----------------------|---|
| CommandAbort   | ABORTING              | HOLDING, STOPPING, RUNNING, RESTARTING, or HELD |
| CommandHold    | HOLDING               | RUNNING or RESTARTING                           |
| CommandReset   | IDLE                  | COMPLETE, ABORTED, or STOPPED                   |
| CommandRestart | RESTARTING            | HELD (and the phase has not failed)             |
| CommandStart   | RUNNING               | IDLE (and the phase has not failed)             |
| CommandStop    | STOPPING              | HOLDING, RUNNING, RESTARTING, or HELD           |

## Events

The following table provides a brief description for each event associated with the **Batch Phase Execution** control. There are no parameters associated with any of these events.

| Event                | Description   |
|----------------------|---|
| OnAbort              | Fired by the Batch Phase Execution control when the phase is in the ABORTING state. Only happens after the Initialize method is invoked.              |
| OnHold               | Fired by the Batch Phase Execution control when the phase is in the HOLDING state. Only happens after the Initialize method is invoked.               |
| OnRun                | Fired by the Batch Phase Execution control when the phase is in the RUNNING state. Only happens after the Initialize method is invoked.               |
| OnRestart            | Fired by the Batch Phase Execution control when the phase is in the RESTARTING state. Only happens after the Initialize method is invoked.            |
| OnStop               | Fired by the Batch Phase Execution control when the phase is in the STOPPING state. Only happens after the Initialize method is invoked.              |
| OnIdle               | Fired by the Batch Phase Execution control when the phase is in the IDLE state. Only happens after the Initialize method is invoked.                  |
| OnLoadCheckpointFail | Fired by the Batch Phase Execution control during startup if there are no valid checkpoints to load and the CheckPointEnable property is set to TRUE. |
| OnPowerUp            | Fired by the Batch Phase Execution control during execution of the Initialize method. Only called if the Initialize method is successful.             |





## Application notes and example phase

This section contains application notes and an example of a generic PC-based phase. The example uses the sample application (SampleDemo) that is installed with FactoryTalk Batch. For more information on the sample application, see the *FactoryTalk Batch Getting Results Guide*.

### Application notes

The following are points to remember when programming PC-based phases:

- There should be a data type mismatch between the request data tags configured in the area model and the tags configured on the Visual Basiccontrol. In the area model, the tags must be configured as integers for the FactoryTalk Batch Server to work correctly; in the control they are configured as strings. The OPC call made from the Server converts the strings to integers. All other tags should have a one-to-one mapping of tag types.
- If you remove tag locking from the phase class configured in FactoryTalk Batch Equipment Editor, the FactoryTalk Batch Server will try to do the conversion from the tag type configured in the class to the type that is used in the instance. This is useful for PC-based phases because this allows you to create the phase class parameters and reports of any data type you choose. At the instance level, you can then change the actual instance tag types of the parameters and reports to strings, which are required by the PC-based phase. This is especially useful for material-enabled phases where the tag type of the AMOUNT report tag must be real, and the tag type of the FEED\_COMPLETE report tag must be enumeration.
- The tag separator used in FactoryTalk Batch Equipment Editor must be removed for all PC-based phase parameter, report, and request data tags. By default, the tags are set up as [Phasename].PO1 for parameters, [Phasename].RO1 for report parameters, and [Phasename].QO1 for request data parameters. The period (.) tag separator must be removed in order for the tags to be used. The tags should be [Phasename]PO1 for parameters, [Phasename]RO1 for report parameters, and [Phasename]QO1 for request data parameters.
- The **Name** in the FactoryTalk Batch Equipment Editor **Edit Equipment Module** dialog box must agree exactly with the **Name** property of the phase.
- Phase controls can be added to Visual Basic forms as control arrays. In this case, the phase control's property index number corresponds to the number at the end of the Equipment Module's name. For example,

if CHARGE1 is the Equipment Module name, the corresponding Visual Basic Phase Control name created as part of a control array is CHARGE(1). The 1 is the **Index** property of the control. The programmer must decide when to use a control array or individual names for phase controls.

- The phase control sets the **StepIndex** property to the **StepIndexInitialValue** upon entering the control event (**OnRun**, **OnStop**, and so on.). Use this value to reference the first step in the **OnRun** event logic. The user can subsequently set the step index to any value (see the example phase).
- Remember that the code is executed based on the value in the **Interval** property. For example, if the **Interval** value is 1000 milliseconds, the code is repeated every second.
- If a download of parameters is required, place it in the first step, as shown in the accompanying example for the **OnRun** event. This approach ensures that the download is only issued once. The next step should wait until the request register is reset to zero by FactoryTalk Batch before proceeding through the remaining steps.
- If assigning a value to a variable, and it is not in a case select structure or some other IF statement structure, it will have the value assigned to it each time that event fires. This could lead to errors in code execution of the control event.
- If using multiple phase controls on the same form, make sure that each phase control variable is unique to the phase if declared in the general declaration section. The example program is for only one phase, but if there were other phase controls, the variables used in one could not be used in the other. This could lead to one phase overwriting values needed by another phase. It is conceivable that there could be exceptions.
- To run a Visual Basic phase, make an executable file (.exe). Start the Visual Basic phase, start the FactoryTalk Batch Server, check that the communication between the two is good (tag verification should also be done at this point until all tags verify successfully). The Visual Basic phase is now ready to have a recipe run against it.

## PC-based phase example

This PC-based phase example provides a starting point for creating your own PC-based phases.

```
'=====
===
'This program was written using Microsoft Visual Basic 6 and
'is intended to be used as a starting point for creating your
'own PC-Based Phases.
'=====
===
'Phase request definitions
Const NO_REQUEST_PENDING = 0
Const REQUEST_ALL_PARAMETERS = 1000
Const UPLOAD_ALL_PARAMETERS = 2000
```

```

Private Sub Form_Load()

    'Initialize the PC-Based Phase Control
    PH_CHECK_A.Initialize
End Sub

Private Sub PH_CHECK_A_OnAbort()

    'Terminate the current state of the phase
    PH_CHECK_A.TerminateState

End Sub

Private Sub PH_CHECK_A_OnHold()

    'Terminate the current state of the phase
    PH_CHECK_A.TerminateState

End Sub

Private Sub PH_CHECK_A_OnIdle()

    'Terminate the current state of the phase
    PH_CHECK_A.TerminateState

End Sub

Private Sub PH_CHECK_A_OnRestart()

    'Terminate the current state of the phase
    PH_CHECK_A.TerminateState
End Sub

Private Sub PH_CHECK_A_OnRun()
'The following Select Case performs the processing steps
'for the PH_CHECK_A logic. The initial step index for the
'phase is zero ("0").

    Select Case PH_CHECK_A.StepIndex

        Case 0:
            'Wait until no requests are pending
            If PH_CHECK_A.Request = NO_REQUEST_PENDING Then
                'Ask server to download all parameters for phase
                PH_CHECK_A.Request = REQUEST_ALL_PARAMETERS
                PH_CHECK_A.StepIndex = 10
            End If

        Case 10:
            'Wait until parameters have been downloaded
            If PH_CHECK_A.Request = NO_REQUEST_PENDING Then
                PH_CHECK_A.StepIndex = 20
            End If

        Case 20:
            'This case statement should contain the code required

```

```

'to perform the function of the phase
PH_CHECK_A.StepIndex = 30

Case 30:
'This case statement should contain the code required
'to put report data into report registers
PH_CHECK_A.Report(1) = PH_CHECK_A.Parameter(1)
PH_CHECK_A.StepIndex = 40

Case 40:
'Wait until no requests are pending
If PH_CHECK_A.Request = NO_REQUEST_PENDING Then
'Upload reports
PH_CHECK_A.Request = UPLOAD_ALL_PARAMETERS
PH_CHECK_A.StepIndex = 50
End If

Case 50:
'Wait until upload has completed
If PH_CHECK_A.Request = NO_REQUEST_PENDING Then
PH_CHECK_A.StepIndex = 60
End If

Case 60:
PH_CHECK_A.TerminateState

Case Else
PH_CHECK_A.StepIndex =
PH_CHECK_A.StepIndexInitialValue
End Select
End Sub

Private Sub PH_CHECK_A_OnStop()
'Terminate the current state of the phase
PH_CHECK_A.TerminateState

End Sub

```

## Create a new PC-based phase

Use Visual Basic to create a new PC-based phase.

### To create a new PC-based phase

1. Using Visual Basic version 6.0 or greater, create a new standard executable and add the **Batch Phase Execution** control to the form.
2. Copy the text from the PC-based phase example into the code window of the project.
3. Change the **Name** property of the **Batch Phase Execution** control to **PH\_CHECK\_A**.
4. Set the **CheckpointEnable** property to **FALSE**.
5. Set the **StepIndexInitialValue** property to **0**.
6. Create an executable file. This file must be running during batch execution.

## Add a new data server definition to an area model

Add a new data server definition (DSDF) to your area model, as well as add a new phase and associated equipment module.

### To add a new data server definition to an area model

1. Open the SampleDemo area model in which the new phase will be utilized.
2. From the **Edit** menu, select **Data Server**, or click the **Edit Data Servers** toolbar button. The **Data Servers** dialog box opens.
3. Click **Add**. The **Add Data Server** dialog box opens.
4. In **Name**, enter the new data server definition name.
5. From **Type**, select **PC Phase OPC**. The default values for the data server type are displayed. The **Watchdog Item Name** must be **WATCHDOG**. Any string may be used for the **Watchdog Item Access Path**, but the same string must be used in the PC-based phase's **Topic** property.
6. Enter the following information for the new data server definition:
  - Watchdog Item Access Path: **Phase** (*user-definable*)
  - Watchdog Item Name: watchdog
7. Click **OK** to return to the **Data Servers** dialog box.
8. Create a new phase named **PHASE**. Assign the required number of parameter, report, and request tags. Generally, these would be equal to the number assigned to the PC-based phase properties.
9. Create an instance of this phase named **PH\_CHECK\_A** in the appropriate units. Be sure to use the same name for the equipment module as the **Name** property defined for the PC-based phase.
10. Assign the new data server definition to each of the tags in the newly defined equipment module. Do not change the **Item** names. These should follow standard conventions and start with the name assigned to the instance of the control in the container application. (Refer to the *FactoryTalk Batch Equipment Editor User Guide* for information on assigning data servers.)
11. Save the area model with the new changes.
12. Open FactoryTalk Batch Recipe Editor.
13. Open the recipe to which the new phase will be added.
14. Add the phase and associated transition to the recipe.
15. Verify and save the recipe.

## Use limit tags on a PC-based phase

To obtain the Parameter Limits or Report Parameter Limits using a Visual Basic application, configure additional phase parameter tags in the PC-based phase to be used as addresses for the limit tags in the area model.

## To use limit tags on a PC-based phase

1. In Visual Basic, during PC-based phase configuration, specify additional phase parameter tags on the **Custom Properties** dialog box of the **Batch Phase Execution** control. The number of phase parameter tags required is equal to the normal number of phase parameter tags plus the number of limit tags specified in the area model.
2. In FactoryTalk Batch Equipment Editor, assign a **Verification** method to the PC-based phase parameters or phase report parameters.
3. Select the equipment module and enable the Parameter Limit Tags or the Report Limit Tags by checking **Enable Limit Tags**.
4. From the **Parameter** or **Report Limit Tags** tab, double-click a Limit Tag to open the **Edit Equipment Module Tag** dialog box. Change the **Item Name** to one of the phase parameter tag addresses (names) defined in Visual Basic. The **Item Name** is referenced in code to display Limit values. For example, PHASE1P01M changes to *PH\_CHECK\_APO3*.
5. At runtime, when the PC-based phase requests the Limit Tags, the FactoryTalk Batch Server loads the limit values into the additional phase parameter tags. The PC-based phase can access the value of these tags using the **get\_RequestData** method. The **set\_RequestData** is used to reset the value of the tag.

## Add limit tags to a PC-based phase

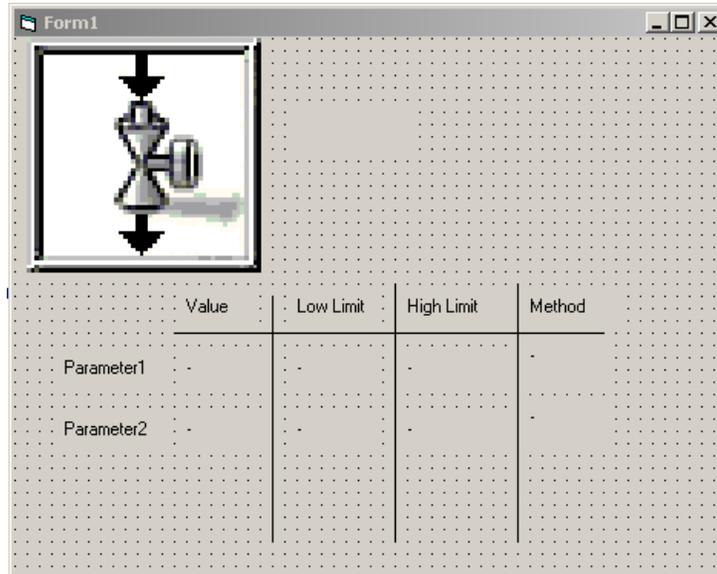
Use FactoryTalk Batch Equipment Editor and Visual Basic to add limit tags to a PC-based phase.

The code example below could be used to display Parameter Limit tags using a Visual Basic application, and is an extension of the code example given in the **PC-based phase example** section.

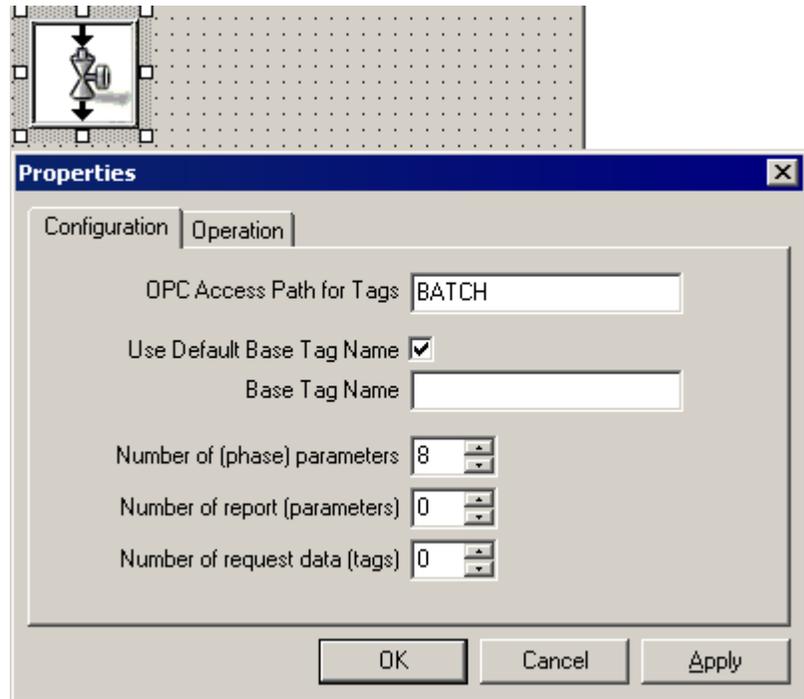
## To add limit tags to a PC-based phase

1. In FactoryTalk Batch Equipment Editor, verify that the PC-based phase has **Tag Locking** enabled and two parameters.
2. Assign a **Verification** method of **High/Low** to both of the parameters.
3. On the PC-based phase **Parameter Limit Tags** tab, check **Enable Limit Tags**. The following six additional tags are created:
  - PARMTR01H
  - PARMTR01L
  - PARMTR01M
  - PARMTR02H
  - PARMTR02L
  - PARMTR02M
4. On the **Tags** tab, double-click one of the new limit tags.

5. On the **Edit Equipment Phase Tag** dialog box, change the limit tag **Item Name** to a phase parameter tag name. Repeat this for each limit tag. These names are referenced in code to display the limit tag values. For example:
  - PHASE1Po1M changes to PH\_CHECK\_AP03
  - PHASE1Po1L changes to PH\_CHECK\_AP04
  - PHASE1Po1H changes to PH\_CHECK\_AP05
  - PHASE1Po2M changes to PH\_CHECK\_AP06
  - PHASE1Po2L changes to PH\_CHECK\_AP07
  - PHASE1Po2H changes to PH\_CHECK\_AP08
6. In Visual Basic, create a form similar to the one shown here:



- On the **Custom Properties** dialog box, add phase parameter tags to the **Batch Phase Execution** control. These tags are referenced in code to display the status of the phase, parameter values, and limit tag values.



- Add code to map the phase parameter tags defined in the area model to the labels on the form. For example:

```
Private Sub Form_Load()
    PH_CHECK_A.Initialize
    lState.Caption = PH_CHECK_A.Status
End Sub
Private Sub PH_CHECK_A_OnRun()
    lState.Caption = PH_CHECK_A.Status
    lParameterValue(0).Caption = PH_CHECK_A.Parameter(1)
    lParameterValue(1).Caption = PH_CHECK_A.Parameter(2)
    lParameterMethod(0).Caption =
PH_CHECK_A.Parameter(3)
    lParameterLowLimit(0).Caption =
PH_CHECK_A.Parameter(4)
    lParameterHighLimit(0).Caption =
PH_CHECK_A.Parameter(5)
    lParameterMethod(1).Caption =
PH_CHECK_A.Parameter(6)
    lParameterLowLimit(1).Caption =
PH_CHECK_A.Parameter(7)
    lParameterHighLimit(1).Caption =
PH_CHECK_A.Parameter(8)
End Sub
PH_CHECK_A.TerminateState
lState.Caption = PH_CHECK_A.Status
End Sub
```

- Save the Visual Basic project and make a new executable.



Tip: No additional coding is necessary to enable signature requests on a PC-based phase. If a PC-based phase parameter or report parameter is configured with a Verification method and a signature template, the FactoryTalk Batch Server generates a signature request if the recipe parameter or report parameter limit tag values exceed their defined limits.





# Index

## A

- ActiveX control** 35
- ActiveX custom control** 9
  - add to a Visual Basic project 15
  - place on a form 15

**add phase to a recipe** 26

## C

- C++ project** 19
- Command Verification Policies** 21
- configure DDEBaseTagName property** 20
- control events** 20

## D

- data tag**
  - request 35
- data type mismatch** 35
- distribute PC-based phase applications** 22

## E

- executable**
  - create 19
- execution of PC-based phase** 10

## L

**Limit Tags** 39

## M

- material-enabled phases** 35
- methods**
  - program 17
  - summary 31
- MFC AppWizard** 19

## O

- OnRestart Event**
  - program 18

## P

- PC-based phase**
  - definition 9
  - distribute applications 22
  - executable 19

- run in a recipe 26
- set properties 16

### Phase Execution Control

- add to a Visual Basic project
  - 15 C++ project 19
  - place on a form 15

### program the PC-based phase 17

- events 18

### properties

- set 16

## R

### recipe

- add a PC-based phase 26
- run with a PC-based phase 26

### remove tag locking 35

### request data tag 35

## T

- tag locking** 35
- tag separator** 35

## U

### update the area model 23, 38

- upload request**
  - warm restart 26

## V

**verification method** 21

## W

- warm restart**
  - upload requests 26

# Rockwell Automation support

Use these resources to access support information.

|   |  |  |
|---|--|--|
| <b>Technical Support Center</b>                         | Find help with how-to videos, FAQs, chat, user forums, and product notification updates.                   | <a href="http://rok.auto/support">rok.auto/support</a>             |
| <b>Knowledgebase</b>                                    | Access Knowledgebase articles.   | <a href="http://rok.auto/knowledgebase">rok.auto/knowledgebase</a> |
| <b>Local Technical Support Phone Numbers</b>            | Locate the telephone number for your country.  | <a href="http://rok.auto/phonesupport">rok.auto/phonesupport</a>   |
| <b>Literature Library</b>                               | Find installation instructions, manuals, brochures, and technical data publications.                       | <a href="http://rok.auto/literature">rok.auto/literature</a>       |
| <b>Product Compatibility and Download Center (PCDC)</b> | Get help determining how products interact, check features and capabilities, and find associated firmware. | <a href="http://rok.auto/pcdc">rok.auto/pcdc</a>                   |

## Documentation feedback

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

## Waste Electrical and Electronic Equipment (WEEE)



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

Rockwell Automation maintains current product environmental information on its website at [rok.auto/pec](http://rok.auto/pec).

Allen-Bradley, expanding human possibility, Logix, Rockwell Automation, and Rockwell Software are trademarks of Rockwell Automation, Inc.

EtherNet/IP is a trademark of ODVA, Inc.

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

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

Connect with us.    

[rockwellautomation.com](http://rockwellautomation.com) expanding **human possibility**<sup>™</sup>

AMERICAS: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444

EUROPE/MIDDLE EAST/AFRICA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640

ASIA PACIFIC: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846