Introduction

Opto 22's SNAP PAC System and *groov* are flexible tools that offer many choices for developing your monitoring, control, and data acquisition projects.

This flexibility provides the power to do everything you need, but the choices you make can sometimes cause problems: hard-to-follow spaghetti code[1], unnecessarily difficult maintenance, or human-machine interfaces (HMIs) that don't make sense to the people who use them. A little advance planning can help avoid these problems.

This technical note suggests some best practices you can use when developing your control programs and HMIs, to help make building and maintaining your projects easier. These suggestions come from the fields of software development and user interface design as well as the first-hand experience of Opto 22 engineers, integrators, and customers.

Included are tips on building, documenting, and backing up your projects.

Building Your Projects

HMI First

Before building your flowchart structure, plan out your HMI.

Understanding how the process is going to be controlled and monitored from the HMI (whether it be graphical or an operator panel) helps determine how you write your chart code. Plan the inputs, outputs, and variables needed to monitor and control the system before you begin.

If you code the machine operation and user interface first, you'll spend much less time later modifying the chart for human interaction with the system.

Building HMIs: PAC Display and *groov*

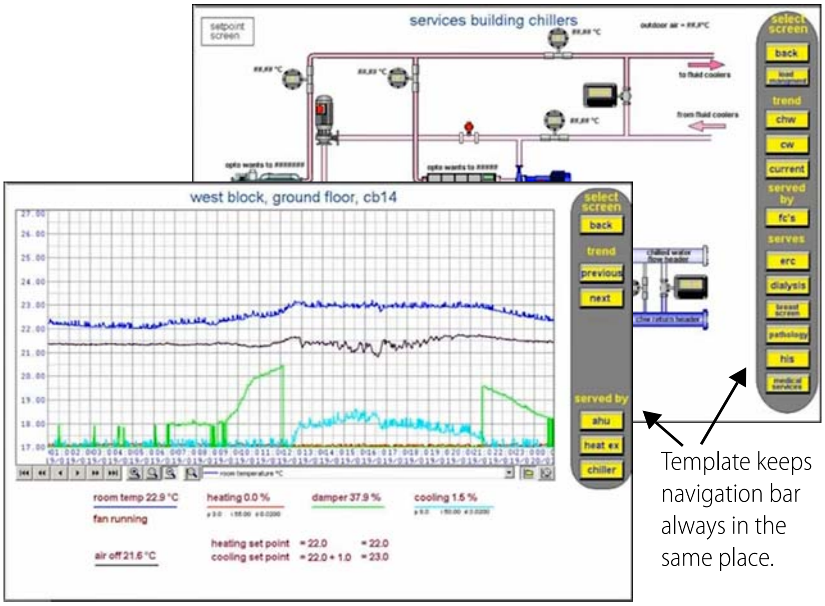
Plan your HMI to be as clean and simple as possible. Watch your end users and talk with them about what they need to do, and then design the interface so they can accomplish their tasks quickly and easily, without distractions or doubt. For specifics, see the white paper, *Building an HMI that Works: New Best Practices for Operator Interface Design*.

Consistency

Keep your PAC Display windows and *groov* pages consistent, so users know what to expect when they move around in the interface. The easiest way to maintain consistency is to develop templates for the windows or pages in your project.

[1] Here's a definition of spaghetti code from Wikipedia: http://en.wikipedia.org/wiki/Spaghetti_code

Standardize fonts and colors. Put common elements, such as navigation bars, always in the same place. When you build a new window or page, just copy the template.[2]



Template keeps navigation bar always in the same place.

For consistency in SuperTrends, adjust the scales of trend pens so that they match, if possible. Scales that match make the trend more intuitive. For example, PID Setpoint and Input should have the same scale. Also adjust time scales so that they make sense and are readable in Runtime.

Consistent color coding in HMIs is a useful indicator as long as you don't have too many colors. (And remember that a certain number of your users will be colorblind, so don't rely on color alone to distinguish items.) In SuperTrends, always use the same pen color for the same signal types. For example, for air temperature always use blue trend lines, but don't use the same blue for a chiller temperature in the same project.

Make line weights consistent, too. For example, use a thicker line for pushbuttons and fields that require operator input, and a very thin line for indicator lights and data readouts.

Window Management

Keep the number of windows to a minimum in your project. Ask yourself if a separate window is necessary, for example for a warning message. If it's specific to the window, consider using a graphic with visibility settings. If the message applies to several windows and you have several messages, try using one warning message window with a string populated from a generic error message trigger variable in the controller.

Instead of making navigation part of your window template, you could create a separate window just for navigation. Lock this window into a fixed position and make it always stay on top, so it can never disappear or be covered. The advantage of the separate window is

[2] Learn more in the PAC Display Tips & Tricks video.

that it’s easier to change navigation: you make changes in one place and don’t have to remember to copy and paste them into other windows.

De-clutter your interface by automatically closing unnecessary windows when others open.

Views in *groov*

groov Build has two views, one for the HMI on a computer or tablet, and the other for the HMI on a handheld device like a phone or mini tablet.

It’s easiest to start building pages in the view that reflects the device most of your users will use for the HMI. But whichever view you start in, switch frequently to the other view to rearrange gadgets and text to be most effective. And be sure to update the other view before moving on to build another page.

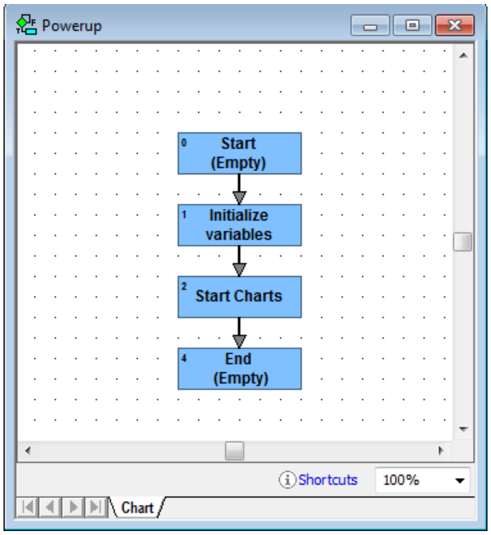
Building Strategies: PAC Control

Here are some basic guidelines to follow when building or updating your control strategy and the flowcharts in it. For more help and an example of how to get from your real-world control problem to a functioning control system, see Chapter 4, “Designing Your Strategy,” in the *PAC Control User’s Guide*. If you’re new to PAC Control, you may find our three QuickStart videos useful.

Whether you’re new to PAC Control or experienced, you’ll find useful information in the *Optimizing PAC Project System Performance Technical Note* (form 1776). Read it before creating your strategy.

Charts

A PAC Control strategy can have many flowcharts, and several can be running simultaneously: up to 16 on a SNAP PAC R-series controller, up to 32 on a SNAP PAC S-series, and up to 64 on SoftPAC.



The Powerup chart (see example at right) is really a special-purpose chart and should be kept simple. It’s best to avoid looping logic in the Powerup chart. Instead, use flow-through logic: initialize your variables, start other charts, and then end. If you have a significant amount of initialization, put initialization code in a separate chart named *Initialization*. Then in the Powerup chart, use the Call Chart command to call the Initialization chart before starting other charts.

Some experienced PAC Control programmers recommend placing a “default values” block at the beginning of the Powerup chart to reset values for important variables in case of a power cycle. Instead of making each variable persistent, you can write their values periodically to a persistent table. The default values block checks to see if a value is zero; if it is, it's replaced by the value from the table.

Add a block named End (empty) to the end of the Powerup chart (and other flow-through charts) just to make it clear that the chart stops. By the way, you don't need to put the Stop Chart command at the end of a chart; if there is no more logic, the chart will stop by itself.

Charts that monitor essential or time-critical pieces of your process, such as emergency stops on dangerous equipment or I/O that must be monitored continuously, should use looping logic so they are constantly running. Looping logic should also be used for alarm control and communication with an external program or device.

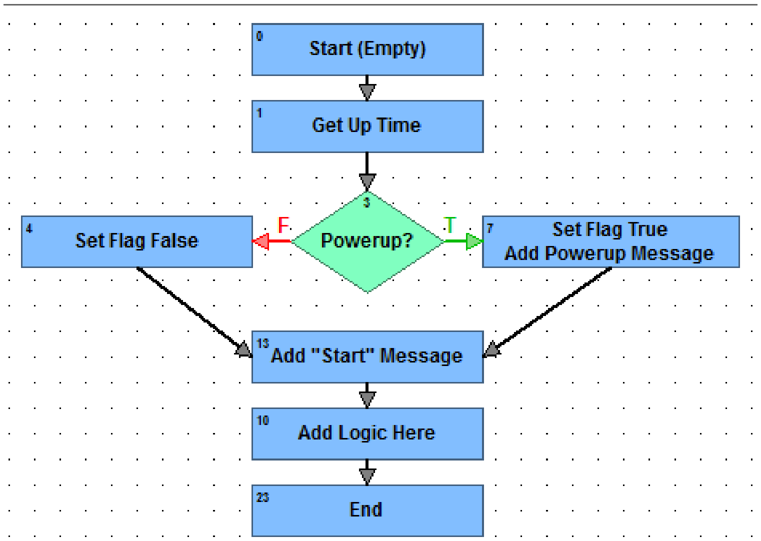
Take advantage of sample charts and code shared by Opto 22 engineers as well as by other customers on the OptoForums. Here's a list of the greatest sample hits available on the forums. Pay special attention to the samples under "Our Product Support Group (PSG) Recommends." Your strategy should include a chart similar to the I/O Enabler, for example, to make sure that any I/O unit that inadvertently goes offline is promptly re-enabled.

For a complex strategy you may want to create a marshalling chart—a chart to monitor all charts. The marshalling chart can gather status data, check chart sequencing, perform simple error checking, and act as the "glue" for a large strategy.

Space within Charts

Keep charts as simple as possible. Lay out your charts neatly, so it's easy to follow chart logic. Keep all your blocks the same size, and make your connection lines neat and clearly visible.

Try to keep charts compact so you don't have to scroll side to side. When possible, standardize on a size for Action blocks that is not as tall as the default. The smaller size saves vertical space in the chart. Use the smallest reasonable space between blocks to keep the chart more compact, but make sure connections and arrowheads are clearly visible.



If you need to zoom out and squint to see the whole chart, consider breaking it into smaller pieces using methods that are appropriate for your application:

- Break the chart into two or more charts based on subprocesses.
- Create reusable subroutines for groups of commands you may need to use in more than one place.[3]
- Put parts of the process into more compact OptoScript blocks.[4]

When you use multiple charts, be aware that they can and do run at the same time, and all variables are global in nature except for those in subroutines. Subroutine variables are local—another advantage of creating subroutines for code you use in multiple places.

Flow

Make your flowchart flow in a logical direction. In regions where people read top to bottom and left to right, make the process flow top to bottom, left to right. Remember that it's easy to scroll up and down but harder to scroll sideways, so try to keep your chart from getting too wide. Let your logic flow down rather than to the side.

Leave Block 0 empty and label it as "Start (empty)," so it's clear to everyone that it is the starting point for this chart. Leaving Block 0 empty also gives you space for later changes: if you have to go back and add logic to the beginning of the chart, all you have to do is insert a new block after Block 0. (See the Powerup chart example on page 3.)

On flow-through charts (those that don't loop), put in an End (empty) block as well, so it's clear that the chart stops there. The End block also makes it easy to catch unconnected logic when using Find to search for bad connections.

Use intuitive coloring to make the flow of logic clear. In condition blocks, for example, you could use red for connecting the false exit and green for connecting the true exit (or choose other colors suited to your application; just be consistent in their use).

IP Addressing

Rack-mounted SNAP-PAC R-series controllers control I/O on the same rack (and can control other racks as well). When you add the I/O unit for the PAC R itself—an I/O unit that's local to the control engine—use the Ethernet Loop-back Address for On-Board I/O Unit. The loopback IP address is127.0.0.1.

The advantage of the loopback address is that if the controller's IP address changes, you don't have to reconfigure the I/O unit address in your strategy. All you need to do is to modify the controller to point to the new address.

Literals and Variables

It's tempting to use literals like 10 or 3.141. But there is no way to do a global search for such a value. You're better off creating variables such as MinutesToWait or PI_TO_3_DECIMAL_PLACES. You can search for these in your chart and easily rename or

[3] See Chapter 12, "Subroutines," in the *PAC Control User's Guide*.

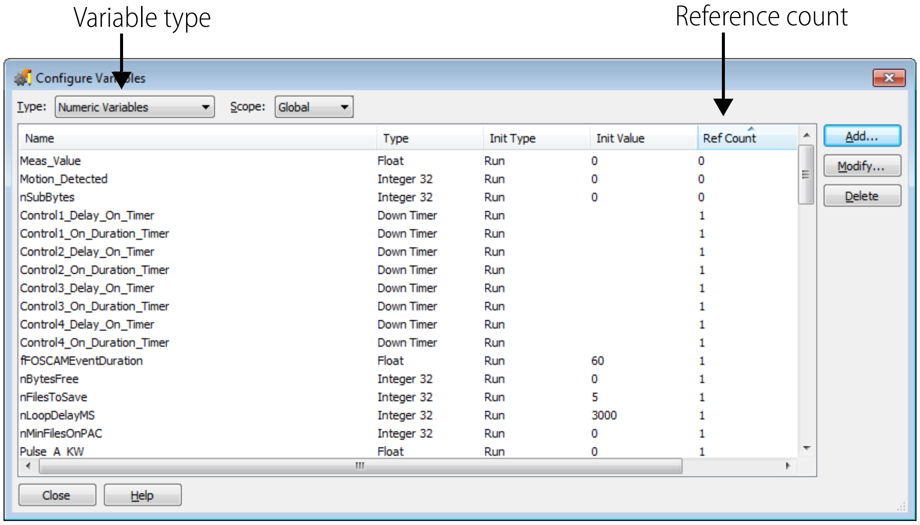
[4] See the *PAC Control User's Guide* Chapter 11, "Using OptoScript." OptoScript saves space (and time) especially for mathematical expressions, complex loops and conditions, and string handling.

change their values later if needed. You can also change them during program execution for debugging.

Many commands return a status result (the "Put Status In" parameter). To simplify troubleshooting, create a separate variable for each instance rather than putting them all in the same variable.

Before you debug, it's a good idea to check reference counts on all variables. It's easy to create a variable early on and then end up not using it. Sometimes tags are added to a strategy and only used in PAC Display, though; these should not be deleted even though their reference count is zero. If you do add a PAC Display-only variable, note that in the variable's Description field so there's no doubt.

To check reference counts, in the Strategy Tree, double-click the Variables folder. Choose the Type of variable from the dropdown list. Click the Ref Count heading to sort on that column. All the variables of that type with reference counts of zero are listed first:



Type: Numeric Variables

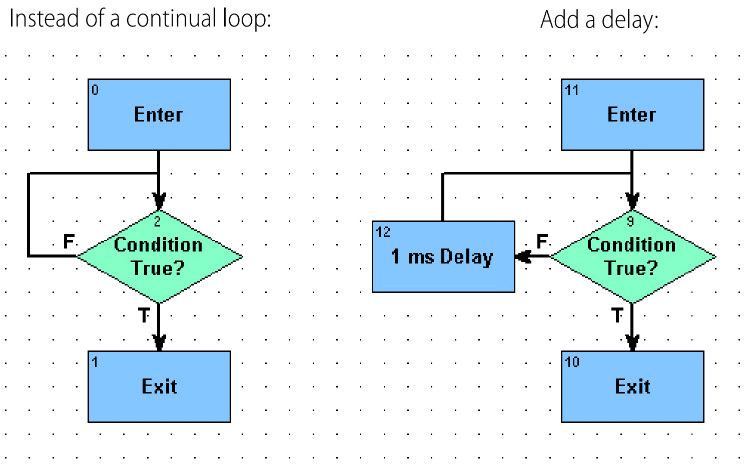
Scope: Global

Name	Type	Init Type	Init Value	Ref Count
Meas_Value	Float	Run	0	0
Motion_Detected	Integer 32	Run	0	0
nSubBytes	Integer 32	Run	0	0
Control1_Delay_On_Timer	Down Timer	Run		1
Control1_On_Duration_Timer	Down Timer	Run		1
Control2_Delay_On_Timer	Down Timer	Run		1
Control2_On_Duration_Timer	Down Timer	Run		1
Control3_Delay_On_Timer	Down Timer	Run		1
Control3_On_Duration_Timer	Down Timer	Run		1
Control4_Delay_On_Timer	Down Timer	Run		1
Control4_On_Duration_Timer	Down Timer	Run		1
fFOSCAMEventDuration	Float	Run	60	1
nBytesFree	Integer 32	Run	0	1
nFilesToSave	Integer 32	Run	5	1
nLoopDelayMS	Integer 32	Run	3000	1
nMinFilesOnPAC	Integer 32	Run	0	1
Pulse_A_KW	Float	Run	0	1

Timers vs. Delays

While waiting for a condition to become true, or in a looping chart, use at least a 1 millisecond delay to give up the time slice[5] and allow other charts to run.

[5] See the technical note, *Optimizing PAC Project System Performance* (form 1776).



Instead of a continual loop:

Add a delay:

In other cases, it is usually best to use timer logic to cause a delay. Delay commands are often hidden within a block of other commands, so it may not be obvious why a chart with a delay takes longer than expected. The timer will make it more obvious.

While looping logic is necessary in some charts, often it's a safer practice to use flow-through logic with timers instead of delays. For more information on using timers and on building efficient strategies, be sure to see the *Optimizing PAC Project System Performance Technical Note* (form 1776).

Documenting your Projects

Documenting in PAC Control

PAC Control strategies are partially self-documenting because they're typically flowchart-based. The *PAC Control User's Guide* (Chapter 4) walks you through a design example of a sprinkler system. As you can see by this example, once you have your process laid out as a flowchart, you've already partly designed and documented your system or subsystem.

But the documentation inherent in your flowcharts doesn't give enough information for other developers—or even you—to come back and maintain the system later.

Here are a few easy ways to document your strategy for easier maintenance.

Names

Naming conventions are often the subject of lively debate among developers.[6] Naming is a matter of style; what's important is to think about the best way for you to name elements

[6] Such as in this OptoForum discussion on Naming Conventions and how they relate to global variables, multi-threaded logic, and use of pointers

in your strategy—charts, variables, I/O, and so on—and then to use them consistently. Pick standards that work for you, write them down, and stick to them.

PAC Control gives you plenty of room—50 characters—to create useful names. So names can include information such as:

- Equipment function (pump, chiller, tank)
- Physical location of the device (bldgA, NE, line12)
- I/O type (ai for analog input, do for digital output)
- Variable data type (int32, float)
- Units of value, if applicable (g for gallons, tempC for temperature in degrees C)
- Scope (for example, include a chart or process name in the variable name)

Remember that your names are sorted alphanumerically in the Strategy Tree, and also in the PAC Display and *groov* tag pickers. Think about how you want to find tags in these lists, and then use this alphanumeric sorting to your advantage.

Also remember that on a small mobile device, it's the first several digits that are the most important for distinguishing one item from another, because you may not be able to see the entire name.

So when you assign names, build them to make searching and distinguishing easy. For example:

- If you want your fan units grouped together, then name them:

  fan1_alarm

  fan1_flow

  fan1_speed

- If you want all your alarms grouped together, then name them:

  alarmFan1

  alarmFan2

  alarmFan3

To keep I/O and variable names compact, you can use upper- and lowercase letters instead of spaces or underscores (for example, fan1Alarm instead of fan1_alarm).

For additional suggestions on naming, see "Naming Conventions" in the *PAC Control User's Guide*. There's no right or wrong in naming conventions. Just use what will work best for your process.

One caution: Avoid using generic names like temp1 that could easily be used in multiple charts at the same time, causing conflicts and confusion.

When naming blocks in a flowchart, some PAC Control programmers include the key functions performed in the block. This additional information lets you remember where the logic is performed, without opening the block. Balance this advantage against the disadvantage of larger blocks in your flowchart and decide which you prefer.

Comments

Comments are text you add to your flowchart or code to explain what the system is doing and how. You might think you'll remember what a flowchart does or why you chose to use one variable rather than another, but when time has passed and you've moved on to the next subsystem or process, you might forget.

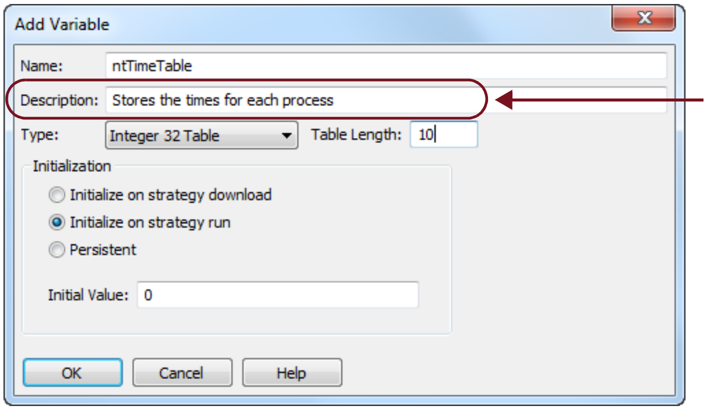
Comments are vital for someone who comes back to your strategy weeks or years later to add points or update a process—and that someone could be you.

Comments and helpful names go together: both contribute to making your code easier to understand and therefore easier to maintain. And if you consistently use specific names that follow your company naming conventions, you won't have to explain them as much through comments.

So make it easier for yourself or the next engineer: use specific names and add comments.

In PAC Control, comments can be added in several ways:

- When creating charts, variables, and I/O points, usethe Description field to add short but clear descriptions.



Add Variable

Name: ntTimeTable

Description: Stores the times for each process

Type: Integer 32 Table

Table Length: 10

Initialization

○ Initialize on strategy download

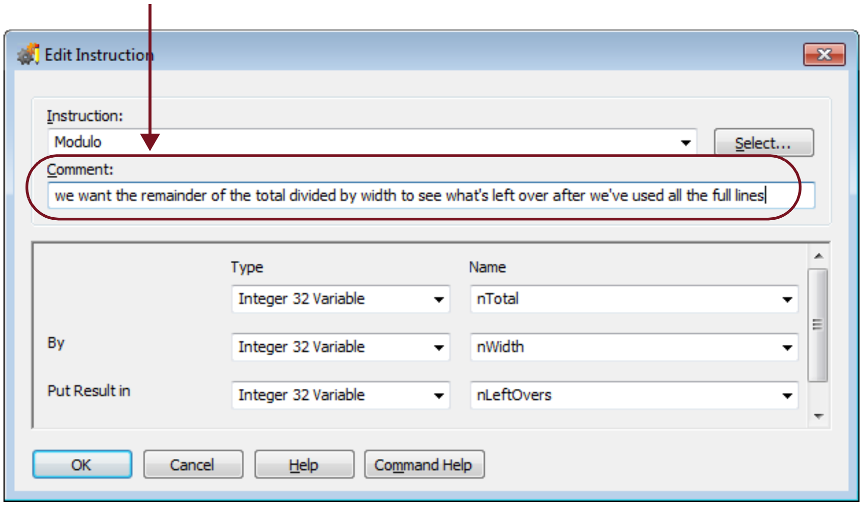
● Initialize on strategy run

○ Persistent

Initial Value: 0

Description for the variable

- Add a brief comment when you add an instruction inside a block:


Comment added to instruction

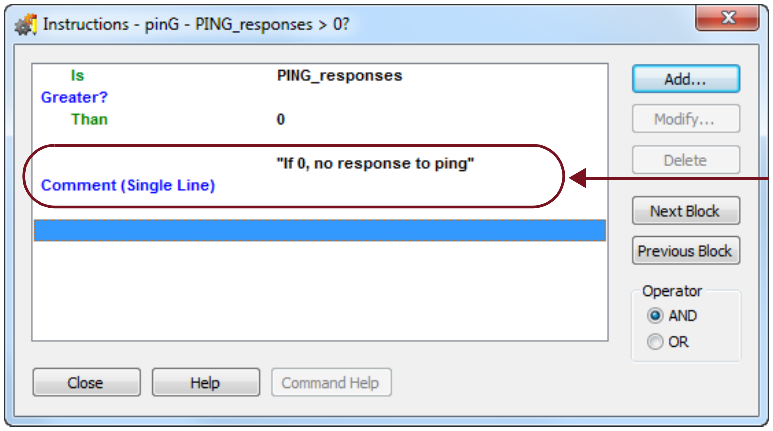
Edit Instruction

Instruction: Modulo

Comment: we want the remainder of the total divided by width to see what's left over after we've used all the full lines

	Type	Name
	Integer 32 Variable	nTotal
By	Integer 32 Variable	nWidth
Put Result in	Integer 32 Variable	nLeftOvers

- Inside an action or condition block, use the Comment (Single Line) command to explain what is happening in the process:


Instructions - pinG - PING_responses > 0?

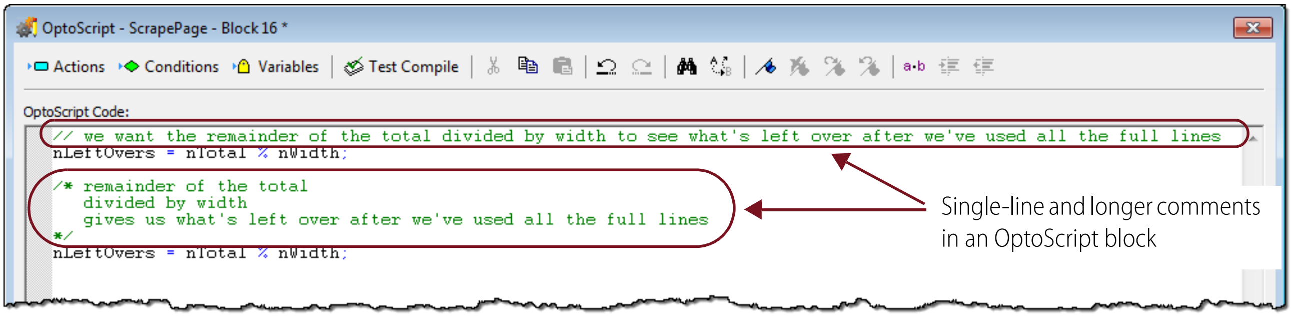
```
    Is                      PING_responses
Greater?
    Than                    0

                            "If 0, no response to ping"
Comment (Single Line)
```

Operator: ● AND ○ OR

Comment command inside block

- Inside an OptoScript block, explain the code by using `//` for a single-line comment or `/*` at the beginning and `*/` at the end of longer comments:

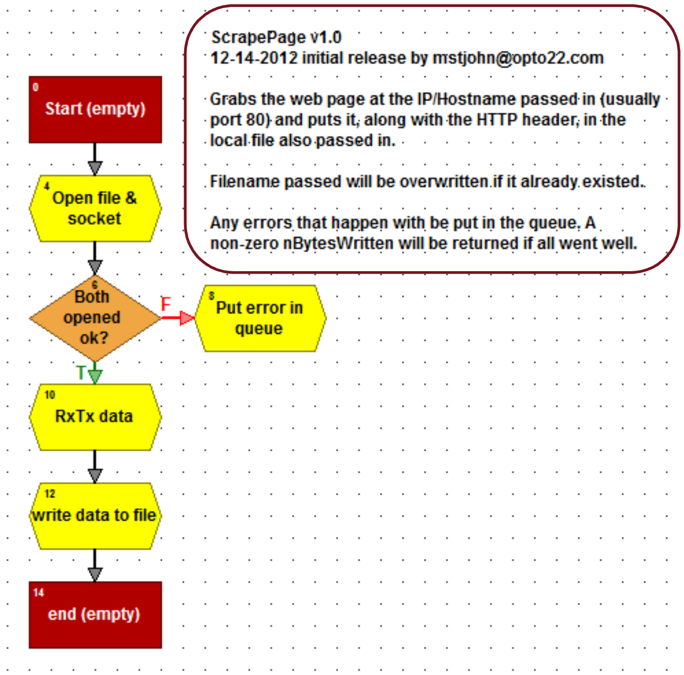


```
// we want the remainder of the total divided by width to see what's left over after we've used all the full lines
nLeftOvers = nTotal % nWidth;

/* remainder of the total
   divided by width
   gives us what's left over after we've used all the full lines
*/
nLeftOvers = nTotal % nWidth;
```

Single-line and longer comments in an OptoScript block

- Directly on a flowchart, use the Text Tool to describe what the chart does or explain the purpose of a block.



ScrapePage v1.0
12-14-2012 initial release by mstjohn@opto22.com

Grabs the web page at the IP/Hostname passed in (usually port 80) and puts it, along with the HTTP header, in the local file also passed in.

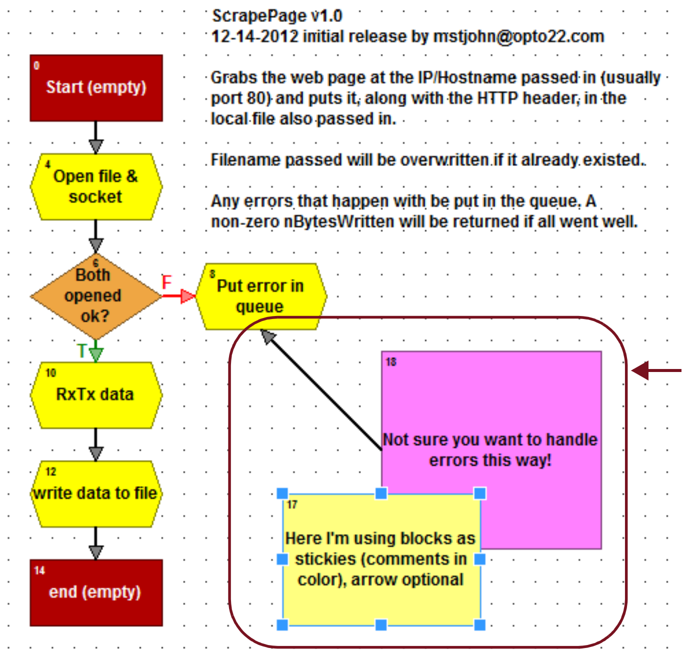
Filename passed will be overwritten if it already existed.

Any errors that happen with be put in the queue. A non-zero nBytesWritten will be returned if all went well.

Note on flowchart made with Text Tool

The Text Tool is also handy for temporary notes to yourself as you're programming, for example to remind yourself of additional logic to add or things to test before downloading the strategy.

- For temporary notes you can even add a colored block and then type your comments as the block name. (These are best as temporary notes because they decrease the effectiveness of the Edit > Find Missing Connections feature.)



ScrapePage v1.0
12-14-2012 initial release by mstjohn@opto22.com

Grabs the web page at the IP/Hostname passed in (usually port 80) and puts it, along with the HTTP header, in the local file also passed in.

Filename passed will be overwritten if it already existed.

Any errors that happen with be put in the queue. A non-zero nBytesWritten will be returned if all went well.

Not sure you want to handle errors this way!

Here I'm using blocks as stickies (comments in color), arrow optional

Temporary note in colored block

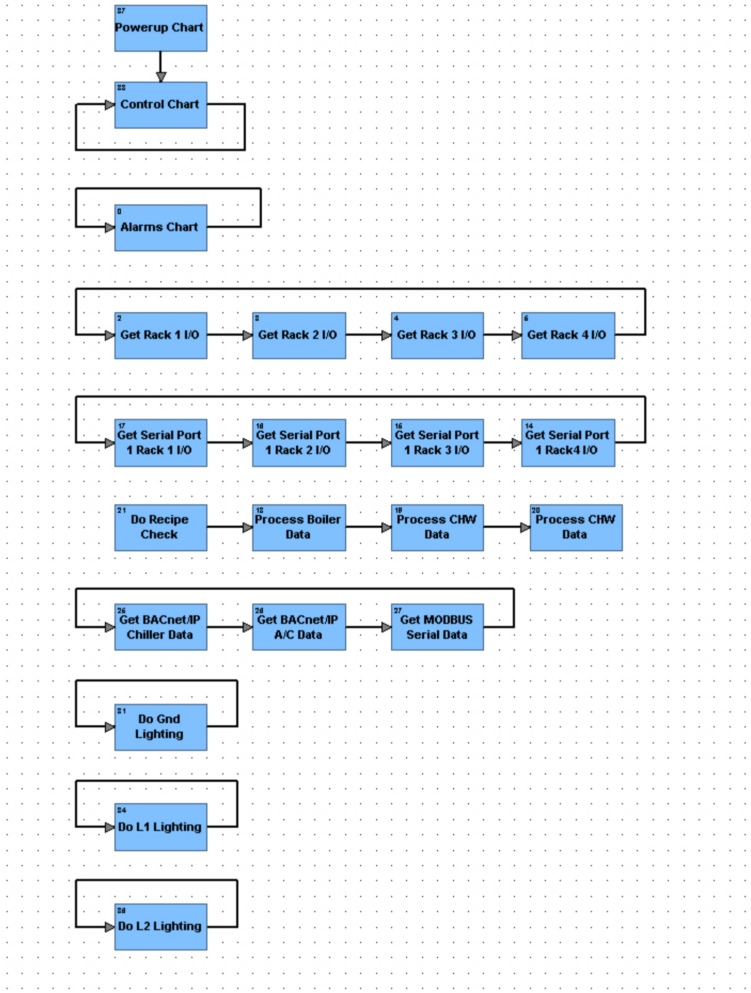
For more ways to make your OptoScript code easier to maintain, watch the video: Three Steps for Building Maintainable Code.

Big-Picture Strategy Documentation

On the Powerup chart, use the Text Tool to write at least a short explanation of the purpose of the strategy and where to find relevant documents or related files. This is also an excellent place to note changes that come later; be sure to include dates and names.

It can also be helpful to have a small document (a text file or PDF) that explains the strategy's big picture as well as key details. Be clear about the purpose of the strategy and what problem it solves. Avoid too much detail, as this document will need to be maintained as well. Remember to update it if plans change along the way.

In large strategies, consider building a reference chart that provides an overall view of the process. This is an easy visual way to show the how the charts link, run, and follow each other. An example is shown below. Again, remember to update it as the process changes.



Documenting in PAC Display and *groov*

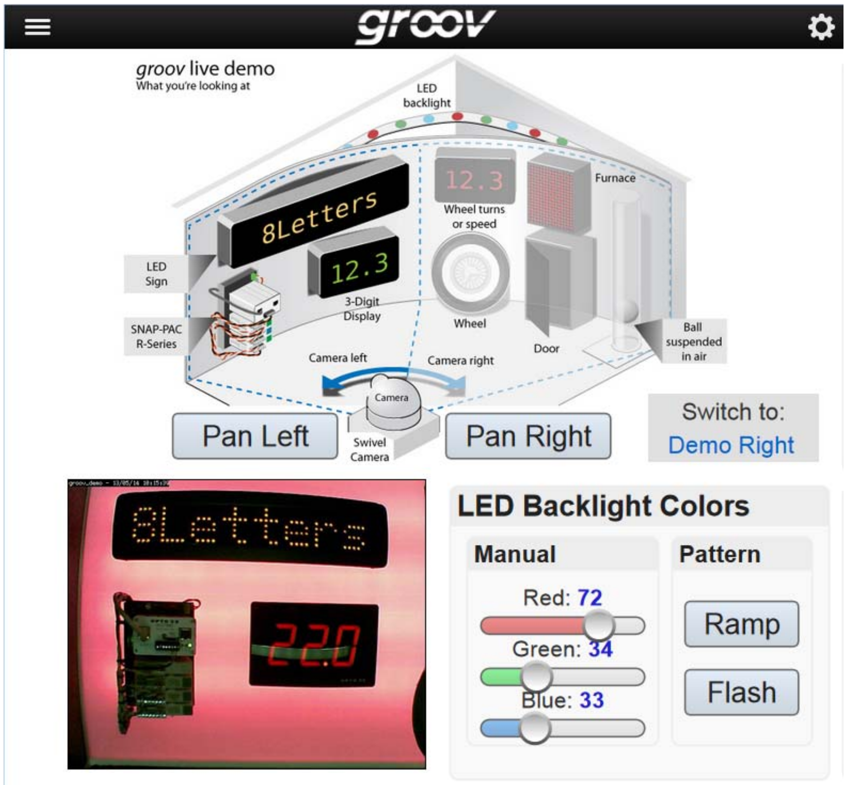
Passwords

IMPORTANT: Make sure you keep track of passwords you set up for PAC Display and *groov*. If you lose or forget your password, Product Support can't help you. Passwords are your responsibility; make sure you keep them safe but accessible to the people who need them.

Documentation for Users

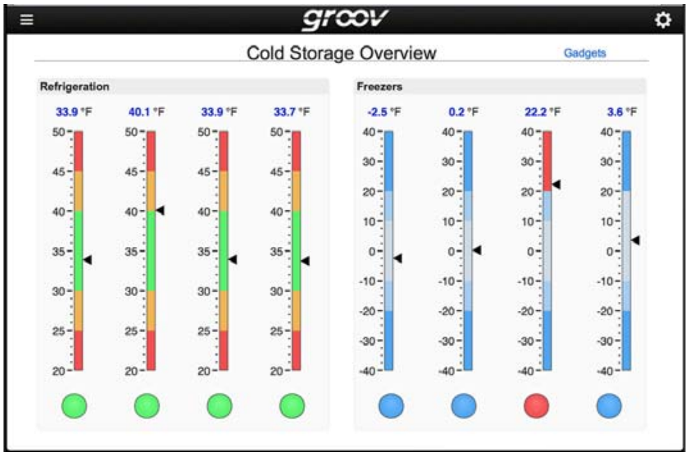
When building your HMI screens in PAC Display or *groov*, make sure to provide the clear descriptions your users will need to use your interface.

Use graphic representations to help orient users:



But don't include unnecessary images that provide no real information, such as spinning pumps or cutaway views of boilers.[7]

Label items on the screen clearly, and group items logically for the operator with a descriptive label for the group.

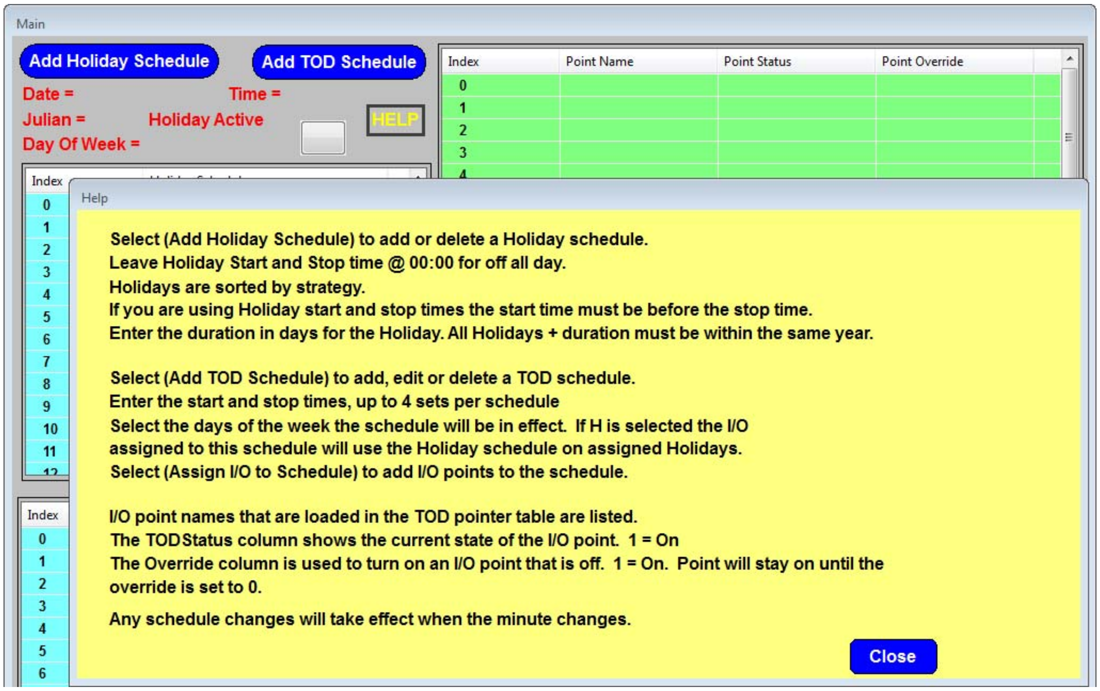


Group items that go together, and label the group.

[7] See the white paper, *Building an HMI that Works: New Best Practices for Operator Interface Design*.

Make sure buttons clearly state the action that happens if they're pushed. And when an operator takes an action, be sure the system provides feedback in the form of a system status statement or a specific acknowledgment that the action has in fact occurred.

A Help window or page in your HMI can provide useful information the user may need without cluttering up the interface itself:



Select (Add Holiday Schedule) to add or delete a Holiday schedule.
Leave Holiday Start and Stop time @ 00:00 for off all day.
Holidays are sorted by strategy.
If you are using Holiday start and stop times the start time must be before the stop time.
Enter the duration in days for the Holiday. All Holidays + duration must be within the same year.

Select (Add TOD Schedule) to add, edit or delete a TOD schedule.
Enter the start and stop times, up to 4 sets per schedule
Select the days of the week the schedule will be in effect. If H is selected the I/O assigned to this schedule will use the Holiday schedule on assigned Holidays.
Select (Assign I/O to Schedule) to add I/O points to the schedule.

I/O point names that are loaded in the TOD pointer table are listed.
The TODStatus column shows the current state of the I/O point. 1 = On
The Override column is used to turn on an I/O point that is off. 1 = On. Point will stay on until the override is set to 0.

Any schedule changes will take effect when the minute changes.

Help windows or pages like these can also be useful for the next developer—or to remind yourself at a later date. In fact it's a good idea to create an unused PAC Display window or a hidden *groov* page for Developer Notes about the HMI: its basic purpose, the thinking behind it, and how it interacts with your PAC Control strategy.

Backing Up Your Projects

Regularly backing up your projects and storing them in a safe, known place are key to project maintenance as well as recovery from possible problems.

For example, suppose you have a problem with a running PAC Control strategy and you need to debug it. If you've made changes to the code since the last time you downloaded the strategy, you can't debug without replacing the strategy that's currently running. But if you've backed up the running strategy, you can use that copy to debug it.

If you don't back up and archive your projects, Opto 22 Product Support can't help you. For example, there is no way to get a running strategy out of a controller, unless you have already archived it and stored it to the controller's flash memory.

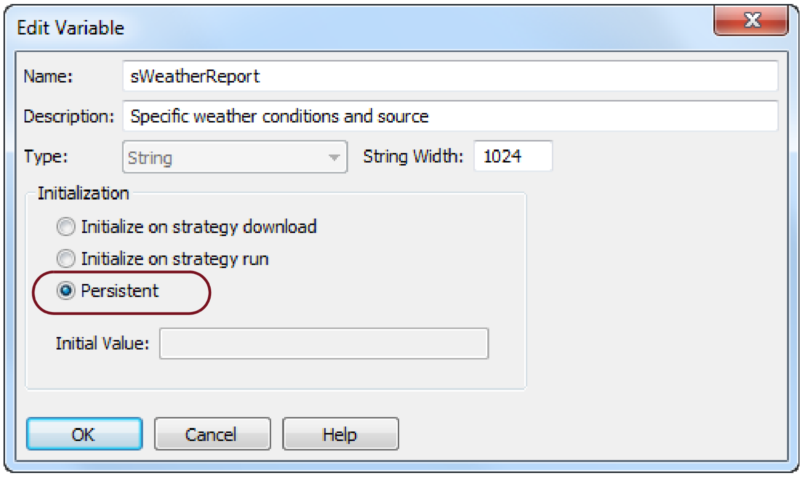
Also remember that hard drives can fail. Keep copies of your archived strategy not just on your local PC but also in at least one other safe location.

PAC Project files and *groov* projects can all be backed up or archived, and we strongly recommend that you do so regularly, following steps in the user's guides.

- In PAC Control's Config mode, use File > Archive Strategy and File > Strategy Options (the Archive tab). Archive both to your computer and to the controller. See the *PAC Control User's Guide*, chapters 5 and 7.
- In PAC Display Configurator: File > Archive Project. See the *PAC Display User's Guide*, chapter 4.
- For OptoDataLink projects, simply copy the <*project name*>.xml file to another location.
- In *groov* Build: File > Back up Project. See the *groov User's Guide*, chapter 3.

To avoid confusion, standardize on where these backups are kept. And use appropriate tools to keep track of backups plus the documentation that describes the project's design and functionality. You can use a code repository such as git or SVN (both designed for software versioning[8]), a directory structure you set up with permissions, or whatever system works for you.

In PAC Control, also remember to back up your important variable values, for example, those you've designated as Persistent:



Edit Variable

Name: sWeatherReport

Description: Specific weather conditions and source

Type: String

String Width: 1024

Initialization

○ Initialize on strategy download

○ Initialize on strategy run

● Persistent

Initial Value:

Although these values will persist through a power cycle, they do NOT persist when you:

- Upgrade firmware
- Download a strategy with a different name
- Clear RAM
- Change ANY persistent variable (even length or type)

[8] Note that because PAC Control strategies are binary files, you won't be able to do a "diff" between files. But you will have the files under version control. For more details, see this OptoForum post on versioning.

Methods for backing up persistent variables before performing one of these actions differ depending on the situation. One popular way is to use PAC Display recipes. For help, see our archived webinar, Using Recipes in PAC Display.

For Help

Remember you're not alone in this process. Training, pre-sales support, and post-sales support are always free.

How-to videos, archived webinars, and product documentation are always available on our website, www.opto22.com. Or for *groov*, see groov.com.

Pre-sales Engineering

Phone:	800-321-6786 951-695-3000
Email:	systemseng@opto22.com

Product Support

Email is the recommended way to reach us, especially because just the act of explaining your issue may help show the way to an answer. As our Product Support Team Lead likes to quote inventor Charles Kettering, "A problem well stated is a problem half solved."

Phone:	800-TEK-OPTO (800-835-6786 toll-free in the U.S. and Canada) 951-695-3080 Monday through Friday, 7 a.m. to 5 p.m. Pacific Time
Fax:	951-695-3017
E-mail:	support@opto22.com
Website:	www.opto22.com

OptoForums

The OptoForums on our website are also a great place for getting help at any time and for brainstorming ideas with Opto 22 in-house experts as well as other Opto 22 customers and integrators with decades of experience. You can post questions on the forums and respond to others' questions as well.

Many thanks to the OptoCommunity on the Forums for their contributions to the best practices in this document.