

White Paper

Tips on Using PhaseManager with RSBizWare Batch

*This paper provides helpful tips
for using the new RSBizWare
Batch PhaseManager with
RSBizWare Batch.*

*by Don Smith
Senior Process Automation Consultant
Rockwell Software*



Introduction

PhaseManager™ is a collection of features that integrates the RSBizWare™ Batch software with the RSLogix™ 5000 programming software and the Logix5000™ family of controllers. The integration of these products simplifies the configuration and maintenance of your Batch automation system, provides a superior means of communication between the Batch Server and the Logix5000 controller, and significantly reduces the programming effort required to develop the phase logic code that resides in your Logix5000 controller.

What is PhaseManager?

In the Batch Equipment Editor, you create a phase to represent a group of equipment that performs a minor processing activity in a manufacturing plant, and you create an equipment module when you add a specific instance of a phase to a unit.

PhaseManager is a collection of features in RSBizWare Batch, RSLogix 5000, and an addition of capabilities in the Logix5000 family of controllers that extend the concept and definition of Batch phases to the RSLogix 5000 software and the Logix5000 family of controllers.

PhaseManager adds an equipment phase object to the Logix5000 family of controllers and gives you the ability to create equipment phases in the RSLogix 5000 programming software. You can create or update equipment phases in RSLogix 5000 by adding or modifying them directly in your project, or by synchronizing your project with an area model that contains PhaseManager equipment modules. Likewise, you can create or update equipment modules in your area model by synchronizing with an RSLogix 5000 project file that contains PhaseManager equipment phases.

Tip 1: PXRQ and the Phase Instruction

PXRQ – the phase external request command

This command requires several things to work:

- 1) A data array to store information, `q[]`, type integer.
- 2) A phase instruction to store information in as part of the request, `req`, type phase instruction.

Note that you must have a *new phase instruction register* for each PXRQ. When a phase is reset, the Phase Instruction is cleared and ready to use again. If, during the life of the running logic time, a PXRQ is used, it must use a phase instruction register that has been reset. You may find it convenient to create the phase instruction register as an array, and then increase the index each time you use the phase instruction register. I would suggest using an integer, `req_index`, which you increment each time you use a PXRQ. In the Resetting Logic, reset the `req_index = 0`. This concept is demonstrated in Tip 2, below.

See the Quick Reference for Configuring the PXRQ Instruction located in the *RSBizWare Batch PhaseManager User's Guide* (`batch_phasemanager.pdf`) for information on the value that should be placed in the `q[]` array for each command class to achieve a specific functionality.

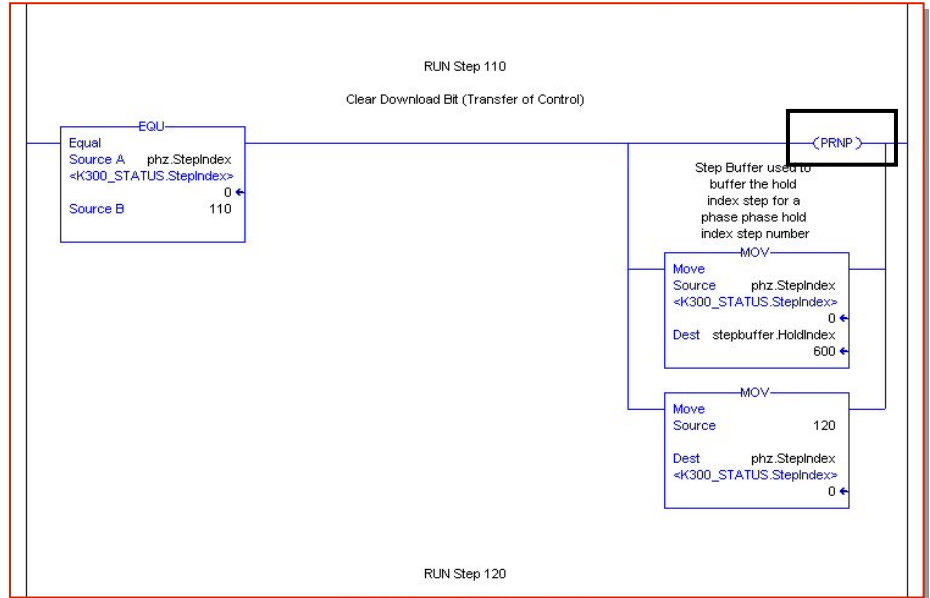
Tip 2: Transfer of Control

Transfer of control (TOC) must work in the same fashion as outlined in Tip 1 – a new phase instruction register is required each time you generate a request. Again, using an index and incrementing it each time works well. You must set aside enough space for the number of times PXRQ is used across all the TOC states. If a phase does a TOC three times and has two requests each time, then the array needs to be **6** ($3 \times 2 = 6$). Again, reset the index in Resetting Logic.

Tips on Using PhaseManager with RSBizWare Batch

To build a TOC phase:

1. The first line resets the Phase New Parameters:



2. Set local variables for the PXRQ instruction:

Name	Usage	Alias For	Base Tag	Data Type	Style	Description
phz	<normal>	K300_STATUS(C)	K300_STATUS(C)	PHASE		
q	<normal>			DINT[4]	Decimal	data array for pxrq...
ReqParm	<normal>			PHASE_INSTRUCTION[10]		request for param...
reqparm_index	<normal>			DINT	Decimal	index for request f...
stepbuffer	<normal>			StepBuffer		Step Buffer used t...

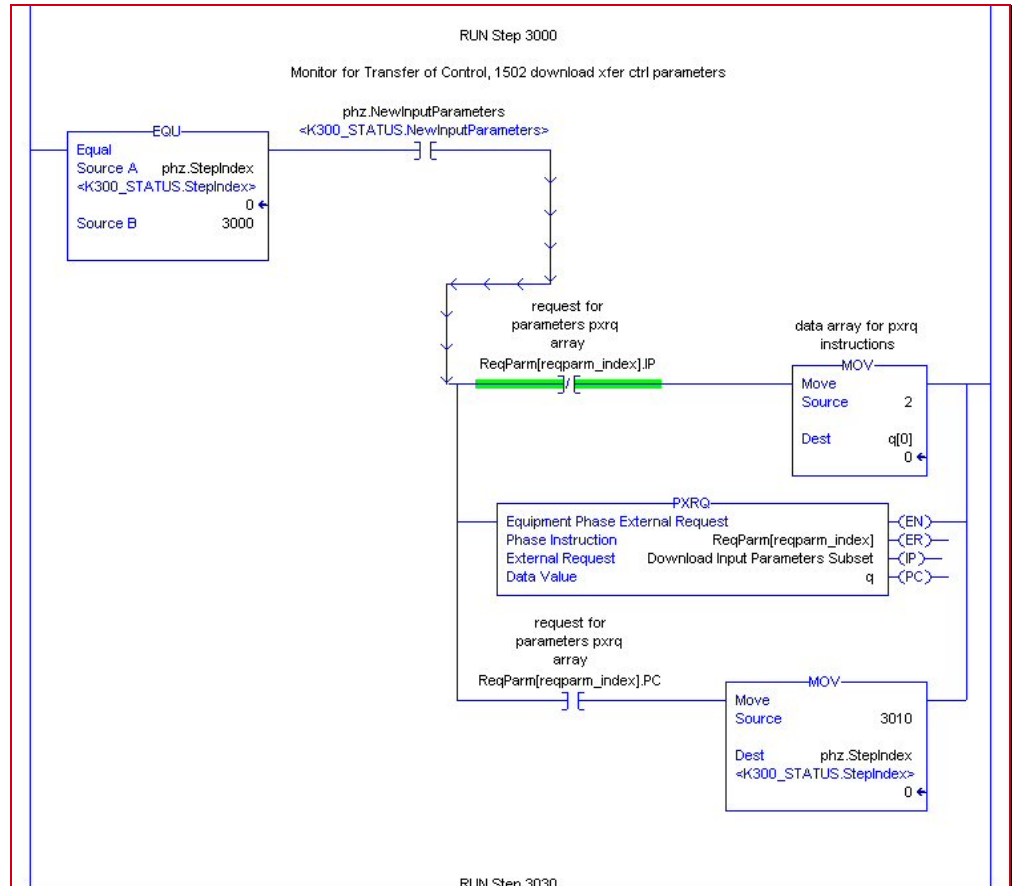
The *q* array is a DINT, set at a size of 4

The *ReqParm* of type PHASE INSTRUCTION, set to a size of 10, as required by your phase design.

The *reqparm_index* is a DINT.

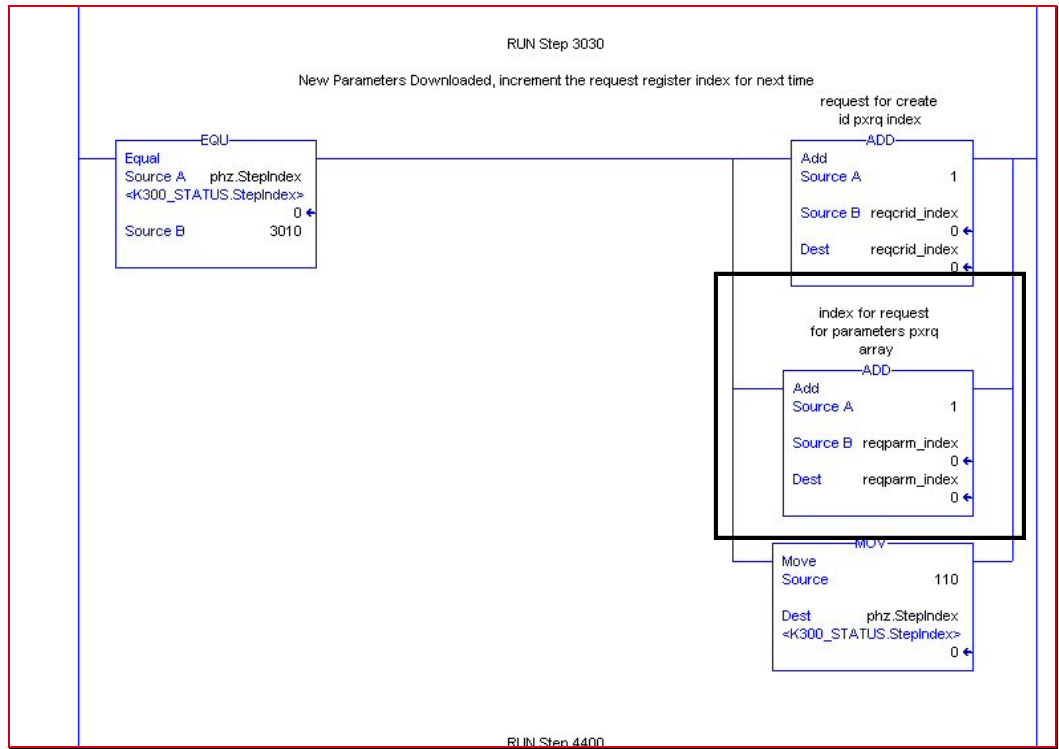
Tips on Using PhaseManager with RSBizWare Batch

- Now, wait for an indication that the phase has new parameters (Phase.NewInputParameters, formerly the DL bit), then request the TOC parameters.

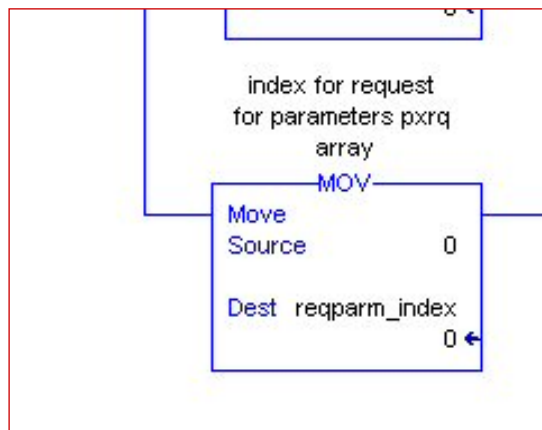


Tips on Using PhaseManager with RSBizWare Batch

4. After the new parameters are received, increment the index:



5. Be sure to reset the index back to zero in the resetting logic.

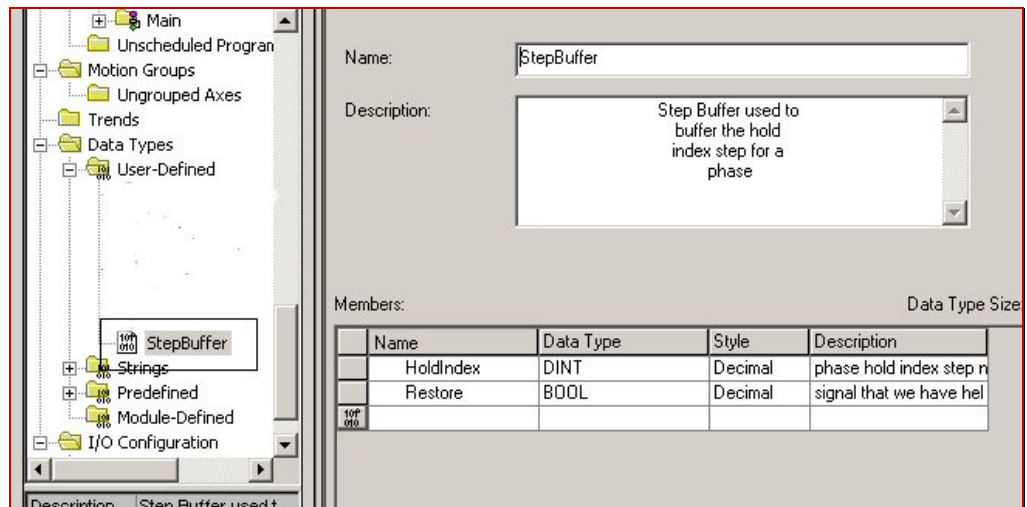


It is a good idea to put a fault monitor in the failure logic to check if the index exceeds the size you have allocated. If the index gets too big, use PFL to fault the phase.

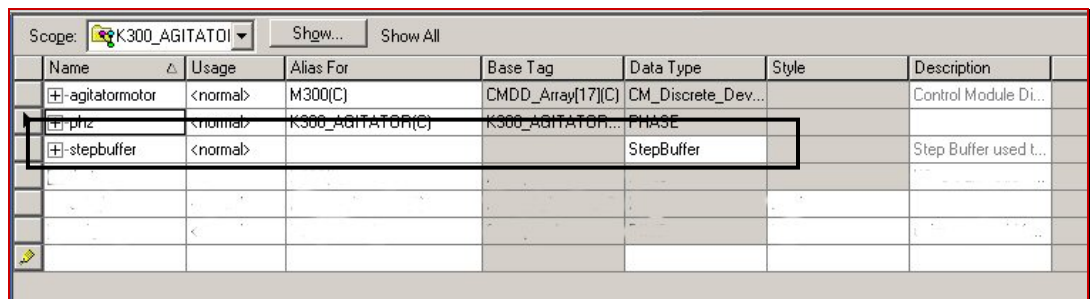
Tip 3: Hold Logic

PhaseManager does not have the native StepBuffer / Hold Index management that is part of the PLI with OPC-based phases. The following steps discuss building hold index management:

1. Build a User Defined Data Type and call it **StepBuffer**. It should contain the following elements:
 - **HoldIndex** – of DINT data type, which holds the desired Running State Re-entry Step Index for any given time.
 - **Restore** – of BOOL data type; condition when to restore the Step Index back to the StepBuffer.



2. Build a Local Phase Tag of type StepBuffer; call it **StepBuffer**.



Tips on Using PhaseManager with RSBizWare Batch

- The states of Running, Holding and Resetting will each have some logic. Place the logic at the top of the each state routine. The logic (shown in Structured Text format) is as follows:

```
// For the Running logic, if the Restore flag is on restore the holdindex to the stepindex.
```

```
IF StepBuffer.Restore THEN
```

```
    Phz.StepIndex := StepBuffer.HoldIndex;
```

```
    StepBuffer.Restore := 0;
```

```
END_IF;
```

```
// For the holding logic, set the restore flag;
```

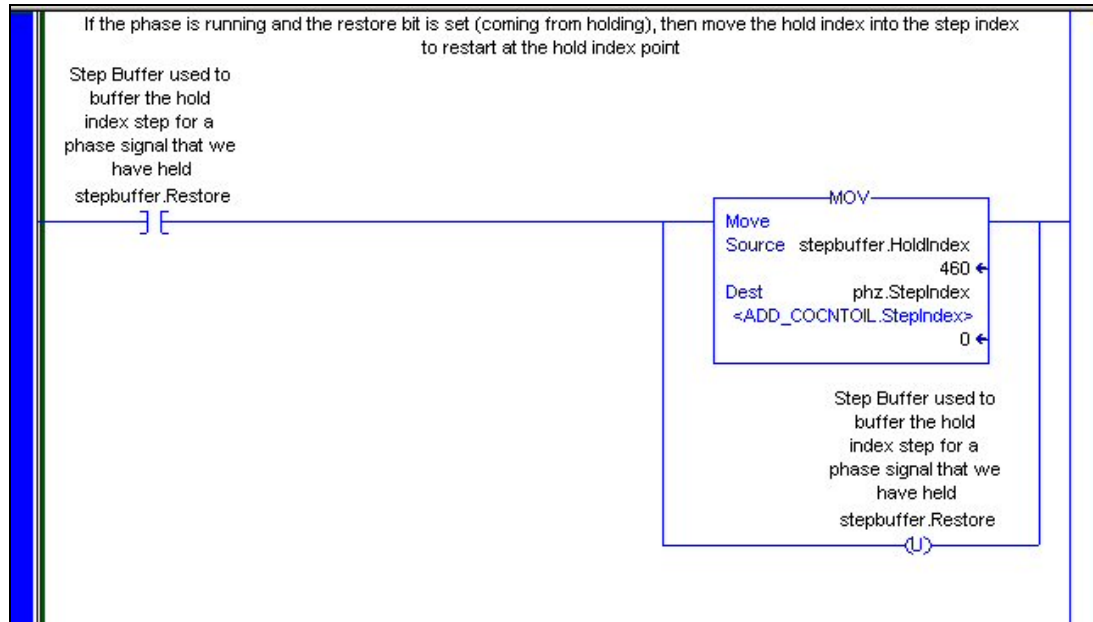
```
StepBuffer.Restore := 1;
```

```
// For the Resetting logic, turn off the RESTORE flag (in case we aborted / stopped from a hold state).
```

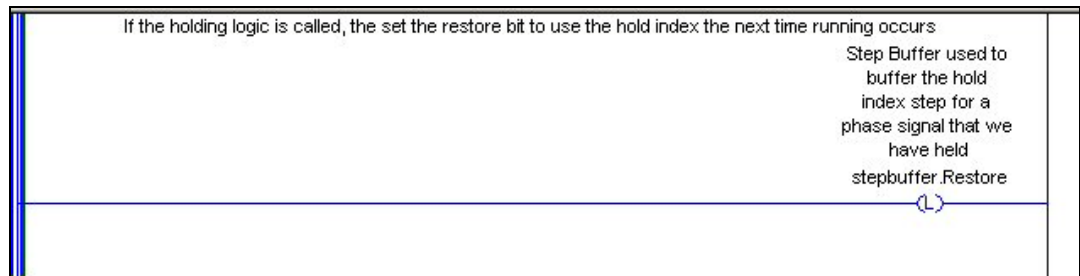
```
StepBuffer.Restore := 0;
```

The ladder implementation of the structured text is shown in the three examples below:

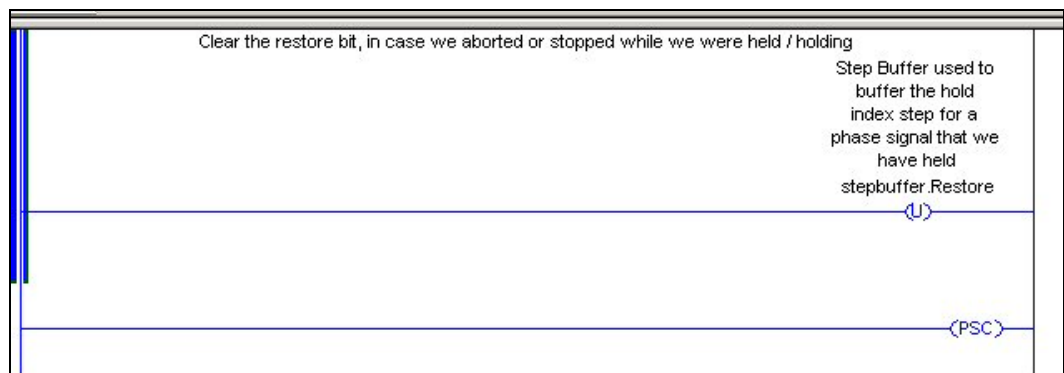
Running Logic



Holding Logic



Resetting Logic



Tips on Using PhaseManager with RSBizWare Batch

- In the running logic:

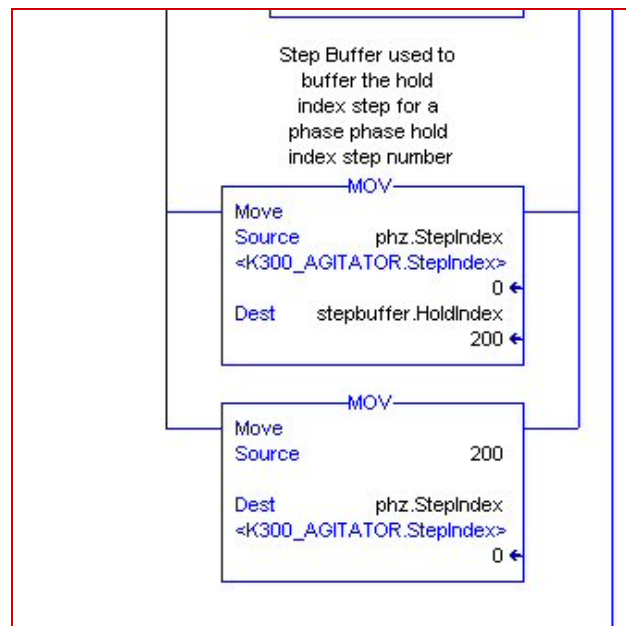
```
// set the HoldIndex to the current step if this is a desired point of return from hold
```

```
IF Phz.StepIndex = 200 THEN
```

```
    Step Instructions ...
```

```
    StepBuffer.HoldIndex := Phz.StepIndex
```

```
END_IF;
```



Tip 4: Unit Tags

Unit Tags still access the CLX registers using an OPC topic. If you have unit tags you must create an additional CLX (of type OPC / RSLinx) in the area model. Since we are using OPC to talk to the CLX, you must also create the watchdog handshake routine in the CLX (to make ensure OPC communication integrity).

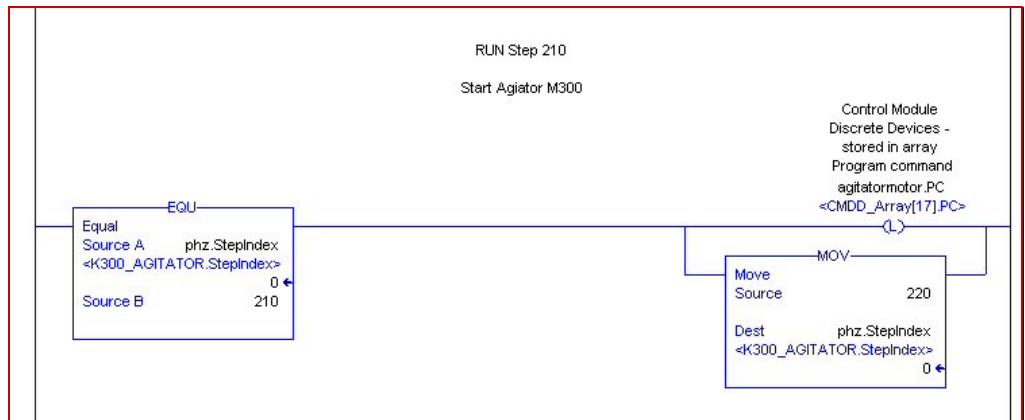
Tip 5: Using Local Aliases

There are major advantages in using local aliases to build phase logic. If you use a local alias to global tags, you can copy and paste code between phases.

For example:

Name	Usage	Alias For	Base Tag	Data Type	Style	Description
[-] agitatormotor	<normal>	M300(C)	CMDD_Array[17](C)	CM_Discrete_Dev...		Control Module Di...
[+] phz	<normal>	K300_AGITATOR(C)	K300_AGITATOR...	PHASE		
[-] stepbuffer	<normal>			StepBuffer		Step Buffer used t...

The "phz" tag is an alias to the phase named K300_Agiterator. The "agitatormotor" is aliased to the control module M300. In the logic then:

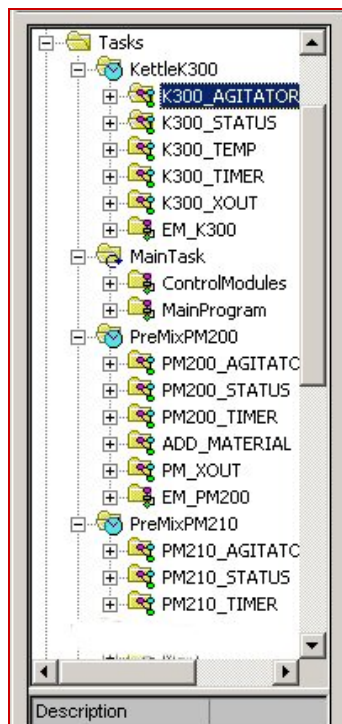


In this example, the code uses the **phz** alias for all references to the phase; all commands to the control module are to the alias control module. This code can be copied and pasted into another agitator phase. By implementing aliases that point to a new phase and control module, the phase would be complete and ready to run.

You can also alias the local parameters and reports to global tags if you want to make them visible on an HMI or to another phase.

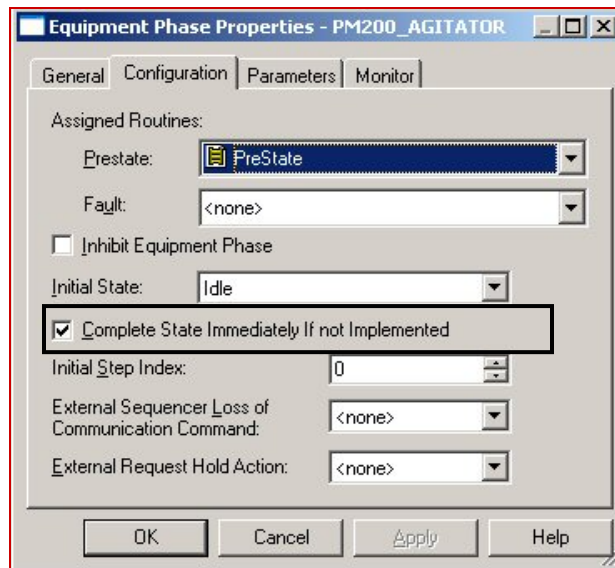
Tip 6: Group the Phases by Unit

After running a SYNC from the Equipment Editor (select **Synchronize Logix5000 Data Servers** from the **Edit** menu), all the phases will be placed in the “unscheduled program”. It is a good practice to move the phases into periodic tasks named after the units, as shown below. The SYNC will still synchronize changes made to either the area model or the phases even though you have moved them to a different task. The periodic task should only be as fast as needed to conserve CPU; 100 – 500 msec for most processes.



Tip 7: Do Not Implement if Not Required

If you do not have any logic to put in a phase, such as the holding, re-starting or resetting logic, you do not need to create a phase state routine for it. But make sure you have the "Complete State Immediately if not Implemented" box checked on the phase properties, as shown below.



LISTEN.
THINK.
SOLVE.SM

**Rockwell
Automation**

www.rockwellautomation.com

Power, Control and Information Solutions

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 SA/NV, Vorstlaan/Boulevard du Souverain 36, 1170 Brussels, 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