OPTIMIZING PAC PROJECT SYSTEM PERFORMANCE

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.

The technical note covers PAC Project running on *groov* EPIC®, SoftPAC, and SNAP PAC controllers. It also covers *groov* RIO® and SNAP I/O units. 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 and design of the control system
- The type of control engine
- The algorithms used in control system logic
- How remote I/O data is accessed
- How HMI or SCADA software accesses system data

This technical note is in three sections:

- "Host Communication Performance on the PAC Control Engine" on page 1
- "Strategy Logic and I/O Communication Performance" on page 8
- "HMI, SCADA, and Other Software 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 ON THE PAC CONTROL ENGINE

Host communication refers to communication between host software (like PAC Project, an HMI, or SCADA systems) and the PAC control engine, whether that control engine is in the device itself, as with *groov* EPIC processors and SNAP PAC controllers, or on a Microsoft® Windows® PC running SoftPAC.

This section focuses on factors related to the choice of platform (*groov* EPIC, SNAP PAC, or SoftPAC) as well as how the PAC Control strategy is architected and programmed. In a later section, we'll address factors related to HMI, SCADA, and OPC scanning.

Improving host communication performance is almost an art, because so many variables are involved. To begin understanding why, let's take a look at the CPU differences in SNAP PAC, *groov* EPIC, and SoftPAC.

CPU differences

SNAP PAC controllers have a single-core CPU and an embedded real-time operating system with a limited number of processes. Running the PAC Control strategy is the CPU's primary function.

***groov* EPIC** processors have a multi-core CPU that runs a custom build of the open-source Linux® operating system (OS). In addition to the PAC control engine, *groov* EPIC can run a variety of other processes simultaneously, including Ignition or Ignition Edge® from Inductive Automation®, Node-RED, a *groov* View HMI, an MQTT data service, the EPIC's RESTful API, and necessary Linux processes. Your industrial application may include some or all of these, so the number of processes vying for CPU time varies depending on how you use *groov* EPIC.

MADE IN THE USA

SoftPAC is similar to *groov* EPIC in the sense that its PAC control engine also must deal with competing processes on the CPU. SoftPAC runs on a Microsoft Windows PC that may be running an assortment of other processes at the same time—PAC Display or a third-party HMI, a SCADA system, and anything else that could run on Windows, plus Windows processes. The number and CPU usage of other processes greatly affect host communication, and host communication efficiency greatly affects other processes.

Implications for host communication

These differences in CPU construction and number of processes make a huge difference in how your PAC Control strategy is handled.

In SNAP PAC controllers, multitasking uses a *round-robin* of PAC Control flowcharts. The round-robin continuously sequences through the host task and each of the active charts, giving each one a time slice of about 1 millisecond (ms). (A chart is considered to be *active* if it is running or suspended.) The very small size of the time slices allows multiple tasks to run almost simultaneously.

In *groov* EPIC and SoftPAC, the OS (Linux or Windows) determines the core on which a task or chart runs. There is no round-robin of PAC Control charts. The OS must balance the full processing load—not just PAC Control, but all tasks running on the device. Charts and processes are distributed among the available CPU cores. Tasks (including running charts) can be interrupted if a higher-priority task needs to run. So the amount of time each chart gets will never be identical.

Diagnosing high CPU usage

When CPU usage is very high, you may see symptoms like these:

- Timeouts in charts that handle communications
- Delays in refreshing a PAC Display HMI
- Slow updating of variables in the debugger
- Long response times in the controller status dialog box
- In *groov* EPIC or SoftPAC, delays or timeouts in other processes

Diagnosing the reasons for high CPU usage is difficult, however.

On SNAP PAC controllers, you cannot view CPU usage. But you know that running the PAC Control strategy is the primary job for the controller. If your controller is a SNAP PAC R-series, you can improve performance by upgrading to a SNAP PAC S-series controller (or better yet, a *groov* EPIC processor) for strategy logic, using the existing R1 or R2 as an I/O unit. With any SNAP PAC controller, understanding how the round-robin works and then making some basic changes can improve performance. See "SNAP PAC controllers: round-robin" on page 3 for more information.

On *groov* EPIC and the PC where SoftPAC resides, the number of processes running on the CPU makes diagnosis more difficult. You can usually view CPU usage, but it may be shown as a percentage or a number, and the value depends on the number of CPU cores and threads. A good diagnostic test is to stop the PAC Control strategy and see how that impacts CPU usage. Then stop each of the main processes the CPU is using, one at a time, and see the effect on CPU usage.

Understanding PAC Control tasks

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

- *groov* EPIC processors and SoftPAC on Windows can run up to 64 charts simultaneously, in addition to the default host task.
- SNAP PAC S-series control engines can run up to 32 charts simultaneously, in addition to the default host task.

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

The control engine executes the following tasks:

- Default host task
- Alternate host task, if used. (See the PAC Control command, "Start Alternate Host Task.") Note that any alternate host tasks count against the maximum number of active charts allowed.
- Interrupt chart, if present. (The interrupt chart is a special chart used with legacy serial I/O units that have an interrupt line. If the Interrupt chart is present but not used, delete it so it will not be active in the background.)
- All other active charts in your strategy, which means all charts that are either running or suspended. Charts that are stopped are not active tasks. A subroutine uses the same task as the chart that called it.

The host task

The host task is used for communication between the control engine and host devices, including computers running PAC Control, PAC Display, or SCADA/HMI software. A host can be any device that communicates to an Opto 22 controller using the Opto 22 PAC Control Host Protocol.

The default host task:

- 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 PAC Control strategy present on the controller.
- Is very efficient, and takes up no CPU time if there are no communication requests pending
- Cannot be started, stopped, or suspended
- Uses TCP port 22001 by default

Alternate host tasks can be started using the PAC Control command, "Start Alternate Host Task." See the *PAC Control Command Reference* for details, including TCP port selection.

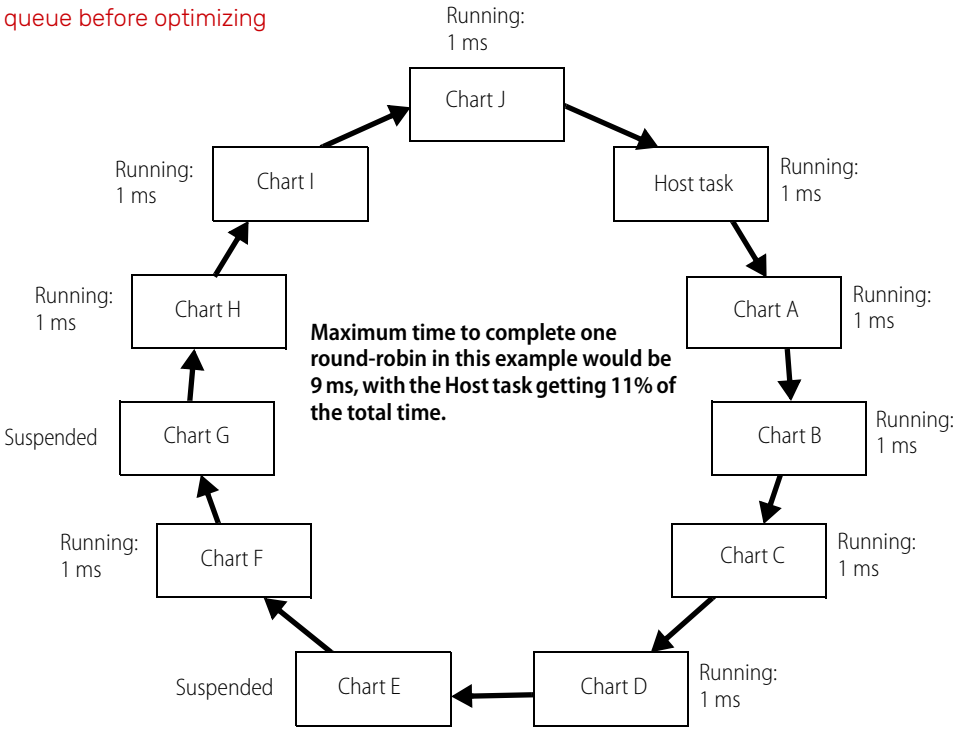
SNAP PAC controllers: round-robin

By understanding how SNAP PAC controllers use 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.

In order for tasks to run simultaneously, the controller gives each task a 1 ms time slice and rotates through them in a round-robin. 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 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 being executed in a round-robin rotation, the progression of tasks, and the time each task is taking:

SNAP PAC controllers: Sample task queue before optimizing



Improving host communication

No matter which model controller you use, however, you can improve throughput with some simple changes in your strategy. Here are five ways to optimize strategy execution:

- Add delays for more efficient CPU use (below)
- Write efficient conditional logic (page 5)
- Keep tasks to a minimum (page 5)
- Run charts on a schedule (page 7)
- Start an alternate host task using another TCP port (page 7)

Add delays for more efficient CPU use

The most important way you can use the CPU more efficiently and give the host task more processing time is to take advantage of the Delay (mSec) command in your chart logic. Adding delays makes an enormous difference:

- On SNAP PAC, if a chart is waiting for a condition to become true and doesn't have a delay in the conditional loop, the chart will spend the full time slice continually checking the condition. With a delay, the chart will give up its time slice and move on to the next chart.
- On *groov* EPIC, if you run a strategy with 10 charts, nine of them that use delays and one that doesn't, that one monopolizes the core it's running on. It may be difficult to see unless you look in the charts, but that missing delay is why the CPU usage is higher.

We recommend using delays in all looping charts. A looping chart is one that continually loops through logic, rather than running straight through and ending. We also recommend using delays in all subroutines,

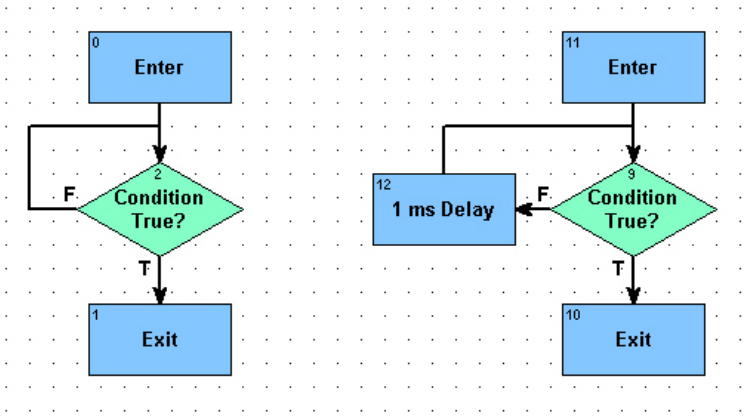
because subroutines use the same task as the calling chart. Two Delay commands are available, one in milliseconds and one in seconds. The smallest possible delay is 1 ms.

Write efficient conditional logic

Delays are especially useful in a conditional loop. If the chart continually loops waiting for the condition to be true, the chart uses its full time slice or is always running, occupying the CPU unnecessarily. That time could be used much more efficiently elsewhere, whether for the PAC Control strategy, other processes running on the controller or PC, or both.

Evaluate how frequently the condition must be checked. For example, temperatures typically change slowly and may need checking only every few minutes; a switch may need to be checked every 100 ms; an emergency condition needs to be checked regularly. Once you've determined the frequency needed, insert a reasonable delay within the loop by using one of the Delay commands.

The following graphic shows two conditional loops. In the example on the left, the chart constantly runs until the condition becomes true. However, on the right, the delay allows the control engine and the processor to run other tasks while waiting for the condition to become true.

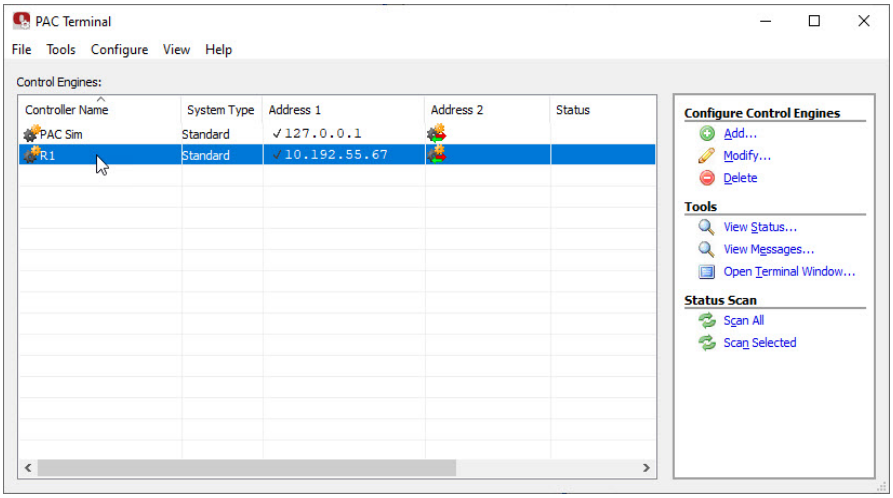


Keep tasks to a minimum

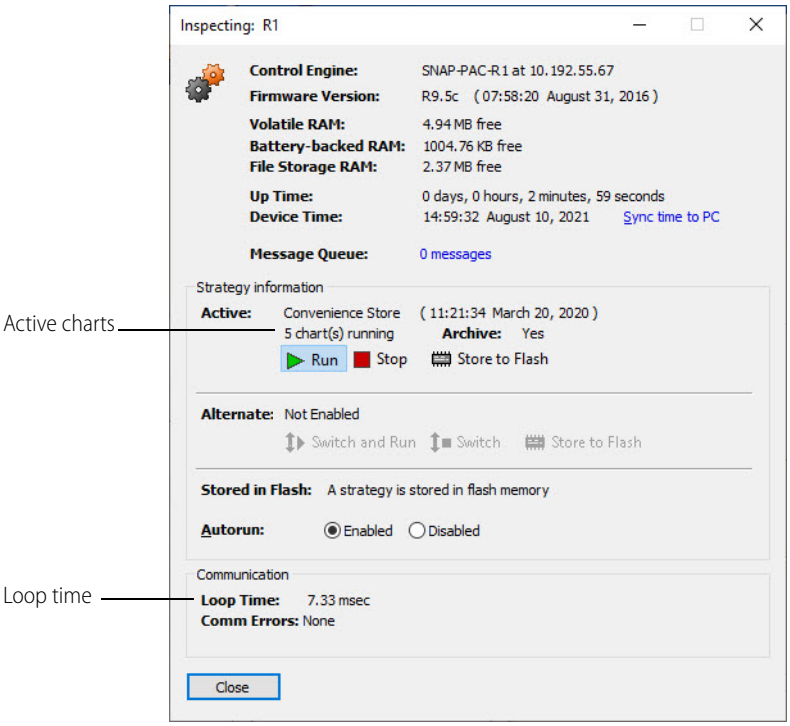
Another basic way to improve host communication throughput is to design your strategy to run fewer charts simultaneously. Although you can have up to 16, 32, or 64 charts running simultaneously—in addition to the default host task—optimal host communication performance is achieved when fewer charts are running.

To see whether fewer charts 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 the controller to inspect the status of the control engine.
3. Note 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.

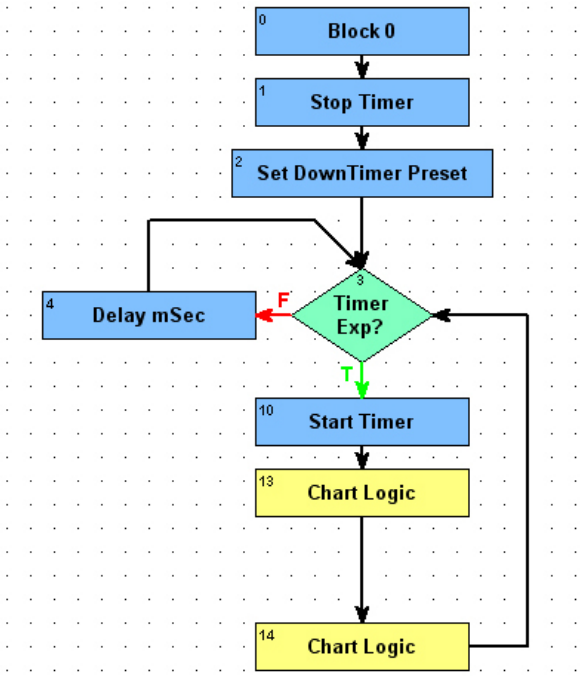
Most strategies can be designed effectively with only five or ten charts running at once. If your host port throughput is not what you expect it to be and you have many charts running at once, you can consolidate some of the active charts to reduce the number of charts running at any given time.

Run charts on a schedule

When writing flowcharts that run continuously, consider 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 (as discussed on page 4) to let the control engine move on before checking the timer again.

The delay allows other charts, the host task, and other processes to run more frequently.



Start an alternate host task

To divide up the host workload, you can use the PAC Control command Start Alternate Host Task. For example, with an alternate host task, human-machine interface (HMI) communication can be directed to an alternate TCP host port, leaving the default host free for other uses.

For *groov* EPIC, you can add an alternate host task for *groov* View or MQTT communication without affecting firewall security. For example, when configuring MQTT in *groov* Manage, you can choose the port number that matches the alternate host task port you've set up in PAC Control (see "MQTT Communications (groov EPIC, groov RIO)" on page 13). Similarly, in *groov* View, you can choose an alternate host task port number when configuring an EPIC controller device. Because both the host task and the requestor (MQTT or *groov* View) are on the EPIC, no firewall change is necessary.

However, to use an alternate host task for other communications with *groov* EPIC (for example, from a PC running PAC Display), you must create a new firewall rule to open the alternate host port. Opening a port can affect system security, so choose carefully whether to do so.

If you add alternate host tasks, note the following:

- Each additional host task counts against the total number of charts that can run simultaneously.
- Each additional host task has a different TCP port number. Change the OPC tags in the SCADA system to reflect the appropriate TCP port numbers.
  - If you are using PAC Display, configure additional control engine definitions with the appropriate TCP port numbers for each additional host task.
  - Note that OptoOPCServer will start separate scanner threads for each IP address/TCP port combination.

Here is one approach:

1. Use the default host task (TCP port 22001) for the PAC Control debugger.
2. For data that needs to be updated frequently, read those tags using one of the additional host tasks.
3. For data that needs to be updated less frequently, read those tags using another additional host task.

See details for the Start Alternate Host Task command in the *PAC Control Command Reference*.

STRATEGY LOGIC AND I/O COMMUNICATION PERFORMANCE

Throughput between the controller and I/O is also affected by strategy design. For exmple, you can take advantage of commands that communicate with several I/O points or an entire module at once.

Here are some ways to design your strategy for more efficient I/O communication:

Use I/O Unit commands

PAC Control's rich features allow you to select among 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 Ex" or "Move Numeric Table to I/O Unit Ex" allow reading or writing to 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 these commands minimize the number of transactions to the I/O unit while maximizing the amount of data retrieved 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.

See the I/O Unit Commands group in the *PAC Control Command Reference* (form 1701) for details on these commands.

Use Digital Module commands

Digital Module commands, like I/O Unit commands, let you read or write to multiple points in a single efficient transaction. For example, you can read the states of all digital points on one module at once, or get and clear all counters or latches with one command. The result of a Digital Module command is typically a bitmask or table.

Using whole-module commands like these minimizes the number of transactions and improves throughput. See the Digital Module Commands group in the *PAC Control Command Reference* (form 1701) for details on these commands.

Use Ethernet I/O units

***groov* EPIC systems** support Ethernet-based I/O units only, both *groov* RIO and SNAP PAC Ethernet I/O units. Ethernet is very fast and efficient and can transport large amounts of data in each transaction. Also, because individual charts in the PAC Control strategy and some other processes running on *groov* EPIC can each communicate independently with Ethernet I/O units, overall system performance is improved.

SNAP PAC systems can use either Ethernet or serial I/O units, but Ethernet I/O units provide much better performance. 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 Ethernet I/O units.

In SNAP PAC system applications requiring higher 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 turn off the control engine portion give the R1 and R2 the better overall I/O unit performance.

Use efficient techniques for serial I/O units in SNAP PAC systems

When you are required to use serial I/O units (SNAP PAC systems only; serial units are not supported in *groov* EPIC), 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 has a higher priority, then consider placing all lower priority serial I/O units on one serial port and dividing 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 HMIs 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.

Consolidate 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 the controller's message queue for errors

Check the controller's message queue to determine if there are any errors occurring in the controller that might impact system performance.

NOTE: If you used logic from the Message Queue Logger sample strategy in your strategy, any errors are moved to a string table and possibly logged to the computer's hard drive.

Some users clear errors from the controller's message queue using strategy logic. PAC Control commands used to remove errors are:

- Remove Current Error and Point to Next Error
- Clear All Errors

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.

Handle 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, the control engine delays chart execution while it waits for a response each time it tries to access the I/O unit that is off line. In addition, for serial I/O, the remaining I/O units on that serial port are not accessed 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 & Utilities section of the Support > Downloads page on opto22.com. Search for the sample strategy name.

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

HMI, SCADA, AND OTHER SOFTWARE PERFORMANCE

When using host software—an HMI, a SCADA system, or an OPC server, for example—one major key to performance is the way the I/O tags are scanned. Any time I/O tags are scanned through the control engine's host port, all communication on that host port is slowed during any transaction that includes I/O tags. The host task processes one transaction at a time. While a transaction is in progress, any other commands received on that host port are buffered and handled in order.

When a transaction includes only strategy variables and tables, the host task can read those items directly. But for I/O scanned through the control engine's host port, the host task has to spend extra time to scan the related I/O units and get those tags. When it has all the I/O tags and strategy variables and tables, it then sends the response and moves on to the next command that it buffered.

When possible, it is much better to scan I/O tags directly from the I/O units. One thing to note is that when using alternate host tasks, each host task is completely independent of the others. If one host task is busy scanning I/O, it doesn't impact the other host tasks. The ability to scan I/O directly or use alternate host tasks varies depending on the software you are using; details are in the next several sections.

Here are some ways to improve performance for systems using the following HMI, SCADA, and other software:

- PAC Display HMI (groov EPIC and SNAP PAC)—page 11
- groov View HMI (groov EPIC)—page 11
- Ignition from Inductive Automation (groov EPIC)—page 12
- Node-RED (groov EPIC, groov RIO)—page 13
- MQTT Communications (groov EPIC, groov RIO)—page 13
- OptoOPCServer (groov EPIC, groov RIO, and SNAP PAC)—page 14
- RESTful API (groov EPIC, groov RIO, and SNAP PAC)—page 15
- Software Development Kits—page 16

PAC Display HMI (*groov* EPIC and SNAP PAC)

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 scan only strategy 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 change from the default significantly 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 Tools > AutoCorrect Tags. Correct any discrepancies that are identified. Then choose Tools > Regenerate I/O Scanner Tag Names.
- Use the default PAC Display option of scanning Ethernet I/O unit tags directly from the I/O units. This option requires an Ethernet network configuration where the I/O units are accessible by the PAC Display computer. If the I/O units are on a different subnet than the control engine's host port, then add a second network adapter to the computer to give it access to the I/O network. To configure, choose Configure > Runtime > I/O Unit Tags.
- Use reasonable Refresh Group time intervals.
- To optimize table element scanning, see "Enable table optimizations in OptoOPCServer" on page 14.
- If you use PAC Display on multiple computers to scan the same controller, you can improve throughput by configuring each of the PAC Display projects to scan using OptoOPCServer. Configure one computer to run OptoOPCServer, so that a single scanner (OptoOPCServer) scans the controller. See more information about OptoOPCServer on page 14.
- For serial I/O units in SNAP PAC systems, keep I/O tags in refresh groups that are separate from PAC Control strategy tags. Put strategy variable and table tags into refresh groups that do not have I/O tags.
- 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, keep it 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 tags through the control engine, the controller has to scan all of the appropriate serial I/O units one at a time, and it can reply to the host only after getting responses from all appropriate I/O units. Also see the previous topics, "Use I/O Unit commands" on page 8 and "Use efficient techniques for serial I/O units in SNAP PAC systems" on page 9.

groov View HMI (*groov* EPIC)

Here are several ways to improve performance with your *groov* View HMI. See the *groov View User's Guide* (form 2027) for more information.

Avoid reading I/O directly via a PAC Control strategy

When adding devices in *groov* View's Build mode (Configure > Devices and Tags > Add New Device), consider the most efficient way to scan tags.

The Opto 22 Controller device can be used to scan all tags, including I/O tags, via the PAC control engine's default host task (TCP port 22001). But as with other host software, scanning I/O tags through the host port can slow performance significantly. Instead, use one (or both) of these two options:

- Add an Opto 22 I/O Unit device to your *groov* View project and configure the gadgets to use the I.O Unit device tags. With this option, *groov* View scans the I/O tags directly from the I/O unit (UDP on port 2001), which improves performance. It is common to have both

Opto 22 Controller and I/O Unit devices configured in a project. Be sure to add Opto 22 I/O Unit devices at the beginning of your project; making these changes later is much more tedious.

- Edit the PAC Control strategy to copy the I/O points you want to tables or variables and configure the gadgets to use those tags. This option eliminates the need for *groov* View to scan I/O tags

Choose your trend type wisely

For real-time monitoring, use Classic Trends. Use Interactive Trends for long-term monitoring.

- Classic trends store and report data directly, without any special interpretation.
- Interactive trends have to store multiple down-sampled copies of their datasets in order to stay interactive. Storing adds both time and space overhead and also can lead to accuracy issues when you zoom out of the trend to view longer time scales.

Download long-term trend data regularly

The download links in trends can be saved and used without signing in, allowing you to automate trend data backups in any way you like. Doing so can both prevent data loss and give you more flexibility in how you use your trend data.

Use the Device Health inspectors

In *groov* View's Build mode, you can see a real-time report of *groov* View's connection with a device (Configure > Devices and Tags > [select your device] > Device Health). The report can tell you if the data you're seeing from your devices is lagging behind real time.

Remove unneeded gadgets from the page stash

Gadgets left in the page stash in Build mode are still considered part of the page and are scanned when that page is open in any browser. Some gadgets may be in the stash because they are used in the Desktop & Tablet layout but not the Handheld layout, or vice versa; those should stay. But any gadgets that aren't used in the project should be removed for better performance.

Design your images for their intended display size

groov View doesn't know how you plan to use images in your project and doesn't resize them for you. Use images sized for your project's needs. For example, if your images will be used only on a small screen, use small images. Reducing image size saves heavily on data transferred to your computers out in the field, where internet connections may be spotty.

Reduce project backup size

If you don't need to back up your trend data or event log, leave them out of your project backups. Trend data in particular can balloon the size of your project backups. If you don't need it, leave it out when backing up your project.

Ignition from Inductive Automation (*groov* EPIC)

Ignition runs on *groov* EPIC, but the Ignition Driver Module can access *groov* EPIC, *groov* RIO, and SNAP PAC I/O units. The Driver Module can be configured to use the control engine's default host task (TCP port 22001) or an alternate host task TCP port number (see "Start an alternate host task" on page 7).

- All tags are scanned from the selected host task, including I/O tags.
- All items marked public in the strategy are scanned.

To improve scanning and overall host task performance:

- Scan only the required tags by marking only those tags as public in the PAC Control strategy.

- If possible, scan only PAC Control strategy variables and tables, not I/O. Use strategy logic to move I/O to tables or variables and scan those, instead of scanning the I/O directly.

Node-RED (*groov* EPIC, *groov* RIO)

Node-RED on *groov* EPIC uses the PAC control engine's default host task (TCP port 22001) for all tags, including I/O tags. Alternate host tasks are not supported. To improve scanning and overall host task performance, scan only PAC Control strategy variables and tables. Use strategy logic to move I/O to tables or variables and scan those instead of scanning the I/O directly.

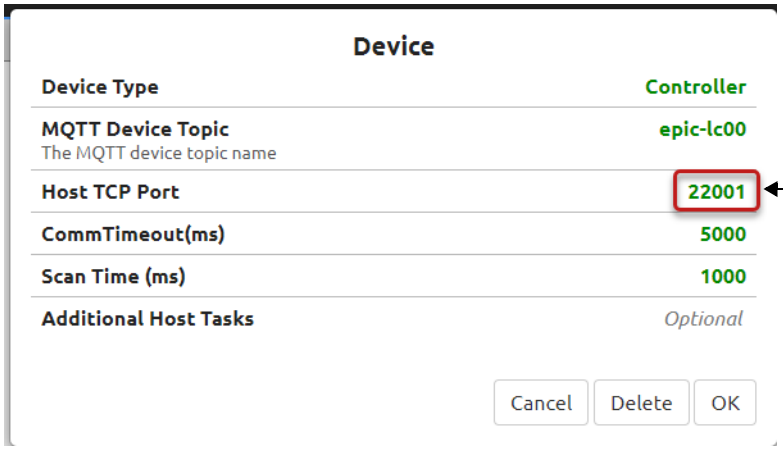
When using Node-RED on either *groov* EPIC or *groov* RIO, also use these tips:

- Whenever possible, increase the interval on inject nodes set to repeat. By default, repeating nodes have an interval of one second, starting one second after you deploy the flow. Many tasks do not need to be triggered this often. Increasing the interval reduces unnecessary processing.
- Reduce the number of things happening at the same time. Offsetting injects and using prime numbers as intervals are great ways to reduce overhead. For example, two injects set to repeat every 3 and 5 seconds will inject at the same time only every 15 seconds.

MQTT Communications (*groov* EPIC, *groov* RIO)

groov EPIC and *groov* RIO can use MQTT for efficient data communication with both the EPIC controller and I/O units. When you configure MQTT, you choose the device type: Controller or OptoMMP (see image below).

Controller device type—For an EPIC processor running PAC Control, you choose the Controller MQTT device type, and MQTT communications use the controller's host port (TCP port 22001) for all tags, including I/O tags. You can set up the service to access an alternate host task by changing the Host TCP Port from the default to the alternate host task port (Home > MQTT > MQTT Configuration > Device):



You can change the Host TCP Port for MQTT from the default Host Task to an alternate host task port number. See "Start an alternate host task" on page 7.

To optimize MQTT from the controller, create a PAC Control chart that writes all the variables to a table. Then use delays or timer logic in the chart to manage how often that table is published.

It is also more efficient to make just that table public, rather than a lot of individual variables. All items marked as public in the PAC Control strategy are scanned. For best performance, scan only the required tags by marking only those items as public.

OptoMMP device type—For *groov* EPIC and *groov* RIO I/O units, you choose the OptoMMP device type. With this choice, MQTT communications use the I/O unit's OptoMMP port 2001 to scan I/O directly, instead of scanning through the controller's host port. All I/O tags you have made public using *groov* Manage are

scanned. Note that you can configure only one tag at a time in *groov* Manage; if you have a large number of tags, configuration can be cumbersome.

OptoOPCServer (*groov* EPIC, *groov* RIO, and SNAP PAC)

OptoOPCServer™ allows any OPC 1.x or 2.x client to access data from *groov* EPIC (if using the PAC Controller rather than the CODESYS Controller), *groov* RIO, and SNAP PAC Ethernet-based devices. OPC clients range from SCADA software systems to custom-designed OPC clients.

Make sure you identify and correct any configuration issues. Make sure all OPC tags have the correct OPC Item ID, IP address, TCP port, syntax, and OPC Scanner type. For details, see Chapter 4, "Opto 22 OPC Item Definitions," in the *OptoOPCServer User's Guide*.

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:

- Use the MMIO (memory map I/O) scanner for I/O tags. TCP port 2001 is the default port for *groov* RIO modules, SNAP PAC Ethernet brains, the I/O side of SNAP-PAC-R1 and -R2 controllers, and the I/O side of *groov* EPIC processors.
- Strategy variables use the CONT (control engine) scanner. TCP port 22001 is the default port for the control engine's host task. It is best to scan control engines for variables and tables only, not I/O tags.
- You can also use an alternate host task for OptoOPCServer. See "Start an alternate host task" on page 7.

The OptoOPCServer can also optimize table elements to further enhance throughput. (See "Enable table optimizations in OptoOPCServer" on page 14.) The best performance occurs when table element tags are put into their own OPC groups; do not mix I/O tags into these OPC groups. Limit the number of tags in each group to 2,500.

If your SNAP PAC system uses serial I/O units, avoid scanning 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 "PAC Display HMI (groov EPIC and SNAP PAC)" on page 11. If you must scan serial I/O tags, try to keep it to a minimum.

Minimize the number of OptoOPCServers

While it is possible to have multiple OPC servers actively scan the same controllers and Ethernet-based I/O units, multiple scans cause 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 application. For information on configuring a shared OPC server, see form 1439, the *OptoOPCServer User's Guide*.

If you are using a SCADA system with OptoOPCServer, it is best to run the OptoOPCServer on the same computer as the SCADA's data server.. 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.

Enable table optimizations in OptoOPCServer

The OptoOPCServer can optimize common numeric table element requests by merging them into a single table request within each OPC group (requires OptoOPCServer version R8.2a or higher). Using a single table request within each group reduces the number of transactions needed to read the same amount of data,which improves overall data throughput.

To make the most of the table optimization feature, use chart logic to consolidate numeric tags that need to be scanned into four large tables: two float tables (one for tags that have to be scanned more quickly and the

other for tags that are scanned at a slower rate) and two integer 32 tables (again, one for tags that have to be scanned more quickly and the other for tags that are scanned at a slower rate).

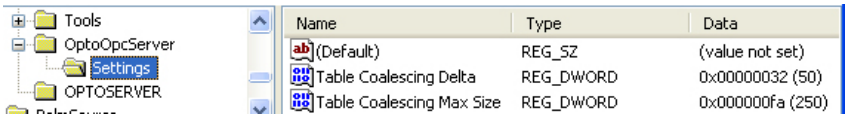
Then configure the OPC client to access the four larger tables, setting one update rate for the faster tag tables and a separate rate for the slower tag tables.

To enable table optimization, follow these steps on the computer running OptoOPCServer:

1. To ensure that OptoOPCServer is not running, shut down all OPC clients. Then open the Windows Task Manager. Click the Processes tab, select "Show processes from all users." If the optoopc.exe process is listed, select it and stop the process.
2. From the Windows start menu, run regedit.exe.
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 key is case-sensitive, as is 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).

You see registry entries that look like this:



Reduce overhead for SCADA systems

SCADA systems are designed to control, monitor, or read I/O devices. In the SCADA world, both I/O and controller tags are considered to be I/O. SCADA tags can be read-only or read-write. Note that both options require a read. If the SCADA has I/O tags that are intended only to be written to, the SCADA still activates 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, causing I/O data to be read unnecessarily and adding 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, however, 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.

RESTful API (*groov* EPIC, *groov* RIO, and SNAP PAC)

On *groov* RIO, all REST API calls go to *groov* Manage through port 443.

On SNAP PAC, all REST API calls use port 443 (https) and are then handled by the PAC control engine's default host task (TCP port 22001). Alternate host tasks are not supported.

On *groov* EPIC, REST API calls for PAC Control strategy tags and I/O tags go through port 443 and are then handled by the control engine's default host task (TCP port 22001). Alternate host tasks are not supported.

When you design your API calls for SNAP PAC and *groov* EPIC, remember how busy the default host task is (see "The host task" on page 3). To improve scanning and overall host task performance:

- Scan only PAC Control strategy variables and tables, not I/O. Use strategy logic to move I/O to tables or variables and scan those, instead of scanning the I/O directly.
- Scan only the necessary tags and make only those tags public. Although it is more convenient to scan all tags of a particular type, performance is better if only necessary tags are scanned.
- Keep scan rates low. Scan only as often as required, based on the tag type and your application.

Software Development Kits

When developing your own custom programs, you can reduce processing through the default host task by using one of Opto 22's free OptoMMP SDKs to scan I/O directly. A free Controller SDK is also available; it is best used only for scanning PAC Control strategy tables and variables.

PAC-DEV-OPTOMMP-DOTNET

The .NET OptoMMP Software Development Kit for *groov* EPIC, *groov* RIO, and SNAP PAC is used to scan I/O tags directly instead of needing to go through the controller's host port. It uses OptoMMP port 2001.

PAC-DEV-OPTOMMP-CPLUS

The C++ OptoMMP Software Development Kit for *groov* EPIC, *groov* RIO, and SNAP PAC is also used to scan I/O tags directly instead of needing to go through the controller's host port. It uses OptoMMP port 2001.

PAC-DEV-CONTROLLER-DOTNET

This .NET Controller Software Development Kit for *groov* EPIC and SNAP PAC is used to scan strategy table and variable tags from the PAC Controller's host port. Because you can select which TCP port to connect to, it can be used with the default host task (TCP port 22001) or an alternate host task.