Controller Message Queue Logger Example Notes

Introduction

Opto 22’s SNAP PAC System is programmed using PAC Control software. The Controller Message Queue Logger example consists of an ioControl program (called a *strategy*) and an ioDisplay program (called a *project*). ioControl and ioDisplay are part of the ioProject software suite. ioProject is the predecessor to the PAC Project software suite.

You can use the sample strategy as a starting point for building your strategy or you can incorporate the logic into an existing strategy. Likewise, you can use the sample ioDisplay project as a starting point for building your project or you can simply incorporate the historic logging into your existing project.

This document describes how the example works and how to use it.

Note – The controller’s message queue is an expanded version of what used to be called the controller’s error queue. The message queue contains error, warning, and informational messages, and can also include user messages.

PAC Project and ioProject Compatibility

The strategy was created using ioControl Basic R7.0d. The project was created using ioDisplay Configurator Basic R7.0c. The strategy and project can be converted to newer versions of ioProject or PAC Project software, as needed. This document will use the term Control to refer to ioControl and PAC Control, and Display to refer to ioDisplay and PAC Display.

Purpose of Message Queue Logger

The message queue logger Control strategy copies message queue entries to a persistent string table. The related message logger Display project logs each new index of the persistent string table to the computer’s hard drive. This provides greater visibility and insight into your system and makes it significantly easier to troubleshoot problems when they arise. By implementing message logging in your system from the beginning, all error, warning, informational, and user messages are automatically logged to your computer’s hard drive as they occur. This provides a history of how your system runs under normal conditions. When problems occur, you will have a history of normal messages as well as error messages to help troubleshoot the cause.

If you periodically review the files logged to the computer’s hard drive, you may be able to identify problems before you even notice symptoms. Also, when a problem does occur, it will be helpful to be able to review the log files of when the system was functioning correctly.

Overview of Functionality

Control Strategy

The Powerup chart executes a block called “Startup Messages” in order to capture key information regarding the date and time of each occurrence of the controller powering up as well as each occurrence of the strategy being started. This provides a method of determining if power to the controller was interrupted or if a user stopped and restarted the strategy.

The Powerup chart then starts the chart named Message_Queue_Logger_Chart. This chart checks the controller’s message queue and copies the messages to a persistent string table. The Display project logs each new entry in the string table to a historic log file on the computer’s hard drive. As long as no changes are made to the persistent string table configuration, the data in the string table will be retained, even if a modified version of the strategy is downloaded.

Note – Persistent variables retain their values through downloads of updated copies of the same strategy. However, persistent variables will be cleared and recreated if the any aspect of the persistent variable is changed. For example, a persistent variable will not retain its value if the name, table length, or string table width is changed.

Display Project

The key aspect of the Display project is the configuration of a historic data log to log each new entry in the string table to the computer’s hard drive. There is also a summary window that displays the total number of times the controller has powered up and the total number of times the strategy has been started since the last time the strategy was downloaded to the controller. It also shows the current index used for copying queue messages to the table, and the index that the Display project is using to log messages from the table. The two indexes are equal when the Historic log has logged all of the messages.

Integrating Message Queue Logger Functionality into Your System

You can either start with the examples as the foundation of your system or you can add the code into your existing strategy and project.

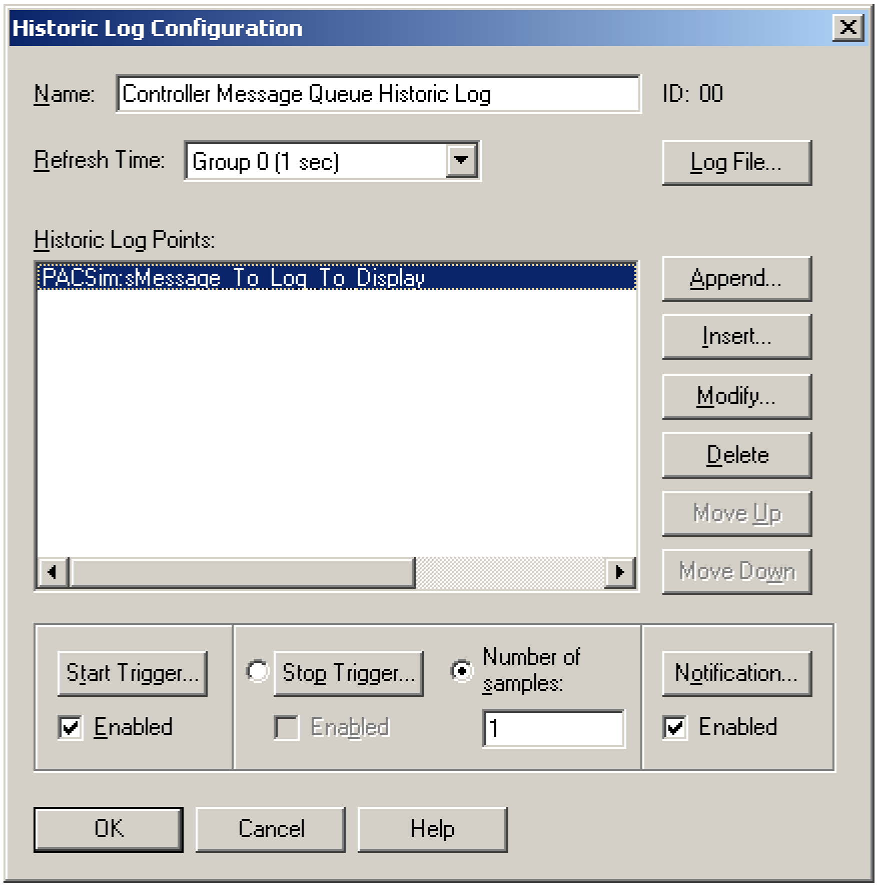
Control Strategy

Export each of the charts from the example strategy, and then import them into your strategy. When importing the charts, choose the option to create new charts rather than importing the logic into existing charts. This is true of the Powerup chart from the example as well. After importing the charts, copy the 2 yellow blocks (blocks 76 and 67) from what was the Powerup chart in the example, and paste them into your Powerup chart. They should be the first blocks executed after Block-0 in your Powerup chart. Then you can delete that chart. This will leave the Message_Queue_Logger_Chart as part of your strategy.

Note - It is always a good idea to leave Block-0 empty in all charts because it makes it easy to insert logic at the beginning of the chart when the need arises.

Display Project

Create a Historic Data Log (Configure menu > Historic Data Log) with the Historic Log Point being the string named sMessage_To_Log_To_Display.



Historic Log Configuration

Name: Controller Message Queue Historic Log

ID: 00

Refresh Time: Group 0 (1 sec)

Historic Log Points:

PACSim:sMessage_To_Log_To_Display

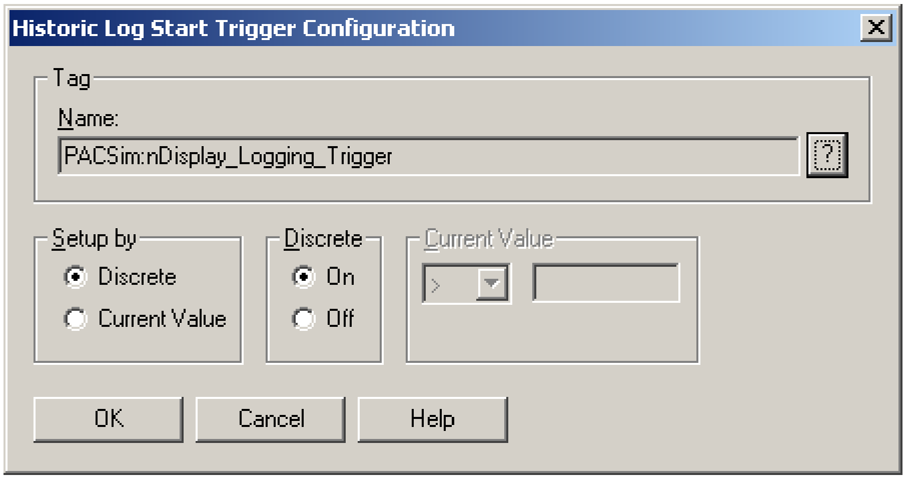
Start Trigger... ☑ Enabled

○ Stop Trigger... ☐ Enabled

● Number of samples: 1

Notification... ☑ Enabled

Configure a Start Trigger that is set up by Discrete ‘On’ using the integer tag named nDisplay_Logging_Trigger.



Historic Log Start Trigger Configuration

Tag

Name: PACSim:nDisplay_Logging_Trigger

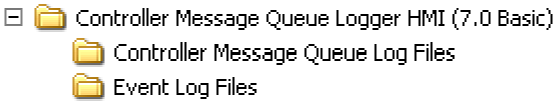
Setup by: ● Discrete ○ Current Value

Discrete: ● On ○ Off

Current Value: >

Set the Number of samples to 1, and then configure the Notification to be set up by Discrete 'Clear' using the same tag used for the Trigger (nDisplay_Logging_Trigger).

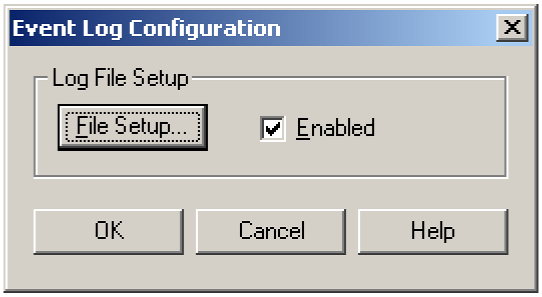
Using Windows Explorer (or similar), create 2 subdirectories from the directory where your Display project is located. One should be named "Controller Message Queue Log Files" and the other "Event Log Files".



- Controller Message Queue Logger HMI (7.0 Basic)
  - Controller Message Queue Log Files
  - Event Log Files

Click the Log File button and set the Directory Path Name to be "Controller Message Queue Log Files". This will keep all of the message queue log files together. Set the Rollover period to Months, and set the number of files to retain to a value of 60. This will allow you to accumulate a history.

Next, Configure the Event Log (Configure menu > Event Log) to log to file by selecting the checkbox named Enabled:



☑ Enabled

Click the File Setup button and set up the Directory Path Name, file Rollover, and Number of files to retain using similar values as the Historic Data Log.

Description of Important Strategy Variables

stMessage_Queue_Log_Table:

- This table stores the messages before they are removed from the controller's message queue. When the table has filled up, it wraps back around to the beginning (index 0), so it is essentially a circular buffer.
- You can make this table longer if you want to have a longer running list of errors.
- You can adjust the table *width* (the maximum size of each string it can hold) depending on how long or short your tagnames are. The initial width of 300 characters takes into account all the information available from the controller's message queue and assumes the maximum size (50 characters) for all tag names. It also leaves room for possible future expansion.
- This is a persistent table, so the data will remain in the table when the strategy is stopped and started, when power to the controller is

turned off and back on, and when modified copies of the strategy are downloaded. Because of this, the related index variables are also persistent.

Startup_Message_Chart

nController_Powerup_Count:

- Keeps track of the number of times the controller has powered up since the last strategy download.

nStrategy_Start_Count:

- Keeps track of the number of times the strategy has been started since the last strategy download.

nUp_Time_New (persistent variable):

- Controller Up Time at the time the strategy was started; used for determining if the controller has just powered up.

nUp_Time_Previous (persistent variable):

- Controller Up Time at the time the strategy was started the previous time; used for determining if the controller has just powered up.

Message_Queue_Logger_Chart

The main variables are related to Logging errors to the Display Historic Log file and making sure the trigger handshaking does not get out of sequence.

nDisplay_Historic_Logging_Enable_Flag: Set this variable True (1) if you are logging to Display Historic Log; set it False (0) to disable Display Historic Logging.

nDisplay_Logging_Trigger: This variable is set True (1) by strategy logic when there are messages in the string table that have not yet been logged; it set False (0) by the Display notification feature indicating it has logged the message.

Version Information

Version 1.0:

7/2/08 Created initial version.