

Optimizing PAC Project System Performance

Introduction

This technical note provides tips to help you design your PAC Project™ system and program your PAC Control™ strategy to ensure optimum system performance. These tips are also helpful if you have already designed your system, and can be used during or after system startup.

All aspects of system performance are addressed, including:

- How fast the system responds to conditions
- Human Machine Interface (HMI) or Supervisory Control and Data Acquisition (SCADA) responsiveness

Control system performance is influenced by many factors, including:

- The size of the control system
- The design of the control system
- The algorithms used in control system logic
- How remote I/O data is accessed
- How HMI or SCADA software accesses system data

All of this is detailed in the following sections:

["Host Communication Performance" on page 2](#)

["Strategy Logic and I/O Communication Performance" on page 8](#)

["HMI, SCADA, and OPC Server Performance" on page 10](#)

Optimizing system throughput takes time and effort. Implement the optimization in stages and check the performance after completing each step to evaluate the results.

Host Communication Performance

By understanding how the control engine uses multitasking, you can increase the host task's frequency, and design efficiencies into your strategy to optimize throughput between the PC and the control engine.

This section includes the following topics:

- [“Multitasking Terminology” on page 2](#)
- [“Understanding PAC Control Multitasking” on page 3](#)
- [“Adjusting the Host Task Frequency” on page 4](#)

Multitasking Terminology

Chart. A PAC Control flowchart programmed by the user.

Multitasking. A method that enables multiple charts or tasks to share the single CPU of the PAC control engine. Only one chart or task executes instructions at any given time. Multitasking schedules which chart or task will execute logic at any given time. The scheduling is handled in a round-robin fashion.

Round-robin. A method of continuously sequencing through the host task and each of the active charts, giving each chart equal time slices. The round-robin can have up to 16 or 32 charts running simultaneously in addition to the host task, depending on the model of control engine used. However, optimal host communication performance is achieved when fewer charts are running.

Time slice. PAC Control allocates a 1 mSec time slice to the host task and each of the active charts in the round-robin. The very small size of the time slices allows multiple tasks to run almost simultaneously.

Active chart. A chart is considered to be *active* if it is running or suspended. An active chart takes up one of the available slots in the round-robin.

Host Task. Used for communications between host devices and the control engine. Host devices include computers running PAC Control or PAC Display™, or SCADA/HMI software, or Operator Interface Terminals (OITs) like the OptoTerminal-G70/G75. A host can be any device that communicates to a PAC Controller using the Opto22 PAC Control Host Protocol.

The host task has the following attributes:

- Handles host communication initiated by remote clients such as PAC Control, PAC Display, OPC clients, and HMI nodes
- Always runs. The host task runs even when there is no strategy present on the controller.
- Is very efficient, and gives up its time slice when there are no communication requests pending

- Cannot be started, stopped, or suspended by the user
- Part of the round-robin, just like all active charts. Because of this, you can optimize host communication performance by writing effective strategy logic.

Understanding PAC Control Multitasking

PAC Control engines allow multiple charts to run simultaneously. The total allowed depends on the controller type:

- SNAP PAC S-series control engines can run up to 32 charts simultaneously, in addition to the host task.
- SNAP PAC R-series control engines can run up to 16 charts simultaneously, in addition to the host task.

Total tasks in the round-robin rotation include:

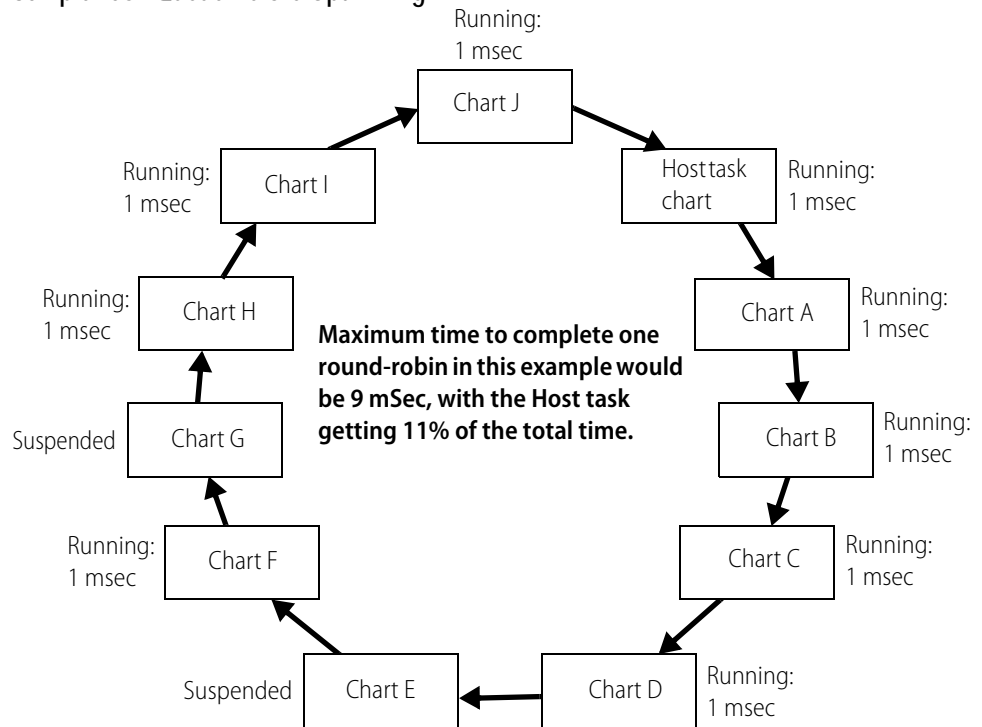
- The Host task
- The Interrupt chart, if present
- All other active charts in your strategy, which means all charts that are either running or suspended. Charts that are stopped are not part of the round-robin.

NOTE: A subroutine assumes the time slice of the chart that called it.

In order for tasks to run simultaneously, the controller uses multitasking to give each task a 1 mSec time slice. The control engine rotates through the tasks in a round-robin fashion. If a task is finished before its time slice is up, the control engine releases the time slice and moves to the next task sooner. For example, a suspended chart uses no time, even though it is counted as a task in the round-robin rotation. Also, a chart that is in the process of executing a Delay command does not use any time.

The following sample task queue shows charts executing in a round-robin rotation, the progression of tasks, and the time each task is taking:

Sample Task Queue Before Optimizing



Adjusting the Host Task Frequency

Use the information in the following topics to optimize the host task frequency:

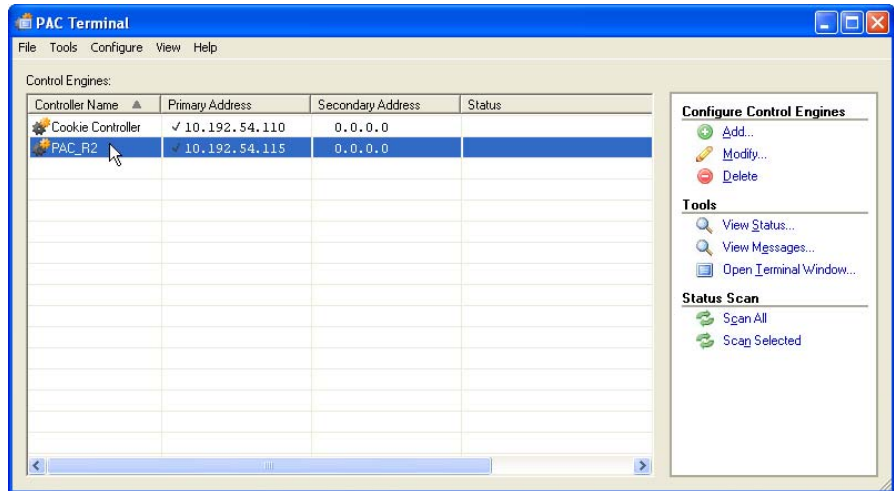
- [“Keeping Tasks in the Round-Robin to a Minimum”](#) (below)
- [“Writing Efficient Conditional Logic”](#) on page 6
- [“Running Charts on a Schedule”](#) on page 7

Keeping Tasks in the Round-Robin to a Minimum

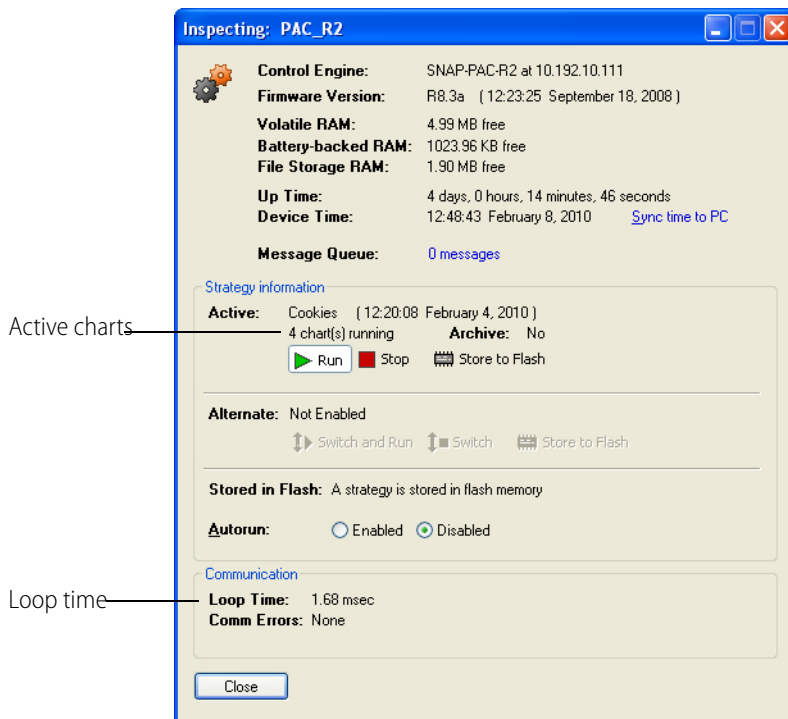
One way to improve host communication throughput is to design your strategy to have fewer tasks in the round-robin so that the host task gets its time slice more frequently. The round-robin completes more quickly and gives the host task time slices more frequently. Like a user chart, the host task is a task in the round-robin of charts, with each task getting equal time slices. With more charts running, it takes longer to complete the round-robin; with fewer charts, the round-robin completes more quickly.

To see whether fewer tasks in the round-robin will help, run this simple test:

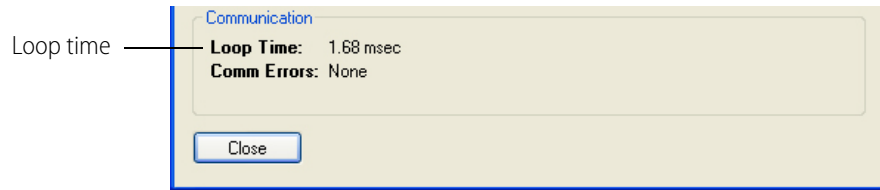
1. Select Start > All Programs > Opto 22 > PAC Project > Tools > PAC Terminal.



2. While the strategy is running, double-click on the controller to inspect the status of the control engine.
3. Take note of the communication loop time and the number of charts that are running.



4. When permissible, stop the strategy and check the communication loop time again.



If the loop time was significantly lower when the strategy was stopped, you may be able to improve throughput by minimizing the number of charts that run simultaneously.

Most strategies can be designed effectively with only five or ten charts running simultaneously.

If there are 32 charts running simultaneously, then there are 33 tasks in the round-robin, including the host task. The host task will get 1/33rd of the overall processing time (3%). However, if there are nine charts running simultaneously, then there are ten tasks in the round-robin. The host task will be getting 1/10th of the overall processing time (10%).

So if your host port throughput is not what you expect it to be and you have many charts running simultaneously, you can consolidate some of the active charts to reduce the overall number of charts running at any given time.

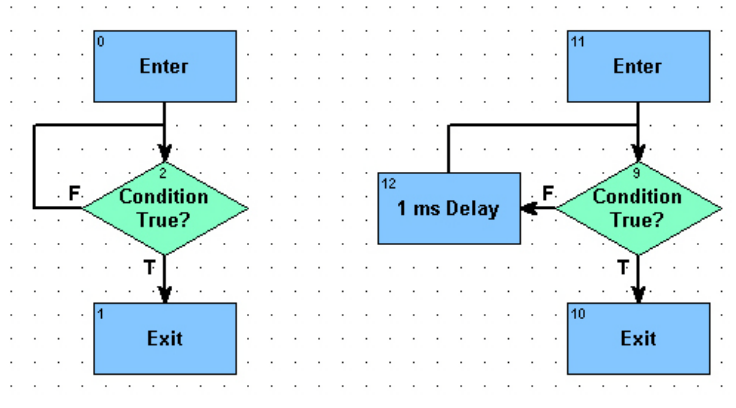
Writing Efficient Conditional Logic

Another method to give the host task more processing time is to take advantage of the fact that whenever the controller encounters a Delay command in chart logic, it will give up the remainder of the time slice for that chart (the smallest delay is 1 mSec and time slices are 1 mSec each). When the round-robin has come back around to that chart, the controller will check to see if the delay time has expired, and if not, it will again give up the time slice for that chart. Checking to see if a delay has expired takes essentially no time, so this means the round-robin will complete more quickly.

Within a chart, if a condition block continually loops waiting for the condition to be true, the entire time slice for the chart is used up, and this is inefficient.

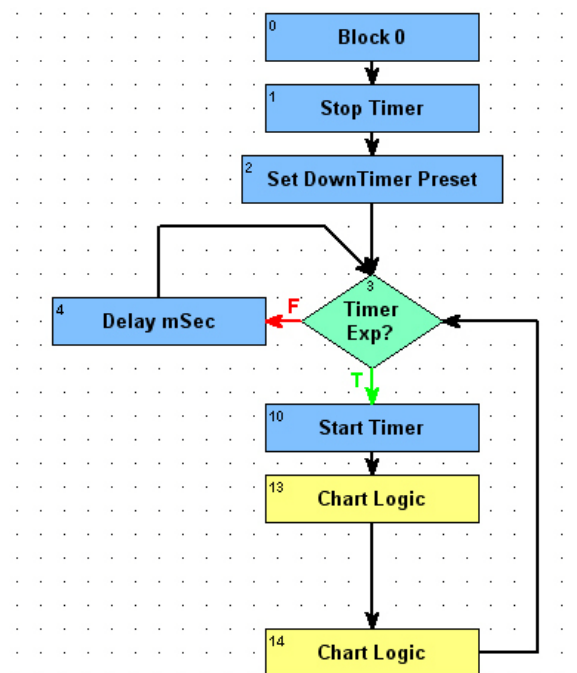
Evaluate how frequently the condition must be checked. Then within the loop, insert a reasonable delay by using the Delay (mSec) command. The Delay command causes the chart to immediately give up its time slice. When the delay expires, the chart will inspect the condition again. In the meantime, the chart releases processor time to other charts.

The following graphic shows two conditional loops. In the example on the left, the entire time slice is used up waiting for the condition to become true. However, on the right, the delay allows the control engine to run other tasks while waiting for the condition to become true.



Running Charts on a Schedule

When writing flow charts that run continuously, there is usually not much thought given to how often that logic really needs to be executed. If it is critical that the chart run as fast as possible all the time, like an Emergency Stop (E-Stop) chart, that's fine, but most charts don't need that kind of speed. Once you determine how often each chart really needs to run, you can run it on a schedule using a timer at the top of the chart. When the timer expires, restart the timer, and execute the logic. If the timer has not expired, use the Delay (mSec) command with a value of 1 (as discussed on [page 6](#)) to give up the time slice before checking the timer again. This allows other charts and the host task to run more frequently.



Strategy Logic and I/O Communication Performance

Throughput between the controller and I/O is affected by strategy design. You can take advantage of I/O unit commands to communicate with several I/O points at once, and in some situations you can use more than one port to communicate with serial I/O.

This section includes the following topics:

- [“Using I/O Unit Commands” on page 8](#)
- [“Using Ethernet I/O Units” on page 8](#)
- [“Using Serial I/O Units” on page 8](#)
- [“Consolidating Data from Common I/O Units” on page 9](#)
- [“Handling I/O Errors Efficiently” on page 9](#)

Using I/O Unit Commands

PAC Control's rich features allow you to select between different commands to access I/O. Some commands, like Turn On or Move, allow accessing one point per transaction. Other commands, like Move I/O Unit to Numeric Table or Move Numeric Table to I/O Unit, allow the reading of all points on an I/O unit in a single transaction, which is much more efficient.

It is most efficient to use I/O unit commands to read the I/O because this minimizes the number of transactions to the I/O unit and maximizes the amount of data retrieved from the I/O unit in each transaction. I/O unit commands typically put the I/O data into tables, so write your logic to directly access the I/O table data or move the table data to variables and write your logic to access these variables.

Using Ethernet I/O Units

Using Ethernet-based I/O units, such as the SNAP-PAC-EB1 and EB2, in a PAC system provides much better performance than serial I/O units. Ethernet is very fast and efficient, and can transport large amounts of data in each transaction.

In addition, using Ethernet I/O units instead of serial I/O units significantly improves throughput because each chart in the controller can communicate independently with the I/O units.

In applications requiring the highest performance, instead of using the SNAP-PAC-EB1 or EB2 as the I/O unit, consider using the SNAP-PAC-R1 or R2. A PAC R has both I/O unit and control engine functionality. However, when used as an I/O unit, the R-series' more powerful hardware and the ability to shut down the control engine portion gives the R1 and R2 the best overall I/O unit performance.

Using Serial I/O Units

When there are requirements to use serial I/O units, make sure to employ efficient techniques to maintain system robustness.

The best controller to use with serial I/O units is the SNAP-PAC-S2. The SNAP-PAC-S2 has four I/O-capable serial ports.

Each chart can have only one serial port open at a time and each serial port can be used by only one chart at a time. Because of this, it helps to use one chart per port to simultaneously access I/O on multiple ports. Each chart should communicate with I/O on one port only. For example, if I/O is connected to Serial 1 and Serial 2, it is best to communicate to the I/O on Serial 1 from Chart1 and the I/O on Serial 2 from Chart2.

If the I/O on certain I/O units have a higher priority, then consider placing the lower priority serial I/O units on one serial port and divide higher priority I/O units among the remaining serial ports.

In a strategy with serial I/O units, it is very important to implement the I/O-scanning charts first. Use I/O unit commands to move I/O points to tables. This is the most efficient way to scan the I/O points. I/O unit commands access all I/O points on the I/O unit and store the data in tables in the strategy. The I/O point values moved to tables provide cached values that can be used by strategy logic and HMI's without having to scan the data one point at a time. Design the rest of the strategy logic based on the values cached in tables.

Consolidating Data from Common I/O Units

Minimize the number of charts that must communicate to a common I/O unit. If many charts use the data from the same I/O unit, appoint one chart to access the I/O unit, allowing the other charts to share the data. This works best when using I/O unit commands.

Check Controller's Message Queue for Errors

1. Check the controller's message queue to determine if there are any errors occurring in the controller that might impact system performance.
 - a. If logic from the Message Queue Logger sample strategy was implemented in the strategy, then any errors would have been moved to a string table and possibly logged to the computer's hard drive.
 - b. Some users clear errors from the controller's message queue using strategy logic.
 - i. PAC Control commands used to remove errors are:
 - Remove Current Error and Point to Next Error
 - Clear All Errors
 - ii. It may be necessary to temporarily comment-out this type of logic in order to be able to determine if errors are occurring in the controller.

Handling I/O Errors Efficiently

If the control engine encounters an error when communicating to I/O, it disables communication to the I/O unit. Disabling communication ensures that control engine performance is maintained. If an I/O timeout error occurs and communication with the I/O

unit is not disabled, it will delay chart execution while it waits for a response each time it tries to access the I/O unit that is offline. In addition, for serial I/O throughput to the remaining I/O units on that serial port will drop while the control engine tries to communicate.

For efficient I/O unit communication management and error logging, see the I/O Enabler and the Message Queue Logger sample strategies provided with PAC Control and also available from the [Samples & Freeware](#) section of the downloads page on www.opto22.com.

Investigate and correct the root cause of any I/O unit communication error to maintain the best overall performance.

HMI, SCADA, and OPC Server Performance

This section includes the following topics:

- [“Improving Data Throughput For PAC Display” on page 10](#)
- [“Improving Data Throughput For OPC Clients” on page 11](#)
- [“Minimize the Number of OPC Servers” on page 11](#)
- [“Enabling Table Optimizations in the OptoOPCServer” on page 12](#)
- [“SCADA Implementation of I/O Tags” on page 13](#)

Improving Data Throughput For PAC Display

When using PAC Display on a system with Ethernet I/O Units, the default configuration is for PAC Display to scan I/O tags directly from the I/O units and only scan strategy tags (variables and tables) from the control engine. This default configuration provides the best overall throughput. It is possible to change the configuration to scan all tag types, including I/O tags, through the control engine, however this reduces overall throughput.

To optimize data throughput for PAC Display, do the following:

- Identify and correct any configuration issues. In PAC Display Configurator, open the project and choose Configure > AutoCorrect Tags. Correct any discrepancies that are identified.
- Use the default PAC Display option of scanning Ethernet I/O unit tags directly from the I/O units. This requires an Ethernet network configuration where the I/O units are accessible by the PAC Display computer. This can be configured by viewing Configure > Runtime > I/O Unit Tags.
- Keep I/O tags in refresh groups that are separate from PAC Control strategy tags. Group strategy variable and table tags into refresh groups that do not have I/O tags.
- Use reasonable Refresh Group time intervals.
- In order to optimize the scanning of table elements, see [“Enabling Table Optimizations in the OptoOPCServer” on page 12](#).

- For Serial I/O units, the most efficient method is to use I/O Unit commands (instead of individual I/O point commands) in strategy logic to move serial I/O unit tags into tables. Then scan the data from the tables instead of scanning I/O points directly. If you must scan serial I/O tags directly, you should keep this to a minimum because serial I/O tags must be scanned through the control engine. In addition, only one command can be active on a serial port at a time. If a host scans serial I/O information through the control engine, it causes the controller to have to scan all of the appropriate serial I/O units at that time, and it can only reply to the host after getting the responses from all appropriate I/O units. See also the previous topics, [“Using I/O Unit Commands” on page 8](#) and [“Using Serial I/O Units” on page 8](#).

Improving Data Throughput For OPC Clients

The OptoOPCServer™ allows any OPC 1.x or 2.x client to access data from SNAP-PAC Ethernet-based devices. OPC clients range from SCADA software systems to custom designed OPC clients. Identify and correct any configuration issues. Do all OPC tags have the correct OPC Item ID, IP address, TCP port, syntax, and OPC Scanner type?

When scanning Ethernet I/O unit tags, the best performance is obtained by scanning the I/O units directly rather than through the control engine. To do this, use the MMIO (memory map I/O) scanner. Also, use separate OPC Groups for tags on separate I/O units. I/O tags should use the “MMIO” scanner. TCP port 2001 is the default port for Ethernet brains and the I/O side of R1 and R2 controllers. Strategy variables will use the “CONT” scanner. TCP port 22001 is the default port for the controller’s host task.

It is best to scan control engines for variables and tables only (not I/O tags). The OptoOPCServer can also optimize table elements to further enhance throughput. (See [“Enabling Table Optimizations in the OptoOPCServer” on page 12](#).) The best performance will occur if table element tags are put into their own OPC groups; do not mix I/O tags into these OPC groups.

In addition, it is best to limit the number of tags in each group to 2,500.

If the system uses serial I/O units, avoid scanning of these tags if at all possible. Instead of scanning serial I/O tags through the control engine, access cached values in tables in the control engine. For more information, see [“Improving Data Throughput For PAC Display” on page 10](#). If you must scan serial I/O tags, try to keep it to a minimum.

Minimize the Number of OPC Servers

While it is possible to have multiple OPC Servers actively scan the same controllers and Ethernet-based I/O units, this causes additional data requests for each device to process. This keeps the host communication task busier and causes a device response slowdown to all of the OPC Servers.

One advantage of the OPC architecture is that multiple clients can access a single OPC Server. This way, the host task only has to process requests from a single source. The OptoOPCServer was designed specifically to be a high performance, multi-threaded

application. For information on configuring a shared OPC server, see form 1439, the *OptoOPCServer User's Guide*.

If you are a SCADA user, it is best to run the SCADA's data server on the same computer that operates the OptoOPCServer. SCADA clients share the data from the SCADA server. This is a typical feature of the SCADA, and it's the best implementation for data access.

Enabling Table Optimizations in the OptoOPCServer

The OptoOPCServer can optimize common table element requests by merging them into a single table request. This is performed within each OPC group. Overall data throughput will be much better because there will be fewer transactions to read the same amount of data.

NOTE: This feature is available in version R8.2a or newer of OptoOPCServer.

To make the most of the table optimization feature, use chart logic to consolidate numeric tags that need to be scanned into 4 large tables: 2 float tables (one for tags that have to be scanned more quickly and the other for tags that are scanned at a slower scan rate) and 2 integer 32 tables (one for tags that have to be scanned more quickly and the other for tags that are scanned at a slower scan rate).

Then reconfigure the OPC tags to access the 4 larger tables. There should be one update rate for the faster tag tables and one update rate for slower tag tables.

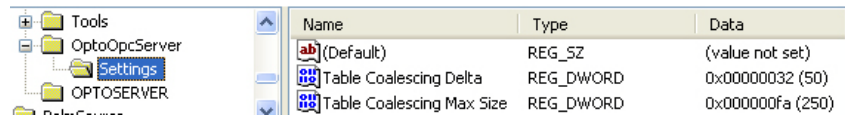
To enable this table optimization feature, do the following steps on the computer running the OptoOPCServer:

1. To ensure the OptoOPCServer is not running, shut down all OPC clients, and then open the Windows Task Manager. Click the Processes tab, select "Show processes from all users," and verify that the program `optoopc.exe` is not listed.
2. Click on Start, and run `regedit`.
3. For 32-bit Windows, browse to `HKEY_LOCAL_MACHINE\Software\Opto22\`.
For 64-bit Windows, browse to `HKEY_LOCAL_MACHINE\Software\Wow6432Node\Opto22\`.

NOTE: There is no space in Opto22 in this registry key.

4. Create a key named `OptoOpcServer`. (This is case-sensitive, as are all data in the registry).
5. Open the `OptoOpcServer` key and create a key named `Settings`.
6. Open the `Settings` key and create a `DWORD` value named `Table Coalescing Delta`. Set the value to 32 hex (50 decimal).
7. Create another `DWORD` value named `Table Coalescing Max Size`, and set it to FA hex (250 decimal).

This will result in registry entries that look like this:



Name	Type	Data
ab(Default)	REG_SZ	(value not set)
Table Coalescing Delta	REG_DWORD	0x00000032 (50)
Table Coalescing Max Size	REG_DWORD	0x000000fa (250)

SCADA Implementation of I/O Tags

SCADA systems are designed around the monitoring or reading of I/O devices. In the SCADA world, both I/O and controller tags are considered to be I/O. SCADA tags have attributes of read-only and read-write. Note that both options require a read, so that if the SCADA has I/O tags that are only intended to be written to, the SCADA will activate these tags for reading even though the value is never used. When the tags are activated, the OPC server begins reading them from the appropriate devices. This is a limitation because it causes I/O data to be read unnecessarily and adds to the overhead of the system.

For tags intended to be write-only, minimize the number of tags. In addition, place these tags in their own group with a very relaxed group update rate, such as 15 to 20 seconds. Do not set the group update too close to 30 seconds because many SCADA OPC clients have an internal timeout of 30 seconds. If the OPC client does not receive a data update within this time, the client treats this condition as an OPC Server exception and will falsely report an OPC Server failure.

Other Considerations

Consider also the controller model being used, and whether you want to add additional host tasks.

Controller Model

If the controller is an R1 or R2, consider upgrading to an S1 or S2 controller for strategy logic and use the existing R1/R2 as an I/O unit for the I/O points on that rack. The S1/S2 is faster and should provide improvement in strategy performance and host port throughput.

Additional Host Tasks

In conjunction with using an S1/S2 controller for running the strategy, additional host tasks can be used. You may want to add 1 or 2 additional host tasks using the PAC Control strategy command Start Alternate Host Task. You will need to reconfigure the HMI or SCADA accordingly:

1. Each additional host task counts against the total number of charts in the round robin.
2. When adding host tasks, each host task will have a different TCP port number, so the OPC tags in the SCADA will need to be changed to reflect the appropriate TCP port numbers.

- a.** This will result in the OptoOPCServer starting separate scanner threads for each host task on the controller.
 - b.** If PAC Display is being used, you will need to configure additional controllers with the appropriate TCP port numbers for each additional host task.
 - 3.** One approach would be as follows:
 - a.** Use the default host task (TCP port 22001) for the PAC Control debugger.
 - b.** For data that needs to be updated faster, read those tags using one of the additional host tasks;
 - c.** For data that needs to be updated slower, read those tags using the other additional host task.