Form 1710

PAC CONTROL USER'S GUIDE LEGACY EDITION

PAC CONTROL USER'S GUIDE
LEGACY EDITION

Form 1710-170517—May 2017

OPTO 22
Automation made simple.

43044 Business Park Drive • Temecula • CA 92590-3614
Phone: 800-321-OPTO (6786) or 951-695-3000
Fax: 800-832-OPTO (6786) or 951-695-2712
www.opto22.com

Product Support Services
800-TEK-OPTO (835-6786) or 951-695-3080
Fax: 951-695-3017
Email: support@opto22.com
Web: support.opto22.com

PAC Control User's Guide, Legacy Edition
Form 1710-170517—May 2017

Opto 22
Automation Made Simple.

Table of Contents

Chapter 1: Welcome to PAC Control 1

Introduction 1

For Developers: SNAP PAC REST API. 1

About this Guide. 2

Choosing Documentation 3

Document Conventions 3

Help, Documents, and Opto22.com Resources 4

Product Support 4

Installing PAC Control 5

System Requirements 5

Chapter 2: PAC Control Tutorial 7

Introduction 7

Opening the Strategy 8

Saving the Strategy 9

Examining the Strategy 10

The Strategy Tree Window 11

Docking the Strategy Tree 12

Using Quick Find 13

Opening a Chart 13

PAC Control Blocks and Connections 15

Opening a Block 16

Add a New Block and a New Connection 17

Rename a Block 17

Select a Continue Block's Destination 18

Adding a Command 19

Configuring a Control Engine. 23

Compiling the Strategy. 26

Running the Strategy 27

Inspecting Messages 29

Stepping Through the Chart 29

Auto Stepping 31

Compiling and Downloading the Change 34

Using a Watch Window. 36
Closing the Strategy and Exiting 43
What's Next? 43

Chapter 3: What Is PAC Control? 45

Introduction 45
About PAC Control 45
Control System Example 46
General Control Concepts 47
Automation 47
Control Engines 47
Digital and Analog Inputs and Outputs 47
SNAP Serial Communication Modules 48
Key Features 48
PAC Control Terminology. 49
PAC Control Main Window. 54
Status Bar 54
Mode 55
Toolbars 55
Strategy Tree 57
Windows and Dialog Boxes in PAC Control 58
Using Tabs to View Open Windows 59
Docking Windows 60
Zooming in a Chart or Subroutine Window 61
Panning a Chart 62
Redrawing a Chart or Subroutine Window 63
Changing Column Width in a Dialog Box 64
Sorting Data in a Dialog Box 65
Customization Features. 65
Setting Decimal, Binary, or Hex Display Mode 65
Setting Up Applications to Launch from PAC Control 67
Using PAC Control's Command-Line Interface Tool 68
Keyboard and Mouse Shortcuts 76
Online Help 76

Chapter 4: Designing Your Strategy 77

Introduction 77
Steps to Design 77
Solving the Problem 78
Building the Strategy 80
Using and Improving the Strategy 84
Basic Rules 84
Chart Guidelines 85
Naming Conventions 85
Instruction Examples 87
Creating Messages to Display On Screen 87

Using an Interrupt Chart for Event/Reactions (*mistic* only) 88
Error Handling 90
Counting 91
Using a Count Variable for Repetitive Actions 91
Programming Case Statements 92
Using a Timer 93
Using a Flag Lock 94
Pointers and Indexing 96
Optimizing Throughput 98

Chapter 5: Working with Control Engines 99

Introduction 99
Configuring Control Engines. 99
Defining a Control Engine on Your PC 100
Associating the Control Engine with Your Strategy 103
Using Network Segmenting in PAC Control 103
Using Ethernet Link Redundancy in PAC Control. 104
System Architecture for Ethernet Link Redundancy 105
Configuring Ethernet Link Redundancy 108
Using Strategies with Link Redundancy 109
Using Redundant Controllers 109
Changing or Deleting a Control Engine 109
Changing a Control Engine's Definition 109
Changing the Control Engine that Receives the Downloaded Strategy 110
Removing a Control Engine's Association with a Strategy 110
Deleting a Control Engine from Your PC 111
Inspecting Control Engines and the Queue. 111
Inspecting Control Engines in Debug Mode 111
Viewing the Message Queue 114
Inspecting Control Engines from PAC Terminal 116
Archiving Strategies on the Control Engine. 116
Restoring Archived Strategies from the Control Engine. 118
Moving Control Engines from One PC to Another. 119
Downloading Files Using PAC Terminal 120

Chapter 6: Working with I/O 123

Introduction 123
Choosing a Configuration Tool 123
Importing I/O Configuration into PAC Control 125
Copying I/O Configurations 125
About I/O Units 126
SNAP Serial-Based (*mistic*) I/O Units 128
Non-SNAP Serial-Based (*mistic*) I/O Units 129
Adding an I/O Unit 130
Changing the Baud Rate for Serial I/O Units 133
Changing the Baud Rate and Mode for Serial I/O Units 133

Tuning the I/O Unit Timeout Value for Ethernet I/O Units 134
Changing Configured I/O Units 135
Deleting Configured I/O Units 135
Adding I/O Points 136
Adding a Digital I/O Point 136
Adding an Analog I/O Point 139
Configuring a Serial Module 144
Changing I/O Unit and Point Configuration 144
Replace a SNAP Brain or Rack-mounted Controller 145
Moving a Configured I/O Module or Point 146
Copying a Configured I/O Point 147
Changing a Configured I/O Point 150
Deleting a Configured I/O Point 150
Configuring PID Loops 150
PIDs and Strategies 150
Adding a PID Loop (SNAP PAC) 151
Adding a PID Loop (*mistic*) 155
Setting PID Loop Control Options (*mistic* PIDs) 159
Changing a PID Loop (Ethernet or *mistic*) 161
Deleting a PID Loop (Ethernet or *mistic*) 161
Configuring Event/Reactions 162
Adding a MOMO Event or Reaction (*mistic* I/O Units Only) 166
Event/Reaction Configuration Example 168
Using Event/Reaction Groups (*mistic* I/O Units Only) 168
Changing Configured Event/Reactions (*mistic* I/O Units Only) 170
Deleting Event/Reactions (*mistic* I/O Units Only) 171
Inspecting I/O in Debug Mode 171
Inspecting I/O Units and Saving Settings to Flash 172
Inspecting Digital I/O Points 173
Inspecting Analog I/O Points 175
Inspecting Event/Reactions 177
Inspecting and Tuning PID Loops 179
Inspecting a PID (SNAP PAC) 179
Determining System Lag 180
Tuning a PID Loop (Ethernet) 185
Inspecting a PID Loop (*mistic*) 191
Using Watch Windows for Monitoring 193
Creating a Watch Window 193
Opening an Existing Watch Window 194
Working in Watch Windows 195

Chapter 7: Working with Strategies 197

Introduction 197
Creating a New Strategy 197
Opening a Strategy 198
Opening an Existing Strategy 198
Opening an ioControl Strategy 198

Opening a Recently Used Strategy 198
Loading a Strategy or Mode at Startup 198
Opening Strategies in PAC Control 199
Opening an OptoControl Strategy 199

Saving and Closing 199
Saving the Strategy and All Charts 199
Saving the Strategy and Some Charts 199
Saving the Strategy to a New Name 200
Saving Before Debugging 200
Closing a Strategy 200

Saving a Strategy to Flash 200
Saving to Flash Once 200
Saving to Flash on Every Download 201

Archiving Strategies on the Computer 201

Compiling and Downloading 202
Compiling and Downloading in One Step 202
Switching the Strategy 204
Compiling without Downloading 205
Downloading Only 205
Background Downloading 206
Downloading Without Using PAC Control 207
Changing Download Compression 209

Running a Strategy Manually 210

Running a Strategy Automatically (Autorun) 210
Protecting a Running Strategy 211

Stopping a Strategy 211

Debugging 211
Choosing Debug Level 211
Changing Debugger Speed 212
Pausing a Chart or Subroutine 212
Stepping Through a Chart or Subroutine 213
Setting and Removing Breakpoints 215
Managing Multiple Breakpoints 217
Interpreting Elapsed Times 218

Viewing and Printing 219
Viewing Strategy Filename and Path 219
Viewing an Individual Chart or Subroutine 219
Viewing All Charts in a Strategy 220
Printing Chart or Subroutine Graphics 221
Viewing and Printing Strategy or Subroutine Commands 223
Viewing and Printing Strategy or Subroutine Elements 224
Viewing and Printing a Cross Reference 226
Viewing and Printing a Bill of Materials 226
Viewing and Printing I/O Point Mappings 227

Searching and Replacing 228
Searching 229
Replacing 230

Legacy Options . . . 232
Existing Strategies . . . 232
New Strategies . . . 232
Enabling Legacy Options . . . 233
Advanced Options . . . 235

Chapter 8: Working with Flowcharts . . . 237

Introduction . . . 237
Creating a New Chart . . . 237
Working with Chart Elements . . . 238
Chart Components . . . 238
Changing the Appearance of Elements in a Chart Window . . . 239
Drawing Blocks . . . 241
Naming Blocks . . . 242
Connecting Blocks . . . 243
Adding Text . . . 245
Selecting Elements . . . 246
Moving Elements . . . 246
Cutting, Copying, and Pasting Elements . . . 247
Deleting Elements . . . 247
Changing Element Color and Size . . . 248
Opening, Saving, and Closing Charts . . . 248
Opening a Chart . . . 248
Saving a Chart . . . 249
Closing a Chart . . . 249
Copying, Renaming, and Deleting Charts . . . 249
Copying a Chart . . . 249
Renaming a Chart . . . 250
Deleting a Chart . . . 250
Printing Charts . . . 251
Exporting and Importing Charts . . . 251
Exporting a Chart . . . 251
Importing a Chart . . . 252

Chapter 9: Using Variables and Commands . . . 253

Introduction . . . 253
About Variables . . . 253
Types of Data in a Variable . . . 254
Variables in PAC Control . . . 255
Persistent Data . . . 256
Literals . . . 257
Adding Variables . . . 257
Adding Tables . . . 260
Adding Table Variables . . . 260
Setting Initial Values in Variables and Tables During Strategy Download . . . 261
Changing a Configured Variable . . . 265

Deleting a Variable . . . 265
Viewing Variables in Debug Mode . . . 266
  Viewing Numeric, String, and Communication Handle Variables . . . 266
  Viewing Pointer Variables . . . 268
  Viewing Numeric and String Tables . . . 268
  Viewing Pointer Tables . . . 270
Adding Commands . . . 270
Changing a Command . . . 274
Deleting a Command . . . 274
  Permanently Deleting a Command . . . 274
  Commenting Out a Command . . . 275
Cutting or Copying a Command . . . 275
Pasting a Command . . . 276
Configuring a Continue Block. . . . 276
Viewing and Printing Chart Instructions . . . 276

Chapter 10: Programming with Commands . . . 277

Introduction . . . 277
Analog Point Commands. . . . 278
  Offset and Gain Commands . . . 278
  Analog Totalizers . . . 279
  Minimum/Maximum Values . . . 279
  Analog Totalizers . . . 279
  Analog Points and OptoScript Code . . . 279
  Reading or Writing Analog Points . . . 279
Chart Commands . . . 279
  About the Task Queue . . . 280
Communication Commands . . . 280
  Communication Handles . . . 281
  Using TCP and UDP Communication Handles . . . 283
  Using the Control Engine's File System . . . 301
  Moving Files via FTP . . . 308
  Using Serial Communication Handles to Communicate with Serial Devices . . . 312
  Sending an Email . . . 315
  Retrieving a Webpage . . . 323
Control Engine Commands. . . . 327
  Commands Relating to Permanent Storage . . . 327
Digital Point Commands . . . 327
  States, Latches, and Counters . . . 328
  Totalizers . . . 329
  Pulses . . . 330
  IVAL and XVAL . . . 330
  Additional Commands to Use with Standard Digital Points . . . 331
  Standard Digital Points and OptoScript Code . . . 331
  Reading or Writing Digital Points . . . 331
Error Handling Commands . . . 331

IO Enabler Sample Strategies 332
Event/Reaction Commands 332
Understanding Event/Reactions (*mistic* Only) 332
Using an Interrupt Chart with the Generating Interrupt? Command (*mistic* Only) 333
I/O Unit Commands 334
Commands for Ethernet Link Redundancy 335
Table Commands 335
I/O Unit—Event Message Commands 336
I/O Unit—Memory Map Commands 337
I/O Unit—Scratch Pad Commands 338
Logical Commands 340
Understanding Logical Commands 341
Mathematical Commands 342
Using Integers 342
Using Floats 342
Mixing and Converting Integers and Floats 345
Miscellaneous Commands 345
Comment Commands 345
PID—Ethernet Commands 345
What is a PID? 346
PID Loops on I/O Units 347
Algorithm Choices —Ethernet 347
PID—mistic Commands 349
What is a PID? 349
Using PIDs on mistic I/O Units 349
Velocity PID Equation (PID—mistic) 350
Configuration Tips (PID—mistic) 351
Tuning Guidelines (PID—mistic) 352
Tuning Graphs (PID—mistic) 355
Pointer Commands 355
Understanding Pointers 356
Advantages of Using Pointers 356
Referencing Objects with Pointers 356
Simulation Commands 357
String Commands 358
Using Strings 359
String Length and Width 360
Using Numeric Tables as an Alternative to Strings 360
Strings and Multitasking 360
Adding Control Characters to a String 361
Sample String Variable 361
Sample String Table 361
String Data Extraction Examples 362
String Building Example 363
Comparison to Visual Basic and C 364
Convert-to-String Commands 365
ASCII Table 366

Time/Date Commands 366
Timing Commands 367
    Delay Commands 367
    Using Timers 368

Chapter 11: Using OptoScript 369

Introduction 369
About OptoScript 369
When to Use OptoScript. 370
    For Math Expressions 371
    For String Handling 371
    For Complex Loops 373
    For Case Statements 375
    For Conditions 376
    For Combining Expressions, Operators, and Conditions 377
OptoScript Functions and Commands 378
    Standard and OptoScript Commands 378
    Using I/O in OptoScript 378
OptoScript Syntax. 379
    More About Syntax with Commands 380
OptoScript Data Types and Variables 380
    Variable Name Conventions 381
    Using Numeric Literals 381
    Making Assignments to Numeric Variables 381
    Using Strings 381
    Working with Pointers 383
    Working with Tables 385
OptoScript Expressions and Operators 385
    Using Mathematical Expressions 386
    Using Comparison Operators 386
    Using Logical Operators 387
    Using Bitwise Operators 388
    Precedence 388
OptoScript Control Structures 388
    If Statements 389
    Switch or Case Statements 389
    While Loops 390
    Repeat Loops 391
    For Loops 391
Using the OptoScript Editor 392
Troubleshooting “Unable To Find” Errors 396
Troubleshooting Syntax Errors 398
Debugging Strategies with OptoScript 398

Chapter 12: Using Subroutines 399

Introduction 399

About Subroutines 399
Data Types for Subroutines 400
Creating Subroutines 401
Tips for Subroutines 401
Drawing the Flowchart 401
Configuring Subroutine Parameters 403
Adding Commands and Local Variables 406
Compiling and Saving the Subroutine 407
Using Subroutines. 407
Adding a Subroutine to a Strategy 407
Using a Subroutine in Strategy Logic 409
Debugging Subroutines 411
Viewing Subroutines 411
Viewing All Subroutines in a Strategy 412
Printing Subroutines. 412

Appendix A: Troubleshooting 413

How to Begin Troubleshooting. 413
Strategy Problems. 414
If You Cannot Delete an Item 414
If You Have Memory Problems 415
Checking Communication with the Control Engine 416
Resolving Communication Problems 417
Matching PAC Control Configuration to the Real World 417
Resolving TCP/IP Cannot Connect Errors (-412) 417
Other Troubleshooting Tools. 418
Checking Detailed Communication Information Using PAC Message Viewer 418
Checking File Versions for Opto 22 Software 420
Problems with Permissions in Windows 421

Appendix B: Errors and Messages 423

Introduction 423
Types of Errors 423
PAC Control Errors 423
Queue Messages 424
Status Codes 425
Naming Conflicts 425
List of Common Messages 425

Appendix C: PAC Control Files 433

Introduction 433
Files Related to a Strategy. 433
Files Associated with a Subroutine 434
Files in the PAC Control Directory 434

Appendix D: Sample Strategy 435

Introduction 435
Factory Schematic 435
Description of the Process 436
Dough Vessel 436
Chip Hopper 436
Oven 436
Inspection Station 436
Conveyor 436
Emergency Stops 436
Required I/O 437
Analog I/O 437
Digital I/O 437

Appendix E: OptoScript Language Reference 439

Introduction 439
OptoScript Comparison with Standard Programming Languages 439
Function Comparison 440
Variable Comparison 444
Notes to Experienced Programmers 444
Variable Database and Other Surprises 444
PAC Control's Target Audience 444
Language Syntax 445
OptoScript Lexical Reference 446
Token Syntax Legend 446
Literals and Names 446
Keywords (Reserved Words) 447
Operators 447
Comments 448
OptoScript Grammar Syntax Reference 449

Appendix F: Legacy High-Density Digital Module Commands 455

About High-Density Digital Modules 455
Comparing SNAP High-Density and Standard Digital Modules 456
Counting on High-Density Digital Modules 456
Using HDD Module Counters 457
Using HDD Module Commands 457
Individual Point 458
All Points on a Module 458
All HDD Modules on the I/O Unit 460

Index 463

1: Welcome to PAC Control

Introduction

Welcome to PAC Control™, Opto 22's visual control language for Opto 22 SNAP PAC control systems and the input/output (I/O) units that work with them.

PAC Control makes it easy to write control applications with little or no programming experience. If you know how to design a control application and can draw some flowcharts to describe it, you already know the basics of PAC Control. At the same time, PAC Control provides a complete and powerful set of commands, as well as the OptoScript™ programming language, to meet your most demanding industrial control needs.

PAC Control comes in two forms: PAC Control Basic™ and PAC Control Professional™.

- **PAC Control Basic** is included in the purchase of an Opto 22 SNAP PAC controller and is also available as a free download from our website, www.opto22.com. PAC Control Basic includes both flowchart and OptoScript programming, subroutines, a graphical debugger, and about 400 commands.
- **PAC Control Professional** is available for purchase either separately or as part of the complete PAC Project™ Professional software suite. The Professional version of PAC Control adds:
  - Support for controller redundancy and Ethernet link redundancy to controllers and I/O units
  - Support for Legacy hardware, including serial *mistic*® I/O units with a SNAP PAC S-series controller
  - A conversion utility to help you move older OptoControl™ strategies to PAC Control
  - Additional "Pro version only" PAC Control commands

For information on features supported by a specific controller, see the controller documentation.

For Developers: SNAP PAC REST API

If you're a developer who'd like to use PAC Control strategy tags in communications with other devices, the Opto 22 SNAP PAC REST API is a secure and powerful way to do just that. The API is available in SNAP PAC R-series and S-series controllers with PAC firmware R9.5a and higher. To configure https access to your PAC's RESTful server and learn how to call the API, visit developer.opto22.com.

About this Guide

This user's guide not only teaches you how to use PAC Control, but also provides programming instruction and tips. The separate *PAC Control Command Reference* describes in detail all PAC Control programming commands and instructions.

This guide assumes that you are already familiar with Microsoft® Windows® on your personal computer, including how to use a mouse, standard commands, and menu items to open, save, and close files. If you are not familiar with Windows or your PC, refer to the documentation from Microsoft and your computer manufacturer.

This guide covers both PAC Control Basic and PAC Control Professional.

The Basic icon indicates features or functions that apply only to PAC Control Basic.

The Pro icon indicates features or functions that apply only to PAC Control Professional.

Here's what is in this user's guide:

1: Welcome to PAC Control—Information about the guide and how to reach Opto 22 Product Support.

2: PAC Control Tutorial—A tutorial designed to help you use PAC Control as quickly as possible. The chapter leads you through a sample strategy that you can manipulate, download, and run in Debug mode.

3: What Is PAC Control?—An introduction to PAC Control, key terminology, and the main windows and toolbars.

4: Designing Your Strategy—Programming in PAC Control: how to get from your real-world control problem to a working strategy.

5: Working with Control Engines—How to configure and communicate with control engines.

6: Working with I/O—How to configure and communicate with input/output (I/O) units, I/O points, and PID loops.

7: Working with Strategies—Detailed steps for creating, compiling, and running strategies.

8: Working with Flowcharts—Detailed steps for creating and working with the flowcharts that make up your strategy.

9: Using Variables and Commands—Steps for configuring the seven types of variables you can use in programming: communication handle, numeric, string, pointer, numeric table, string table, and pointer table variables. Also shows how to use the commands that control the I/O and variables you've configured.

10: Programming with Commands—Important tips on using PAC Control commands to accomplish what you want in your strategy.

11: Using OptoScript—Details on the optional scripting language available in PAC Control for complex loops, string handling, and mathematical expressions.

12: Using Subroutines—How to use subroutines to streamline your strategy development.

A: Troubleshooting—Tips for resolving communication problems and other difficulties you may encounter.

B: Errors and Messages—Types of errors, where you'll see them, and the causes of common errors.

C: PAC Control Files—A list of all PAC Control files located in the PAC Control directory and in any strategy directory.

D: Sample Strategy—An illustration and description of the sample "Cookies" strategy used in Chapter 1.

E: OptoScript Language Reference—Details about OptoScript code, including comparisons to other languages, lexical reference, and notes to experienced programmers.

F: Legacy High-Density Digital Module Commands—Provides information on the deprecated high-density module commands used with older high-density modules.

Index—Alphabetical list of key words and the pages where they are located.

Choosing Documentation

This user's guide contains the latest information you need to use PAC Control with your SNAP PAC system. However, if you are using Opto 22 "legacy" products designed to work with pre-SNAP PAC systems, see Opto 22 form 1710, *PAC Control User's Guide, Legacy Edition*. The legacy version includes references to pre-SNAP PAC devices and commands, which are not included in this guide.

This legacy version of the PAC Control User's Guide contains the latest information you need to use PAC Control with your SNAP PAC system as well as information for using Opto 22 "legacy" products designed to work with pre-SNAP PAC systems. For a version designed for a SNAP PAC system that does not include older legacy products, see 700.

For information on what we mean by "legacy" and how to migrate from an older system to a SNAP PAC system, see form 1688, the *SNAP PAC System Migration Technical Note*.

To enable legacy I/O units and commands in PAC Control, see "Legacy Options" on page 232.

Document Conventions

The following conventions are used in this document:

- Italic typeface indicates emphasis and is used for book titles. (Example: "See the *PAC Display User's Guide* for details.")
- Names of menus, commands, dialog boxes, fields, and buttons are capitalized as they appear in the product. (Example: "From the File menu, select Print.")
- File names appear either in all capital letters or in mixed case, depending on the file name itself. (Example: "Open the file TEST1.txt.")
- Key press combinations are indicated by plus signs between two or more key names. For example, Shift+F1 is the result of holding down the Shift key, then pressing and releasing the F1 key. Similarly, Ctrl+Alt+Delete is the result of pressing and holding the Ctrl and Alt keys, then pressing and releasing the Delete key.
- "Click" means press and release the left mouse button on the referenced item. "Right-click" means press and release the right mouse button on the item.
- Menu commands are referred to with the Menu > Command convention. For example, "File > Open Project" means to select the Open Project command from the File menu.
- Numbered lists indicate procedures to be followed sequentially. Bulleted lists (such as this one) provide general information.

Help, Documents, and Opto22.com Resources

To help you learn, understand, and use PAC Control systems, we offer a wide variety of options:

- Help is available in PAC Control and in all of the applications in the PAC Project™ Software Suite from Opto 22.
  - To see screen and field-level help, click the Help button on any PAC Control screen. While entering commands (instructions), click the Command Help button to see details about the command.
  - In the PAC Control menu bar, click Help to view the list of help topics.
  - Click Help > Manuals to open the PDF versions of the documents that are installed when you install PAC Control, including:
    - Opto 22 form 1700, the *PAC Control User's Guide*—Shows how to install and use PAC Control.
    - Opto 22 form 1701, the *PAC Control Command Reference*—Contains detailed information about each command available in PAC Control.
    - Opto 22 form 1703, the *PAC Control Commands Quick Reference*—Lists all PAC Control commands plus their OptoScript code equivalents and arguments.
    - Opto 22 form 1704, the *PAC Manager User's Guide*, and other guides provided for specific hardware—Help you install, configure, and use controllers and I/O units.
- Our website at www.opto22.com offers has a broad range of resources—from helpful videos to online blogs, forums, and news to free online self-training. We even offer free hands-on training at our headquarters in Temecula, California.

Product Support

If you have any questions about PAC Control, you can call, fax, or email Opto 22 Product Support.

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
Email:	support@opto22.com
Opto 22 website:	www.opto22.com

NOTE: Email messages and phone calls to Opto 22 Product Support are grouped together and answered in the order received.

When calling for technical support, you can help us help you *faster* if you can provide the following information to the Product Support engineer:

- Software product and version (available by clicking Help > About in the application's menu bar). (When contacting us, please send a screen capture of the Help > About dialog box.)
- Opto 22 hardware part numbers or models that you're working with.
- Firmware version (available in PAC Manager by clicking Tools > Inspect).
- Specific error messages you saw.
- Version of your computer's operating system.

Installing PAC Control

PAC Control is a component of the PAC Project Software Suite, a comprehensive set of software tools for industrial automation, remote monitoring, and data acquisition projects in any line of business.

Installation is easy and quick, and you can download PAC Project directly from the Opto 22 Support > Downloads webpage.

The PAC Project installer includes both the Basic (free) version of the software suite, and the software for PAC Project Professional. Click here for a comparison of PAC Project Basic and PAC Project Professional features.

To install PAC Control:

1. Download PAC Project from the Opto 22 Support > Downloads webpage.
2. Navigate to the folder where you downloaded PAC Project, and then double-click the installation file (PAC_Project_<release number>.exe) to begin installation.

NOTE: You must have an OptoPassword to install the Professional version of PAC Control. For details, see the Opto 22 website, www.opto22.com, Support > Downloads tab, and click the link for OptoPassword (located in the text above the Search field).

If you have trouble installing PAC Control, contact Opto 22 Product Support at 800-835-6786 (toll-free in the U.S. and Canada) or 951-695-3080.

System Requirements

Installation Requirements

Here's what you need to install and run PAC Control. If you are using PAC Display, please note its requirements in form 1702, the *PAC Display User's Guide*.

- A computer with a standard or mainstream core processor and (at least) the minimum memory required for your version of Microsoft Windows. (Low-end CPUs are not recommended.) Additional memory may be required for some configurations.
- One of the following operating systems:
  - Microsoft Windows 10 Professional (32-bit or 64-bit)
  - Windows 8.1 Professional (32-bit or 64-bit)
  - Windows 7 Professional (32-bit or 64-bit)
  - Windows Vista® Business (32-bit only)

  NOTE: PAC Project will not install on Windows XP or older Windows operating systems. Embedded operating systems are not tested or supported.
- Ethernet capability.
- VGA or higher resolution monitor (Super VGA recommended). Minimum size: 800x600 with small fonts.

- Mouse or other pointing device.
- (Optional) Installed Windows printer.
- For available hard disk requirements, please see the PAC Project Release Notes.
- Compatible control engine and I/O unit(s), as shown on page 6.

Important Note on Disk Drives

Opto 22 applications, including PAC Control, perform best when using files from a local hard disk. Network drives may be used, but performance may suffer and depends upon the speed and reliability of the network. While it may be possible to use other drive types, such as key chain USB drives and memory cards, their use is not recommended. They are better suited for transferring files rather than directly accessing them.

Compatible Controllers and I/O Units

The following controller and I/O unit combinations are compatible with PAC Control Basic and PAC Control Professional as shown:

Using this controller	PAC Control Basic supports these I/O units	PAC Control Professional supports these I/O units
SNAP PAC S-series	SNAP-PAC-R1 SNAP-PAC-R1-B SNAP-PAC-R1-W* SNAP-PAC-R2 SNAP-PAC-R2-W* SNAP-PAC-EB1 SNAP-PAC-EB1-W* SNAP-PAC-EB2 SNAP-PAC-EB2-W* SNAP-PAC-SB1 SNAP-PAC-SB2 G4EB2 G4D32EB2 G4D32EB2-UPG SNAP Ultimate I/O SNAP Ethernet I/O SNAP Simple I/O E1 and E2**	SNAP-PAC-R1 SNAP-PAC-R1-B SNAP-PAC-R1-W* SNAP-PAC-R2 SNAP-PAC-R2-W* SNAP-PAC-EB1 SNAP-PAC-EB1-W SNAP-PAC-EB2 SNAP-PAC-EB2-W* SNAP-PAC-SB1 SNAP-PAC-SB2 G4EB2 G4D32EB2 G4D32EB2-UPG SNAP Ultimate I/O SNAP Ethernet I/O SNAP Simple I/O E1 and E2** B3000 serial SNAP-BRS G4D16R, G4D32RS, G4A8R B100 and B200
SNAP PAC R-series	SNAP-PAC-R1 SNAP-PAC-R1-B SNAP-PAC-R2 SNAP-PAC-EB1 SNAP-PAC-EB2 G4EB2 G4D32EB2 G4D32EB2-UPG SNAP Ultimate I/O SNAP Ethernet I/O SNAP Simple I/O E1 and E2**	SNAP-PAC-R1 SNAP-PAC-R1-B SNAP-PAC-R2 SNAP-PAC-EB1 SNAP-PAC-EB2 G4EB2 G4D32EB2 G4D32EB2-UPG SNAP Ultimate I/O SNAP Ethernet I/O SNAP Simple I/O E1 and E2**
SoftPAC	SNAP-PAC-R1 SNAP-PAC-R1-B SNAP-PAC-R1-W* SNAP-PAC-R2 SNAP-PAC-R2-W* SNAP-PAC-EB1 SNAP-PAC-EB1-W* SNAP-PAC-EB2 SNAP-PAC-EB2-W* G4EB2 G4D32EB2 G4D32EB2-UPG SNAP Ultimate I/O SNAP Ethernet I/O SNAP Simple I/O E1 and E2**	SNAP-PAC-R1 SNAP-PAC-R1-B SNAP-PAC-R1-W* SNAP-PAC-R2 SNAP-PAC-R2-W* SNAP-PAC-EB1 SNAP-PAC-EB1-W* SNAP-PAC-EB2 SNAP-PAC-EB2-W* G4EB2 G4D32EB2 G4D32EB2-UPG SNAP Ultimate I/O SNAP Ethernet I/O SNAP Simple I/O E1 and E2** B3000 serial SNAP-BRS G4D16R, G4D32RS, G4A8R B100 and B200
* Wired+Wireless PACs and I/O can be used on a wired Ethernet network or on a wireless LAN. ** E1 and E2 I/O units are supported for the features available under the OptoMMP protocol only, not for all Optomux features. See the *E1 and E2 User's Guide* (form 1563) for details.		

2: PAC Control Tutorial

OPTO 22

Introduction

In this chapter, we'll start with a sample strategy: a control application for a simple cookie factory. You'll learn how to work with strategies, open and manipulate flowcharts, work with variables and I/O points, configure a control engine, compile and download a strategy, run it in Debug mode, make an online change, and more. The tutorial can be used with either PAC Control Basic or PAC Control Professional. (The images in the tutorial show PAC Control Basic.)

The best way to use the tutorial is to sit down at your computer and follow it through. To start, all you need is PAC Control, which allows you to do everything up to the point of downloading your strategy. To do the complete tutorial you will need the controller and I/O included in a SNAP PAC Learning Center. For more information, go to the Opto 22 website, www.opto22.com, and search on the part number SNAP-PACLC.

In this Chapter

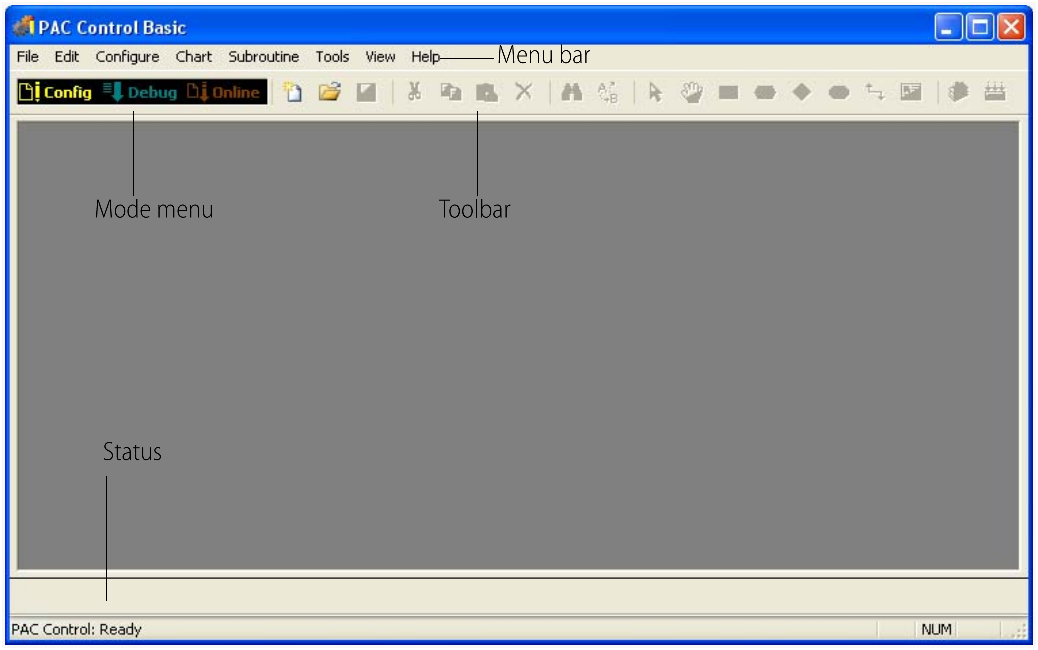
Opening the Strategy 8
Saving the Strategy 9
Examining the Strategy 10
Opening a Chart 13
Opening a Block 16
Adding a Command 19
Configuring a Control Engine 23
Compiling the Strategy 26
Running the Strategy 27
Compiling and Downloading the Change 34
Using a Watch Window 36
Closing the Strategy and Exiting 43
What's Next? 43

Opening the Strategy

A strategy is a complete control program developed in PAC Control. Our sample strategy controls a cookie factory. The sample strategy is described in detail on page 435, but for now, let's just open and explore it.

1. Open PAC Control:
   - In Windows 7 and Windows Vista, press the Windows Start key, and then click Programs > Opto 22 > PAC Project 9.6 > PAC Control.
   - In Windows 10 and Windows 8.1, press the Windows Start key, type **`PAC Control 9.6`** and then press the Enter key.

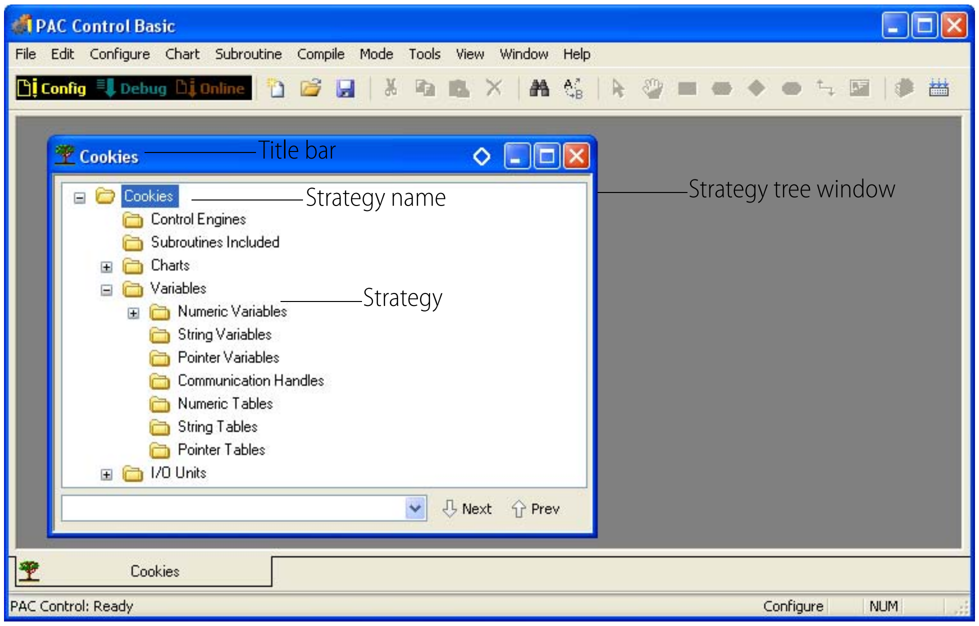
   The PAC Control main window opens.



2. On the PAC Control toolbar, click the Open Strategy button .
3. In the Open Strategy dialog box, browse to C:\Users\Public\Documents\Opto 22\PAC Project 9.6\Control Basic Examples\ioCookies.

   (If you're using PAC Control Pro, substitute "Control Pro Examples" for "Control Basic Examples" in the folder location.)
4. Click Cookies.idb to select it, and then click Open to open the strategy in PAC Control.

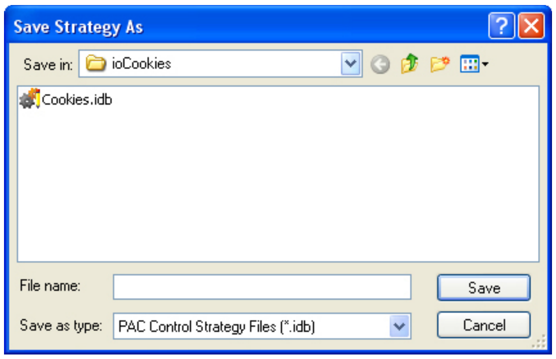
The Cookies strategy opens and the PAC Control window now shows the Cookies strategy in the Strategy tree window. (Yours may look somewhat different.)



Saving the Strategy

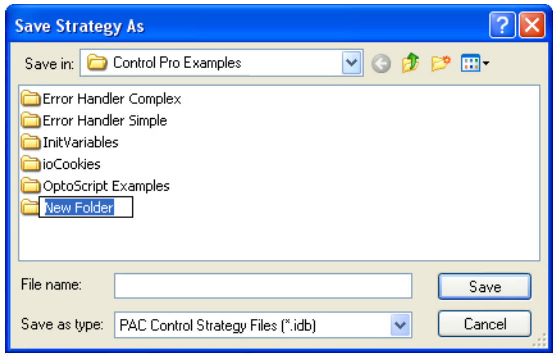
Now let's save the strategy to a new name, so we can change it while leaving the original intact.

1. In the PAC Control menu bar, click File > Save Strategy As.



2. Since each PAC Control strategy must be located in its own directory, you cannot save the strategy to a new name in its current location.
   a. In the Save Strategy As dialog box, click the Up One Level button to move up to the Examples directory.
   b. Click the Create New Folder button

The new folder appears in the list.

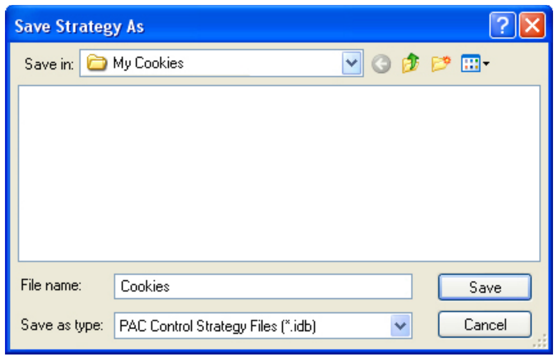


c. Type `My Cookies` to replace New Folder.

d. Double-click the folder to open it.

e. In the File name field, type `Cookies`.

The dialog box now looks like this:



File name: Cookies

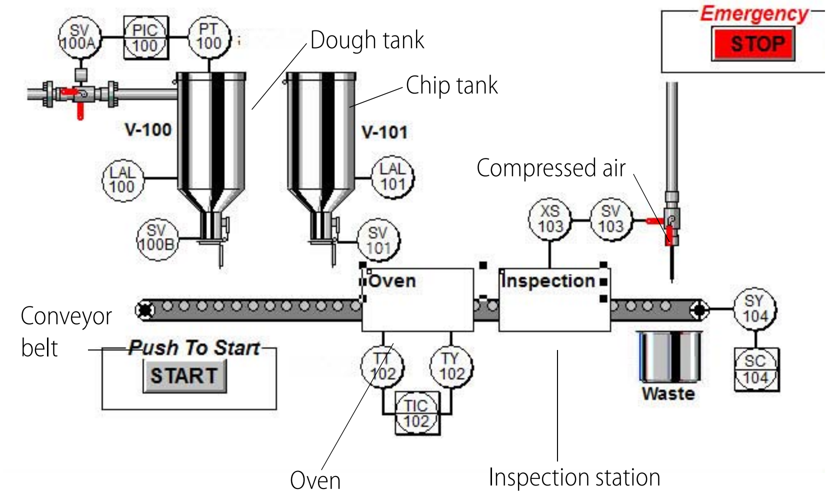
f. Click Save.

Your new Cookies strategy is saved in the My Cookies directory.

Examining the Strategy

Briefly, our cookie factory consists of:

- A tank of pre-mixed cookie dough
- A tank of chocolate chips
- An oven
- A visual inspection station
- A conveyor belt
- Compressed air to blow rejected cookies off the belt



The process starts when a ball of dough drops on the belt. It moves along under the chip tank to receive some chips, and then it moves into the oven to be baked. The next stop is an inspection, where rejected cookies are blown off the belt and good cookies move along to shipping. Should anything go wrong, we also have some built-in alarms to stop the process.

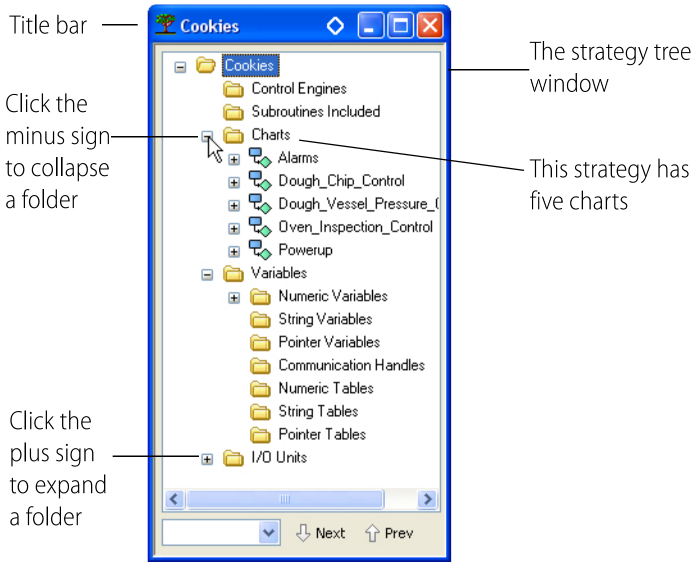
The best way to see all the components of the strategy is to look at the Strategy Tree.

The Strategy Tree Window

To configure and run a strategy, you must leave the strategy tree window open—if you close the strategy tree window, the strategy closes, too.

To expand a folder in the strategy tree, click the plus sign; click the minus sign to collapse the folder.

To move the strategy tree window, click and drag title bar. To minimize, maximize, or dock it, click the buttons in title bar. To resize the window, drag any edge.



Expanding the folders in the strategy tree reveals that the strategy includes five flowcharts (in the Charts folder), eight Numeric Variables, and one Mixed I/O Unit (with both analog and digital points). The Strategy Tree not only shows you all components of the strategy but it also provides shortcuts to many common PAC Control activities, such as opening flowcharts.

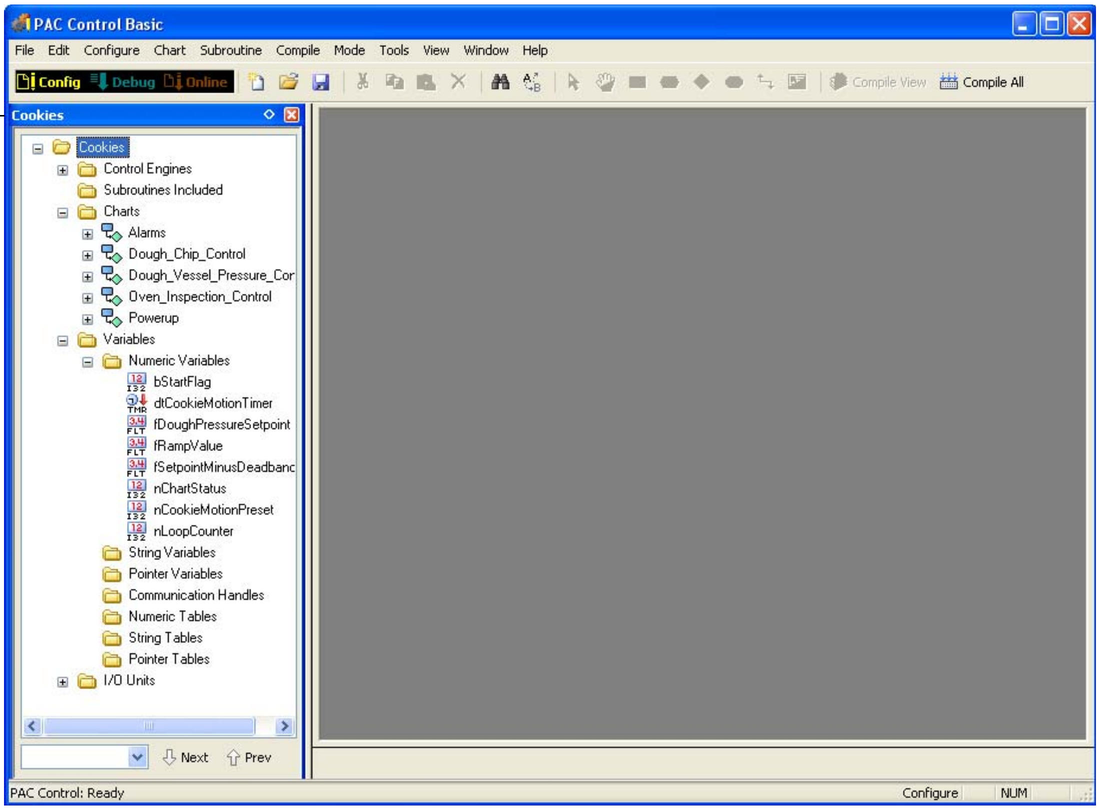
Docking the Strategy Tree

Since the strategy tree is so useful, you'll probably want to keep it visible while you create and debug your strategy. To keep the strategy tree window visible, you can dock it in a separate frame.

1. Click the docking button ◇ in the upper-right corner of the strategy tree window.

   The strategy tree moves into its own frame at the left side of the main window.

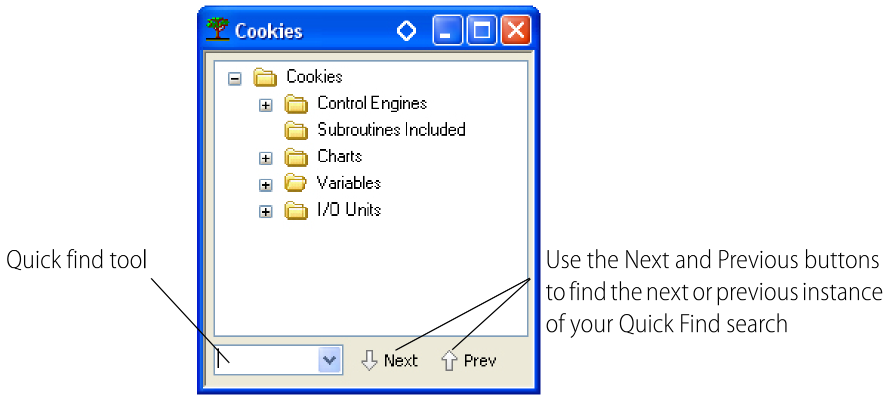
Docked strategy tree



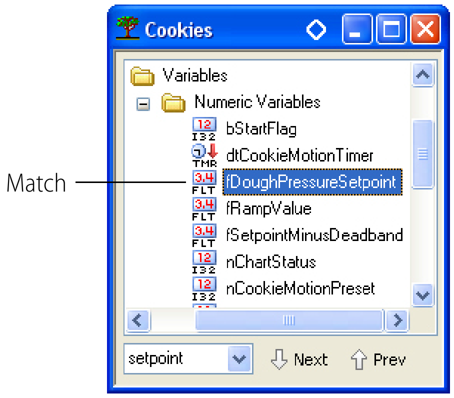
2. To change the width of the strategy tree's frame, move your mouse over the right side of the frame. When the pointer changes to the Windows Splitter icon ⟺, click and then drag the side to make the frame wider or narrower.

Using Quick Find

The quick find tool at the bottom of the strategy tree helps you to find items quickly in the tree. Use the Next and Prev buttons (or F3 and Shift+F3) to find multiple matches. Any partial match will work.


Cookies
Cookies
Control Engines
Subroutines Included
Charts
Variables
I/O Units
Next
Prev
Quick find tool
Use the Next and Previous buttons to find the next or previous instance of your Quick Find search

For example, if you enter the word, “setpoint,” when you click Next, the first match is found.



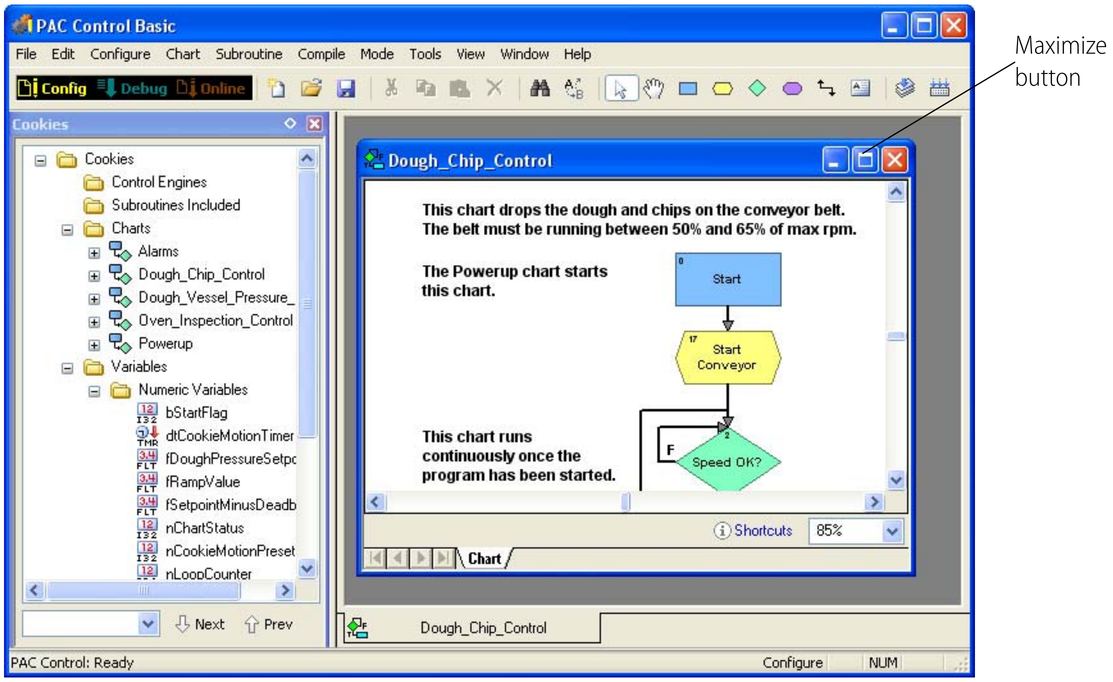
When you click Next again, the next match is found.

Opening a Chart

Every control process can be planned and mapped out using one or several PAC Control flowcharts, or charts. Because flowcharts are easy to follow, PAC Control strategies are easy to understand. For an example, let’s take a look at the Dough_Chip_Control chart.

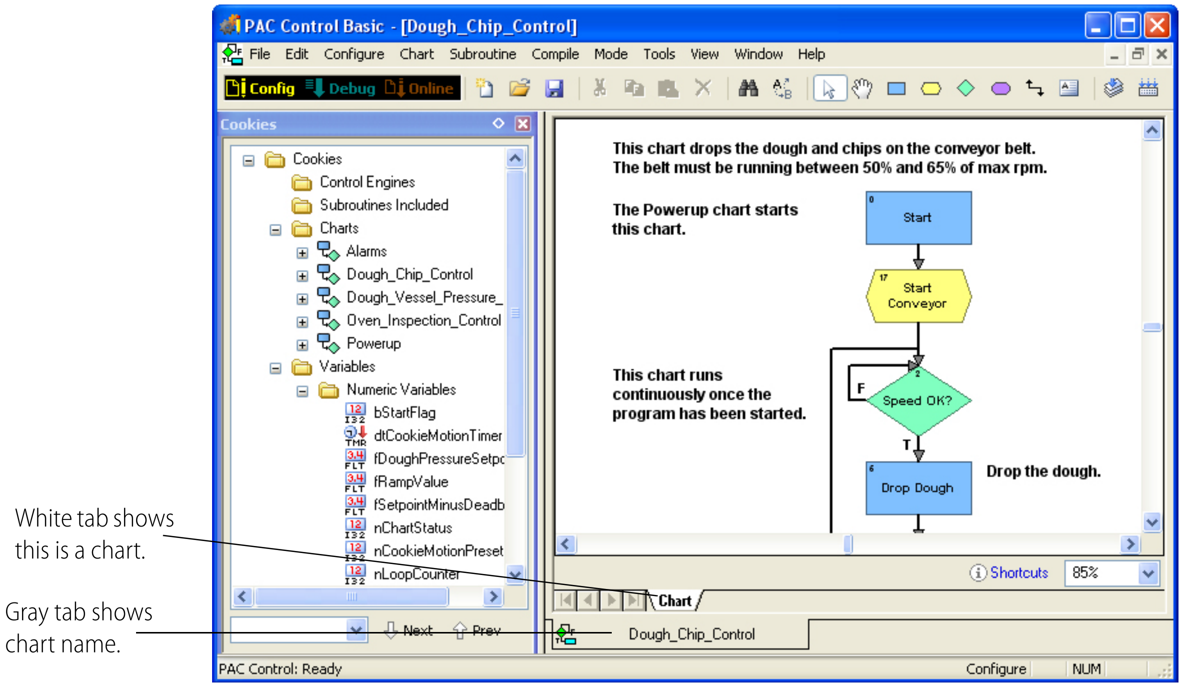
1. With the Charts folder expanded, double-click the Dough_Chip_Control chart name on the Strategy Tree. (You can also open a chart by clicking Chart > Open in the PAC Control menu bar, and then double-clicking the chart name.)

The chart appears somewhere in the large frame of the main window.



2. Click and drag the title bar of the chart window if necessary to see the Maximize button at the upper right. Click the Maximize button to make the chart fully fit the frame.

Note the tabs at the bottom of the main window; the white tab tells you you're viewing a chart, and the gray tab shows the chart's name.



PAC Control Blocks and Connections

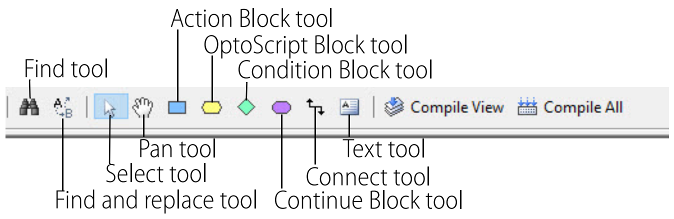
Let's take a closer look at what this chart does. Even without its descriptive comments, it's easy to see that this program segment begins by starting a conveyor belt. If the belt isn't running at the correct speed, the process goes into a loop until the speed is correct. When it is correct, dough is dropped on the belt, and then chips are dropped on the dough. The process then loops back to re-evaluate the conveyor speed, waiting if it's incorrect, and dropping more dough and chips if it's correct.

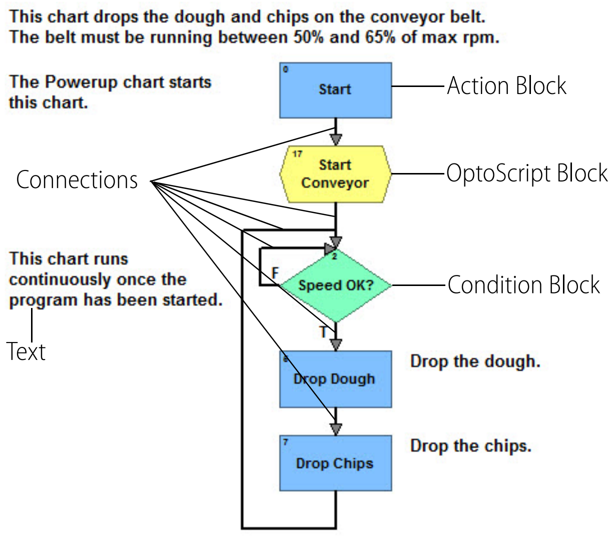
The rectangular-shaped blocks are called **action blocks**. They do things.

The diamond-shaped blocks are **condition blocks**. They decide things.

Charts may also contain other blocks, including oval-shaped **continue blocks**, which route the program logic back to another block in the same chart, and hexagon-shaped **script blocks**, which contain instructions and logic written in PAC Control's built-in OptoScript programming language.

Connections link the blocks together and show how the program logic flows. Action blocks exit through just one connection, since they always go in one direction. Condition blocks exit through two connections, one for a true evaluation and the other for a false evaluation.



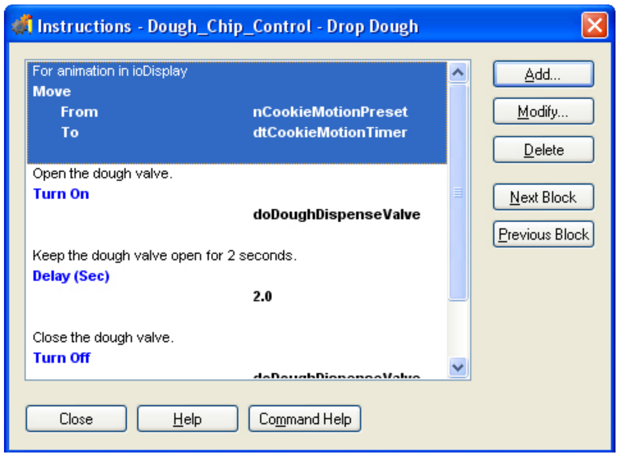

This chart drops the dough and chips on the conveyor belt.
The belt must be running between 50% and 65% of max rpm.
The Powerup chart starts this chart.
Start
Action Block
Start Conveyor
OptoScript Block
Connections
This chart runs continuously once the program has been started.
Speed OK?
F
T
Condition Block
Text
Drop Dough
Drop the dough.
Drop Chips
Drop the chips.

Opening a Block

Let's see what's in a block.

1. Double-click the Drop Dough block.

   The Instructions dialog box appears.

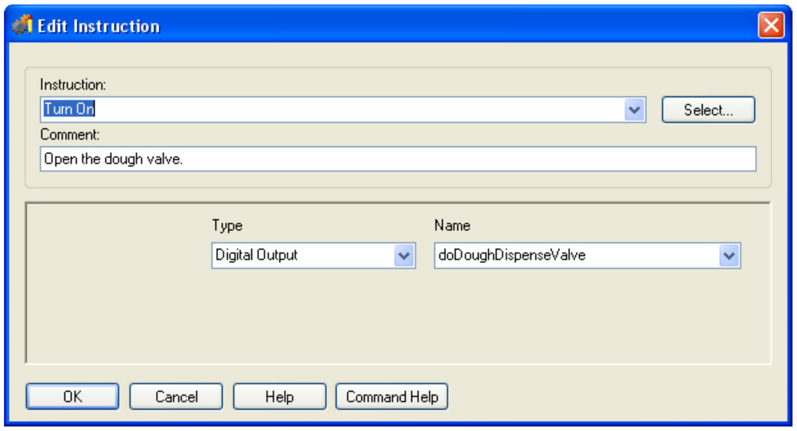


```
For animation in ioDisplay
Move
    From                    nCookieMotionPreset
    To                      dtCookieMotionTimer

Open the dough valve.
Turn On
                            doDoughDispenseValve

Keep the dough valve open for 2 seconds.
Delay (Sec)
                            2.0

Close the dough valve.
Turn Off
```

This block contains four instructions: Move, Turn On, Delay (Sec), and Turn Off. Each one has a description above it.

2. Double-click the Turn On instruction to see more about it. (You could also click it once and click Modify).

   The Edit Instruction dialog box for the Turn On command appears.



Instruction: Turn On

Comment: Open the dough valve.

Type	Name
Digital Output	doDoughDispenseValve

Here we see the details of the command: It turns on doDoughDispenseValve (the digital output point for the dough dispensing valve). In other words, it opens the valve.

3. Close the Edit Instruction dialog box by clicking OK or Cancel.
4. Close the Instructions dialog box by clicking Close or by pressing Esc (the Escape key).

Add a New Block and a New Connection

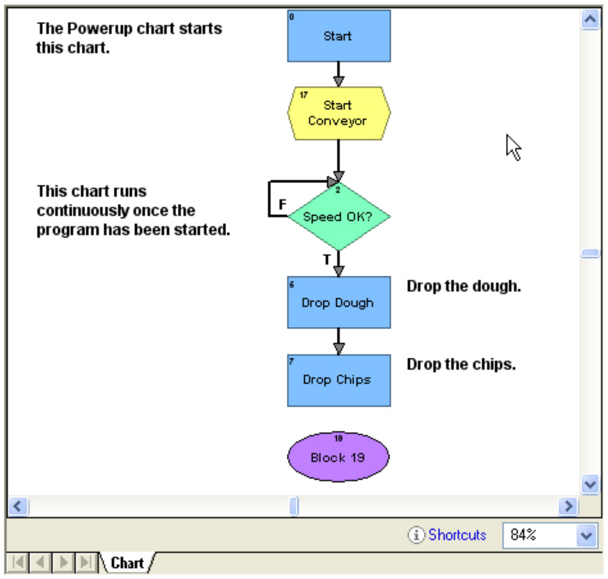
Before we leave the Dough_Chip_Control chart, let's make a cosmetic change. We noted earlier that we didn't have any continue blocks in this chart. Let's add one to replace the long connection that loops from the Drop Chips block up to the Speed OK? block.

1. Delete the existing long connection line by clicking it to select it, and then pressing the Delete key.
2. Now click the Continue Block tool in the toolbar.

   When you bring the mouse back into the chart area, an oval outline appears, representing the new continue block.
3. Position the oval below the Drop Chips block, and then click the mouse.

   A continue block appears with the default name Block (and the block number).
4. If you move the mouse pointer again, a new oval outline follows. Deactivate the Continue Block tool by right-clicking the mouse, or by pressing Esc.

   Your screen should now look something like this:



5. Connect the Drop Chips block to the continue block:
   - **a.** Click the Connect tool .
   - **b.** Click once in the Drop Chips block.

   NOTE: When using connections, always click first in the block that the connection is coming from.
6. If you move your pointer around the screen, you see a connection following the pointer. Click the top of the continue block to complete the connection. Deactivate the Connect tool by right-clicking the mouse button or by pressing Esc.

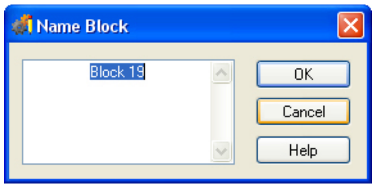
Rename a Block

Let's rename the continue block.

1. Click the continue block to select it.

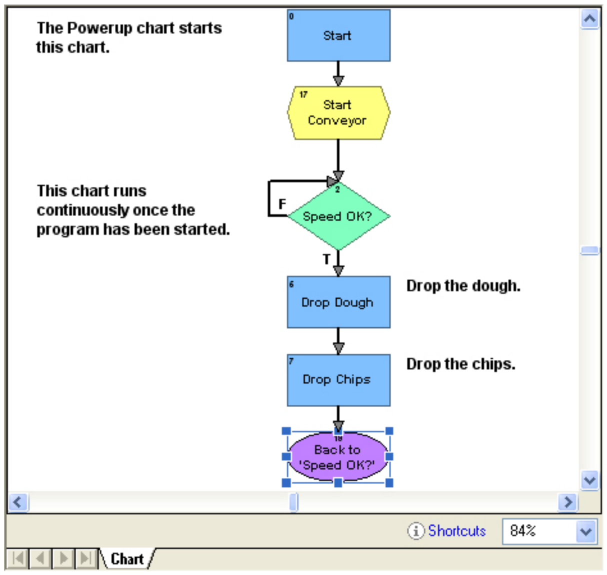
2. Right-click the continue block and select Name from its pop-up menu.

   The Name Block dialog box appears.



3. Type `Back to 'Speed OK?'` over the selected text, and then click OK.

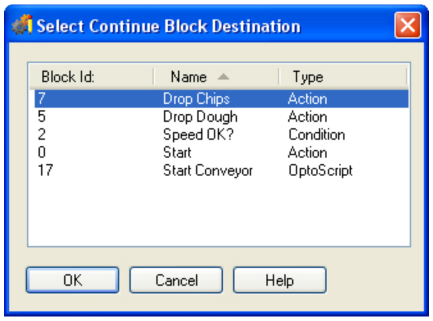
   The chart now looks something like this:



Select a Continue Block's Destination

Now let's give the continue block its instructions.

1. Double-click the continue block to see a list of all the blocks in the chart.



Block Id:	Name	Type
7	Drop Chips	Action
5	Drop Dough	Action
2	Speed OK?	Condition
0	Start	Action
17	Start Conveyor	OptoScript

2. Click the Speed OK? block to select it, and then click OK.

   When the program reaches the continue block, it returns to the Speed OK? block. We haven't changed the way the program works; we've just done the same thing in a different way.

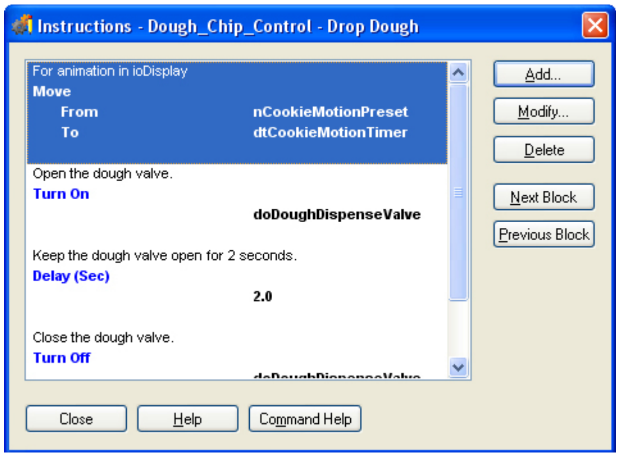
Continue blocks are useful in a complex chart to avoid crossing connection lines.

Adding a Command

Now we're going to add a command (also called an "instruction") to a block in the Dough_Chip_Control chart. This command will keep track of the number of cookies we produce.

NOTE: To enable legacy commands in PAC Control for a specific strategy, see "Legacy Options" on page 232.

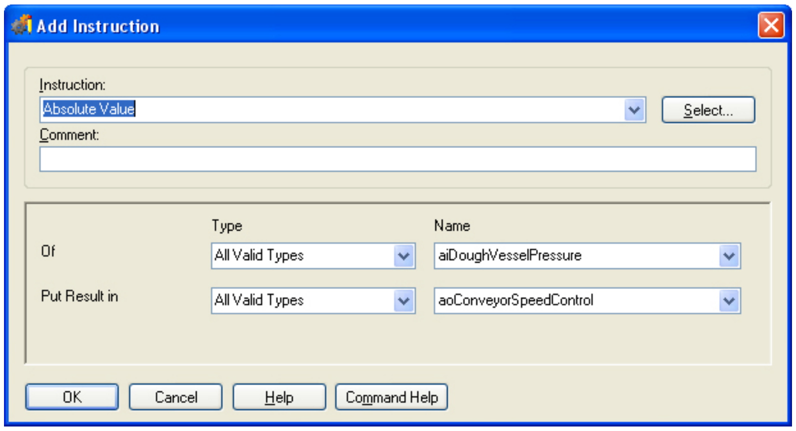
1. Double-click the Drop Dough block.



Instructions - Dough_Chip_Control - Drop Dough

```
For animation in ioDisplay
Move
    From                    nCookieMotionPreset
    To                      dtCookieMotionTimer

Open the dough valve.
Turn On
                            doDoughDispenseValve

Keep the dough valve open for 2 seconds.
Delay (Sec)
                            2.0

Close the dough valve.
Turn Off
```

2. We'll add the new instruction (command) between the Turn On and Delay (Sec) commands.
   a. Click anywhere on Delay (Sec) to highlight this command.
      This highlight indicates the position where the next command is added.
   b. Click Add to open the Add Instruction dialog box.



Add Instruction

Instruction: Absolute Value

Comment:

	Type	Name
Of	All Valid Types	aiDoughVesselPressure
Put Result in	All Valid Types	aoConveyorSpeedControl

Finding Commands in the Instructions Dialog Box. There are several ways to find commands.

- If you know the exact name of the command you want, you can start typing it as soon as you open the Instructions dialog box. (Don't worry about typing over the `Absolute Value` command—It appears in the Instruction field only because it's the first command in the alphabetical list of commands).

As you type, the first command matching the letters you have typed is filled in automatically. As you continue typing, PAC Control displays the next matching command. When you see the command you want, press Enter or click the Add button to add it to the block.

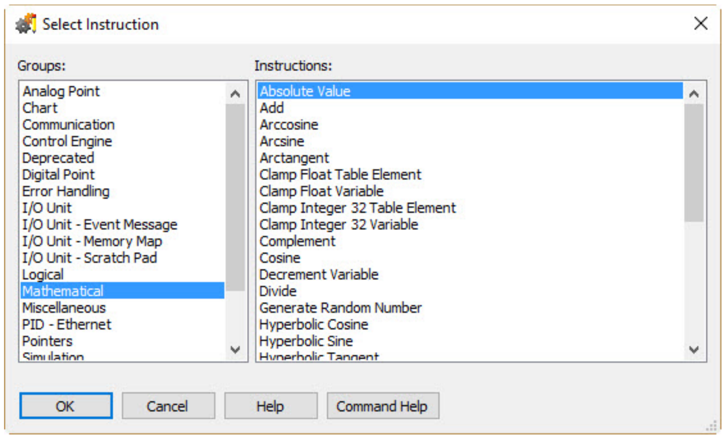
For example, if you want to create a Move command:

As you type:	This command appears:
`m`	`Make Integer 64`
`mo`	`Modulo`
`mov`	`Move`

- One way to add a command is to click the Instruction drop-down list, and then scroll through the list of commands to find the one you want.

A third way to add a command is the one we'll use now.

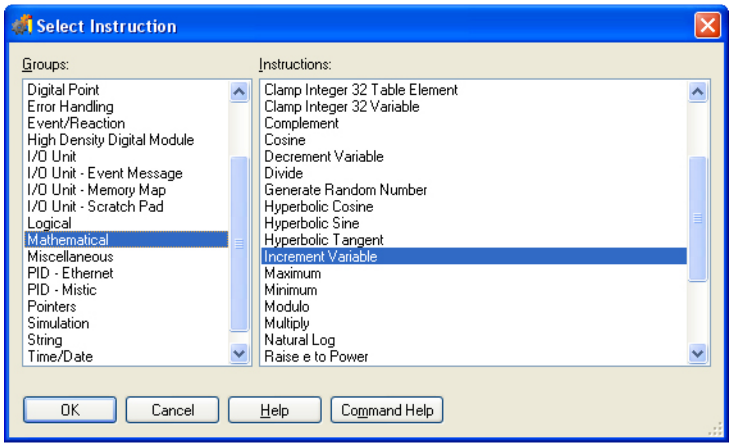
3. In the Add Instruction dialog box, click Select.



Command groups are listed on the left, and commands in the highlighted group are listed on the right.

4. The command we want will increase a counter. Counting sounds like math, so let's try the Mathematical group.

Click Mathematical on the left to highlight it, then scroll through the list on the right until you find the `Increment Variable` command. Click it once.



NOTE: If you need information about a command, click the Command Help button.

5. Click OK, and this command is displayed in Instruction filed of the Add Instruction dialog box.

The cursor is automatically moved to the next field, which is Comment. Comments are optional but very useful; they can help someone else understand the purpose of the instruction.

6. In the Comment field, type `Increment a counter of cookies produced`.

7. Next, click the Type down-down list (the field that currently reads "All Valid Types").

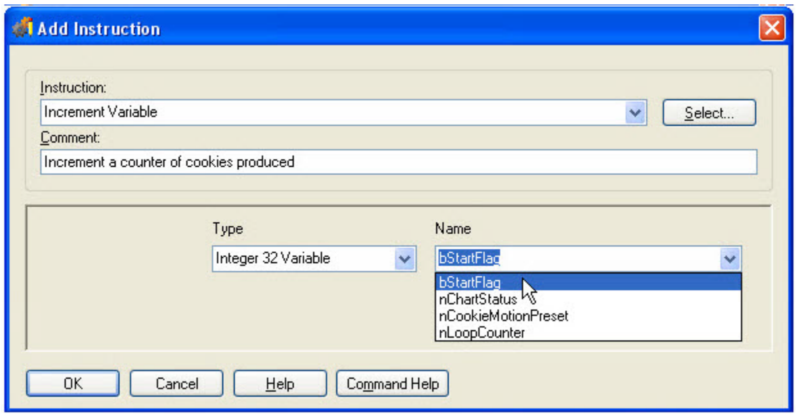
This list shows what those valid types are: a float variable, an integer 32 variable, and an integer 64 variable.

Counters are integers, so select Integer 32 Variable.

8. Now we're going to create the integer 32 variable to increment, which will be called nCookie_Counter.

a. Click the Name down-down list (the field that currently reads "bStartFlag").

The drop-down list shows all variables currently configured as integer 32 variables:


Instruction: Increment Variable

Comment: Increment a counter of cookies produced

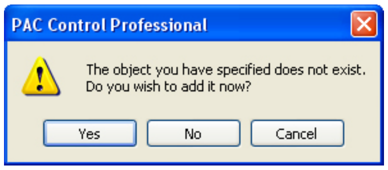
Type: Integer 32 Variable

Name: bStartFlag

nCookie_Counter is not in the list, because we haven't created it. We'll create it now using what we call "on-the-fly configuration."

b. Click `bStartFlag` to select it, and then type the new variable name right over it: `nCookie_Counter`

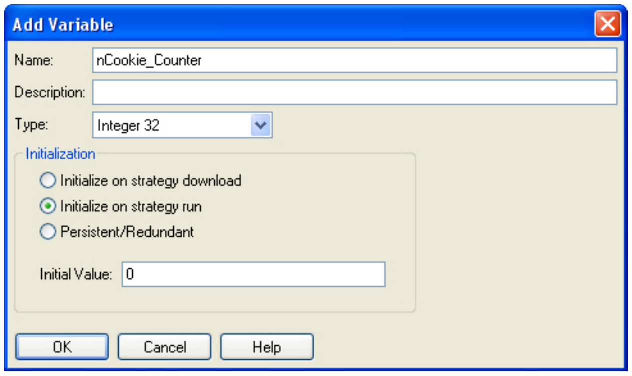
c. Click OK to close the dialog box, and the following message appears.



The object you have specified does not exist.
Do you wish to add it now?

d. Click Yes to create the new nCookie_Counter variable.

Notice that the name and type have already been filled in.

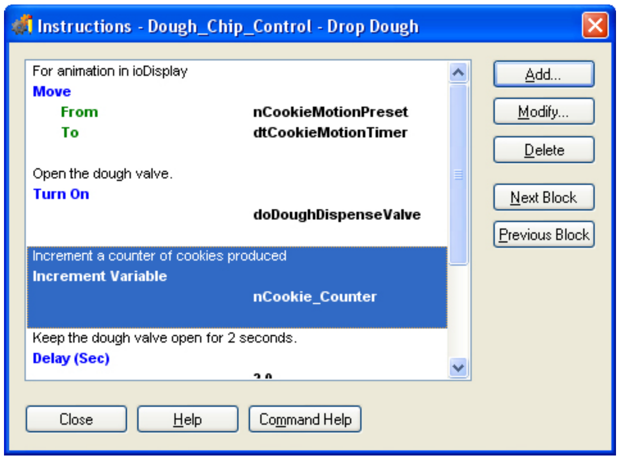


Name: nCookie_Counter
Description:
Type: Integer 32
Initialization
○ Initialize on strategy download
◉ Initialize on strategy run
○ Persistent/Redundant
Initial Value: 0

e. Add a description, if you wish. Leave the initial value at zero, which is the default. Then click OK to save your changes and close the dialog box.

9. In the Add Instruction dialog box, click OK to close it.

The new instruction appears in the Drop Dough Instructions dialog box.



```
For animation in ioDisplay
Move
    From                nCookieMotionPreset
    To                  dtCookieMotionTimer

Open the dough valve.
Turn On
                        doDoughDispenseValve

Increment a counter of cookies produced
Increment Variable
                        nCookie_Counter

Keep the dough valve open for 2 seconds.
Delay (Sec)
```

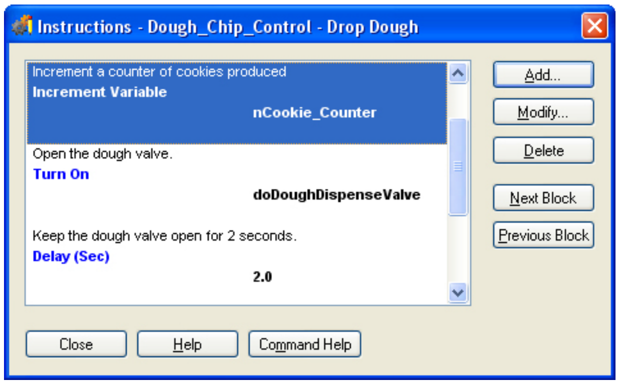
But maybe it makes more sense to start the counter at the beginning of the process, rather than in the middle, after some dough has already been dropped.

1. With the Increment Variable command highlighted, right-click to display the pop-up menu, and then click Cut.

You can also use Ctrl+X to cut. Cutting puts the instruction in the Windows Clipboard.

2. Now click Turn On to highlight it. Right-click to display the pop-up menu, and then click Paste.

You can also use Ctrl+V to paste. The Increment Variable command is pasted above the highlighted instruction, like this:



Increment a counter of cookies produced
Increment Variable
nCookie_Counter

Open the dough valve.
Turn On
doDoughDispenseValve

Keep the dough valve open for 2 seconds.
Delay (Sec)
2.0

3. Click Close to return to the chart.
4. To save the changes you've made, click the Save Strategy button on the toolbar, and then click OK.
5. To see your new variable in the strategy tree:
   a. If the Variables folder is collapsed, click the plus sign to expand it.
   b. If the Numeric Variables subfolder is collapsed, click the plus sign to expand it.

   Your new nCookie_Counter variable should be in the list in alphabetical order.

Congratulations! You've just added a cookie counter variable. Now we're ready to download the strategy to a control engine. But first we have to tell PAC Control that we have a control engine.

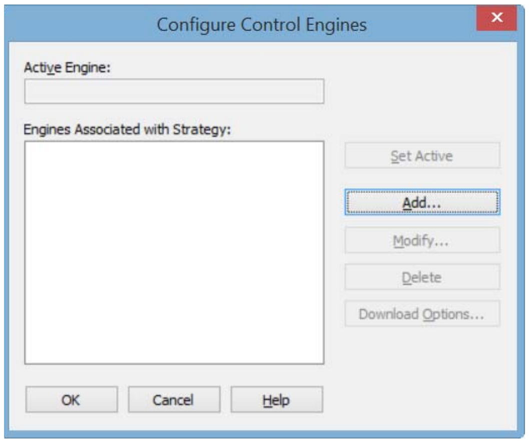
Configuring a Control Engine

Up to this point, we've been able to play around in PAC Control without hardware. Now it's time to configure a control engine (a controller).

1. If you have a PAC Control-compatible controller and I/O unit you can use, make sure they are on the same network as your PC, and make sure you have loaded the most recent firmware.

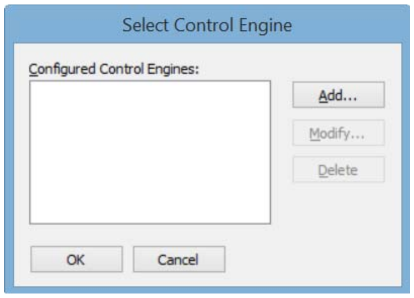
   For a list of compatible controllers and I/O units, see page 6. For instructions to load firmware, see form 1704, the *PAC Manager User's Guide*.
2. Turn the I/O unit off and then back on again.
3. Double-click the Control Engines folder on the Strategy Tree, or select Configure > Control Engines.

   The Configure Control Engines dialog box appears.



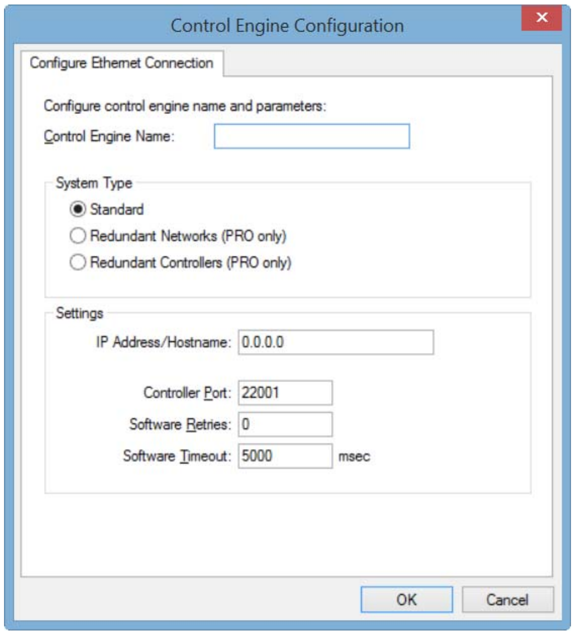
Since we haven't configured a control engine yet, there are no control engines in the list.

4. Click Add.



5. Click Add again to add a control engine.

The Control Engine Configuration dialog box appears.


Control Engine Configuration
Configure Ethernet Connection
Configure control engine name and parameters:
Control Engine Name:
System Type
Settings
OK
Cancel

System Type: [✓ Standard] Redundant Networks (PRO only) Redundant Controllers (PRO only)

IP Address/Hostname:	0.0.0.0
Controller Port:	22001
Software Retries:	0
Software Timeout:	5000 msec

6. Enter Cookie Controller as the control engine name.

   The name can contain letters, numbers, spaces, and most other characters except colons and square brackets. Spaces cannot be used as first or last characters.

7. Under System Type, make sure Standard is selected.

8. Enter the control engine's IP address.

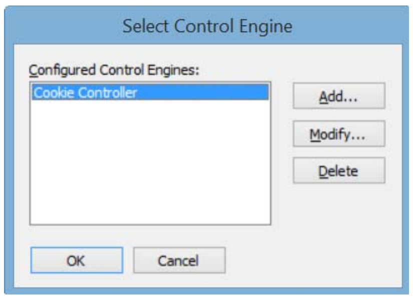
   On hardware such as a SNAP PAC R-series controller, he IP address is usually written on a sticker on the side of the unit. If an IP address has not been assigned to the control engine, see form 1704, the *PAC Manager User's Guide*, for configuration instructions.

   On a SoftPAC controller, if PAC Control and SoftPAC are on the same PC, use the loopback IP address, 127.0.0.1. However, if SoftPAC is on a different PC, use the address for that PC's network interface card (NIC). In this case, the SoftPAC PC's NIC must be configured with a static IP address.

Pro *NOTE: In PAC Control Professional, a second IP address field is available, so you can designate a secondary communication path to the control engine should the primary one fail. For more information, see "Using Ethernet Link Redundancy in PAC Control" on page 104.*

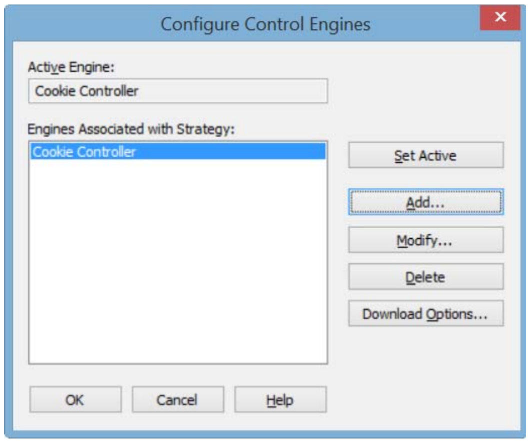
9. Make sure that you have not changed the values in the Port, Retries, and Timeout fields, and then click OK.

   The newly configured control engine appears in the Select Control Engine dialog box.



10. Click the new Cookie Controller control engine to select it, and then click OK.

   The new control engine appears in the Configure Control Engines dialog box.



Since you have only one configured control engine at this point, it is automatically set as the active control engine. If there were more than one control engine, you would have to select it and click Set Active to load it into the Active Engine field.

11. Click OK to close the Configure Control Engines dialog box.

    On the Strategy Tree, the new control engine appears as the first entry in the Control Engines folder.

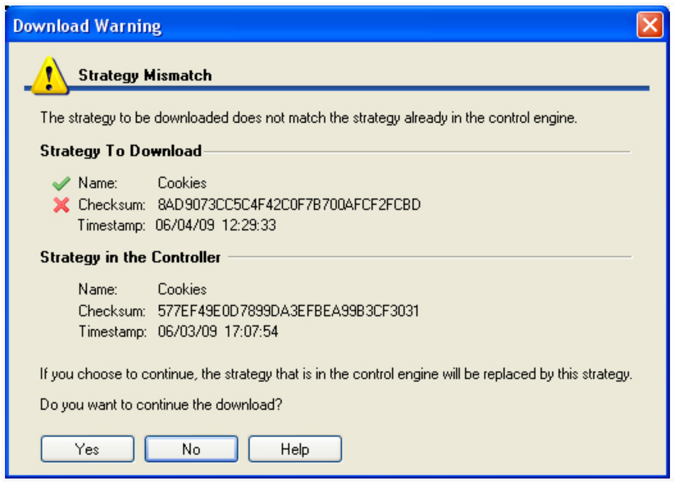
Compiling the Strategy

The simplest way to compile a strategy is to enter Debug mode. The strategy is saved and compiled before changing modes.

NOTE: The remainder of the tutorial is designed to work with the controller and I/O on a SNAP PAC Learning Center. For more information, go to the Opto 22 website, www.Opto22.com and search on the part number, SNAP-PACLC.

1. Click the Debug mode button Debug on the toolbar (or select Debug from the Mode menu).
2. In the Save Strategy dialog box, click Yes to save the strategy.
3. If you see a Powerup Clear Expected message, click OK.

   You may see a Download Warning message like this one:



Strategy Mismatch

The strategy to be downloaded does not match the strategy already in the control engine.

Strategy To Download

✓ Name: Cookies
✗ Checksum: 8AD9073CC5C4F42C0F7B700AFCF2FCBD
Timestamp: 06/04/09 12:29:33

Strategy in the Controller

Name: Cookies
Checksum: 577EF49E0D7899DA3EFBEA99B3CF3031
Timestamp: 06/03/09 17:07:54

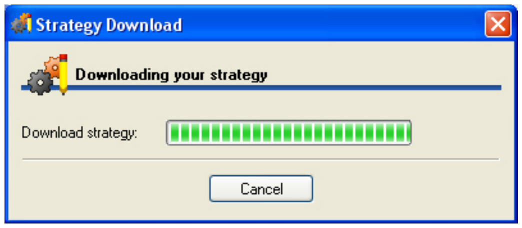
If you choose to continue, the strategy that is in the control engine will be replaced by this strategy.

Do you want to continue the download?

   Or you may see a message that the control engine's memory has been cleared.

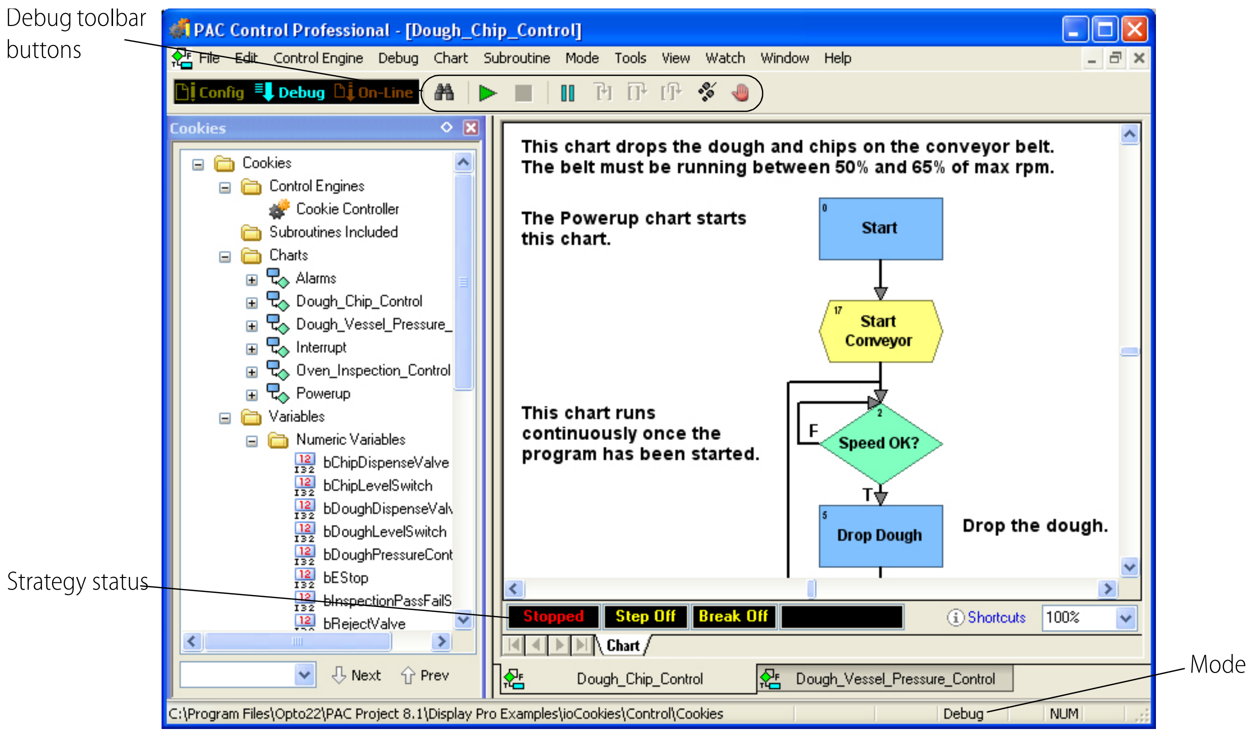
4. Click Yes to proceed.

   Two additional dialog boxes appear. The first dialog displays the progress as the strategy is compiled. The second dialog shows progress as the strategy is downloaded to the control engine.



Assuming the strategy was compiled and downloaded successfully, you are now in Debug mode.

In the PAC Control window, you'll notice that the Debug toolbar is now available. The mode is shown at the bottom of the main window. The open chart window shows that the strategy is stopped, as shown in the following figure:



Running the Strategy

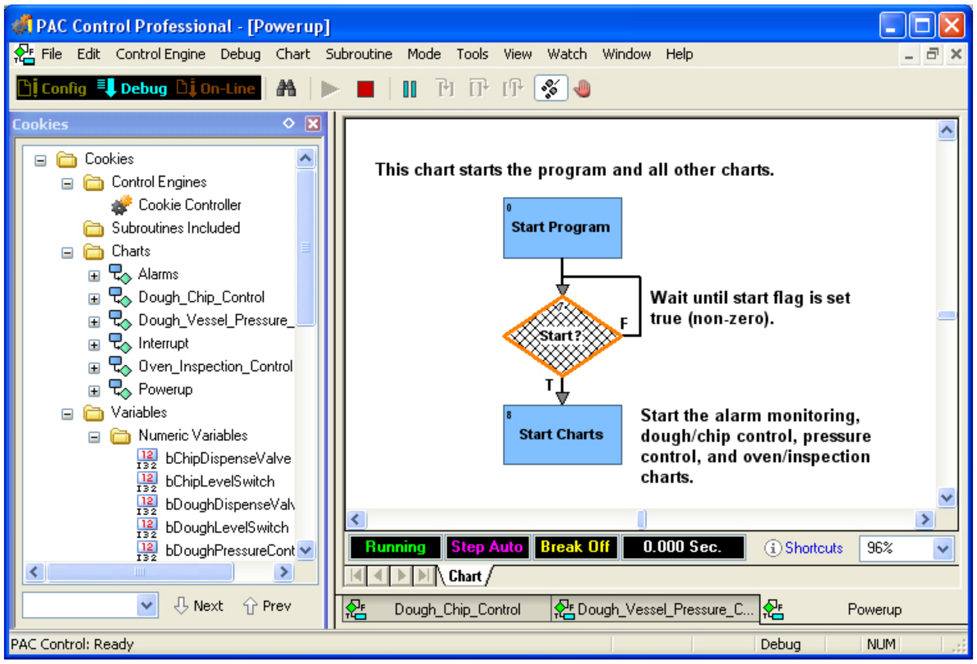
In Debug mode, we're going to run our strategy and examine it. We'll see how the strategy run affects variables, how the command blocks are executed, and so on. The first chart to run in any strategy is the Powerup chart, so we'll look at it first.

1. Double-click the Powerup chart on the Strategy Tree. When it opens, notice that it says Stopped at the bottom left.
2. Click the Run Strategy button ▶.

   A dialog box asks if you are sure you want to run the strategy.
3. Click Start Strategy.

A progress window appears, perhaps only briefly. And at the bottom of the chart window, the word Stopped changes to Running. Let's try pausing the program to see where we are.

4. Click the Pause Chart button ⏸.

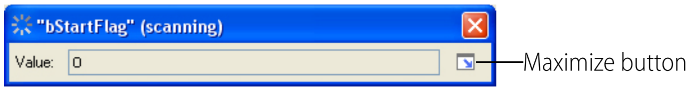


The hatch marks on the Start? block indicate that this command block was about to be executed when we clicked Pause. Apparently the program isn't getting past this block. Notice that the False exit routes right back to the top of the Start? block, while the True exit moves on to Start Charts. We can see that if the start flag had been true (non-zero), the program would have gone right into the Start Charts block. Since we didn't get that far, the start flag must be zero.

And in fact it is. We planned it that way because we wanted someone (for example, a factory operator) to start the process intentionally. We can simulate this ourselves in PAC Control by manually setting the flag to a non-zero value.

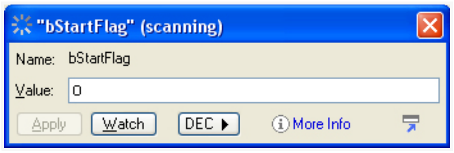
5. Double-click the bStartFlag variable on the Strategy Tree.

   A little box appears.



In this dialog box you can view the variable value, but you cannot change it unless you maximize the dialog box.

6. Click the Maximize button.



This dialog box displays current information on the variable bStartFlag. You can see that the variable is initialized to zero on strategy download.

7. Highlight the value zero in the dialog box and type `1` to replace it.

The field turns purple, indicating a change has been made but not implemented.

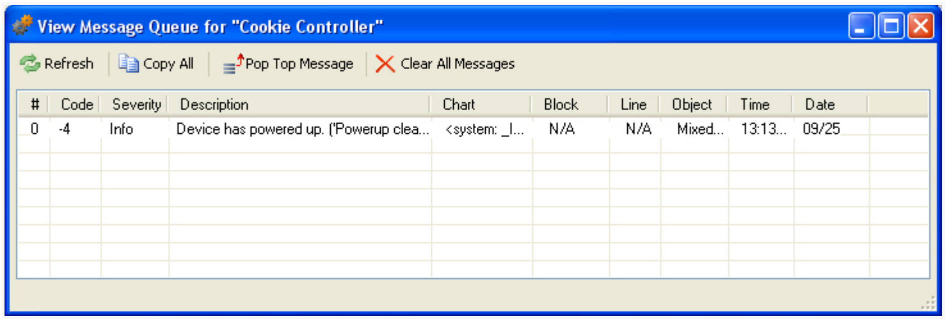
8. Click Apply to implement the change. Click the Minimize button in the bStartFlag View Variable dialog box and move it out of the way. Click the Powerup tab to make the Powerup chart the active window.

Inspecting Messages

1. Look for a light-blue INFO box in the status bar at the bottom of the window.

The status bar in the main PAC Control window tells you when an information, warning, or error message has been placed in the message queue. Messages can help in troubleshooting your strategy. In this example an Information message is in the message queue.

2. Click the INFO box or select Control Engine > Messages to view the View Message Queue dialog box.



#	Code	Severity	Description	Chart	Block	Line	Object	Time	Date
0	-4	Info	Device has powered up. ('Powerup clea...	<system: _I...	N/A	N/A	Mixed...	13:13...	09/25

This message tells you that the I/O unit has powered up.

3. Close the dialog box to return to the Powerup chart.

Stepping Through the Chart

Now let's step through the chart to see what's happening.

1. Click inside the Powerup chart, and then click the Step Over button .

The hatch marks move from Start? to Start Charts. We've just moved (stepped) to the next block.

2. Click the button again.

PAC Control executes the Start Charts block and steps to the next command block. Since there are no more blocks in this chart, we are actually exiting this chart and moving on to new instructions in another chart. The Powerup chart has stopped, and you can see the word Stopped at the bottom left of the chart.

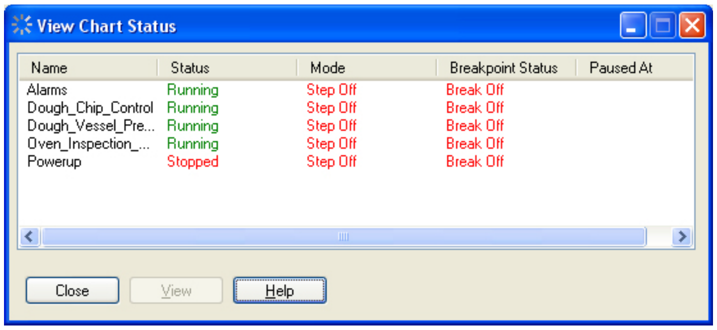
In the View Variable dialog box, the bStartFlag value reverts to zero, because the Start Charts block set the flag back to zero.

3. Close the bStartFlag View Variable dialog box. In the Powerup chart window, click the Pause Chart button to turn stepping off.

Then close the Powerup chart.

4. Double-click the Charts folder on the Strategy Tree.

The View Chart Status dialog box appears.



Name	Status	Mode	Breakpoint Status	Paused At
Alarms	Running	Step Off	Break Off	
Dough_Chip_Control	Running	Step Off	Break Off	
Dough_Vessel_Pre...	Running	Step Off	Break Off	
Oven_Inspection_...	Running	Step Off	Break Off	
Powerup	Stopped	Step Off	Break Off	

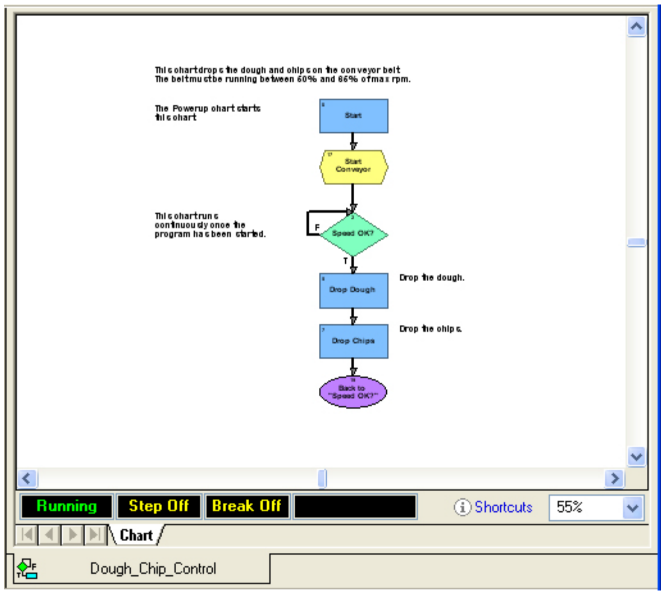
This dialog box shows us all the charts in our strategy. As you can see, four of the charts are running, and one (Powerup) is stopped.

Powerup is stopped because it has already done its job. Let's check the running charts.

5. Close the View Chart Status dialog box and click inside the Dough_Chip_Control chart again.

6. If your mouse has a wheel, hold down the Ctrl key and move the mouse wheel up or down to change the zoom.

If your mouse doesn't have a wheel, there are several other ways to zoom out, too. You can right-click on the chart and select Zoom from the pop-up menu. You can select Zoom Out from the View menu. You can press the + or - keys on the keyboard. After zooming out, the chart looks something like this:


This chart drops the dough and chips on the conveyor belt
The belt must be running between 60% and 66% of max rpm.
The Powerup chart starts this chart
Start
Start Conveyor
This chart runs continuously once the program has been started.
F
Speed OK?
T
Drop Dough
Drop the dough.
Drop Chips
Drop the chips.
Back to "Speed OK?"
Running
Step Off
Break Off
Shortcuts
55%
Chart
Dough_Chip_Control

You can zoom back in if you wish by holding down the Ctrl key and moving the mouse wheel down or one of the other methods.

The chart's status bar indicates that it is running, but we can't see much happening.

7. Click the Pause Chart button.

   One of the command blocks appears with hatch marks.

8. Now click the Step Over button.

   The next command block is hatched.

9. Continue clicking and watch how the program proceeds from Speed OK? to Drop Dough to Drop Chips to Back to 'Speed OK?' and back up to Speed OK?

   A pulsating green border appears around a block when the commands inside the block are being executed. While you are stepping through, and anytime the Pause Chart button is clicked, the chart status indicates Step On.

10. Click the Pause Chart button again to run the strategy at full speed.

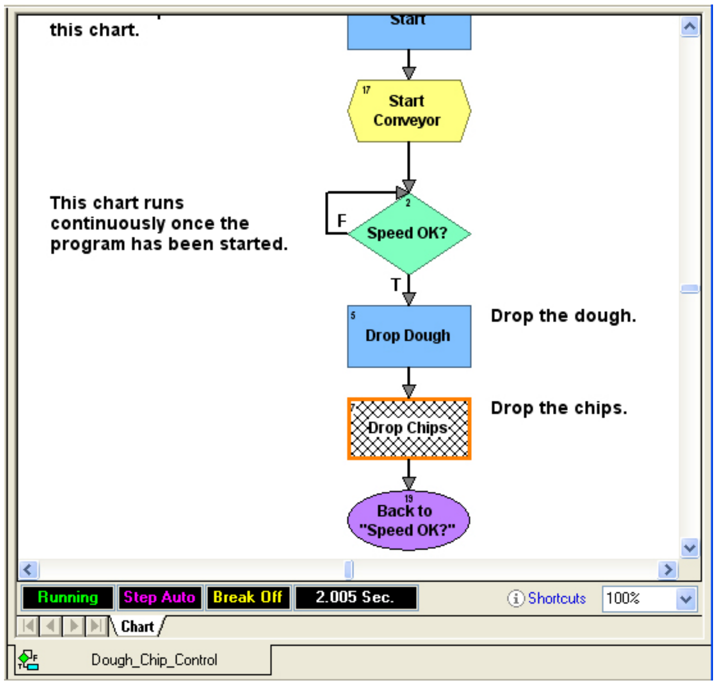
    Step Off now appears in the chart status bar.

Auto Stepping

The final stepping option is auto stepping. You can select this option whether or not the chart is currently paused.

1. Click the Auto Step Chart button and watch the program move from block to block.

   At one time or another, your chart looks like this:



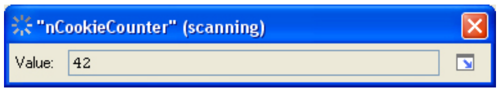
   Notice that in the PAC Control toolbar, the Run and Auto Step Chart buttons are depressed. The chart status bar shows us the chart is running and in Step Auto mode. The time indicator to the right of Break Off shows the time it took for the most recent block to execute.

2. Click the Auto Step Chart button again to end auto stepping.

Step Off appears, indicating the program is again running without interruption.

Now let's see how many cookies we've produced to this point.

3. On the Strategy Tree, double-click the numeric variable nCookie_Counter.


"nCookieCounter" (scanning)

Value: 42

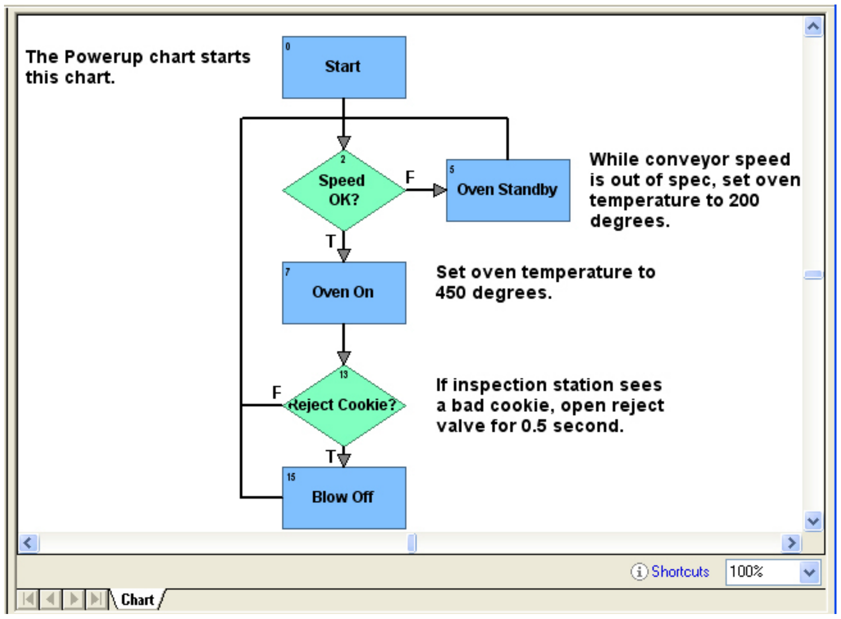
The Value field should increase every time a cookie is produced, adding to the total number of cookies produced since the strategy run began. The nCookie_Counter above shows this figure as 42. (Yours may be different.)

But nCookie_Counter tells us the total number of cookies put on the conveyor belt, without considering that some of them may be rejected by the inspection station. We need to subtract the number of bad cookies so that nCookie_Counter keeps track of the number of cookies sent out the door, not just sent to the oven.

4. Close the nCookie_Counter View Variable window and click the Configure mode button on the toolbar.

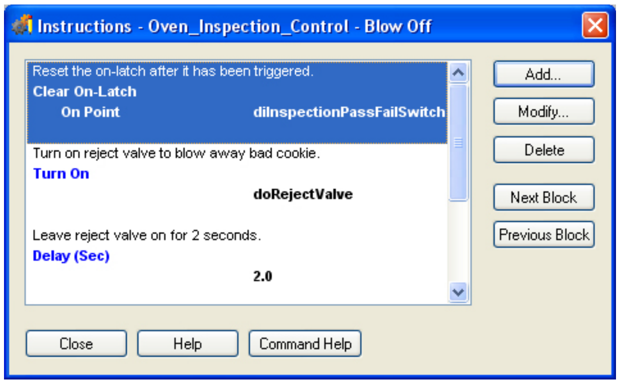
5. Double-click the Oven_Inspection_Control chart on the Strategy Tree to open it.

The chart window looks something like this:


The Powerup chart starts this chart.
0
Start
2
Speed OK?
F
5
Oven Standby
While conveyor speed is out of spec, set oven temperature to 200 degrees.
T
7
Oven On
Set oven temperature to 450 degrees.
F
13
Reject Cookie?
If inspection station sees a bad cookie, open reject valve for 0.5 second.
T
15
Blow Off
Shortcuts
100%
Chart

Near the bottom of the chart, the Reject Cookie? block determines whether a bad cookie has been found. If one has, the strategy moves to the next block, Blow Off, which is where the bad cookie gets blown off the conveyor. When that happens, we want to decrement the nCookie_Counter variable.

6. Double-click the Blow Off block to open its instructions window:



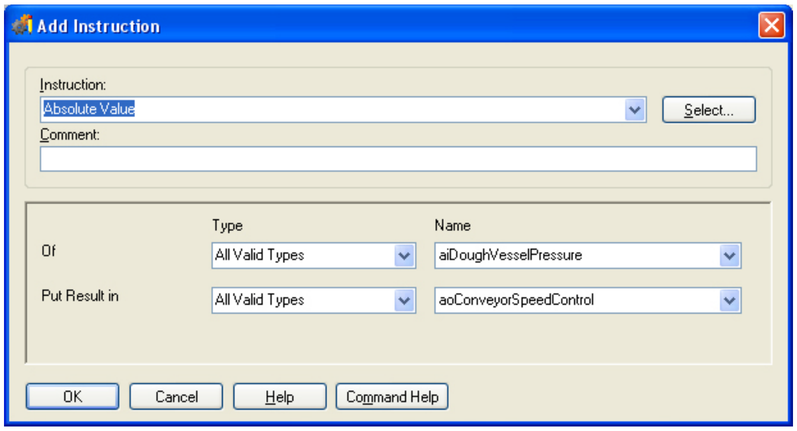
```
Reset the on-latch after it has been triggered.
Clear On-Latch
    On Point                    diInspectionPassFailSwitch

Turn on reject valve to blow away bad cookie.
Turn On
                                doRejectValve

Leave reject valve on for 2 seconds.
Delay (Sec)
                                2.0
```

This command block is executed only when a bad cookie has been found. This block first resets the on-latch triggered by the bad cookie, so that the next cookie won't be marked bad, too. The block then turns on the reject valve. The valve stays on for two seconds before being shut off. Let's decrement the counter after the cookie is gone and the valve is shut.

7. Scroll down and click on the open spot, below the other instructions.

   The highlighted line marks the position of the next command to be added.

8. Click Add.



Instruction: Absolute Value
Comment:

	Type	Name
Of	All Valid Types	aiDoughVesselPressure
Put Result in	All Valid Types	aoConveyorSpeedControl

You can use PAC Control without a mouse, and to demonstrate how, we'll use only the keyboard to enter data in this dialog box.

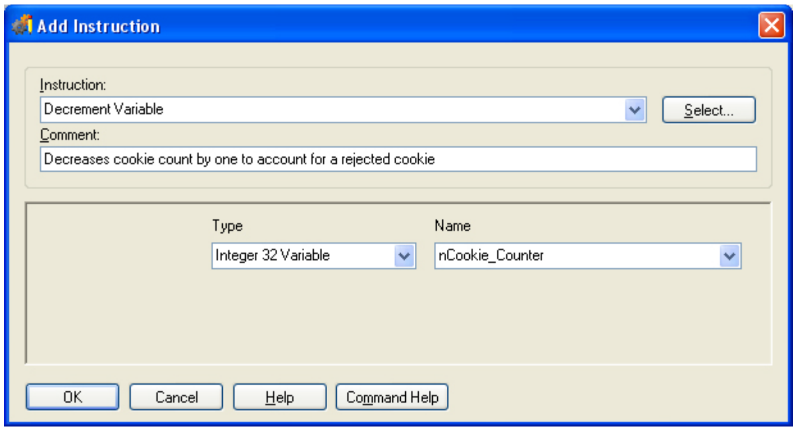
9. Type **dec** in the Instruction field.

   The Decrement Variable command appears, since it's the first command that starts with that text pattern. This is the command we want to use.

10. Press Tab twice to move to the Comment field, and (if you want to) type a comment.
11. Press Tab again to move to the Type field. Press the down arrow on your keyboard twice to select Integer 32 Variable.
12. Press Tab again to advance to the Name field, and then press the down arrow until you see nCookie_Counter.
13. Press Tab again and notice that an outline appears on the OK button.

An outlined button means that pressing the space bar or Enter is equivalent to clicking the button.

Now the dialog box looks like this:



Instruction: Decrement Variable

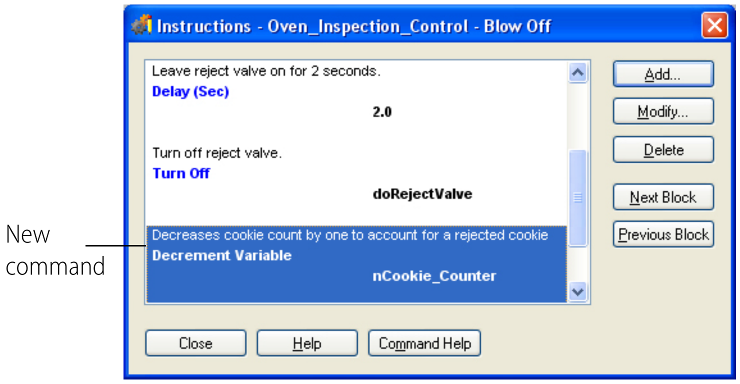
Comment: Decreases cookie count by one to account for a rejected cookie

Type: Integer 32 Variable

Name: nCookie_Counter

14. Press Enter.

The dialog box closes and the new command appears in the Instructions window.



Leave reject valve on for 2 seconds.
Delay (Sec)
2.0

Turn off reject valve.
Turn Off
doRejectValve

Decreases cookie count by one to account for a rejected cookie
Decrement Variable
nCookie_Counter

15. Click Close to return to the Oven_Inspection_Control chart.

Compiling and Downloading the Change

Now we'll compile and download the modified strategy.

1. Click the Debug mode button Debug on the toolbar.

   A message box appears, warning you that changes have been detected and asking if you want to save them before downloading.

2. Click Yes to continue.

   Another warning notes that the strategy name, check sum, or timestamp differs from that in the control engine and asks if you want to continue.

3. Click Yes.

4. On the toolbar, click the Run Strategy button ▶. In the Strategy Tree, double-click the bStartFlag variable. Maximize the dialog box, change the value to 1, and click Apply. Close the dialog box.
5. Click inside the Oven_Inspection_Control chart to make it the active chart, and then click the Auto Step Chart button.

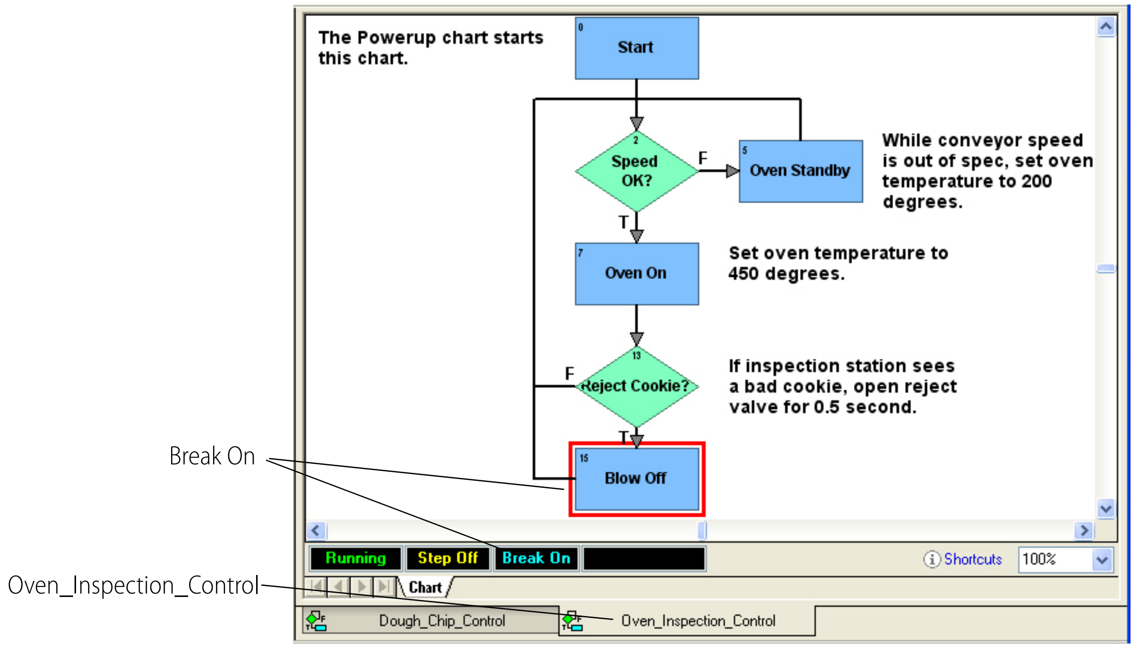
   You see three blocks being processed: Speed OK?, Oven On, and Reject Cookie? The strategy doesn't move into the Blow Off block. That's because an inspector has to flag bad cookies, but we don't have an inspector right now. So we'll simulate what would happen by tripping the digital input that flags a bad cookie.
6. In the Oven_Inspection_Control chart, click the Auto Step Chart button again to stop auto stepping.

   Step Auto changes to Step Off in the status bar.
7. Click the Pause Chart button.

   Step Off changes to Step On.
8. Click the Breakpoint tool and click once on the Blow Off block.

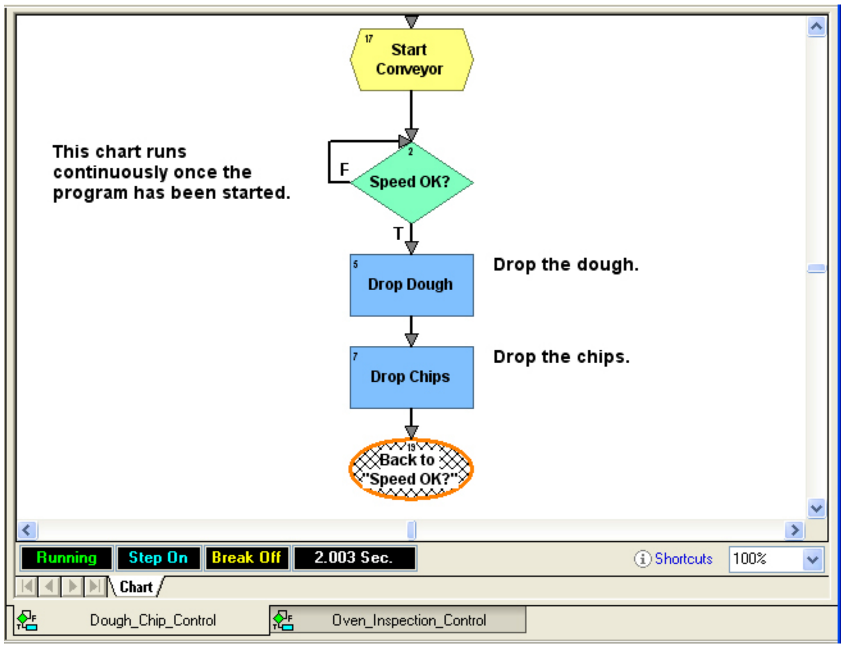
   A breakpoint hand appears on the block.
9. Click the right mouse button or press ESC to release the tool. Notice that Break On now appears in the chart status bar.
10. Click the Breakpoint tool and click once on the Blow Off block.

    A red box appears around the Blow Off block, and Break On now appears in the chart status bar.


Break On
Oven_Inspection_Control
The Powerup chart starts this chart.
0 Start
2 Speed OK?
F
5 Oven Standby
While conveyor speed is out of spec, set oven temperature to 200 degrees.
T
7 Oven On
Set oven temperature to 450 degrees.
F
13 Reject Cookie?
If inspection station sees a bad cookie, open reject valve for 0.5 second.
T
15 Blow Off
Running
Step Off
Break On
Shortcuts
100%
Chart
Dough_Chip_Control
Oven_Inspection_Control

11. Click the Dough_Chip_Control tab at the bottom of the PAC Control main window.

    The chart appears.
12. Click the Pause Chart button to pause the chart.



This chart runs continuously once the program has been started.

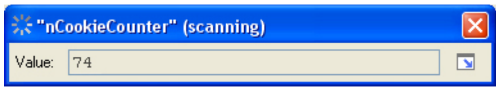
Drop the dough.

Drop the chips.

Using a Watch Window

To see clearly what we're doing, we'll create a watch window to monitor cookie production.

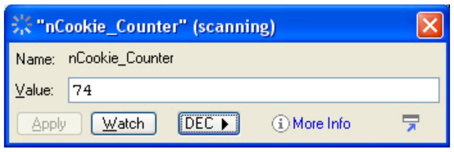
1. In the Strategy Tree, double-click nCookie_Counter.


"nCookieCounter" (scanning)

Value: 74

The value is frozen at some number, such as the 74 shown above. Since the counter is no longer increasing, we can see that cookie production has temporarily stopped.

2. Click the Maximize icon (in the lower right corner) to display the scanning options.

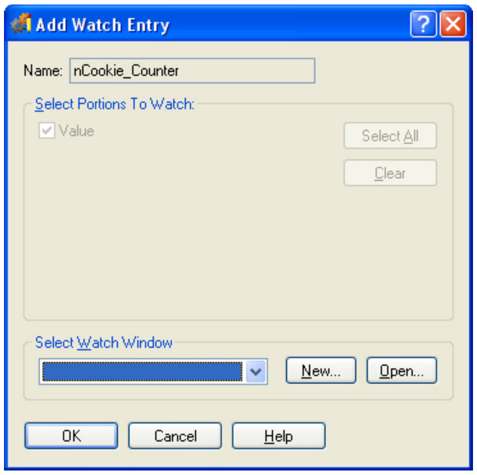

"nCookie_Counter" (scanning)

Name: nCookie_Counter

Value: 74

3. Click Watch.

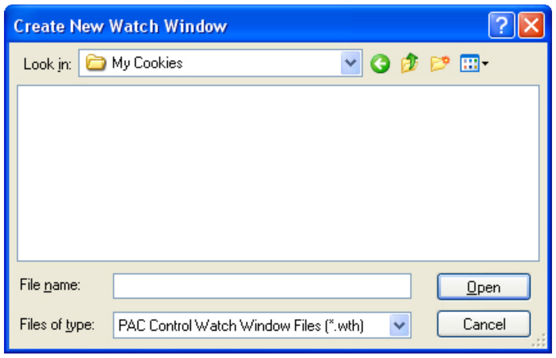
The Add Watch dialog box opens.



Name: nCookie_Counter

4. Since there is no watch window available to select, we'll create one.

   Click New.



Look in: My Cookies

File name:

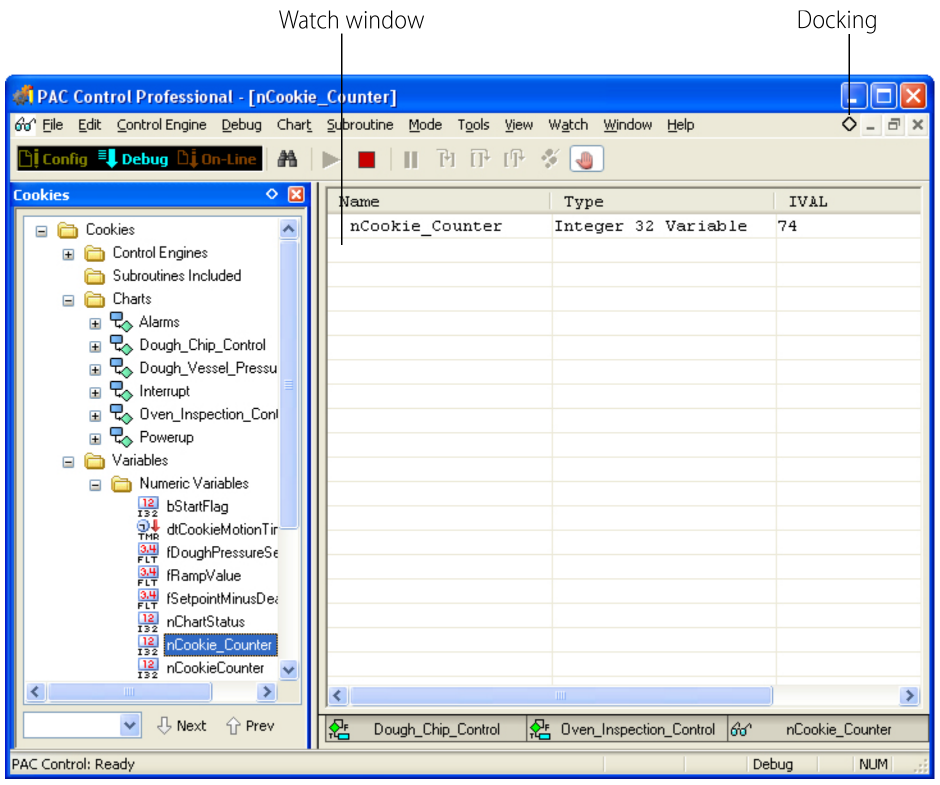
Files of type: PAC Control Watch Window Files (*.wth)

5. Make sure the My Cookies directory appears in the Look in field. Type a name for the watch window in the File name field. Then click Open.

   The watch window name appears in the Add Watch dialog box, and the new watch window appears behind it.

6. In the Add Watch Entry dialog box, click OK. Close the nCookie_Counter view variable dialog box.

The new watch window looks something like this:

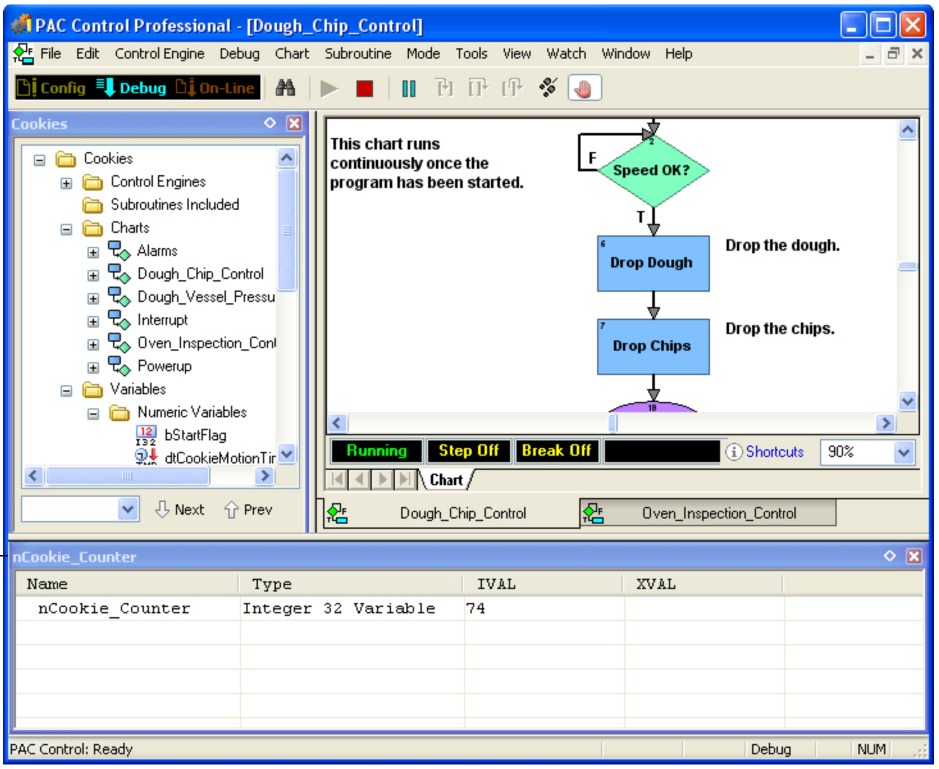


Name	Type	IVAL
nCookie_Counter	Integer 32 Variable	74

Since we want to be able to see chart windows as well as the watch window, let's dock the watch window at the bottom.

7. In the watch window, click the docking button ◇ in the upper-right corner.

The watch window moves to the bottom of the main window.



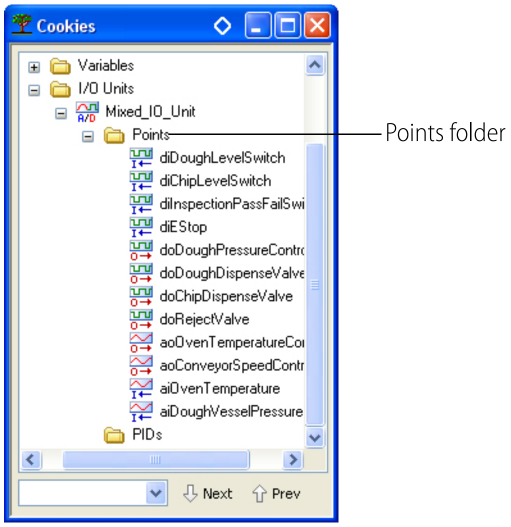
Name	Type	IVAL	XVAL
nCookie_Counter	Integer 32 Variable	74	

Now we'll trip the latch that signals a bad cookie.

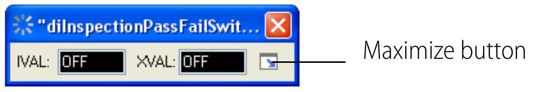
8. On the Strategy Tree, under the I/O Units folder at the bottom of the window, expand Mixed_IO_Unit by clicking the plus sign at the left of the name.

You see a folder named Points.

9. Expand the Points folder to display the digital I/O points configured for this I/O unit.



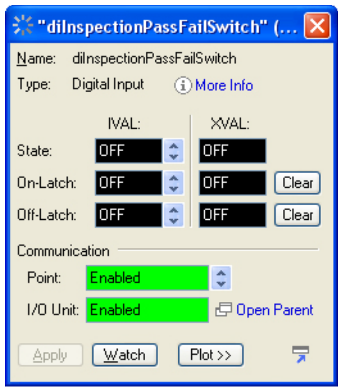
10. Double-click diInspectionPassFailSwitch.



"diInspectionPassFailSwit...

IVAL: OFF XVAL: OFF

11. In the minimized dialog box, click the Maximize button.


"diInspectionPassFailSwitch" (...

Name: diInspectionPassFailSwitch
Type: Digital Input More Info

	IVAL:	XVAL:	
State:	OFF	OFF	
On-Latch:	OFF	OFF	Clear
Off-Latch:	OFF	OFF	Clear

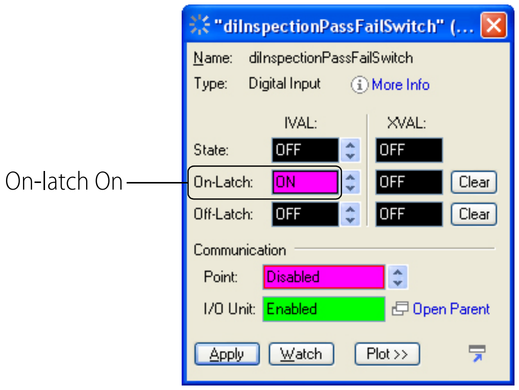
Communication
Point: Enabled
I/O Unit: Enabled Open Parent

NOTE: Don't worry if the red error message "I/O unit not configured" or XVAL values appear in this dialog box. This occurs because the strategy is configured for sample hardware that probably doesn't correspond to your actual I/O units and modules. PAC Control can't locate this hardware. That's okay for this example.

When an inspection finds a bad cookie, the on-latch attached to the I/O point diInspectionPassFailSwitch is supposed to go on. We're going to trip it manually.

12. Click one of the arrows in the On-Latch IVAL field to change it to On. Also make sure that the Enable comm field says No.

The dialog box should look like this:



"diInspectionPassFailSwitch" (...

Name: diInspectionPassFailSwitch
Type: Digital Input More Info

	IVAL:	XVAL:	
State:	OFF	OFF	
On-Latch:	ON	OFF	Clear
Off-Latch:	OFF	OFF	Clear

Communication
Point: Disabled
I/O Unit: Enabled Open Parent

13. Click Apply.

The On-Latch IVAL field turns green after a second or two, indicating the latch is on.

14. Click Add Watch.

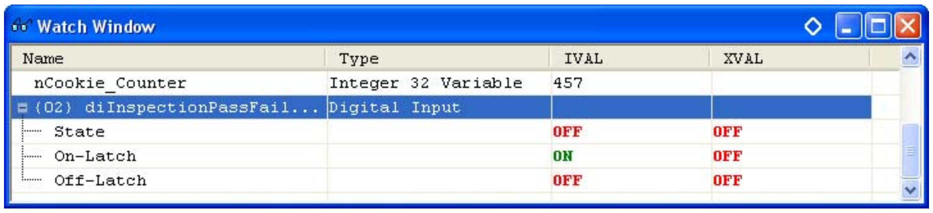
We'll add the variable to our watch window, so we can see what happens.

15. In the Add Watch dialog box, leave all portions checked. Click OK to add the variable to the Cookie Watch window. Close the view point dialog box.

16. In the Cookie Watch window, click the plus sign next to (02) diInspectionPassFailSwitch.

Your screen may show only part of the words.

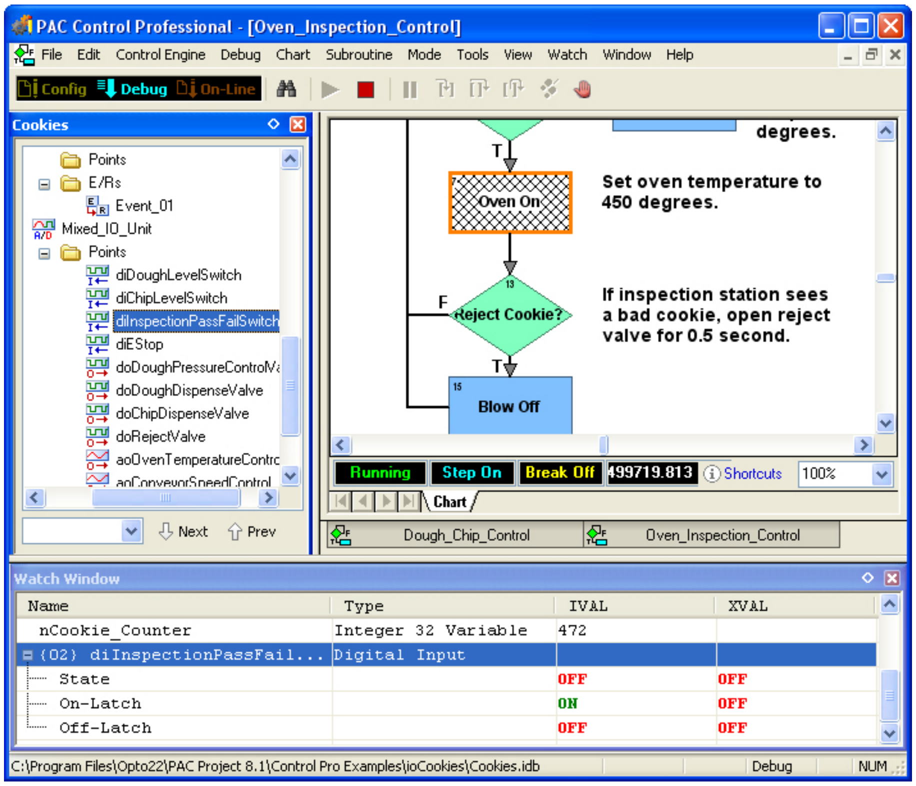
17. Move your cursor over the right side of the Name column until the cursor changes shape. Then click and drag the column to make it wider, until you can see all the words.



Name	Type	IVAL	XVAL
nCookie_Counter	Integer 32 Variable	457	
(02) diInspectionPassFail...	Digital Input		
State		OFF	OFF
On-Latch		ON	OFF
Off-Latch		OFF	OFF

18. Click in the Oven_Inspection_Control chart to make it active, and move the scroll bars until you can see the Blow Off block at the bottom.

Your window now looks something like this:

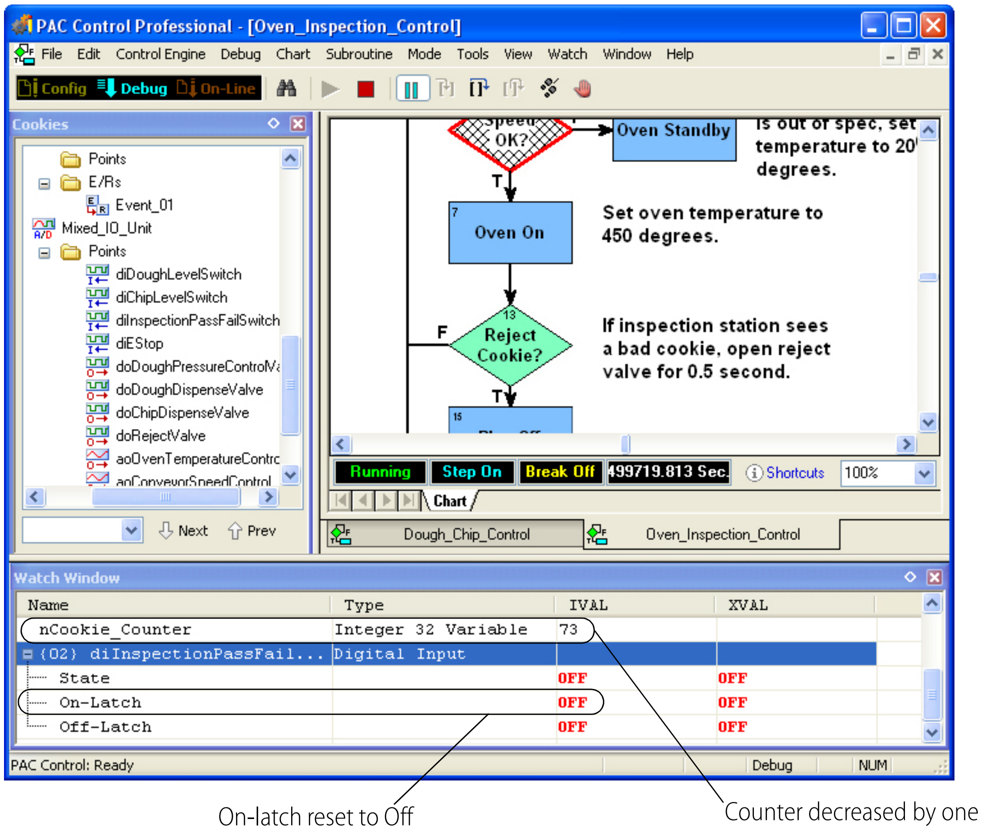


Name	Type	IVAL	XVAL
nCookie_Counter	Integer 32 Variable	472	
(02) diInspectionPassFail...	Digital Input		
State		OFF	OFF
On-Latch		ON	OFF
Off-Latch		OFF	OFF

C:\Program Files\Opto22\PAC Project 8.1\Control Pro Examples\ioCookies\Cookies.idb Debug NUM

19. Click the Step Over button as many times as it takes to get the hatch marks to the Blow Off block. Now watch the nCookie_Counter IVAL value in the watch window as you click the button again.

A bad cookie was found, so the counter was decreased by one. At the same time, the on-latch was reset to Off, as you can also see in the watch window.



Name	Type	IVAL	XVAL
nCookie_Counter	Integer 32 Variable	73	
{02} diInspectionPassFail...	Digital Input		
State		OFF	OFF
On-Latch		OFF	OFF
Off-Latch		OFF	OFF

20. Click the Auto Step Chart button to go back to auto stepping.

The counter does not decrease again, because the on-latch is no longer set. But the counter won't increase until we start the Dough_Chip_Control chart again.

21. Click the Dough_Chip_Control chart tab. Click the Pause Chart button to unpause the chart. Verify that Step On changes to Step Off in the chart status bar.

The watch window shows the nCookie_Counter value going up again.

Closing the Strategy and Exiting

Before we finish this tutorial, you may want to explore the sample strategy on your own. You can double-click items in the Strategy Tree or open up other charts to see what they do. You can also double-click command blocks to see what they contain.

1. When you're ready to stop, click the Stop Strategy button in the toolbar.

   This action prevents the strategy from running continuously unattended on the control engine.

   A dialog box asks if you are sure you want to stop the strategy.

2. Click Stop Strategy.
3. To close the strategy, select File > Close Strategy; or select File > Exit to quit PAC Control. If dialog boxes ask whether you want to remove breakpoints and take charts out of stepping mode, click Yes.

What's Next?

Your introduction to PAC Control is now complete. Using the sample Cookies strategy, you have learned how to:

- Open, save, and close a strategy
- Work with the Strategy Tree
- Work with charts and add commands in blocks
- Configure a control engine
- Compile, run, step through, and add breakpoints to a strategy
- Make an online change
- Use a watch window to monitor variables and I/O points.

The rest of this book expands on the knowledge you've just acquired. Now may be a good time to look at the table of contents or thumb through the book and familiarize yourself with its contents. Some sections you may want to read; others you'll probably just refer to as needed.

3: What Is PAC Control?

OPTO 22

Introduction

The tutorial in Chapter 2 introduced you to PAC Control without explaining much about it. In this chapter we'll learn more about PAC Control and see its main windows and toolbars.

In this Chapter

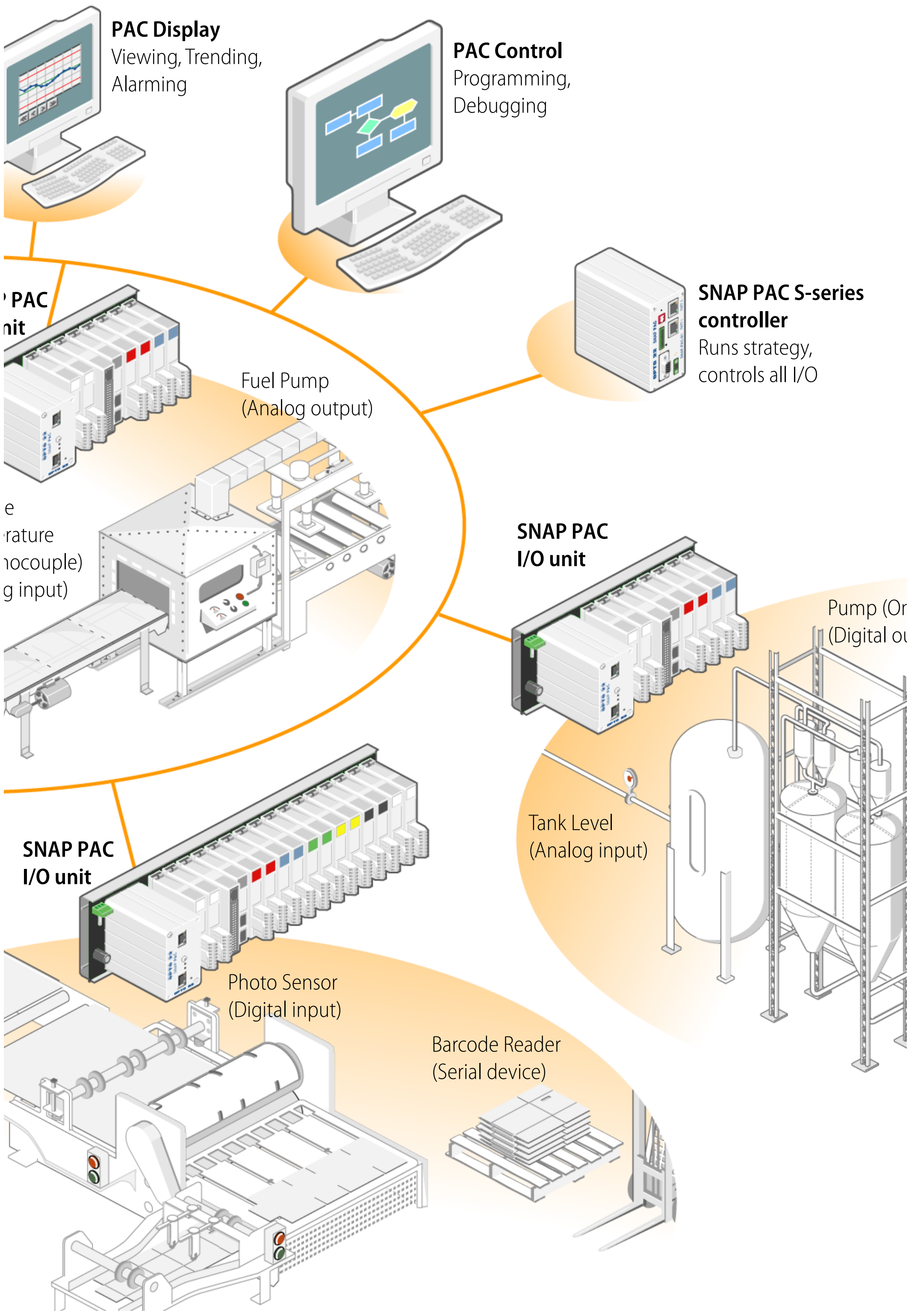
About PAC Control 45
General Control Concepts....................47
PAC Control Terminology 49
PAC Control Main Window 54
Windows and Dialog Boxes in PAC Control 58
Customization Features 65
Online Help 76

About PAC Control

PAC Control is a programming language based on flowcharts. Whether you have PAC Control Basic or PAC Control Professional, you use PAC Control to develop software that monitors and controls all kinds of equipment and sensors, from a simple heating system to a complex factory. The software you develop controls the Opto 22 hardware that runs your heating system or factory.

The diagram below shows how PAC Control on your PC can work with the hardware in your control system. This example shows a small system; yours may be even smaller or considerably larger and more complex. The diagram uses a SNAP-PAC-S1 as the control engine. Input/output (I/O) points on the subordinate SNAP PAC I/O units monitor and control the analog, digital, and serial devices connected to them. All these terms are defined in the following pages.

Control System Example



General Control Concepts

Automation

Automation is a way of adding intelligence to an industrial process. When you automate a process, you are less dependent on human action to carry out the process. The process generally becomes faster, more accurate, and more reliable. For example, take a look at the tank level and pump combination in the diagram on the previous page. Instead of having a person watch the level in the tank and turn the pump on or off when the level gets too low or too high, you can automate the process by installing a processor and I/O. The processor and I/O respond in milliseconds and are on the job 24 hours a day.

Control Engines

In an Opto 22 PAC Control system, the control engine is the programmable component in Opto 22 controllers that provides the intelligence required for automation.

Using PAC Control, you create a software program (a strategy) that tells the control engine how every aspect of a process should work. You download the strategy to an Opto 22 control engine which runs it as a stand-alone application. On a SNAP PAC controller, the strategy is stored in the controller's electronic memory; the development PC can be turned off or used for other operations while the control engine runs the program. Whenever necessary you can modify the strategy and download it again to the control engine.

In the diagram on the previous page, one SNAP PAC S-series controller runs the program that controls the three areas of automation.

Digital and Analog Inputs and Outputs

An industrial process can include many different hardware components: switches, pumps, tanks, valves, furnaces, conveyors, photo sensors, thermocouples, and so on. All the components communicate with the control engine in the controller by way of input/output (I/O) points.

Input points are wired to hardware that brings information into the control engine from the process. Examples of devices that can be wired to input points are thermocouples, switches, and sensors. The control engine takes the information from the input points—such as whether a switch is on or what temperature is registered on a sensor—processes it using the software instruction set, and returns information to the process through output points.

Output points are wired to hardware that receives information from the control engine and uses this information to control components of the process. For example, lights, motors, and valves are all devices that can be wired to output points. Using an output point, the control engine might turn on a light or open a valve.

There are two types of I/O points, digital and analog:

- **Digital points** can be either on or off (True or False). Push buttons and LEDs are examples of digital devices. An LED is either on or off; it has no other possible state.

  In the Control System Example on page 46, the photo sensor is an example of a digital input device. The photo sensor is either on or off. When it turns off as the sheet passes through its beam, the digital I/O module tells the control engine it is off, and the control engine responds as programmed to stamp the sheet.

The pump is an example of a digital output device. Based on information from the tank level input, the control engine turns the pump on or off as programmed.

- **Analog points** have a range of possible values. Temperature and pressure are examples of analog information. Temperature might be any number in a range, -2 or 31.65 or 70.1 or many other possible numbers.

  In the Control System Example on page 46, the tank level sensor is an analog input device. It registers the changing level of water in the tank and reports the level to the control engine, which responds by turning the pump on or off.

  The fuel pump is an example of an analog output device. Based on information about the temperature in the furnace, the control engine adjusts the pressure of fuel through the pump as programmed.

SNAP Serial Communication Modules

In the PAC Control system, some I/O modules do not contain standard analog or digital input/output points, but are used for communication over a serial network.

SNAP serial communication modules provide one or two channels for sending data to or receiving data from a serial device (RS-232 or RS-485/422) or a protocol like PROFIBUS or Controller Area Network (CAN) that can communicate over a serial network. In the Control System Example on page 46, the barcode reader is an example of a serial device.

Key Features

See the following topics for some of PAC Control's key features:

- "Copying I/O Configurations" on page 125

  Using a configuration file, you can copy an I/O configuration from one strategy to another. Or, using PAC Manager, you can send the configuration to multiple I/O units at the same time.
- "Using Network Segmenting in PAC Control" on page 103

  You can leverage the two independent Ethernet network interfaces on a SNAP PAC controller to segment the control network from the company network.
- "Using Ethernet Link Redundancy in PAC Control" on page 104

  (Pro only) If you are using SNAP PAC controllers and PAC Control Professional, you can use Ethernet link redundancy to set up alternate network links.
- "Configuring Event/Reactions" on page 162

  (Pro only) For *serial-based* mistic *I/O units* you can configure specific reactions to events when they occur.
- "Persistent Data" on page 256

  You can configure the data in most variables to be persistent. The variable's value is saved in the controller's memory; it does not change when the strategy is run, stopped, or started, and it does not change if the strategy is changed and downloaded again.

- "Pointer Commands" on page 355

  For advanced programming, you can create **pointers** that store the memory address of a variable or some other PAC Control item, such as a chart, an I/O point, or a PID loop. You can perform any operation on the pointer that you could perform on the object the pointer points to.

- "Setting Initial Values in Variables and Tables During Strategy Download" on page 261

  When you are adding table variables in PAC Control, you can set all table elements to one initial value. Or you can set each individual table element to its own value by creating an initialization file to download with your PAC Control strategy.

- "Using Subroutines" on page 407"12: Using Subroutines" on page 399

  You can use subroutines that are independent from strategies, but that can be called from any chart or other subroutine in your strategy. Subroutines offer two ways to work with variables and other logical elements: they can be passed in or they can be local to the subroutine.

- "Configuring PID Loops" on page 150

  A proportional integral derivative (PID) control loop (often referred to as a PID loop) monitors a process variable, compares the variable's current value to a desired value (a setpoint), and calculates an output to correct error between the setpoint and the variable. Because the calculation is complex, it is done by a mathematical formula, or algorithm, that you can then adjust (tune) for each PID loop.

PAC Control Terminology

action block

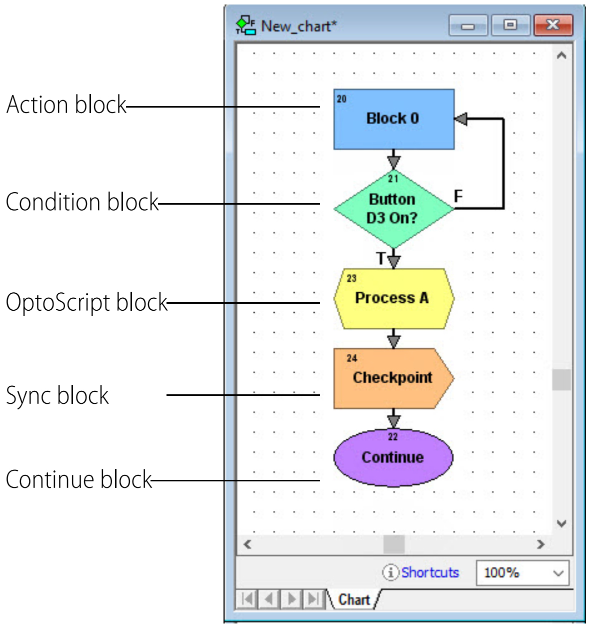
Action blocks are rectangular blocks in a flowchart that contain one or more instructions (actions) that do the work of the strategy, such as turning things on or off, setting variables, and so on. Several instructions can be placed in one action block. Action blocks can have many entrances, but only one exit.

analog point

Analog points have a range of possible values. Temperature and pressure are examples of analog information. Temperature might be any number in a range, -2 or 31.65 or 70.1 or many other possible numbers. Analog points can be either inputs (such as a tank level sensor) or outputs (such as a fuel pump).

blocks

A chart is made up of action blocks, condition blocks, continue blocks, and OptoScript blocks connected by arrows, which show how the process flows. A chart used with redundant controllers also has sync blocks.



sync block

The sync block tool is visible only when controller redundancy support is enabled. A sync block is used to synchronize redundant controllers at strategic locations in a *transactional* chart (a chart that contains sync blocks). For more information, see form 1831, the *SNAP PAC Redundancy Option User's Guide*.

condition block

Condition blocks are diamond-shaped blocks in a flowchart that contain questions (conditions) that control the logical flow of a strategy. Condition blocks can have many entrances, but only two exits: True and False. The block can contain more than one condition, and you can use AND or OR to indicate whether all conditions must be true to exit True, or whether any one condition must be true to exit True.

continue block

Continue blocks are oval-shaped blocks in a flowchart that route the flow of execution to an action, condition, or OptoScript block. They do not contain any instructions, but store only the name of the next block to execute. Continue blocks can have many entrances, but no exits, since the exit is defined within the block. Continue blocks avoid awkward connections between two blocks that are far apart.

digital point

Digital points can be either on or off (true or false). Push buttons and LEDs are examples of digital devices: an LED is either on or off; it has no other possible state. Digital points can be either inputs (such as a photo sensor) or outputs (such as a pump).

external value

The external value (XVAL) is the real-world value measured or set by an I/O point. The PAC Control strategy reads and writes only to internal values, and transfers internal values to external values if communication to the associated I/O unit is enabled.

flowcharts

Since most control applications are complex, the strategy typically consists of a series of process flowcharts, or *charts*, that all work together. Each chart controls one aspect of the strategy—one piece of the automated process. Together, all the charts constitute the strategy. The total number of charts in a strategy is limited only by the amount of memory available in the control engine.

A chart can be running, suspended, or stopped. A *running chart* is actively performing its assigned task. A *suspended chart* is temporarily paused. A *stopped chart* is inactive. Every chart in a PAC Control strategy can change the status of any other chart in the strategy, yet every chart is independent of every other chart. Any combination of charts can be running simultaneously, up to the maximum limit allowed on the control engine. (See also, multitasking.)

Every strategy automatically contains a Powerup chart. The Powerup chart is automatically started when the strategy begins running, so it starts other charts. All other charts you create, based on the needs of your process.

input point

Input points are wired to hardware that brings information into the brain from the process. Examples of devices that can be wired to input points are thermocouples, switches, and sensors. The control engine in the brain takes the information from the input points—such as whether a switch is on or what temperature is registered—processes it using commands in the strategy, and returns information to the process through output points.

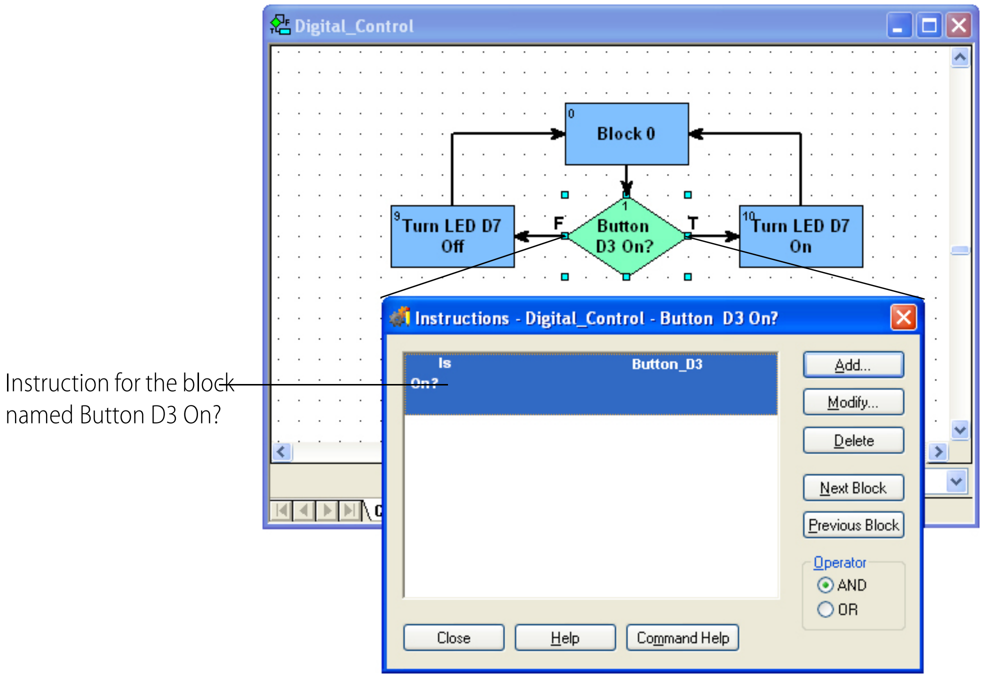
instructions (commands)

PAC Control commands, or instructions, tell the control engine what to do at each step in the flowchart to control the process. Each block in a chart contains one or more instructions, such as *Convert Number to String* or *Start Counter* or *Chart Running?*

Commands are in two forms: Actions and Conditions.

- **Action commands** do something in the process; for example, *Convert Number to String* and *Start Counter* are both action commands. On the flowchart they appear as instructions in action blocks, which are rectangular in shape. They may also appear in hexagonal OptoScript blocks.
- **Condition commands** check a condition and are in the form of a question. *Chart Running?* and *Variable False?* are examples of condition commands. They appear as instructions in condition blocks, which are diamond-shaped, or in hexagonal OptoScript blocks. Condition commands are questions that always have two possible answers, either yes or no (true or false). The answer determines the flow of logic and what happens next in the chart.

The instruction shown below is for the condition block, *Button D3 On?* This block contains only one instruction. As you look at the chart, you can see that the answer to the question in the instruction determines whether the process flows to *Turn LED D7 On* or to *Turn LED D7 Off.*



Is On?	Button_D3

internal value

The internal value (IVAL) is the value read or written by the PAC Control strategy. Internal values are transferred to external values only when communication to the I/O unit is enabled.

multitasking

The control engine can run several charts seemingly at once, each performing a different task through a time-slicing technique called multitasking (also called multicharting). Opto 22SNAP Ultimate I/O control engines can run up to eight charts plus one host task simultaneously; SNAP PAC R-series and SNAP-LCE controllers can run up to 16 charts plus one host task simultaneously; SNAP-PAC S-series controllers can run up to 32 charts plus one host task simultaneously; a SoftPAC controller can run up to 64 charts plus one host task simultaneously. The host task is an invisible chart used to communicate to a PC, which may be running PAC Control in Debug mode or PAC Display.

Each chart in a running or suspended state counts toward the total that can run simultaneously. Charts that are stopped do not. When the Powerup chart is running, it also counts.

The actual order and timing for running tasks is not deterministic—that is, it is not always the same, but depends on priorities at any given time. For example, communication may sometimes take a higher priority than a running chart.

OptoScript block

OptoScript blocks are hexagonal blocks in a flowchart that contain OptoScript code. OptoScript is a procedural language that can simplify tasks such as math computations, string handling, and complex loops and conditions. OptoScript blocks can have more than one entrance but only one exit.

output point

Output points are wired to hardware that receives information from the brain and uses this information to control components of the process. For example, lights, motors, and valves are all devices that can be wired to output points. Using an output point, the control engine in the brain might turn on a light or open a valve.

pointer

A pointer does not store the value of a variable; instead, it stores the memory address of a variable or some other PAC Control item, such as a chart, an I/O point, or a PID loop. You can perform any operation on the pointer that you could perform on the object the pointer points to. Pointers are an advanced programming feature and are very powerful, but they also complicate programming and debugging.

Because pointers can point to any data type, pointer tables can store an assortment of data types in a single table.

Pointers allow one level of indirection within PAC Control; a pointer cannot point to another pointer. After you add a pointer through the "Add Variable Dialog Box" on page 258, you can use it wherever the object itself would be used.

strategy

The software program you create using PAC Control Basic or PAC Control Professional. The strategy includes all the definitions and instructions (commands) necessary to control the process that one Opto 22 controller handles. Since most control processes are complex, the strategy typically consists of a series of process flowcharts that all work together, with each chart controlling one piece of the automated process. You may have several PAC Control systems, each controlling a different process and therefore running different strategies. Or you may have two or more PAC Control systems controlling identical processes in different areas and running the same strategy.

variables

A variable is a holding place that represents a piece of information in a strategy, such as the parameters for communication, temperature reported by a thermocouple, the name of a chart, or a group of words and numbers to be sent to a display. The information a variable represents is called the *value* of the variable. As a strategy runs, the variable's name remains the same, but its value may change. For example, the value of a variable named Oven_Temperature may change several times while its strategy is running, but its name remains Oven_Temperature.

A variable stores one of six types of data: floating point, integer, timer, string, point, or communication handle. When you create the variable, you designate the type of data it contains.

- A **floating point** (or float) is a numeric value that contains a decimal point, such as 3.14159, 1.0, or 1234.2. A good example of a float variable is one that stores readings from an analog input, such as a thermocouple.
- An **integer** is a whole number with no fractional part. Examples of integer values are -1, 0, 1, 999, or -456. The state of a switch, for example, could be stored in an integer variable as 1 (on) or 0 (off).
- A **timer** stores elapsed time in units of seconds with resolution in milliseconds. Up timers count up from zero, and down timers start from a value you set and count down to zero. For example, you could set a down timer to make sure a value is updated at precise intervals.

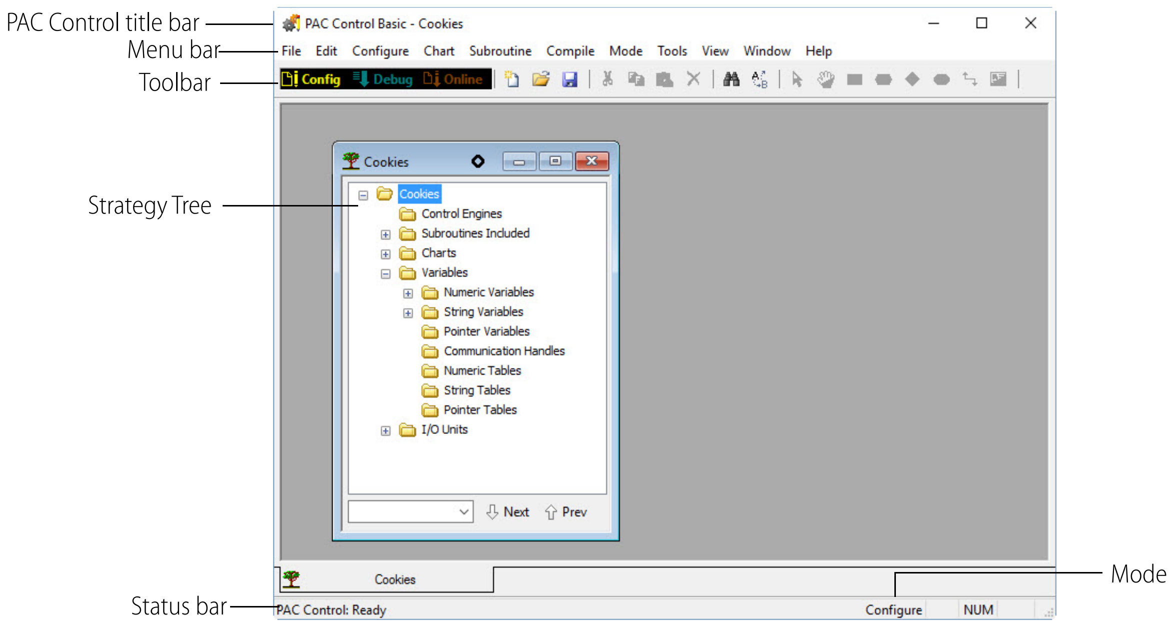
- A **string** stores text and any combination of ASCII characters, including control codes and extended characters. For instance, a string variable might be used to send information to a display for an operator to see.

  A string variable can contain numeric characters, but they no longer act as numbers. To use them in calculations, you must convert them into floating point or integer numbers. And a numeric value to be displayed on a screen must be converted into a string first.
- A **pointer** does not store the value of a variable; instead, it stores the memory address of a variable or some other PAC Control item, such as a chart or an I/O point.
- **Communication handles** store parameters needed for communication with other devices.

You can use variables that are individual pieces of information, and you can also use *table variables*, which are groups of related information in the form of a table.

PAC Control Main Window

With a strategy open, the main window in PAC Control Basic and PAC Control Pro look similar to this:



Since PAC Control uses standard Microsoft Windows conventions, you'll recognize the title bar and menu bar and already be familiar with some of the menus, such as File, Edit, View, Window, and Help. This section discusses some things that may not be familiar.

Status Bar

The status bar at the bottom of the window shows you information about PAC Control. When you move your mouse over the toolbar, a description of the button you're pointing to appears in the status bar. The status bar indicates the mode PAC Control is in and indicates messages or errors helpful in debugging.

To hide or show the status bar, choose View > Status Bar. A check mark next to the menu item means the status bar will show; no check mark means it is hidden.

Mode

You can run PAC Control in three modes: Configure, Debug, or Online. The current mode is shown in the status bar, on the right. Toolbars and menus change depending on the mode.

- **Configure mode** is used to create, modify, save, and compile strategies, flowcharts, and subroutines; and to configure control engines, I/O, and variables.
- **Debug mode** is used to download, run, and debug strategies and to view control engine and communication status and errors while the strategy is running.
- **Online mode** is a scaled-down version of Configure mode, used to change a strategy while it is running. You cannot add variables and I/O in Online mode, but you can change ones that already exist.

NOTE: When you change a chart in Online mode, a new copy of that chart is downloaded to the control engine, but the old one is not deleted. After you have made a few online changes, the additional chart copies begin to take up memory. To avoid memory problems, be sure you stop the strategy after making several online changes and completely compile and download it to clear out old chart copies.

To change modes, choose the mode you want from the Mode menu, or click its button in the toolbar.

Toolbars

Toolbars give you shortcuts for doing many things that also appear on menus. Toolbars in PAC Control include standard Windows buttons like New and Save as well as special buttons for PAC Control functions. The tools that you can use depend on which mode you're in. Tools not currently available are grayed out.

The following toolbars are standard in PAC Control.

"Mode Selection Toolbar"

"Configure Mode and Online Mode Toolbar"

"Debug Mode Toolbar"

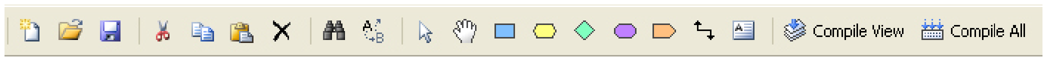
Mode Selection Toolbar

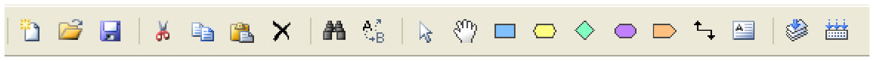
The mode selection tools are always available.

Tool	Description
Config	Configuration Mode
Debug	Debug Mode
Online	Online Mode

Configure Mode and Online Mode Toolbar

The following toolbar appears in configure mode and online mode.



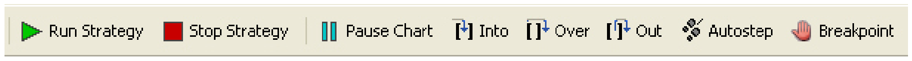


Tool	Name
🗋	New Strategy
📂	Open Strategy
💾	Save Strategy
✂	Cut
⧉	Copy
📋	Paste
✕	Delete
🔍	Find
A→B	Find and Replace
↖	Select Tool

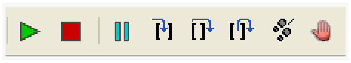
Tool	Name
✋	Pan Tool
▭	Action Block Tool
⬭	OptoScript Block Tool
◇	Condition Block Tool
⬬	Continue Block Tool
⬠	Checkpoint Block Tool
↳	Connect Tool
▤	Text Tool
⚙	Compile Active View
▦	Compile All

Debug Mode Toolbar

The following toolbar appears in debug mode.



In a smaller window, the debug toolbar looks like this.

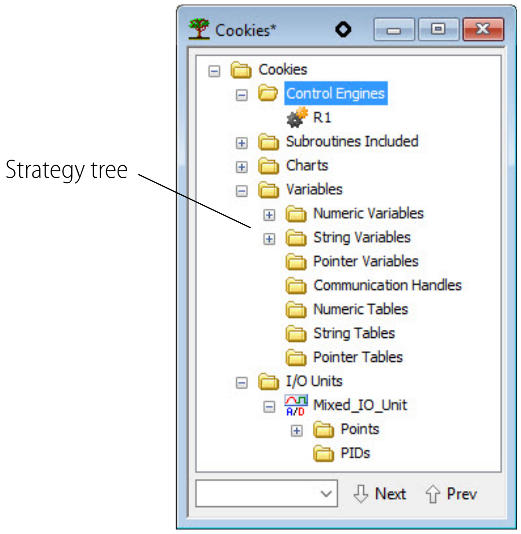


Tool	Name
▶	Run Strategy
■	Stop Strategy
⏸	Pause Chart
[↓]	Into

Tool	Name
[]↓	Over
[↑]	Out
⁂	Auto Step
✋	Breakpoint

Strategy Tree

The Strategy Tree opens when you open a strategy, and closing it is equivalent to closing the strategy. The Strategy Tree shows you all the elements of your strategy: control engines, flowcharts, subroutines, variables, I/O units and points, and PIDs.



The Strategy Tree works just like Windows Explorer: you can expand or collapse folders to view or hide what is in them. You can easily see what is in your strategy, open elements to change them by double-clicking them, or open a pop-up menu by right-clicking on an element.

Each element in the strategy is represented by a button, shown just to the left of its name. The table below shows the buttons and what they represent.

Button	Description	Button	Description
⚙	Control Engine	I32	Integer 32 Table
◇	Chart	I64	Integer 64 Table
▲	Subroutine	STR	String Table
12 I32	Integer 32 Variable	PTR	Pointer Table
75 I64	Integer 64 Variable	D	Digital I/O Unit
3.4 FLT	Float Variable	A/D	Mixed I/O Unit
TMR	Down Timer Variable	I←	Analog Input Point
TMR	Up Timer Variable	I←	Digital Input Point
AB STR	String Variable	O→	Analog Output Point
CH	Communication Handle	O→	Digital Output Point
FLT	Float Table	PID	PID Loop
→ PTR	Pointer Variable		

Windows and Dialog Boxes in PAC Control

Windows and dialog boxes in PAC Control follow Microsoft Windows standards. You can minimize, maximize, move, resize, and tile them as needed. See your Microsoft Windows documentation for information on how to perform these functions.

The following topics describe other useful features in PAC Control:

"Using Tabs to View Open Windows" on page 59

"Docking Windows" on page 60

"Zooming in a Chart or Subroutine Window" on page 61

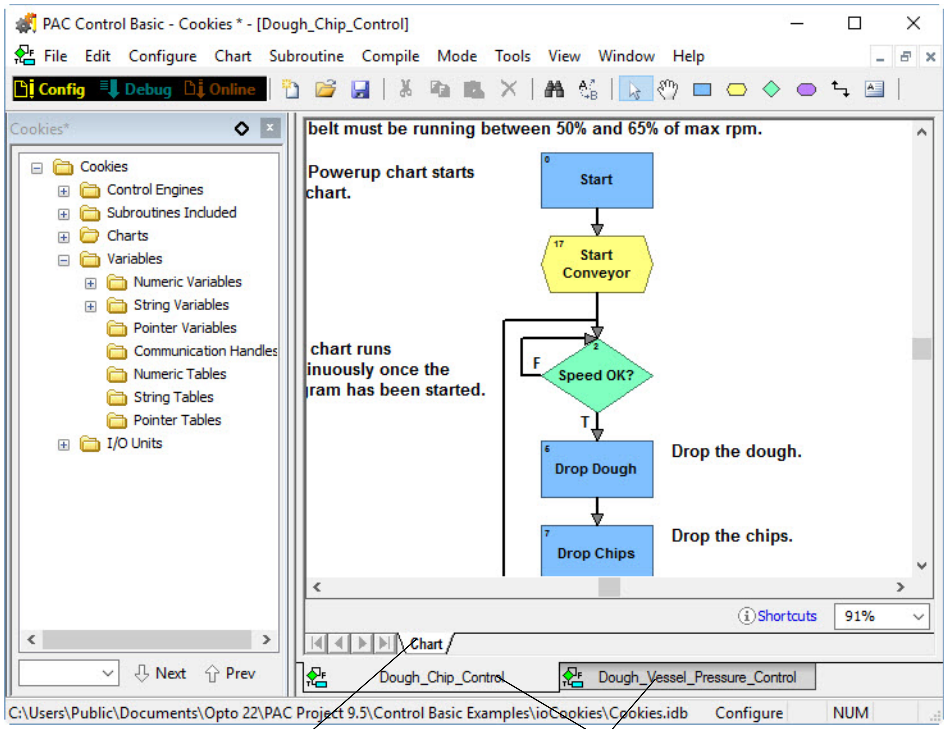
"Redrawing a Chart or Subroutine Window" on page 63

"Changing Column Width in a Dialog Box" on page 64

"Sorting Data in a Dialog Box" on page 65

Using Tabs to View Open Windows

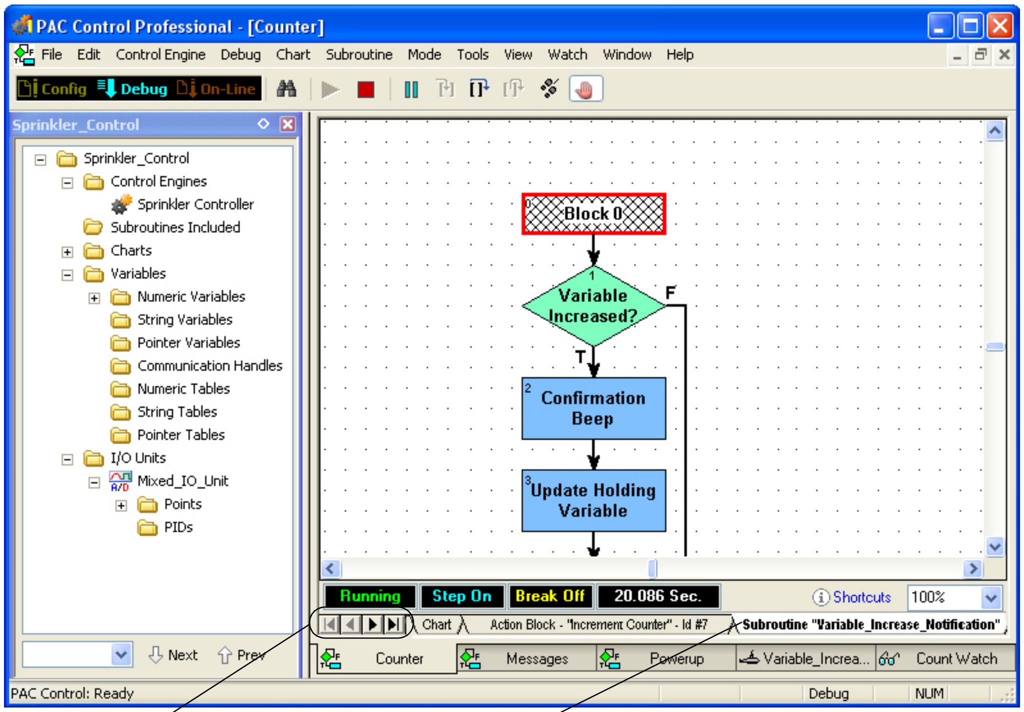
When multiple windows are open—especially if they are maximized—it can be difficult to know where you are, and windows can become lost behind each other. However, you can click the tabs at the bottom of the main PAC Control window to move among chart windows, subroutine windows, watch windows, and blocks you may be stepping through for debugging.


PAC Control Basic - Cookies * - [Dough_Chip_Control]
File Edit Configure Chart Subroutine Compile Mode Tools View Window Help
Config Debug Online
Cookies*
Cookies
Control Engines
Subroutines Included
Charts
Variables
Numeric Variables
String Variables
Pointer Variables
Communication Handles
Numeric Tables
String Tables
Pointer Tables
I/O Units
Next
Prev
belt must be running between 50% and 65% of max rpm.
Powerup chart starts chart.
chart runs inuously once the ram has been started.
Start
Start Conveyor
Speed OK?
F
T
Drop Dough
Drop the dough.
Drop Chips
Drop the chips.
Shortcuts
91%
Chart
Dough_Chip_Control
Dough_Vessel_Pressure_Control
C:\Users\Public\Documents\Opto 22\PAC Project 9.5\Control Basic Examples\ioCookies\Cookies.idb
Configure
NUM

White tabs show where you are when stepping through a chart or subroutine.

Gray tabs show open windows. Click a tab to bring its window into view.

The upper layer of tabs appears when you are stepping through your strategy in Debug mode. It acts as a kind of call stack to let you see how you got to the current block or command.



Click the arrow buttons to see tabs that are not visible or are only partly visible.

This white tab shows you are stepping inside the *Variable_Increase_ Notification* subroutine, which was called by the *Increment Counter* action block on the gray tab at left. The gray *Chart* tab farther left shows the chart this action block is in. Click any tab to see how you got to where you are.

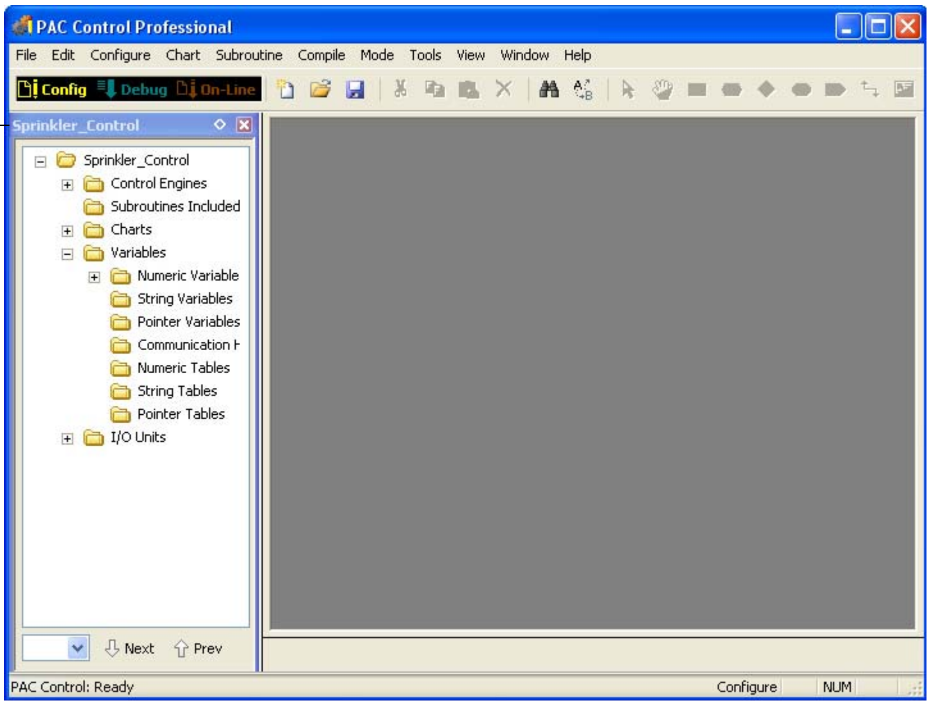
Docking Windows

You can place the strategy Tree and watch windows where you want them (*dock* them) in the PAC Control main window. Docked windows do not have tabs but are contained in their own frames. PAC Control remembers their position the next time you open that strategy.

- To dock a window, click the docking button ◇ in the window's title bar.

The window moves to its own frame. (Compare the following figure with the one on page 54 to see the difference in the Strategy Tree window.)

Docked strategy Tree



- To change the docked position, click and drag the window's title bar to any edge of the main window.
- To change the docked window's width or height, click and drag the edge of its frame in the direction you want.
- To free the window from its docked position, click the docking button ◇ in its title bar.

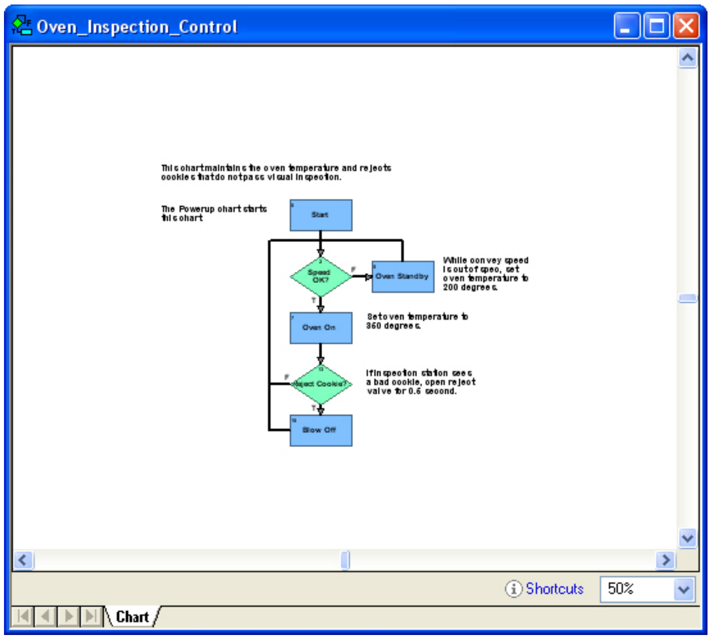
Zooming in a Chart or Subroutine Window

You can change the scale of any chart or subroutine window by using the zoom feature. There are several ways to zoom:

- Hold down the Ctrl key and move the mouse wheel up and down.
- Press the + or - keys on the keyboard.
- Right-click anywhere on a chart or subroutine window to display a pop-up menu. Select Zoom and choose In, Out, or Reset (to return the zoom to 100 percent).
- From the View menu, choose Zoom In, Zoom Out, or Zoom Reset.
- To pick the zoom percentage you want, click the zoom field at the bottom right of the window and choose one of the preset zoom levels. The "Zoom to fit" option resizes the chart to fit in the window.

NOTE: Zooming always takes place with reference to the center point of the window.

Here is a window at 50 percent zoom:



This chart maintains the oven temperature and rejects cookies that do not pass visual inspection.

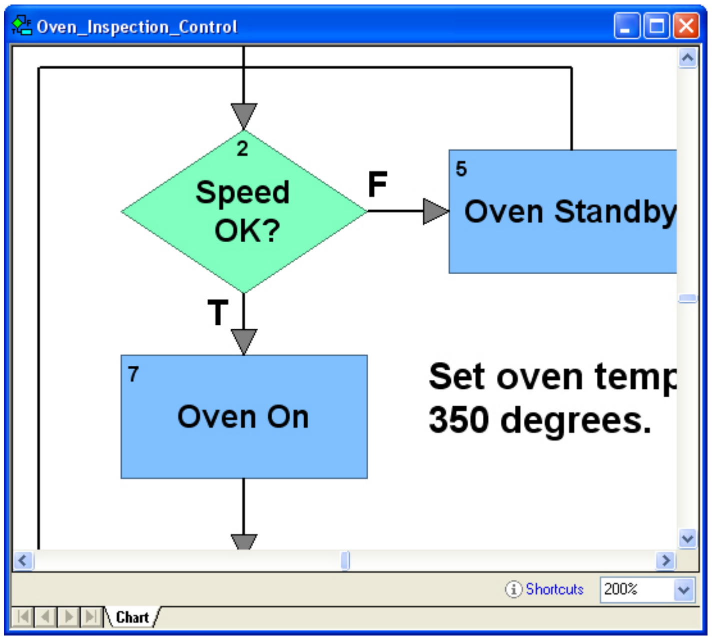
The Powerup chart starts this chart

While convey speed is out of spec, set oven temperature to 200 degrees.

Set oven temperature to 350 degrees.

If inspection station sees a bad cookie, open reject valve for 0.5 second.

The same window at 200 percent zoom looks like this:

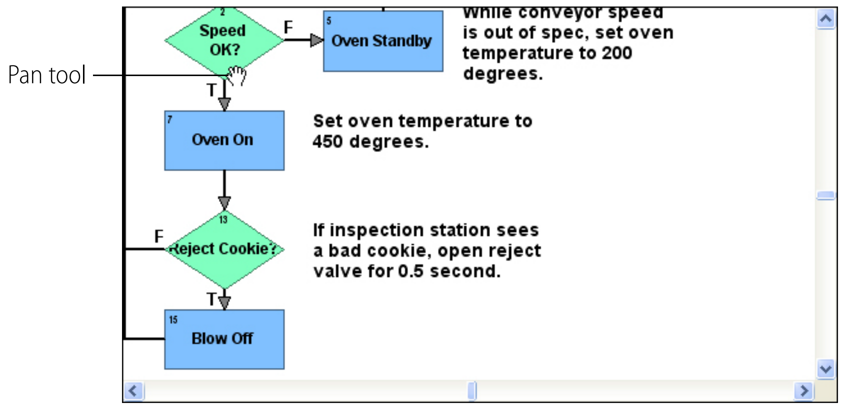


Panning a Chart

The pan tool allows you to easily pan around a flowchart.

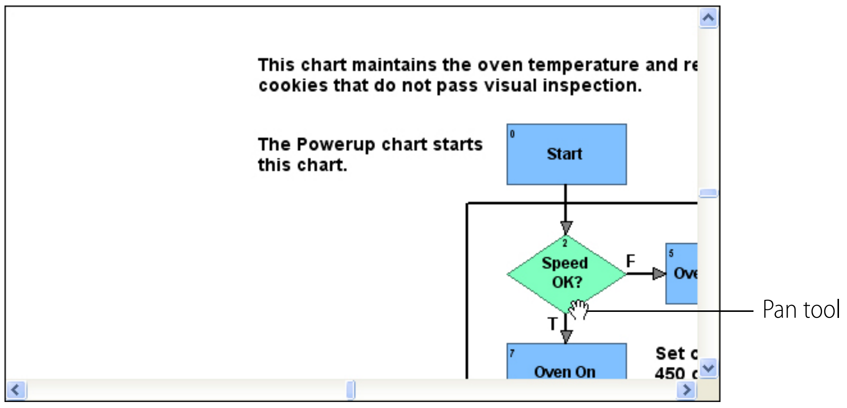
1. Using the pan tool , click and hold to grab a portion of a chart.

You can access the pan tool either from the toolbar or by holding down the space bar and clicking the left mouse button.


Pan tool
Speed OK?
F
Oven Standby
While conveyor speed is out of spec, set oven temperature to 200 degrees.
T
Oven On
Set oven temperature to 450 degrees.
F
Reject Cookie?
If inspection station sees a bad cookie, open reject valve for 0.5 second.
T
Blow Off

2. Move the pan tool to a new position and release.

   The chart moves to a new position.

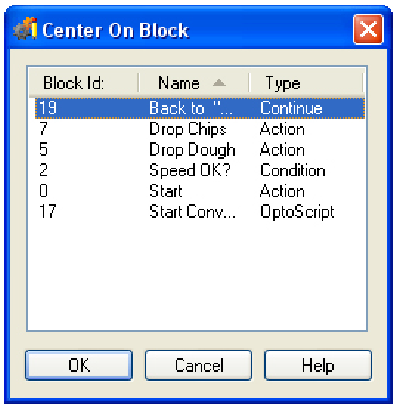

This chart maintains the oven temperature and re cookies that do not pass visual inspection.
The Powerup chart starts this chart.
Start
Speed OK?
F
Ove
T
Pan tool
Oven On
Set c 450 c

Redrawing a Chart or Subroutine Window

If you want to move quickly to a particular block, you can redraw a chart or subroutine window with any block at its center.

1. With a chart or subroutine open in the active window, select View > Center On Block, or right-click in the chart or subroutine window and choose Center On Block from the pop-up menu.

The Center On Block dialog box appears, listing all blocks in the chart.



Block Id:	Name	Type
19	Back to "...	Continue
7	Drop Chips	Action
5	Drop Dough	Action
2	Speed OK?	Condition
0	Start	Action
17	Start Conv...	OptoScript

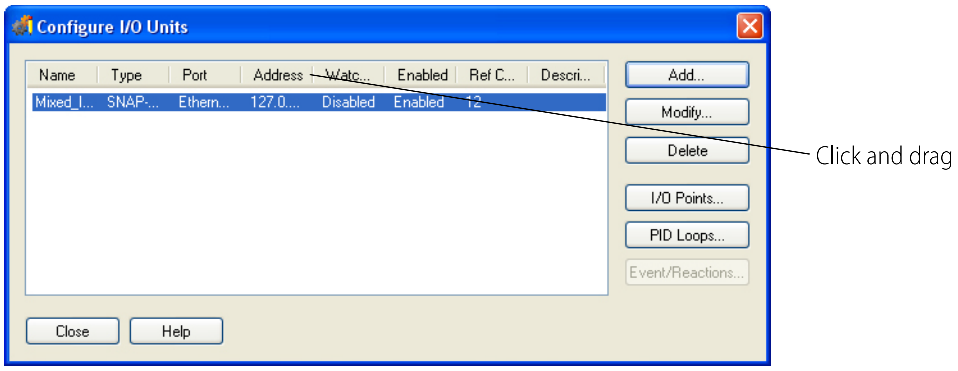
2. Double-click the block you want, or click it once and click OK.

The chart or subroutine is redrawn with the selected block in the center of the window.

Changing Column Width in a Dialog Box

Many dialog boxes include several columns of information. To see all the data in some columns, you may need to make columns wider.

- To widen or narrow a column, click the right edge of the column label and drag it horizontally.

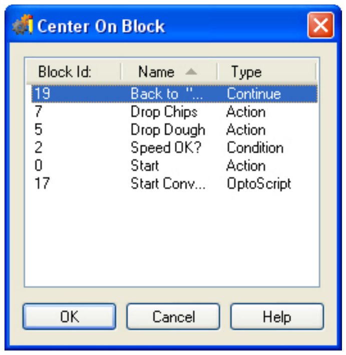


Name	Type	Port	Address	Watc...	Enabled	Ref C...	Descri...
Mixed_I...	SNAP-...	Ethern...	127.0....	Disabled	Enabled	12	

- To resize a column to the exact width of its longest entry, double-click the line that separates the column from the one on its right.

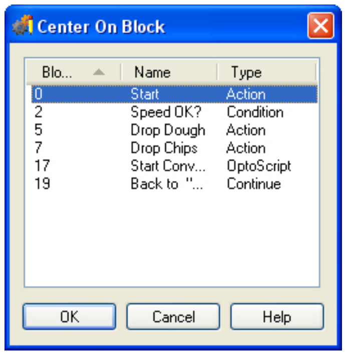
Sorting Data in a Dialog Box

In some dialog boxes with columns of data, you can sort the information in the way you want to see it. The Center On Block dialog box provides an example. The blocks in this dialog box normally appear in alphabetical order by the block name.



Block Id	Name	Type
19	Back to "...	Continue
7	Drop Chips	Action
5	Drop Dough	Action
2	Speed OK?	Condition
0	Start	Action
17	Start Conv...	OptoScript

- To sort by block number instead, click the Block Id column label.



Blo...	Name	Type
0	Start	Action
2	Speed OK?	Condition
5	Drop Dough	Action
7	Drop Chips	Action
17	Start Conv...	OptoScript
19	Back to "...	Continue

- To sort in the opposite order (descending numbers instead of ascending numbers), click the Block Id column label again.

Some dialog boxes don't allow custom sorting. If you click a column label in these dialog boxes, nothing happens.

Customization Features

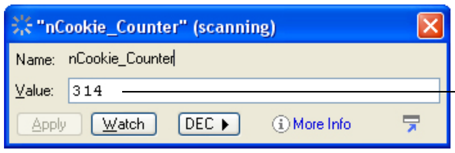
This section describes PAC Control special features for displaying numeric data, launching other applications from PAC Control, and modifying strategies using PAC Control's Command-Line Interface tool.

See also "Legacy Options" on page 232 and "Advanced Options" on page 235.

Setting Decimal, Binary, or Hex Display Mode

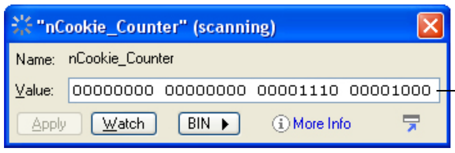
You can set up PAC Control to show the values of all integers and integer tables in decimal (the default), binary, or hexadecimal (hex) format. Binary and hex formats help you use masks. Hex format is particularly useful for entering integer literals and initial values for a digital-only Ethernet brain. Hex view is available in all modes: Configure, Debug, and Online.

The default view is in decimal notation, as shown in the following figure.



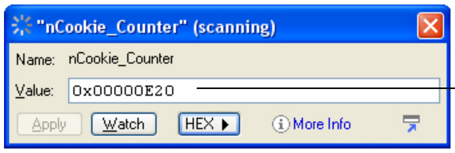
Name: nCookie_Counter
Value: 314

To view integers and integer tables in binary view, choose View > Binary Integer Display, or click the DEC button and choose Binary. The integers appear as shown below.


"nCookie_Counter" (scanning)
Apply
Watch
BIN
More Info
Binary view. All the bits cannot be shown at once.

Name: nCookie_Counter
Value: 00000000 00000000 00001110 00001000

To view integers and integer tables in hex view, click BIN in the dialog box, and then select Hexadecimal. Here's how the integers appear in hex:

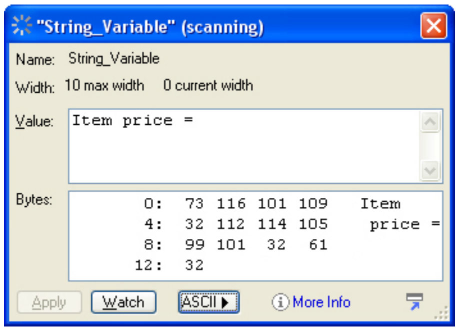

"nCookie_Counter" (scanning)
Apply
Watch
HEX
More Info
Hex view. The 0x before the number indicates that the number is in hex.

Name: nCookie_Counter
Value: 0x00000E20

To return to decimal view, click HEX and then select Decimal.

Setting Hex String View

You can also set strings to appear in hex notation. Here is the View Variable window showing a string in regular notation:

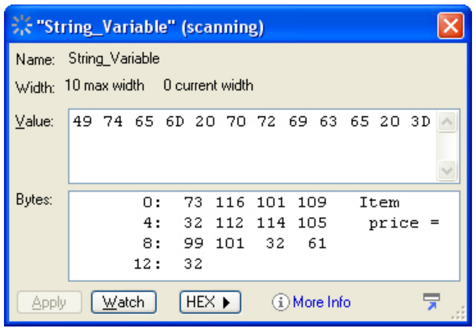


Name: String_Variable
Width: 10 max width 0 current width
Value: Item price =
Bytes:

```
 0:  73 116 101 109    Item
 4:  32 112 114 105     price =
 8:  99 101  32  61
12:  32
```

To change to hex notation, click the ASCII button on the dialog box.

Here's how the string appears in hex:



Name: String_Variable
Width: 10 max width 0 current width
Value: 49 74 65 6D 20 70 72 69 63 65 20 3D
Bytes:

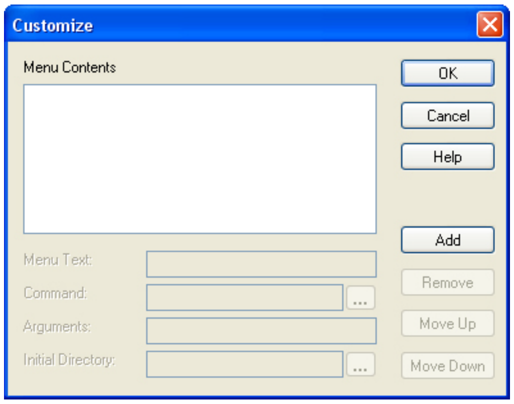
```
 0:  73 116 101 109   Item
 4:  32 112 114 105    price =
 8:  99 101  32  61
12:  32
```

Setting Up Applications to Launch from PAC Control

You may find it useful to launch other software applications directly from PAC Control. For example, you may want to launch PAC Display, Notepad, or the Calculator from PAC Control. You can set up PAC Control so that these applications appear in the Tools menu.

NOTE: If you launch an application from within PAC Control when that application is already running, a second instance of the application may open. It depends on the application; some check to see whether the application is already running, and some do not.

1. With PAC Control open, choose Tools > Customize.
   The Customize dialog box appears.



2. Click Add. In the Menu Text field, type the name of the application as you want it to appear in the Tools menu.
3. In the Command field, type the path for the application's executable file, or click the Browse button ... and navigate to the file.
4. (Optional) In the Arguments field, type any necessary command line parameters.
5. (Optional) In the Initial Directory field, type the directory the application should default to when it runs. For example, this is the directory the application would show when you open or save files.
6. Repeat the steps to add other applications. To change an application's position in the menu list, highlight it and click the Move Up or Move Down keys.
7. When you have finished adding applications, click OK. The PAC Control main window is displayed, and the applications you've added now appear in the Tools menu.

Using PAC Control's Command-Line Interface Tool

PAC Control's Command-Line Interface (CLI) tool was built for customers who want a fast and easy way to modify existing strategies without using the PAC Control graphical user interface.

CLI is a command-line program that runs in Microsoft's Command Prompt window. It can:

- Move a configured I/O module or I/O point to an empty position on the same or different I/O unit
- Add new variables
- Create a control engine download (.cdf) file

You can perform operations one at a time, or you can run multiple commands in a batch.

Responses to commands (and error codes) are logged to a file. For details, see "Checking CLI Command Output."

CLI Commands and Syntax

CLI commands are not case-sensitive. You can run them from either a normal or an Admin ("elevated privilege") Command Prompt window.

The syntax for a CLI command is:

`<CLI executable program> <strategy filename> /<operation> <arguments>`

where:

- **`<CLI executable program>`** is your PAC Control filename and extension:
  - For PAC Control Basic, the program is **`control.basic.exe`**
  - For PAC Control Professional, it is **`control.pro.exe`**
- **`<strategy filename>`** is the strategy to modify; for example: **`cookies.idb`**

  Note that you can perform operations on only one strategy at a time.

`/<operation> <arguments>` is the operation to perform and its required arguments (described on page 69). Multiple arguments are separated by semicolons.

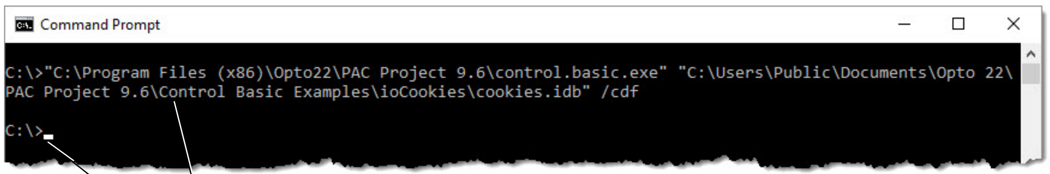
IMPORTANT: Paths, filenames, and arguments that contain a space must be enclosed in straight quotes (`" "`).

Depending on the folder you run CLI from, you may need to enter the path and filename for the CLI executable program, the strategy, or both.

For example, if PAC Control Basic is installed on your C:\ drive, the path and filename for the CLI executable program is:

```
"C:\Program Files (x86)\Opto22\PAC Project 9.6\control.basic.exe"
```

Because the path contains spaces, the path and filename have been enclosed in straight quotes.



```
C:\>"C:\Program Files (x86)\Opto22\PAC Project 9.6\control.basic.exe" "C:\Users\Public\Documents\Opto 22\
PAC Project 9.6\Control Basic Examples\ioCookies\cookies.idb" /cdf

C:\>_
```

Example of a CLI command and response.
Paths, filenames, and arguments that include spaces must be enclosed in straight quotes.
Note that the response is a new prompt.

When typing CLI commands:

- Type a space *after*:
  - The CLI executable program name.
  - The strategy filename.
  - The operator.
- Type a semicolon (;) *between* arguments. Don't put spaces before or after the semicolons.
- If your path, filename, or arguments include spaces, you must enclose them with straight quotes (**" "**).

The following table describes the CLI operations and arguments. Required characters are shown in **bold, red color** to make them easier to see.

To do this:	...use this operation:	...with these arguments:
Move a configured I/O module to an empty position	**/mm**	<source I/O unit>**;**<source module position>**;**<target I/O unit>**;** <target module position>
		where: • <source I/O unit> is the name of the I/O unit that contains the module[a] you want to move. • <source module position> is the current position number[a] of the module to move. • <target I/O unit> is the name of the I/O unit you want to move the module to. • <target module position> is the position number[a] where you want to move the module. Note that the position must be empty (that is, not configured with another module).
Example: `C:\>control.pro.exe cookies.idb /mm Mixed_IO_Unit;8;Sprinkler_Control;4`		
Move a configured I/O point to an empty position	**/mp**	<source I/O unit>**;**<source point name>**;**<target I/O unit>**;** <target module position>**;**<target point position>
		where: • <source I/O unit> is the name of the I/O unit that contains the point you want to move. • <source point name> is the name of the point to move. • <target I/O unit> is the name of the I/O unit you want to move the point to. • <target module position> is the position number[a] of the module where you want to move the point. • <target point position> is the point number *on the module*[a] where you want to move the point. Note that the position must be empty (that is, not configured with another point).
Example: `C:\>control.pro.exe cookies.idb /mp Mixed_IO_Unit;doRejectValve;Valve_Unit;2;0`		

To do this:	...use this operation:	...with these arguments: (Continued)
Add a variable *NOTE: This operation does not support pointer variables or pointer table variables.*	**/av**	<variable type>**;**<variable name>**;**<description>**;**<initialization type>**;**<initial value>[**;**<string width>][;<table length>]
		where: • <variable type> indicates the type of variable. Enter one of the following values.

Type this:	...if the variable is a:
INT32_VAR	32-bit integer variable
INT64_VAR	64-bit integer variable
FLOAT_VAR	Float variable
STR_VAR	String variable
DNTMR_VAR	Downtimer variable
UPTMR_VAR	Uptimer variable
CMHDL_VAR	Communication handle variable
INT32_TBL	32-bit table variable
INT64_TBL	64-bit table variable
FLOAT_TBL	Float table variable
STR_TBL	String table variable

• <variable name> is the name of the variable. The name must follow PAC Control conventions (for example, it cannot start with or contain spaces, it cannot start with a number, and so forth). Maximum length is 50 characters.

• <description> is a brief description of the variable (in straight quotes, if it contains spaces). To leave the description field blank, enter a space in straight quotes (like this: " "). Maximum length for the description is 127 characters.

• <initialization type> indicates how to initialize the variable. Enter one of the following values.

Type this:	...to:
ON_DOWNLOAD	Initialize the variable on strategy download.
ON_RUN	Initialize the variable when the strategy runs.
PERSISTENT	(Not applicable to timer variables.) This argument makes the variable persistent. For more information about persistent variables, see "Persistent Data" on page 256.

• <initial value> is the variable's initial value.
NOTE: Not applicable to timer variables. For timers, enter a space for this argument.

To do this:	...use this operation:	...with these arguments: (Continued)
		• [;<string width>][;<table length>] Applicable only to strings and table variables. – Strings: The string width. Example: **`15`** The maximum width you can give a string is 1024 characters. – String tables: The string width and table length. Example: **`15;99`** Note the semicolon between the numbers. The maximum width you can give a string is 1024 characters. The maximum table length is 1,000,000 elements. – Integer and float tables: The table length. Example: **99**
Example: `C:\>control.pro.exe cookies.idb /av INT32_VAR;nFlavor;"Cookie Flavor";ON_RUN;0`		
Perform a series of commands	**/f**	• <filename of a text file that contains CLI commands> This feature lets you issue one command to perform all the commands in a text file. Each command must follow CLI command syntax: <CLI executable program> <strategy filename> /<operation> <arguments> Each command must be on a separate line. (Don't use word wrapping in the file.) Each command is processed in order. Responses (and, if applicable, errors) are logged to a file. For details, see "Checking CLI Command Output."
Example: `C:\>control.pro.exe cookies.idb /f "C:\Documents\my batch commands.txt"`		
Compile a strategy into a .cdf file	**/cdf**	*This operation has no arguments.* The output is saved in the folder where PAC Control is installed.
Example: `C:\>control.pro.exe cookies.idb /cdf`		

[a]Remember that module numbers and point numbers are zero-based; the first position is number 0 (not 1).

The syntax to load a strategy in PAC Control's graphical user interface is:

`<CLI executable program> <strategy filename>`

Example: `C:\>control.pro.exe cookies.idb`

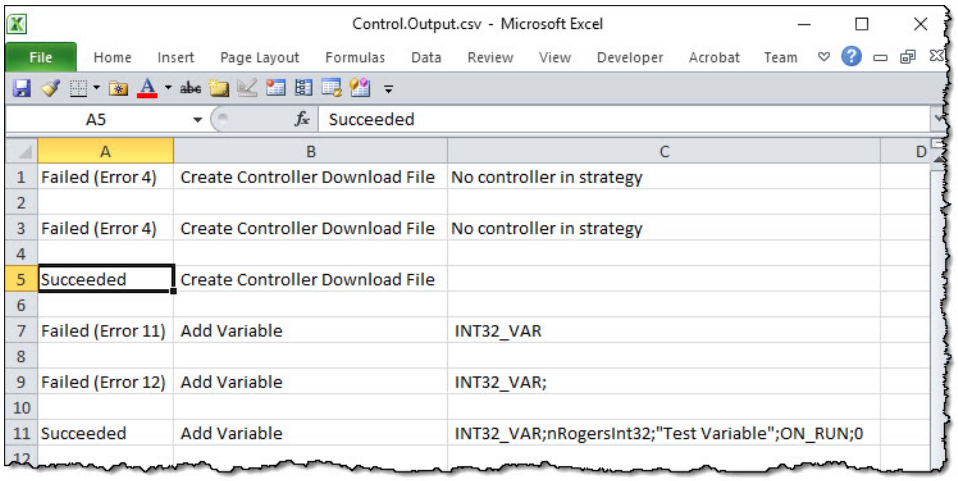
Checking CLI Command Output

For each CLI command you enter, the system logs a record in a Control.Output.csv file. (If the file already exists, new messages are appended to the file.) You can view the Control.Output.csv file to see whether or not a CLI command was successful. The file's .csv format lets you open the file in text editors (like Notepad), as well as in spreadsheet programs (like Excel).

- When a command is successful, CLI logs "Succeeded," and the name of the operation you performed.
- When a command fails, the message shows "Failed," followed by an error code (described on page 72), the name of the operation, and the arguments entered with the operation.

Control.Output.csv files are located in the same folder as the strategy used in the CLI command. For example, if the strategy in your command is located in the C:\MyStrategies folder, the Control.Output.csv is saved in C:\MyStrategies.

Example of a Control.Output.csv file opened in Microsoft Excel®.
In this example, Excel has parsed the data into columns.



	A	B	C
1	Failed (Error 4)	Create Controller Download File	No controller in strategy
2			
3	Failed (Error 4)	Create Controller Download File	No controller in strategy
4			
5	Succeeded	Create Controller Download File	
6			
7	Failed (Error 11)	Add Variable	INT32_VAR
8			
9	Failed (Error 12)	Add Variable	INT32_VAR;
10			
11	Succeeded	Add Variable	INT32_VAR;nRogersInt32;"Test Variable";ON_RUN;0

Error Codes

This table explains the error codes you might see in the Control.Ouput.csv log file.

Operation	Error code	Meaning
all	1	No operation was entered. Please enter an operation.
/f (Command file)	2	The text file containing CLI commands could not be opened.
all	3	CLI could not open the strategy.
all	4	The strategy does not have a configured controller
/cdf (Compile to .CDF file)	5	CLI could not create the .cdf file.
	6	CLI could not re-open the .cdf file.
	7	CLI could not create the .crn file.
	8	CLI could not open the intermediate file.
	9	CLI could not append to the .crn file.

Operation	Error code	Meaning (Continued)
/av (Add Variable)	10	The variable type argument is missing.
	11	The variable name argument is missing, or too long (maximum length is 50 characters), or is formatted incorrectly, or already in use, or is a programming “reserved word.” (Reserved words cannot be used as variable names.)
	12	The description argument is missing or too long. The maximum size for a description is 127 characters.
	13	The initialization kind argument is missing.
	14	The initial value argument is missing.
	15	(Applicable only to string and string table variables.) The string width argument is missing or too long. The maximum size you can give a string is 1024 characters.
	16	(Applicable only to table variables.) The table length argument is missing or too long. (Maximum table length is 1,000,000 elements.)
	17	Cannot create the variable.
	18	Cannot get an ID for the new variable.
	19	Cannot add the variable to the strategy.

Operation	Error code	Meaning (Continued)
/mm (Move Module)	20	The source I/O unit is missing. Please enter the name of the source I/O unit.
	21	The source module position is missing. Please enter a source module position number. Remember that module positions are zero-based (count up from 0 instead of from 1).
	22	The target I/O unit is missing. Please enter the name of the target I/O unit.
	23	The target module position is missing. Please enter a target module position number. Remember that module positions are zero-based (count up from 0 instead of from 1).
	24	The source I/O unit is not in the strategy. Check the name and re-enter.
	25	The source module position is out-of-range. Check the number and re-enter. Remember that module positions are zero-based (count up from 0 instead of from 1).
	26	The target I/O unit is not in the strategy. Check the name and re-enter.
	27	The target module position is out-of-range. Check the number and re-enter. Remember that module positions are zero-based (count up from 0 instead of from 1).
	28	A module already exists at the target module position. Enter an empty module position.
	29	No module is defined at the source module position. Check the number and re-enter.
	30	One or more points on the source module are referenced by a PID loop, and the target I/O unit is not the source I/O unit. To move the module, remove the PID loops, or make sure the source module and the target module are the same.
	31	The target module cannot be used because it's not compatible with the source module.
	32	The source I/O unit cannot be verified.
	33	The target I/O unit cannot be verified.
	34	The source module position is out-of-range. Check the number and re-enter. Remember that module positions are zero-based (count up from 0 instead of from 1)
	35	The target module position is out-of-range. Check the number and re-enter. Remember that module positions are zero-based (count up from 0 instead of from 1)
	36	The points on the source module are not allowed on the target I/O unit.

Operation	Error code	Meaning (Continued)
/mp (Move Point)	40	The source I/O unit is missing. Please enter the name of the source I/O unit.
	41	The source point name is missing. Please enter the name of the source point.
	42	The target I/O unit is missing. Please enter the name of the target I/O unit.
	43	The target module position is missing. Please enter a target module position number. Remember that module positions are zero-based (count up from 0 instead of from 1).
	44	The target point position is missing. Please enter a target position number. Remember that point positions are zero-based (count up from 0 instead of from 1) and are the position on the module (not the absolute point position).
	45	The source I/O unit is not in the strategy. Check the name and re-enter.
	46	The source point name is not in the strategy. Check the name and re-enter.
	47	The target I/O unit is not in the strategy. Check the name and re-enter.
	48	The target module position is out-of-range. Check the number and re-enter. Remember that module positions are zero-based (count up from 0 instead of from 1).
	49	The target point position is out-of-range. Check the number and re-enter. Remember that point positions are zero-based (count up from 0 instead of from 1) and are the position on the module (not the absolute point position).
	50	A point already exists at the target point position. Enter an empty point position.
	51	The point on the source module is referenced by a PID loop, and the target I/O unit is not the source I/O unit. To move the point, remove the PID loop, or make sure the source module and the target module are the same.
	52	The target point cannot be used because it's not compatible with the source point.
	53	The source I/O unit cannot be verified.
	54	The target I/O unit cannot be verified.
	55	The source point name cannot be verified.
	56	The target module position is out-of-range. Check the number and re-enter. Remember that module positions are zero-based (count up from 0 instead of from 1).
	57	The target point position is out-of-range. Check the number and re-enter. Remember that point positions are zero-based (count up from 0 instead of from 1) and are the position on the module (not the absolute point position).
	58	The point on the source module is not allowed on the target I/O unit.
	59	Quadrature modules must be moved to an even numbered channel.
	60	Quadrature modules require two adjacent channels. The second channel is occupied by another point.

Keyboard and Mouse Shortcuts

Use the following keyboard and mouse shortcuts to view and navigate a chart.

Action	Shortcut
Center on Block 0	0 or Numeric Keypad 0
Center on Block...	J
Block Preview	Middle-click on a block
Move to Top Left	Numeric Keypad 7
Move to Top	Numeric Keypad 8
Move to Top Right	Numeric Keypad 9
Move to Left	Numeric Keypad 4
Move to Center	Numeric Keypad 5
Move to Right	Numeric Keypad 6

Action	Shortcut
Move to Bottom Left	Numeric Keypad 1
Move to Bottom	Numeric Keypad 2
Move to Bottom Right	Numeric Keypad 3
Zoom In	+
Zoom Out	–
Zoom to Fit and Center	F
Zoom to 100%	Z
Pan	Space + Left Mouse Button

Online Help

To open Help within PAC Control, click Help > Help Topics in the PAC Control menu bar, or click the Help button in any dialog box. Help buttons in dialog boxes are context-sensitive and provide help specifically on that dialog box. Buttons labeled "Command Help" give specific information on the command (instruction) you are currently using. (The command must be highlighted to open Command Help.)

For brief explanations of buttons, move your mouse over the button and look in the status bar.

In Action and Condition Instructions dialog boxes, let your mouse rest for a second on a command, and you'll see a list of the variable types used in that command. (To show or hide the variable types list, in Configure mode, click PAC Control Options > Show/Hide Instruction Type Information.)

To open copies of PAC Control manuals and quick reference cards, click Help > Manuals. You will need Adobe Acrobat Reader to open these files. You can also find information, documents, and support on the Opto 22 website.

4: Designing Your Strategy

OPTO 22

Introduction

This chapter introduces you to PAC Control programming—how to design a PAC Control strategy to control your automated process. For additional important information on using PAC Control commands (instructions) to program your strategy effectively, see page 277 and the individual command information in the *PAC Control Command Reference*.

NOTE: Because our focus in PAC Project is on the SNAP PAC System, PAC Control initially shows only SNAP PAC I/O units and the commands used with them. When you're using the SNAP PAC system only, hiding legacy I/O units and commands makes it simpler and less confusing to build your strategy. However, the legacy capabilities are still there and can be made visible in a specific strategy as needed. For information, see "Legacy Options" on page 232.

In this Chapter

Steps to Design....77
Basic Rules....84
Instruction Examples....87
Optimizing Throughput....98

Steps to Design

How do you get from your real-world control problem to a working PAC Control strategy that solves it? Here's an outline of the basic approach; we'll fill in the details on the following pages.

First, solve the problem.

- Define the problem.
  - What am I trying to do?
  - What inputs and data do I have to work with?
  - What do the outputs have to be?
  - How many times does the process have to be repeated?

- Design a logical sequence of steps to solve the problem.
  - Break down the larger process task into sub-process tasks.
  - Break down the sub-process tasks into detailed steps.
- Test the steps.

Next, build the strategy.

- Configure hardware.
  - Control Engines
  - I/O units
  - I/O points
- Determine necessary variables and configure them.
- Create charts, one for each sub-process, and add instructions.
- Compile and debug each chart.
- Compile and debug the whole strategy.

Finally, use and improve the strategy.

Now let's take a look at each of these steps in order.

Solving the Problem

You can avoid a lot of extra time and rework if you define and solve your problem before you ever start building a flowchart in PAC Control.

Defining the Problem

Suppose, for example, you want to automate a simple lawn sprinkler system. Start by asking yourself (or others) questions about the control process you're trying to do.

What are you trying to do? Start with the big picture and work down to the smaller details.

- Big picture: I'm trying to control this sprinkler system.
- Smaller details:
  - The sprinklers should turn on every Sunday and every Wednesday.
  - They should not turn on if it's raining.
  - They should start at six o'clock in the morning.
  - They need to run for 15 minutes.

What inputs and data do you have to work with? List all the inputs. Describe the data they provide. Check for any inputs or data you need but don't have.

Input	Data the input provides
Hygrometer	Humidity (percentage from 0–100%)
Day	Day of the week (Sunday through Saturday)
Time	Hour of the day (24-hour clock)

Input	Data the input provides
Sprinkler status	Whether sprinklers are currently running (on or off)
Input from timer	Length of time sprinklers have been on (in minutes)

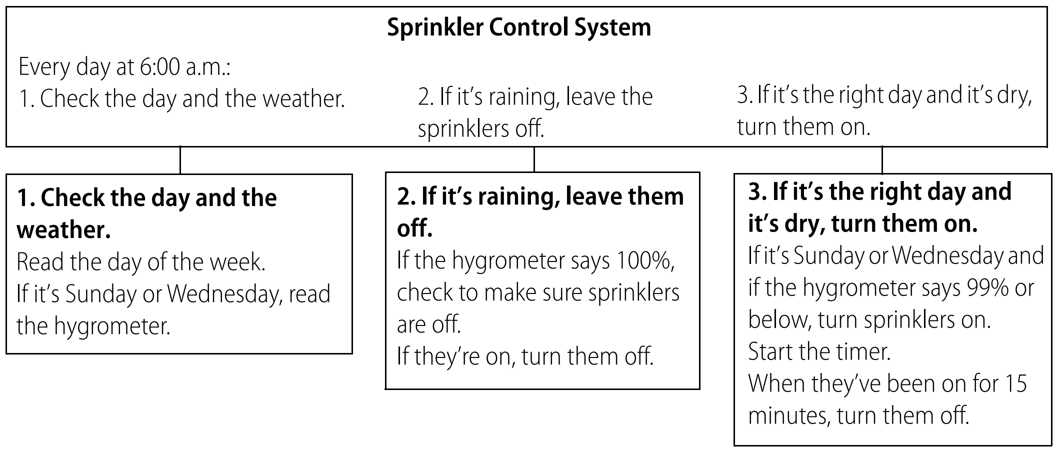
What do the outputs need to be? List all the outputs. Describe the results that are needed. Check for any outputs you need but don't have.

Output	What it does
Sprinkler switch	Turns sprinklers on or off
Timer control	Sets timer

How many times does the process have to be repeated? Our example is a simple sprinkler system in which the process happens twice a week on certain days. In a more complex system, however, you might have one section of the sprinklers turn on, followed by other sections, repeating the process several times in different areas and then repeating the whole pattern on certain days.

Designing a Logical Sequence of Steps to Solve the Problem

Now that you've determined what you're trying to do and what your inputs, data, and outputs look like, you're ready to outline the steps needed to solve the control problem. Think about how you would do the job if you were doing it by hand. Again, work from the top down, starting with big steps and then breaking those big steps into smaller steps.



Sprinkler Control System

Every day at 6:00 a.m.:
1. Check the day and the weather.
2. If it's raining, leave the sprinklers off.
3. If it's the right day and it's dry, turn them on.

1. Check the day and the weather.
Read the day of the week.
If it's Sunday or Wednesday, read the hygrometer.

2. If it's raining, leave them off.
If the hygrometer says 100%, check to make sure sprinklers are off.
If they're on, turn them off.

3. If it's the right day and it's dry, turn them on.
If it's Sunday or Wednesday and if the hygrometer says 99% or below, turn sprinklers on.
Start the timer.
When they've been on for 15 minutes, turn them off.

If a human being were doing this job, this level of instruction would probably be just fine. With a computer, however, the instructions need more detail. For example, the first instruction, "Every day at 6:00 a.m.," would have to be expanded more like this:

a. Read the hour.

b. If the hour equals six, go on to the next step.

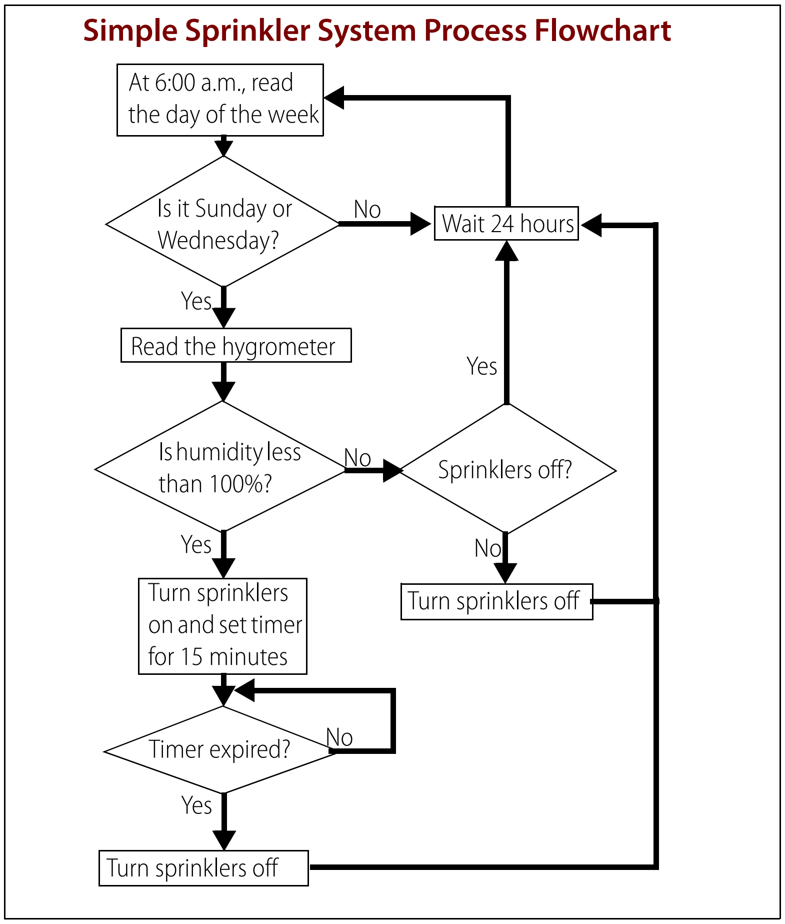
c. If the hour does not equal six, go back to step a.

Testing the Steps

Now that you have your steps in place, run through each sub-process in your mind to see if anything is missing. Is there an answer for every “if”? Are all the possibilities taken into account?

It may help you to draw out your control process as a flowchart. Often it is easier to see decision points and responses to decisions in a flowchart. You’re still working at the human level at this point; just be aware that the computer may require more details.

Here is a possible flowchart for the simple sprinkler control problem.


Simple Sprinkler System Process Flowchart
At 6:00 a.m., read the day of the week
Is it Sunday or Wednesday?
No
Wait 24 hours
Yes
Read the hygrometer
Yes
Is humidity less than 100%?
No
Sprinklers off?
Yes
No
Turn sprinklers on and set timer for 15 minutes
Turn sprinklers off
Timer expired?
No
Yes
Turn sprinklers off

Building the Strategy

Once you have the control problem solved in detail, you can begin building the strategy in PAC Control. Now you’ll add all the logical details the computer needs.

Configuring Hardware

The first step in building your strategy is to configure your control engine, I/O units, and I/O points. (For more information and step-by-step procedures, see Chapter 5: Working with Control Engines and Chapter 6: Working with I/O) You can add more I/O units and points later if needed, but it saves time to configure them now.

When you solved this control problem, some of the inputs and outputs you defined were physical I/O, such as the hygrometer and the switch to turn on the sprinklers. You configure these physical inputs and outputs as the I/O points.

Here are the I/O points you might configure for the simple sprinkler system:

Physical I/O	Type	I/O Point Name
Hygrometer	Analog Input	Hygrometer
Sprinkler status	Digital Input	Sprinkler_Status
Sprinkler switch	Digital Output	Sprinkler_Switch

Determining and Configuring Variables

Some of the inputs and outputs you defined when you solved the problem were not physical I/O, but information. For example, the day of the week is not a physical input; it's a piece of information you can get using a command in PAC Control. These inputs and outputs become variables.

You also need variables to store the information that input points give you. And you may need variables for manipulating information.

To determine the variables you need, think about the pieces of information your process requires. Then look at the PAC Control commands in the *PAC Control Command Reference*. Find the commands you need and check them to see what types of variables each command requires.

For example, you need to know what day of the week it is. Will the control engine give you this information as a string (for example, "Wednesday") or in some other format? When you check the command Get Day of Week, you discover that the day is returned as a number (Wednesday = 3), so you set up this variable as a numeric integer.

Here are some variables you may need. (See Chapter 9: Using Variables and Commands for more information and step-by-step procedures to add variables.) You can change them or add other variables later if necessary.

Variable Needed	Type	Name in PAC Control
Hour of the day	Numeric (32-bit integer)	Time_of_Day
Day of the week	Numeric (32-bit integer)	Day_of_the_Week
Humidity	Numeric (float)	Humidity
Down timer	Timer	Timer

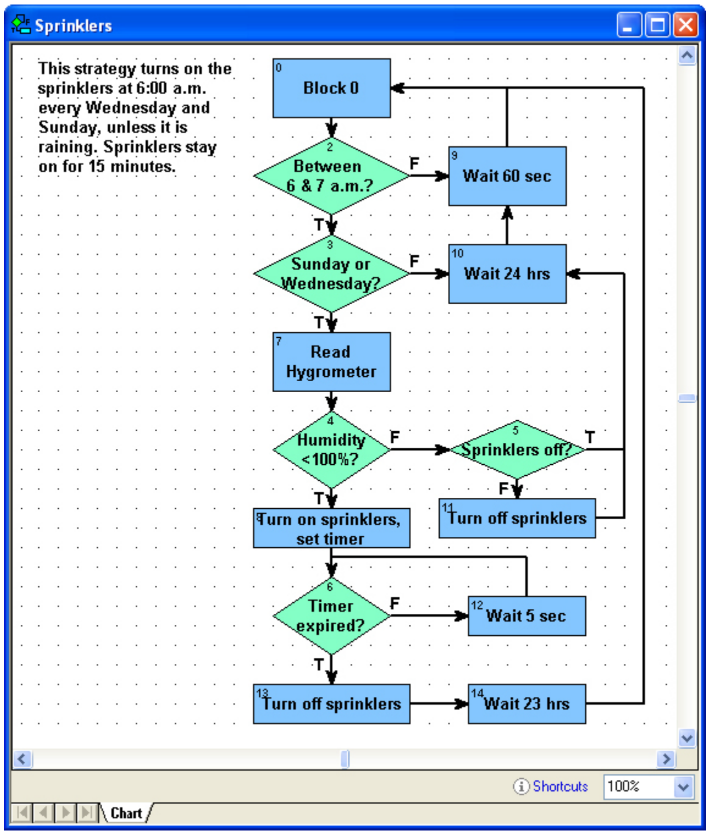
Creating PAC Control Charts and Adding Instructions

If you have already drawn the process in a flowchart, this step will go faster. PAC Control is based on flowcharts because they are a natural way to show a control process. And the block names and instructions you add to charts—just like the names of variables—are in normal, everyday language. (For more information and step-by-step procedures, see Chapter 8: Working with Flowcharts and Chapter 9: Using Variables and Commands).

The difference between the flowchart you drew before and the chart you create in PAC Control is that the flowchart was designed for human beings, but the PAC Control chart must be more detailed so the computer can follow it.

Because the chart is more detailed—and because most control processes are far more complex than our sprinkler system—you will usually create multiple charts for a PAC Control strategy. Break the process down into logical chunks, or modules, and then create a chart for each module. A modular design makes it easier to test and to update your strategy.

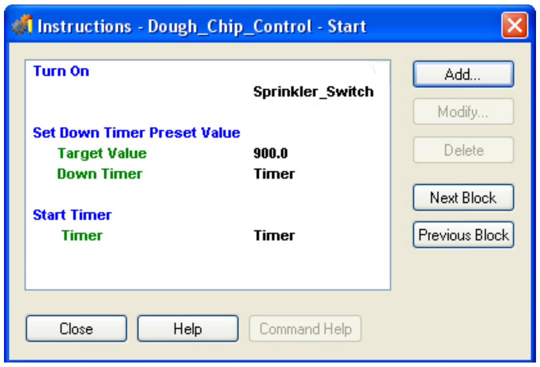
Even for a simple process like our sprinkler control, there is no single "correct" way to solve the control problem and build the strategy. Instead, there are many possible ways. The following chart shows one example:



One difference between this chart and the human-level chart on page 80 is that several delays have been added to the chart. There is one before rechecking the hour of the day, one before rechecking the day of the week, one before rechecking whether the timer has expired, and finally one before starting the flowchart over again.

This sprinkler system is not time-critical. If the sprinklers turn on at a little past 6:00 a.m. or the grass is watered an extra few seconds, it doesn't matter. Delays give the control engine time to do other things. (For more information on delays and using control engine time effectively, see page 98.)

Each block in the chart contains the PAC Control commands (instructions) that do the work in the system. For example, here are the instructions for the block "Turn on sprinklers, set timer":



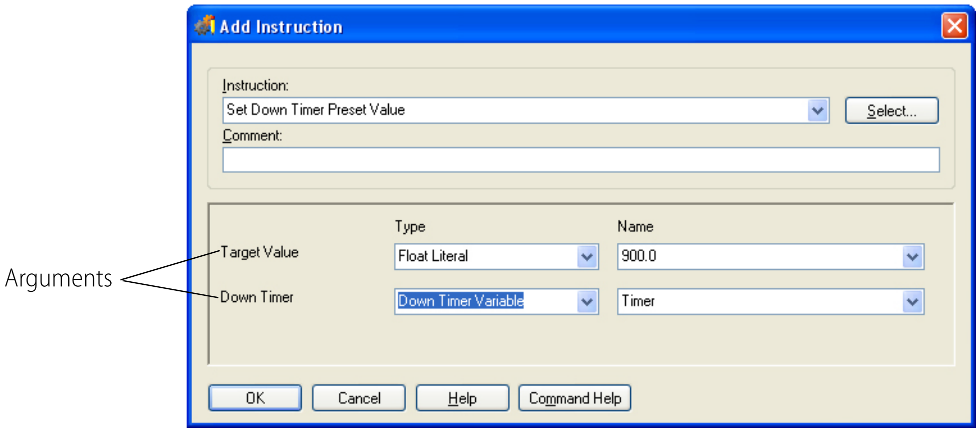
Turn On
Sprinkler_Switch

Set Down Timer Preset Value
Target Value 900.0
Down Timer Timer

Start Timer
Timer Timer

As you create charts and enter instructions, keep referring to the *PAC Control Command Reference*. The *Command Reference* describes the "arguments" for each command, which are pieces of information you must enter in the instruction. You can also press F1 to open online help for commands, which gives you the same information as in the printed *Command Reference*.

For example, one of the commands shown above, Set Down Timer Preset Value, has two arguments. The *Command Reference* and online help tell you that the first argument—the target value, or the value from which the timer counts down—can be a float variable or a float literal (entered in either decimal or hexadecimal form), and that the second argument is the name of the down timer variable. You need this information when you enter the instruction in the Add Instruction dialog box.



Instruction: Set Down Timer Preset Value
Comment:

	Type	Name
Target Value	Float Literal	900.0
Down Timer	Down Timer Variable	Timer

The sample flowchart we've shown uses standard PAC Control commands only. If you have a programming background, you may wish to incorporate OptoScript blocks in your flowchart. OptoScript is a procedural language that can simplify some programming tasks, including string handling, math calculations, and complex loops and conditions. See Chapter 11: Using OptoScript for more information.

Compiling and Debugging the Strategy

When all charts are completed and their instructions added, the next step is to compile and debug the strategy. But first, check your hardware. Check cabling and connections, and make sure the actual I/O matches the configured I/O in the strategy.

Now compile and debug the strategy. (See Chapter 7: Working with Strategies for more information and step-by-step procedures. If you have problems, see Appendix A: Troubleshooting.Opto 22 form 1700, the *PAC Control User's Guide*.) If you have multiple charts, debug each one separately and then debug the strategy as a whole. It is easier to find errors in one flowchart than in a group of interrelated ones.

Make sure the connections between charts are accurate. For example, is a chart started at the correct block in another chart?

Use the debugging tools discussed in Chapter 6 to find errors by stepping through a chart and setting breakpoints to discover which block or line contains the problem. And if you've tried everything and it still doesn't work, contact Opto 22 Product Support. (See page 4.)

Using and Improving the Strategy

Any strategy is one of several possible ways to solve a control problem. One way may be more efficient under some circumstances; another way may be better for others. As you use the strategy over time, as control needs change, and as you become more knowledgeable about PAC Control, you'll find ways to make the strategy better.

Basic Rules

The sprinkler strategy we just created is a simple example. This section gives basic rules to keep in mind when you're solving more complex control problems.

When you create a new PAC Control strategy, a Powerup chart is automatically included. You add all the other charts you need, and the total number of charts in the strategy is limited only by the control engine's memory. Be aware, however, that the maximum number of charts *that can be running at any one time* is based on the control engine you are using:

- 16 charts on a SNAP PAC R-series or SNAP-LCE controller
- 32 charts on a SNAP PAC S-series controller
- 64 charts on a SoftPAC controller
- 8 charts on SNAP Ultimate I/O

Program logic moves through a flowchart in one of two ways: flow-through or loop.

- A chart with **flow-through logic** performs a set of specific commands and then stops. It has a beginning and an end, and at the end is a condition or action block that has no exit.

  Subroutines, the Powerup chart, and any other chart that does not need to run continuously (such as an interrupt chart) should always have flow-through logic. In complex strategies, be careful when using delays and condition looping (waiting for something to occur) in charts with flow-through logic.
- A chart with **loop logic** has no end. It loops through a set of actions and conditions continuously. A loop-logic chart can have several paths through which the direction of the logic may flow, depending on certain criteria. Use loop logic for a chart that needs to run continuously. (The simple sprinkler chart has loop logic; it runs continuously.)

Chart Guidelines

As you design your strategy, put each sub-process of your overall control problem into a separate chart within the strategy. As noted above, on a SNAP Ultimate brain, you can have a maximum of eight charts (plus one host task) running at once. On a SNAP PAC R-series or SNAP-LCE controller, a maximum of 16 charts (plus one host task) can run at once. On a SNAP PAC S-series controller, 32 charts (plus one host task) can run at once. On a SoftPAC controller, 64 charts (plus one host task) can run at once. (For more information, see "Optimizing Throughput" on page 98.)

In general, follow these chart guidelines:

- Use the Powerup chart just to set initial values for variables, perform setup commands, and start the main charts. Use flow-through logic so the Powerup chart will stop as soon as the other charts begin.
- Create a few charts to monitor essential or time-critical pieces of your process, such as emergency stops on dangerous equipment or I/O that must be monitored continuously. These charts should use loop logic so they are constantly running.
- Create an Interrupt chart only for critical event/reactions that send interrupts through direct lines from an I/O unit to the controller. See "Using an Interrupt Chart for Event/Reactions (mistic only)" on page 88 and "Using an Interrupt Chart with the Generating Interrupt? Command (mistic Only)" on page 333.
- If a set of operations is used multiple times in your strategy or is used in more than one strategy, you can put that logic in a subroutine and call the subroutine when needed. Calling a subroutine doesn't require as much time as calling a chart or starting a new chart. In addition, a subroutine starts as soon as it is called in the context of the chart that is calling the subroutine. A Start Chart command is simply placed in the task queue and started in turn. See Chapter 12: Using Subroutines for more information.
- Use the text tool to type a descriptive title in each chart and add any necessary explanations. Keep blocks far enough apart to easily see the flow, and if possible, design the chart so its entire flow can be seen within one window. If a chart becomes too complex, split it into smaller charts.
- When you use OptoScript code, remember that it is not self-documenting. Be sure to add comments so that you or others can easily see what the code is doing. (Similar comments are also useful in other PAC Control flowchart blocks.) Use OptoScript blocks within a flowchart for operations they make easier, such as string handling and math calculations, but keep the logic and purpose of the strategy clear in the flowchart.
- Within a chart, leave Block 0 empty. This makes it easy to insert logic at the beginning of the chart by adding a block between Block 0 and the next block. If a block contains more instructions than can be easily traced and debugged, break it down into two or more sequential blocks. Never place another block in the flowchart before Block 0. Charts always start with Block 0, which cannot be deleted.

Naming Conventions

To save trouble in the long run, it's wise to establish naming conventions for charts, blocks, variables, I/O units and points, control engines, subroutines, and so on. If you name these elements in a consistent way from the beginning, it is easier to find them in lists, to see related elements, and to know their functions without opening them.

Since PAC Control automatically alphabetizes these names, you can plan naming schemes to place similar items together in the way you expect to use them.

Names have a maximum length of 50 characters. They can be all upper-case characters, or they can be mixed case.

In chart names, for example, you might include a few letters indicating whether the chart monitors critical tasks, is a master chart calling others, or is periodically called:

- Mntr_Tank_Leak (constantly running chart that monitors a critical situation)
- Mstr_Conveyor_Process (master chart that calls others for a sub-process)
- Call_Message_Display (chart called by a master chart for a specific purpose)

Block names should tell what the block does, so you can understand a chart without needing additional explanations. Block names should describe the purpose of the instructions within them. Use a question for the name of a condition block, with a true exit reflecting the answer yes.

- Read Thermometer (action block)
- Temperature Too High? (condition block)
- Turn On Pump 1 *or* Pump 1 On (action block)
- Pump On? (condition block)

Variable names can include Hungarian notation to indicate the type of variable, since the variable type is not always apparent in its name. For example a variable named *Month* might be either a string or an integer. However, using Hungarian notation, *sMonth* indicates a string, and *nMonth* indicates an integer.

Another good practice is to identify variables used in a single chart differently than variables used by more than one chart or process, especially if you have a large chart and want to separate these types of variables. For variables used only in a table, you can include the table name in the variable name. For example, you might use the name Fail_Count_in_Config_Values for a variable to be used in a table named Config_Values.

The following table shows suggested notation for use in PAC Control:

Variable type	Letter
integer 32 variable	n
integer 32 variable used as Boolean	b
integer 32 table	nt
integer 64 variable	nn
integer 64 table	nnt
float variable	f
float table	ft
down timer	dt
up timer	ut
string variable	s
string table	st

Variable type	Letter
pointer variable	p
pointer table	pt
digital I/O unit	dio
mixed I/O unit	mio
analog input point	ai
analog output point	ao
digital input point	di
digital output point	do
chart	cht
communication handle	cmh

In I/O point names, you may want to indicate the point's function, the state of the device when the point is active, its location, or a schematic reference. You can abbreviate names of familiar devices and write out less familiar names. Include the information you need in the order in which it will be most useful to you:

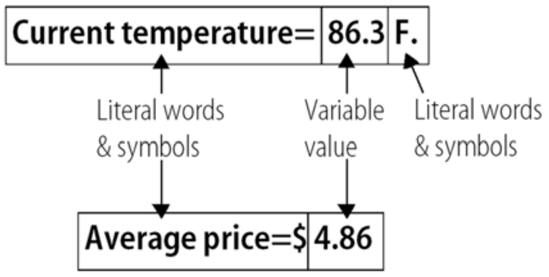
- Heater_Switch (You have only one heater.)
- Htr6_Switch_SW23B (You have many heaters, and the schematic reference is needed.)
- Cnvyr_Speed_Encoder_BldgA (You want all conveyors to appear together.)
- BldgA_Cnvyr_Speed_Encoder (You want all points in Building A to appear together.)

Instruction Examples

This section includes examples of common instructions you may want to use. See Chapter 10: Programming with Commands. For additional information on programming with PAC Control. If you need to use math calculations, strings, or complex loops and conditions in your strategy, also see Chapter 11: Using OptoScript for examples of OptoScript. OptoScript is a procedural language within PAC Control that can make programming easier, especially if you are an experienced programmer.

Creating Messages to Display On Screen

You may need to create messages to display on screen, for example to give data to operators. Typically these messages consist of some specific, literal words and symbols followed by a variable value and maybe some additional literal words or symbols.

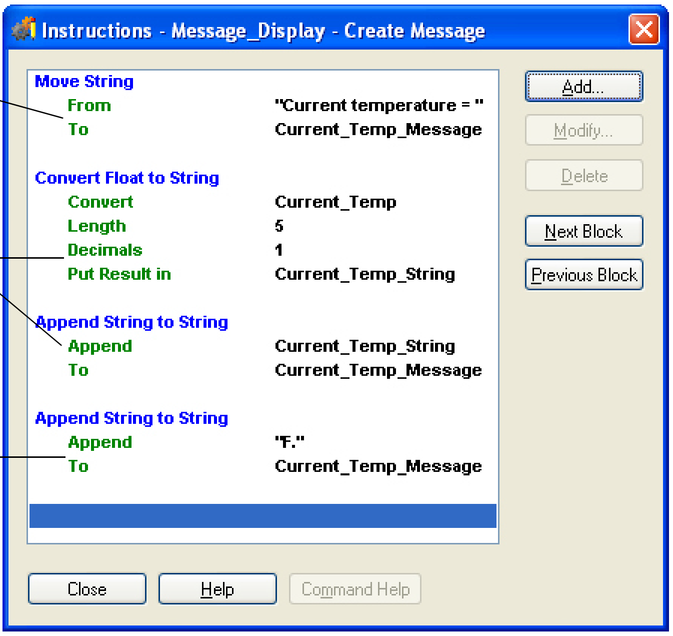


You can create the message in a single block in your PAC Control chart, like this:

1. To enter the literal text, use the command Move String. Remember to include spaces where you want them to appear, such as after the equal sign.

2. Add the variable value by converting it to a string and appending it to the string you just created.

3. If needed, append another literal string.



Move String	
From	"Current temperature = "
To	Current_Temp_Message
Convert Float to String	
Convert	Current_Temp
Length	5
Decimals	1
Put Result in	Current_Temp_String
Append String to String	
Append	Current_Temp_String
To	Current_Temp_Message
Append String to String	
Append	"F."
To	Current_Temp_Message

Alternatively, you can create the message by placing the following OptoScript code within a single OptoScript block:

```
FloatToString(Current_Temp, 5, 1, Current_Temp_String);
Current_Temp_Message = "Current temperature = " + Current_Temp_String +
"F.";
```

OptoScript can be more efficient for string handling than standard PAC Control commands. See Chapter 11: Using OptoScript for more information.

Using an Interrupt Chart for Event/Reactions (*mistic* only)

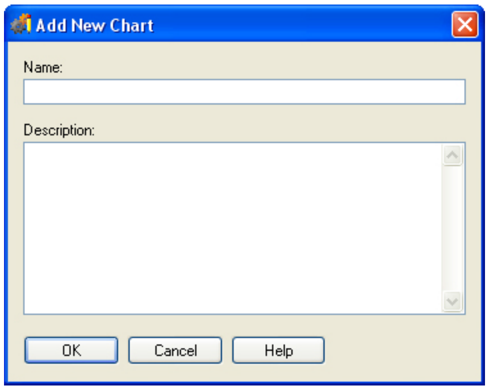
An Interrupt chart is primarily intended to be used with *mistic* serial brains and event/reactions. The purpose of an Interrupt chart is to handle interrupts from *mistic* serial brains specially wired to a SNAP PAC S-series controller. These interrupts are generally used for critical events requiring immediate action. If you create a chart named Interrupt, PAC Control will compile the chart so that it is used correctly.

If an interrupt chart exists in the strategy, it will be started and suspended automatically when the strategy is run. When a *mistic* interrupt is received by the controller, the Interrupt chart will execute.

To create an Interrupt chart:

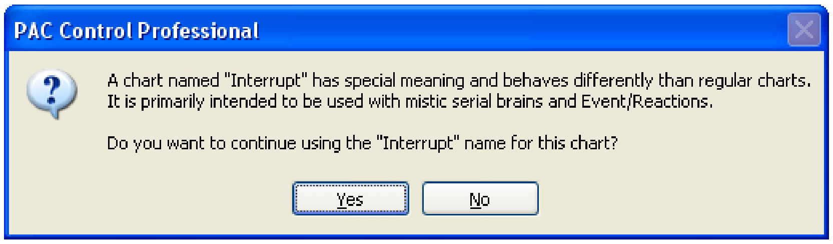
1. With your strategy open and in Configure mode, select Chart > New, or right-click the Charts folder on the Strategy Tree and select New from the pop-up menu.

The Add New Chart dialog box appears.



2. Enter the name Interrupt (which is not case sensitive), then enter a description if you want and click OK.

A message appears saying that an Interrupt chart has a special meaning and behavior.


PAC Control Professional

A chart named "Interrupt" has special meaning and behaves differently than regular charts. It is primarily intended to be used with mistic serial brains and Event/Reactions.

Do you want to continue using the "Interrupt" name for this chart?

3. Click Yes.

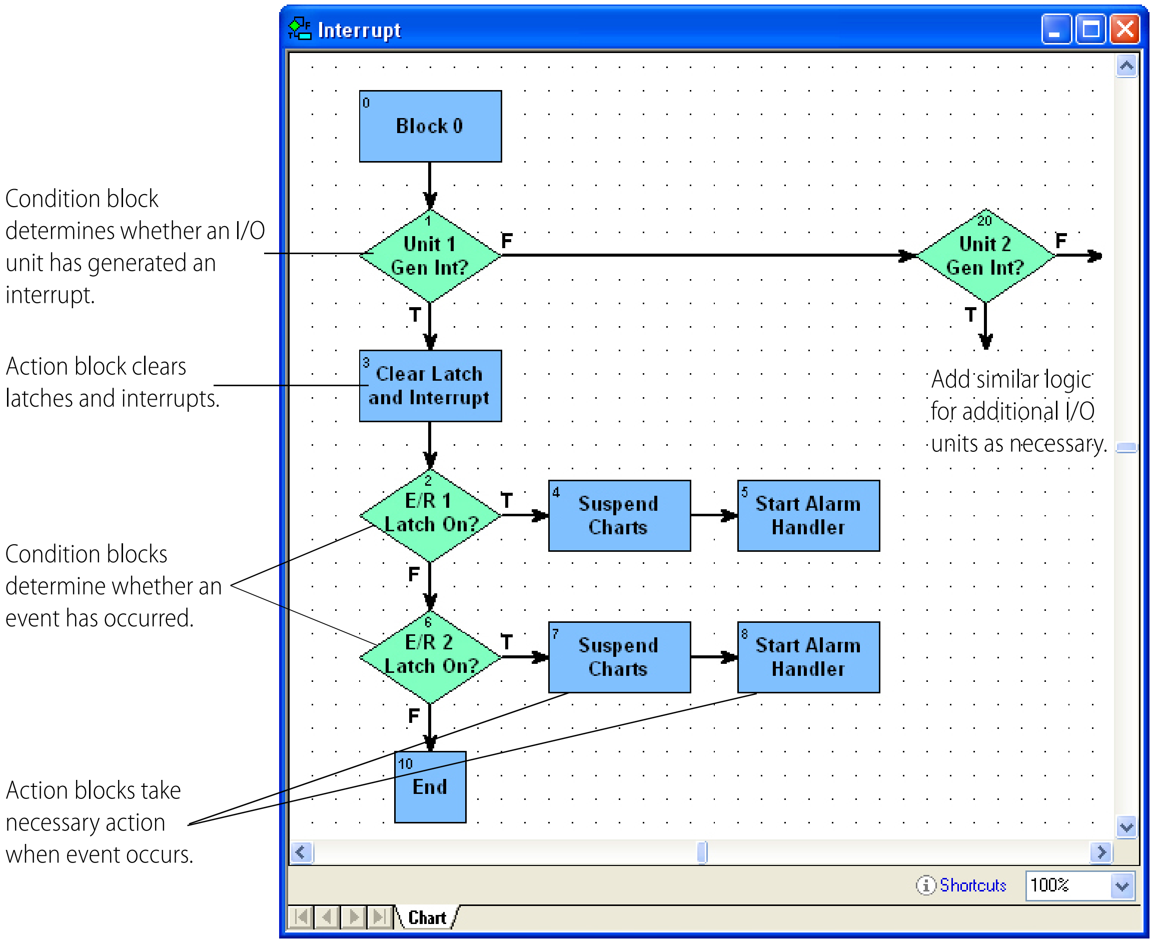
A chart named Interrupt is created.

Adding Logic to an Interrupt Chart

To use the new Interrupt chart for event/reactions, you need to create three basic blocks:

- Condition blocks that determine which I/O unit and event/reaction generated the interrupt
- Action block(s) to clear latches and interrupts, so the Interrupt chart won't keep on running
- Action block(s) to take whatever action is necessary when the event/reaction occurs, for example to start an alarm chart or stop a process.

Here's how a simple interrupt chart might look:


Interrupt
0 Block 0
1 Unit 1 Gen Int?
F
T
20 Unit 2 Gen Int?
F
T
3 Clear Latch and Interrupt
2 E/R 1 Latch On?
T
F
4 Suspend Charts
5 Start Alarm Handler
6 E/R 2 Latch On?
T
F
7 Suspend Charts
8 Start Alarm Handler
10 End
Shortcuts
100%
Chart
Condition block determines whether an I/O unit has generated an interrupt.
Action block clears latches and interrupts.
Condition blocks determine whether an event has occurred.
Action blocks take necessary action when event occurs.
Add similar logic for additional I/O units as necessary.

Notice that the Interrupt chart uses flow-through logic. When it finishes (and there are no interrupts), the interrupt chart goes back to Block 0, becomes suspended, and waits for the next interrupt.

See also:

- For information on using interrupt commands, see "Using an Interrupt Chart with the Generating Interrupt? Command (mistic Only)" on page 333.
- For more information about time in the task queue, see "Optimizing Throughput" on page 98.
- For more information about event/reactions, see "Event/Reaction Commands" on page 332.

Error Handling

Every strategy should have a way to handle errors. It's important to check values and errors that are returned from the commands in your strategy. Communication to an I/O unit, for example, will be automatically disabled if a command sends it variable values that are clearly wrong, such as a memory map address in an incorrect format.

Counting

Most standard digital inputs can be used as counters to count the number of times the input changes from off to on. The availability of counters depends on the brain's capabilities, and the speed of counters depends on the module.

For information on brain capabilities, see Opto 22 form 1689, the *SNAP PAC Brains Data Sheet*.

For more information on module counting speed, see the module's specifications on the Opto 22 website, www.opto22.com. On the website, select Products > SNAP PAC System > Brains and I/O > Digital I/O.

NOTE: You can use counters on high-density modules with SNAP-PAC-R1 and SNAP-PAC-R1-B controllers, and SNAP-PAC-EB1 and SNAP-PAC-SB1 brains with firmware 8.1 or newer.

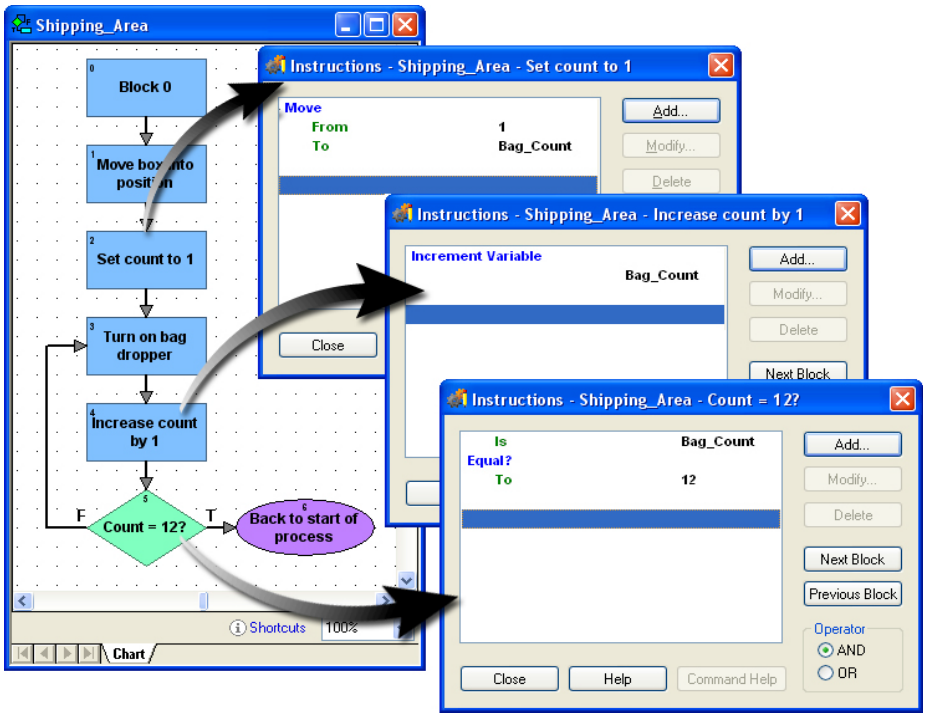
Before using a counter, you must configure the point as a counter. (See "Adding a Digital I/O Point" on page 136, or use PAC Manager.) Counters on Ethernet and SNAP PAC I/O units do not need to be started, and they cannot be stopped. Therefore, do not use the Start Counter and Stop Counter commands with SNAP PAC I/O units. However, you can use Get Counter and Get & Clear Counter.

Counters on *mistic* I/O units must first be started using the Start Counter command. In addition to Start Counter, you can use the following commands on *mistic* I/O units: Stop Counter, Get Counter, Get & Clear Counter, and Clear Counter.

Using a Count Variable for Repetitive Actions

A numeric variable for counting is useful when the same action needs to be repeated a specific number of times. For example, suppose your process includes filling a packing box with a dozen bags of cookies. You can use a count variable to track how many bags are dropped into the box.

Here's part of a chart showing how you would use a count variable in this way:



Move
From 1
To Bag_Count

Increment Variable
Bag_Count

Is Bag_Count
Equal?
To 12

Operator: [✓ AND] OR

If you are an experienced programmer, you'll notice that this example is a `for` loop. You can use the following OptoScript code within a single OptoScript block to accomplish the same result:

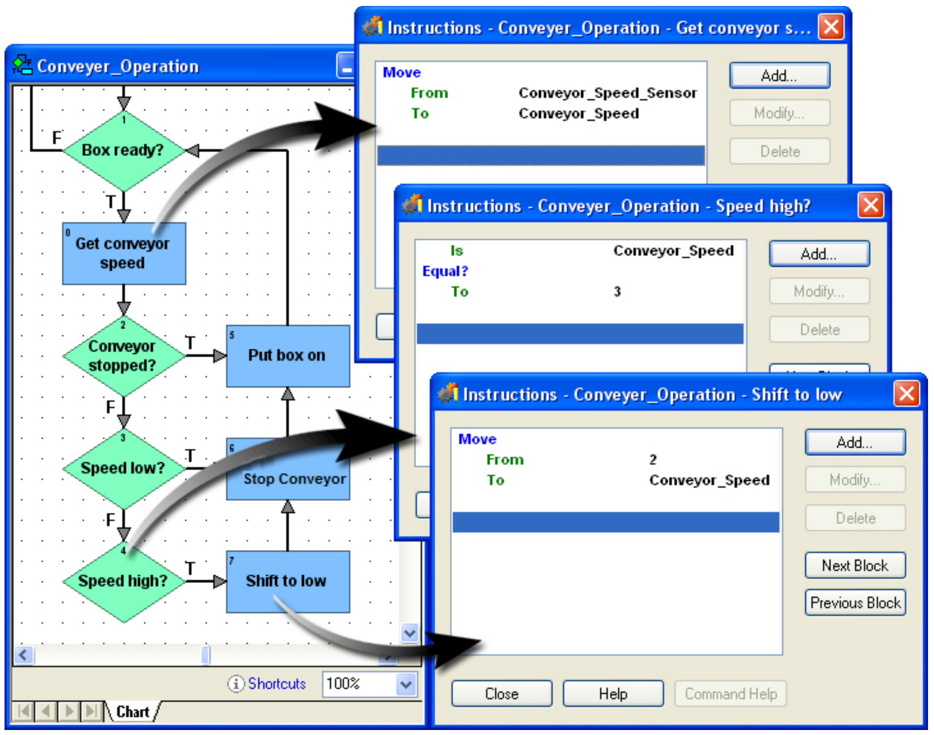
```
for Bag_Count = 1 to 12 step 1 //Count 12 items dropped into box
  Bag_Dropper = 1; //Turn on bag dropper
next
```

See Chapter 11: Using OptoScript for more information on using code to streamline control structures in your strategy.

Programming Case Statements

A frequent need in programming is to create case statements, also called "switch" statements, "if/then/else" statements, or "nested if" statements. These statements create multiple decision points: if the condition is A, then X will happen; if the condition is B, then Y will happen, and so on. (This example uses regular PAC Control commands; seeChapter 11: Using OptoScript for another option that takes up less space in your flowchart.)

For example, suppose you have a conveyor with three speeds: high (3), low (2), and stopped (1). Sometimes it runs at high speed, sometimes at low speed; but before a box can be put on the conveyor, the conveyor must be stopped. And it can be stopped only from low speed, not from high speed. Here's a portion of a chart showing the case statements that control this process:



Instructions - Conveyer_Operation - Get conveyor s...

Move	
From	Conveyor_Speed_Sensor
To	Conveyor_Speed

Instructions - Conveyer_Operation - Speed high?

Is	Conveyor_Speed
Equal?	
To	3

Instructions - Conveyer_Operation - Shift to low

Move	
From	2
To	Conveyor_Speed

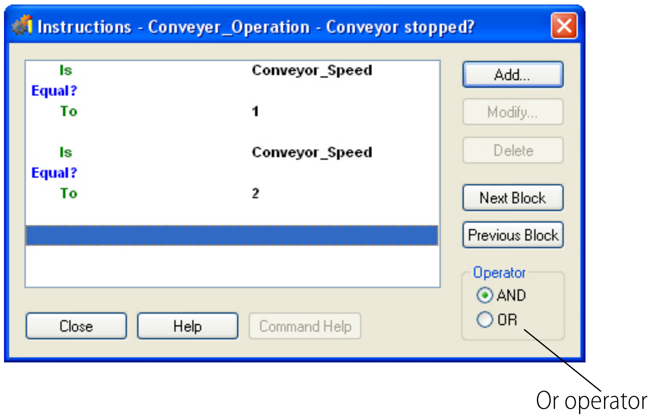
Conveyor speed is checked and moved (copied) into a variable.

Each condition block checks for a particular speed. This one checks for high speed (3).

If speed is too high, it is shifted down to the next level, until the conveyor is finally stopped and the box can be put on.

The example above shows three cases, because there are three possible speeds and each speed demands a different action.

If you had only two possibilities (for example, if the box could also be put on the conveyor at low speed), you could handle both possibilities within one condition block. For example, you could put two Equal? commands in the same condition block, and check the Or operator, as shown at right.


Instructions - Conveyer_Operation - Conveyor stopped?

```
Is                Conveyor_Speed
Equal?
To                1

Is                Conveyor_Speed
Equal?
To                2
```

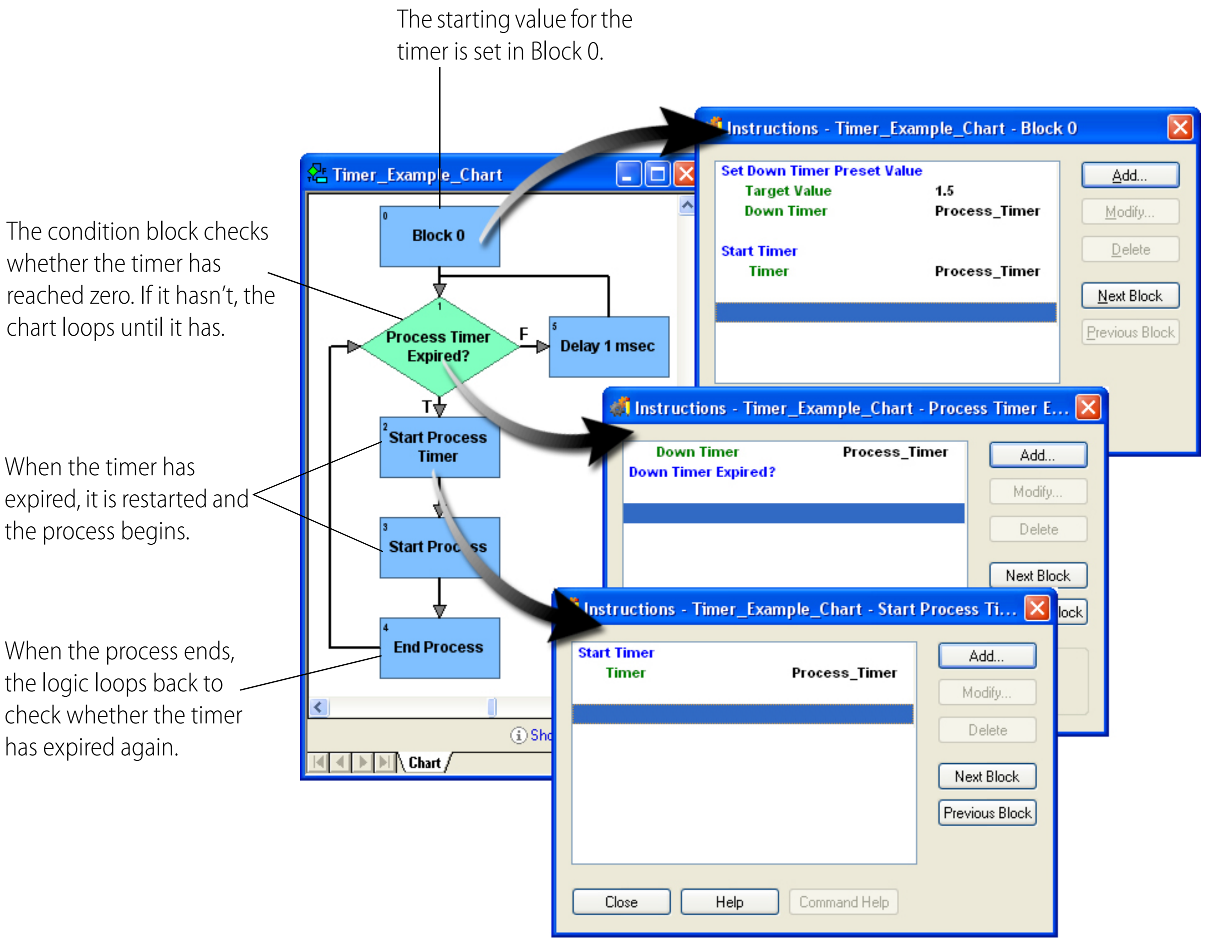
Operator
AND
[✓ OR]

Using a Timer

Timers come in two forms:

- Up timers count up from zero and are generally used to measure how long something takes.
- Down timers count down to zero from a number you set and are frequently used to determine when events should begin.

Here's an example of a down timer. This process starts every 1.5 seconds.


The starting value for the timer is set in Block 0.
The condition block checks whether the timer has reached zero. If it hasn't, the chart loops until it has.
When the timer has expired, it is restarted and the process begins.
When the process ends, the logic loops back to check whether the timer has expired again.
Timer_Example_Chart
0 Block 0
1 Process Timer Expired?
F
T
5 Delay 1 msec
2 Start Process Timer
3 Start Process
4 End Process
Chart

Instructions - Timer_Example_Chart - Block 0

```
Set Down Timer Preset Value
    Target Value          1.5
    Down Timer            Process_Timer

Start Timer
    Timer                 Process_Timer
```

Instructions - Timer_Example_Chart - Process Timer E...

```
    Down Timer            Process_Timer
Down Timer Expired?
```

Instructions - Timer_Example_Chart - Start Process Ti...

```
Start Timer
    Timer                 Process_Timer
```

Add...
Modify...
Delete
Next Block
Previous Block
Close
Help
Command Help

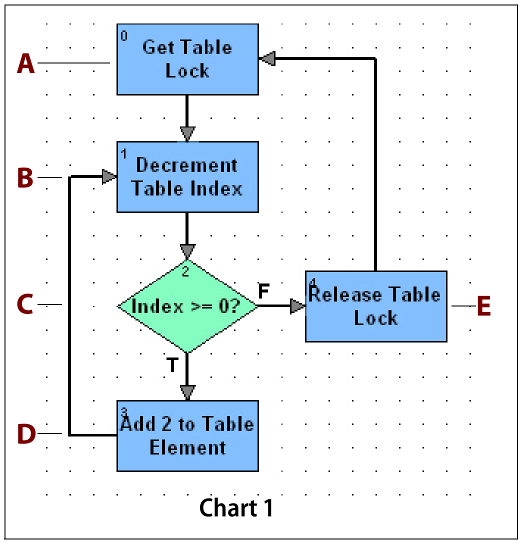
For additional details, see "Using Timers" on page 368.

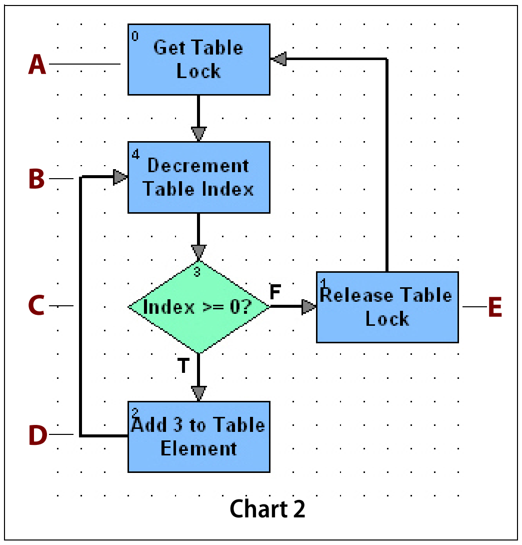
Using a Flag Lock

The Flag Lock and Flag Unlock commands allow you to give a task exclusive access to one or more objects—such as tables, integers, or I/O units—until the task is complete. For example, if you want two different charts to change the values in the same table, it is important for one chart to complete its task before allowing the second chart to access the table.

To demonstrate, the following charts show how to construct two charts in a strategy so that each chart can access the same table without interference from the other chart.

Chart 1 and Chart 2 are very similar. The main difference is that one block (**D**) in Chart 1 adds 2 to each table element, while in Chart 2 it adds 3.





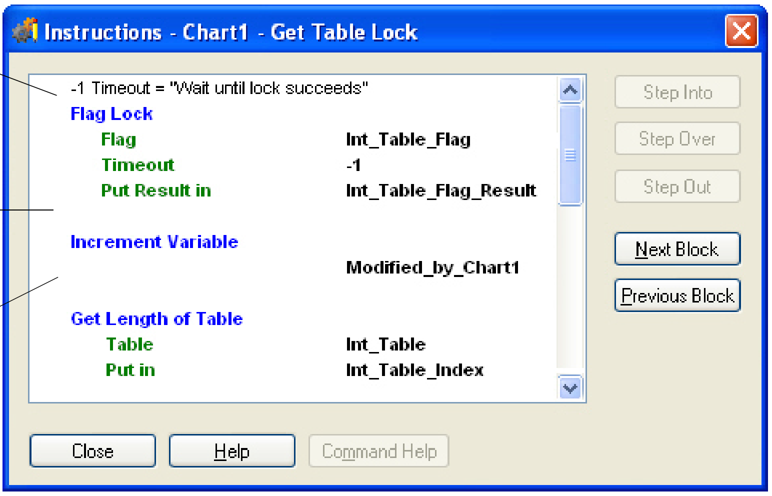
Here's what each block accomplishes:

A—*Get Table Lock.* Uses the following instructions to wait until Chart 2 unlocks the flag before it locks the flag on the table. It also tracks the number of times Chart modifies the table, and it gets the length of the table. The Timeout value of -1 makes the chart wait the other chart has successfully unlocked the flag on the table. See also, "Flag Lock Timeout Values" on page 95.

Flag lock will wait until Chart 2 unlocks the flag on the table. See also, "Flag Lock Timeout Values" on page 95.

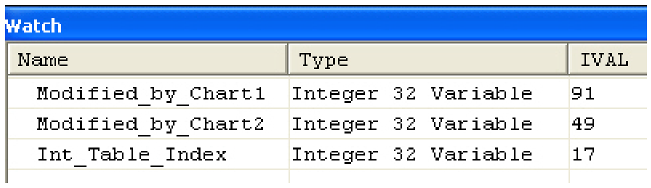
Tracks the number of times the table has been modified by Chart1

Gets the length of the table and puts it in Int_Table_Index



```
-1 Timeout = "Wait until lock succeeds"
Flag Lock
    Flag                  Int_Table_Flag
    Timeout               -1
    Put Result in         Int_Table_Flag_Result

Increment Variable
                          Modified_by_Chart1

Get Length of Table
    Table                 Int_Table
    Put in                Int_Table_Index
```

Using the numeric variables Modified_by_Chart1 and Modified_by_Chart2 (from Chart 2), a watch window can track the number of times the table has been modified by each chart. For more information, see "Using a Watch Window" on page 36.



Name	Type	IVAL
Modified_by_Chart1	Integer 32 Variable	91
Modified_by_Chart2	Integer 32 Variable	49
Int_Table_Index	Integer 32 Variable	17

B—*Decrement Table Index.* The table has 100 elements, numbered from 0 to 99. This instruction block starts at element 99 and on each loop decrements by 1 to go to the next element. This continues until it reaches index 0.

C—*Index >=0.* If the index is greater or equal to zero, the chart proceeds to **D**. If the index equals -1, the chart goes to **E**.

D—*Add 2 (or 3) to Table Element.* Adds two (or three) to the value of the current table index.

E—*Release Table Lock.* This block uses a *Force unlock* of zero, which means it will unlock only for the chart that locked the flag. In this case, if Chart 1 locked the flag, only Chart 1 can unlock the flag.

Flag Lock Timeout Values

The following chart shows the effect of different Timeout values used with the Flag Lock command:

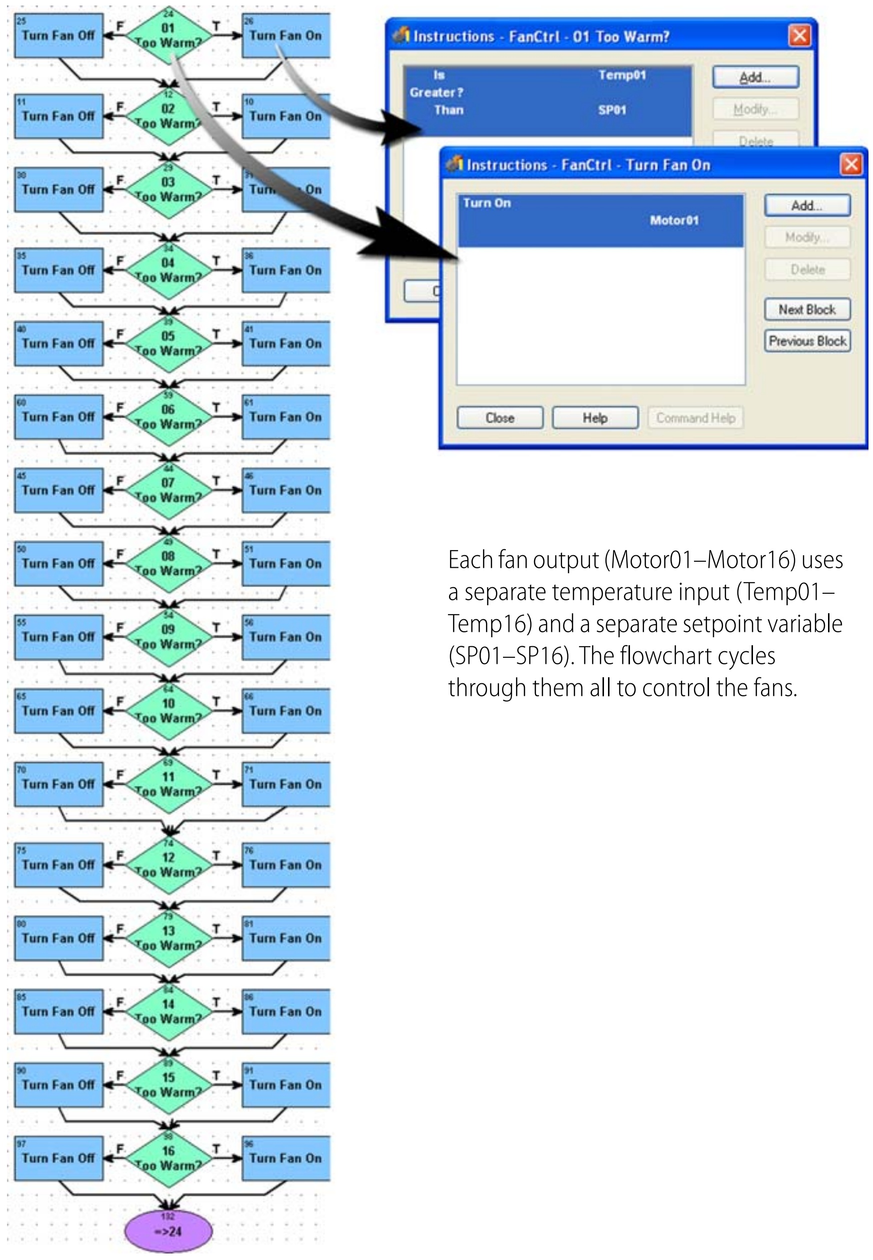
Timeout	Effect
-1	Wait until success. The application will wait (block) as long as it takes to lock the flag. Since it never gives up, the only value that can be returned is 0.
0	Don't wait at all. Succeeds only if the flag is available to be locked immediately. Makes this a non-blocking call. If the flag is not already locked, a return of 0 (Success) is returned. If the flag is locked, a -17 (Already Locked) error is returned.
Value greater than 0	Wait maximum of value milliseconds to lock the flag. If it succeeds within the specified time, 0 is returned; otherwise, a -37 (Timeout on Lock) error is returned.
Value less than 0	Produces a-8 (Invalid Data) error

Pointers and Indexing

The following examples show three possible ways to control 16 fans based on setpoints.

Without Pointers

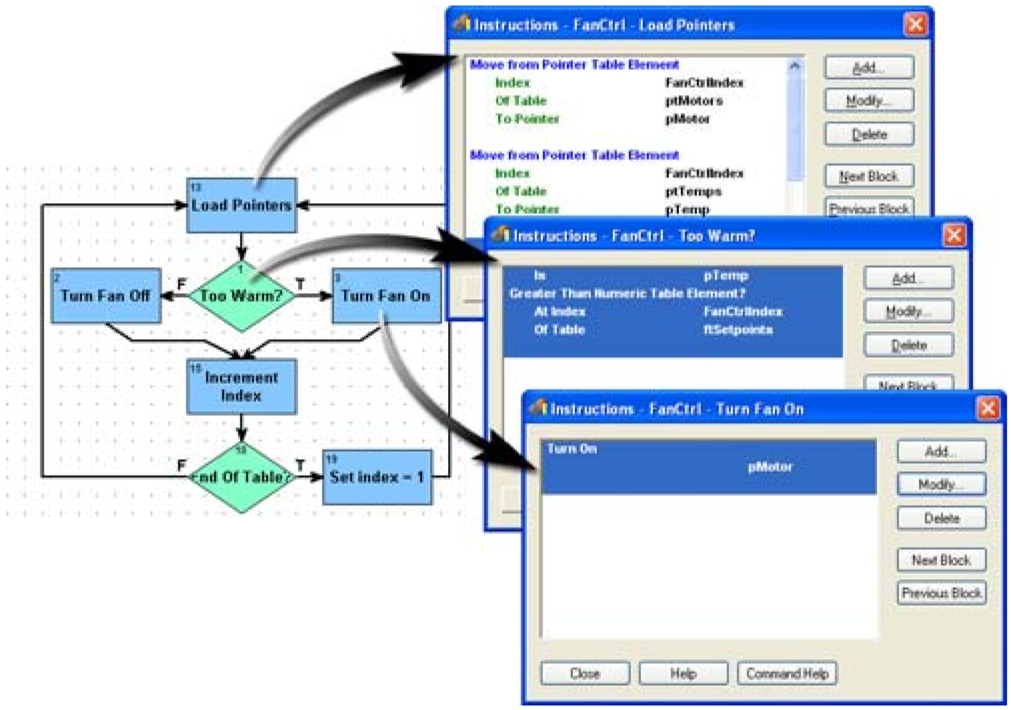
The first way, shown below, uses individual fan outputs and setpoint variables to determine whether to turn off or on the appropriate fan. It's a simple way to solve the problem but takes a long time to program and produces a large, repetitive section in the flowchart. It also consumes more control engine memory, due to the large number of conditional and action blocks.



Each fan output (Motor01–Motor16) uses a separate temperature input (Temp01–Temp16) and a separate setpoint variable (SP01–SP16). The flowchart cycles through them all to control the fans.

With Pointers and Indexing

The second way uses pointers and indexing to accomplish the same result in a smaller space, using less control engine memory and fewer programming steps. The fan outputs and the temperature inputs are referred to by pointers. The setpoints, instead of requiring individual variables, are simply numbers in a table.

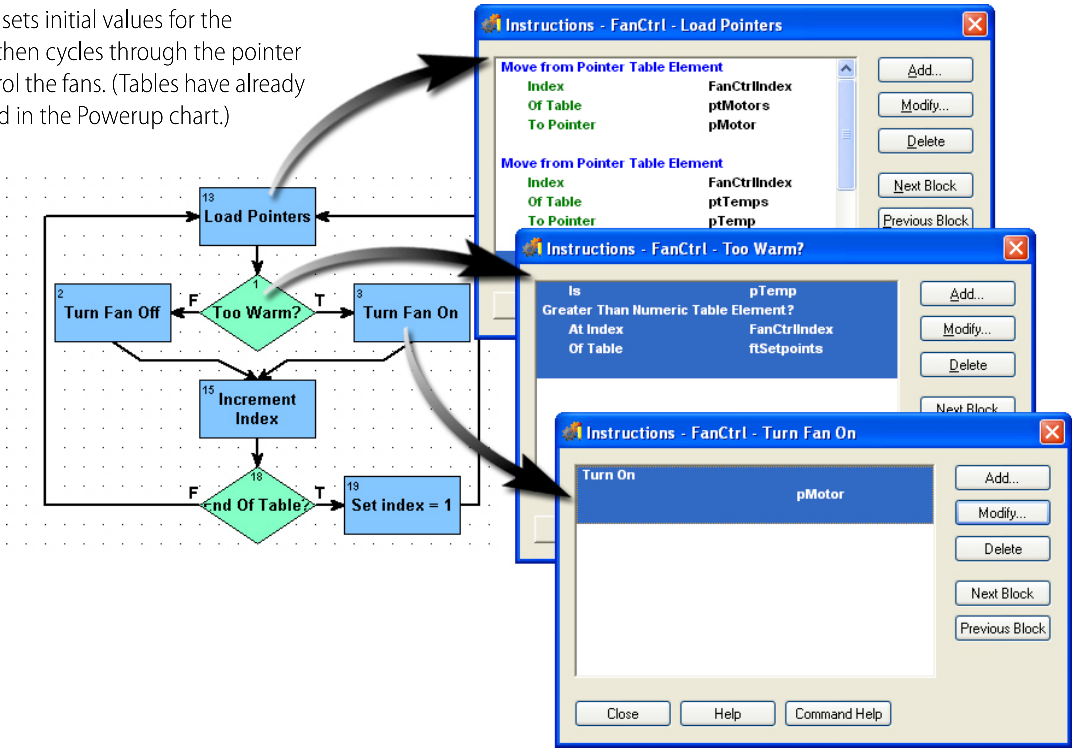


```
Move from Pointer Table Element
    Index               FanCtrlIndex
    Of Table            ptMotors
    To Pointer          pMotor

Move from Pointer Table Element
    Index               FanCtrlIndex
    Of Table            ptTemps
    To Pointer          pTemp
```

```
    Is                  pTemp
Greater Than Numeric Table Element?
    At Index            FanCtrlIndex
    Of Table            ftSetpoints
```

```
Turn On
                        pMotor
```

The flowchart sets initial values for the pointers, and then cycles through the pointer tables to control the fans. (Tables have already been initialized in the Powerup chart.)



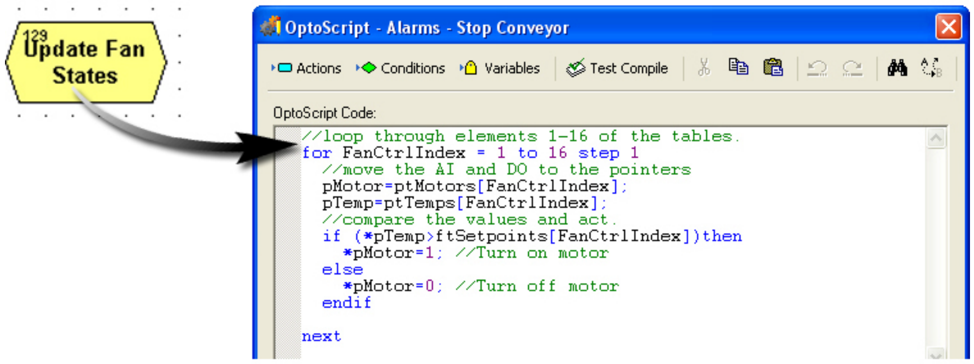
```
Move from Pointer Table Element
    Index               FanCtrlIndex
    Of Table            ptMotors
    To Pointer          pMotor

Move from Pointer Table Element
    Index               FanCtrlIndex
    Of Table            ptTemps
    To Pointer          pTemp
```

```
    Is                  pTemp
Greater Than Numeric Table Element?
    At Index            FanCtrlIndex
    Of Table            ftSetpoints
```

```
Turn On
                        pMotor
```

Using OptoScript

The third way also uses pointers and indexing, but places all the action in an OptoScript block. See "11: Using OptoScript" for more information on using OptoScript.



```
//loop through elements 1-16 of the tables.
for FanCtrlIndex = 1 to 16 step 1
  //move the AI and DO to the pointers
  pMotor=ptMotors[FanCtrlIndex];
  pTemp=ptTemps[FanCtrlIndex];
  //compare the values and act.
  if (*pTemp>ftSetpoints[FanCtrlIndex])then
    *pMotor=1; //Turn on motor
  else
    *pMotor=0; //Turn off motor
  endif

next
```

Optimizing Throughput

For tips on how to design your PAC Project™ system and program your PAC Control™ strategy to ensure optimum system performance see Opto 22 form 1776, *Optimizing PAC Project System Performance*.

For additional information, see Opto 22 form 1702, the *PAC Display User's Guide*.

5: Working with Control Engines

Introduction

This chapter shows you how to configure and work with control engines. See "Compatible Controllers and I/O Units" on page 6.

In this Chapter

Configuring Control Engines 99
Using Network Segmenting in PAC Control 103
Using Ethernet Link Redundancy in PAC Control 104
Using Redundant Controllers 109
Changing or Deleting a Control Engine 109
Inspecting Control Engines and the Queue 111
Archiving Strategies on the Control Engine 116
Restoring Archived Strategies from the Control Engine 118
Downloading Files Using PAC Terminal 120

Configuring Control Engines

Before you can use a control engine to run a strategy, you must first define the control engine on your PC and then associate the control engine with your PAC Control strategy.

- See "Defining a Control Engine on Your PC" below to identify the connection through which the PC and the control engine communicate. Because this process writes to the Windows Registry on your PC, you must define control engines for each computer that uses your strategy. (If your computer can boot to two operating systems, you must configure control engines for each operating system.) You can define control engines in PAC Control or in the software utility, PAC Terminal.
- See "Associating the Control Engine with Your Strategy" on page 103 to identify which defined control engine is the active control engine. Although you can associate several control engines with the same strategy if necessary, the strategy can be downloaded to only one at a time. The control engine set to receive the download is called the active engine. You must use PAC Control for associating the control engine with your strategy.

Defining a Control Engine on Your PC

You can use either PAC Terminal or PAC Control to define a control engine on your PC. Follow these steps to define a control engine using the PAC tool of your choice.

1. Start either PAC Terminal or PAC Control, and then open the Select Control Engine dialog box:

Using PAC Terminal:

– In Windows 7 and Windows Vista, press the Windows Start key, and then click Programs > Opto 22 > PAC Project 9.6 > Tools > PAC Terminal.

– In Windows 10 and Windows 8.1, press the Windows Start key, type **`PAC Terminal 9.6`** and then press the Enter key.

From PAC Terminal's menu bar, click Configure > Control Engines. When the Select Control Engine dialog box opens, go to step 2.

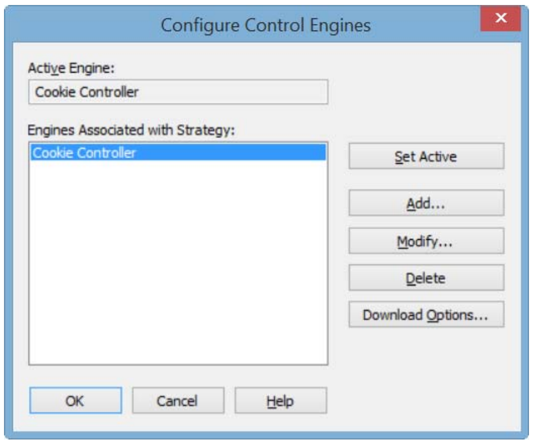
Using PAC Control:

– In Windows 7 and Windows Vista, press the Windows Start key, and then click Programs > Opto 22 > PAC Project 9.6 > PAC Control.

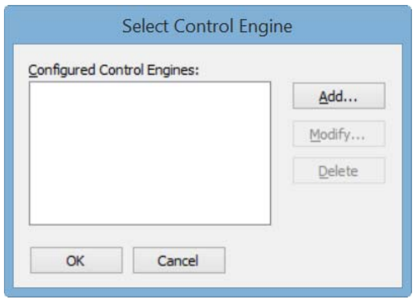
– In Windows 10 and Windows 8.1, press the Windows Start key, type **`PAC Control 9.6`** and then press the Enter key.

a. Open a strategy in either Configure mode or Online mode.

b. From PAC Control's menu bar, click Configure > Control Engines. The Configure Control Engines dialog box appears.



c. Click Add.
The Select Control Engine dialog box appears.

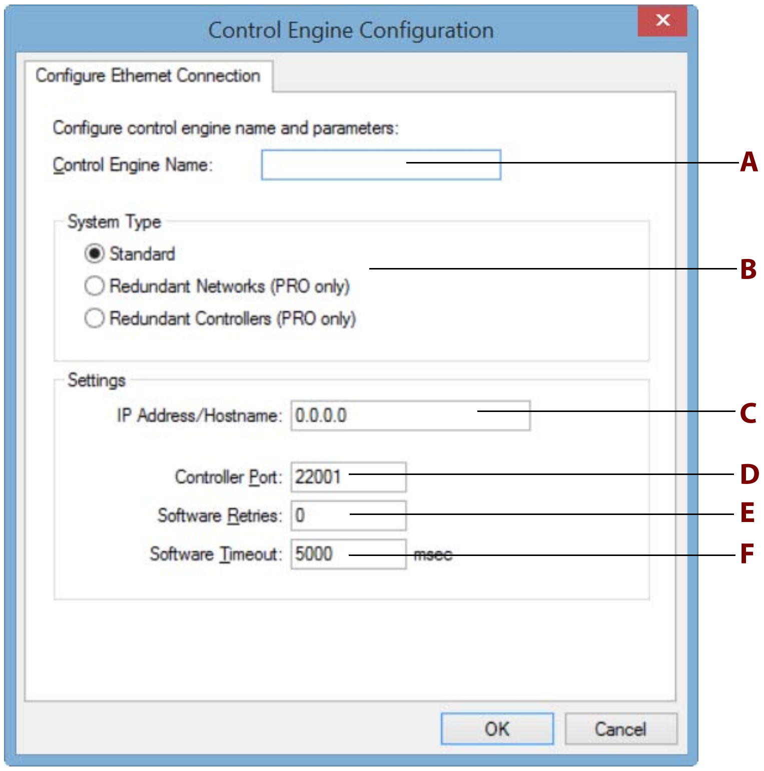


2. The Select Control Engine dialog box lists all the control engines configured on your system, whether or not they are associated with your strategy.

– If the control engine you want appears in the list, it has already been defined on this PC, click to highlight the control engine's name and then click OK. Then skip to "Associating the Control Engine with Your Strategy" on page 103.

– If the control engine you want is not in the list, click Add.

The Control Engine Configuration dialog box appears.



Configure control engine name and parameters:

Control Engine Name:

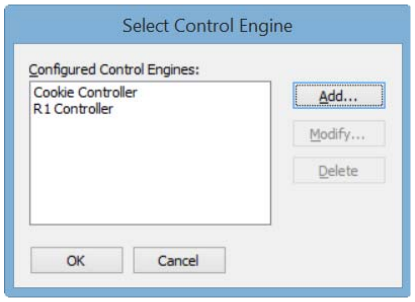
System Type
[✓ Standard]
Redundant Networks (PRO only)
Redundant Controllers (PRO only)

Settings
IP Address/Hostname: 0.0.0.0
Controller Port: 22001
Software Retries: 0
Software Timeout: 5000 msec

3. Complete the fields as described in "Control Engine Configuration Dialog Box" below.

4. Click OK.

The Select Control Engines dialog box appears, displaying your new control engine in the list.



Configured Control Engines:
Cookie Controller
R1 Controller

5. In the Select Control Engine dialog box, highlight the control engine you want to associate with the strategy, and then click OK. Continue with "Associating the Control Engine with Your Strategy" on page 103.

Control Engine Configuration Dialog Box

A—*Control Engine Name.* Enter a descriptive name for the control engine. Valid characters are letters, numbers, spaces, and most other characters except colons and square brackets. Spaces cannot be used as first or last characters.

B—*System Type.* Select the system type:

- Standard: A normal system not using redundant networks or redundant controllers.
- Redundant Networks (PRO only): A system that has redundant networks, which is also referred to as *link redundancy*. When selected, two IP address boxes are available under Settings for the primary and secondary networks. See "Using Ethernet Link Redundancy in PAC Control" on page 104 for more information.
- Redundant Controllers (PRO only): Use with a system configured with the SNAP PAC Redundancy Option. When selected, three IP address boxes are available under Settings for Controller 1, Controller 2, and Arbiter. For more information on setting up the Redundancy Option, see form 1831, the *SNAP PAC Redundancy Option User's Guide*.

C—*IP Address/Hostname.* The IP Address/Hostname text boxes available depend on your System Type. Each IP address is entered in decimal notation (for example, `192.9.200.24`).

Hostnames are supported only in standard configurations, not redundant ones. In order to use a hostname, your network must have a domain name server (DNS) and the hostname must already have been entered in your DNS either by you or someone in your IT department.

- For Standard: Enter the IP address or hostname of the control engine.
- For Redundant Networks (PRO only): Enter the Primary and Secondary IP addresses. The Primary address is ETHERNET 1 on a SNAP PAC controller, and the Secondary IP address is ETHERNET 2. Both IP addresses use the same port number. See "Using Ethernet Link Redundancy in PAC Control" on page 104 for more information.
- For Redundant Controllers (PRO only): For the SNAP PAC Redundancy Option. Enter addresses for Controller 1, Controller 2, and Arbiter. For more information on setting up the Redundancy Option, see form 1831, the *SNAP PAC Redundancy Option User's Guide*.

D—*Port.* Enter the control engine's IP port number. The default of 22001 is the port of the host task on the control engine; this default normally should not be changed.

E—*Retries.* Retries indicate the number of times PAC Control will reattempt communications with the control engine. Since retries are automatically handled by the protocol (TCP/IP), enter zero here.

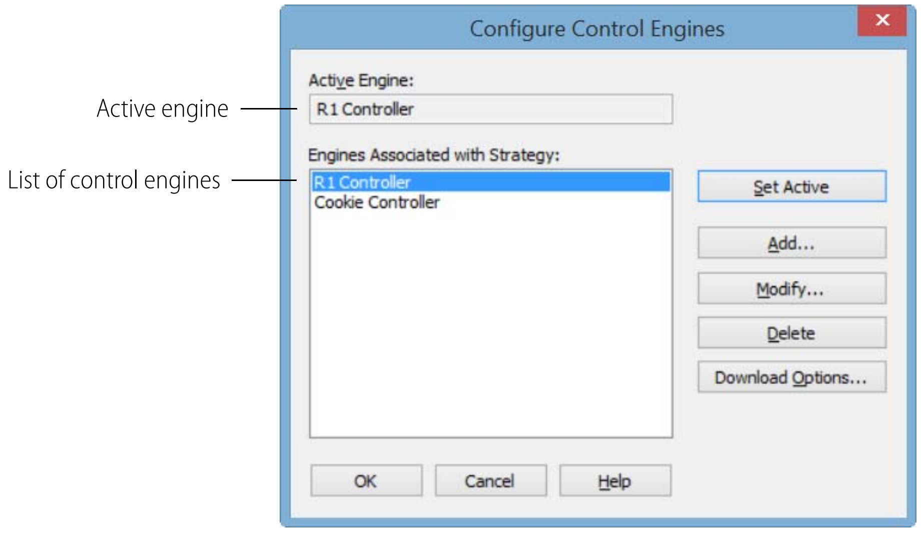
F—*Timeout.* Enter the timeout value in milliseconds. Timeout value is the length of time PAC Control tries to establish communication through the port. If it fails, it tries again as many times as specified in D. Any positive integer is valid. For Ethernet, 3–5 seconds (3000–5000 milliseconds) is a good starting point.

Associating the Control Engine with Your Strategy

After you have defined the control engine on your PC (see page 100), it can be associated with your strategy.

1. If you are in PAC Terminal, close PAC Terminal. Open the strategy in PAC Control (Configure mode or Online mode) and from the menu bar, click Configure > Control Engines.

   In the Configure Control Engines dialog box, the engine's name appears in the list.



Active Engine: R1 Controller

Engines Associated with Strategy:
R1 Controller
Cookie Controller

2. Check to make sure the correct control engine appears in the Active Engine field. If not, highlight the one you want and click Set Active.

   Only one control engine can be active. If only one control engine is listed, it automatically becomes the Active Engine.

Your control engine configuration is complete.

Using Network Segmenting in PAC Control

You can take advantage of the two independent Ethernet network interfaces on a SNAP PAC controller to segment the control network from the company network. Because the two network interfaces are completely independent and have separate IP addresses, they can be set up on two separate networks. Host traffic can communicate on one, while the controller communicates with the I/O units on the other.

To segment networks, start by assigning a secondary IP address to the controller in PAC Manager, following instructions in Opto 22 form 1704, the *PAC Manager User's Guide*. The secondary IP address is used for communication through the controller's Ethernet 2 interface. Remember that this interface must be on a completely separate network segment.

You don't need to do anything special in PAC Control. Configure only one control engine, using the IP address of the Ethernet interface to be used for host communication. The control engine will direct communication to I/O units through the other interface, based upon the I/O unit's IP address. The IP addresses of the I/O units must be compatible with the secondary IP address of the controller.

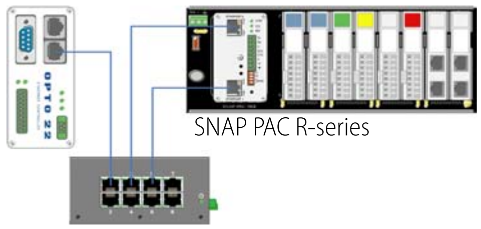
NOTE: PAC Display Basic and other hosts on the company network cannot communicate with I/O units when they are on a separate network. To solve this problem, you can purchase PAC Project Professional, or PAC Display Professional and OptoOPCServer (version 7.0 and newer) separately; these applications support network segmenting by communicating with I/O units through the controller. Also, you can add a second network interface card (NIC) and assign it an IP address compatible with the I/O unit network.

Using Ethernet Link Redundancy in PAC Control

Ethernet link redundancy provides a backup network connection in case the primary connection route becomes unavailable. If that happens, the controller uses a secondary communication route.

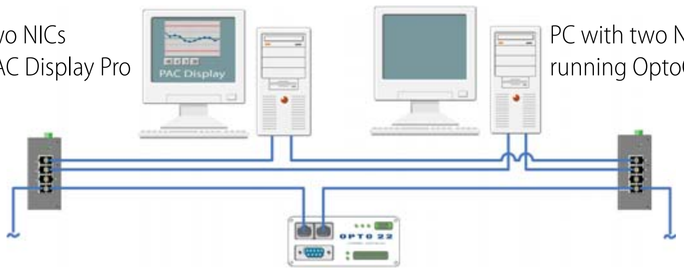
There are two types of link redundancy in PAC Project, which you can use individually or combined:

- **Controller to I/O**: When link redundancy is configured from the controller to I/O, if communication to the primary I/O unit IP address fails, communication switches automatically to the secondary I/O unit IP address, which usually is the second Ethernet network interface on a SNAP PAC R-series controller. See "Commands for Ethernet Link Redundancy" on page 335 for more information on primary and secondary I/O units.



NOTE: If your system has redundant controllers, in order to use link redundancy from the controller to the I/O, you must connect ETHERNET 2 on the controllers and on the I/O units to the same switch. No other devices can be connected to that switch.

- **PAC Display Pro to controller**: When a PAC Display Professional project is installed on a PC with two network interface cards (NICs), the project can be configured to use the primary and secondary IP addresses to access data from the PAC Control strategy running on the control engine. One address is always active. If the primary address is unavailable, then the secondary address becomes the active address, and the PAC Display scanner automatically shifts to the secondary address. If the secondary address fails, then the primary address becomes the active address and automatically tries the primary address again. For details, see form 1702, the *PAC Display User's Guide*.


PC with two NICs running PAC Display Pro
PC with two NICs running OptoOPCServer
SNAP PAC S-series

NOTE: If your system has redundant controllers, you cannot use link redundancy between PAC Display Pro and the controller. For more information on using redundant controllers, see Opto 22 form 1831, the SNAP PAC Redundancy Option User's Guide.

System Architecture for Ethernet Link Redundancy

As shown in the examples below, you can set up your system in several ways to take advantage of the link redundancy capability in PAC Control. For additional information, see the SNAP PAC controllers' user's guides. For details on how PAC Display works with link redundancy, see form 1702, the *PAC Display User's Guide*.

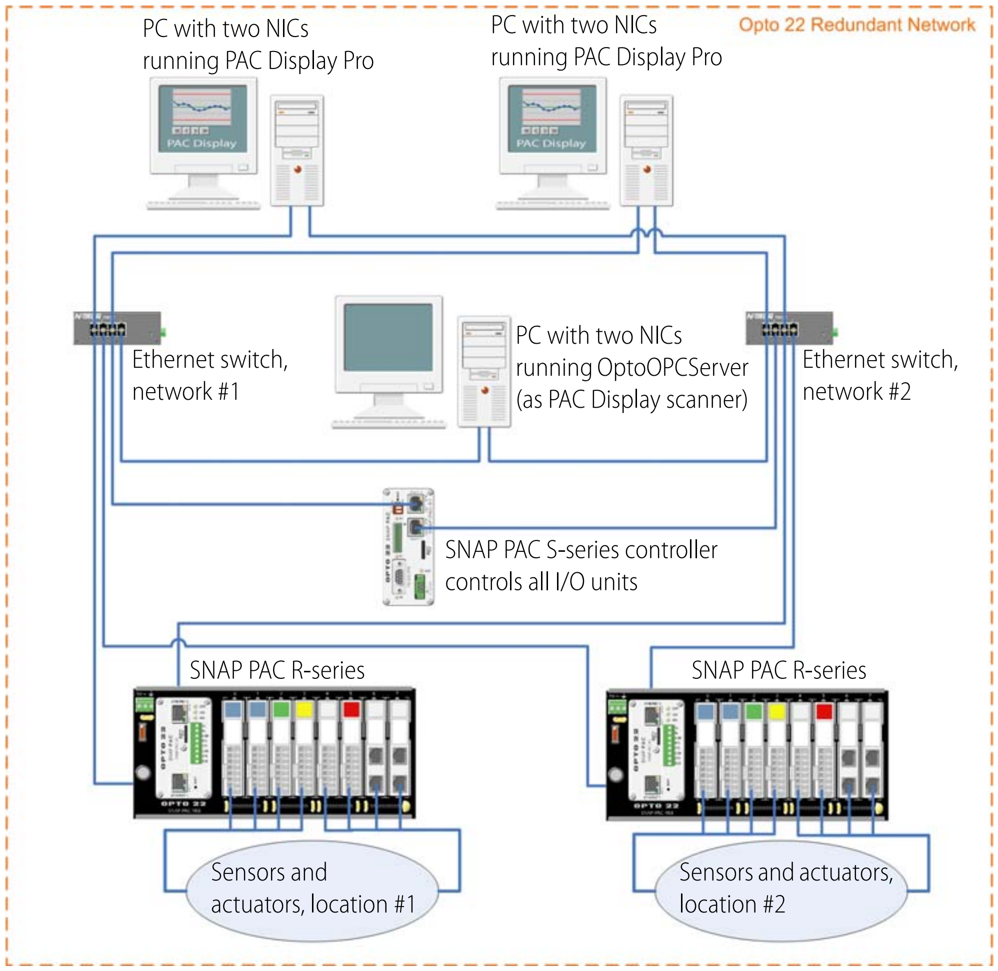
Using a SoftPAC Controller. In examples 1 and 2 below, the PAC S-series controller can be replaced by a SoftPAC controller. The PC running SoftPAC must have two network interface cards (NICs) configured with static primary and secondary IP addresses on different subnets.

Example 1: Ethernet Link Redundancy

In this example, the primary concern is that the Ethernet network may need maintenance or may fail, leaving the computer running PAC Display Professional, the computer running the OptoOPCServer (as the PAC Display scanner), the controller, and the I/O units unable to communicate.

The solution is to connect all these devices on two networks; if one network goes down, devices can communicate on the other. Note that each computer has two network interface cards (NICs). I/O units are controlled by the controller. Each I/O unit is connected to its own group of sensors and actuators, but all are connected to the same two networks.

If one network needs maintenance, for example to replace a switch, you can safely shut it down using the Set Target Address commands in PAC Control and options in PAC Display Professional.

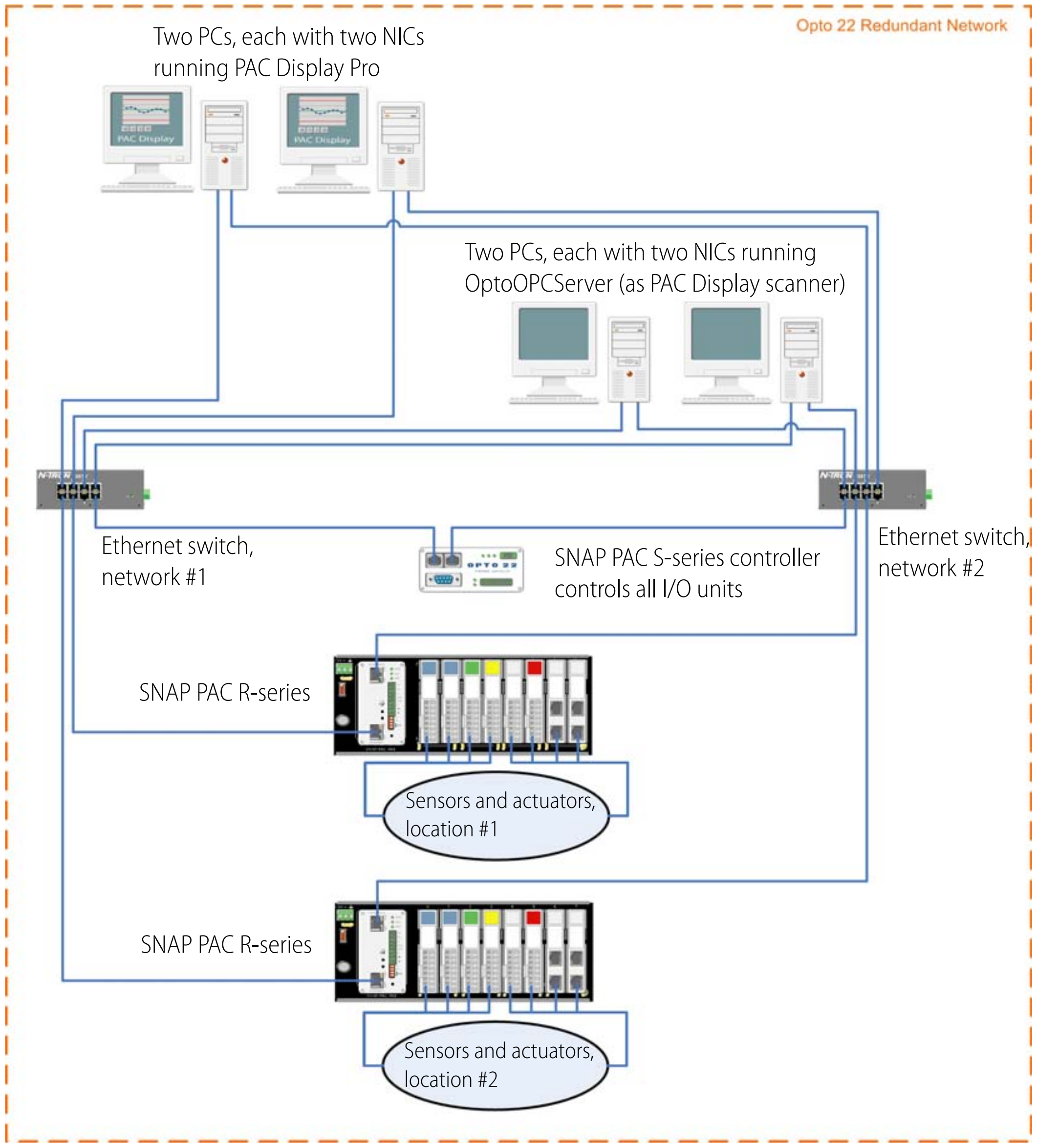

PC with two NICs running PAC Display Pro
PC with two NICs running PAC Display Pro
Opto 22 Redundant Network
PAC Display
PAC Display
Ethernet switch, network #1
PC with two NICs running OptoOPCServer (as PAC Display scanner)
Ethernet switch, network #2
SNAP PAC S-series controller controls all I/O units
SNAP PAC R-series
SNAP PAC R-series
Sensors and actuators, location #1
Sensors and actuators, location #2

Example 2: Ethernet Link, Computer, and Software Redundancy

The second example of link redundancy illustrates concern not only about the stability of the Ethernet network, but also about the computers and the software run on them. Any of these things—the network, a computer, or the OptoOPCServer or PAC Display Professional software running on the computer—may need maintenance or may fail.

The solution here is to provide duplicate computers and software, and connect all devices on two networks. Each computer has two NICs. I/O units are all controlled by the controller. Each I/O unit is connected to its own group of sensors and actuators, but both are connected to the same two networks.

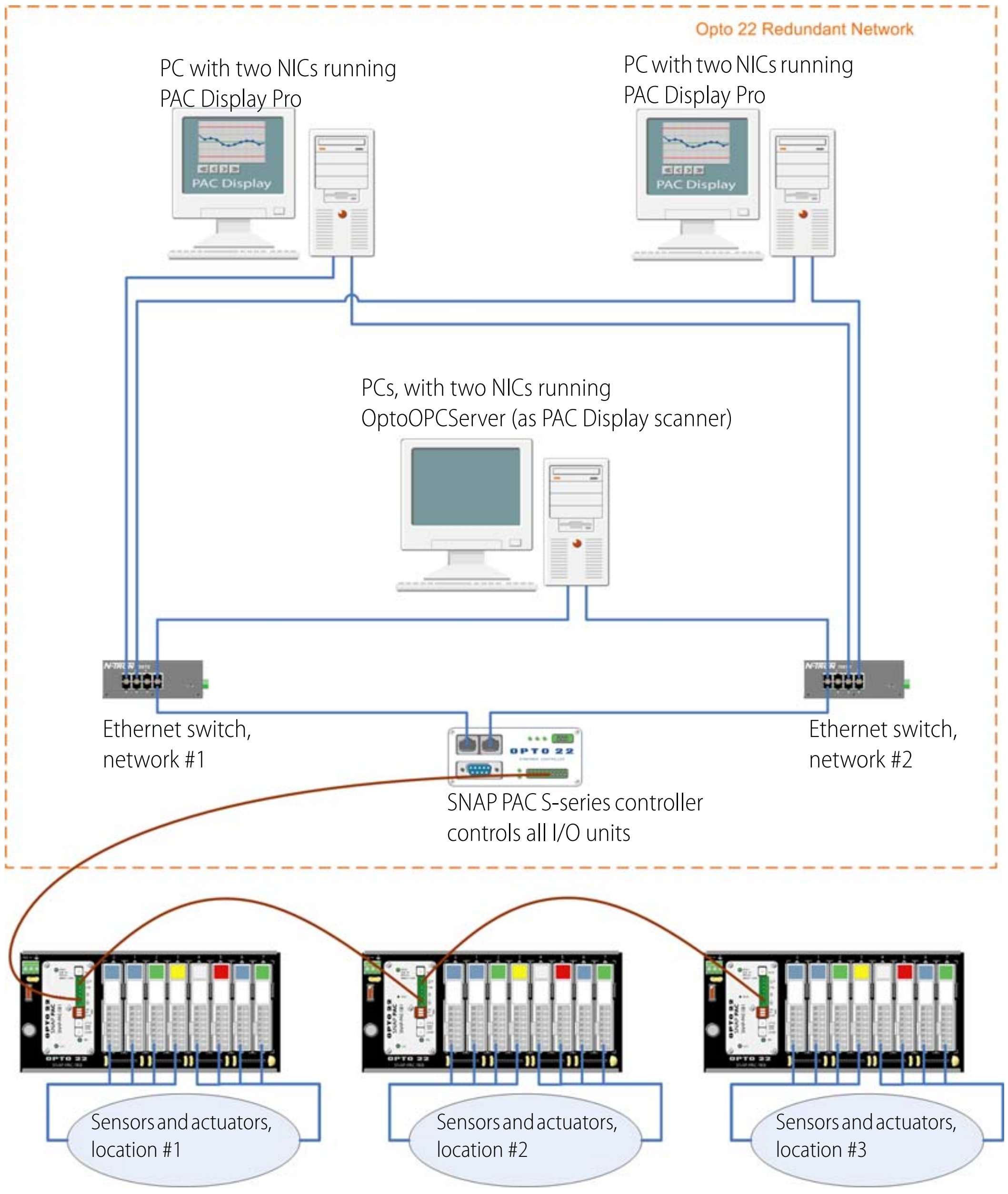
For more information on using Ethernet link redundancy with PAC Display, see form 1702, the *PAC Display User's Guide*.


Opto 22 Redundant Network
Two PCs, each with two NICs running PAC Display Pro
Two PCs, each with two NICs running OptoOPCServer (as PAC Display scanner)
Ethernet switch, network #1
SNAP PAC S-series controller controls all I/O units
Ethernet switch, network #2
SNAP PAC R-series
Sensors and actuators, location #1
SNAP PAC R-series
Sensors and actuators, location #2

Example 3: Ethernet Link Redundancy with Serial I/O Units

This third example of link redundancy shows a SNAP PAC S-series controller with serial I/O units. The redundancy is in the Ethernet networks between the controller and computers running PAC Display

Professional and OptoOPCServer. This example provides link redundancy for the Ethernet network. Again, note that the computers all have two NICs.


Opto 22 Redundant Network
PC with two NICs running PAC Display Pro
PAC Display
PC with two NICs running PAC Display Pro
PAC Display
PCs, with two NICs running OptoOPCServer (as PAC Display scanner)
Ethernet switch, network #1
Ethernet switch, network #2
SNAP PAC S-series controller controls all I/O units
Sensors and actuators, location #1
Sensors and actuators, location #2
Sensors and actuators, location #3

Configuring Ethernet Link Redundancy

Ethernet link redundancy from PAC Display Pro to a control engine is easy to configure. First, make sure the secondary IP address has been assigned to the controller in PAC Manager. Then in PAC Control, just enter both the primary and secondary IP addresses when you configure the control engine (see page 99). Once you have set up link redundancy for the control engine you are using in your PAC Control strategy, it works automatically in PAC Display.

To configure link redundancy from the control engine to I/O units, enter both the primary and secondary IP addresses when you configure the I/O unit (see "Adding an I/O Unit" on page 130).

Using Strategies with Link Redundancy

Although PAC Display and OptoOPCServer (as the PAC Display Pro scanner) automatically shift to the secondary address when the primary address is not available, the PAC Control debugger does not. In PAC Control, you control which address receives communication, so you know exactly how communication is occurring during debugging. To change between primary and secondary control engine addresses, follow the steps in "Changing the Control Engine that Receives the Downloaded Strategy" on page 110.

Once you have changed the address, when you enter Debug mode, you are communicating through the address you chose. Downloading the strategy to a SNAP PAC controller through one of its Ethernet interfaces replaces any strategy that was downloaded earlier through its other interface.

Using Redundant Controllers

The SNAP PAC Redundancy Option allows you to configure two redundant SNAP-PAC-S controllers that use the same PAC Control strategy. By utilizing *sync blocks* in the strategy and *persistent/redundant variables*, critical data is synchronized in both controllers so that the inactive controller can take over at any time and become the active controller. To learn more about the SNAP PAC Redundancy Option and how to set up a strategy using sync blocks and persistent/redundant variables, see Opto 22 form 1831, the *SNAP PAC Redundancy Option User's Guide*.

NOTE: If your system has redundant controllers, you cannot use link redundancy between PAC Display Pro and the controller.

Changing or Deleting a Control Engine

See the following topics to change or delete a control engine:

- "Changing a Control Engine's Definition" (below)
- "Changing the Control Engine that Receives the Downloaded Strategy" on page 110
- "Removing a Control Engine's Association with a Strategy" on page 110
- "Deleting a Control Engine from Your PC" on page 111

Changing a Control Engine's Definition

Whenever necessary, you can change a control engine's definition on your PC—its name, the port the PC uses to communicate with it, or the PC port setup (such as timeouts and retries). These changes can be made either in PAC Control or in PAC Terminal.

1. Choose one of the following:
   - **In PAC Control:** With the strategy open in Configure mode or Online mode, right-click the control engine name on the Strategy Tree and choose Modify from the pop-up menu.

   – **In PAC Terminal:** From the PAC Terminal menu bar, click Configure > Control Engine.

   The Select Control Engine dialog box appears.

2. In the Select Control Engine dialog box, click the control engine you want to change and click Modify.
3. Make the necessary changes, following the same steps you would for configuring the control engine initially. (For help, see "Defining a Control Engine on Your PC" on page 100.)

Changing the Control Engine that Receives the Downloaded Strategy

You can configure several control engines for a strategy, but only one at a time can receive the downloaded strategy. The control engine that receives the strategy is called the active engine.

NOTE: In PAC Control Professional, you can also configure a control engine for Ethernet link redundancy (see "Using Ethernet Link Redundancy in PAC Control" on page 104 for more information). In this case, you can also choose whether the primary or secondary IP address will receive the strategy.

To change the control engine that receives the downloaded strategy, follow these steps:

1. Make sure the strategy is open and in Configure or Online mode.
2. On the Strategy Tree, right-click the name of the control engine you want to set as the active engine.

   If its name does not appear in the Strategy Tree, follow the steps in "Configuring Control Engines" on page 99.
3. From the pop-up menu, choose Set Active.

   The active engine moves to the top of the list in the Strategy Tree.
4. If the control engine is configured for Ethernet link redundancy, right-click the name of the control engine on the Strategy Tree again. From the pop-up menu, choose Use Primary IP Address or Use Secondary IP Address.

Removing a Control Engine's Association with a Strategy

If you no longer want to use a control engine with a strategy, you can remove its association with the strategy. This action does not delete the control engine's definition on your PC.

***CAUTION**: Do not delete the control engine from within the Select Control Engine dialog box. Doing so will delete it from the PC as well as the strategy.*

1. Make sure the strategy is open in Configure mode or Online mode.
2. On the Strategy Tree, right-click the name of the control engine you want to remove. From the pop-up menu, choose Delete.

   The control engine is no longer associated with your strategy.

Deleting a Control Engine from Your PC

If you are sure that a control engine will no longer be used with your PC, you can delete it using PAC Terminal. Deleting the control engine removes its definition only on the PC you are using.

1. Start PAC Terminal as follows:
   - In Windows 7 and Windows Vista, press the Windows Start key, and then click Programs > Opto 22 > PAC Project 9.6 > Tools > PAC Terminal.
   - In Windows 10 and Windows 8.1, press the Windows Start key, type **`PAC Terminal 9.6`** and then press the Enter key.
2. Right-click the name of the control engine in the list.

   CAUTION*: Make sure you are highlighting the right one. You cannot undo a deletion.*
3. From the pop-up menu, choose Delete.

   The control engine is no longer defined on the PC.

Inspecting Control Engines and the Queue

You may want to inspect or change control engines while you are running the strategy in Debug mode. See the following topics to view control engine information and the engine's message queue, either from PAC Control in Debug mode or from PAC Terminal:

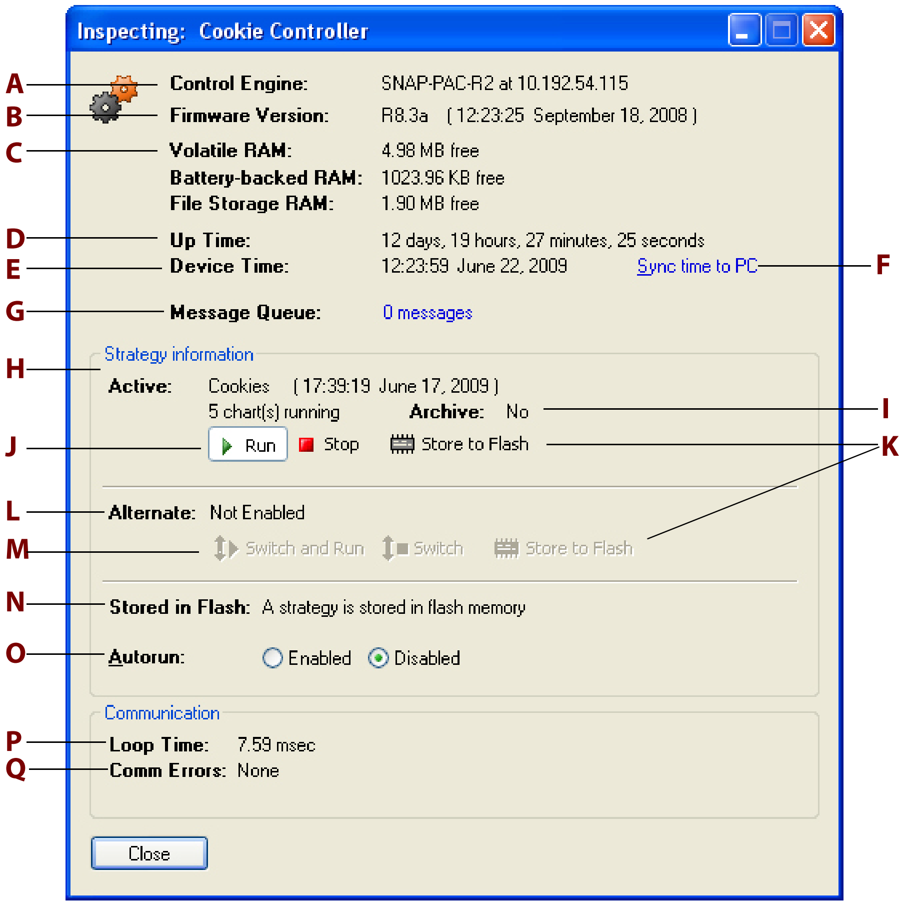
- "Inspecting Control Engines in Debug Mode" (below)
- "Viewing the Message Queue" on page 114
- "Inspecting Control Engines from PAC Terminal" on page 116

Inspecting Control Engines in Debug Mode

With the strategy running in Debug mode, choose Control Engine > Inspect.

You can also double-click the active engine (the first one under the Control Engines folder) on the Strategy Tree, or right-click the control engine and choose Inspect from the pop-up menu.

The Inspecting dialog box opens.



Control Engine: SNAP-PAC-R2 at 10.192.54.115
Firmware Version: R8.3a (12:23:25 September 18, 2008)
Volatile RAM: 4.98 MB free
Battery-backed RAM: 1023.96 KB free
File Storage RAM: 1.90 MB free
Up Time: 12 days, 19 hours, 27 minutes, 25 seconds
Device Time: 12:23:59 June 22, 2009 Sync time to PC
Message Queue: 0 messages

Active: Cookies (17:39:19 June 17, 2009)
5 chart(s) running Archive: No

Alternate: Not Enabled

Stored in Flash: A strategy is stored in flash memory

Autorun: Enabled [✓ Disabled]

Loop Time: 7.59 msec
Comm Errors: None

Here you see data relating to the control engine. If you are using Ethernet link redundancy in PAC Control Professional, remember that the information shown is for the IP address you chose before entering Debug mode. (See "Using Strategies with Link Redundancy" on page 109 for more information.)

A—*Control Engine.* Type of device the control engine is running on and the device's IP address.

B—*Firmware Version.* Version number of the firmware (kernel) loaded on the device, and the date the firmware was released.

C—*Memory Available.* Amount of memory (RAM) available on the control engine. For example, a SNAP R-series controller has a total of 16 MB of RAM on the control side, 2 MB of which is battery-backed. Volatile RAM shows the amount of total RAM available for use. Battery-backed RAM shows the amount available in battery-backed RAM, where persistent variables, variables initialized on download, the autorun flag, and the strategy archive are stored. File Storage RAM shows the space available in the control engine's file system See the *PAC Manager User's Guide* for more information about the file system. See also form 1646, the *SNAP PAC Memory Usage Technical Note*.

D—*Up Time.* Total time that the control engine has been running since powerup.

E—*Device Time.* Current date and time recorded on the control engine.

F—*Sync time to PC.* Click to synchronize the control engine's time and date to that of the PC running PAC Control.

G—*Message Queue.* Number of messages (information, warning, and error messages) encountered when attempting to run the strategy on the control engine (up to a maximum of 1000 messages). (See "Viewing the Message Queue" below.) Click to open the View Messages dialog box, listing the details of any information, warning, or error messages.

H—*Active.* Name of the strategy currently running on the control engine, the date and time it was downloaded, and the number of charts currently running.

I—*Archive.* Information about the strategy currently archived on the control engine.

J—*Run/Stop.* Start or stop the active strategy.

K—*Store to Flash.* Click to store either the active or alternate strategy to flash memory. Only one strategy at a time can be stored in flash memory. This button is enabled if there is a strategy stored in RAM, even if the strategy is currently stored in flash. See Burned in flash (**N**) on this dialog box.

L—*Alternate.* Name of the alternate strategy, and the date and time it was downloaded.

M—*Switch and Run/Switch.* Click Switch and Run to make the alternate strategy the active strategy *and* run it in the control engine. Click Switch to make the alternate strategy the active strategy without running it.

N—*Stored in flash.* Shows whether a strategy is currently stored in flash memory.

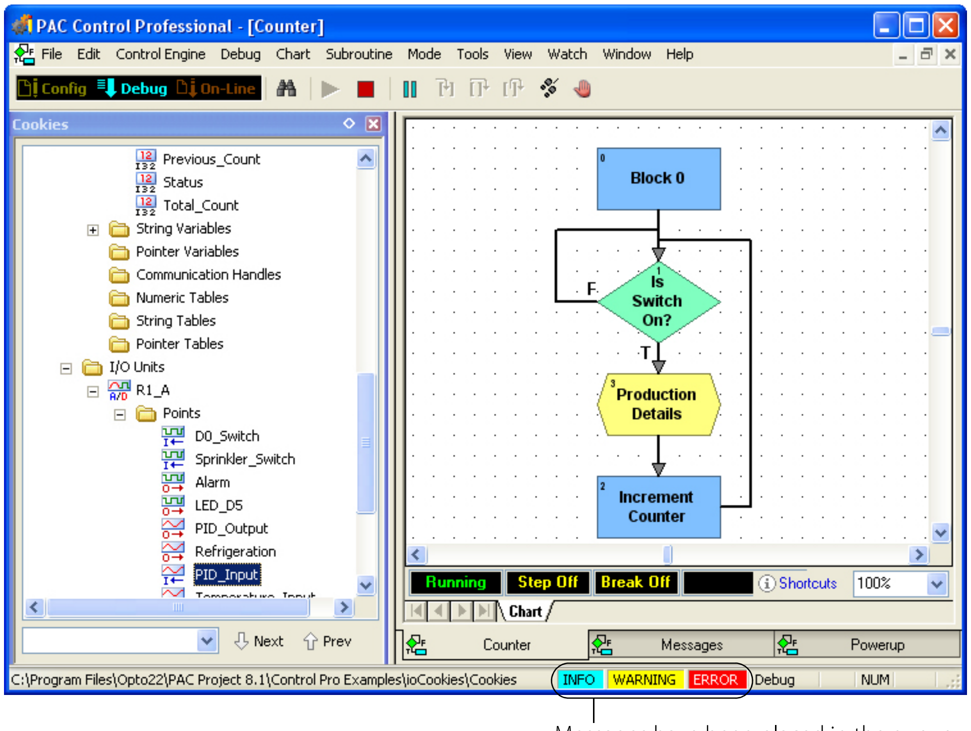
O—*Autorun.* Click to indicate whether the strategy should automatically run when the control engine is restarted. The strategy must be stored to flash to autorun. See Store to Flash (above) and "Saving a Strategy to Flash" on page 200.

P—*Loop Time.* Time required to gather the inspection data from the control engine (the average time taken for a single transaction).

Q—*Comm Errors.* Any communication errors.

Viewing the Message Queue

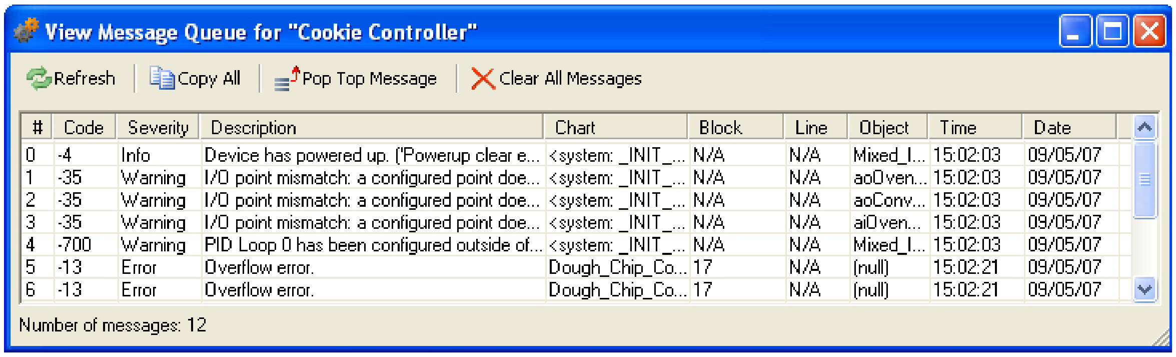
The message queue holds error, information, and warning messages. All may be helpful in troubleshooting. When a message is placed in the queue, a blue INFO, yellow WARNING, or red ERROR box appears in the PAC Control status bar, as shown in the following diagram.



Messages have been placed in the queue.

To see the message queue:

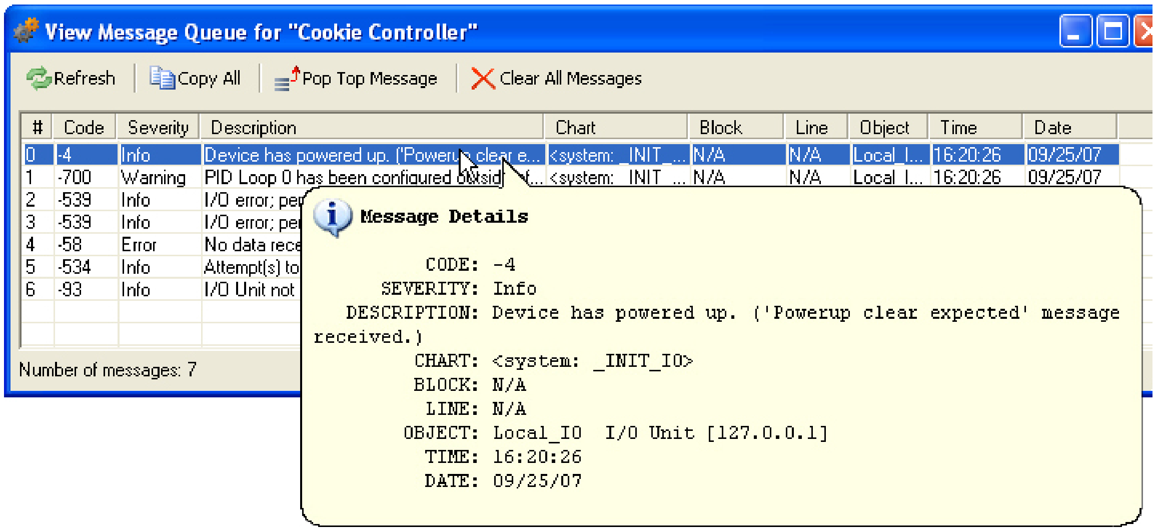
1. Click the INFO, WARNING, or ERROR box.



#	Code	Severity	Description	Chart	Block	Line	Object	Time	Date
0	-4	Info	Device has powered up. ('Powerup clear e...	<system: _INIT_...	N/A	N/A	Mixed_I...	15:02:03	09/05/07
1	-35	Warning	I/O point mismatch: a configured point doe...	<system: _INIT_...	N/A	N/A	aoOven...	15:02:03	09/05/07
2	-35	Warning	I/O point mismatch: a configured point doe...	<system: _INIT_...	N/A	N/A	aoConv...	15:02:03	09/05/07
3	-35	Warning	I/O point mismatch: a configured point doe...	<system: _INIT_...	N/A	N/A	aiOven...	15:02:03	09/05/07
4	-700	Warning	PID Loop 0 has been configured outside of...	<system: _INIT_...	N/A	N/A	Mixed_I...	15:02:03	09/05/07
5	-13	Error	Overflow error.	Dough_Chip_Co...	17	N/A	(null)	15:02:21	09/05/07
6	-13	Error	Overflow error.	Dough_Chip_Co...	17	N/A	(null)	15:02:21	09/05/07

Number of messages: 12

2. To see all the information in the column, drag the edge of a column heading.
3. To see all of the information for one message, double-click the message.



#	Code	Severity	Description	Chart	Block	Line	Object	Time	Date
0	-4	Info	Device has powered up. ('Powerup clear e...	<system: _INIT_...	N/A	N/A	Local_I...	16:20:26	09/25/07
1	-700	Warning	PID Loop 0 has been configured outsid [illegible]t...	<system: _INIT_...	N/A	N/A	Local_I...	16:20:26	09/25/07
2	-539	Info	I/O error; pe						
3	-539	Info	I/O error; pe						
4	-58	Error	No data rece						
5	-534	Info	Attempt(s) to						
6	-93	Info	I/O Unit not						

Number of messages: 7

```
Message Details

        CODE: -4
    SEVERITY: Info
 DESCRIPTION: Device has powered up. ('Powerup clear expected' message
received.)
       CHART: <system: _INIT_IO>
       BLOCK: N/A
        LINE: N/A
      OBJECT: Local_IO  I/O Unit [127.0.0.1]
        TIME: 16:20:26
        DATE: 09/25/07
```

4. To refresh the display with the latest messages, click Refresh.

5. To copy all messages in table format for transfer to another program such as Excel, click Copy All.

6. To delete the top (oldest) message on the list, click Pop Top Message.

7. To delete all messages, click Clear Messages.

8. Close the dialog box to return to the Inspect Control Engine dialog box.

Any changes you have made to the queue are reflected there.

Message Queue Information

Each message in the View Messages dialog box includes the following information:

Code. The message or error code number (see "List of Common Messages" on page 425) or *User* if the message was placed in the queue using the command Add Message to Queue.

Severity. Information, Warning, or Error.

Chart and Block. The chart and block being executed when the error occurred. If the error occurred someplace outside the strategy, for example when trying to connect to an I/O unit, Chart shows <system>. If the error occurred in a subroutine, Chart shows the chart that called the subroutine, and Block indicates the name of the subroutine plus the block number in the format `<sub name>.<block number>`. For example, error #4 above ("Cannot divide by zero") occurred in block 1 of the subroutine Variable_Increase_Notification, which was called by the Temperature_Control chart.

Line. If you are in Full Debug mode, the line being executed when the error occurred.

Object. The table, I/O unit, or other object affected by the message. In errors #1 and #2 above, the control engine was unable to communicate with I/O unit EIO_C. The unit's IP address is shown for easy reference.

Time and Date. When the error occurred.

Inspecting Control Engines from PAC Terminal

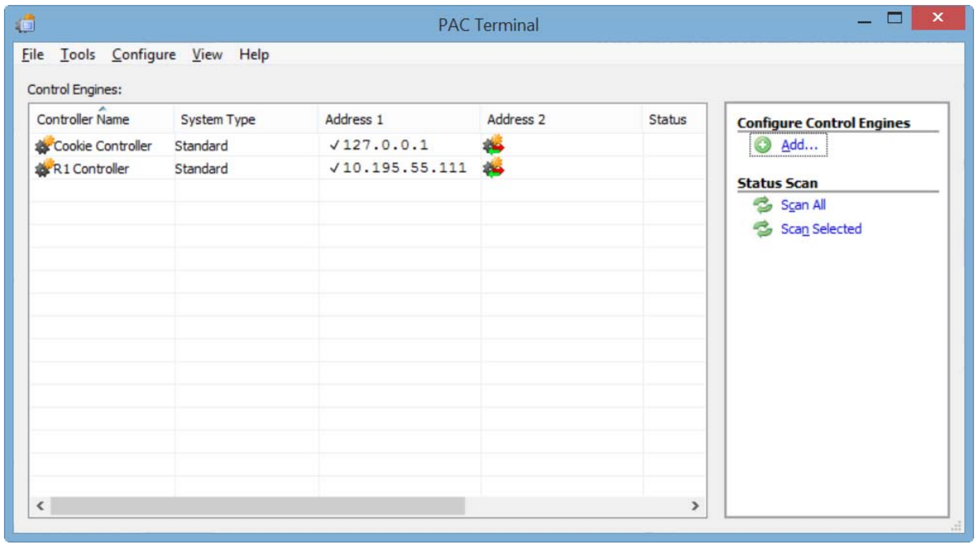
You can also inspect control engines from PAC Terminal.

1. If PAC Terminal isn't running, start it as follows:

– In Windows 7 and Windows Vista, press the Windows Start key, and then click Programs > Opto 22 > PAC Project 9.6 > Tools > PAC Terminal.

– In Windows 10 and Windows 8.1, press the Windows Start key, type **`PAC Terminal 9.6`** and then press the Enter key.

The PAC Terminal window appears.



Control Engines:

Controller Name	System Type	Address 1	Address 2	Status
Cookie Controller	Standard	✓127.0.0.1		
R1 Controller	Standard	✓10.195.55.111		

2. Double-click the control engine you want to see (or right-click it and choose Status from the pop-up menu).

The Inspect Control Engine dialog box appears. The dialog box is explained on page 111.

Archiving Strategies on the Control Engine

Archiving strategies on the control engine provides a backup in case original strategy files on the computer are lost. When you archive a strategy to the control engine on an Opto 22 controller, a zipped file is saved in battery-backed RAM. On a SoftPAC controller, a zipped file is saved on the PC's hard drive. In either case, if power to the control engine is lost, the archive is still there.

We recommend you archive both to the computer (during strategy development) and to the control engine (when the strategy is completed). Archiving to the control engine as well as the computer makes sure that an older strategy can always be found and updated, even after personnel changes occur and years pass. See also, "Archiving Strategies on the Computer" on page 201.

Archive files are date and time stamped, and zipped for compact storage. The archive file name on the control engine is in one of the following formats:

Path\Filename.Download.D02282000.T114351.zip
Path\Filename.Online.D02282000.T114351.zip

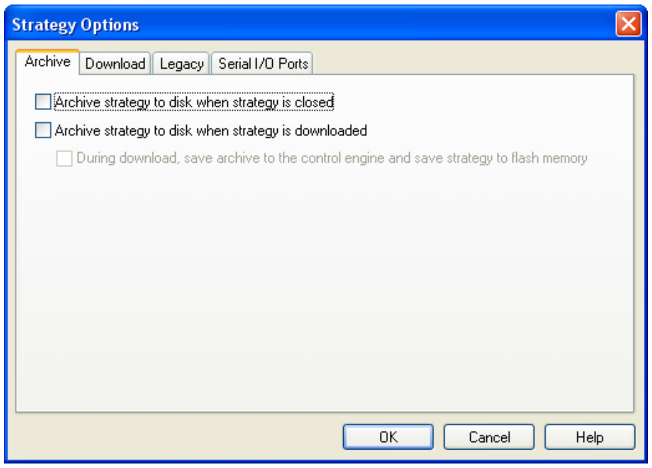
The date stamp (D) is in the format mm/dd/yyyy. In the examples above, the date is February 28, 2000. The time stamp (T) is in the format hh/mm/ss. In the examples above, the time is 51 seconds past 11:43 A.M.

Make sure the control engine has sufficient memory available to store the archive file. Since only one strategy at a time can be on the control engine, only the latest archive for that strategy is on the control engine. Other archives are erased during strategy download.

In addition to the archived strategy, the archive file also stores persistent variables and variables that are initialized on strategy download.

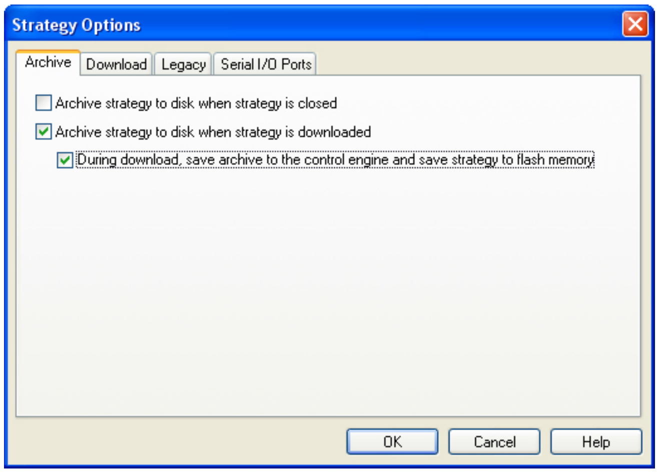
To archive a strategy to the control engine:

1. In PAC Control, choose File > Strategy Options.



☐ Archive strategy to disk when strategy is closed
☐ Archive strategy to disk when strategy is downloaded
☐ During download, save archive to the control engine and save strategy to flash memory

2. On the Archive tab in the Strategy Options dialog box, select “Archive strategy to disk when strategy is downloaded” Also select “During download, save archive to the control engine and save strategy to flash memory.”



☐ Archive strategy to disk when strategy is closed
✓ Archive strategy to disk when strategy is downloaded
✓ During download, save archive to the control engine and save strategy to flash memory

3. Click OK.

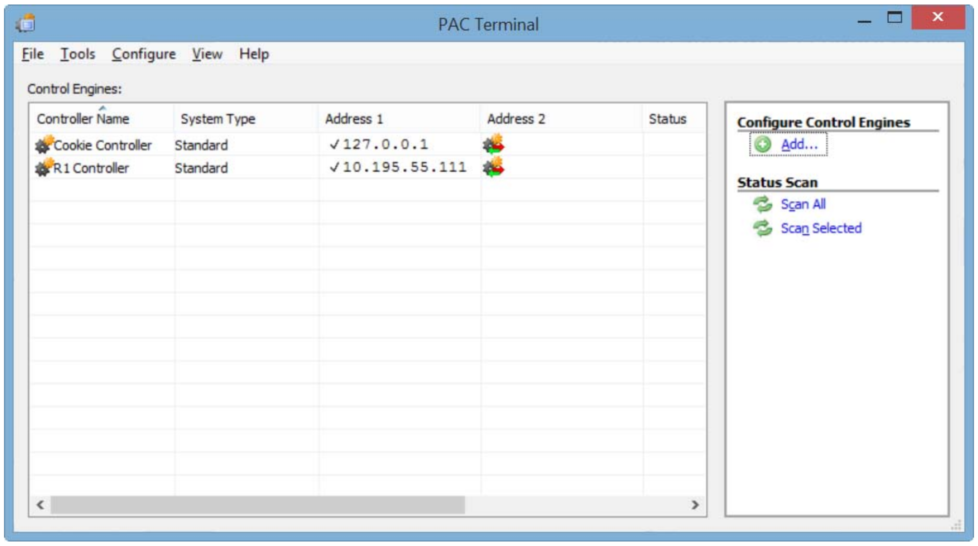
   The strategy will be archived to the computer and to the control engine when it is downloaded. Any archive already on the control engine will be replaced by the new archive. In addition, the strategy will be saved to flash memory, so it will still be available if power to the controller or SNAP Ultimate brain is turned off.

Restoring Archived Strategies from the Control Engine

If original strategy files are lost or damaged, you can use PAC Terminal to restore the strategy archive from the control engine to a computer.

1. If PAC Terminal isn't already open, start it as follows:
   - In Windows 7 and Windows Vista, press the Windows Start key, and then click Programs > Opto 22 > PAC Project 9.6 > Tools > PAC Terminal.
   - In Windows 10 and Windows 8.1, press the Windows Start key, type **`PAC Terminal 9.6`** and then press the Enter key.

   The PAC Terminal window appears.



Controller Name	System Type	Address 1	Address 2	Status
Cookie Controller	Standard	✓127.0.0.1		
R1 Controller	Standard	✓10.195.55.111		

2. Right-click the control engine and choose Upload > Strategy Archive from the pop-up menu.
3. In the Save PAC Control Strategy Archive As dialog box, navigate to the folder where you want to save the archive file. Keep the file name as it is, so you can see the date and time it was originally downloaded to the control engine. Click Save.

   A dialog box shows progress as the archive file is uploaded to the computer.
4. Navigate to the zipped archive file. Assuming you are using WinZip, double-click the file name. Highlight all files and click Extract. Extract them to the location you want.
5. When all files are extracted, double-click the .idb file to open the strategy. If necessary, re-link subroutines and files run before or after a strategy.

   Re-linking may be necessary because the directory structure in the zip file may not match what was originally on the computer. The zip file structure is as follows:

   Root (.idb, chart files, .inf)
       Subroutines
       Control engine files

Control_Engine_Name_1
  Before run file
  After run file
Control_Engine_Name_2
  Etc.

Moving Control Engines from One PC to Another

Sometimes it's necessary to transfer a strategy to another computer, such as when the computer has to be replaced, or when another PC will be used to run the control application. Instead of typing in each control engine on the new computer, you can open the ControllerNames.ini text file and merge the control engine names from a *source PC* with those on a *target PC*.

NOTE 1: Use this procedure only if you have PAC Control version R9.4a or newer. For versions prior to R9.4a, see the OptoKnowledgeBase article KB80603 on the Opto 22 website, http://opto22.com. However, the best method is to install the newest software and use the method described here.

NOTE 2: When PAC Control version R9.4a or newer is first used, it automatically migrates the control engine definitions from older versions. If you then use an older version of PAC Control to add another control engine, the new control engine will not automatically be available in the R9.4 or newer version, because it checks the older versions of control engine definitions only the first time it is run.

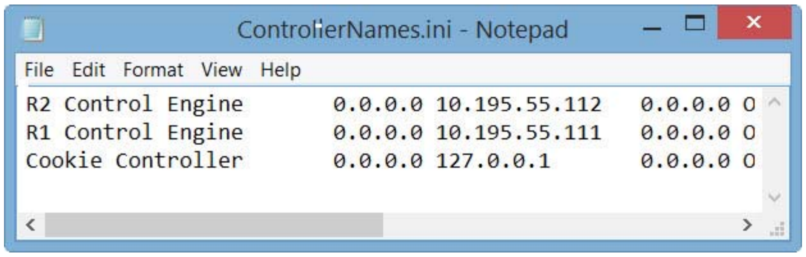
To move all the control engines from a source PC to a target PC:

1. On the source PC, locate the Controller.ini file. This file can be found in the following location:

   C:\ProgramData\Opto 22\PAC Project\ControllerNames.ini

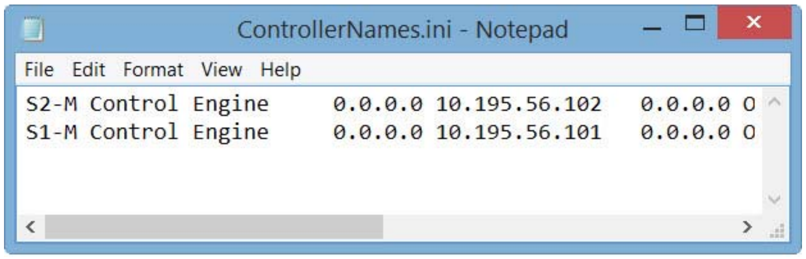
2. Send the file by email or other means so that it can be opened on the target PC.
3. On the target PC, open the ControllerNames.ini files for the source and target PCs with a text editor such Notepad.

Source



```
R2 Control Engine       0.0.0.0 10.195.55.112   0.0.0.0 0
R1 Control Engine       0.0.0.0 10.195.55.111   0.0.0.0 0
Cookie Controller       0.0.0.0 127.0.0.1       0.0.0.0 0
```

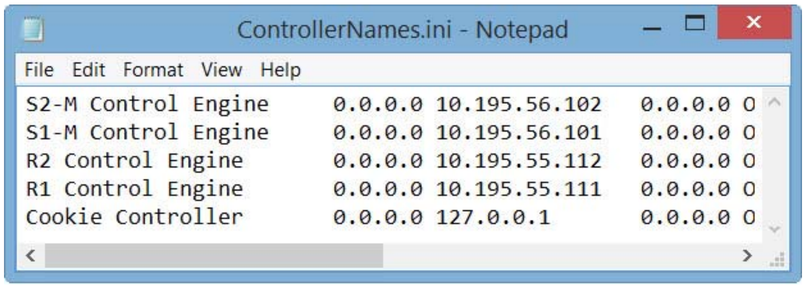
Target



```
S2-M Control Engine     0.0.0.0 10.195.56.102   0.0.0.0 0
S1-M Control Engine     0.0.0.0 10.195.56.101   0.0.0.0 0
```

4. Copy the list of control engines from the source PC file and paste them to the end of the list of control engines in the target PC file.

   Target and Source merged on target PC



```
S2-M Control Engine     0.0.0.0 10.195.56.102   0.0.0.0 0
S1-M Control Engine     0.0.0.0 10.195.56.101   0.0.0.0 0
R2 Control Engine       0.0.0.0 10.195.55.112   0.0.0.0 0
R1 Control Engine       0.0.0.0 10.195.55.111   0.0.0.0 0
Cookie Controller       0.0.0.0 127.0.0.1       0.0.0.0 0
```

5. Save the target ControllerNames.ini file on the target PC before closing it.

   All of the control engines in the merged file will now be available in PAC Control.

Downloading Files Using PAC Terminal

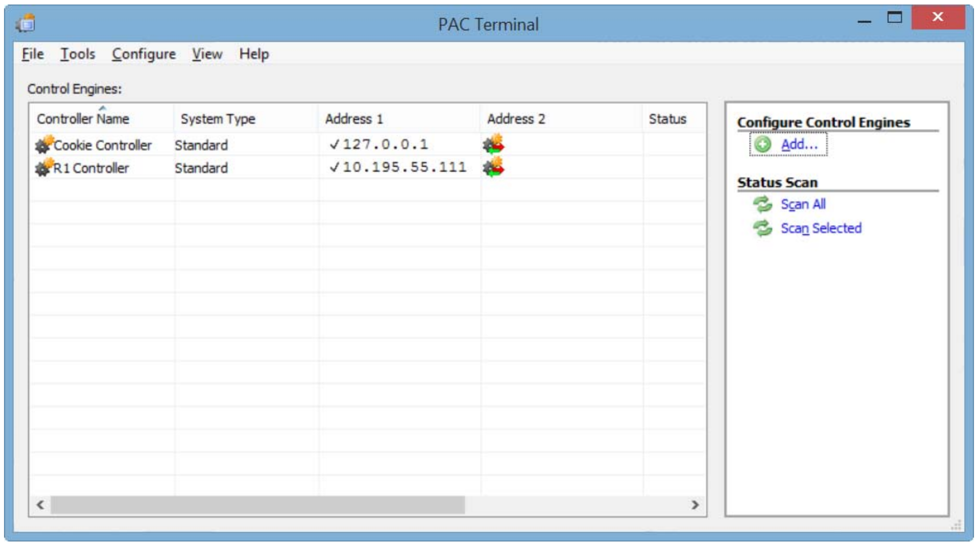
Using PAC Terminal, you can download PAC Control strategies or Forth files related to the strategy, such as library or initialization files, directly to a control engine, without having to open each program.

For information on using the brain's file system to store data and manipulate it within your strategy, see "Using the Control Engine's File System" on page 301.

NOTE: If you are downloading a PAC Control strategy that requires other files, be sure to download the files in the correct order (for example, library file, then strategy file, then initialization file). If you need to set initial values for individual table elements on strategy download only, see "Setting Initial Values in Variables and Tables During Strategy Download" on page 261.

1. If PAC Terminal isn't already open, start it as follows:
   – In Windows 7 and Windows Vista, press the Windows Start key, and then click Programs > Opto 22 > PAC Project 9.6 > Tools > PAC Terminal.
   – In Windows 10 and Windows 8.1, press the Windows Start key, type **`PAC Terminal 9.6`** and then press the Enter key.

The PAC Terminal window appears.



Controller Name	System Type	Address 1	Address 2	Status
Cookie Controller	Standard	✓127.0.0.1		
R1 Controller	Standard	✓10.195.55.111		

2. Right-click the control engine and choose Download > Forth File from the pop-up menu.
3. In the Download File dialog box, click Browse.
4. In the Open dialog box, locate the file you want to download. When the full path appears, click OK.

   If necessary to find the file, choose All Files from the Files of Type drop-down menu.

   The download begins, and a dialog box shows its progress.

6: Working with I/O

OPTO 22

Introduction

In addition to configuring a control engine to run your strategy, you also need to configure input/output hardware to do the work: turning things on, setting temperatures, monitoring controls, and so on.

This chapter shows you how to configure and work with I/O units, I/O points, and PID loops.

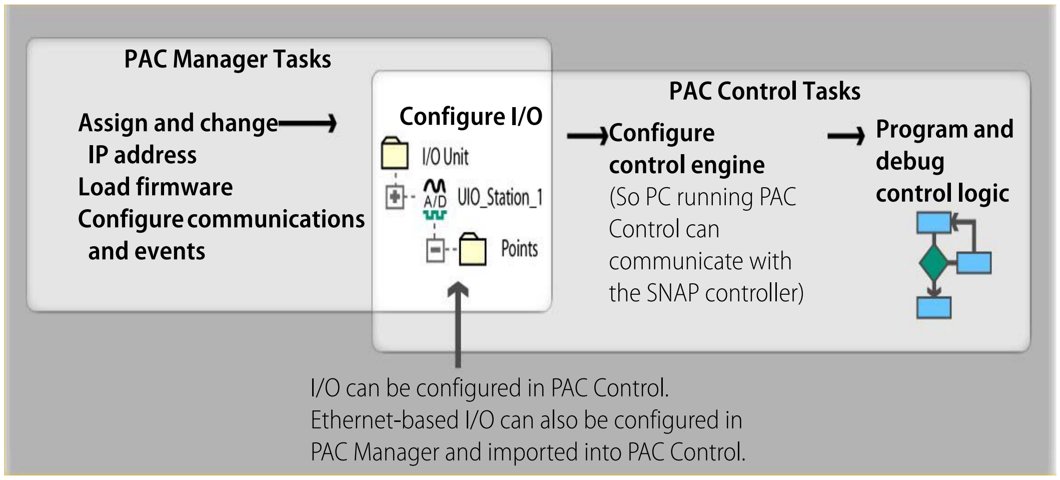
In this Chapter

Choosing a Configuration Tool 123
About I/O Units 126
Adding an I/O Unit 130
Adding I/O Points 136
Configuring a Serial Module 144
Changing I/O Unit and Point Configuration 144
Configuring PID Loops 150
Configuring Event/Reactions 162
Inspecting I/O in Debug Mode 171
Inspecting and Tuning PID Loops 179
Using Watch Windows for Monitoring 193

Choosing a Configuration Tool

Configuring I/O is one of your major planning steps in developing a PAC Control strategy. Generally it's best to configure all I/O units, points, and PID loops at once, before you start building flowcharts.

There are two tools you can use for configuration: PAC Control and PAC Manager. These two tools serve different purposes, but they overlap when it comes to configuring I/O. The image on the next page compares their functions.


PAC Manager Tasks
Assign and change IP address
Load firmware
Configure communications and events
Configure I/O
I/O Unit
UIO_Station_1
Points
PAC Control Tasks
Configure control engine
(So PC running PAC Control can communicate with the SNAP controller)
Program and debug control logic
I/O can be configured in PAC Control.
Ethernet-based I/O can also be configured in PAC Manager and imported into PAC Control.

I/O units and points must be configured to match the PAC Control strategy you will run. You can configure most I/O units and point functions either in PAC Control or in PAC Manager.

NOTE: You can configure E1 and E2 brain boards like any other I/O unit if you have E1/E2 firmware R1.2a (and higher) and PAC Project 9.5000 (and higher). Also, if a SNAP PAC controller communicates with the E1 or E2, the controller must have PAC firmware R9.5a (or higher) to use this simplified configuration method.

If you are not using these firmware and software versions (or if you prefer to use the previous method to reconfigure existing E1s or E2s), see Opto 22 form 1576, I/O Configuration for E1 and E2 Brain Boards.

For most I/O units, if you are already in PAC Control, configuration is easier there and you can use the loopback IP address forSNAP PAC R-series SNAP Ultimate I/O units controlling themselves. However, some functions for I/O units cannot be configured in PAC Control.

If you use PAC Manager, you can save your configuration to a file, load it to multiple I/O units at once, and use it for referencing points in OPC. However, you cannot use the loopback address in PAC Manager.

Choose your configuration tool based on what you need to do:

Use PAC Control for I/O configuration if	Use PAC Manager for I/O configuration if
• You have only one I/O unit or I/O unit configurations are different. • The strategy will run on SNAP PAC R-series SNAP Ultimate I/O units that are controlling themselves using the loopback IP address, 127.0.0.1 • You are using an Ethernet network for communications (or using a SNAP PAC controller with an SB brain). • The strategy handles all logic; you are not also configuring events and reactions on I/O units.	• You have multiple I/O units whose configurations are exactly the same or similar. • You are using a modem connection (PPP) or SNMP. • You are using event messages or email. • You are configuring events and reactions on the I/O unit in addition to strategy logic. • You are not using PAC Control.

Whichever tool you use for configuring I/O, be aware of the impact if you later change configuration. For example, if you configure I/O in PAC Manager, download the configuration file to I/O units, and then later add a point in PAC Control, remember that your configuration file doesn't contain that point.

If you use PAC Manager, follow instructions in Opto 22 form 1704, the *PAC Manager User's Guide*. When you have finished configuration and saved the configuration file, you can import it into PAC Control following the steps in "Importing I/O Configuration into PAC Control" below.

If you use PAC Control, follow the steps beginning with "Adding an I/O Unit" on page 130.

Importing I/O Configuration into PAC Control

If you have configured all I/O units, points, and PID loops in PAC Manager, follow these steps to import the configuration file into a PAC Control strategy.

1. Open the strategy in PAC Control. In the Strategy Tree, right-click the I/O units folder. From the pop-up menu, choose Import.
2. Navigate to the configuration file you created and saved in PAC Manager. Double-click it to open it.

   The configuration information is imported. You can expand the I/O units folder to see the imported units and their points.

   If you need to configure additional I/O units from within PAC Control, see "Adding an I/O Unit" on page 130. To tune PID loops, see "Inspecting and Tuning PID Loops" on page 179.

Copying I/O Configurations

If you have two strategies that use similar I/O units and points, you can export an I/O configuration from one strategy into a file, and then import it into the other strategy.

If you need similar configurations for several Ethernet-based I/O units, you can use PAC Manager to send it to multiple I/O units at once. (You cannot use PAC Manager for serial-based I/O units.) For more information on using PAC Manager, see Opto 22 form 1704, the *PAC Manager User's Guide*.

Creating the Configuration Export File

1. Open the strategy you are copying from in PAC Control. In the Strategy Tree, right-click the I/O Units folder and choose Export from the pop-up menu.

   The Export I/O Units to an Opto Tag Database dialog box appears.
2. Navigate to the location where you want to place the export file. Type the file name, and click Save.

   The export file is created. It is a comma-delimited ASCII file. If you wish, you can open it in Notepad or Excel.

Importing the Configuration File

When you import the I/O configuration file, it does not delete any I/O units or points that are already there. If the import file contains I/O units with the same names as those already in the strategy, you can choose whether to update them. Updating changes points that have the same name and adds new points, but does not delete points.

1. Open the strategy into which you want to import the I/O configuration.
2. In the Strategy Tree, right-click I/O Units and choose Import from the pop-up menu.
3. Navigate to the location of the export file you created. Highlight its name and click Open.

The I/O units and points are updated from the configuration file. To see them, click the plus sign next to the I/O Units folder on the Strategy Tree.

About I/O Units

In PAC Control, the term *I/O unit* usually refers to a mounting rack with an I/O processor (rack-mounted controller, brain, or brain board) and up to 16 I/O modules attached. The following table shows processors, racks, and I/O modules that can be used in PAC Control:

Processor Part Number	Compatible Racks	Max # Modules	Module types
The following I/O units are supported in PAC Control Basic and PAC Control Professional:			
SNAP-PAC-R1 SNAP-PAC-EB1	SNAP-PAC-RCK4 (SNAP-M16) SNAP-PAC-RCK8 (SNAP-M32) SNAP-PAC-RCK12 (SNAP-M48) SNAP-PAC-RCK16 (SNAP-M64)	4 8 12 16	SNAP analog, digital, and serial
SNAP-PAC-SB1			SNAP analog or digital (no serial)
SNAP-PAC-R2 SNAP-PAC-EB2 SNAP-UP1-M64 SNAP-ENET-S64	SNAP-PAC-RCK4 (SNAP-M16) SNAP-PAC-RCK8 (SNAP-M32) SNAP-PAC-RCK16 (SNAP-M64)	4 8 16	SNAP analog, digital, and serial (no high-speed digital)
SNAP-PAC-SB2			SNAP analog and digital (no serial)
G4EB2	G4PB32H PB32HQ	32	G4 simple digital Quad Pak simple digital
G4D32EB2	(Rack included)	32	G4 simple digital
G4D32EB2-UPG	G4D32RS	32	G4 simple digital
SNAP-PAC-R1-B SNAP-UP1-ADS SNAP-B3000-ENET SNAP-ENET-RTC	SNAP-B4M	4	SNAP analog, digital, and serial
	SNAP-B8M SNAP-B8MC SNAP-B8MC-P	8	
	SNAP-B12M SNAP-B12MC SNAP-B12MC-P	12	SNAP analog, digital, and serial (standard digital in positions 0–7 only; high-density digital in any position)
	SNAP-B16M SNAP-B16MC SNAP-B16MC-P	16	
SNAP-UP1-D64 SNAP-ENET-D64	SNAP-D64RS	16	SNAP digital (limited digital functions; no high-density digital)

Processor Part Number	Compatible Racks	Max # Modules	Module types
The following I/O units are supported in PAC Control Professional Only:			
B3000 (serial) B3000-B (serial)	SNAP-B4M	4	SNAP analog and standard digital (no high-density digital; no serial)
	SNAP-B8M SNAP-B8MC SNAP-B8MC-P	8	
	SNAP-B12M SNAP-B12MC SNAP-B12MC-P	12	SNAP analog and standard digital (no high-density digital; no serial; standard digital in positions 0–7 only)
	SNAP-B16M SNAP-B16MC SNAP-B16MC-P	16	
SNAP-BRS	SNAP-B8M SNAP-B8MC SNAP-B8MC-P	8	SNAP standard digital only (no serial or high-density digital. Max. 32 points)
G4D16R	Brick	16	G4 digital (max. 16 points)
G4D32RS	Brick	32	G4 digital (max. 32 points)
G4A8R	Brick	8	G4 analog (max 8 points
B100	PB16 G4PB16	16	G1, G4, or Quad Pak digital (max. 16 points)
B200	PB16A	16	G1 analog (max 16 points)

CAUTION: *Make certain you use a rack shown as compatible for the processor. Using any other rack will severely damage the brain or controller.*

For most serial-based brain boards, racks are limited to 16 points of either analog or digital I/O, so each I/O unit is either digital or analog.

E1 and E2 I/O units hold up to 16 points of digital (E1) or analog (E2) I/O.

G4EB2 I/O units are always 32 points of digital I/O. If used with G4 modules, inputs and outputs can be mixed anywhere on the rack; if used with Quad Pak modules, points are in groups of four inputs or four outputs.

Racks for most Ethernet-based brains, however, hold up to 64 points and can be either digital only or both analog and digital. If the rack attached to a SNAP Ethernet-based brain accommodates both analog and digital modules, the I/O unit includes both analog and digital modules.

In most cases the "I/O unit" you configure in PAC Control is the same as the physical I/O unit (rack, brain, and I/O modules). The entire rack of points is configured as one I/O unit, because that's how the points are addressed by the brain.

(PAC Control Pro) In two cases, however, the "I/O unit" in PAC Control does not exactly correspond to the physical I/O unit, because these brains address their I/O modules in a different way:

- **A B3000 or B3000-B serial** addresses up to four groups of 16 points on the largest rack. Each group of 16 points must be configured as a separate I/O unit, either analog or digital. For more information, see "SNAP Serial-Based (mistic) I/O Units." "SNAP Serial-Based (mistic) I/O Units" on page 128. Note that some PAC Control commands communicate with all the points on one I/O unit at once. For more information about these commands, see Opto 22 form 1776, *Optimizing PAC Project System Performance Technical Note*.

- **A G4D32RS brick** contains the equivalent of two 16-module units, and the brain board addresses them separately. When you configure it in PAC Control, notice that it is called a G4D16RS. Configure two G4D16RS I/O units for each brick. For more information, see "Non-SNAP Serial-Based (mistic) I/O Units" on page 129. *NOTE: The Ethernet-based G4D32EB2 I/O unit is not treated this way; it is called a G4EB2 and configured as a single 32-module unit.*

SNAP Serial-Based (*mistic*) I/O Units

Brains	Racks
B3000 or B300-B (serial) SNAP-BRS	SNAP-B series (SNAP-BRS is limited to 8-module racks)

(Pro only) SNAP B-series mounting racks can hold either 4, 8, 12, or 16 Opto 22 SNAP I/O modules. The serial B3000 or B3000-B brain can use any size B-series rack and both analog and digital modules. It cannot use serial or high-density digital modules. The SNAP-BRS uses 4-point SNAP digital modules only (no high-density digital) and only on an 8-module rack.

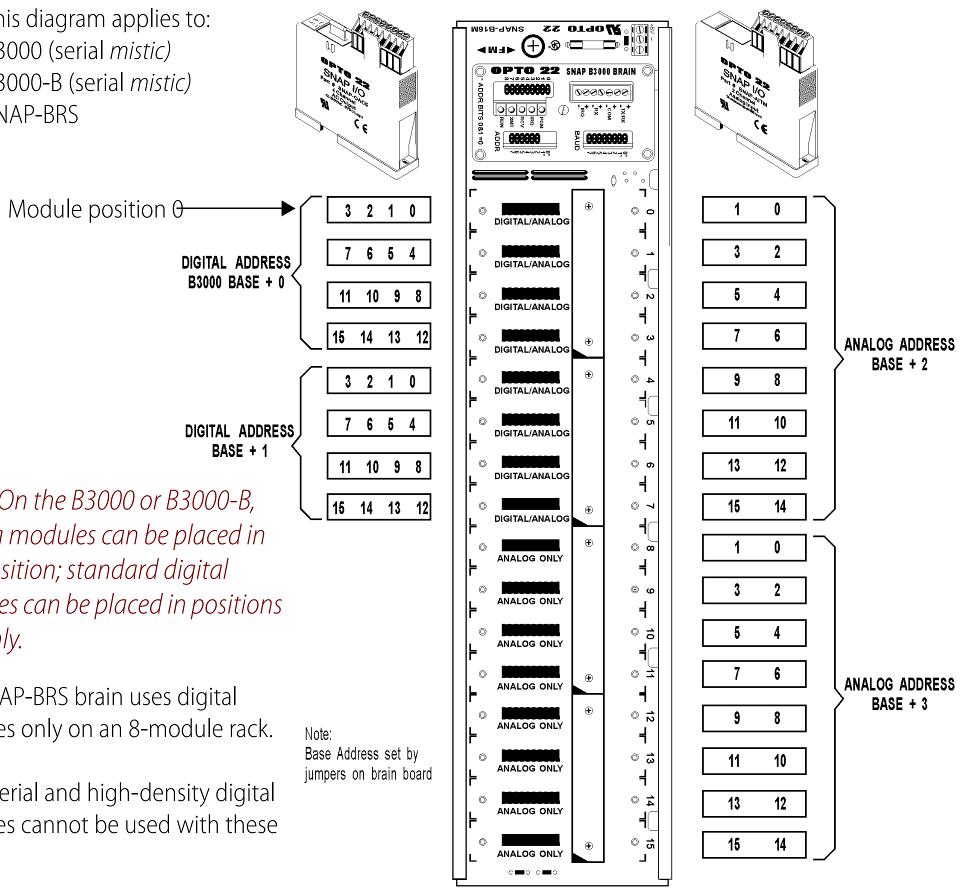
Analog modules can be placed in any position on these racks. For the larger racks, 4-point SNAP digital modules can be placed in positions 0–7 only. Each SNAP digital module contains four input or four output points. SNAP analog modules supported by the B3000 or B3000-B contain either one or two points.

Each point on the rack is numbered; when you configure the point or read or write to it, you reference it by its number. Each SNAP brain is really up to four logical brains in one: two digital 16-channel multifunctional brains plus (for a B3000 or B3000-B) up to two analog 16-channel multifunctional brains.

Jumpers on the SNAP brain set the base address, which must be zero, four, or a multiple of four. The base address is the first digital address, digital channels 0–15. The second digital address, digital channels 16–31, is the base address plus one. On a B3000 or B3000-B, the first analog address, analog channels 0–15, is the base address plus two; the second analog address, analog channels 16–31, is the base address plus three.

The following diagram shows the largest rack as an example. Note that two numbers are shown for each analog module. If you are using analog modules that contain only one point, ignore the upper point number.

This diagram applies to:
B3000 (serial *mistic)*
B3000-B (serial *mistic)*
SNAP-BRS



NOTE: On the B3000 or B3000-B, analog modules can be placed in any position; standard digital modules can be placed in positions 0–7 only.

The SNAP-BRS brain uses digital modules only on an 8-module rack.

SNAP serial and high-density digital modules cannot be used with these brains.

Non-SNAP Serial-Based (*mistic*) I/O Units

Brain boards	Racks
G4D16R G4D32RS G4A8R	Bricks (brain is built into rack)
B100 B200	PB16 G4PB16 PB16A

(Pro only) Because non-SNAP serial-based brain boards generally have a maximum of 16 modules on the rack, and each module has only one point, addressing points is simple. Point numbers correspond to module positions on the rack, 0–15, except for the following:

- If you are using Quad Pak modules, each module contains four points: module position zero contains points 0–3, module position one contains points 4–7, and so on.

- A G4D32RS brick contains the equivalent of two 16-module units, and the brain board addresses them separately. When you configure it in PAC Control, notice that it is called a G4D16RS. Configure two G4D16RS I/O units for each brick.

  NOTE: The Ethernet-based G4D32EB2 I/O unit is treated differently; it is called a G4EB2 and configured as a single 32-module unit. If you upgrade a G4D32RS rack with a G4D32EB2-UPG brain and cover, you will need to add it as a new G4EB2 I/O unit and then either move each point over from the old I/O units to the new one, or reconfigure points.

Adding an I/O Unit

In PAC Control, the term *I/O unit* usually refers to a mounting rack with an brain or brain board and I/O modules attached.

NOTE: If you have already configured I/O in PAC Manager, you can also add I/O units or points if necessary from within PAC Control, as you add commands to the blocks in your strategy.

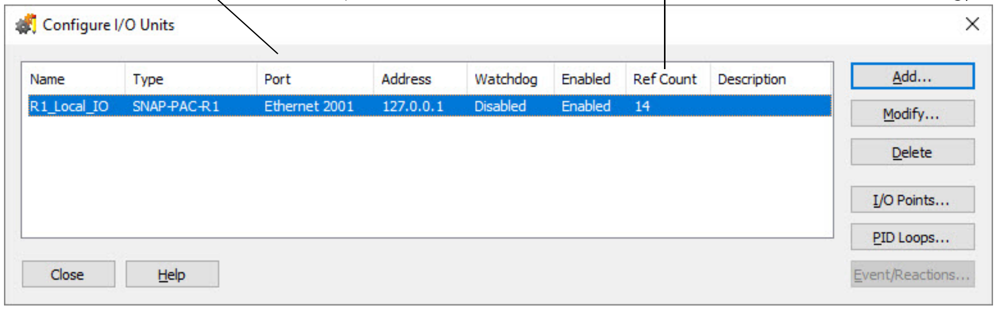
To enable legacy I/O units in PAC Control for a specific strategy, see "Legacy Options" on page 232.

1. Make sure the strategy is open and in Configure mode. On the Strategy Tree, double-click the I/O Units folder. (You can also select Configure > I/O from the menu bar.)

   The Configure I/O Units dialog box opens, showing all configured I/O units.

You configure most of the data in these columns when you add or modify an I/O unit.

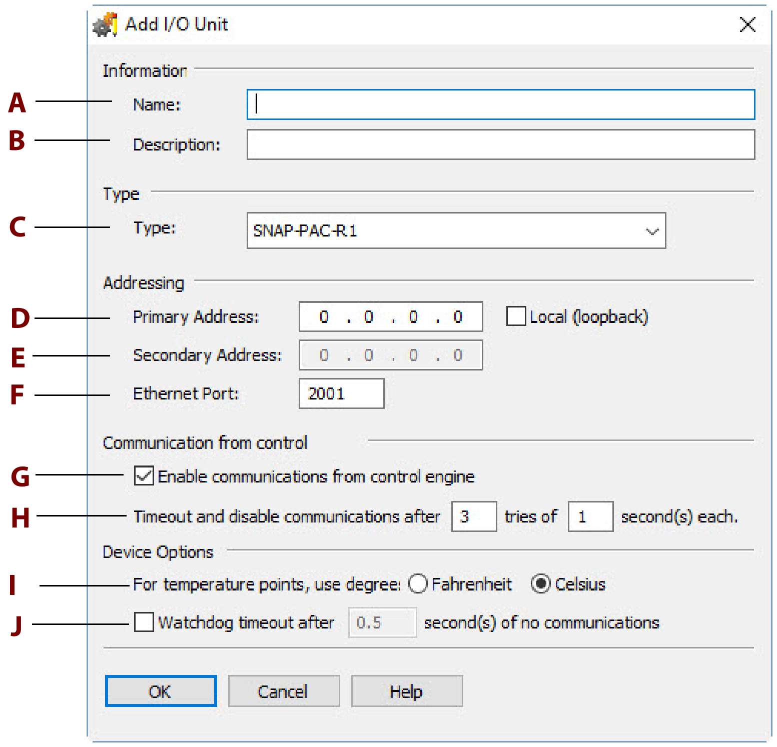
The Ref Count (Reference Count) is a field that you don't configure. It represents the number of I/O points configured on the I/O unit *plus* the number of times the I/O unit is referenced in the strategy.



Name	Type	Port	Address	Watchdog	Enabled	Ref Count	Description
R1_Local_IO	SNAP-PAC-R1	Ethernet 2001	127.0.0.1	Disabled	Enabled	14	

2. To configure a new I/O unit, click Add or double-click anywhere in the box below any listed units.

The Add I/O Unit dialog box appears.



Information

Name:

Description:

Type

Type: SNAP-PAC-R1

Addressing

Primary Address: 0 . 0 . 0 . 0 ☐ Local (loopback)

Secondary Address: 0 . 0 . 0 . 0

Ethernet Port: 2001

Communication from control

☑ Enable communications from control engine

Timeout and disable communications after 3 tries of 1 second(s) each.

Device Options

For temperature points, use degree: ○ Fahrenheit ● Celsius

☐ Watchdog timeout after 0.5 second(s) of no communications

3. Complete the fields as described in "Add I/O Unit Dialog Box" below.

Add I/O Unit Dialog Box

Use this dialog box to add a new I/O unit to the strategy or edit an existing unit.

For information on saving the I/O Unit Configuration, see "Inspecting I/O Units and Saving Settings to Flash" on page 172.

A—*Name.* Enter a name for the I/O unit. The name must start with a letter and may contain letters, numbers, and underscores. (Spaces are converted to underscores.)

B—*Description.* (Optional) Enter a description of the unit.

C—*Type.* When adding an I/O unit, select the type of I/O unit from the drop-down list. When upgrading an older I/O unit to a SNAP PAC brain, possible upgrades are available in the drop-down list.

D—*Primary Address.* For Ethernet I/O units, type the IP address of the brain attached to the I/O unit. If the unit is local (the same SNAP PAC brain on which the strategy will run), select Local (loopback). This sets the address to a loopback address, `127.0.0.1`. This tells the brain to talk to itself, so if you change the brain's IP address, you don't have to change the address for the I/O unit.

For serial units, type the unit's address (valid range is 0–255).

E—*Secondary Address.* (Optional, Pro only, Ethernet-based I/O units only) To designate a secondary I/O unit for communication if this I/O unit is unavailable, enter the secondary unit's IP address. Note that both I/O units use the same port number.

F—*Port.* For Ethernet-based I/O units, communication with the control engine is Ethernet, and the default port number is 2001. If you have changed this port for security purposes, also change it here. (See the controller's user guide for details.)

For serial-based I/O units, the communication port and parameters are shown for you. The Serial Address drop-down menu shows which addresses are currently being used. The port must be configured on the Strategy Options dialog box. Also, binary/CRC settings on your hardware are required. For more information, see "Changing the Baud Rate for Serial I/O Units" below.

G—*Enable Communications.* Select whether you want communication to the I/O unit enabled or disabled on startup. Enabled is the default. Disabling communication to an I/O unit is the same as disabling communication to all points on the unit.

H—*Timeout.* Enter the number of times the control engine should try to communicate with the I/O unit, and enter the length of time the control engine should wait for a response when communicating with this I/O unit. If after the first communication attempt there is no response within the timeout period, the control engine retransmits. If there is no response within the timeout period, it transmits again. This repeats according to the number of tries specified. If no response is received within the final timeout period, communication to the I/O unit is disabled. The default number of tries is 3. The default timeout period is 1 second.

In most cases, it is a good idea to start with the default values and only make changes if your system has problems using the defaults.

CAUTION*: If the timeout is long and the I/O unit is turned off or unreachable, the control engine could take quite a while to execute a command that talks to I/O. See also, "Tuning the I/O Unit Timeout Value for Ethernet I/O Units" on page 134.*

I—*Fahrenheit/Celsius.* (Analog and Mixed units only) Choose whether temperatures will be handled in Fahrenheit or Celsius.

J—*Watchdog.* Select whether you want a Watchdog on the unit (not available on remote simple I/O units). The default is No (disabled). If you select Yes, a new field appears; enter the Watchdog timeout value (in seconds). The default timeout is 0.5 seconds.

With a Watchdog, the I/O unit monitors activity on the port. If no communication is received for the specified interval, the unit assumes a Watchdog state. All selected outputs will then immediately go to a specified value, as configured in the Watchdog field of the Add Analog Point or Add Digital Point dialog boxes. (See "Adding I/O Points" on page 136.)

K—*Maximum Analog and High-Density Digital Scantime.* (Ethernet-based I/O units only) You can decrease the analog and high-density digital scan time to make sure the scanner isn't slowed or stopped by heavy communication on the network. The Default is 1000 msec. This feature is also available in PAC Manager where you can fine tune the scan time on the fly.

NOTE: Digital Scantime applies only to non-PAC I/O units. PAC I/O units use Digital Feature Resolution Value instead.

For more information on optimizing scanner performance, see "Optimizing Throughput" on page 98 in this guide.

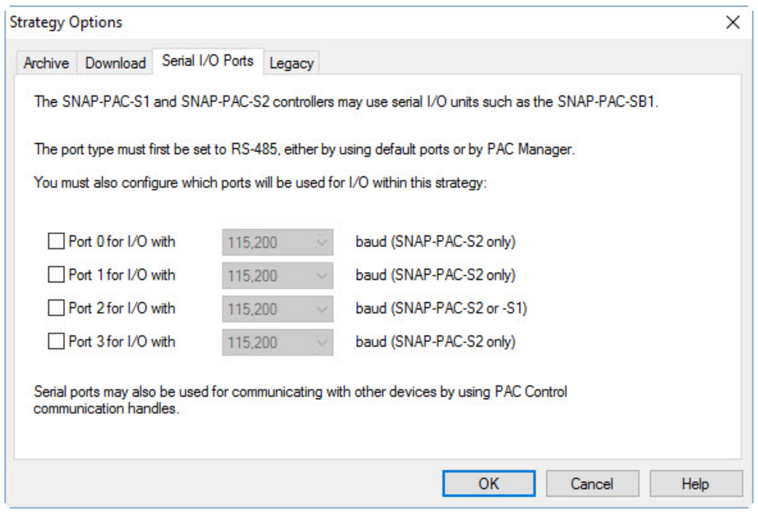
Changing the Baud Rate for Serial I/O Units

NOTE: This section is for PAC Control Basic users. If you have the Pro version, see "Changing the Baud Rate and Mode for Serial I/O Units" on page 133.

If the default baud rate shown in the Add I/O Unit dialog box is not correct, you can change it. This baud rate is for communication through the RS-485 port to serial I/O units. It does not affect other serial ports on the controller or communication handles using these ports.

NOTE: Once you have configured a serial port for use with an I/O unit, do not configure a communication handle on the same port.

1. Make sure the strategy is open and in Configure mode.
2. (*mistic* only) Make sure that the *mistic* I/O units and commands are enabled. If not, see "Enable mistic I/O Units and Commands" on page 234.
3. Choose File > Strategy Options, then click the Serial I/O Ports tab.



The SNAP-PAC-S1 and SNAP-PAC-S2 controllers may use serial I/O units such as the SNAP-PAC-SB1.

The port type must first be set to RS-485, either by using default ports or by PAC Manager.

You must also configure which ports will be used for I/O within this strategy:

☐ Port 0 for I/O with	115,200	baud (SNAP-PAC-S2 only)
☐ Port 1 for I/O with	115,200	baud (SNAP-PAC-S2 only)
☐ Port 2 for I/O with	115,200	baud (SNAP-PAC-S2 or -S1)
☐ Port 3 for I/O with	115,200	baud (SNAP-PAC-S2 only)

Serial ports may also be used for communicating with other devices by using PAC Control communication handles.

4. Select the correct port for the controller, and then choose the correct baud rate from the drop-down list and click OK.

   NOTE: If you are configuring a SNAP-PAC-S1 controller you must use port 2. If you are using a SNAP-PAC-S2, see Opto 22 form 1704, the PAC Manager User's Guide, for the default serial port assignments or to change a serial port assignment.

Changing the Baud Rate and Mode for Serial I/O Units

This section is for PAC Control Professional users. If you have the Basic version, see "Changing the Baud Rate for Serial I/O Units" on page 133.

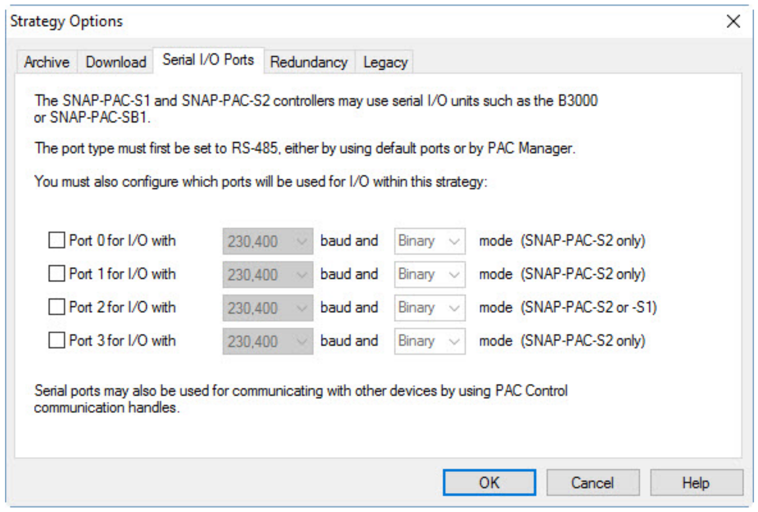
NOTE: Once you have configured a serial port for use with an I/O unit, do not configure a communication handle on the same port.

If the default baud rate shown in the Add I/O Unit dialog box is not correct, you can change it. This baud rate is for communication through the RS-485 port to serial I/O units. It does not affect other serial ports on the controller or communication handles using these ports.

In addition you can also choose Binary or ASCII mode. The controller will talk to all I/O units on that port in the same mode.

NOTE: There is no ASCII version of OptoMMP protocol (which is used on SNAP PAC SB brains), so if a port is set to ASCII, you won't be able to mix mistic *and SNAP PAC SB brains as you can in Binary mode.*

1. Make sure the strategy is open and in Configure mode.
2. (*mistic* only) Make sure that the *mistic* I/O units and commands are enabled. If not, see "Enable mistic I/O Units and Commands" on page 234.
3. Choose File > Strategy Options, then click the Serial I/O Ports tab.



The SNAP-PAC-S1 and SNAP-PAC-S2 controllers may use serial I/O units such as the B3000 or SNAP-PAC-SB1.

The port type must first be set to RS-485, either by using default ports or by PAC Manager.

You must also configure which ports will be used for I/O within this strategy:

☐ Port 0 for I/O with	230,400	baud and	Binary	mode	(SNAP-PAC-S2 only)
☐ Port 1 for I/O with	230,400	baud and	Binary	mode	(SNAP-PAC-S2 only)
☐ Port 2 for I/O with	230,400	baud and	Binary	mode	(SNAP-PAC-S2 or -S1)
☐ Port 3 for I/O with	230,400	baud and	Binary	mode	(SNAP-PAC-S2 only)

Serial ports may also be used for communicating with other devices by using PAC Control communication handles.

4. Select the correct ports for the controller, and then choose the correct baud rate and mode from the drop-down lists and click OK.

   If you are configuring a SNAP-PAC-S1 controller, you must use port 2 for serial I/O units. If you are using a SNAP-PAC-S2, see Opto 22 form 1704, the *PAC Manager User's Guide*, for the default serial port assignments or to change a serial port assignment.

   In addition, make sure to jumper the non-SNAP PAC brain for the correct setting, either Binary CRC (cyclic redundancy check) or ASCII CRC. Although at the brain level you can choose Binary/ASCII and CRC/Checksum independently, the Opto 22 drivers support only binary CRC or ASCII CRC.

   ASCII CRC is only for *mistic* brains. If a port is set for ASCII but has a SNAP-PAC-SB1/2 on it, the compiler will generate a warning.

Tuning the I/O Unit Timeout Value for Ethernet I/O Units

The default I/O Unit timeout value is 1 second. However, for best overall performance you can tune the value for your system.

A SNAP PAC controller automatically retries 2 times before disabling communication to an I/O unit that is unresponsive, and it uses the same timeout value for the first attempt and the 2 retries. So, with a 1 second timeout value and 2 retries, it takes 3 seconds before an unresponsive unit is taken offline. This means the chart or command that is trying to read or write to that I/O unit will be delayed by 3 seconds.

When using SNAP PAC Ethernet brains (SNAP-PAC-R1, SNAP-PAC-R1-B, SNAP-PAC-R2, SNAP-PAC-EB1, or SNAP-PAC-EB2) on a wired LAN (local area network), a timeout value of 0.1 seconds may be fine. When using a WAN (wide area network), a wireless network, the Internet, or you are not sure what the networking technology is, it may be best to use the default value of 1 second. However, if the controller's message queue shows loss of communication with Ethernet I/O units, it may be necessary to increase the timeout value.

For Advanced Users:

If you are an advanced user with a lot of Ethernet experience, you may want to determine normal response times from a brain using an Ethernet capture utility such as Wireshark® (www.wireshark.org). Look for the fastest response time, the *normal* or typical response time, and the longest response time.

Adjust the timeout value to be larger than the maximum observed response time. It is best not to use a timeout value that is too short because this will cause unnecessary communication retries and timeouts. We recommend making the timeout be at least 50% to 100% longer than the maximum observed response time. In addition, we recommend not using a timeout value that is less than 0.1 seconds. For example, if the value is 0.1 seconds, then communication to an unresponsive I/O unit will be disabled within 0.3 seconds. If the network connectivity is very slow, you may need to use a significantly longer timeout value such as 3 to 5 seconds or longer.

Changing Configured I/O Units

1. To change a configured I/O unit, make sure the strategy is open and in Configure mode.
2. Find the I/O unit's name on the Strategy Tree. Double-click it to open the Edit I/O Unit dialog box. See "Add I/O Unit Dialog Box" on page 131.
3. Make the necessary changes and click OK.

You can also change an I/O unit from the Configure I/O Units dialog box by double-clicking the unit's name or highlighting it and clicking Modify.

Deleting Configured I/O Units

You cannot delete an I/O unit if it has I/O points configured or if the I/O unit is referenced in a PAC Control command.

***CAUTION**: Be careful when deleting I/O units. You cannot undo a deletion.*

1. To delete a configured I/O unit, make sure the strategy is open and in Configure mode.
2. Find the I/O unit's name on the Strategy Tree. Right-click it and choose Delete from the pop-up menu.

   The I/O unit is deleted from the strategy.

You can also delete an I/O unit from the Configure I/O Units dialog box by highlighting the unit's name and clicking Delete.

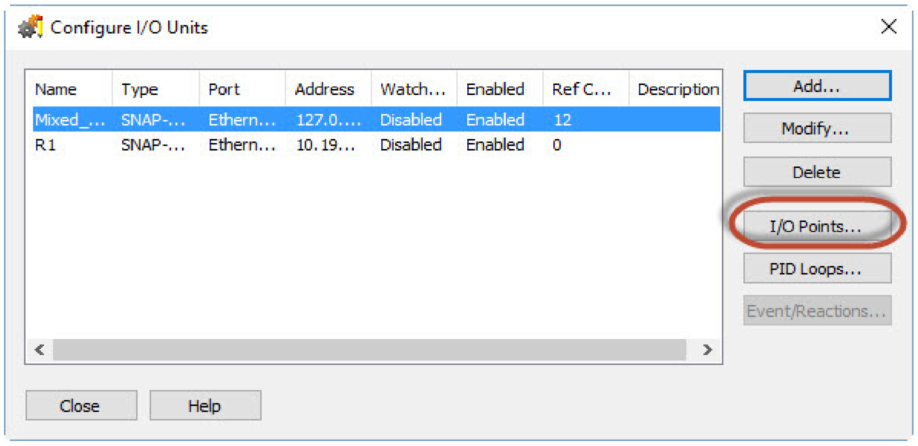
Adding I/O Points

Before you add an individual I/O point, such as a sensor or a switch, you must add the I/O unit the point is on. See "Adding an I/O Unit" on page 130.

Adding a Digital I/O Point

1. With the strategy open and in Configure mode, double-click the I/O Units folder (not the individual unit's icon) on the Strategy Tree.

   The Configure I/O Units dialog box appears.



Configure I/O Units

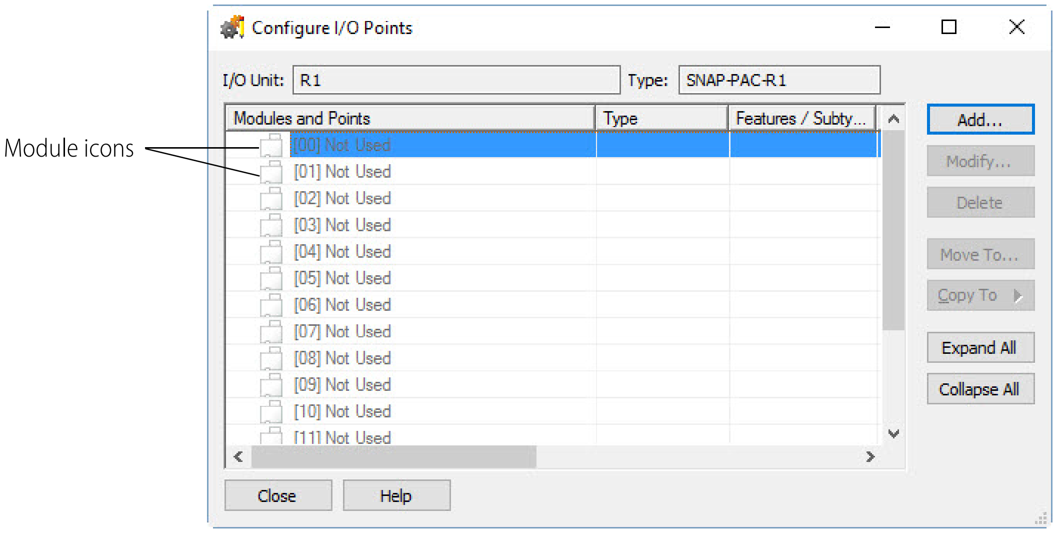
Name	Type	Port	Address	Watch...	Enabled	Ref C...	Description
Mixed_...	SNAP-...	Ethern...	127.0....	Disabled	Enabled	12	
R1	SNAP-...	Ethern...	10.19...	Disabled	Enabled	0	

2. Highlight the I/O unit the points are on, and click I/O Points.

   The Configure I/O Points dialog box appears.

3. **If you are using G1 or G4 single-point modules** (for example with a G4EB2 or E1 brain), double-click the point you want to use. Skip to step 11.

4. **If you are using SNAP I/O modules,** notice the module icons in the dialog box.



Configure I/O Points

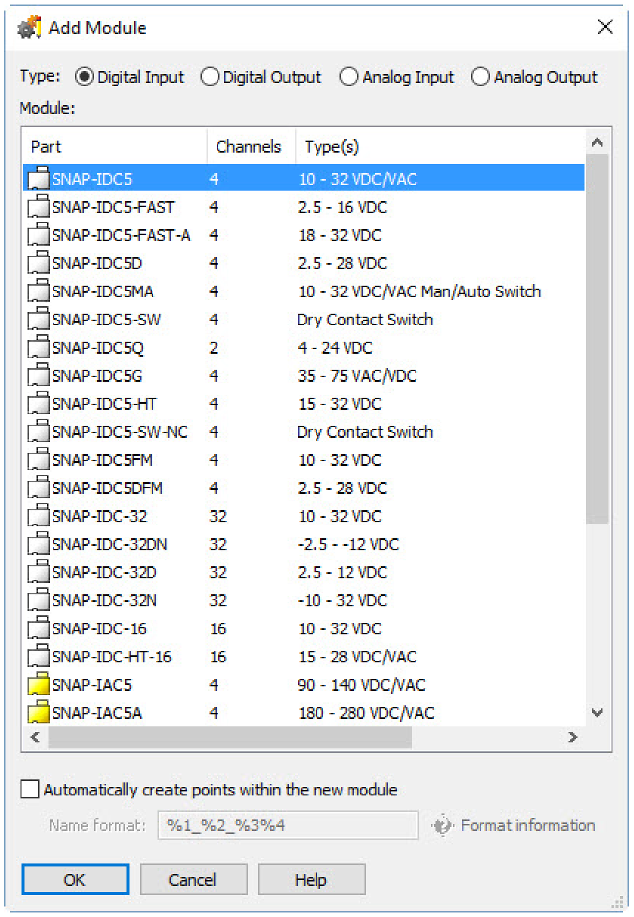
I/O Unit: R1 Type: SNAP-PAC-R1

Modules and Points	Type	Features / Subty...
[00] Not Used		
[01] Not Used		
[02] Not Used		
[03] Not Used		
[04] Not Used		
[05] Not Used		
[06] Not Used		
[07] Not Used		
[08] Not Used		
[09] Not Used		
[10] Not Used		
[11] Not Used		

NOTE: This example shows a SNAP PAC I/O unit, so both digital and analog modules can be configured on the same rack.

5. Highlight the number that represents the module's position on the rack, then click Add.

A list of available modules appears.


Add Module

Type: (● Digital Input) ○ Digital Output ○ Analog Input ○ Analog Output

Module:

Part	Channels	Type(s)
SNAP-IDC5	4	10 - 32 VDC/VAC
SNAP-IDC5-FAST	4	2.5 - 16 VDC
SNAP-IDC5-FAST-A	4	18 - 32 VDC
SNAP-IDC5D	4	2.5 - 28 VDC
SNAP-IDC5MA	4	10 - 32 VDC/VAC Man/Auto Switch
SNAP-IDC5-SW	4	Dry Contact Switch
SNAP-IDC5Q	2	4 - 24 VDC
SNAP-IDC5G	4	35 - 75 VAC/VDC
SNAP-IDC5-HT	4	15 - 32 VDC
SNAP-IDC5-SW-NC	4	Dry Contact Switch
SNAP-IDC5FM	4	10 - 32 VDC
SNAP-IDC5DFM	4	2.5 - 28 VDC
SNAP-IDC-32	32	10 - 32 VDC
SNAP-IDC-32DN	32	-2.5 - -12 VDC
SNAP-IDC-32D	32	2.5 - 12 VDC
SNAP-IDC-32N	32	-10 - 32 VDC
SNAP-IDC-16	16	10 - 32 VDC
SNAP-IDC-HT-16	16	15 - 28 VDC/VAC
SNAP-IAC5	4	90 - 140 VDC/VAC
SNAP-IAC5A	4	180 - 280 VDC/VAC

☐ Automatically create points within the new module

Name format: %1_%2_%3%4

6. Choose the module type and then the exact module from the lists.

7. To add points automatically, select "Automatically create points within the new module."

Use the "Name format" text field to create a format for the point names. Enter any combination of text and the following format codes:

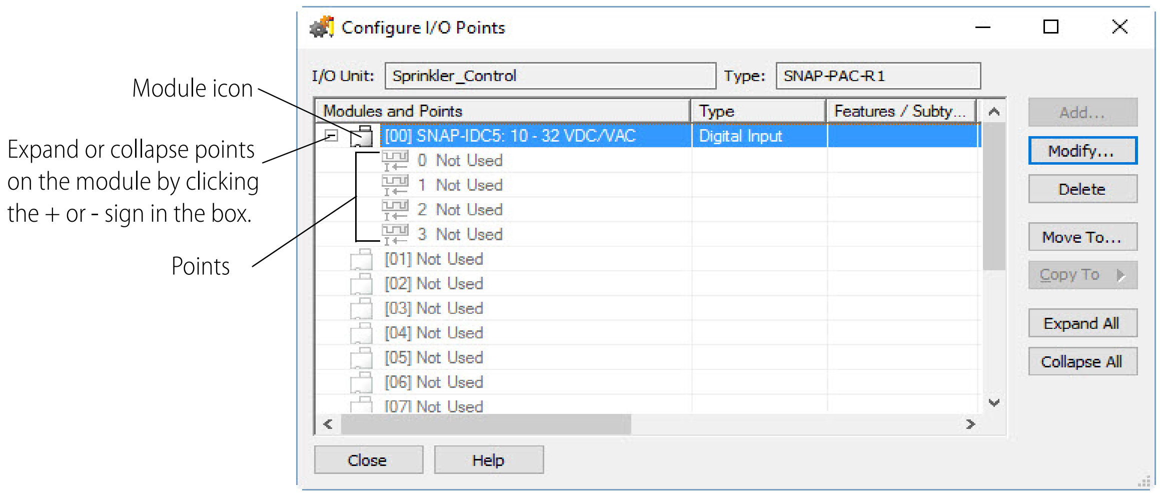
%1 = point type (for example, *ai* or *do*)
%2 = I/O unit name
%3 = module position
%4 = channel position
%5 = a unique identifier that is generated automatically

Example: %1_%2_%3%4 would expand to something like di_MyIoUnit_0801.
Example: %1ABC_%5 would expand to something like aoABC_473

8. Click OK.

9. In the Configure I/O Points dialog box, click the plus sign next to the new module to expand it.

Notice that the module icon is color-coded to reflect the type of module being configured: white for digital DC input, red for digital DC output, yellow for digital AC input, and black for digital AC output.



I/O Unit: Sprinkler_Control Type: SNAP-PAC-R1

Modules and Points	Type	Features / Subty...
[00] SNAP-IDC5: 10 - 32 VDC/VAC	Digital Input	
0 Not Used		
1 Not Used		
2 Not Used		
3 Not Used		
[01] Not Used		
[02] Not Used		
[03] Not Used		
[04] Not Used		
[05] Not Used		
[06] Not Used		
[07] Not Used		

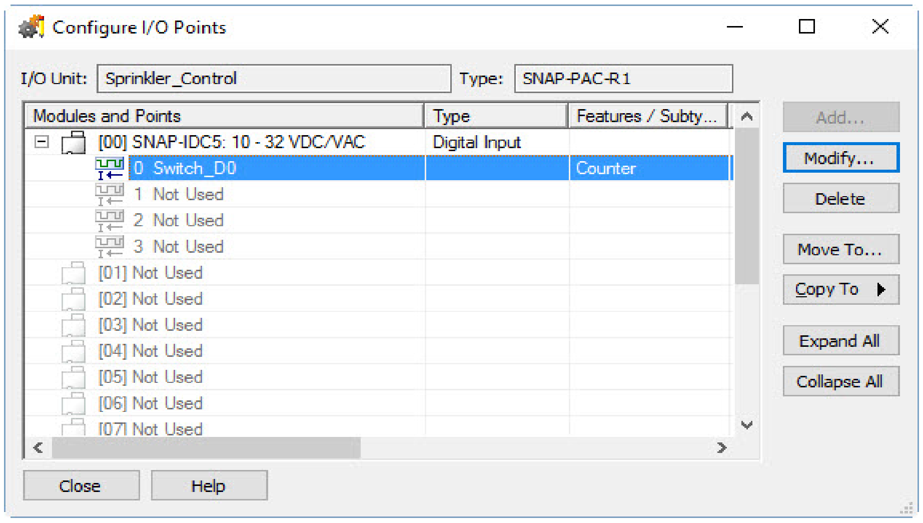
10. Highlight the point you want to configure and click Add.

11. Complete the fields as described in "Add Digital Point Dialog Box" below.

12. When you have completed the fields, click OK.

The new point appears in the list.

Here is an example of how it might look on a SNAP I/O unit:

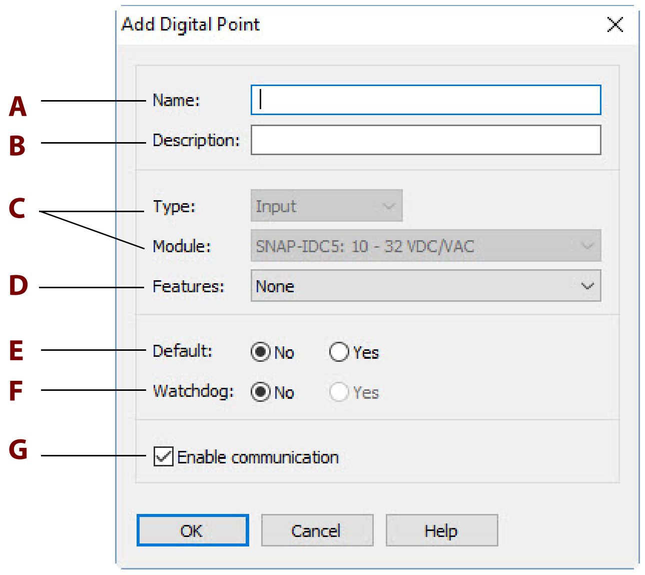


I/O Unit: Sprinkler_Control Type: SNAP-PAC-R1

Modules and Points	Type	Features / Subty...
[00] SNAP-IDC5: 10 - 32 VDC/VAC	Digital Input	
0 Switch_D0		Counter
1 Not Used		
2 Not Used		
3 Not Used		
[01] Not Used		
[02] Not Used		
[03] Not Used		
[04] Not Used		
[05] Not Used		
[06] Not Used		
[07] Not Used		

NOTE: If you need to add several similar points, see "Copying a Configured I/O Point" on page 147.

Add Digital Point Dialog Box



Name:
Description:
Type: Input
Module: SNAP-IDC5: 10 - 32 VDC/VAC
Features: None
Default: [✓ No] Yes
Watchdog: [✓ No] Yes
[✓] Enable communication

A—*Name.* Enter a name for the point. The name must start with a letter and may contain letters, numbers, and underscores. (Spaces are converted to underscores.)

B—*Description.* (Optional) Enter a description of the point.

C—*Type/Module.* For non-SNAP I/O units, choose the module type and the exact module from the drop-down lists. For a G4EB2, all 32 digital points are addressed as a single I/O unit; if you are using the G4EB2 with Quad Pak modules, make sure you configure each group of four points as either all inputs or all outputs.

For SNAP I/O units: Type and module are already filled in for you, as shown in the example above.

D—*Features.* To use a feature of the module, choose it from the drop-down list. You can configure some input modules with a counter, totalizer, or other feature. (Inputs automatically have both on-latches and off-latches.)

E—*Default.* To set a default state for the point when the strategy is run, click Yes and choose the state (Off or On). To leave the point in the state it was before, click No.

F—*Watchdog.* (Output modules only) To set a Watchdog, click Yes and choose On or Off from the drop-down list.

G—*Enable Communication.* Select whether you want communication to this I/O point enabled or disabled on startup. Enabled is the default.

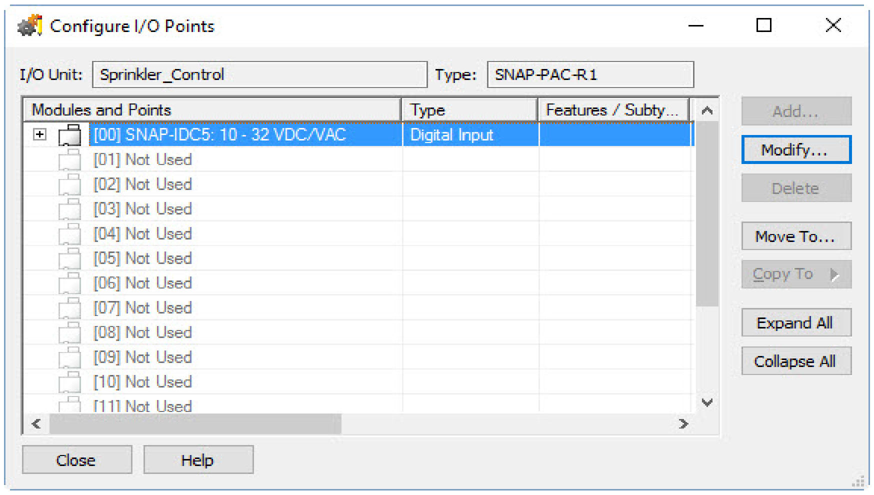
Adding an Analog I/O Point

1. With the strategy open and in Configure mode, double-click the I/O Units folder (not the individual unit's icon) on the Strategy Tree.
2. In the Configure I/O Units dialog box, highlight the I/O unit the points are on, and click I/O Points.

The Configure I/O Points dialog box appears.

3. (Pro only) **If you are using a non-SNAP I/O unit (*not* including the B3000 or B3000-B),** double-click the channel you want to use. Skip to step 8.

4. **If you are using a SNAP I/O unit (including the B3000 or B3000-B)**, notice the module icons in the Configure I/O Points dialog box.



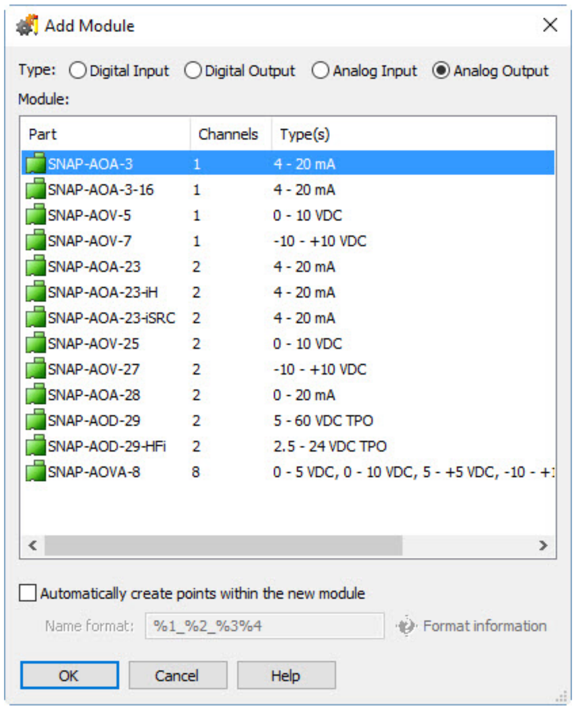
I/O Unit: Sprinkler_Control   Type: SNAP-PAC-R1

Modules and Points	Type	Features / Subty...
[00] SNAP-IDC5: 10 - 32 VDC/VAC	Digital Input	
[01] Not Used		
[02] Not Used		
[03] Not Used		
[04] Not Used		
[05] Not Used		
[06] Not Used		
[07] Not Used		
[08] Not Used		
[09] Not Used		
[10] Not Used		
[11] Not Used		

NOTE: In this example, a digital module has already been added in position zero. This example shows a SNAP PAC I/O unit, so both digital and analog modules can be configured on the same rack.

5. Highlight the number of the analog module's position on the rack, then click Add.

6. Choose the module type (Analog Input or Analog Output).



Type: ( ) Digital Input ( ) Digital Output ( ) Analog Input [✓ Analog Output]

Module:

Part	Channels	Type(s)
SNAP-AOA-3	1	4 - 20 mA
SNAP-AOA-3-16	1	4 - 20 mA
SNAP-AOV-5	1	0 - 10 VDC
SNAP-AOV-7	1	-10 - +10 VDC
SNAP-AOA-23	2	4 - 20 mA
SNAP-AOA-23-iH	2	4 - 20 mA
SNAP-AOA-23-iSRC	2	4 - 20 mA
SNAP-AOV-25	2	0 - 10 VDC
SNAP-AOV-27	2	-10 - +10 VDC
SNAP-AOA-28	2	0 - 20 mA
SNAP-AOD-29	2	5 - 60 VDC TPO
SNAP-AOD-29-HFi	2	2.5 - 24 VDC TPO
SNAP-AOVA-8	8	0 - 5 VDC, 0 - 10 VDC, 5 - +5 VDC, -10 - +1

☐ Automatically create points within the new module

Name format: %1_%2_%3%4

7. To add points automatically, select "Automatically create points within the new module."

Use the “Name format” text field to create a format for the point names. Enter any combination of text and the following format codes:

%1 = point type (for example, *ai* or *do*)

%2 = I/O unit name

%3 = module position

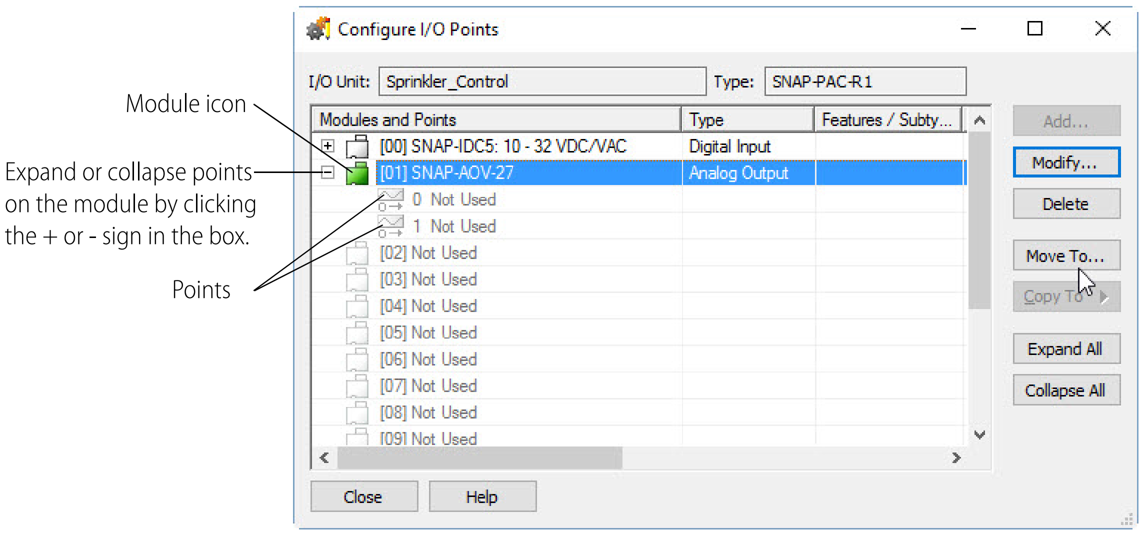
%4 = channel position

%5 = a unique identifier that is generated automatically

Example: %1_%2_%3%4 would expand to something like di_MyIoUnit_0801.

Example: %1ABC_%5 would expand to something like aoABC_473

8. In the Configure I/O Points dialog box, click the plus sign next to the new module to expand it. Notice that the module icon is color-coded to reflect the type of module being configured: blue for analog input, green for analog output.



I/O Unit: Sprinkler_Control Type: SNAP-PAC-R1

Modules and Points	Type	Features / Subty...
[00] SNAP-IDC5: 10 - 32 VDC/VAC	Digital Input	
[01] SNAP-AOV-27	Analog Output	
0 Not Used		
1 Not Used		
[02] Not Used		
[03] Not Used		
[04] Not Used		
[05] Not Used		
[06] Not Used		
[07] Not Used		
[08] Not Used		
[09] Not Used		

9. Highlight the point you want to configure and click Add.

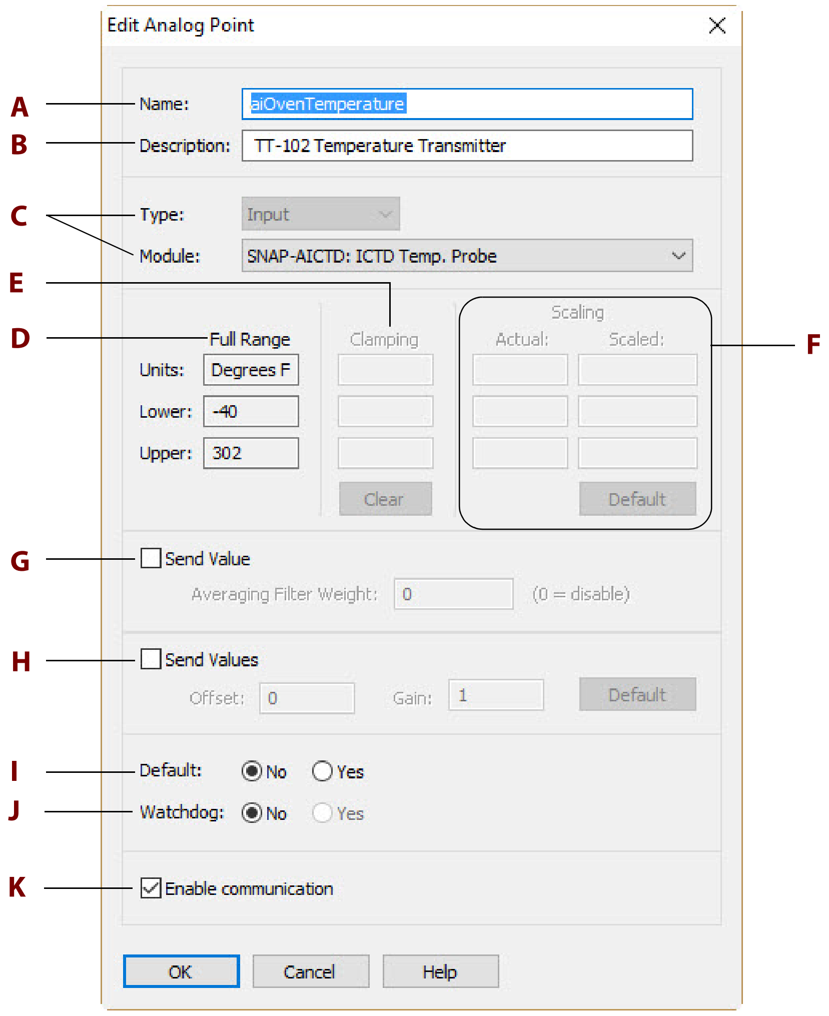
10. Complete the fields as described in “Add Analog Point Dialog Box” below.

11. When you have completed the fields, click OK.

The new point is added.

NOTE: If you need to add several similar points, see “Copying a Configured I/O Point” on page 147.

Add Analog Point Dialog Box



Edit Analog Point

Name: aiOvenTemperature

Description: TT-102 Temperature Transmitter

Type: Input

Module: SNAP-AICTD: ICTD Temp. Probe

	Full Range	Clamping	Scaling Actual:	Scaling Scaled:
Units:	Degrees F			
Lower:	-40			
Upper:	302			

☐ Send Value
Averaging Filter Weight: 0 (0 = disable)

☐ Send Values
Offset: 0 Gain: 1

Default: [✓ No] Yes

Watchdog: [✓ No] Yes

☑ Enable communication

A—*Name.* Enter a name for the point. The name must start with a letter and may contain letters, numbers, and underscores. (Spaces are converted to underscores.)

B—*Description.* (Optional) Enter a description of the point.

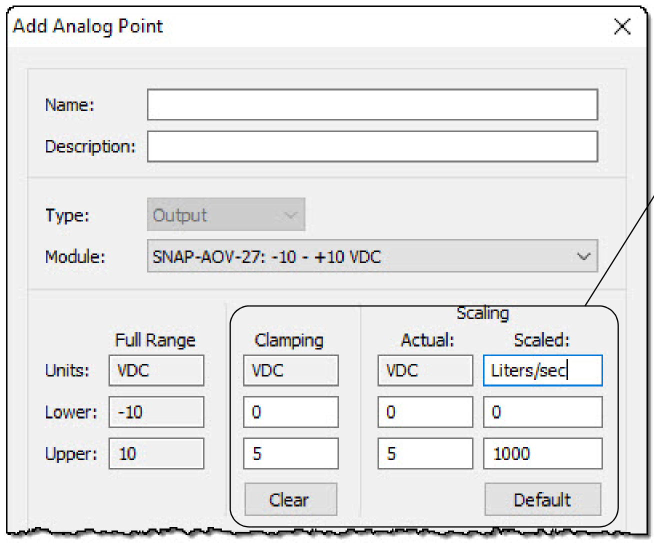
C—*Type/Module.* For non-SNAP I/O units, choose the module type and the exact module from the drop-down lists.

For SNAP I/O units: Type and module are already filled in for you. You may be able to choose a different range from the drop-down list.

D—*Full Range.* Full range and units for this module.

E—*Clamping.* (Outputs only—optional) Enter upper and lower clamp if necessary to limit output to the device attached to the point, no clamp is applied. To empty the fields, click Clear.

F—*Scaling.* Use to assign custom units and values to the module. For example, you could scale the readings of a -10 to +10 VDC input point to measure its input as zero liters per second when the real-world reading is zero VDC, and 1000 liters per second when the real-world reading is 5 VDC. This example would look like this:



Type: Output
Module: SNAP-AOV-27: -10 - +10 VDC

	Full Range	Clamping	Actual:	Scaled:
Units:	VDC	VDC	VDC	Liters/sec
Lower:	-10	0	0	0
Upper:	10	5	5	1000

In this example, units are changed to liters/second and lower and upper values are changed. Although the module has an output of -10 to +10 volts, the device attached to the point outputs only 0–5 volts. Scaling reflects the device's range.
In this case Clamping protects the device by ensuring that out-of-range voltage will never be sent to it.

Custom scaled values are floating point values.

- **Analog input points:** Analog inputs typically have under-range and over-range capability, which means you can specify a lower or upper value beyond the standard value. Also note that with firmware R8.0a and higher, you can use regular or inverted scaling. Inverted scaling means the lower value is greater than the upper value; inverted scaling can accommodate a sensor, for example, that is wired to the SNAP module in a way that produces a negative current or voltage.
- **Analog output points:** Analog outputs do not have under-range or over-range capability. Also, you cannot use inverted scaling. The upper value must be greater than the lower value.

Thermocouple values are not linear and cannot be scaled.

To return the units and upper/lower values to the defaults for the module, click Default.

G—*Send Value (Filter Weight).* (Inputs only—optional) To set the average filter weight for an analog output point, check this box and enter the value. A filter weight smooths analog input signals that are erratic or change suddenly. The formula used for filtering is Y = (X - Y)/W + Y, where Y is the filtered value, X is the new unfiltered value, and W is the filter weight. The larger the filter weight you enter, the smoother the analog signal. Filtering is applied to values in engineering units, including minimum and maximum values. It does not apply to values in counts.

H—*Send Values (Offset and Gain).* (Inputs only—optional) To set offset and gain for the analog output point, check this box and enter values. If you don't know what values to enter, you can use PAC Manager to calibrate offset and gain for the point. See the *PAC Manager User's Guide* for steps.

I—*Default.* To set the initial state of the controller's interval value (IVAL) for this I/O point when the strategy is run, click Yes and define the value. To leave the IVAL at its last state, click No.

If communication to the point is disabled, only the internal values (IVALs) are accessed or updated by the control engine. The real-world external values (XVALs) for inputs don't change because they are controlled by external devices. The IVALs for inputs are set and will be overwritten by the XVALs unless communication to the I/O unit or point is disabled.

The default (value) is available for both inputs and outputs. It sets the IVAL for the point in the strategy in the controller when the strategy is run. This can be useful when the I/O unit is not accessible or communication to the I/O unit is intentionally disabled when the strategy first starts. When communication to an I/O unit is disabled, any I/O-related commands in the strategy will act only on the I/O point IVALs because the controller is not able to access the XVALs at the I/O unit when communication to that I/O unit is disabled.

Once communication to the I/O unit is enabled and the I/O unit is accessible, then the first command to access that point will update the IVAL. For example, if you use the Move command to write a different value to an output, both the IVAL and XVAL are updated with the new value. If you use a Move command to read an input, the XVAL is moved to the IVAL as part of reading the input point.

J—*Watchdog.* (Outputs only) To set a Watchdog on this point, click Yes, and define the value to be assigned to the output if the Watchdog is triggered. A Watchdog is triggered if no communication activity is detected on the bus for the amount of time specified in the Watchdog field of this point's I/O unit. For no Watchdog, click No.

K—*Enable.* Select whether you want communication to this I/O point enabled or disabled on startup. Enabled is the default.

Configuring a Serial Module

Use PAC Manager to configure SNAP serial communication modules, including the PROFIBUS-DP and CAN modules. Serial modules can be used with SNAP Ethernet-based I/O units only. For more information, see Opto 22 form 1704, the *PAC Manager User's Guide*.

Changing I/O Unit and Point Configuration

You can change configured I/O units and points as follows:

- If you replace a SNAP brain or rack-mounted controller, you can often change the entire I/O unit at once. This feature does not apply to serial B3000 or B3000-B brains. See the list of allowed replacements in "Replace a SNAP Brain or Rack-mounted Controller" on page 145.
- You can move an I/O point or an entire I/O module to a different position on the same or a different I/O unit. See "Moving a Configured I/O Module or Point" on page 146.
- You can fill in empty points on a module or an I/O unit, or copy a point to another position. See "Copying a Configured I/O Point" on page 147.
- You can also change a point you've configured (page 150) or delete it (page 150).

Replace a SNAP Brain or Rack-mounted Controller

In most cases you can change the entire I/O unit type when you replace any of the following brains or rack-mounted controllers as shown in the table below, and you won't need to reconfigure existing I/O points.

However, if you're replacing an R1 or EB1 with an R2 or EB2, and any points on the I/O unit have the digital features Counter, On Pulse, Off Pulse, Frequency, or Period, you'll need to change those I/O points. You'll see a warning about this when you try to change the I/O unit type.

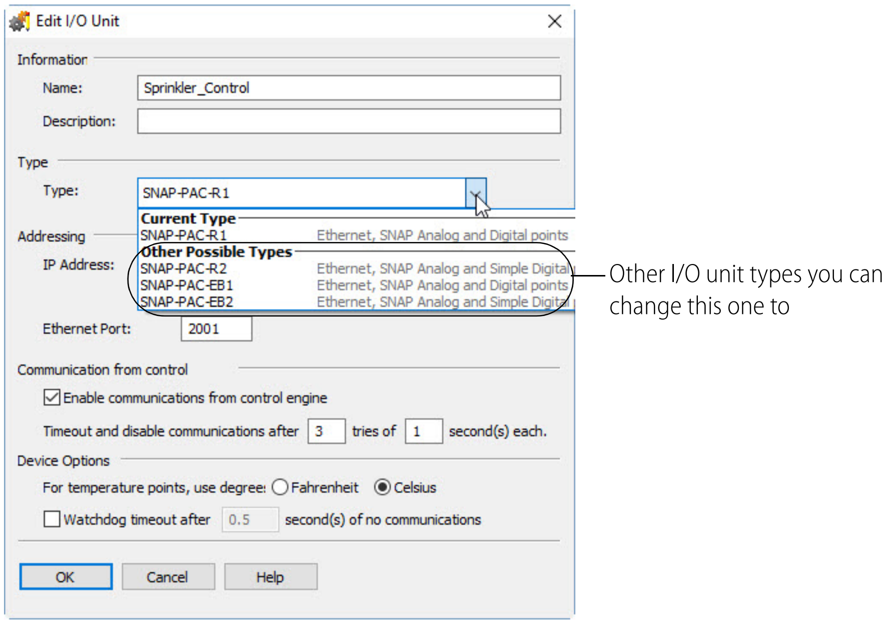
Old I/O unit type	New I/O unit type
SNAP-PAC-R1	SNAP-PAC-R2 SNAP-PAC-EB1 SNAP-PAC-EB2
SNAP-PAC-R1-B	SNAP-PAC-R1*
SNAP-PAC-R2	SNAP-PAC-R1 SNAP-PAC-EB1 SNAP-PAC-EB2
SNAP-PAC-EB1	SNAP-PAC-R1 SNAP-PAC-R2 SNAP-PAC-EB2
SNAP-PAC-EB2	SNAP-PAC-R1 SNAP-PAC-R2 SNAP-PAC-EB1
SNAP-PAC-SB1	SNAP-PAC-R1 SNAP-PAC-EB1
SNAP-PAC-SB2	SNAP-PAC-R1 SNAP-PAC-R2 SNAP-PAC-EB1 SNAP-PAC-EB2 SNAP-PAC-SB1
SNAP-ENET-D64	SNAP-PAC-R1* SNAP-PAC-R2* SNAP-PAC-EB1* SNAP-PAC-EB2*
SNAP-UP1-D64	SNAP-PAC-R1* SNAP-PAC-R2* SNAP-PAC-EB1* SNAP-PAC-EB2*
SNAP-B3000-ENET	SNAP-PAC-R1-B SNAP-PAC-R1* SNAP-PAC-EB1*
SNAP-ENET-RTC	SNAP-PAC-R1-B SNAP-PAC-R1* SNAP-PAC-EB1*
SNAP-UP1-ADS	SNAP-PAC-R1-B SNAP-PAC-R1* SNAP-PAC-EB1*
SNAP-ENET-S64	SNAP-PAC-R1 SNAP-PAC-R2 SNAP-PAC-EB1 SNAP-PAC-EB2
SNAP-UP1-M64	SNAP-PAC-R1 SNAP-PAC-R2 SNAP-PAC-EB1 SNAP-PAC-EB2
GENERIC OPTOMMP DEVICE	SNAP-PAC-R1 SNAP-PAC-R2 SNAP-PAC-EB1 SNAP-PAC-EB2

* This change requires a mounting rack change.

To change an I/O unit type:

1. With the strategy open in Configure mode, double-click the I/O Units folder in the Strategy Tree.
2. Select the I/O unit you want to change and click Modify.

3. Click the Type dropdown list to see the possible types you can change it to.



Edit I/O Unit

Information

Name: Sprinkler_Control

Description:

Type

Type: SNAP-PAC-R1

Current Type

SNAP-PAC-R1	Ethernet, SNAP Analog and Digital points

Other Possible Types

SNAP-PAC-R2	Ethernet, SNAP Analog and Simple Digital
SNAP-PAC-EB1	Ethernet, SNAP Analog and Digital points
SNAP-PAC-EB2	Ethernet, SNAP Analog and Simple Digital

Addressing

IP Address:

Ethernet Port: 2001

Communication from control

☑ Enable communications from control engine

Timeout and disable communications after 3 tries of 1 second(s) each.

Device Options

For temperature points, use degree: ○ Fahrenheit ● Celsius

☐ Watchdog timeout after 0.5 second(s) of no communications

4. Choose the new type, make any other necessary changes, and click OK.

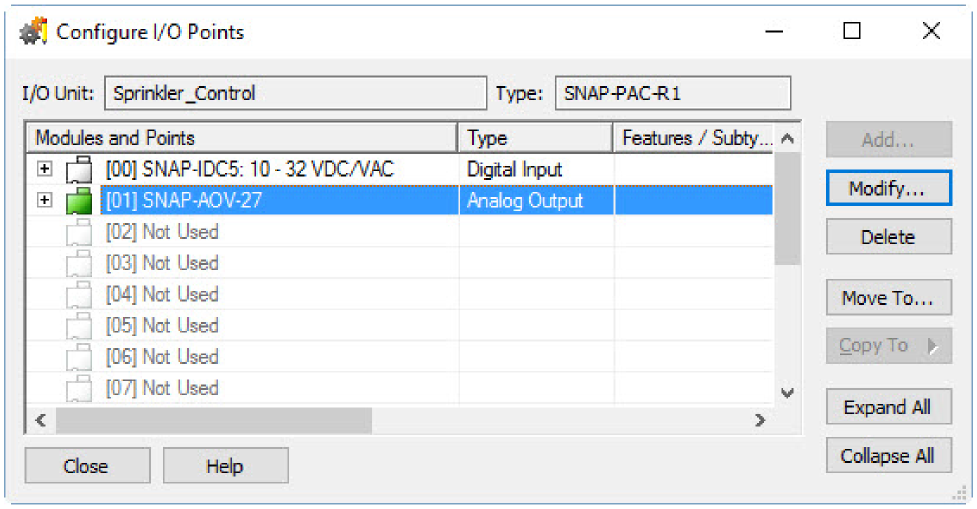
Moving a Configured I/O Module or Point

You can move a configured I/O module or I/O point to an empty position on the same I/O unit or a different I/O unit.

1. With the strategy open in Configure mode, double-click the I/O Units folder on the Strategy Tree.

2. In the Configure I/O Unit dialog box, highlight the unit the module or point is on and click I/O Points.

The Configure I/O Points dialog box opens. For a SNAP I/O unit, it looks something like this:

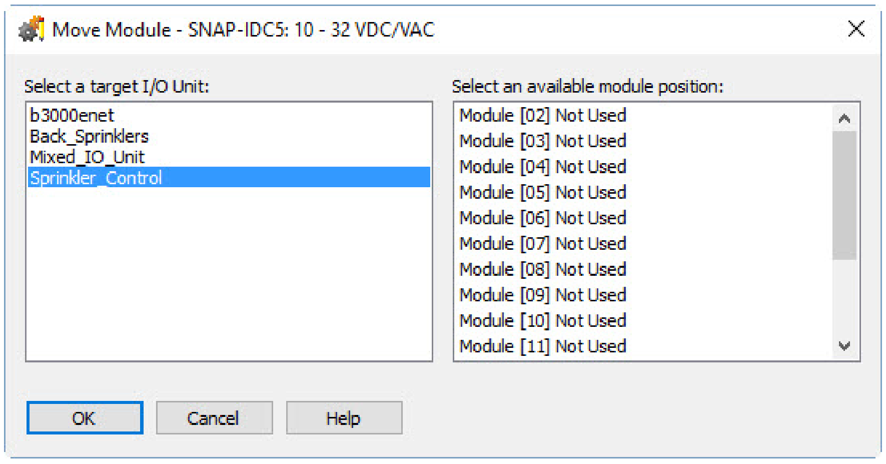


Configure I/O Points

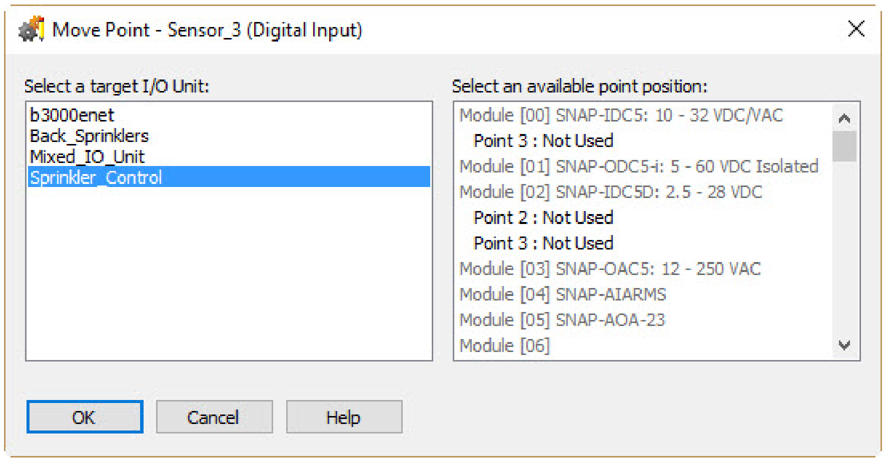
I/O Unit: Sprinkler_Control Type: SNAP-PAC-R1

Modules and Points	Type	Features / Subty...
[00] SNAP-IDC5: 10 - 32 VDC/VAC	Digital Input	
[01] SNAP-AOV-27	Analog Output	
[02] Not Used		
[03] Not Used		
[04] Not Used		
[05] Not Used		
[06] Not Used		
[07] Not Used		

– **To move a module**, select it and click Move To.



– **To move a point**, expand an individual module by clicking its plus sign ⊞ , or expand all modules by clicking Expand All. Select the point you want to move and click Move To.


Move Point - Sensor_3 (Digital Input)
Select a target I/O Unit:
b3000enet
Back_Sprinklers
Mixed_IO_Unit
Sprinkler_Control
Select an available point position:
Module [00] SNAP-IDC5: 10 - 32 VDC/VAC
Point 3 : Not Used
Module [01] SNAP-ODC5-i: 5 - 60 VDC Isolated
Module [02] SNAP-IDC5D: 2.5 - 28 VDC
Point 2 : Not Used
Point 3 : Not Used
Module [03] SNAP-OAC5: 12 - 250 VAC
Module [04] SNAP-AIARMS
Module [05] SNAP-AOA-23
Module [06]
OK
Cancel
Help

3. In the left column, select the I/O unit you want to move the module or point to. (The I/O unit it's on now is selected by default.) In the right column, select the position you are moving the module or point to. Then click OK.

You return to the Configure I/O Points dialog box, and the point has been moved.

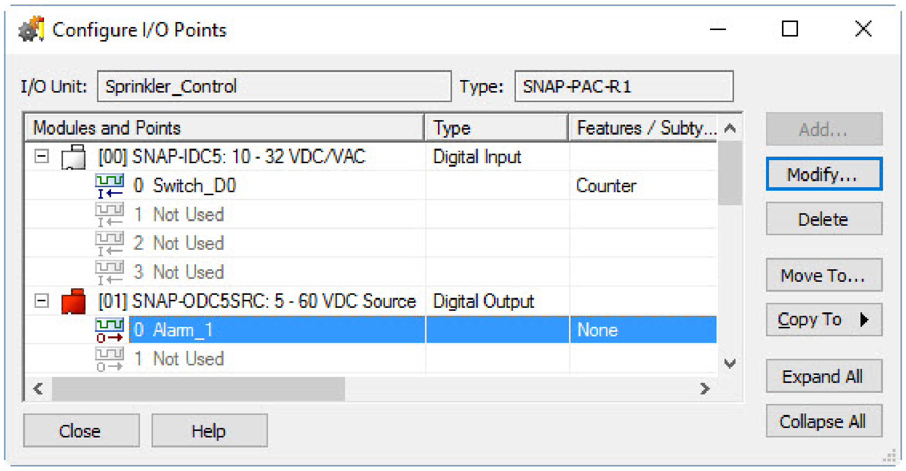
Copying a Configured I/O Point

If you have several points on a SNAP I/O module that are the same, you can copy a configured point:

- To fill empty points on the same module.
- To fill all empty points on the I/O unit.
- To another point on the same or a different I/O unit.

1. With the strategy open in Configure mode, double-click the I/O Units folder on the Strategy Tree to open the Configure I/O Points dialog box.

2. Click Expand All so you can see the points.

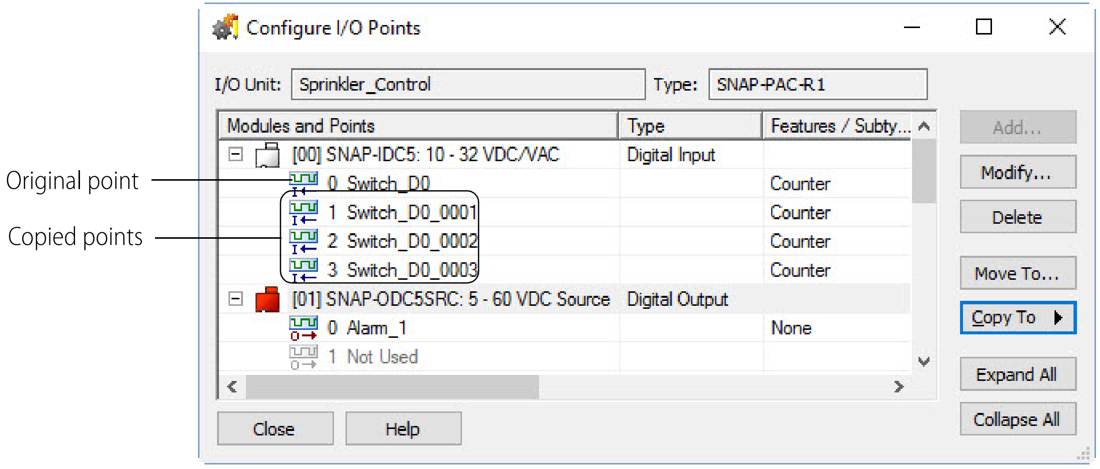


Modules and Points	Type	Features / Subty...
[00] SNAP-IDC5: 10 - 32 VDC/VAC	Digital Input	
0 Switch_D0		Counter
1 Not Used		
2 Not Used		
3 Not Used		
[01] SNAP-ODC5SRC: 5 - 60 VDC Source	Digital Output	
0 Alarm_1		None
1 Not Used		

3. Highlight the point you want to copy and click the Copy To button. From the pop-up menu, choose one of the following:

To copy the point to fill empty points on the module, choose Fill In Module.

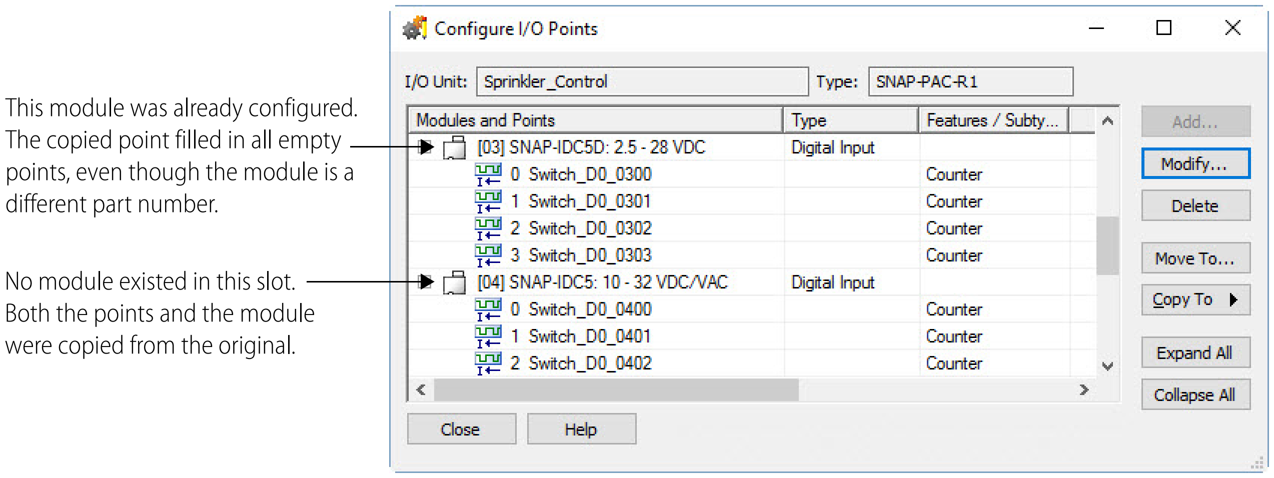
The point is copied to the other empty points on the same module.



Modules and Points	Type	Features / Subty...
[00] SNAP-IDC5: 10 - 32 VDC/VAC	Digital Input	
0 Switch_D0		Counter
1 Switch_D0_0001		Counter
2 Switch_D0_0002		Counter
3 Switch_D0_0003		Counter
[01] SNAP-ODC5SRC: 5 - 60 VDC Source	Digital Output	
0 Alarm_1		None
1 Not Used		

To copy the point to fill all empty points on the I/O unit, choose Fill In I/O Unit.

The point is copied to all empty points on similar modules (for example, to all empty points on digital input modules, whether they are the same or a different part number), and the module and point are copied to all compatible slots without configured modules.

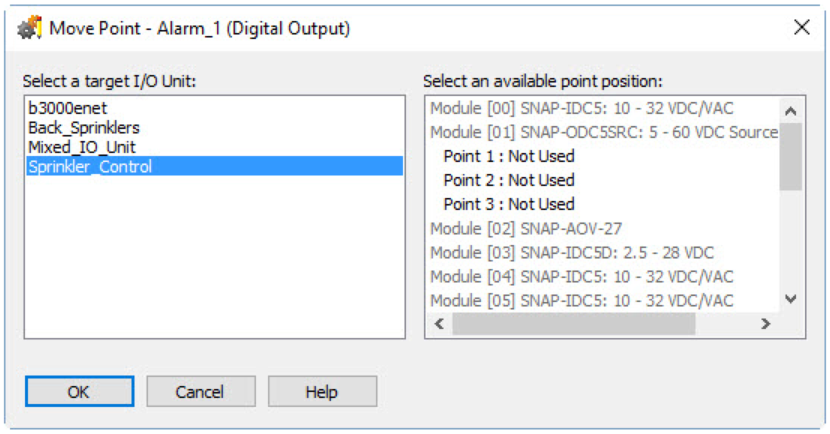

This module was already configured. The copied point filled in all empty points, even though the module is a different part number.
No module existed in this slot. Both the points and the module were copied from the original.
Configure I/O Points
Add...
Modify...
Delete
Move To...
Copy To
Expand All
Collapse All
Close
Help

I/O Unit: Sprinkler_Control Type: SNAP-PAC-R1

Modules and Points	Type	Features / Subty...
[03] SNAP-IDC5D: 2.5 - 28 VDC	Digital Input	
0 Switch_D0_0300		Counter
1 Switch_D0_0301		Counter
2 Switch_D0_0302		Counter
3 Switch_D0_0303		Counter
[04] SNAP-IDC5: 10 - 32 VDC/VAC	Digital Input	
0 Switch_D0_0400		Counter
1 Switch_D0_0401		Counter
2 Switch_D0_0402		Counter

To copy the point to one other point (either on the same or a different I/O unit), choose To Specific.

The Copy To dialog box opens.



Select a target I/O Unit:

b3000enet
Back_Sprinklers
Mixed_IO_Unit
Sprinkler_Control

Select an available point position:

Module [00] SNAP-IDC5: 10 - 32 VDC/VAC
Module [01] SNAP-ODC5SRC: 5 - 60 VDC Source
Point 1 : Not Used
Point 2 : Not Used
Point 3 : Not Used
Module [02] SNAP-AOV-27
Module [03] SNAP-IDC5D: 2.5 - 28 VDC
Module [04] SNAP-IDC5: 10 - 32 VDC/VAC
Module [05] SNAP-IDC5: 10 - 32 VDC/VAC

In the left column, choose the same or another I/O unit. In the right column, choose the location of the point to copy to.

Available points are shown in black. Already configured points, points on a different type of module (digital output instead of digital input, for example), and point numbers not on the rack (such as point 40 on an 8-module rack) are grayed out.

If the point is copied to an empty module slot, the module part number from the copied point is added, too.

4. After you copy a point, change the new point as needed by double-clicking its point name. In the Edit Digital Point dialog box, type the new name and make any other changes.

Changing a Configured I/O Point

1. With the strategy open in Configure mode, expand the I/O Units folder on the Strategy Tree until you can see the I/O point you want to change. Double-click the I/O point name.
2. In the Edit Analog Point or Edit Digital Point dialog box, make the necessary changes. Click OK to save.

Deleting a Configured I/O Point

You cannot delete an I/O point that is referred to in the strategy.

***CAUTION**: Be careful when deleting I/O points. You cannot undo a deletion.*

1. With the strategy open in Configure mode, expand the I/O Units folder on the Strategy Tree until you can see the I/O point you want to delete.
2. Right-click the I/O point's name and choose Delete from the pop-up menu.

You can also delete an I/O point from the Configure I/O Points dialog box by highlighting its name and clicking Delete.

Configuring PID Loops

PID loops (or simply PIDs) are used to drive an input (a process variable) toward a particular value (the setpoint) and keep the input very close to that value by controlling an output. For example, consider temperature control, where the input is a measurement of ambient temperature, the setpoint is the desired temperature, and the output is a heater. The PID for this system will use a mathematical formula that controls the output to maintain a desired temperature, efficiently adjust to changes in setpoint, and compensate for changes in load, such as the influx of cold air. In this example, a temperature sensor (analog input), a thermostat (analog input), and a heater control (analog output) are components of one system, controlled by a PID loop.

This guide assumes that you are already familiar with using PIDs. PID calculations are complex and the physical qualities of systems suitable for PID control differ greatly. This guide includes only basic information for configuring PIDs.

SNAP PAC I/O units support 96 PID loops. An analog/digital SNAP Ethernet I/O unit supports 16 PID loops; a SNAP Ultimate I/O unit supports 32 PID loops. If you have PAC Control Professional, you can also use up to eight PID loops on *mistic* I/O units.

PIDs can control isolated systems or be part of cascaded systems where one loop controls the setpoints or input variables of others. For maximum flexibility, any PID input, setpoint, or output can be determined by PAC Control commands.

PIDs and Strategies

The PID operates at the I/O unit, independently of the controller. However, the controller supports logic commands to read and write various PID parameters and modes. Once configured and initialized, a PID operates until the I/O Unit loses power.

When you change PID configuration in PAC Control, remember that changes are not written to the I/O unit until the strategy is downloaded and run. For a SNAP PAC or other SNAP Ethernet-based I/O unit, be sure to save PID configuration to flash memory following instructions for the I/O unit.

If you subsequently download a different strategy to the control engine, you'll receive an error message (-700) reminding you that a PID loop is still running and that it may conflict with the new strategy. To turn off a PID loop on a SNAP Ethernet-based I/O unit, open PAC Manager and use Inspect mode to change the PID's algorithm to None.

Each PID loop must be individually configured and tuned.

Configuration steps start in "Adding a PID Loop (SNAP PAC)" on page 151, and tuning steps are described in "Inspecting and Tuning PID Loops" on page 179. For additional information, see "PID—Ethernet Commands" on page 345, form 1641, *OptoTutorial: SNAP PAC PID*, and form 2171, *Tuning a PID Control Loop Technical Note*.

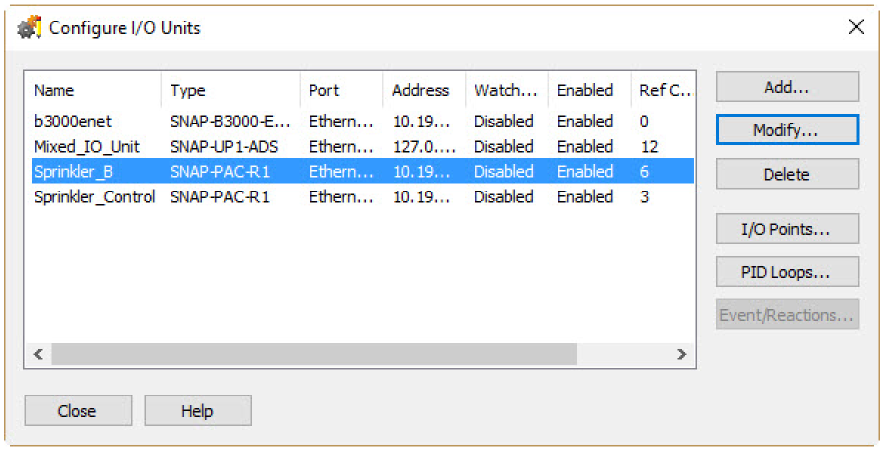
Pro (PAC Control) **For serial-based** *mistic* **I/O units,** skip to "Adding a PID Loop (mistic)" on page 155. For additional information, see "PID—mistic Commands" on page 349.

Adding a PID Loop (SNAP PAC)

NOTE: This section applies to SNAP PAC, Ethernet I/O, and Ultimate I/O units.

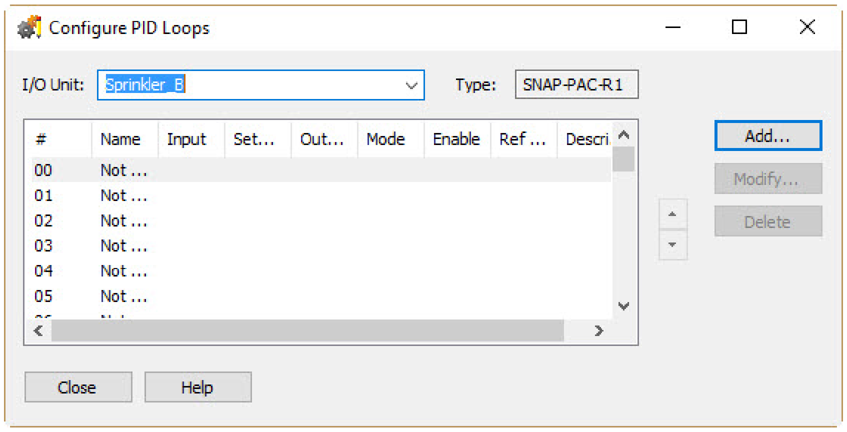
1. With the strategy open and in Configure mode, double-click the I/O Units folder (not the individual unit's icon) on the Strategy Tree.

   The Configure I/O Units dialog box opens.



Name	Type	Port	Address	Watch...	Enabled	Ref C..
b3000enet	SNAP-B3000-E...	Ethern...	10.19...	Disabled	Enabled	0
Mixed_IO_Unit	SNAP-UP1-ADS	Ethern...	127.0....	Disabled	Enabled	12
Sprinkler_B	SNAP-PAC-R1	Ethern...	10.19...	Disabled	Enabled	6
Sprinkler_Control	SNAP-PAC-R1	Ethern...	10.19...	Disabled	Enabled	3

2. Select the I/O unit the PID will be on, and click PID Loops.



#	Name	Input	Set...	Out...	Mode	Enable	Ref ...	Descri
00	Not ...							
01	Not ...							
02	Not ...							
03	Not ...							
04	Not ...							
05	Not ...							

3. Double-click the lowest unused number.

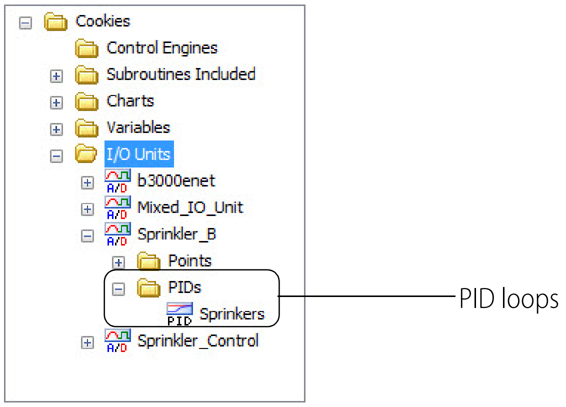
   The Add PID Loop Dialog appears.

4. Complete the fields as described in "Add PID Loop Dialog Box" below.
5. Click OK.

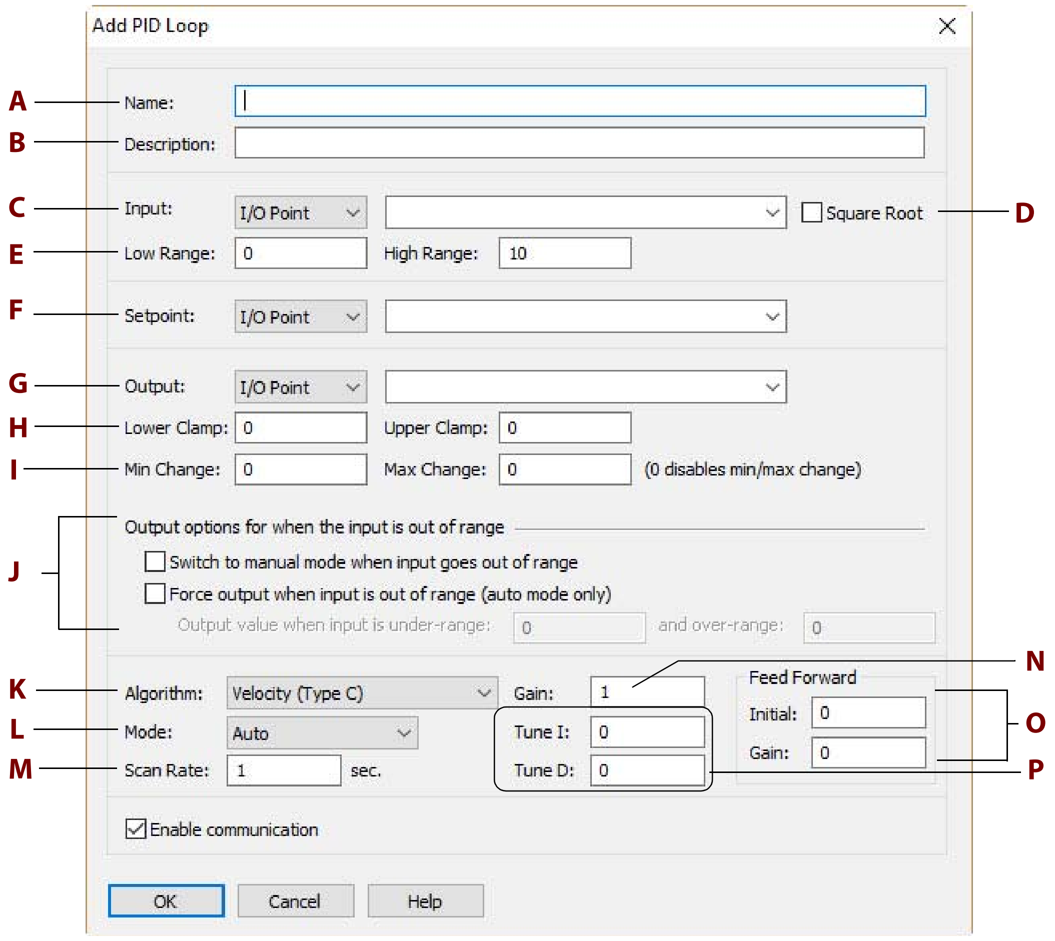
   The new PID appears in the Configure PID Loops dialog box.

6. When you have finished configuring PIDs, click Close.

   PIDs appear in the Strategy Tree under the I/O unit.



Add PID Loop Dialog Box



A—*Name.* Type a unique, descriptive name for the PID. The name must start with a letter and may contain letters, numbers, and underscores (spaces are converted to underscores).

B—*Description.* (Optional) Enter a description of the PID.

C—*Input.* Select the type of input: I/O Point, Host, or PID Output.

- If the PID's process variable comes from an I/O point on the same unit, select I/O Point. Choose the point from the drop-down list or type a point name to configure a new point.
- If the PID's process variable comes from the PAC Control strategy, select Host. Enter an initial value for the input.
- If the PID's process variable is the output of another PID on this brain (a cascading control loop), select PID Output. Choose the PID from the drop-down list.

D—*Square Root.* (Optional) If you chose I/O Point or PID for step C, check this box if the error should be calculated based on the square root of the process variable (applies to flow control systems where volumetric flow is proportional to the square root of a signal from a flow transducer).

E—*Low/High Range.* Set the valid range of the process variable by entering the low range and the high range. (See Output Options for optional responses to out-of-range input.)

F—*Setpoint.* Choose the source for the setpoint: I/O Point, Host, or PID Output.

- To control the setpoint using a device (on the same brain) such as a potentiometer, select I/O Point; choose an I/O point from the drop-down list or type a new point name.
- To control setpoint using PAC Control or PAC Display, select Host. For Initial Value, *enter the most common operating value*.
- If another PID loop will control the setpoint, select PID Output and choose the PID from the drop-down list.

G—*Output.* Choose the destination for the PID output: I/O Point or Host. (To use the output for controlling the setpoint or input of another PID, choose Host.)

H—*Lower Clamp/Upper Clamp.* Enter upper and lower clamp values to prevent the output from exceeding a desirable range. These values should equal the range of the output point, if used. Or choose values to make sure that the output device doesn't shut off (for example, keeping a circulation pump running regardless of the PID output) or that the output never reaches a destructively high setting (for example, keeping a motor below maximum).

I—*Min Change/Max Change.* (Optional) Enter minimum and maximum change values. The output won't respond until the minimum change is reached (for example, you may not want a heater to turn on to correct a 1 degree error). Maximum change prevents too drastic a change in output (for example, you could limit the increase in a pump's output to prevent pipe breakage). The default for both minimum and maximum is zero, which disables the feature.

J—*Output Options.* Choose how the PID should respond if the input goes out of range. To have PAC Control logic or an operator respond, check Switch to manual mode. To force the output to a specific value, check that option and type the output values.

NOTE: If both boxes are checked (forced output and manual mode), the output will be forced and the PID put into manual mode; but if the PID is already in manual mode, the output will not be forced. (You can use the command Get PID Status Flags to determine current settings.)

If neither box is checked and the PID input goes out of range (as defined by E– Low/High Range) then the output will freeze, but only while the input is out of range.

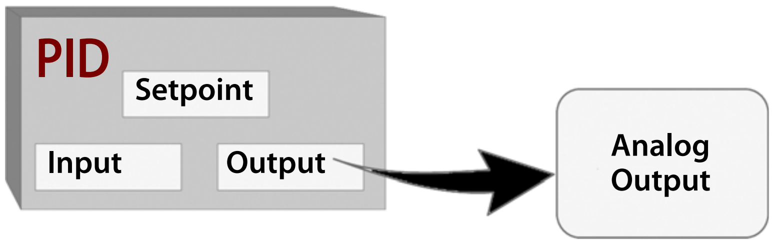
K—*Algorithm.* Choose algorithm: Velocity, ISA, Parallel, or Interacting. For details on algorithms, see "Algorithm Choices —Ethernet" on page 347.

L—*Mode.* Choose the Mode you want the PID to be initialized to. The main difference between Automatic and Manual mode is that in Manual mode the PID loop stops its mathematical calculation. In both modes the PID's output value is still copied to the analog output.

- Auto mode: The PID makes calculations based on the difference between the input and the setpoint resulting in changes to the output, which causes the input to move toward the setpoint.
- Manual mode: The PID stops making changes to the output, but continues to write the PID output to the analog point (or configured destination for the PID output). Manual mode allows

PAC Control logic or an operator to control the PID output, which in turn is written by the PID loop to the analog point (or configured destination of the PID output).

NOTE: If the value of the analog point or configured destination is changed, it will be overwritten by the PID output.



M—*Scan Rate.* Enter a scan rate to determine how often the input is scanned and the controller output is calculated. Minimum value is 0.001 (1 ms). Scan time should be greater than system lag (the time it takes for the controller output to have a measurable effect on the system). Also consider other PIDs and tasks on the brain competing for processing power.

N—*Gain.* Type a positive or negative value for Gain. Heating systems usually require a negative value and cooling systems a positive value. NOTE: Gain is usually refined during the tuning process.

O—*Fd Fwd Initial/Fd Fwd Gain.* (Optional) Enter Feed forward Initial and Feed forward gain values if you need to offset the controller output in your application. These values are constants that are multiplied and added to the controller output; often they are not used in PIDs.

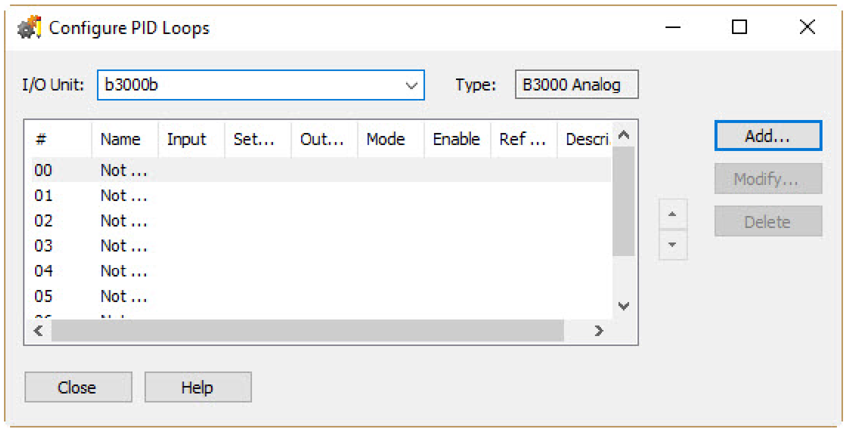
P—*Tune I/Tune D.* (Optional) Type Integral and Derivative settings if you know the desirable settings. However, Integral and Derivative are not essential to basic configuration and are better determined in the tuning process.

Adding a PID Loop (*mistic*)

mistic PID loops can be configured only on B3000 or B3000-B brains and G4A8R bricks. (For PID loops on Ethernet-based I/O units, see page 151.)

1. To configure a *mistic* PID loop, make sure you have the strategy open in Configure mode.
2. On the Strategy Tree, double-click the I/O Units folder (not the individual unit's icon).
3. In the Configure I/O Units dialog box, click the PID Loops button.

The Configure PID Loops dialog box appears, listing all PID loops on the I/O unit.



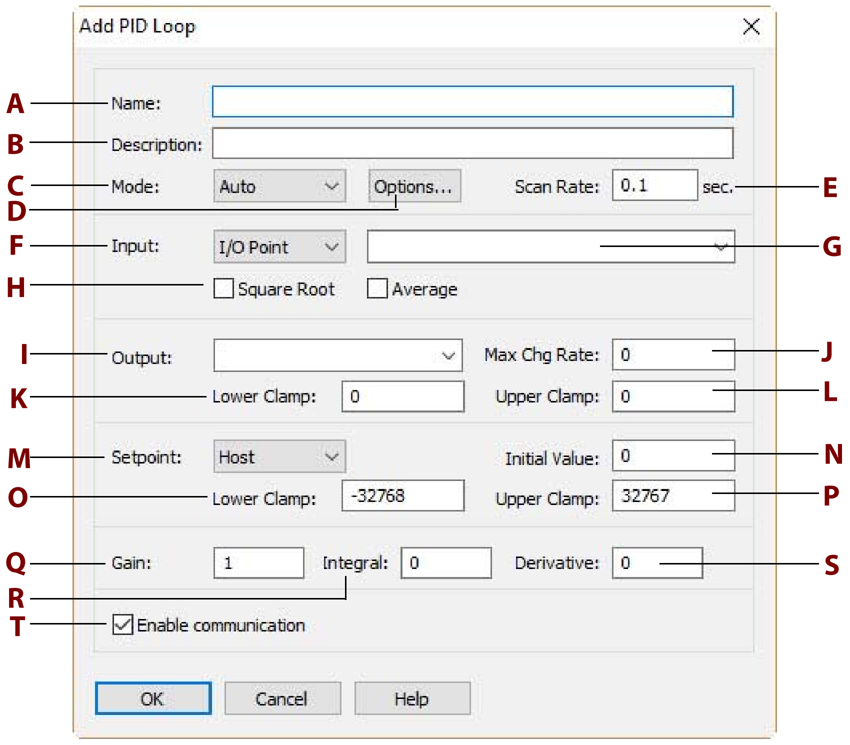
I/O Unit: b3000b Type: B3000 Analog

#	Name	Input	Set...	Out...	Mode	Enable	Ref ...	Descri
00	Not ...							
01	Not ...							
02	Not ...							
03	Not ...							
04	Not ...							
05	Not ...							

4. If the correct I/O unit does not appear in the I/O Unit field, choose it from the drop-down menu.
5. Double-click an unused line (or highlight it and click the Add button).

   The Add PID Loop dialog box opens.
6. If you choose I/O Point as the Input source and Host as the Setpoint source, complete the fields as described in "Add PID Loop Dialog Box (mistic), I/O Point Input" below. However, if you choose Host as the Input source and I/O Point as the Setpoint source, complete the fields as described in "Add PID Loop Dialog Box (mistic), Host Input" on page 158.

Add PID Loop Dialog Box (*mistic*), I/O Point Input



Name:

Description:

Mode: Auto Scan Rate: 0.1 sec.

Input: I/O Point

Square Root Average

Output: Max Chg Rate: 0

Lower Clamp: 0 Upper Clamp: 0

Setpoint: Host Initial Value: 0

Lower Clamp: -32768 Upper Clamp: 32767

Gain: 1 Integral: 0 Derivative: 0

☑ Enable communication

A—*Name.* Enter a name for the PID. The name must start with a letter and may contain letters, numbers, and underscores. (Spaces are converted to underscores.)

B—*Description.* (Optional) Enter a description of the PID.

C—*Mode.* Select whether the initial mode of the PID is automatic or manual. In Automatic mode, the PID is automatically calculated. In Manual mode, no calculation is made.

D—*Options.* To set advanced options, click Options and see "Setting PID Loop Control Options (mistic PIDs)" on page 159.

E—*Scan Rate.* Enter a time (in seconds) representing the interval between PID loop calculations. The smaller the number, the more often the PID will be calculated. The default of 0.1 specifies a PID calculation 10 times per second.

F—*Input.* Select whether the input for the PID calculation will be read from an I/O point or a host device (the controller). Your choice determines the other data required in the Input section of the dialog box. If you choose Host, see the figure in "Add PID Loop Dialog Box (mistic), Host Input" on page 158 for additional fields to complete.

G—*Input drop-down list.* (If I/O Point is selected at **F**) Select from the drop-down list the I/O point providing the input value. If the point doesn't exist, enter a new name and add the point.

H—*Square Root.* (If I/O Point is selected at **F**) Enable or disable square root extraction and averaging of the input value prior to the PID calculation. These options are disabled by default.

I—*Output.* Specify the output to be driven by the PID by selecting an I/O point from the drop-down list. If it doesn't exist, enter a new I/O point name and add the point.

J—*Maximum Change Rate.* Specify the maximum absolute difference allowed for the output value as the result of one PID calculation. For example, a maximum change rate of 10 specifies that even if a PID calculation suggests an output change of 20 units, the maximum change allowed during the loop will be 10 units.

You can use a maximum change rate to avoid sudden, dramatic increases or decreases in the output value. Note that the maximum change rate must be between one percent and 100 percent of the range of the output itself, which is the difference between the output's high-scale value (**L**) and low-scale value (**K**). The value representing this full range will appear by default.

K—*Lower Clamp, Output.* Specify the minimum value allowable for the output. This value cannot be less than the zero-scale value of the output module.

L—*Upper Clamp, Output.* Specify the maximum value allowable for the output. This value cannot be greater than the full-scale value of the output module.

M—*Setpoint.* Select whether the setpoint for the PID calculation will be read from an I/O point or a host device (the controller). The setpoint is the value to which the input will be driven. Your choice determines the other data required in the Setpoint section of the dialog box. If you choose I/O Point, see the figure in "Add PID Loop Dialog Box (mistic), Host Input" on page 158 for additional fields to complete.

N—*Initial Value.* (If Host is selected at **M**) Specify the initial value for the setpoint. This value must be between the lower-clamp and upper-clamp values (**O** and **P**). The default is zero.

O—*Lower Clamp, Setpoint.* (If Host is selected at **M**) Specify the lowest possible setpoint value. Default: -32,768.

P—*Upper Clamp, Setpoint.* (If Host is selected at **M**) Specify the highest possible setpoint value. Default: 32,767.

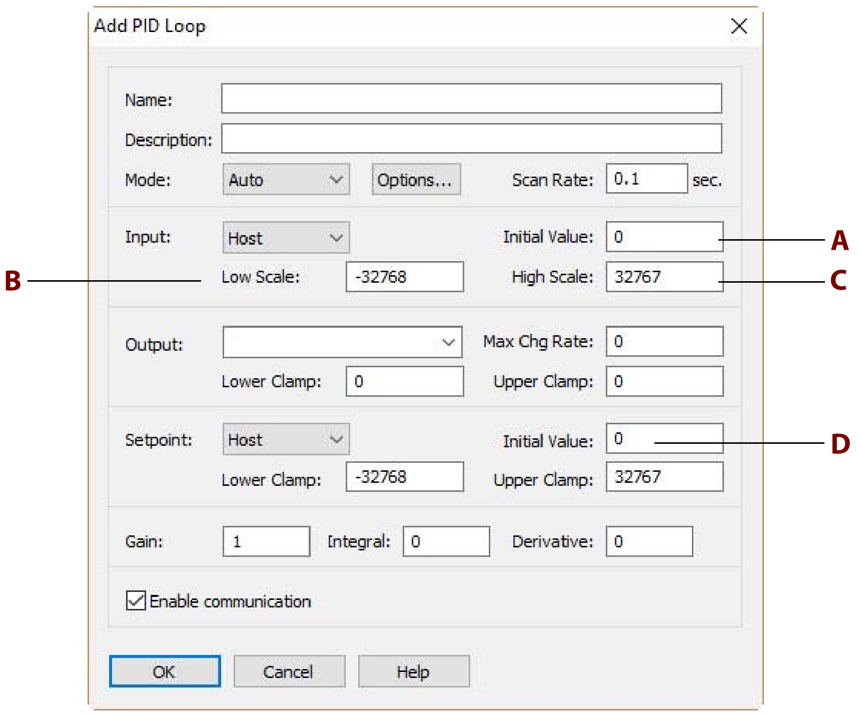
Q—*Gain.* Specify the gain term (P) to be used in the PID calculation. This value can range between -32,768 and 32,767 but must not be zero. The default is one.

R—*Integral.* Specify the integral term (I) to be used in the PID calculation. This value can range between zero and 32,767. The default is zero. (Note: The product of the scan rate and the integral term must be less than or equal to 3,932,100.)

S—*Derivative.* Specify the derivative term (D) to be used in the PID calculation. This value can range between zero and 32,767. The default is zero.

T—*Enable Communication.* Select whether you want communication to this PID loop enabled or disabled.

. Add PID Loop Dialog Box (*mistic*), Host Input



A—*Initial Value.* (If Host is selected as Input source) Specify the initial value for the input for the PID calculation. This value must be between the low-scale and high-scale values (**B** and **C**). By default, this value is set to zero.

B—*Low Scale.* (If Host is selected as Input source) Specify the lowest possible input value. The default is -32,768.

C—*High Scale.* (If Host is selected as Input source) Specify the highest possible input value. The default is 32,767.

D—*Setpoint.* (If I/O Point is selected as Setpoint source) Select from the drop-down list the I/O point providing the setpoint value. If the point doesn't exist, enter a new name and add it.

Setting PID Loop Control Options (*mistic* PIDs)

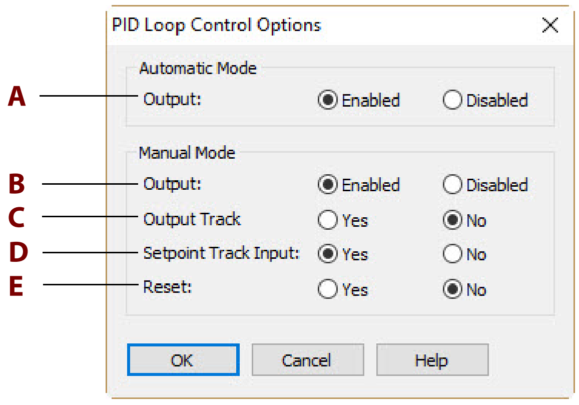
(Pro only) If you clicked the Options button in the Add PID Loop dialog box in step 6 of "Adding a PID Loop (mistic)" on page 155, the PID Loop Control dialog box opens.

1. Set the options as described in "PID Loop Control Options Dialog Box" below.

   NOTE: Since you can switch between automatic and manual modes through the View PID Loop dialog box in Debug mode, you may want to configure options for both modes.

2. When you have set the options, click OK to return to the Add PID Loop dialog box.

PID Loop Control Options Dialog Box



Automatic Mode
Output: [✓ Enabled] Disabled

Manual Mode
Output: [✓ Enabled] Disabled
Output Track: Yes [✓ No]
Setpoint Track Input: [✓ Yes] No
Reset: Yes [✓ No]

A—*Automatic Mode.* If you want the PID calculation transferred to the output, click Enabled (the default). If you do not want the PID calculation transferred to the output, click Disabled. In either case, the PID will continue calculating and will not be reset.

B—*Manual Mode/Output.* If you want the PID calculation transferred to the output, click Enabled (the default). If you do not want the PID calculation transferred to the output, click Disabled.

C—*Manual Mode/Output Track Input.* If you want the output to assume the input value, click Yes. You can use this option to create a signal converter (for example, 4–20 mA input to 0–10 V output), since the output is proportional to the input.

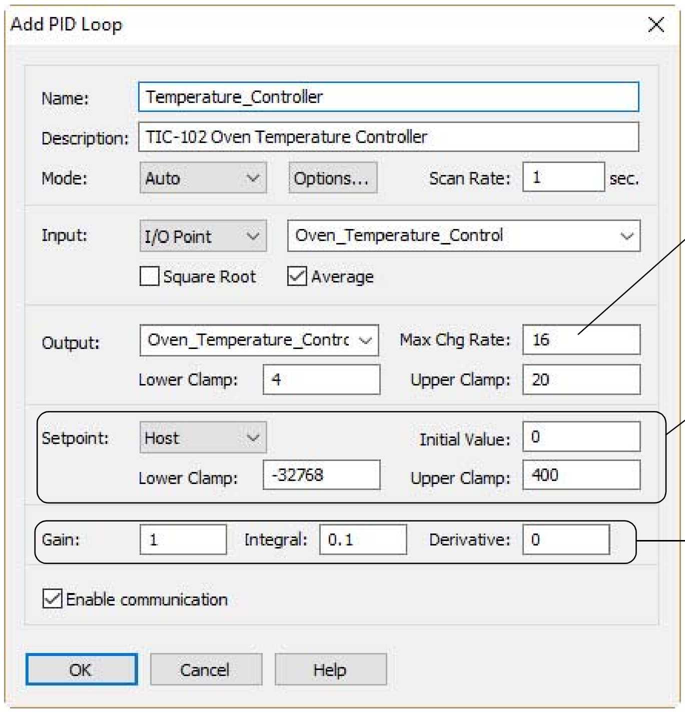
D—*Manual Mode/Setpoint Track Input.* If you want the setpoint to be set to the input value, click Yes. You can use this option to smooth the transfer when returning to automatic mode.

E—*Manual Mode/Reset.* If you want to force the PID loop to reset when entering manual mode, click Yes. This option stops all calculations, sets all process errors to zero, and resets the scan rate timer, leaving the output unchanged. If you want the PID calculation to continue, click No. (When you enter automatic mode, however, you cannot force a reset.)

mistic PID Loop Configuration Example

Here's an example of a completed *mistic* PID configuration for a temperature controller. The PID modifies an output automatically every second.

The input is an I/O point called Oven_Temperature. Its value is averaged before being applied to the PID calculation. The output being driven is an I/O point called Oven_Temperature_Control, which must remain between zero and 100 units.



Add PID Loop

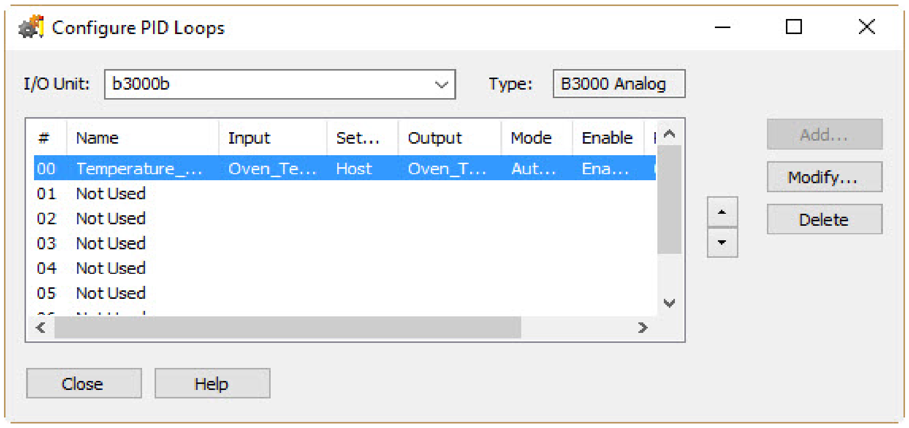
Name: Temperature_Controller
Description: TIC-102 Oven Temperature Controller
Mode: Auto | Options... | Scan Rate: 1 sec.
Input: I/O Point | Oven_Temperature_Control
☐ Square Root ☑ Average
Output: Oven_Temperature_Contrc | Max Chg Rate: 16
Lower Clamp: 4 | Upper Clamp: 20
Setpoint: Host | Initial Value: 0
Lower Clamp: -32768 | Upper Clamp: 400
Gain: 1 | Integral: 0.1 | Derivative: 0
☑ Enable communication

The maximum change rate has been set to the full range (100), which means we are allowing the output to change as much as it needs to during each cycle.

The setpoint is read from the host device (the controller) and is initially set to zero. It is clamped at 400 at the upper end.

For the PID calculation, the gain term is one, the integral term is 0.1, and the derivative term is zero.

The sample PID would appear in the Configure PID Loops dialog box like this:



Configure PID Loops

I/O Unit: b3000b | Type: B3000 Analog

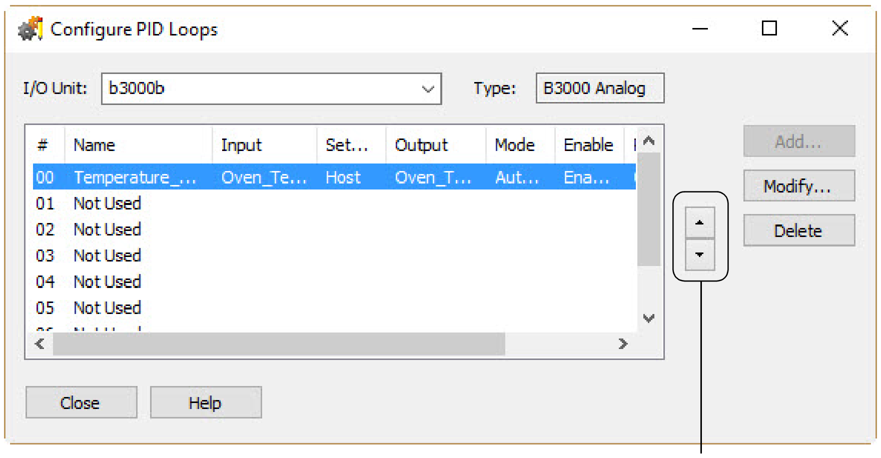
#	Name	Input	Set...	Output	Mode	Enable
00	Temperature_...	Oven_Te...	Host	Oven_T...	Aut...	Ena...
01	Not Used					
02	Not Used					
03	Not Used					
04	Not Used					
05	Not Used					

Changing a PID Loop (Ethernet or *mistic*)

You can change the PID loop's configuration and its position in the I/O unit.

1. Make sure the strategy is open and in Configure mode. On the Strategy Tree, expand the I/O Units folder until you see the PIDs folder for the I/O unit you want. Double-click the PIDs folder.

   The Configure PID Loops dialog box opens, listing all configured PID loops. Remember that the number of PID loops available depends on the I/O unit.



Configure PID Loops

I/O Unit: b3000b Type: B3000 Analog

#	Name	Input	Set...	Output	Mode	Enable
00	Temperature_...	Oven_Te...	Host	Oven_T...	Aut...	Ena...
01	Not Used					
02	Not Used					
03	Not Used					
04	Not Used					
05	Not Used					

Up- and down-arrows

   PID loops are scanned by the I/O unit in the order that they appear in this list.

2. To move the PID loop to a different position on the I/O unit, use the up- and down-arrows in the dialog box.
3. To change the PID loop's configuration, double-click its name to open the Edit PID Loop dialog box. Change the fields as necessary.

   For help in completing the fields, see "Adding a PID Loop (SNAP PAC)" on page 151 or "Adding a PID Loop (mistic)" on page 155.

Deleting a PID Loop (Ethernet or *mistic*)

Only PID loops that have a reference count of zero can be deleted. Be careful when deleting PID loops; you cannot undo a deletion.

1. Make sure the strategy is open and in Configure mode. On the Strategy Tree, expand the I/O units folder until you see the PID loop you want to delete.
2. Right-click the name of the PID loop and choose Delete from the pop-up menu.

   The PID loop is deleted.

   You can also delete a PID loop in the Configure PID Loops dialog box by highlighting it and clicking Delete.

Configuring Event/Reactions

(Pro only) Event/reactions apply to **serial-based** *mistic* **I/O units only**.

NOTE: Similar events and reactions can be configured on a SNAP Ethernet-based I/O unit using PAC Manager, but they can interfere with PAC Control strategy logic unless you are very careful. For more information, see form Opto 22 1704, the PAC Manager User's Guide.

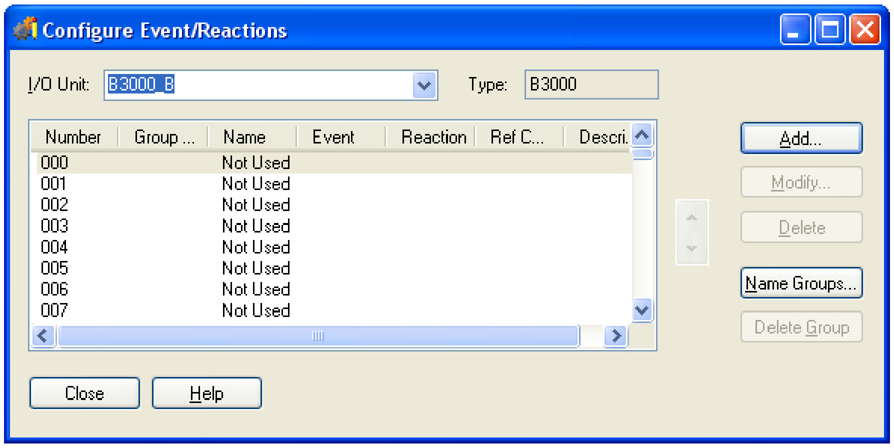
Event/reactions do exactly what their name suggests: they make something happen in response to an event. Their advantage is that they simplify and speed the control engine's job by offloading certain control functions from the control engine to the I/O unit.

Examples of events are a timeout, an input or output reaching a value, or a special digital feature (such as frequency or an on-time totalizer) reaching a value. Examples of reactions include starting an on- or off-pulse, reading and holding a value, or activating or deactivating a *mistic* PID loop.

Event/reactions can be configured on *mistic* digital and analog multifunction I/O units only. You can configure up to 256 event/reactions on a single I/O unit. For more information on using event/reactions, see "Event/Reaction Commands" on page 332.

1. To configure an event/reaction, make sure the strategy is open in Configure mode. On the Strategy Tree, double-click the I/O Units folder (not the individual unit's icon).
2. In the Configure I/O Units dialog box, highlight the multifunction *mistic* I/O unit and click the Event/Reactions button.

   The Configure Event/Reaction dialog box appears, showing all event/reactions contained on the I/O unit (in this example, none).



Number	Group ...	Name	Event	Reaction	Ref C...	Descri.
000		Not Used				
001		Not Used				
002		Not Used				
003		Not Used				
004		Not Used				
005		Not Used				
006		Not Used				
007		Not Used				

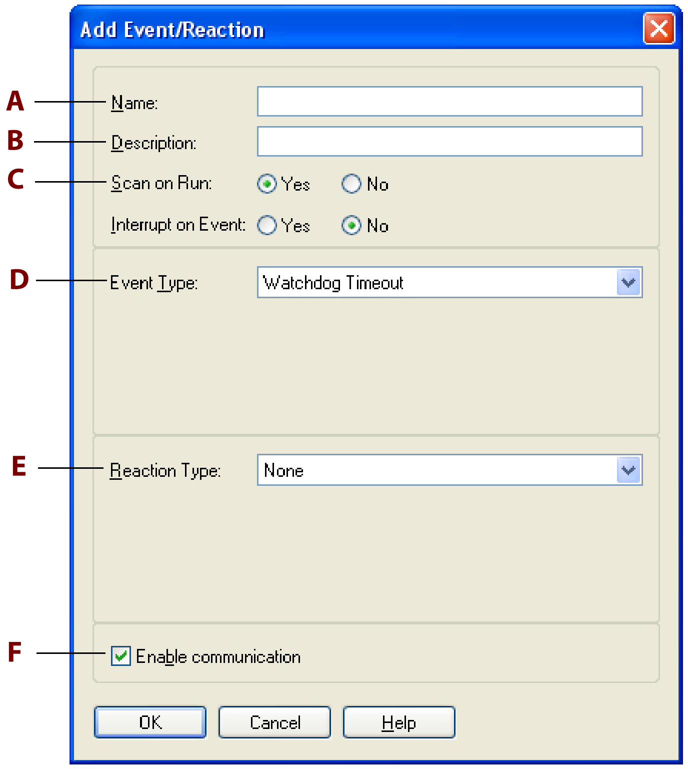
3. If the correct I/O unit is not shown, select the unit from the drop-down list.

   Since only multifunction units support event/reactions, they are the only units listed.
4. Highlight an unused line and click Add, or double-click an unused line.

   The Add Event/Reaction dialog box appears.
5. Complete the fields as described in "Add Event/Reaction Dialog Box" below.
6. Click OK.

The event/reaction appears in the Configure Event/Reaction dialog box.

Add Event/Reaction Dialog Box



Name:

Description:

Scan on Run: [✓ Yes] No

Interrupt on Event: Yes [✓ No]

Event Type: Watchdog Timeout

Reaction Type: None

[✓] Enable communication

This figure shows the fields in the dialog box when you first open it.

Depending on the type of event or reaction you select, other fields may appear.

A—*Name.* Enter a name for the event/reaction. The name must start with a letter and may contain letters, numbers, and underscores (spaces are converted to underscores).

B—*Description.* (Optional) Enter a description of the event/reaction.

C—*Scan on Run.* If you want the I/O unit to begin scanning for the event automatically as soon as the strategy is run, click Yes. If you want scanning to wait until it is started by a command in the strategy, click No.

D—*Event Type.* From the drop-down list, select an event to scan for. Complete additional fields that appear as described in the following tables.

For digital I/O units:

For this type of event	Enter this information
Watchdog Timeout	No information required.

For this type of event	Enter this information
Counter >= Value Quadrature >= Value Totalize On >= Value Totalize Off >= Value On-Pulse >= Value Off-Pulse >= Value Period >= Value Frequency >= Value Quadrature <= Value Frequency <= Value Counter <= Value	Specify the I/O point to be monitored and the value to compare the I/O point against. Select the I/O point from the drop-down list or specify a new name for the point and configure it.
MOMO Match	See "Adding a MOMO Event or Reaction (mistic I/O Units Only)" on page 166.

For analog I/O units:

For this type of event	Enter this information
Watchdog Timeout	No information required.
Analog Input >= Value Analog Input <= Value Analog Output >= Value Analog Output <= Value	Specify the I/O point to be monitored, the value to compare the I/O point against, and the type of comparison value (current, average, maximum, minimum, or total). Select the I/O point from the drop-down list or specify a new name for the point and configure it.

E—*Reaction Type.* From the drop-down list, select the reaction to take in response to the event. Complete additional fields that appear as described in the following tables.

For digital I/O units:

For this type of reaction	Enter this information
None (no reaction)	None
Enable Scan for Event Disable Scan for Event	Select the event from the drop-down list.
Enable Scan for E/R Group Disable Scan for E/R Group	Select the group from the drop-down list.
Disable Scan for All Events	None
Set MOMO Outputs	See "Adding a MOMO Event or Reaction (mistic I/O Units Only)" on page 166.
Start On-Pulse Start Off-Pulse	Specify the I/O point to be pulsed and the length of the pulse in seconds. Select the I/O point from a drop-down list or specify a new name for the point and configure it.

For this type of reaction	Enter this information
Start Counter Stop Counter Start Quadrature Counter Stop Quadrature Counter Clear Counter Clear Quadrature Counter Clear On-Pulse Clear Off-Pulse Clear Period Clear Totalize On Clear Totalize Off Read and Hold Counter Value Read and Hold Quadrature Value Read and Hold Totalize On Value Read and Hold Totalize Off Value Read and Hold On-Pulse Value Read and Hold Off-Pulse Value Read and Hold Period Value Read and Hold Frequency Value	Specify the I/O point to be affected. Select the I/O point from a drop-down list or specify a new name for the point and configure it.

For analog I/O units:

For this type of reaction	Enter this information
None (no reaction)	None
Enable Scan for Event Disable Scan for Event	Select the event from the drop-down list.
Enable Scan for E/R Group Disable Scan for E/R Group	Select the group from the drop-down list.
Disable Scan for All Events	None
Read and Hold Analog Input Data Read and Hold Analog Output Data	Specify the I/O point and type of data (current, average, maximum, minimum or total) to be read. Select the I/O point from a drop-down list or specify a new name for the point and configure it.
Activate PID Loop Deactivate PID Loop	Specify the PID loop to be affected. Select the PID from a drop-down list or specify a new name for the PID and configure it.
Set PID Loop Setpoint	Specify the PID loop to be affected and the setpoint value. Select the PID from a drop-down list or specify a new name for the PID and configure it.
Set Analog Output	Specify the I/O point and the value to set it to. Select the I/O point from a drop-down list or specify a new name for the point and configure it.
Ramp Analog Output to Setpoint	Specify the I/O point, the ramping speed in units per second, and the end point value to ramp to. Select the I/O point from a drop-down list or specify a new name for the point and configure it.

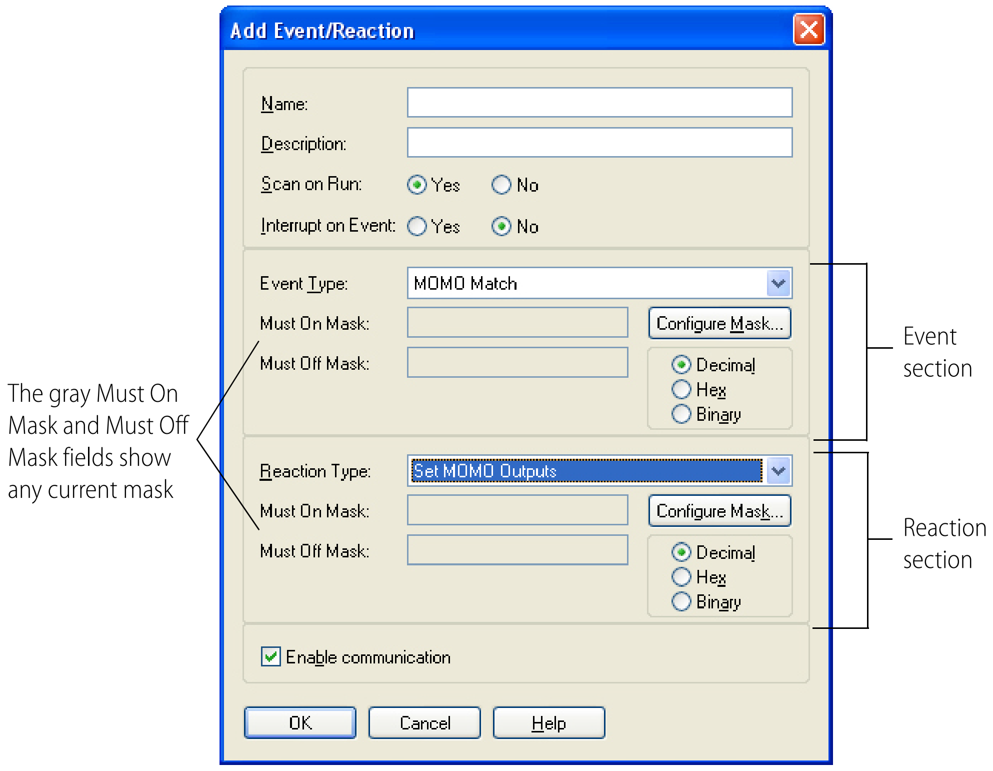
F—*Enable Communication.* To enable communication to this event/reaction, check to enable communication to this event/reaction, or uncheck to disable communication.

Adding a MOMO Event or Reaction (*mistic* I/O Units Only)

(Pro only) On a digital multifunction I/O unit, a special type of event or reaction you can configure is called MOMO (must-on, must-off). A MOMO Match event monitors several inputs and/or outputs on an I/O unit for a match to a specific pattern. A Set MOMO Outputs reaction defines a set of values for outputs on an I/O unit. You can use just a MOMO event, just a MOMO reaction, or both.

If you select the MOMO Match event or the Set MOMO Outputs reaction in the Add Event/Reaction dialog box, additional fields and buttons appear, which are described in the following steps.

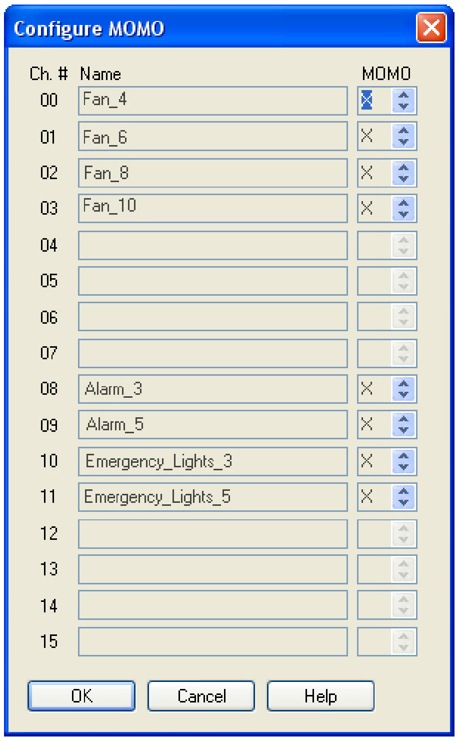
The following figure shows the additional fields and buttons that appear if you select the MOMO Match event or the Set MOMO Outputs reaction:



Name:
Description:
Scan on Run: [✓ Yes] No
Interrupt on Event: Yes [✓ No]
Event Type: MOMO Match
Must On Mask:
Must Off Mask:
[✓ Decimal] Hex Binary
Reaction Type: Set MOMO Outputs
Must On Mask:
Must Off Mask:
[✓ Decimal] Hex Binary
[✓] Enable communication

1. To set up or change the must-on pattern or must-off pattern (called the mask), click Configure Mask in the Event or Reaction sections.

The Configure MOMO dialog box appears, listing all I/O points you can set in the mask.



Configure MOMO

Ch. #	Name	MOMO
00	Fan_4	X
01	Fan_6	X
02	Fan_8	X
03	Fan_10	X
04		
05		
06		
07		
08	Alarm_3	X
09	Alarm_5	X
10	Emergency_Lights_3	X
11	Emergency_Lights_5	X
12		
13		
14		
15		

2. For each I/O point, click the scroll arrows to indicate that the point is On, Off, or X (ignored). When you have finished the mask, click OK.

   You return to the Add Event/Reaction dialog box, and the numerical equivalent of the mask you have set appears in the Must On Mask and Must Off Mask fields.

3. In the Event and Reaction sections, choose whether to display the mask in decimal, hexadecimal, or binary form. The default is decimal.

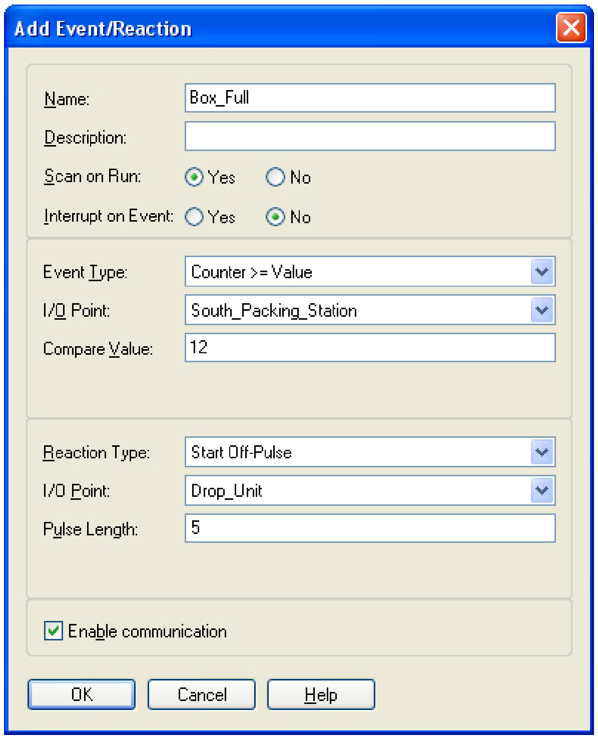
4. When the event/reaction is configured, click OK.

   The new event/reaction appears in the Configure Event/Reaction dialog box.

Event/Reaction Configuration Example

NOTE: Event/reactions are available on serial mistic *I/O units only.*

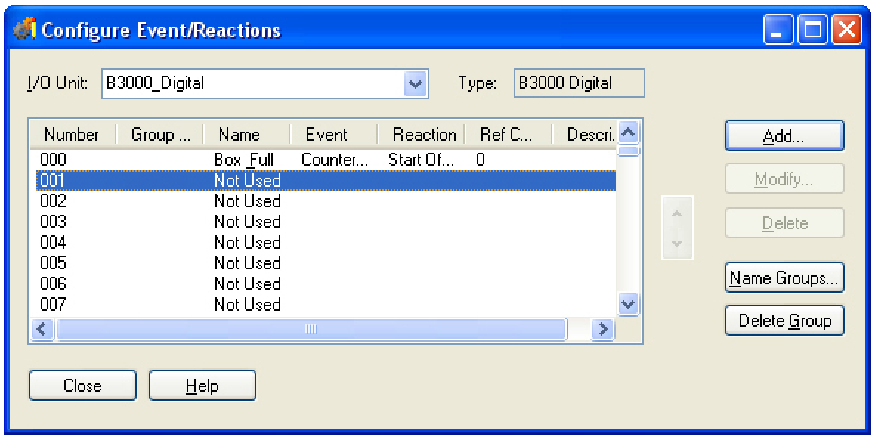
Here's an example of an Add Event/Reaction dialog box for an event/reaction involving a counter:


Add Event/Reaction

Name:	Box_Full
Description:	
Scan on Run:	[✓ Yes] No
Interrupt on Event:	Yes [✓ No]
Event Type:	Counter >= Value
I/O Point:	South_Packing_Station
Compare Value:	12
Reaction Type:	Start Off-Pulse
I/O Point:	Drop_Unit
Pulse Length:	5

✓ Enable communication

This event/reaction would appear in the Configure Event/Reactions dialog box like this:


Configure Event/Reactions

I/O Unit: B3000_Digital Type: B3000 Digital

Number	Group ...	Name	Event	Reaction	Ref C...	Descri...
000		Box_Full	Counter...	Start Of...	0	
001		Not Used				
002		Not Used				
003		Not Used				
004		Not Used				
005		Not Used				
006		Not Used				
007		Not Used				

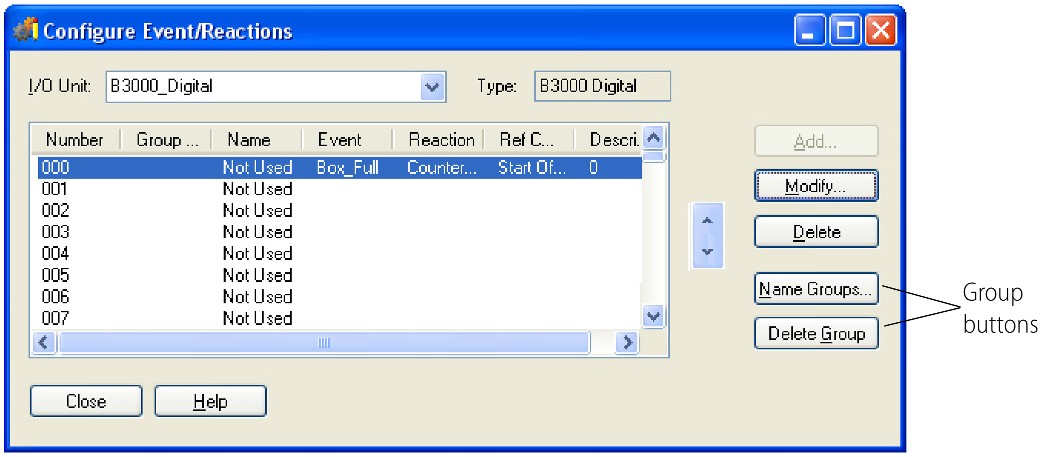
Using Event/Reaction Groups (*mistic* I/O Units Only)

(Pro only) Since you can configure up to 256 event/reactions on an I/O unit, it's useful to be able to divide them into groups of 16. By grouping related event/reactions, you can also take advantage of commands that start or stop scanning of all event/reactions in a group.

Creating Groups

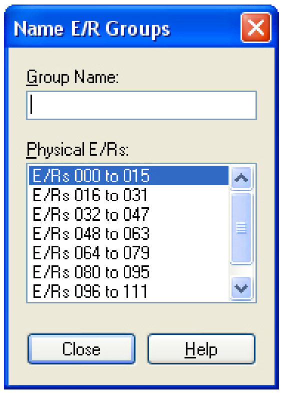
1. Make sure the strategy is open and in Configure mode. On the Strategy Tree, expand the I/O Units folder until you see the E/Rs folder for the I/O unit you want. Double-click the E/Rs folder.

   The Configure Event/Reaction dialog box opens, listing all configured event/reactions.



Number	Group ...	Name	Event	Reaction	Ref C...	Descri...
000		Not Used	Box_Full	Counter...	Start Of...	0
001		Not Used				
002		Not Used				
003		Not Used				
004		Not Used				
005		Not Used				
006		Not Used				
007		Not Used				

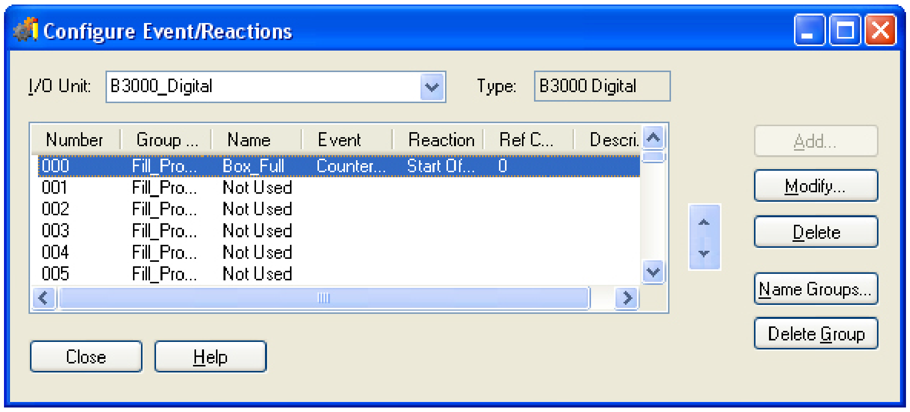
2. To create a group, click the Name Groups button.



3. Highlight a physical group of E/Rs. Click in the Group Name field and type a name.

   Group names must start with a letter and may contain letters, numbers, and underscores. (Spaces are converted to underscores.)

4. When you have finished naming groups, click Close. The names appear in the Group Name column of the Configure Event/Reaction dialog box.



Number	Group ...	Name	Event	Reaction	Ref C...	Descri.
000	Fill_Pro...	Box_Full	Counter...	Start Of...	0	
001	Fill_Pro...	Not Used				
002	Fill_Pro...	Not Used				
003	Fill_Pro...	Not Used				
004	Fill_Pro...	Not Used				
005	Fill_Pro...	Not Used				

Deleting Groups

When you delete a group, you're deleting only the group name, not the event/reactions that were assigned to the group.

To delete a group name, in the Configure Event/Reaction dialog box, select any event/reaction in the group and click the Delete Group button.

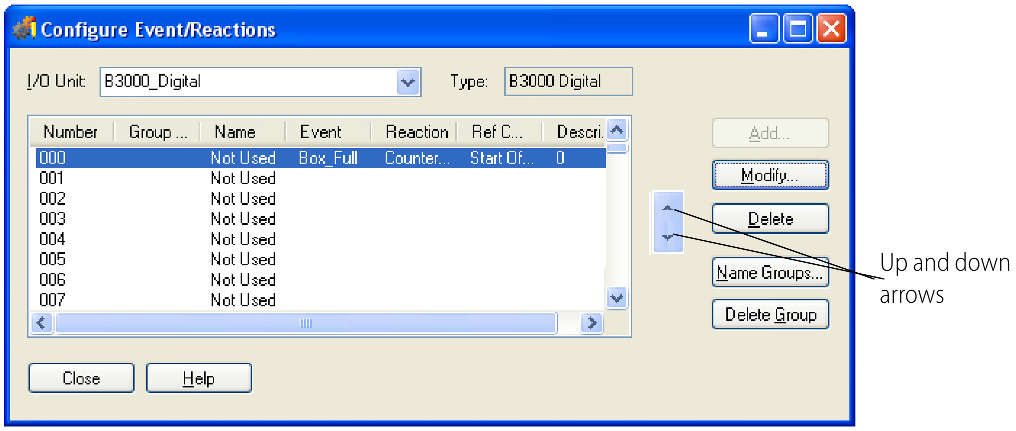
The group name is removed from all 16 event/reactions in the group, and the selected event/reaction appears at the top of the list box.

Changing Configured Event/Reactions (*mistic* I/O Units Only)

You can change an event/reaction's configuration and its position in the I/O unit.

1. Make sure the strategy is open and in Configure mode. On the Strategy Tree, expand the I/O Units folder until you see the E/Rs folder for the I/O unit you want. Double-click the E/Rs folder.

   The Configure Event/Reaction dialog box opens, listing all configured event/reactions.



Number	Group ...	Name	Event	Reaction	Ref C...	Descri.
000		Not Used	Box_Full	Counter...	Start Of...	0
001		Not Used				
002		Not Used				
003		Not Used				
004		Not Used				
005		Not Used				
006		Not Used				
007		Not Used				

   Event/reactions are scanned by the I/O unit in the order that they appear in this list.

2. To move an event/reaction to a different position on the I/O unit, use the up- and down-arrows in the dialog box.

3. To change an event/reaction's configuration, double-click its name to open the Edit Event/Reaction dialog box. Change the fields as necessary.

   For help in completing the fields, see "Configuring Event/Reactions" on page 162."Configuring Event/Reactions."

Deleting Event/Reactions (*mistic* I/O Units Only)

You can delete only event/reactions with a reference count of zero. Be careful when you delete; you cannot undo a deletion.

1. Make sure the strategy is open and in Configure mode. On the Strategy Tree, expand the I/O units folder until you see the event/reaction you want to delete.
2. Right-click the name of the event/reaction and choose Delete from the pop-up menu.

   The event/reaction is deleted.

   You can also delete an event/reaction in the Configure Event/Reaction dialog box by highlighting it and clicking Delete.

Inspecting I/O in Debug Mode

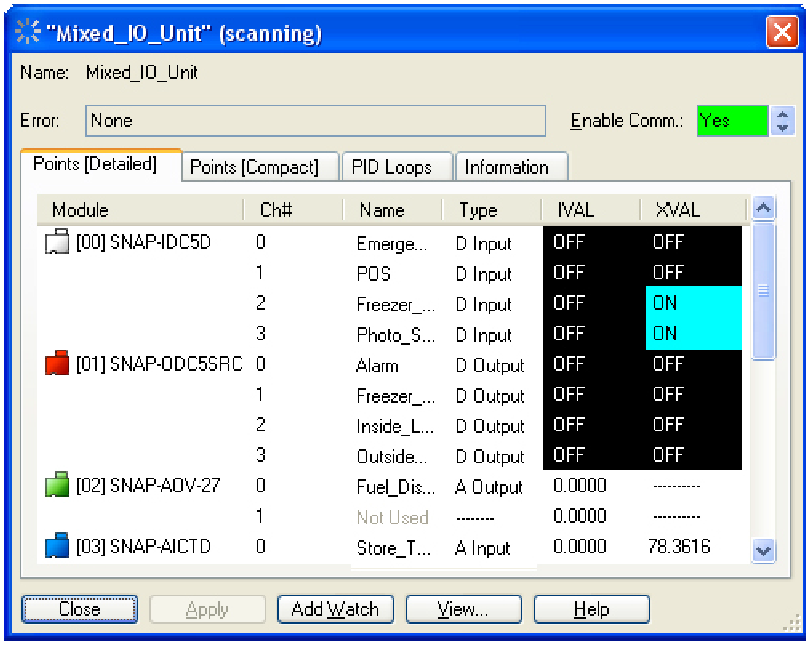
You may want to inspect or change I/O while you are running your strategy in Debug mode. This section shows how to view information about I/O and make changes while the strategy is running.

To monitor several I/O elements at once in a window you can save with your strategy, see "Using Watch Windows for Monitoring" on page 193.

Inspecting I/O Units and Saving Settings to Flash

1. With the strategy running in Debug mode, double-click an I/O unit in the Strategy Tree.

The Inspect I/O Unit dialog box appears, showing information about the unit and its points. The title bar shows the name of the I/O unit and whether scanning is occurring.



Name: Mixed_IO_Unit

Error: None

Enable Comm.: Yes

Module	Ch#	Name	Type	IVAL	XVAL
[00] SNAP-IDC5D	0	Emerge...	D Input	OFF	OFF
	1	POS	D Input	OFF	OFF
	2	Freezer_...	D Input	OFF	ON
	3	Photo_S...	D Input	OFF	ON
[01] SNAP-ODC5SRC	0	Alarm	D Output	OFF	OFF
	1	Freezer_...	D Output	OFF	OFF
	2	Inside_L...	D Output	OFF	OFF
	3	Outside...	D Output	OFF	OFF
[02] SNAP-AOV-27	0	Fuel_Dis...	A Output	0.0000	---------
	1	Not Used	--------	0.0000	---------
[03] SNAP-AICTD	0	Store_T...	A Input	0.0000	78.3616

NOTE: Scanning stops whenever you click a changeable field. It resumes once you click Apply, another button, or an unchangeable field. If scanning resumes before you click Apply, any changes you made are lost.

2. To save the current configuration of the I/O unit to its EEPROM (flash memory), click the Information tab and then click the Set button under Stored Configuration (Flash EEPROM).

The following parameters are saved:

Analog	Digital
• I/O module configuration • Initial output settings • Comm link watchdog time • Temperature conversion type • Input offset and gain settings	• I/O module configuration • Comm link watchdog time

NOTE: Use this option to protect the I/O unit configuration information from being lost when the power is turned off. You should save the configuration settings to flash when they are final.

Saving to flash memory this way is the same as saving to flash by other methods, such as using PAC Manager (see Opto 22 form 1704, the *PAC Manager User's Guide*), and is preferable to using the command Write I/O Unit Configuration to EEPROM.

3. To reset saved parameters to their powerup default values, click the Clear button.

4. To change the I/O unit's current status, click an arrow in the Enable Comm field. Then click Apply.

Yes on a green background means enabled; No on a red background means disabled. If you change it, the background turns magenta until you click Apply.

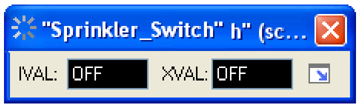
5. To view an individual I/O point, highlight its name in the list.
   - To add an I/O element to a watch window, click Add Watch. See page 193.
   - To open an inspection window to change an I/O point, click View. Then see "Inspecting Digital I/O Points" below for the I/O point you are changing (analog or digital).
6. When you have finished inspecting the I/O unit, click Close.

Inspecting Digital I/O Points

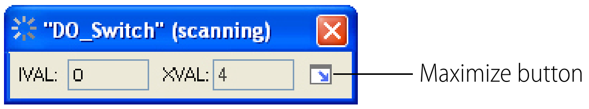
You can inspect a digital point's data, change its status, or set its internal values or external values in Debug mode. To monitor the point in a watch window, see page 193. To change the point, follow these steps.

1. With the strategy running in Debug mode, double-click the I/O point on the Strategy Tree. Or double-click an I/O unit in the Strategy Tree, click the Points (Compact) tab of the Inspect I/O Unit Dialog box, and then double-click the I/O point.

   The small dialog box that appears shows the IVAL and XVAL.
   - The *XVAL*, or external value, is the "real" or hardware value as seen by the I/O unit. This value is external to the control engine.
   - The *IVAL*, or internal value, is a logical or software copy of the XVAL that is in the control engine. The IVAL may or may not be current, since it is updated to match the XVAL only when a strategy in the control engine reads or writes to an I/O point.

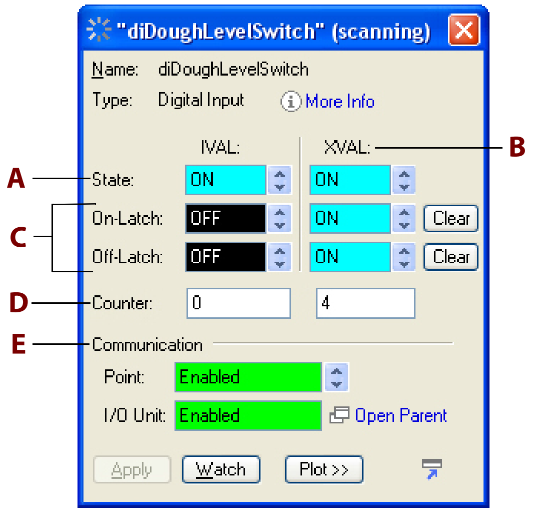


   If the digital point is configured with a counter, the counter values appear instead of the point's status.



2. To change the value or to view more information, click the Maximize button.

The title bar shows the name of the digital point and whether scanning is occurring.



Scanning stops whenever you click a changeable field. It resumes once you click Apply, another button, or an unchangeable field. If scanning resumes before you click Apply, any changes you made are lost. Use the Clear button to immediately clear the field.

Asterisks in a field indicate an out-of-range value. Dashes in an XVAL field indicate a communication error.

3. Change the fields as necessary:

A—*State.* The point's current internal value. Switch between On and Off; then click Apply.

B—*XVAL*. The point's current external value. Switch between On and Off; then click Apply.

C—*On-Latch/Off Latch.* The state of the point's on and off latches.

D—*Counter.* Internal and external feature values if the point has been configured with any special features, such as a counter.

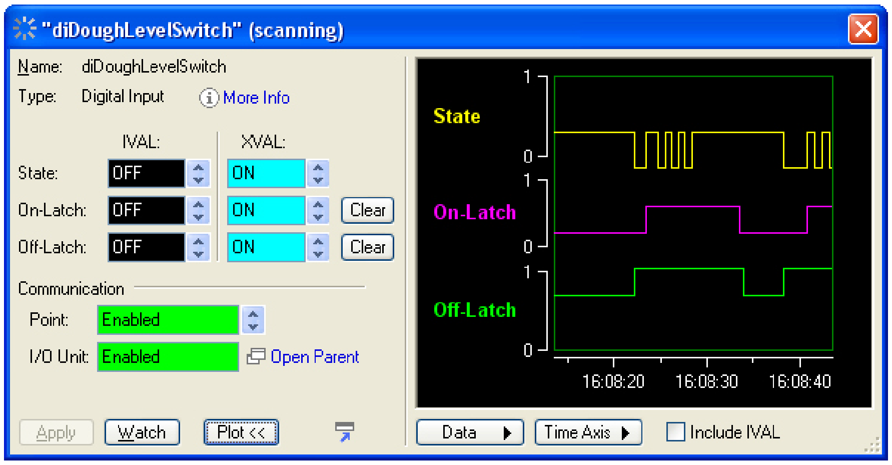
E—*Enable comm.* Current point status: Yes on a green background means enabled, No on a red background means disabled. To change the status, click one of the arrows; then click Apply.

4. To add the point to a watch window, click Watch and see page 193.

5. To view configuration information for the variable, click More Info.

6. To inspection I/O unit's configuration, click Open Parent.

7. (Pro only) To add a plot control, click Plot.



Name: diDoughLevelSwitch
Type: Digital Input

	IVAL:	XVAL:
State:	OFF	ON
On-Latch:	OFF	ON
Off-Latch:	OFF	ON

Communication
Point: Enabled
I/O Unit: Enabled

The Data button allows you to save, copy, or print the current plot. Use the Time Axis button to adjust the resolution. Click Include IVAL to see a concurrent plot of the IVAL.

Inspecting Analog I/O Points

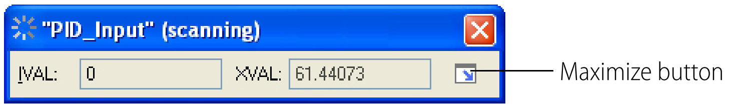
You can review an analog point's data, modify its status, or set its internal values or external values in Debug mode. To monitor the point in a watch window, see page 193. To change the point, follow these steps.

1. With the strategy running in Debug mode, double-click the I/O point on the Strategy Tree. Or double-click an I/O unit in the Strategy Tree, click the Points (Compact) tab of the Inspect I/O Unit dialog box, and then double-click the I/O point.

   The small dialog box that appears shows the IVAL and XVAL, as well as the units.

   – The *XVAL*, or external value, is the "real" or hardware value as seen by the I/O unit. This value is external to the control engine.

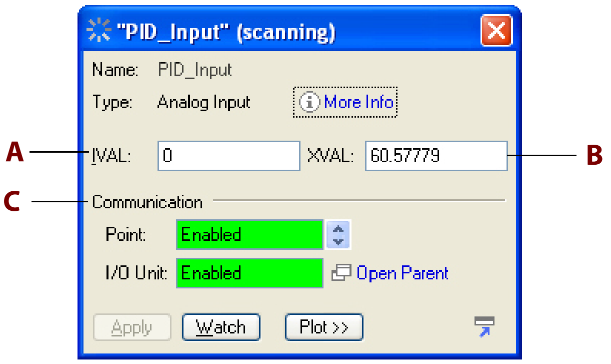
   – The *IVAL*, or internal value, is a logical or software copy of the XVAL that is in the control engine. The IVAL may or may not be current, since it is updated to match the XVAL only when a strategy in the control engine reads or writes to an I/O point.



IVAL: 0 XVAL: 61.44073

2. To change the value or to view more information, click the Maximize button.

The title bar shows the name of the analog point and whether scanning is occurring.



Scanning stops whenever you click a changeable field. It resumes once you click Apply, another button, or an unchangeable field. If scanning resumes before you click Apply, any changes you made are lost.

Asterisks in a field indicate an out-of-range value. Dashes in an XVAL field indicate a communication error.

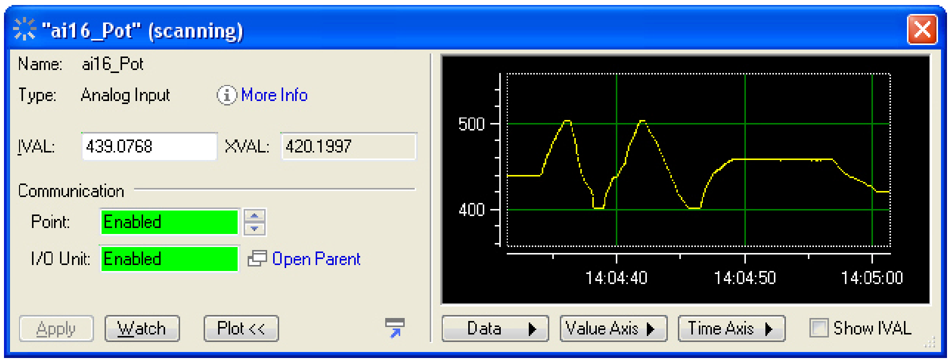
3. Change the fields as necessary:

   A—*IVAL*. The point's current internal value. You can change it to any value within the valid range of the analog point. For an input, the valid range may exceed the apparent range; that is, you may be able to enter a value lower than the zero-scale value or higher than the full-scale value. For an output however, you cannot enter a value outside of the range defined by the zero-scale and full-scale values. After you change it, click Apply.

   B—*XVAL*. The point's current external value. You can change it to any value within the valid range of the analog point; then click Apply.

   C—*Enable comm*. Current point status: Yes on a green background means enabled, No on a red background means disabled. To change the status, click one of the arrows; then click Apply.

4. To add the point to a watch window, click Watch and see page 193.
5. To view configuration information for the variable, click More Info.
6. To inspect the I/O unit's configuration, click Open Parent.
7. (Pro only) To add a plot control, click Plot.



The Data button allows you to save, copy, or print the current plot. Use the Time Axis and Value Axis buttons to adjust the resolution. Click Include IVAL to see a concurrent plot of the IVAL.

Inspecting Event/Reactions

You can review an event/reaction's current state, modify its status, or set its internal values or external values in Debug mode. To monitor the event/reaction in a watch window, see "Using Watch Windows for Monitoring" on page 193. To change the event/reaction, follow these steps.

1. With the strategy running in Debug mode, double-click the event/reaction on the Strategy Tree. Or double-click an I/O unit in the Strategy Tree, click the Information tab of the Inspect I/O Unit Dialog Box, and then double-click the event/reaction.

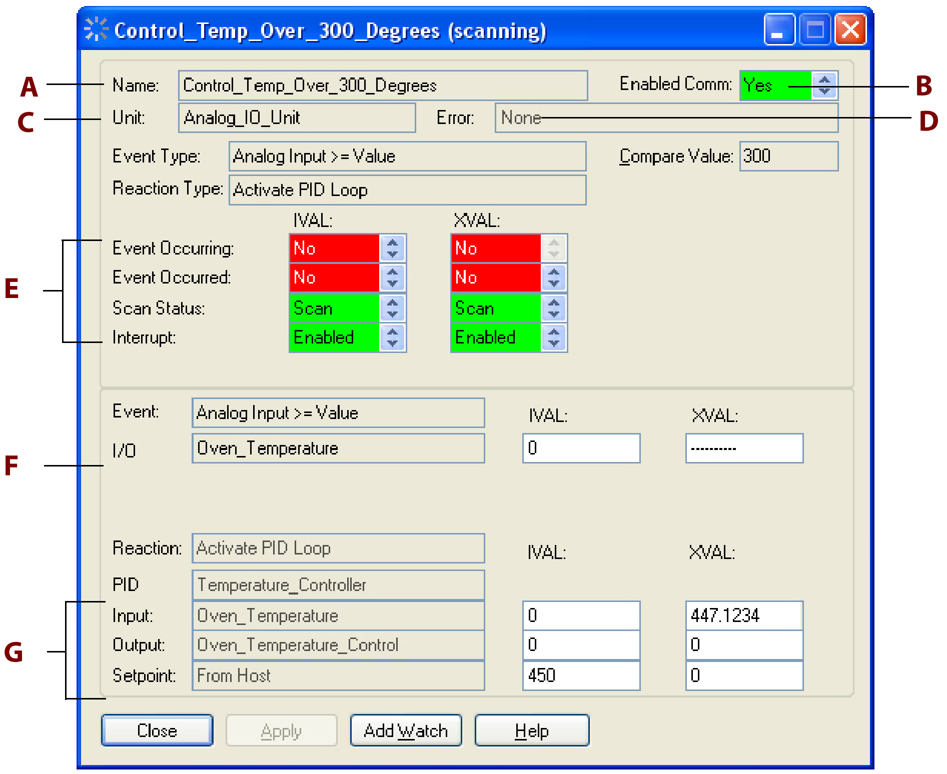
   The View Event/Reaction dialog box appears, showing the configuration parameters for the event/reaction. The title bar shows the name of the event/reaction and whether scanning is occurring.

   Scanning stops whenever you click a changeable field. It resumes once you click Apply, another button, or an unchangeable field. If scanning resumes before you click Apply, any changes you made are lost.

   Asterisks in a field indicate an out-of-range value. Dashes in an XVAL field indicate a communication error.

2. Change the fields as described in "View Event/Reaction Dialog Box" below.
3. To add the event/reaction to a watch window, click Add Watch and see page 193.

View Event/Reaction Dialog Box



Name: Control_Temp_Over_300_Degrees
Enabled Comm: Yes
Unit: Analog_IO_Unit
Error: None
Event Type: Analog Input >= Value
Compare Value: 300
Reaction Type: Activate PID Loop

	IVAL:	XVAL:
Event Occurring:	No	No
Event Occurred:	No	No
Scan Status:	Scan	Scan
Interrupt:	Enabled	Enabled

		IVAL:	XVAL:
Event:	Analog Input >= Value		
I/O	Oven_Temperature	0	----------

		IVAL:	XVAL:
Reaction:	Activate PID Loop		
PID	Temperature_Controller		
Input:	Oven_Temperature	0	447.1234
Output:	Oven_Temperature_Control	0	0
Setpoint:	From Host	450	0

A—*Name.* Name of the event/reaction.

B—*Enabled.* Current status: Yes on a green background means enabled, No on a red background means disabled. To change status, click one of the arrows; then click Apply.

C—*Unit.* The I/O unit on which the event/reaction is configured.

D—*Error.* Any communication error appears here with a red background.

E—*Event Occurring / Event Occurred / Scan Status.* Internal and external values for whether the event occurred or is occurring, whether scanning is occurring, and whether an interrupt is to be generated when the event occurs. If the reaction type involves a read-and-hold operation, a Read & Hold Value field also appears in this area, together with its internal and external values. You can change any of the internal or external values. If you do, click Apply.

F—*Event Parameters.* Parameters of the event, together with their internal and external values. (This example shows an I/O point being monitored for the event. For a MOMO Match event, MOMO data will appear instead. See "MOMO Event/Reactions" below.) You can change the internal or external values. If you do, click Apply.

G—*Reaction Parameters.* Parameters of the reaction, together with their internal and external values. (This example shows the PID to be activated and the sources of its input, output, and setpoint values. We also see the internal and external values of the input, output, and setpoint.) Depending on the reaction type, other parameters that can appear here include an I/O point, an event to be triggered, or MOMO data. You can change any of the internal or external values. If you do, click Apply.

Pro

MOMO Event/Reactions

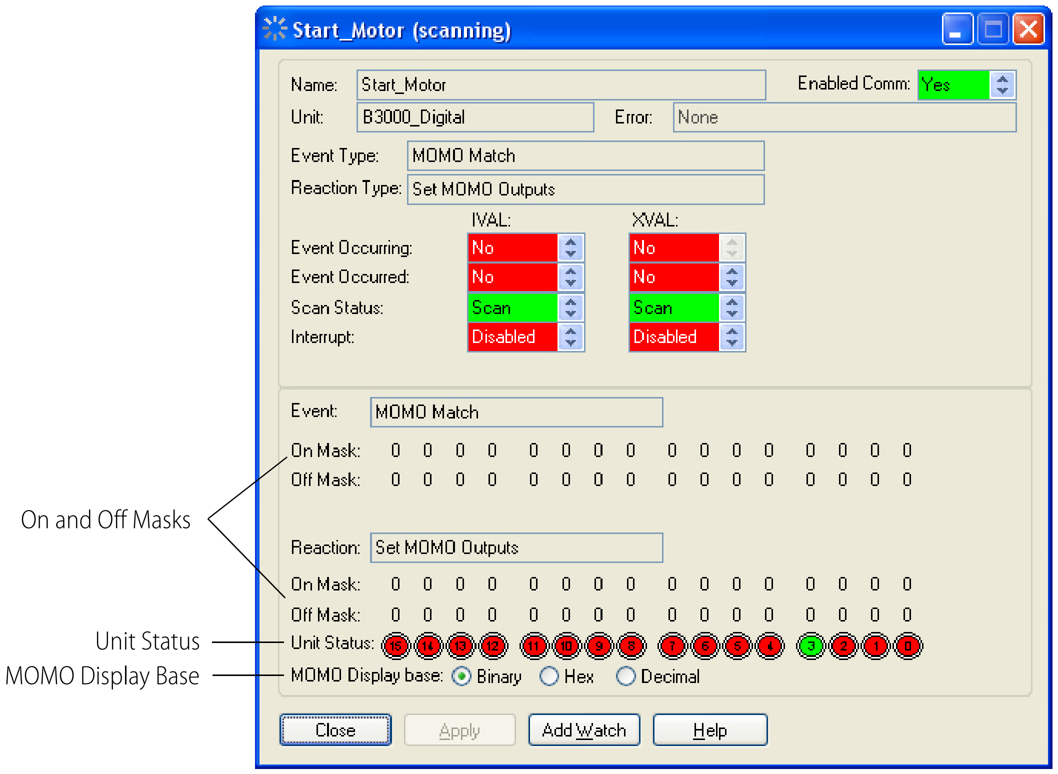
When a MOMO (must-on, must-off) event or reaction is involved, the bottom of the event/reaction dialog box displays the following:

On and Off Masks. Shows On and Off Masks for the MOMO event and reaction.

Unit Status. Shows LEDs representing the external value of the digital I/O unit on which the event/reaction is configured. Green represents one (on), red represents zero (off), and gray represents no value reported. If there is no MOMO reaction, this set of LEDs appears below the event's on and off masks.

Momo Display Base. Shows the display base for the MOMO event or reaction. You can switch among binary, hex, and decimal values.

You cannot change the internal values or external values of the MOMO masks or of the I/O unit status. However, other fields are the same as for other event/reactions.



Start_Motor (scanning)

Name: Start_Motor
Enabled Comm: Yes
Unit: B3000_Digital
Error: None
Event Type: MOMO Match
Reaction Type: Set MOMO Outputs

	IVAL:	XVAL:
Event Occurring:	No	No
Event Occurred:	No	No
Scan Status:	Scan	Scan
Interrupt:	Disabled	Disabled

Event: MOMO Match

```
On Mask:   0 0 0 0   0 0 0 0   0 0 0 0   0 0 0 0
Off Mask:  0 0 0 0   0 0 0 0   0 0 0 0   0 0 0 0
```

Reaction: Set MOMO Outputs

```
On Mask:   0 0 0 0   0 0 0 0   0 0 0 0   0 0 0 0
Off Mask:  0 0 0 0   0 0 0 0   0 0 0 0   0 0 0 0
Unit Status: 15 14 13 12   11 10 9 8   7 6 5 4   3 2 1 0
```

MOMO Display base: [✓ Binary] Hex Decimal

Close | Apply | Add Watch | Help

The example above shows on and off masks for both the MOMO Match event and the Set MOMO Outputs reaction.

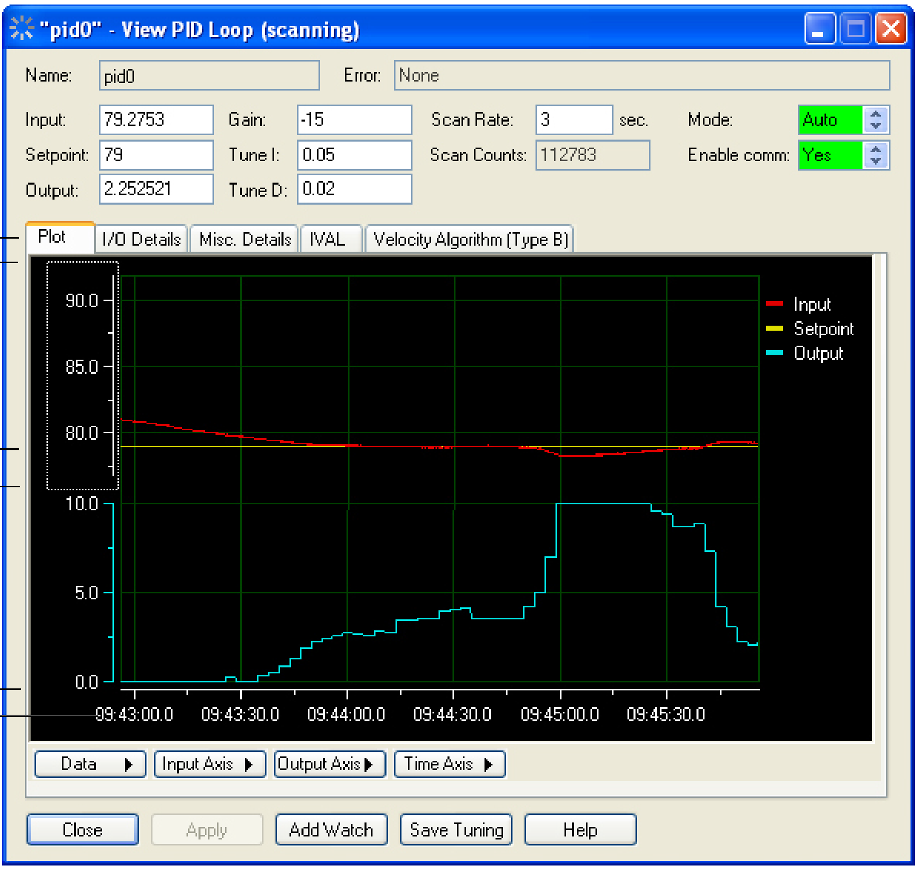
Inspecting and Tuning PID Loops

In Debug mode, you can view PID loops and tune them. This section gives you basic steps for inspecting PIDs, determining system lag, and tuning PIDs. For additional information, see form 2171, *Tuning a PID Control Loop Technical Note*, and form 1641, *OptoTutorial: SNAP PAC PID*. Both forms are available for download from our website at www.opto22.com.

Inspecting a PID (SNAP PAC)

This section applies to SNAP PAC, Ethernet I/O, and SNAP Ultimate I/O units only. For serial *mistic* I/O units, see page 191.

1. With the strategy running in Debug mode, double-click the PID on the Strategy Tree.



Name: pid0
Error: None
Input: 79.2753
Gain: -15
Scan Rate: 3 sec.
Mode: Auto
Setpoint: 79
Tune I: 0.05
Scan Counts: 112783
Enable comm: Yes
Output: 2.252521
Tune D: 0.02

Click a tab to see or change additional data.

Setpoint and Input plot. Adjust resolution using the Input Axis button below the plot. Click and drag on the scale to move the line.

Output plot. Adjust resolution using the Output Axis button. Click and drag on the scale to move the line.

Time axis. Adjust resolution using the Time Axis button. Click and drag left or right to see other times.

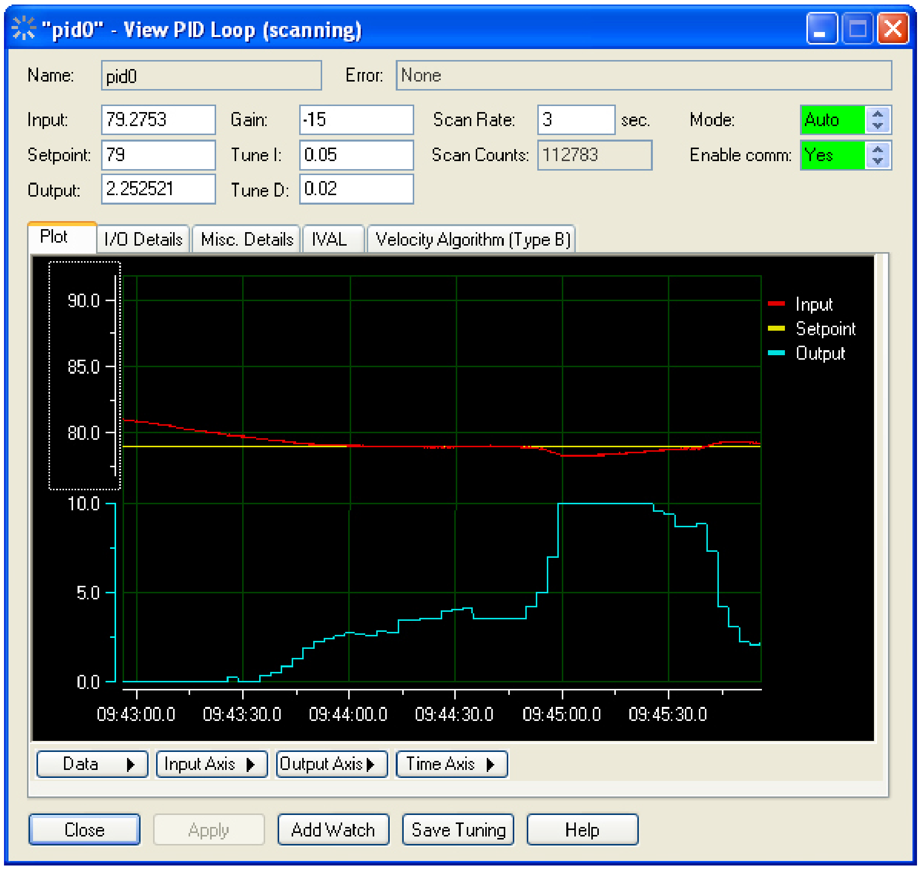
2. View or change PID parameters as necessary. Click the other tabs to see additional data. To tune the PID, see page 185.
3. To add the PID to a watch window, click Add Watch and see page 193.
4. To save, copy, or print the current plot, click the Data button and choose from the pop-up menu.
5. To save changes to any of the PID configuration parameters, click Save Tuning.

Determining System Lag

You can directly control the PID output to determine system lag, which is essential to setting the PID scan rate. Also see form 2171, *Tuning a PID Control Loop Technical Note*, available for download from our website at www.opto22.com.

1. Determine two significantly different output settings that are within the capabilities of your system.

2. With the strategy running in Debug mode, double-click the PID on the Strategy Tree.


"pid0" - View PID Loop (scanning)

Name: pid0 Error: None

Input: 79.2753 Gain: -15 Scan Rate: 3 sec. Mode: Auto

Setpoint: 79 Tune I: 0.05 Scan Counts: 112783 Enable comm: Yes

Output: 2.252521 Tune D: 0.02

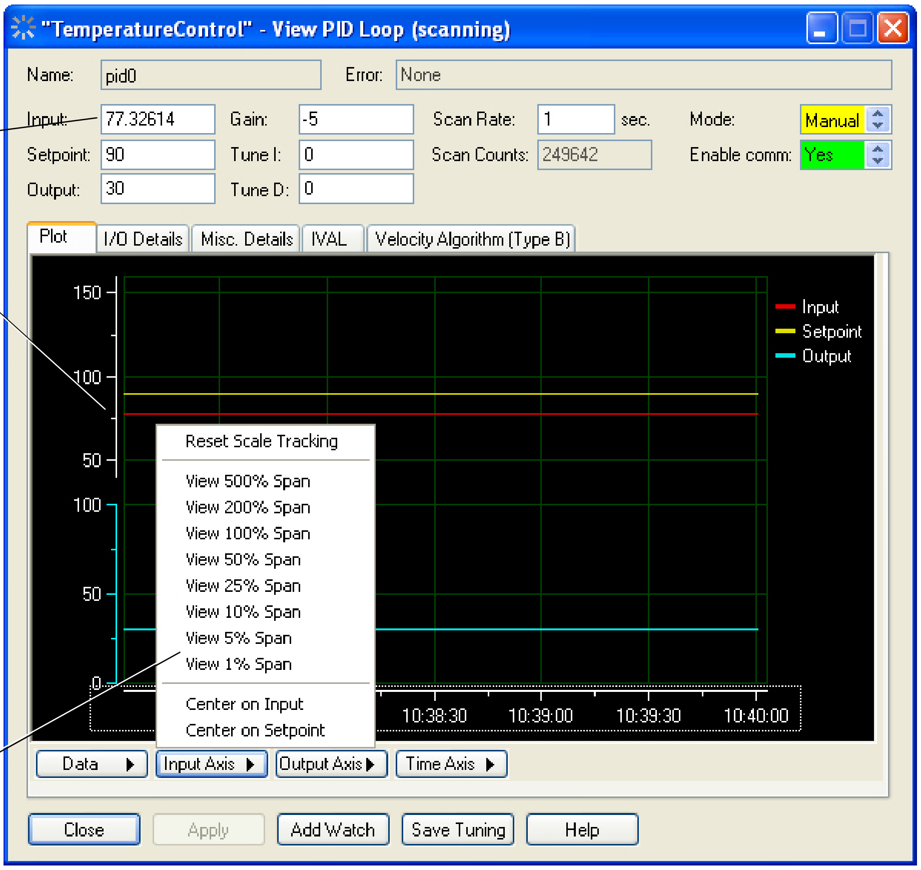
3. Set the Mode to Manual (if not set already) and click Apply.
4. In the Output field, type one of your two output settings and click Apply.

   Use an output value typical of your system but low enough to allow you to change output by 20%.
5. Reset your time axis, if necessary, by clicking the Time Axis and choosing from the pop-up menu. Then choose Reset Scale Tracking from the same menu.

   The span setting varies according to your system; a 3-minute span is often suitable. Until you are familiar with the PID plot, it is recommended that you avoid using the shortest settings (10-seconds or 1-second). After you've observed a change in the input, you can zoom in on the graph, which is described later.
6. Wait for your system to stabilize.

The system is stable when the Input value does not vary significantly (some drift can be expected). Stabilization may take several minutes, depending on the system.

A stable system exhibits little change in the Input value, which is shown numerically and graphically.



Name:	pid0	Error:	None				
Input:	77.32614	Gain:	-5	Scan Rate:	1 sec.	Mode:	Manual
Setpoint:	90	Tune I:	0	Scan Counts:	249642	Enable comm:	Yes
Output:	30	Tune D:	0				

Adjust resolution of the Input Axis by clicking the Input Axis button.

7. Increase the resolution of the Input Axis by clicking the Input Axis menu and choosing a span setting of 1 or 5 percent.

8. Center the Input Axis, if necessary, by clicking the red line at its left end and dragging it up or down until the plot is visible.

9. Under the Time Axis menu, choose Reset Scale Tracking.

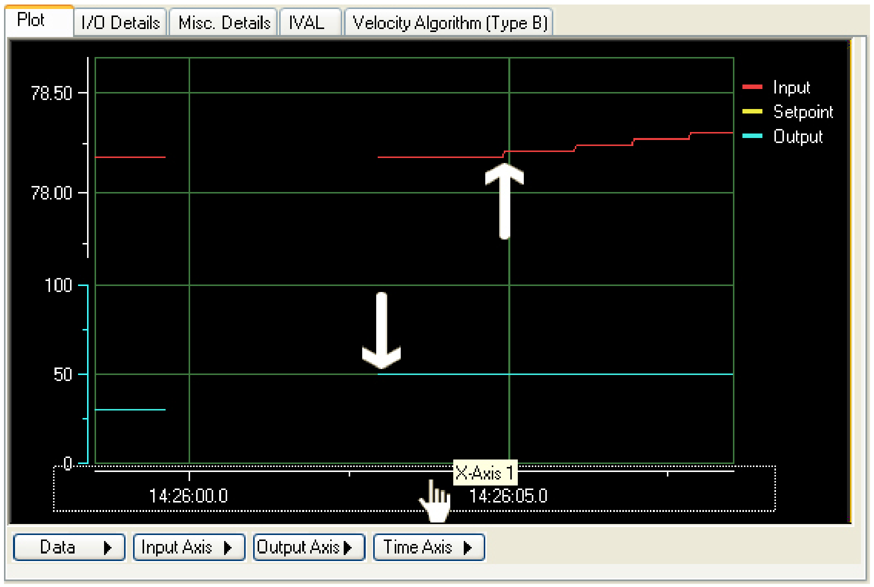
This is a precautionary step, as changing settings on the plot can fix the plot at a certain point in time. Resetting the time axis ensures that you are viewing the real-time values.

10. In the Output field, type the other of your two output settings and click Apply.

A 20% percent increase is a moderate, detectable change for most systems. Your system may require a larger or a smaller change to stay within safety constraints.

11. Wait for a discernible change in the Input axis.

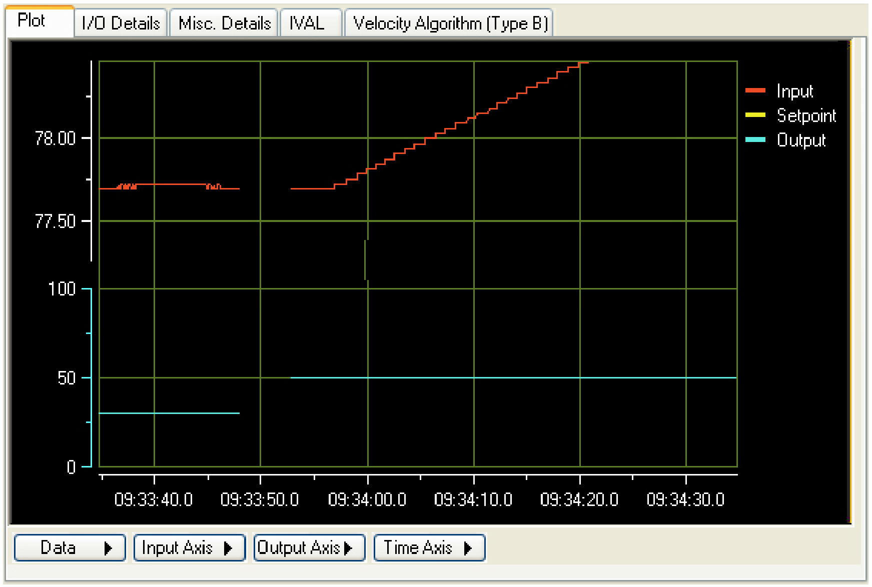
The Input axis (indicated by the upper white arrow added to this picture) begins to respond to the change in output (lower white arrow).



12. Increase the resolution of the Time Axis by clicking the Time Axis button and choosing a lower percentage, such as View 1 Minute Span.

13. Scroll the Time Axis to locate the point at which you changed the Output.

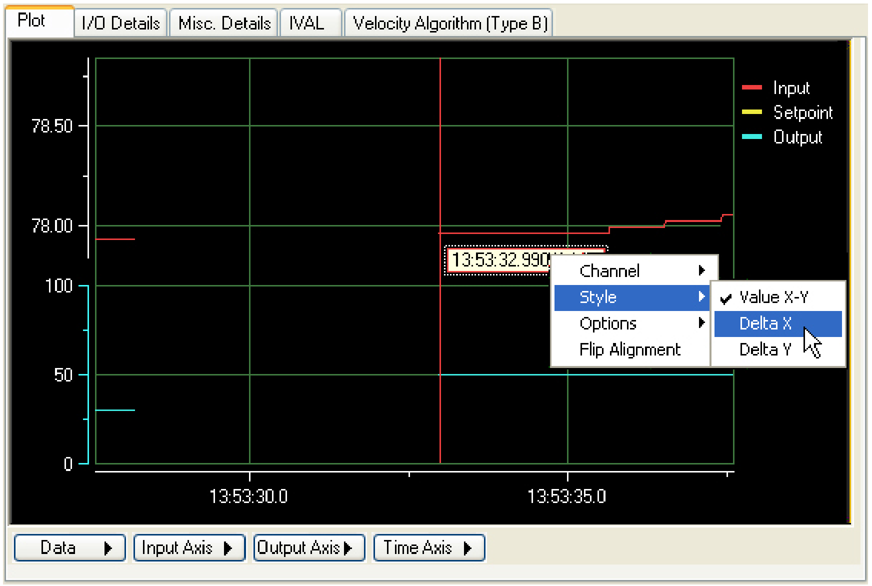
Both the time and input axes should display the point at which the Output changed, the lag, and the point the input changed. If not, adjust the Input axis and Time axis, until this information is displayed.



14. From the Data menu, select Cursor.

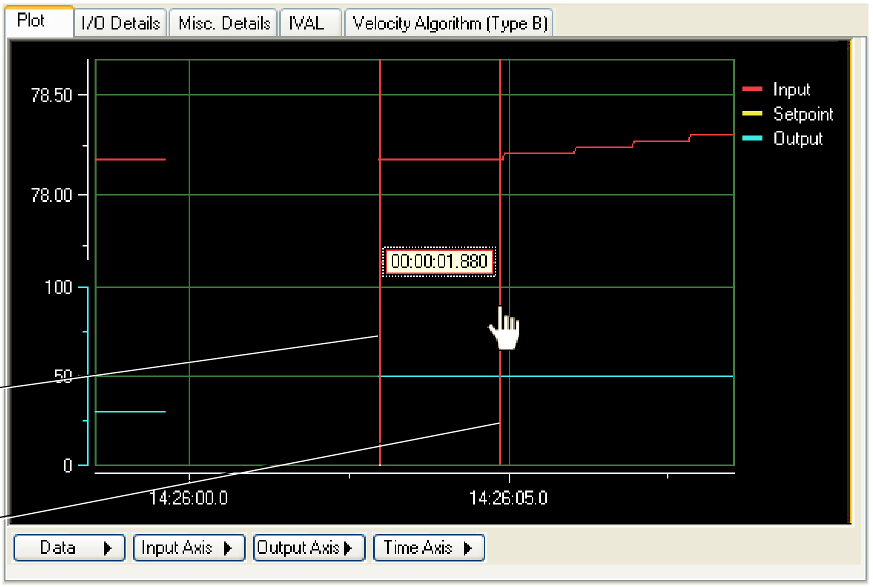
The data cursor, a line with a value bar attached to it, appears on the plot.

15. Right-click the data cursor and choose Delta X from the Style submenu, as shown below.



16. To measure the system lag (the time between the change in output and the change in input), click and drag the first vertical red bar to just after the output change. Drag the second bar to just before the change in input.

The Delta X cursor displays the time difference between the two vertical bars.



Drag the first bar into position after the Output change.

Drag the second bar to a position just before the change in Input.

If you are unable to get the precision you want, you can view the plot at a lower time span, such as 10 seconds. You will need to reposition your plot and the measurement bars of the Delta X cursor.

The example above shows a system lag of 1.88 seconds. Generally, a suitable scan rate can be anywhere from one-third the system lag to two times the system lag.

Considerations in setting scan rate are:

- Slower scan intervals may be easier to tune. The PID controller has time to see the effect of the previous output before calculating a new output.
- Faster scan rates may be necessary to achieve the desired response. When scan intervals are shorter than the system lag, tuning must compensate for any over-correction from the controller output.

Tuning a PID Loop (Ethernet)

NOTE: This section applies to SNAP Ethernet and SNAP Ultimate I/O units only. For information on tuning PID loops on serial-based mistic *I/O units, see "PID—mistic Commands" on page 349.*

Tuning a PID involves manipulating the P, I, and D constants in real time. The following steps should be viewed as general suggestions to show you features that are available for tuning. We highly recommend form 1641, *OptoTutorial: SNAP PAC PID*, and form 2171, *Tuning a PID Control Loop Technical Note*, for more detailed information. These forms are available for download from our website at www.opto22.com.

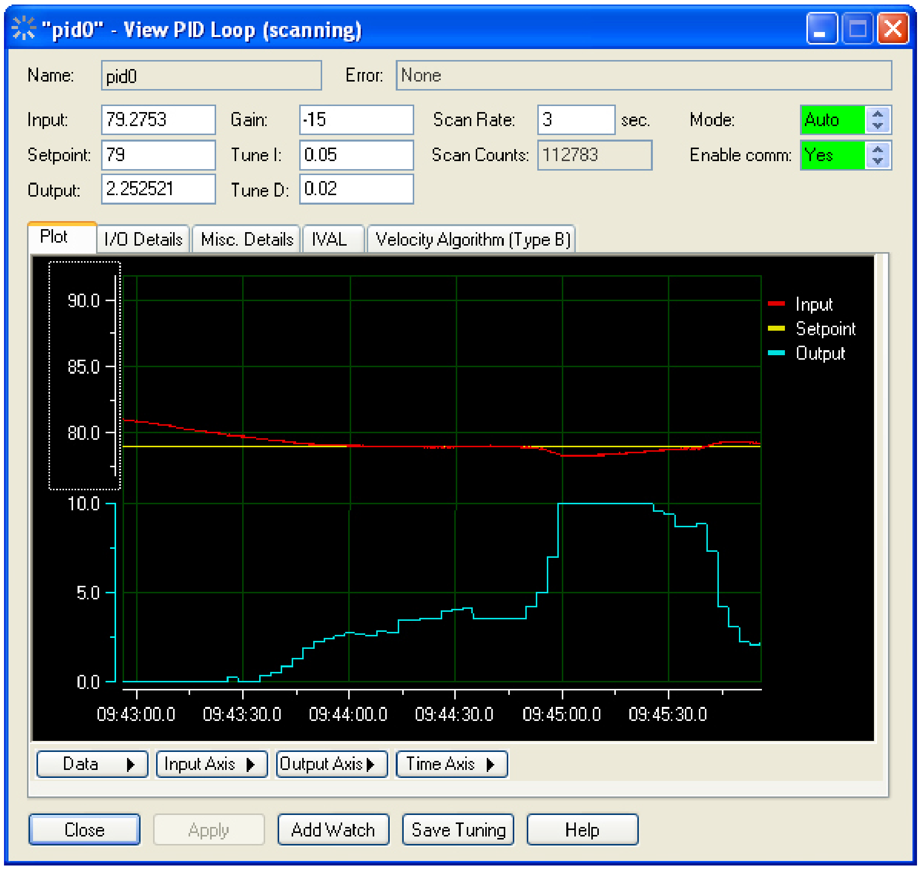
CAUTION*: Before following these procedures, make sure you know the limits of the equipment being controlled and monitored by your PID loop. Also, make sure that these points are configured properly. Any values suggested in these steps are for example only and must be modified according to the capabilities and constraints of your system.*

1. Make sure the following PID features have already been configured:
   - Scan Rate
   - Input
   - Input low range and high range
   - Output
   - Output lower and upper clamp.
   - Algorithm
   - Setpoint. If your setpoint changes during normal operation, tune your PID with the setpoint configured to host, so you can simulate setpoints from an input point or from another PID.
   - Gain. A final gain constant will be determined by tuning, but before you can tune your PID, your gain constant must be either a positive or negative number according to the type of system you have. For example, a heating system reports a negative error when heat needs to be applied; a negative gain constant turns this error into a positive output for the heater. Alternatively, a cooling system reports a positive error when the input exceeds the setpoint; a positive gain constant maintains the positive output to the chiller.
   - Optional, depending on your system: Minimum and maximum changes to Output and Output forcing when the Input is out of range.

2. Download and run your strategy.

   The current PID configuration is written to the I/O unit. You can stop your strategy at this point if you wish, as the PID will continue operating.

3. In Debug mode, double-click the PID on the Strategy Tree.


"pid0" - View PID Loop (scanning)

Name: pid0 Error: None

Input: 79.2753 Gain: -15 Scan Rate: 3 sec. Mode: Auto

Setpoint: 79 Tune I: 0.05 Scan Counts: 112783 Enable comm: Yes

Output: 2.252521 Tune D: 0.02

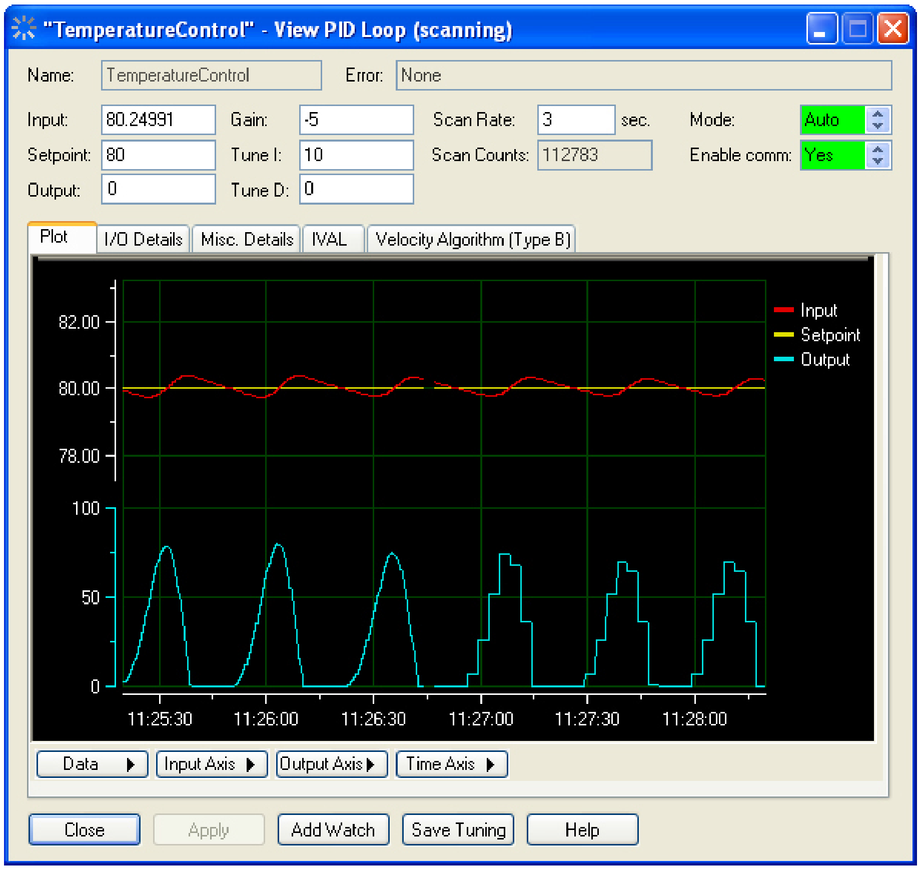
4. Set the PID Mode to Auto (if not set already) and click Apply.
5. Change the Setpoint, if desired, by typing a new setpoint and clicking Apply. (Setpoint must be configured as Host.)

   Depending on the type of system, your PID may maintain a setpoint or respond to changes in setpoint. Experiment with setpoint changes again after tuning the P, I, and D constants.
6. Adjust the span of the input, output, and time axes according to how much change you expect from your system. To set a span, click the axis button and choose from the pop-up menu.
7. If desired, type a new Scan Rate and click Apply.

   For most systems, you should use an appropriate scan rate based on the system lag (see "Determining System Lag" on page 180). However, you can experiment with Scan Rates before tuning the P, I, and D constants or adjust scan rate after tuning.

Here is an example for Scan Rate:

The lag for this system was determined to be about 2 seconds. The left half of the plot reflects a 0.5-second scan rate, while the right half shows a 3-second scan rate. Notice both scan rates have the same effect on the input; however, the 3-second scan rate is using less of the processor's resources.



"TemperatureControl" - View PID Loop (scanning)

Name: TemperatureControl
Error: None

Input: 80.24991
Gain: -5
Scan Rate: 3 sec.
Mode: Auto

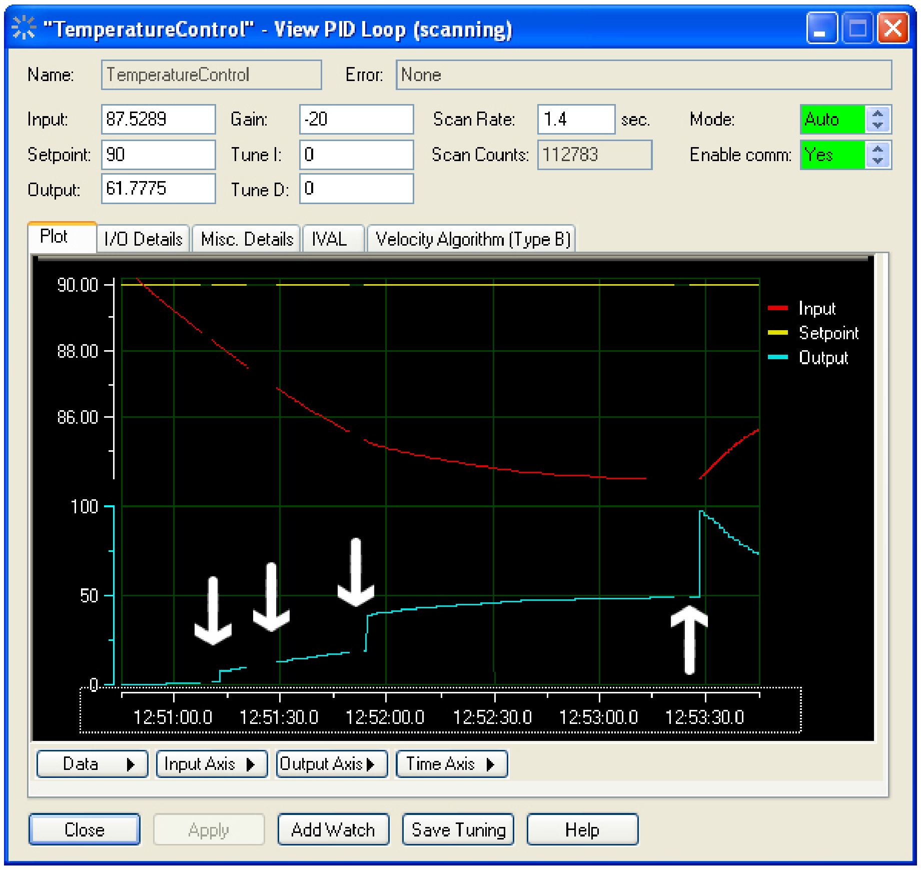
Setpoint: 80
Tune I: 10
Scan Counts: 112783
Enable comm: Yes

Output: 0
Tune D: 0

8. Experiment with gain settings by typing a new value in the Gain field and clicking Apply.

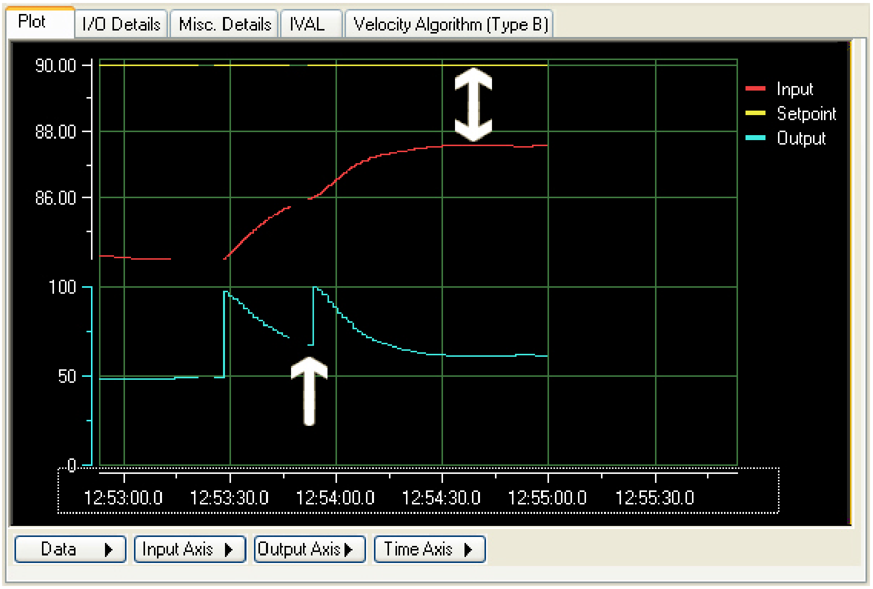
The easiest way to tune most PIDs is to experiment with the gain constant first. Try various gains to see how well the system stays at setpoint and responds to setpoint changes.

In the example below, the white arrows (added for the example) show where gain constants of -2, -5, -10, and -20 were applied:


Name: TemperatureControl
Error: None
Input: 87.5289
Gain: -20
Scan Rate: 1.4 sec.
Mode: Auto
Setpoint: 90
Tune I: 0
Scan Counts: 112783
Enable comm: Yes
Output: 61.7775
Tune D: 0

In this example, a gain setting of -30 revealed an offset error:

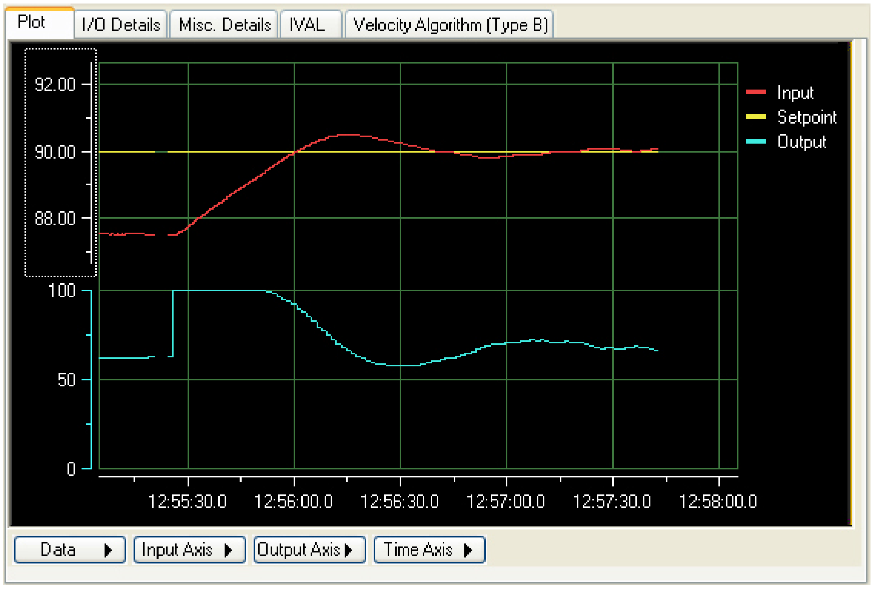
With only a gain constant applied, the input often stabilizes at an incorrect value. In this heating example, a gain setting of -30 drove the input close to the setpoint, but subsequent increases failed to eliminate the offset. It is time to try integral constants to eliminate the offset error.



9. Experiment with the integral constant: in the Tune I field, type a number between 0 and 1 and click Apply. (Your PID may require larger numbers.)

In this example, an integral constant of 0.1 corrected the offset error.

The far left side of the plot shows the offset before an integral constant of 0.1 was applied. This setting eliminated the offset. In many applications, a minor fluctuation around the setpoint is acceptable, and these applications use gain and integral only. In some applications, however, the fluctuations at the setpoint indicate that the gain is too high (too much gain makes a system unstable) or that a derivative constant is required.

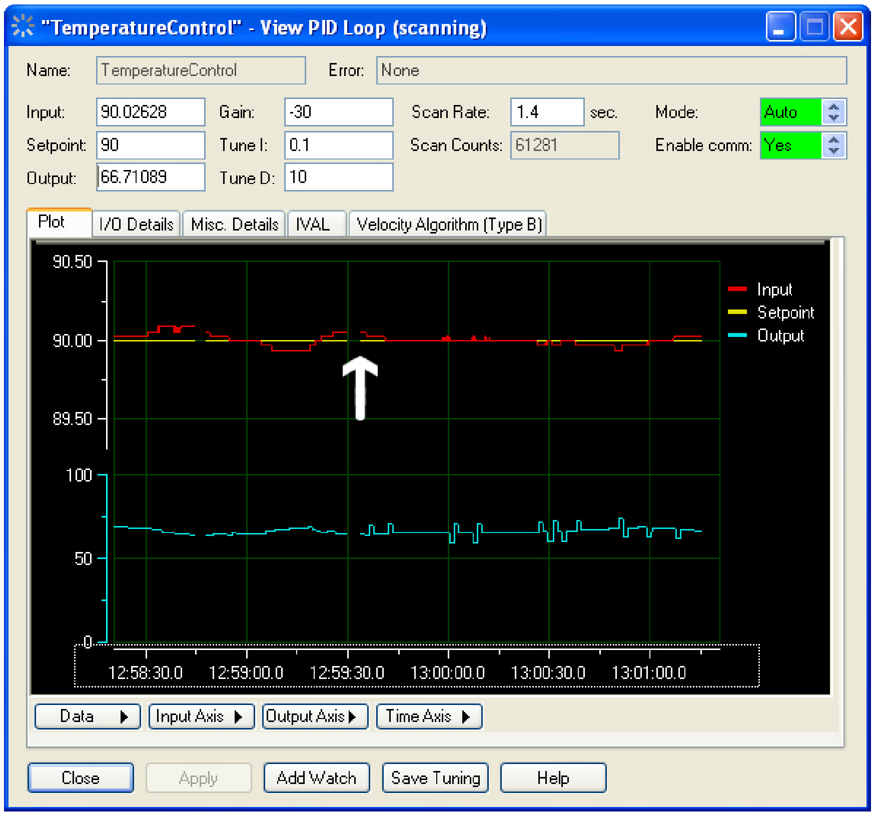


10. If derivative correction is needed, experiment with the derivative constant: in the Tune D field, type 1 and click Apply. (Your PID loop may require a larger number.)

In this example, a derivative of 10 makes a noticeable difference in keeping the input near the setpoint.

Many PID systems are effectively controlled with gain and integral constants only and no derivative constant. In this example, the gain and integral settings are maintaining the temperature at 0.06 from setpoint. To demonstrate the effect of the derivative constant, the resolution of the input axis was increased to show a 1 percent span. At this resolution, the plot reveals changes of 0.01 degrees F.

The left side of the plot shows the effect of gain at -30, integral at 0.1, and no derivative constant. The arrow shows when a derivative constant of 10 was applied. The right side of the plot shows how the derivative constant is keeping the input closer to setpoint.


"TemperatureControl" - View PID Loop (scanning)

Name: TemperatureControl   Error: None

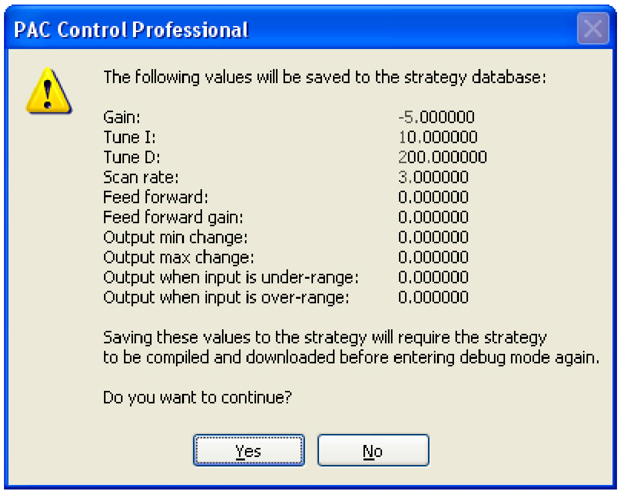
Input: 90.02628   Gain: -30   Scan Rate: 1.4 sec.   Mode: Auto

Setpoint: 90   Tune I: 0.1   Scan Counts: 61281   Enable comm: Yes

Output: 66.71089   Tune D: 10

11. Click Save Tuning to save your tuning parameters to the strategy database.

Changes are lost unless you save them. You may wish to save your tuning parameters when you see any improvement in performance, even if they are not final.


PAC Control Professional

The following values will be saved to the strategy database:

Gain:	-5.000000
Tune I:	10.000000
Tune D:	200.000000
Scan rate:	3.000000
Feed forward:	0.000000
Feed forward gain:	0.000000
Output min change:	0.000000
Output max change:	0.000000
Output when input is under-range:	0.000000
Output when input is over-range:	0.000000

Saving these values to the strategy will require the strategy to be compiled and downloaded before entering debug mode again.

Do you want to continue?

12. Click Yes.

Values are saved to the PAC Control strategy. Remember to save PID parameters to the I/O unit's flash memory, too (see "Inspecting I/O Units and Saving Settings to Flash" on page 172).

Inspecting a PID Loop (*mistic*)

This section applies to serial *mistic* I/O units only. For SNAP Ethernet and SNAP Ultimate I/O units, see page 179.

You can review a PID loop's data, modify its status, or set its internal values or external values in Debug mode. To monitor the PID loop in a watch window, see page 193. To change the PID loop, follow these steps.

1. With the strategy running in Debug mode, double-click the PID loop on the Strategy Tree, or double-click it in the Inspect I/O Unit dialog box.

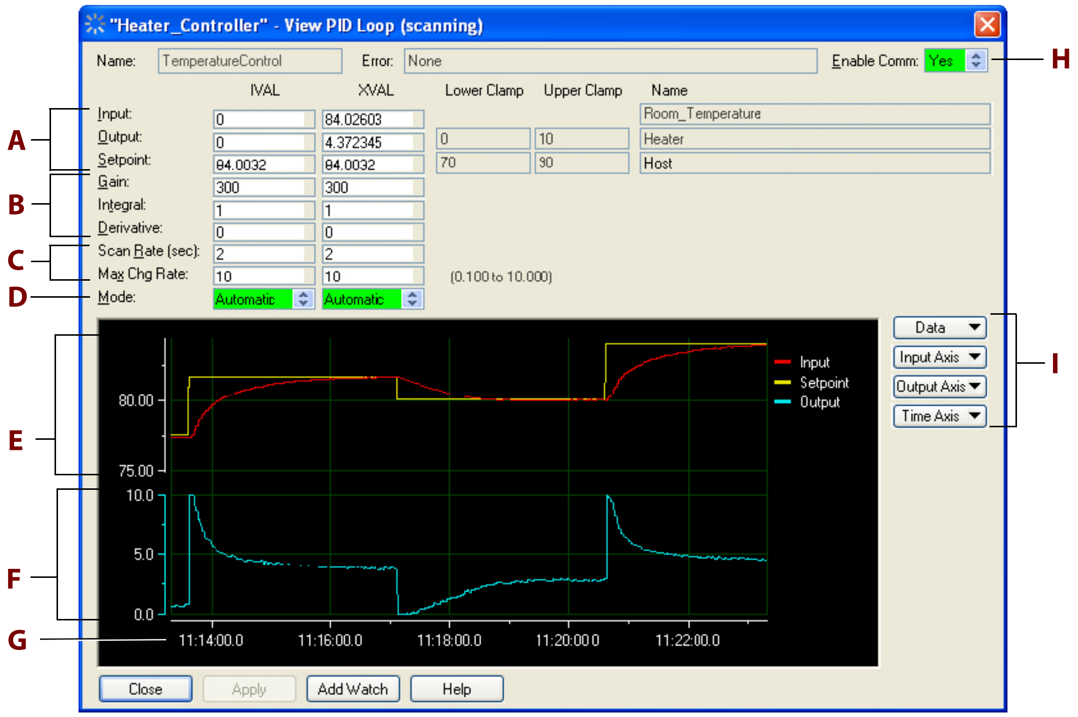
   The View PID Loop dialog box appears (see "View PID Loop (mistic) Dialog" below), showing the configuration parameters for the PID loop and several values you can change. The title bar shows the name of the PID loop and whether scanning is occurring.

   Scanning stops whenever you click a changeable field. It resumes once you click Apply, another button, or an unchangeable field. If scanning resumes before you click Apply, any changes you made are lost.

   Asterisks in a field indicate an out-of-range value. Dashes in an XVAL field indicate a communication error.

2. Change the fields as necessary. See "View PID Loop (mistic) Dialog" below.
3. To add the PID loop to a watch window, click Add Watch and see page 193.
4. To save, copy, or print the current plot, click the Data button and choose from the pop-up menu.

View PID Loop (*mistic*) Dialog


"Heater_Controller" - View PID Loop (scanning)

Name: TemperatureControl Error: None Enable Comm: Yes

	IVAL	XVAL	Lower Clamp	Upper Clamp	Name
Input:	0	84.02603			Room_Temperature
Output:	0	4.372345	0	10	Heater
Setpoint:	84.0032	84.0032	70	90	Host
Gain:	300	300			
Integral:	1	1			
Derivative:	0	0			
Scan Rate (sec):	2	2			
Max Chg Rate:	10	10	(0.100 to 10.000)		
Mode:	Automatic	Automatic			

A—*Input, Output, and Setpoint (current internal and external values).* Change each to any valid value. If lower and upper clamps appear to the right of the value, these clamps define the range of valid values. Otherwise, the valid range is defined by the I/O point itself. If you can change them, click Apply.

B—*Gain, Integral, and Derivative (current internal and external values).* You can change the gain term to any value except zero in the range -32768 to 32767. You can change the integral and derivative terms to any value in the range zero to 32767. If you change them, click Apply.

C—*Scan Rate and Maximum Change Rates (current internal and external values).* You can change the scan rate to any value in the range 0.1 to 6553.5 seconds. You can change the maximum change rate to a value between one percent and 100 percent of the output range (defined by the output's zero-scale and full-scale values). If you change them, click Apply.

Asterisks in an IVAL field indicate that a valid scan rate or maximum change rate hasn't been read yet (usually before the strategy is run).

D—*PID execution mode.* A green background means Automatic, a yellow background means Manual. Click an arrow to change the mode; then click Apply.

E—*Setpoint and Input plot.* Adjust resolution using the Input Axis button at right. Click and drag on the scale to move the line.

F—*Output plot.* Adjust resolution using the Output Axis button. Click and drag on the scale to move the line.

G—*Time axis.* Adjust resolution using the Time Axis button. Click and drag left or right to see other times.

H—*Current status.* Yes on a green background means enabled; No on a red background means disabled. To change status, click one of the arrows; then click Apply.

Using Watch Windows for Monitoring

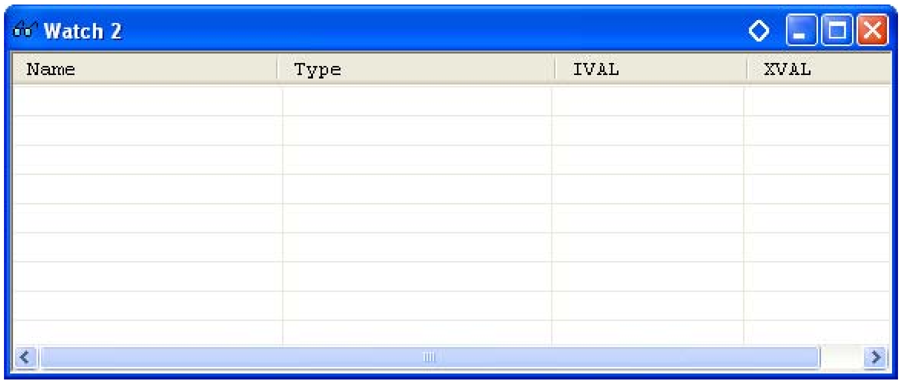
While the strategy is running, you can monitor several strategy elements at once in a watch window: I/O units, digital and analog points, PID loops, variables, even charts. You cannot monitor subroutine parameters or variables that are local to a subroutine in a watch window.

Unlike inspection windows, watch windows can be created the way you want, docked in a position most convenient for you, and are saved with your strategy. You cannot change strategy elements in a watch window, but you can open the inspect dialog box from the watch window and change the element there.

Creating a Watch Window

1. With the strategy open and in Debug mode, choose Watch > New.
2. In the Create New Watch Window dialog box, navigate to the location where you want the watch window file to be kept (usually in the same folder as the strategy). Enter the watch window file name and click Open.

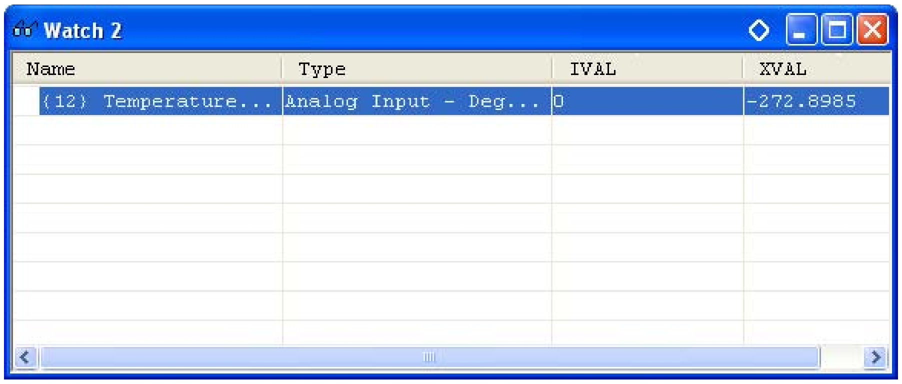
   The empty watch window appears.



3. Add elements you want to watch in this window by clicking them on the Strategy Tree and dragging them into place in the watch window, or by right-clicking them and choosing Watch from the pop-up menu.

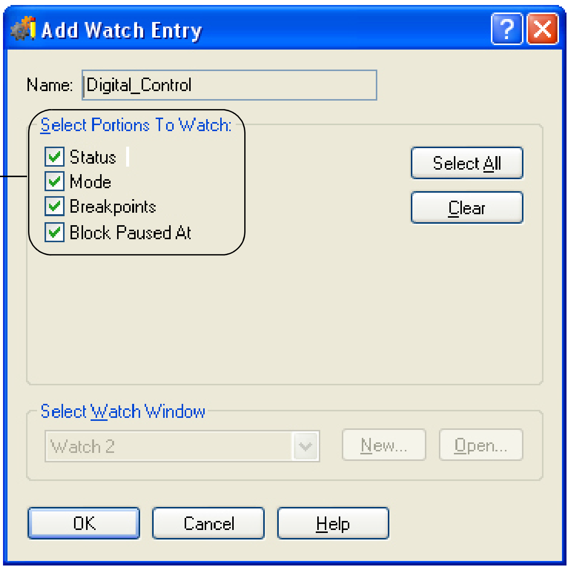
   You can add I/O units, digital and analog points, PID loops, variables, and charts. You cannot add subroutine parameters or variables that are local to a subroutine.

Depending on which element you add and how you add it, it may appear immediately in the window, as shown here.



Name	Type	IVAL	XVAL
(12) Temperature...	Analog Input - Deg...	0	-272.8985

For some elements, the Add Watch Entry dialog box appears, so you can specify what to watch.



Name: Digital_Control

Select Portions To Watch:

- [✓] Status
- [✓] Mode
- [✓] Breakpoints
- [✓] Block Paused At

Select Watch Window: Watch 2

Items in this area vary depending on the element you are watching. This example shows a chart.

4. If an Add Watch Entry dialog box appears, click to place or remove the check mark next to any item. When all the items you want to watch are checked, click OK.

The element is added to the watch window.

The watch window is automatically saved.

Opening an Existing Watch Window

If a watch window was open when you exited Debug mode, it will automatically open again when you re-enter Debug mode. To open other watch windows, follow these steps.

1. Make sure the strategy is open and in Debug mode.

2. Choose Watch > Open.

3. Navigate to the watch window you want to open and double-click its name.

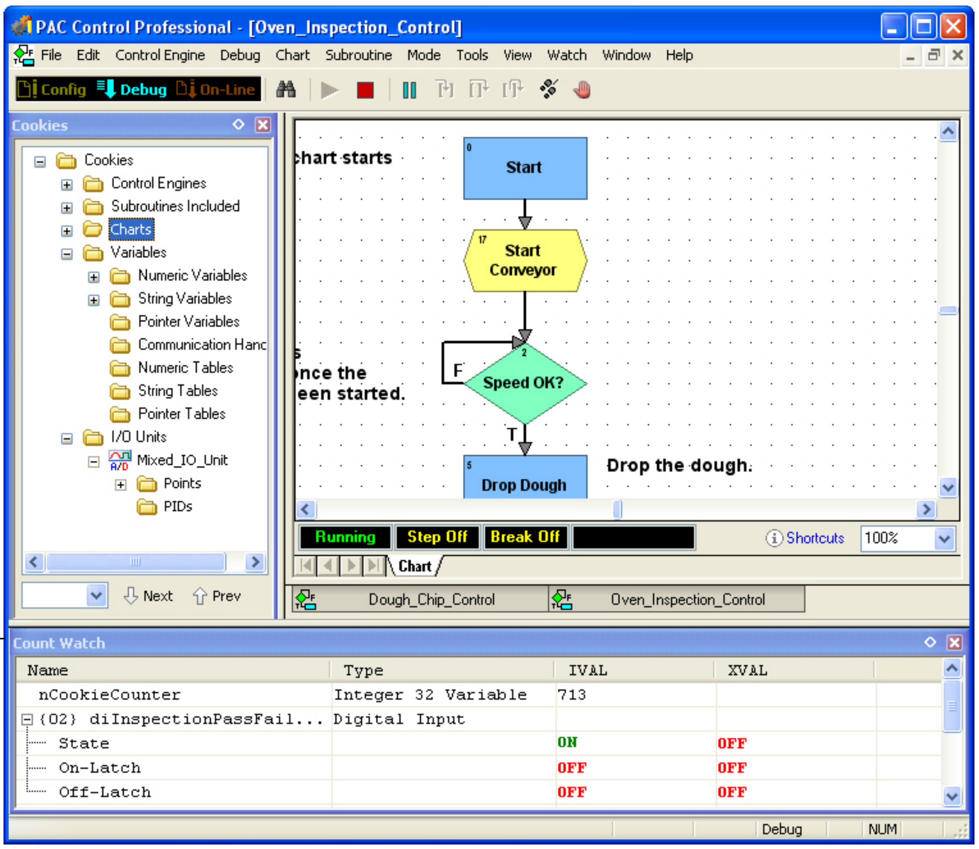
The window opens in the position you left it.

Working in Watch Windows

Watch windows are flexible. You can dock the window where you want it in the PAC Control main window. You can also move, delete, and inspect elements in the window.

- **To dock the watch window**, click the docking button in its title bar.

  The window moves to its own frame.

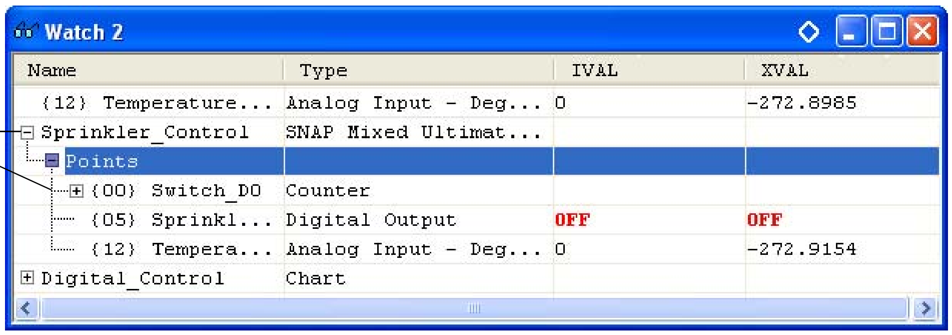


Name	Type	IVAL	XVAL
nCookieCounter	Integer 32 Variable	713	
{02} diInspectionPassFail...	Digital Input		
State		ON	OFF
On-Latch		OFF	OFF
Off-Latch		OFF	OFF

"Docked" watch window

  See "Docking Windows" on page 60 for more information.

- **To expand or collapse watch window** you have added to an I/O unit, PID loop, chart, or table, click its plus or minus sign.



Name	Type	IVAL	XVAL
{12} Temperature...	Analog Input - Deg...	0	-272.8985
⊟ Sprinkler_Control	SNAP Mixed Ultimat...		
⊟ Points			
⊞ {00} Switch_DO	Counter		
{05} Sprinkl...	Digital Output	OFF	OFF
{12} Tempera...	Analog Input - Deg...	0	-272.9154
⊞ Digital_Control	Chart		

Expand or collapse the item by clicking the + or - sign in the box.

- **To rearrange elements** in the watch window list, click the item you want to move and drag it to a new location, or right-click it and choose Move Up or Move Down from the pop-up menu.

You can also sort elements in the window by clicking on the column label. For example, to sort by Type, click the label Type in the column heading. Click again to change the order from ascending (A–Z) to descending (Z–A).

- **To move an element** from one watch window to another, open both windows and drag the element where you want it. To copy an element to another watch window (so it will appear in both windows), hold down the Ctrl key while you drag it.
- **To delete an element**, right-click it and choose Delete from the pop-up menu.
- **To inspect an element**, double-click it.

  The inspect dialog box opens. For information on using it, see "Inspecting Control Engines and the Queue" on page 111.

7: Working with Strategies

Introduction

A strategy is the software program you create in PAC Control. A strategy is similar to a file in any Microsoft Windows program. You use standard Windows menu items to create a new strategy, to open an existing strategy, or to save a strategy. The strategy includes all the definitions and instructions necessary to control your process. This chapter is a step-by-step reference for working with strategies in all three strategy modes: Configure, Debug, and Online.

In this Chapter

Creating a New Strategy 197
Opening a Strategy 198
Saving and Closing 199
Saving a Strategy to Flash 200
Archiving Strategies on the Computer 201
Compiling and Downloading 202
Running a Strategy Manually 210
Debugging 211
Viewing and Printing 219
Searching and Replacing 228
Legacy Options 232
Advanced Options 235

Creating a New Strategy

Each PAC Control strategy must be located in its own directory. When you create a new strategy, you must create a new directory or use an empty one. Each strategy in its own directory keeps all its files in one place and makes it easy to copy a strategy to another location for modification or backup.

1. To create a new strategy, select File > New Strategy, or press Ctrl+N, or click the New Strategy icon on the toolbar.
2. In the New Strategy dialog box, navigate to the directory where you want the strategy to be placed. Create a new folder if necessary.
3. Type the strategy name.

As you can see in the Files of type field, PAC Control files have an extension of .idb.

4. Click Open.

   The new strategy is created. Its Strategy Tree and Powerup charts appear in the PAC Control main window. For information on the main window, see "PAC Control Main Window" on page 54. For programming information, see "4: Designing Your Strategy." For steps to create charts, see "8: Working with Flowcharts"

Opening a Strategy

Only one strategy at a time can be open in PAC Control. If you currently have a strategy open, it must be closed before another is opened. You are prompted to save changes before it closes.

Opening an Existing Strategy

1. To open an existing strategy, select File > Open Strategy, or press Ctrl+O, or click the Open Strategy icon on the toolbar.
2. In the Open Strategy dialog box, navigate to the strategy you want to open and click Open.

   The strategy opens in Configure mode, with the windows in the same position they were when the strategy was closed.

Opening an ioControl Strategy

If you have been using ioControl Basic or Professional with a SNAP PAC controller, migration to PAC Project Basic or Professional is simple. Because the individual programs within the suite are essentially the same (although they have new features and hardware support), you can simply open a strategy in PAC Control and then save it. For safety, we recommend you back up all ioProject files before opening them in PAC Project. See also, "Opening Strategies in PAC Control" below. For more information on migrating your system to SNAP PAC, see form 1680, the *SNAP PAC System Migration Technical Note*.

Opening a Recently Used Strategy

To open a strategy you have recently used, choose its name from the list at the bottom of the File menu. The ten most recently opened strategies are listed.

Loading a Strategy or Mode at Startup

To have PAC Control automatically start up with the strategy that was open when you exited, choose Configure > Options and click to put a check mark next to Load Last Strategy at Startup.

To have PAC Control open strategies in the same mode as when you exited PAC Control, choose Configure > Options and click to put a check mark next to Load Last Mode at Startup.

Opening Strategies in PAC Control

A strategy saved in ioControl version 6.1 or less can be opened in either PAC Control Basic or Professional. A PAC Control Basic strategy can also be opened in PAC Control Professional. However, PAC Control Professional strategies cannot be opened in Basic or in any earlier version of PAC Control.

***CAUTION**: Once a strategy is opened in PAC Control Professional, it can no longer be opened in PAC Control Basic.*

Opening an OptoControl Strategy

If you are moving a strategy from OptoControl to PAC Control Professional, PAC Control will open it and help you convert it. Although many things will convert without difficulty, planning ahead is essential to make the job easier. **Before opening an OptoControl strategy in PAC Control,** read form 1692, the *FactoryFloor to PAC Project Migration Technical Note*.

Saving and Closing

***CAUTION**: Once a strategy is opened in PAC Control Professional, it can no longer be opened in PAC Control Basic.*

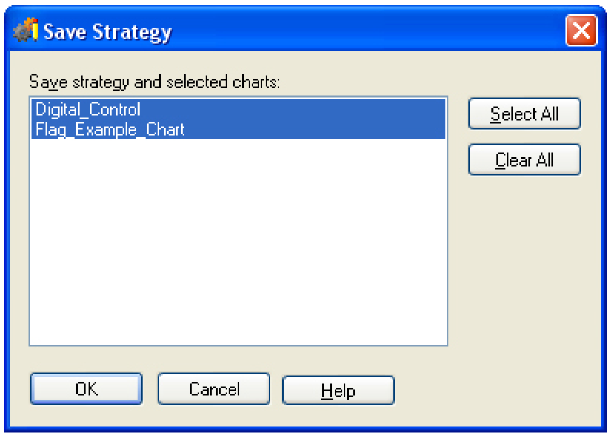
Saving the Strategy and All Charts

To save all your work quickly, choose File > Save All. The strategy and all modified charts and subroutines are saved.

Saving the Strategy and Some Charts

NOTE: You cannot save changes to a subroutine this way. To save a subroutine, use File > Save All, or use Subroutine > Save or Subroutine > Save All.

1. To save changes to some charts but not others, click the Save Strategy icon on the toolbar (or choose File > Save Strategy, or press Ctrl+S).
   The Save Strategy dialog box appears, highlighting all charts modified since the last save.



Digital_Control
Flag_Example_Chart

2. To save some charts and not others, press Ctrl and then click any charts you don't want to save.

   You can also click Select All to select all of the charts, or click Clear All to select none of them.
3. When only the charts you want to save are highlighted, click OK.

   The strategy and the highlighted charts are saved.

Saving the Strategy to a New Name

1. To save the strategy and all its charts under a new name, choose File > Save Strategy As.
2. In the Save Strategy As dialog box, navigate to where you want the new strategy to be. Create a new folder if necessary.

   Remember that each strategy must be in its own directory.
3. In the Strategy Name field, enter the new strategy name. Click Save.

   The strategy and all its charts are saved under the new name in the new directory.

Saving Before Debugging

When you change to Debug mode, you are prompted to save a strategy you have modified. If you don't want to be prompted to save before entering Debug mode, choose Configure > Options and click to remove the check box for Prompt To Save Strategy Before Running Debugger.

Closing a Strategy

To close a strategy, click the close box in the Strategy Tree or choose File > Close Strategy.

NOTE: Since only one strategy at a time can be open in PAC Control, creating a new strategy or opening an existing strategy automatically closes any current strategy first. If you've made changes to the current strategy, you are prompted to save them.

Saving a Strategy to Flash

When you finish working on your strategy and have downloaded it, you should save it to the control engine's flash memory. By default, a strategy is downloaded to the control engine's RAM. Saving it to flash protects the strategy in case of a power loss. You can save it to flash just once, when needed, or save every time the strategy is downloaded.

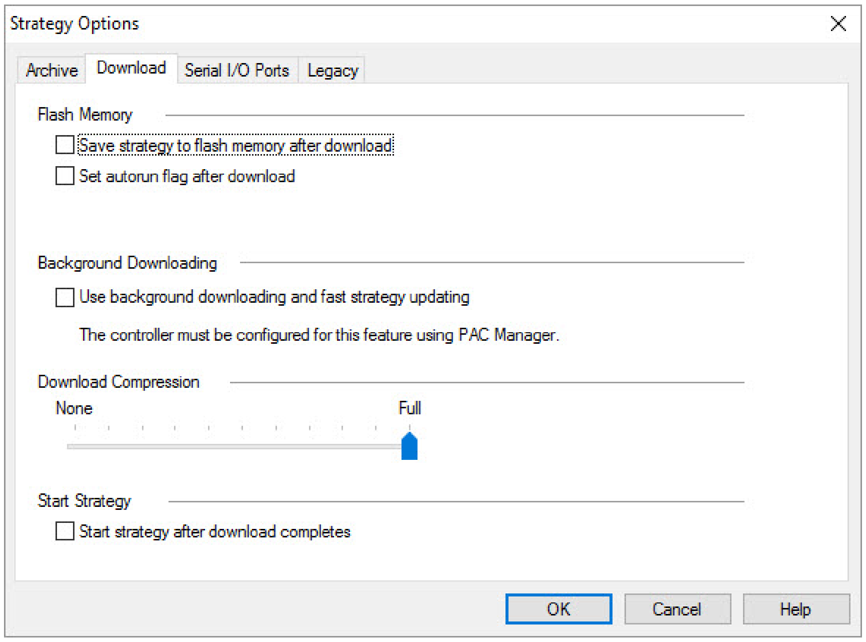
Saving to Flash Once

You can save the strategy to flash at any time when you are in Debug mode. To do so, choose Control Engine > Save Strategy to Flash.

Saving to Flash on Every Download

***CAUTION**: It is possible to wear out flash memory if you save to it many, many times. Use the following steps only when your strategy is finished.*

1. When you have finished the strategy and are in Configure mode, choose File > Strategy Options.
2. In the Strategy Options dialog box, click the Download tab. Check Save strategy to flash memory after download.



Flash Memory
- [ ] Save strategy to flash memory after download
- [ ] Set autorun flag after download

Background Downloading
- [ ] Use background downloading and fast strategy updating

The controller must be configured for this feature using PAC Manager.

Download Compression
None — Full

Start Strategy
- [ ] Start strategy after download completes

If a control engine loses power and then restarts, the autorun flag tells the control engine to automatically start running the strategy that is in flash memory. If the autorun flag is not set, the strategy must be started manually after power is restored to the control engine.

3. To have the strategy run automatically after a control engine restarts, check Set autorun flag after download.
4. Click OK.

Archiving Strategies on the Computer

Archiving strategies to the computer helps you track changes during development and provides a backup in case of a failure on the control engine. Archive files are date and time stamped, and zipped for compact storage that you can copy to another computer or disk for backup. Archives are always placed in the same folder as your strategy. Since a new archive file is created each time you archive a strategy, remember to manually delete any old archive files you do not want to keep.

We recommend you archive both to the computer (during strategy development) and to the control engine (when the strategy is completed). Archiving to the control engine as well as the computer makes sure that an older strategy can always be found and updated, even after personnel changes occur and years pass. See also, "Archiving Strategies on the Control Engine" on page 116.

PAC Control offers three ways to archive strategies to the computer:

- To make an archive at any time, choose File > Archive Strategy. A dialog box shows you the name and location of the archive file.
- To have an archive automatically created whenever the strategy is closed, choose File > Strategy Options. In the Strategy Options dialog box, click Archive strategy to disk when strategy is closed.
- To have an archive automatically created whenever the strategy is downloaded, choose File > Strategy Options. In the Strategy Options dialog box, click Archive strategy to disk when strategy is downloaded.

The archive file name will be in one of the following formats:

Archive method	File name format
Manual archive or archive when strategy is closed	Path\Filename.Archive.D02282007.T114351.zip
Archive on download	Path\Filename.Download.D02282007.T114351.zip
Archive when downloading from online mode	Path\Filename.Online.D02282007.T114351.zip

The date stamp (D) is in the format mm/dd/yyyy. In the examples above, the date is February 28, 2007. The time stamp (T) is in the format hh/mm/ss. In the examples above, the time is 51 seconds past 11:43 A.M.

Compiling and Downloading

TIP: To configure a strategy to start running at completion of the strategy download, select the "Start strategy after download completes" check box on the File > Strategy Options > Download tab.

Before your strategy can be tested or run, it must be compiled and then downloaded to a control engine. When a strategy is compiled, all the commands, OptoScript code, charts, and variable and I/O definitions it contains are verified and converted into a format that the control engine can understand. Then the strategy can be sent (downloaded) to a control engine. Only compiled strategies can be downloaded.

NOTE: Before you can download your strategy, make sure you have downloaded the latest firmware to your control engine. For instructions, see the controller's user's guide.

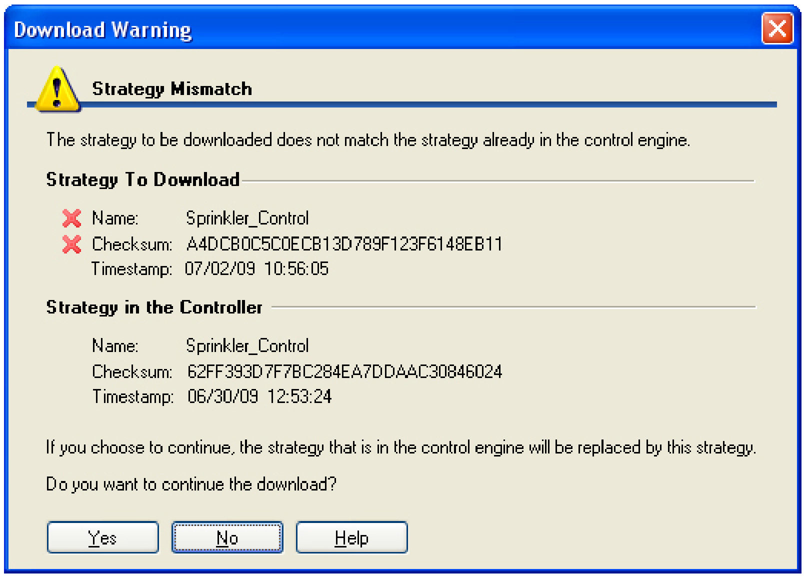
Compiling and Downloading in One Step

NOTE: If you are using Ethernet link redundancy in PAC Control Professional, make sure you are downloading through the IP address you want to use, either the primary or secondary address. See "Using Ethernet Link Redundancy in PAC Control" on page 104 for more information.

1. With the strategy open in PAC Control, click the Debug mode icon Debug on the toolbar, or choose Mode > Debug.

   Changing to Debug mode automatically saves and compiles the strategy, including all code in OptoScript blocks.
2. If you see a Powerup Clear Expected message, click OK.

A download warning message may appear.



Strategy Mismatch

The strategy to be downloaded does not match the strategy already in the control engine.

Strategy To Download

❌ Name:	Sprinkler_Control
❌ Checksum:	A4DCB0C5C0ECB13D789F123F6148EB11
Timestamp:	07/02/09 10:56:05

Strategy in the Controller

Name:	Sprinkler_Control
Checksum:	62FF393D7F7BC284EA7DDAAC30846024
Timestamp:	06/30/09 12:53:24

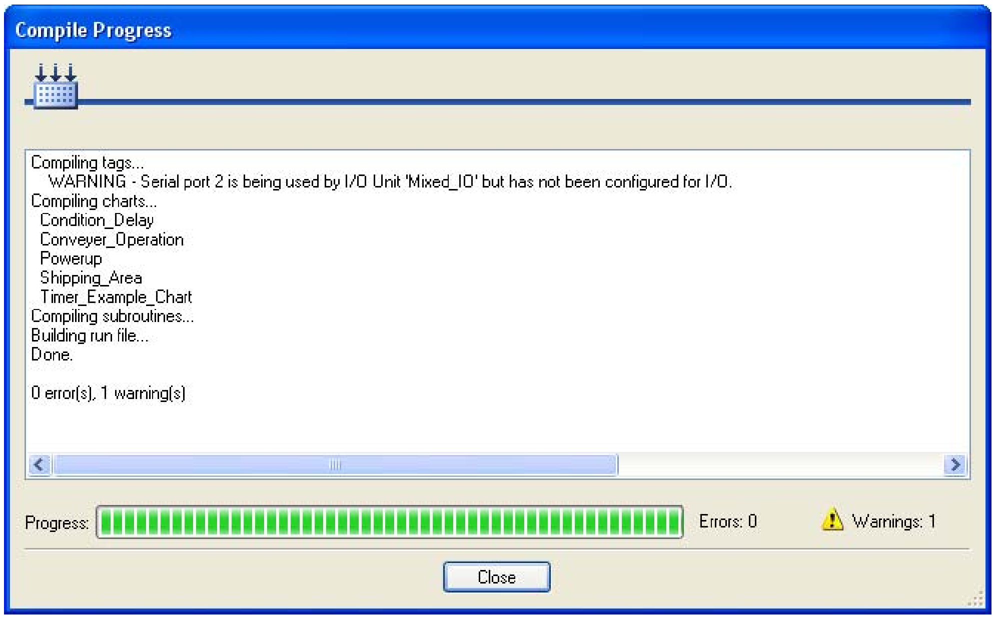
If you choose to continue, the strategy that is in the control engine will be replaced by this strategy.

Do you want to continue the download?

This message tells you that the strategy to be downloaded doesn't match the strategy already loaded on the control engine, either because it is a different strategy or because it has been changed since the last download.

3. To continue the download, click Yes.

As the strategy is compiled, the Compile Progress dialog box appears. This dialog may appear only briefly. However, if there are errors or warnings, the dialog box will stay open. You can copy the errors and warnings by clicking the right mouse button and selecting Copy All as Text.



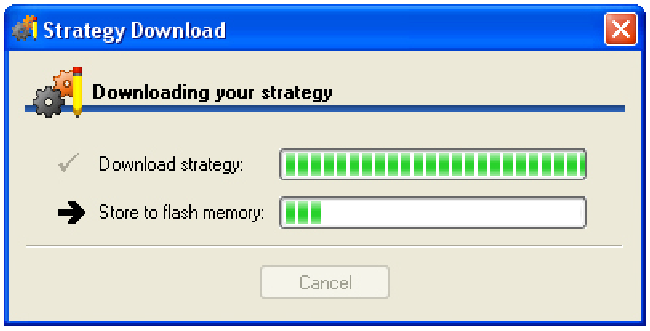
```
Compiling tags...
   WARNING - Serial port 2 is being used by I/O Unit 'Mixed_IO' but has not been configured for I/O.
Compiling charts...
  Condition_Delay
  Conveyer_Operation
  Powerup
  Shipping_Area
  Timer_Example_Chart
Compiling subroutines...
Building run file...
Done.

0 error(s), 1 warning(s)
```

Progress: Errors: 0 Warnings: 1

If there are *errors*, the strategy will not compile. Click Close to continue in Configure mode.

If there are only *warnings*, click Close to continue in Debug mode.

If no errors or warnings occur, the Strategy Download dialog box appears.



The Strategy Download dialog box will vary somewhat depending on your settings.

NOTE: If you are using background downloading (see "Background Downloading" on page 206), while the strategy is downloaded, there may be a very slight degradation in performance. This is due to the overhead on the controller to process the strategy download.

4. (Optional) If there is already an active strategy which is set up to have an alternate strategy, and you want to switch strategies, click Switch & Run to stop the active strategy and run the strategy you just downloaded. Or, you can click Switch to stop the active strategy and make the new strategy the active one (but not yet running).

Cancel aborts the new download. The controller is left as it was before the download occurs.

When the download is complete, you are in Debug mode. (If you receive errors, see "A: Troubleshooting.")

After switching strategies, you can switch back to the original strategy by clicking Control Engine > Inspect in Debug mode and use the Switch and Run or Switch button in the Inspect Control Engine dialog box, or you can use PAC Terminal. When the strategy is switched back, the persistent variables are created and compared.

Switching the Strategy

If there is already an active strategy which is set up to have an alternate strategy (as described in "Background Downloading" on page 206), and you want to switch strategies, do the following:

1. In Debug mode, choose Control Engine > Inspect to open the Inspect Control Engine dialog box.

2. Choose either the Switch and Run or the Switch button.

The Switch & Run button stops the active strategy and runs the strategy you just downloaded. The Switch button stops the active strategy and makes the new strategy the active one (but not yet running).

When the strategy is switched (as described in the previous steps), the following operations occur:

- Persistent variables are compared. If the type and size of the persistent variable are consistent, the contents are copied. If the type and size do not match, a new persistent variable is created. New persistent variables are initialized to zero and new persistent strings are initialized to NULL (empty) strings.

- I/O unit configurations are compared. If the I/O unit configurations match, I/O will not be initialized.
- The Powerup chart is called.

See also, "Background Downloading" on page 206.

Compiling without Downloading

Sometimes you may want to compile without downloading, just to see if a chart, subroutine, or strategy compiles correctly. You can compile the active chart or subroutine only, just the changes you have made to the strategy, or the entire strategy.

Compiling the Active Chart or Subroutine

Whenever a chart or subroutine window is open and active, you can compile just that chart or subroutine. To do so, in Configure mode, click the Compile Active View icon on the toolbar, or choose Compile > Compile the Active Chart. The menu option shows the name of the chart or subroutine you are compiling.

As soon as you choose the menu option, the chart or subroutine is saved and compiled. You are alerted only if errors are found.

Compiling Changes Only

To compile just the changes since the strategy was last compiled, choose Compile > Compile Changes. The menu option shows the name of the strategy you are compiling.

As soon as you choose the menu option, the strategy and all modified charts and subroutines are saved and compiled. You are alerted only if errors are found.

Compiling the Entire Strategy

To compile the entire strategy including all charts and subroutines, in Configure mode, click the Compile All icon on the toolbar, or choose Compile > Compile All. The menu option shows the name of the strategy you are compiling.

As soon as you choose the menu option, the entire strategy is saved and compiled. The Compile Progress dialog box appears. You are alerted if errors are found.

Downloading Only

If your strategy has been compiled, you can download it again quickly. Downloading again is useful if you want to run your strategy from a *clean slate* by reinitializing any variables that are set only on a strategy download.

To download a strategy that has already been compiled, you must be in Debug mode. Choose Control Engine > Download Strategy.

The Download to dialog box appears and your strategy is downloaded.

TIP: To configure a strategy to start running at completion of the strategy download, select the "Start strategy after download completes" check box on the File > Strategy Options > Download tab.

Background Downloading

The Background Downloading feature allows you to download an *alternate* strategy to a controller while the *active* strategy currently on the controller continues to run. With two different strategies downloaded to the controller you can switch rapidly from one strategy to the other.

This allows you to:

- Download a new or updated strategy to the controller and then optionally run it with minimal downtime. For larger strategies, downtime is dramatically reduced.
- Download and test a new or updated strategy but still keep the original strategy available. You can bring back the original very quickly, either by manually switching from one strategy to the other, or—if the original is still burned to flash—by cycling power.

You can use background downloading as an alternative to Online mode, which also allows you to make changes while a strategy is running. The following table describes the differences between Online Mode and Background Downloading:

	Online Mode	**Background Downloading**
Can you add variables, I/O, and charts?	No	Yes
Can you add and change blocks, change instructions, and edit existing charts?	Yes	Yes
What is downloaded in Debug mode?	Only the edited charts are downloaded. All other charts keep running.	The entire alternate strategy is downloaded while the entire active strategy continues to run.
Where does the strategy restart after download?	The affected charts restart at block 0.	If you switch to the alternate strategy after download, the strategy restarts with the Powerup chart.
How is the controller memory affected?	When a chart is downloaded, the old one is not deleted, so this consumes additional memory. To clear memory, the strategy should be stopped and then compiled and downloaded.	Controller memory is cut in half when background download is set it in PAC Manager. Also, one chart is used for the alternate download port, so there is one less chart that can run.

To enable Background Downloading:

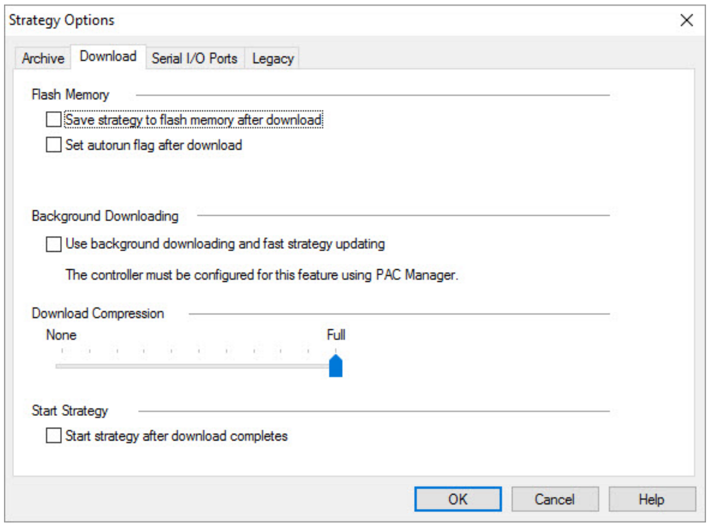
NOTE: Controller memory is cut in half when background downloading is set in PAC Manager. Also, one chart is used for the alternate download port, so there is one fewer chart that can run.

1. Open PAC Manager and do the following steps:
   a. Click the Inspect icon to open the Inspect window.
   b. In the Inspect Window's IP Address field, type the IP address for the SNAP PAC controller.
   c. Click Status Write, then scroll down to Strategy Download Method.
   d. Click on the Value field for Strategy Download Method, select Background from the drop-down menu, and then click Apply.
   e. Under Operation Commands, highlight "Store configuration to flash," and click Send Command.

      The configuration is stored to flash memory and a Success message appears.

   For more information, see Opto 22 form 1704, the *PAC Manager User's Guide*.

2. Turn off the power to the control engine, and then turn it back on again.
3. In Configure mode in PAC Control, choose File > Strategy Options.
4. In the Download tab under Background Downloading, select “Use background downloading and fast strategy updating.”
5. (Optional) Under Flash Memory, select “Save strategy to flash memory after download.”
6. (Optional) To have the strategy run automatically after a control engine restarts, select “Set autorun flag after download.”


Strategy Options

Archive | Download | Serial I/O Ports | Legacy

Flash Memory
- ☐ Save strategy to flash memory after download
- ☐ Set autorun flag after download

Background Downloading
- ☐ Use background downloading and fast strategy updating

The controller must be configured for this feature using PAC Manager.

Download Compression
None — Full

Start Strategy
- ☐ Start strategy after download completes

7. Click OK.

PAC Control is now set to download the strategy in the background when you click the Debug mode icon Debug. Next, see “Compiling and Downloading” on page 202. See also, “Switching the Strategy” on page 204.

Downloading Without Using PAC Control

If you are creating strategies for users who do not have PAC Control on their systems (for example, if you are an integrator or OEM), you can make a control engine download file that can be downloaded to a SNAP PAC controller by using either the graphical interface in PAC Terminal or the command line interface with a batch file. This one download file is built for a specific control engine but can also be downloaded to other similar control engines. It contains everything PAC Control would download, including .per, .inc, and .crn files, control engine-specific files, and initialization information.

In most cases you will want the downloaded strategy to be saved to flash memory and to start automatically (autorun) when power is cycled to the control engine. Before you create the download file, follow the steps in “Saving to Flash on Every Download” on page 201. Check the boxes to have the strategy saved to flash memory after download and to set the autorun flag after download. This information will become part of the download file.

Creating the Control Engine Download (.cdf) File

With the strategy open in PAC Control in Configure mode, right-click the name of the control engine in the Strategy Tree and choose Compile Control Engine Download File from the pop-up menu. (You can also choose Compile > Compile Control Engine Download File.)

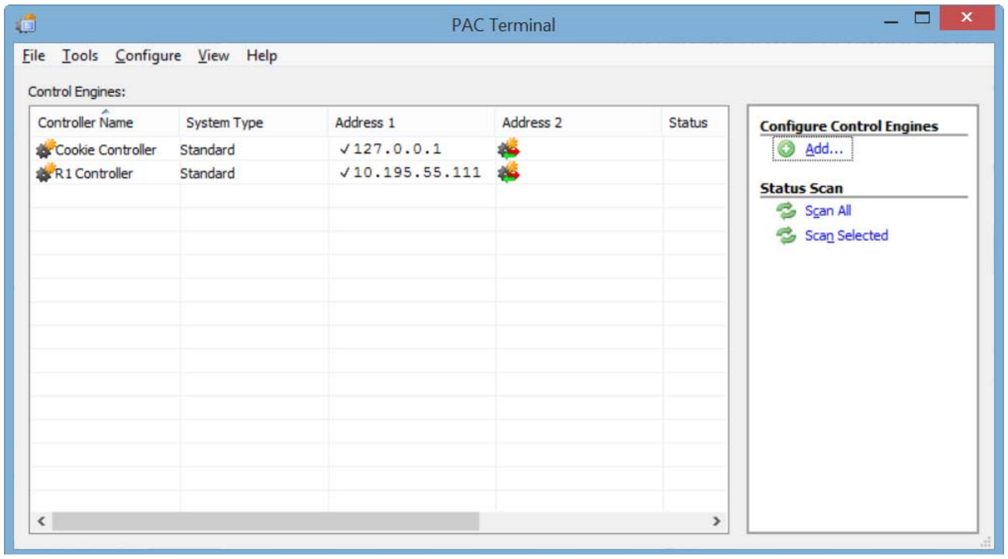
The file is created in the same folder as the strategy, with a .cdf extension and a file name consisting of the strategy's name and the control engine's name (for example, MyStrategy.MyEngine.cdf).

Once the control engine download file is created, it can be downloaded using either the graphical interface in PAC Terminal or the command line interface with a batch file you create.

Downloading the .cdf File using PAC Terminal

1. Start PAC Terminal as follows:
   - In Windows 7 and Windows Vista, press the Windows Start key, and then click Programs > Opto 22 > PAC Project 9.6 > Tools > PAC Terminal.
   - In Windows 10 and Windows 8.1, press the Windows Start key, type **`PAC Terminal 9.6`** and then press the Enter key.

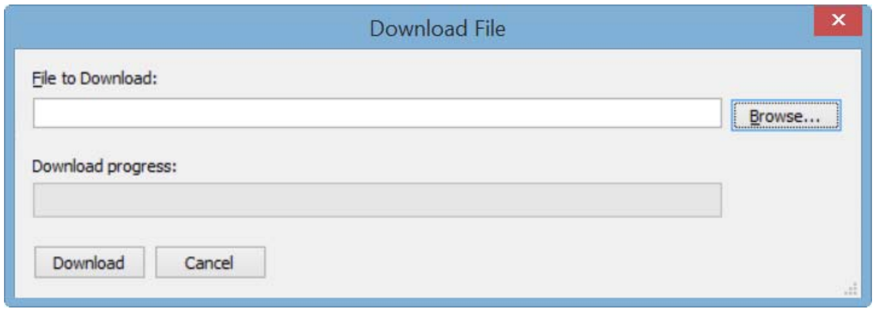
   The PAC Terminal window appears.



Control Engines:

Controller Name	System Type	Address 1	Address 2	Status
Cookie Controller	Standard	✓127.0.0.1		
R1 Controller	Standard	✓10.195.55.111		

2. Right-click the name of the control engine you want to download the file to.
3. In the pop-up menu, choose Download > Control Engine Download File.



4. Enter the path and file name of the .cdf file, or click the Browse button and navigate to it. When the file name appears in the File to Download field, click OK.

   The file is downloaded to the control engine.

Using Command Lines or a DOS Batch File to Perform PAC Terminal Tasks

If you do not want your end user to have to use PAC Terminal, you can use DOS-level commands or create a DOS batch file to download a .cdf file. In addition to downloading a .cdf file, the DOS commands can also run or stop a strategy, or define the control engine on the PC that will download the file. PAC Terminal must be installed on the PC where the batch file is used.

The following table lists actions you may want to take within PAC Terminal:

To do this	Use this
Show help information for termcl	`-h` control_engine_name
Download the specified file to the specified controller. Compression can be between 0 (none) and 10 (max).	`-d` control_engine_name filename [-z compression]
Run the strategy in the specified control engine	`-r` control_engine_name
Stop the strategy in the specified control engine	`-s` control_engine_name
Send a Forth command to the control engine	`-c` control_engine_name Forth_command
Add a control engine definition to the system. Does not modify an existing definition. Example: -a MyCE tcp 10.20.30.40 22001 0 2000	`-a` control_engine_name tcp ip-addr port retries timeout_ms
Download the encrypted SSD file to the controller	`-dp` control_engine_name ssd_filename
Prevent response messages from printing to the DOS screen	Append the DOS *switch* `-q`

This example shows lines included in a batch file that will define the control engine, download the .cdf file to it, and then run the strategy:

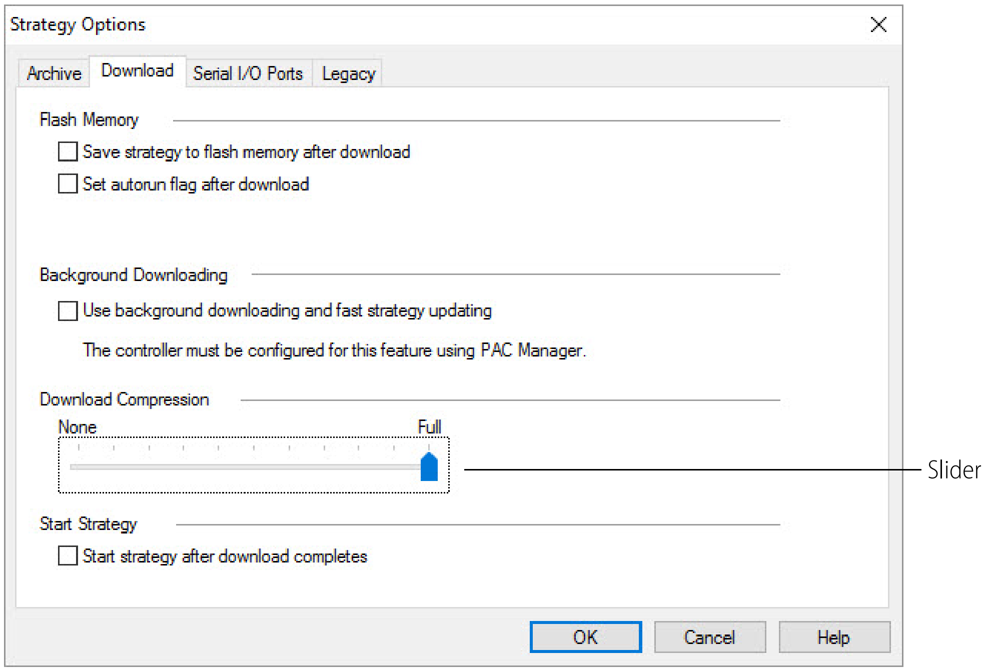
```
termcl -a MyCE tcp 10.20.30.40 22001 0 2000
termcl -d MyCE "c:\My_Project\MyStrategy.MyCE.CDF"
termcl -r MyCE
```

Changing Download Compression

If you have a very large strategy, short timeouts, and slow connections on your network, you may need to decrease the compression level to download the strategy successfully. When you decrease compression, the strategy takes longer to download because it is sent in smaller chunks. If you are having difficulty downloading your strategy, follow these steps to decrease compression:

1. With the strategy open in Configure mode, choose File > Strategy Options.

2. In the Strategy Options dialog box, click the Download tab.



Flash Memory
- [ ] Save strategy to flash memory after download
- [ ] Set autorun flag after download

Background Downloading
- [ ] Use background downloading and fast strategy updating

The controller must be configured for this feature using PAC Manager.

Download Compression
None — Full

Start Strategy
- [ ] Start strategy after download completes

3. Move the slider to the left to reduce compression, and then click OK.
   You may need to experiment with the setting until the strategy downloads successfully.

Running a Strategy Manually

1. With the strategy open, choose Mode > Debug.
2. Click the Run Strategy icon ▶ (or press F5, or select Debug > Run).

   You can also run a strategy from the Inspecting dialog box. See "Inspecting Control Engines in Debug Mode" on page 111.

Running a Strategy Automatically (Autorun)

You can set the strategy to run automatically (autorun) if the control engine loses power and then restarts. In traditional Opto 22 controllers, the OptoControl autorun function was controlled by a jumper. For PAC Control, it's controlled by the autorun flag. If the autorun flag is not set, the strategy must be started manually after power is restored to the control engine. The strategy must be saved in flash memory for autorun to work.

You can set the autorun flag in two ways:

- In Configure mode, save the strategy to flash memory and set the autorun flag every time the strategy is downloaded. (Be careful you do not save to flash too often, as flash memory eventually wears out.) See "Saving to Flash on Every Download" on page 201.

- In Debug mode, save the strategy to flash memory by choosing Control Engine > Save Strategy to Flash. Then right-click the control engine in the Strategy Tree and choose Inspect from the pop-up menu. In the Inspect dialog box, enable Autorun. (See "Inspecting Control Engines and the Queue" on page 111 for more on the Inspect dialog box.)

Protecting a Running Strategy

If you want a strategy to run automatically without interruption, you can protect the strategy by disabling all host communications to the control engine.

To protect the strategy, first make sure the strategy is saved to flash and that the autorun flag is set. To disable host communication, open PAC Manager and set the Control Engine port to 0, and then save that change to flash. For more information on setting this port, see Opto 22 form 1704, the *PAC Manager User's Guide*.

NOTE: This action also disables communication between PAC Display and the control engine.

Stopping a Strategy

To stop the strategy, click the Stop Strategy icon ■ (or press F3, or select Debug > Stop). You can also stop a strategy from the Inspecting dialog box; see "Inspecting Control Engines in Debug Mode" on page 111.

Debugging

Once the strategy is running, if it doesn't appear to be working correctly, you can use several tools in Debug mode to figure out what the problem is. You can pause a chart or subroutine; step into, over, or out of each block; watch it slowly step through the blocks; or add a breakpoint to a block to stop the strategy just before it executes that block.

The chart's or subroutine's status is shown in the lower left-hand corner of its window. This corner shows whether the chart or subroutine is running, stopped, or suspended, and whether the debugging tools, such as stepping and breakpoints, are in effect. The chart or subroutine must be running in order to use these tools.

Choosing Debug Level

You can choose one of two levels of debugging:

- **Minimal Debug** lets you step from block to block, but does not allow you to step into blocks. Less information is downloaded to the control engine for minimal debugging, so downloading the strategy takes less time and less control engine memory. The strategy also runs slightly faster.
- **Full Debug** lets you step into blocks, so you can step through each instruction in an Action or condition block and through each line of OptoScript code in an OptoScript block. If you are using OptoScript, you will probably want to spend the additional time to download your strategy at the full debug level.

To change debug level, make sure you are in Configure mode. From the Configure menu, choose Minimal Debug or Full Debug. The next time you enter Debug mode, the strategy will be compiled and downloaded with the new level.

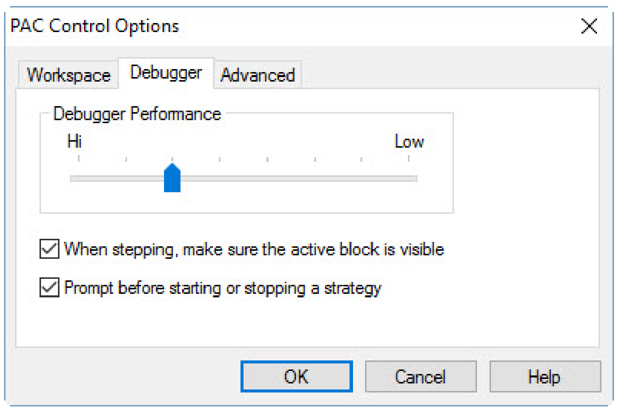
Changing Debugger Speed

Before you enter Debug mode, you may want to consider changing debugger speed. Depending on the number of charts and windows open in PAC Control, and depending on other processing your computer is doing at the same time, you may find that running the debugger affects the computer's or the control engine's performance of other tasks. If necessary, you can slow down the debugger by increasing the time delay between debugging calls to the control engine, therefore leaving more processing time for other tasks.

In addition, a slower setting may be useful when checking communication using PAC Message Viewer (see page 418)).

To change debugger speed, follow these steps:

1. With the strategy in Configure mode, choose Configure > Options. Alternatively, in Debug mode, choose Debug > Options.
2. In the PAC Control Options dialog box, click the Debugger tab.



☑ When stepping, make sure the active block is visible

☑ Prompt before starting or stopping a strategy

3. Click and drag the slider to the speed you want.

   The default speed is shown in the figure above. Since performance varies depending on your hardware and software, you may need to experiment to find the most efficient speed.

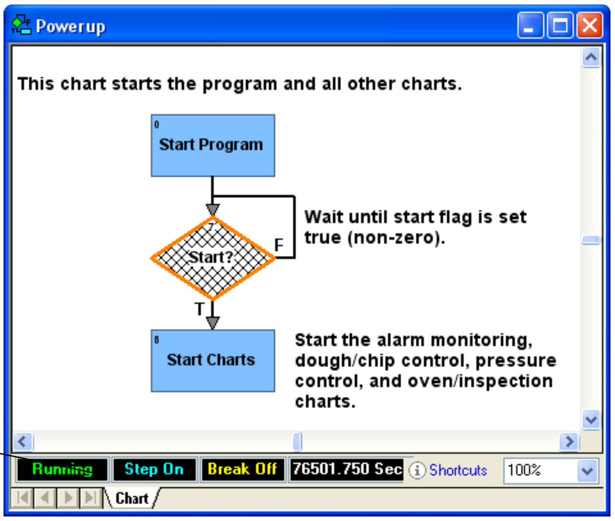
If you don't want the debugger to follow auto stepping, deselect "When stepping, make sure the active block is visible."

Pausing a Chart or Subroutine

You can temporarily stop any running chart or subroutine by pausing it. When you pause a chart or subroutine, it finishes the instruction it was executing, then stops at the next block (in Minimal Debug) or the next line (in Full Debug).

To pause the chart or subroutine in the active window, click the Pause Chart or Pause Subroutine icon, or press F7, or select Debug > Pause Chart or Debug > Pause Subroutine.

Here's an example of a paused chart:


Powerup
This chart starts the program and all other charts.
Start Program
Start?
F
T
Wait until start flag is set true (non-zero).
Start Charts
Start the alarm monitoring, dough/chip control, pressure control, and oven/inspection charts.
Running
Step On
Break Off
76501.750 Sec
Shortcuts
100%
Chart
Status bar—Step On

Hatch marks and a red outline appear on the Start block, indicating that this block is to be executed next. The status bar shows Step On, which means you can step through the chart or subroutine if you wish.

Stepping Through a Chart or Subroutine

When you step through a chart or subroutine, you control the timing and execution of its commands in a running strategy. You can see what commands are being executed when, and you can monitor the status of variables and I/O that are affected by the commands.

There are two types of stepping: single-stepping and automatic stepping. Use single stepping to go through flowchart blocks at your own pace. Use auto stepping to watch the flowchart step automatically.

***CAUTION**: Since stepping through a running chart or subroutine—even auto stepping—slows down execution, be cautious if your strategy is running on real equipment. For example, stepping through a strategy might leave a valve open much longer than it should be.*

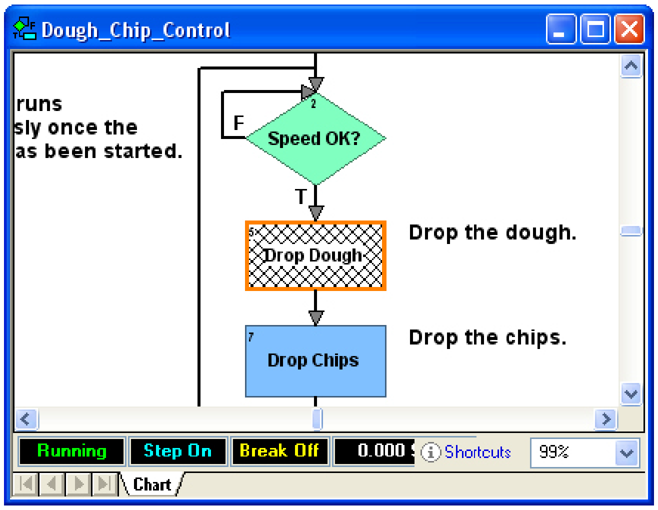
Single Stepping

When you are debugging a strategy, start by using the Step Over icon to go through a chart one block at a time. The Step Over icon may be all you need to find any problems. If necessary, go back to Configure mode and change to full debug (see "Choosing Debug Level" on page 211) so you can step into blocks and execute one line at a time.

1. Pause the chart or subroutine to be stepped through by pressing the Pause Chart or Pause Subroutine icon ⏸.
2. To step to the next command block, click the Step Over icon (or press F10, or select Debug > Step Over).

   The commands in the highlighted block are executed, the hatch marks move to the next command block, and the chart pauses again. Compare the chart below to the one on page 212.

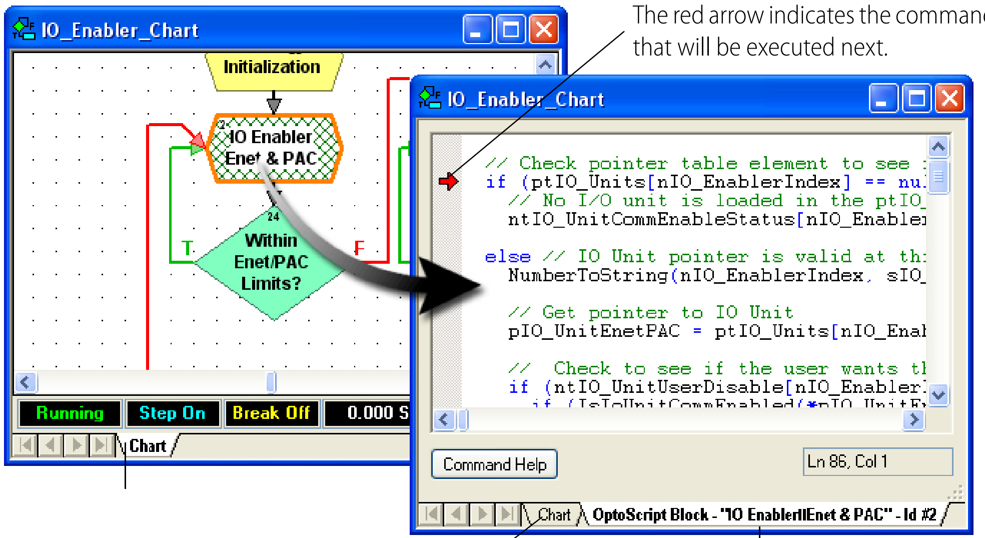
The hatch mark has moved to the next block.



3. If you need to step inside flowchart blocks and move through them one command at a time (or in OptoScript blocks, one line of code at a time), make sure you have downloaded your strategy at the full debug level. See "Choosing Debug Level" on page 211 for help.
4. If the chart or subroutine is not already paused, press the Pause Chart or Pause Subroutine icon ⏸.
5. To step inside the block you are on (the one with the hatch marks), click the Step Into icon (or press F11, or select Debug > Step Into).

The Step Into button takes you inside the current block, so you can step one command at a time.

The red arrow indicates the command that will be executed next.



```
// Check pointer table element to see :
if (ptIO_Units[nIO_EnablerIndex] == nu.
  // No I/O unit is loaded in the ptIO_
  ntIO_UnitCommEnableStatus[nIO_Enable

else // IO Unit pointer is valid at th:
  NumberToString(nIO_EnablerIndex, sIO_

  // Get pointer to IO Unit
  pIO_UnitEnetPAC = ptIO_Units[nIO_Enal

  //  Check to see if the user wants tl
  if (ntIO_UnitUserDisable[nIO_Enabler.
```

The small gray tabs at the left of the white tab show how you got to where you are.

The white tab shows you where you are: inside a chart, a block, or a subroutine called by a chart.

6. To move from line to line, click either the Step Into or the Step Over icon.

   If a command within the block calls a subroutine, Step Into takes you into the subroutine. Step Over skips over the subroutine.

7. To step from a command inside a block and go to the next block, click Step Out .

   Clicking Step Out when you are on a block, rather than inside it, causes the chart to resume. In a subroutine, clicking Step Out takes you out of the subroutine.

Auto Stepping

Auto stepping lets you watch a chart's or a subroutine's logic in slow motion, one block at a time. A chart or subroutine does not have to be paused before auto stepping can begin. To begin auto stepping, click the Auto Step icon , press F8, or select Debug > Auto Step Chart.

Step Auto appears in the status bar. The hatch marks move from block to block as each block's commands are executed. When you reach a block whose code is currently being executed, the highlight around the block becomes changing shades of green instead of solid red (unless the block is executed very quickly).

A chart that contains flow-through logic stops when it has been stepped through. In a chart that contains loop logic, the auto stepping continues until you stop it by pressing the Auto Step icon again.

Setting and Removing Breakpoints

Sometimes you want to see the action at one or two blocks without having to step through an entire chart or subroutine. You can use a breakpoint to stop a running chart or subroutine just before a block is executed.

You can set a breakpoint at any block in any chart or subroutine, whether it is running or stopped, paused or auto stepped. The strategy does not need to be running. You can set up to 16 breakpoints in one chart or subroutine.

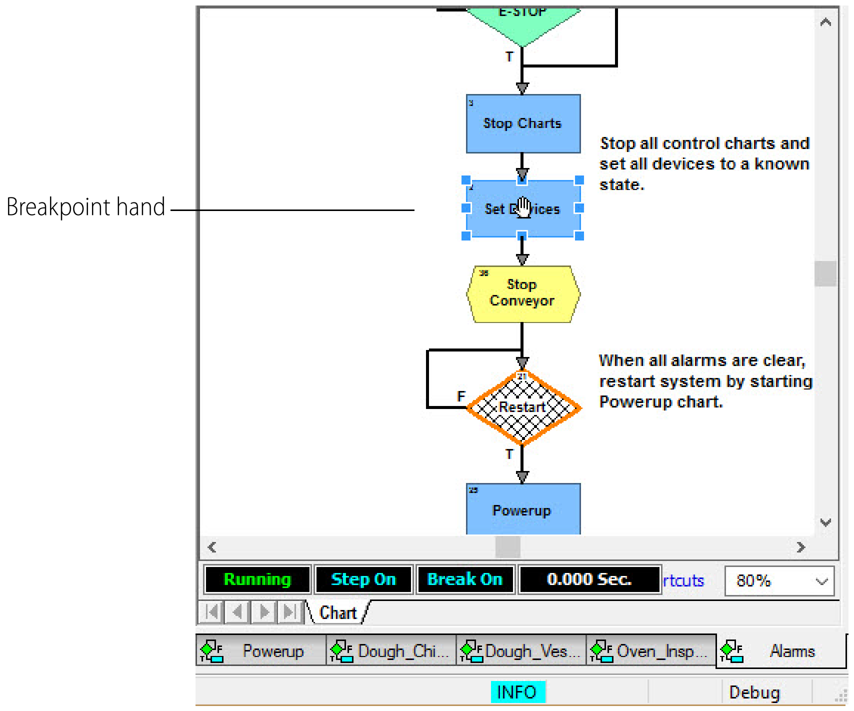
You can also set a breakpoint inside an OptoScript block, but not inside other types of blocks.

To set a breakpoint, follow these steps:

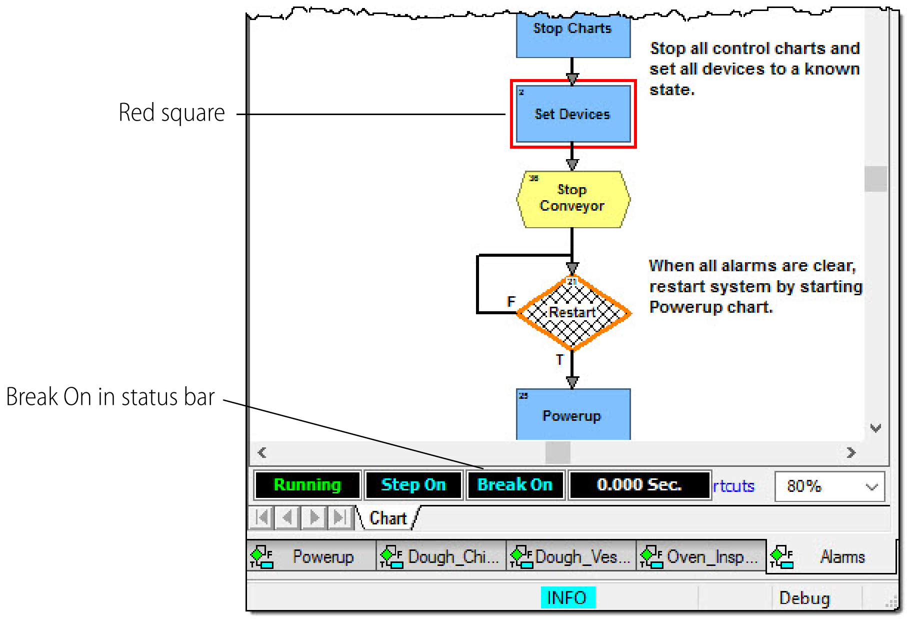
1. With the chart or subroutine open and in Debug mode, click the Breakpoint tool .

   The pointer turns into a hand.

2. To set a breakpoint on a block, click the target block to mark it with the breakpoint hand.



A red square appears around the block and Break On appears in the status bar.

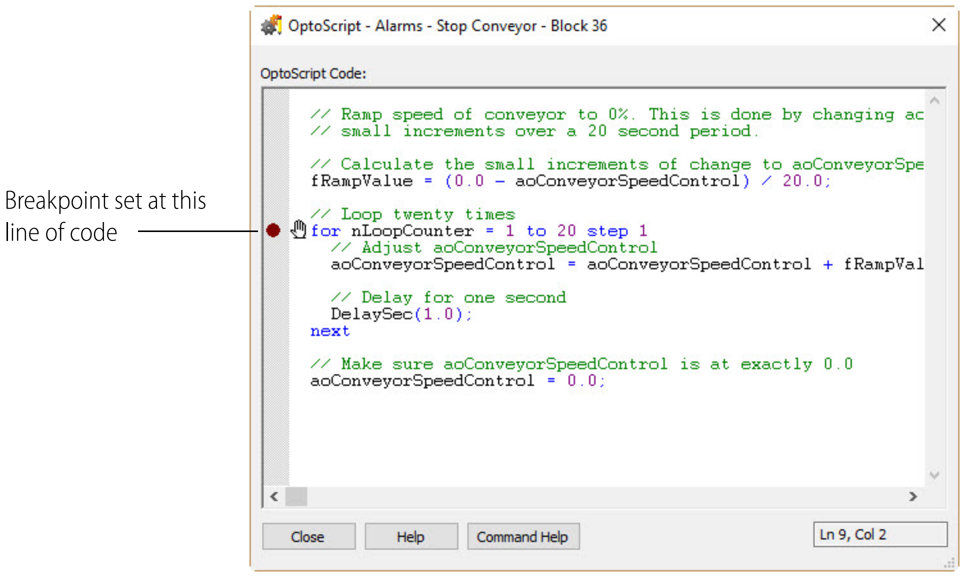


To set a breakpoint within an OptoScript block:

- With the Select tool, double-click the block to open it.
- In the PAC Control toolbar, select the Breakpoint tool, place the palm of the breakpoint hand over the line of code where you want the chart to stop, and then click the left mouse button.

A red dot marks the breakpoint. The chart will stop running before executing the marked line of code.

Note that in an OptoScript block, you can set breakpoints only on lines of code—you can't set a breakpoint on a comment or within a comment block.


Breakpoint set at this line of code
OptoScript - Alarms - Stop Conveyor - Block 36
OptoScript Code:
Close
Help
Command Help
Ln 9, Col 2

```
// Ramp speed of conveyor to 0%. This is done by changing ac
// small increments over a 20 second period.

// Calculate the small increments of change to aoConveyorSpe
fRampValue = (0.0 - aoConveyorSpeedControl) / 20.0;

// Loop twenty times
for nLoopCounter = 1 to 20 step 1
  // Adjust aoConveyorSpeedControl
  aoConveyorSpeedControl = aoConveyorSpeedControl + fRampVal

  // Delay for one second
  DelaySec(1.0);
next

// Make sure aoConveyorSpeedControl is at exactly 0.0
aoConveyorSpeedControl = 0.0;
```

3. Click other blocks or OptoScript lines to set additional breakpoints, or click blocks or lines currently marked with a hand to remove the breakpoint. (Make sure that the palm of the breakpoint hand is directly over the line you want to select.)
4. When you have finished marking or removing breakpoints, click the Breakpoint icon again or click the right mouse button within the window.

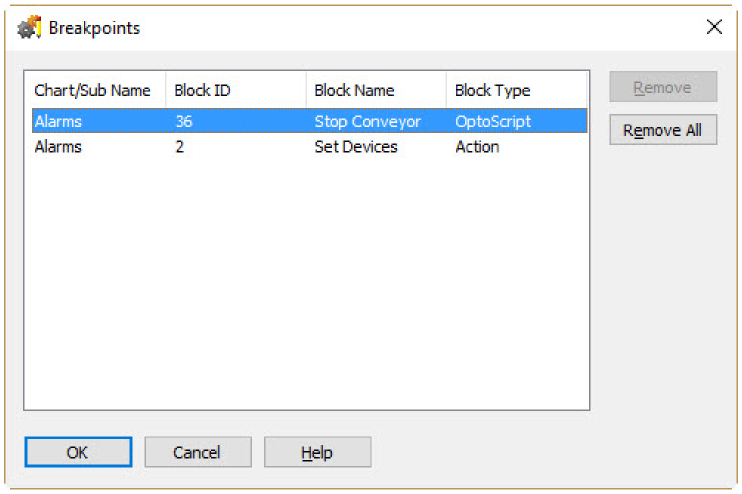
   When the chart or subroutine runs, it pauses just before executing the breakpoint block or line. You can inspect variables or I/O points, disable strategy elements, change values, and so on to see the effect the block has.
5. To single step past the breakpoint, click the Step Block or Step Line button. Or to run the chart or subroutine at full speed after the breakpoint, click the Pause Chart icon.

Managing Multiple Breakpoints

You can see all the breakpoints you have set, and you can remove breakpoints one at a time or all at once. See also, "Setting and Removing Breakpoints" on page 215 to use the breakpoint tool.

1. Press Ctrl+B or click Debug > Breakpoints.

The Breakpoints dialog box appears, showing all the breakpoints set in the strategy.

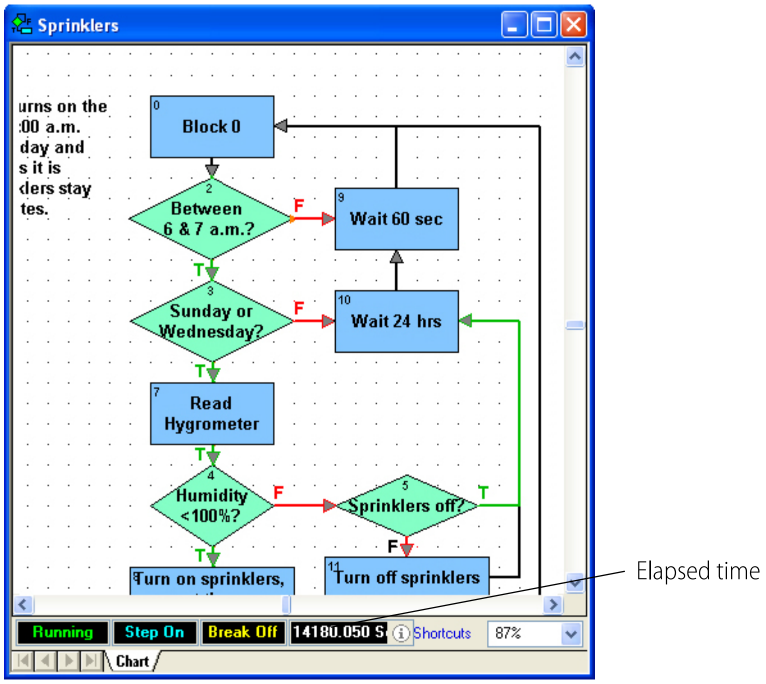


Chart/Sub Name	Block ID	Block Name	Block Type
Alarms	36	Stop Conveyor	OptoScript
Alarms	2	Set Devices	Action

2. To delete a breakpoint, highlight it and click Remove. To delete all breakpoints in the strategy at once, click Remove All.
3. When you have finished making changes, click OK.

Interpreting Elapsed Times

As you debug your strategy, you may notice elapsed time readings appearing in a chart's or subroutine's status bar, as shown below.



Elapsed time readings can help you determine how much time a chart, a subroutine, or a single block takes to execute. The readings have slightly different meanings depending on what you did to make them appear, as described in the table below:

When you . . .	Elapsed time represents . . .
Run a chart or subroutine and pause it	Time since the chart or subroutine started or was last paused
Single step (by line or block)	Time to execute the previous block
Auto step	Time to execute the most recently executed block
Hit a breakpoint	Time since the last pause, or if the chart or subroutine was not paused, elapsed time since it started running

Viewing and Printing

You can view and print several helpful things in a strategy as described in the following topics:

- "Viewing Strategy Filename and Path" (below)
- "Viewing an Individual Chart or Subroutine" on page 219
- "Viewing All Charts in a Strategy" on page 220
- "Printing Chart or Subroutine Graphics" on page 221
- "Viewing and Printing Strategy or Subroutine Commands" on page 223
- "Viewing and Printing Strategy or Subroutine Elements" on page 224
- "Viewing and Printing a Cross Reference" on page 226
- "Viewing and Printing a Bill of Materials" on page 226
- "Viewing and Printing I/O Point Mappings" on page 227

For information on viewing and changing I/O units, see "Inspecting I/O in Debug Mode" on page 171. For variables, see "Viewing Variables in Debug Mode" on page 266.

Viewing Strategy Filename and Path

To see an open strategy's file name and path, choose File > Strategy Information. A dialog box appears showing the path and file name.

Viewing an Individual Chart or Subroutine

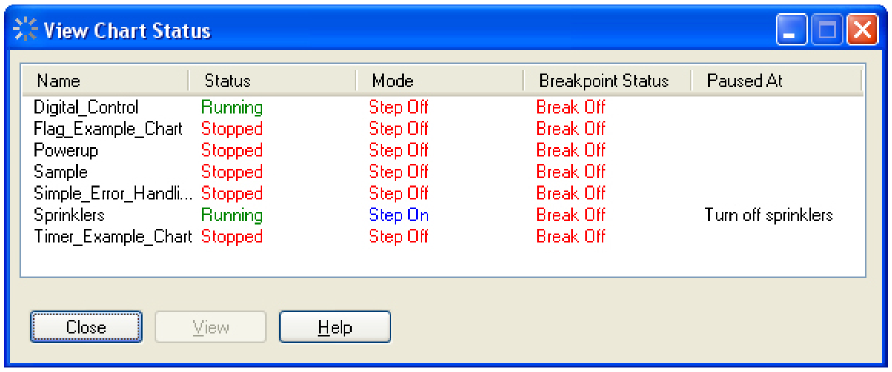
To view an individual chart or subroutine, double-click its name on the Strategy Tree, or choose Chart > Open or Subroutine > Open. You can open as many of these windows as you need. The names of open windows appear on tabs at the bottom of the PAC Control main window. Click a tab to bring its window into view.

Viewing All Charts in a Strategy

You can see the status of all charts at once and change a chart's status without having to open it.

1. Make sure the strategy is open and in Debug mode. On the Strategy Tree, double-click the Charts folder.

   The View Chart Status dialog box appears, showing every chart in the strategy:

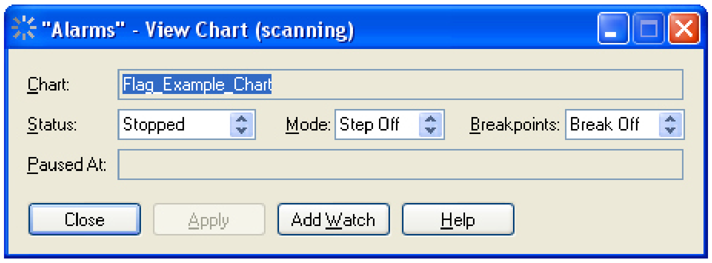


View Chart Status

Name	Status	Mode	Breakpoint Status	Paused At
Digital_Control	Running	Step Off	Break Off	
Flag_Example_Chart	Stopped	Step Off	Break Off	
Powerup	Stopped	Step Off	Break Off	
Sample	Stopped	Step Off	Break Off	
Simple_Error_Handli...	Stopped	Step Off	Break Off	
Sprinklers	Running	Step On	Break Off	Turn off sprinklers
Timer_Example_Chart	Stopped	Step Off	Break Off	

2. To change the status of a chart, double-click the chart name.

   The View Chart dialog box appears, showing the chart name, chart status, run mode, and breakpoint status. If the chart is paused (mode is Step On), the block at which it is paused is shown in the Paused At field. In the figure below, the chart is not paused:



"Alarms" - View Chart (scanning)

Chart: Flag_Example_Chart

Status: Stopped Mode: Step Off Breakpoints: Break Off

Paused At:

   The title bar shows whether scanning is occurring. Scanning stops when you click one of the changeable fields (Status, Mode, and Breakpoints) and resumes once you click Apply, another button, or one of the other fields. If scanning resumes before you click Apply, any changes you made are lost.

3. To stop, run, or suspend a chart, click an arrow in the Status field to select the option. Click Apply.
4. To turn pausing on or off, click an arrow in the Mode field to select Step On or Step Off. Click Apply.
5. To observe or ignore any breakpoints set in the chart, click an arrow in the Breakpoints field to select Break On or Break Off. Click Apply.

   This action does not clear or set breakpoints, but just determines whether the chart stops at breakpoints when it is running.

   Chart changes occur as soon as you click Apply.

6. To add the chart to a watch window so you can monitor it with other strategy elements, click Add Watch. In the dialog box, choose what to watch. Select an existing watch window to add this chart to, or create a new watch window.

   See "Using Watch Windows for Monitoring" on page 193 for more information on watch windows.

7. When you have finished making changes, click Close to return to the View Chart Status dialog box.

Printing Chart or Subroutine Graphics

You can print a chart or subroutine just as it appears on screen. You can also print all charts within a strategy. When printing a single chart or subroutine, you can preview the image to make sure it's what you want before you print it.

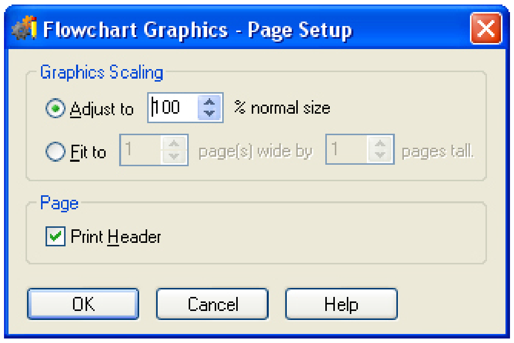
NOTE: If you have trouble printing graphics, set your printer for PostScript emulation.

Setting Up the Page

Before printing graphics, you should verify your page setup, which determines how each graphic appears on a page.

1. In Configure mode, select File > Page Setup.

   The Page Setup dialog box appears.


Flowchart Graphics - Page Setup

Graphics Scaling

(●) Adjust to 100 % normal size

( ) Fit to 1 page(s) wide by 1 pages tall.

Page

[✓] Print Header

2. In the Graphics Scaling area, choose whether you want each flowchart to print at a fixed percentage of normal size or to span a specific number of pages.

   – To print at a fixed percentage, click the Adjust To option and specify any scaling from one percent to 1,000 percent.

     You can type in a number or click the arrows to go up or down to the next increment of 25 percent. Typically, percentages between 50 percent and 200 percent work the best.

   – To print to a specific number of pages, click the Fit To option and select the number of pages wide and tall you would like each chart to print.

     If you choose one for each dimension, each chart prints to a single page. For each dimension, you can specify any integer between one and 255, but be careful. Selecting values of five and five, for example, would cause each chart to print five pages wide and five pages long, a total of 25 pages.

3. (Recommended) To print a header on each page, put a check mark in the Print Header box.

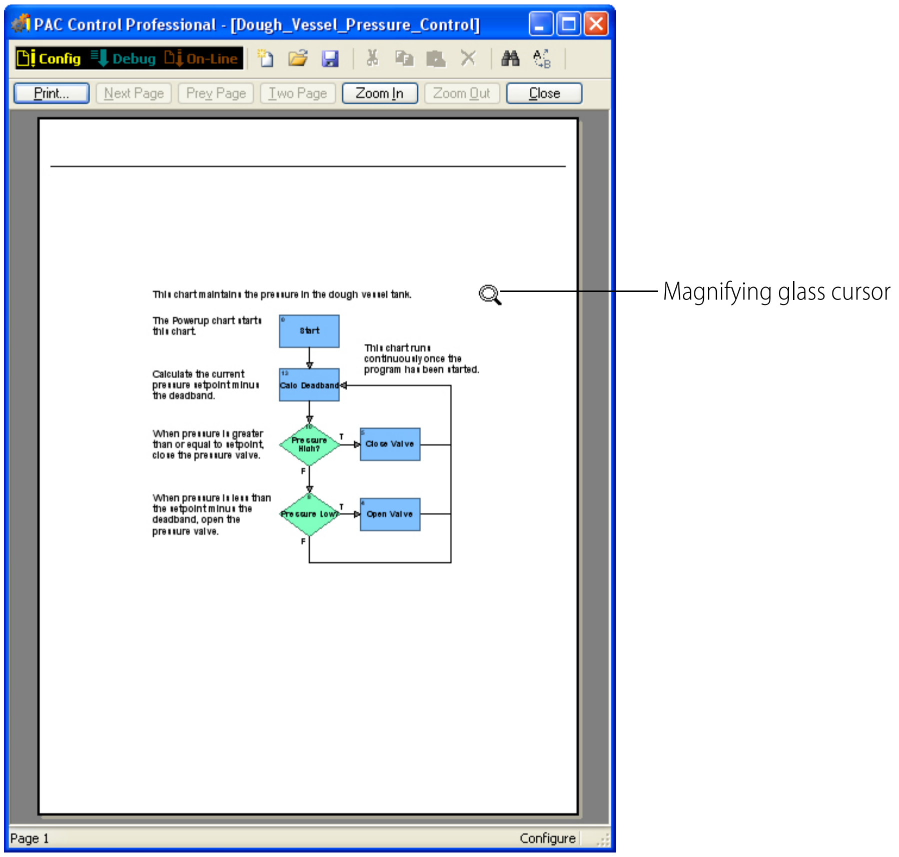
The header lists the strategy name, chart or subroutine name, date and time of printing, page number, and column and row of the page with respect to the full chart printout.

4. Click OK to save your settings.

Previewing a Flowchart Printout

1. To see how a chart or subroutine will print before actually printing it, open the chart or subroutine window.
2. From the Chart or Subroutine menu, select Print Preview Graphics.

   The preview window appears, showing the image as it will print. The cursor becomes a magnifying glass.



This chart maintains the pressure in the dough vessel tank.

The Powerup chart starts this chart.

This chart runs continuously once the program has been started.

Calculate the current pressure setpoint minus the deadband.

When pressure is greater than or equal to setpoint, close the pressure valve.

When pressure is less than the setpoint minus the deadband, open the pressure valve.

3. To zoom in at 200 percent, click where you want to see more closely. Click again to zoom in at 400 percent. Click a third time to return to 100 percent view.

   You can also use the Zoom In and Zoom Out buttons at the top of the window to zoom in or out with respect to the top left corner of the image.
4. If the image spans more than one page, click the Next Page or Prev Page buttons to view the next or previous page. To switch between a single-page view and a double-page view, click the Two Page/One Page button.
5. To print, click the Print icon to open the standard Windows Print dialog box. To change settings before printing, click Close and see "Setting Up the Page" on page 221.

Printing One Chart or Subroutine

1. To print one chart or subroutine, open its window.
2. From the Chart or Subroutine menu, select Print Graphics.
3. In the standard Windows Print dialog box, do one of the following:
   - To print to a printer, select the printer, page range, and number of copies. Click OK.
   - To print to a file, select Print to file and click OK. In the dialog box, type the file name and location.

Printing All Charts in a Strategy

***CAUTION**: You can print all charts included in a strategy, but be sure that's what you want to do before you begin. You cannot cancel once printing has started.*

1. To print all charts within a strategy, open the strategy and check the page setup.

   For help, see "Setting Up the Page" on page 221.
2. Select File > Print All Graphics.

   Printing begins immediately; no Print dialog box appears. Messages inform you of each chart's printing progress. To skip printing a particular chart, click Cancel when its message appears.

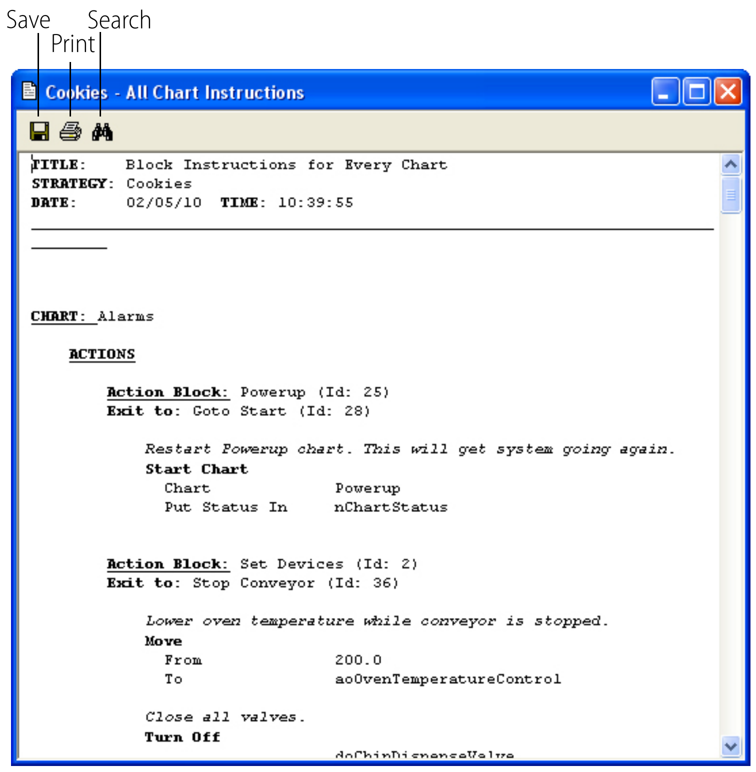
Viewing and Printing Strategy or Subroutine Commands

You must be in Configure mode to view and print commands.

1. To view all commands (instructions) in a chart or subroutine, open its window and select View/Print Instructions from the Chart or Subroutine menu. Choose whether to sort instructions by block name or block ID number.
2. To view all instructions in an entire strategy, select File > View/Print > All Chart Instructions.

   NOTE: Subroutine instructions are not included; you can print them separately.

PAC Control processes the information and displays it in the Instructions window.



```
TITLE:     Block Instructions for Every Chart
STRATEGY:  Cookies
DATE:      02/05/10  TIME: 10:39:55
________________________________________________________
________


CHART: Alarms

    ACTIONS

        Action Block: Powerup (Id: 25)
        Exit to: Goto Start (Id: 28)

            Restart Powerup chart. This will get system going again.
            Start Chart
              Chart                Powerup
              Put Status In        nChartStatus


        Action Block: Set Devices (Id: 2)
        Exit to: Stop Conveyor (Id: 36)

            Lower oven temperature while conveyor is stopped.
            Move
              From                 200.0
              To                   aoOvenTemperatureControl

            Close all valves.
            Turn Off
                                   doChipDispenseValve
```

You may need to resize the window and use the scroll bar to see all the data. Blocks and their instructions are listed in alphabetical or ID number order by type of block: action blocks first, then OptoScript blocks, then condition blocks, and finally continue blocks.

3. To print the data, click the Print icon on the toolbar. To save it to a text file, click the Save icon. To search the data, click the Search icon. When finished, close the window.

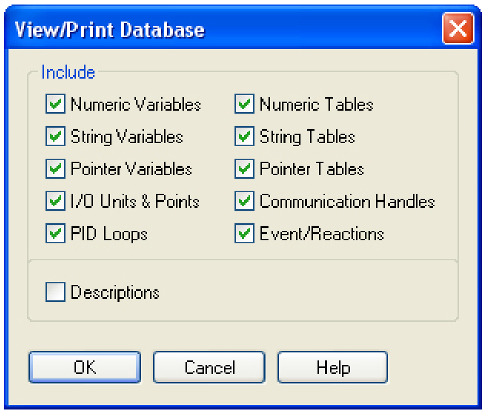
Viewing and Printing Strategy or Subroutine Elements

You must be in Configure mode.

1. To view a summary of I/O elements and variables configured in a strategy, select File > View/Print > Database.

2. To view the same summary for a subroutine, open the subroutine window and select Subroutine > View/Print > Database.

The View/Print Database dialog box appears.

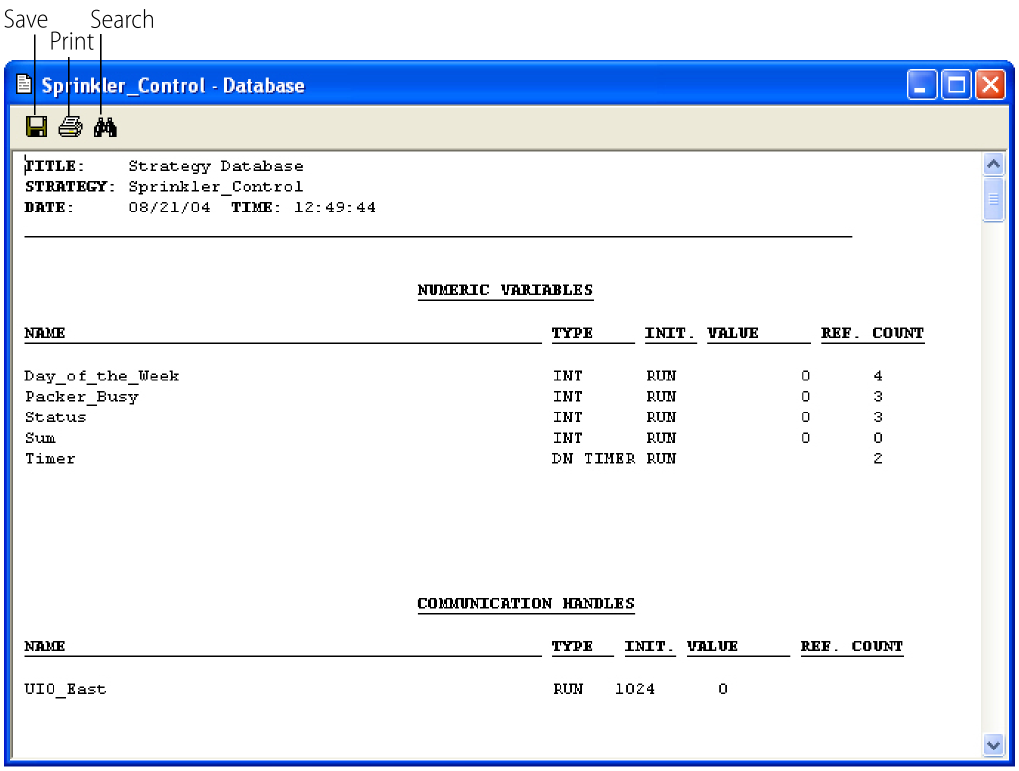


3. Make sure all element types you want to include are checked. Click to uncheck any elements you do not want.

4. To include descriptive comments associated with the elements, click to put a check mark next to Descriptions.

5. Click OK.

PAC Control processes the database and puts the data in the Database window.



```
TITLE:     Strategy Database
STRATEGY:  Sprinkler_Control
DATE:      08/21/04   TIME: 12:49:44

                                NUMERIC VARIABLES

NAME                                          TYPE     INIT. VALUE     REF. COUNT

Day_of_the_Week                               INT      RUN             0     4
Packer_Busy                                   INT      RUN             0     3
Status                                        INT      RUN             0     3
Sum                                           INT      RUN             0     0
Timer                                         DN TIMER RUN                   2


                                COMMUNICATION HANDLES

NAME                                          TYPE   INIT. VALUE     REF. COUNT

UIO_East                                      RUN    1024      0
```

You may need to resize the window and use the scroll bar to see all the data. For each element the name and reference count (that is, how many times the element is used in strategy commands) are shown, plus other information depending on the element type. The figure above shows numeric variables and communication handles.

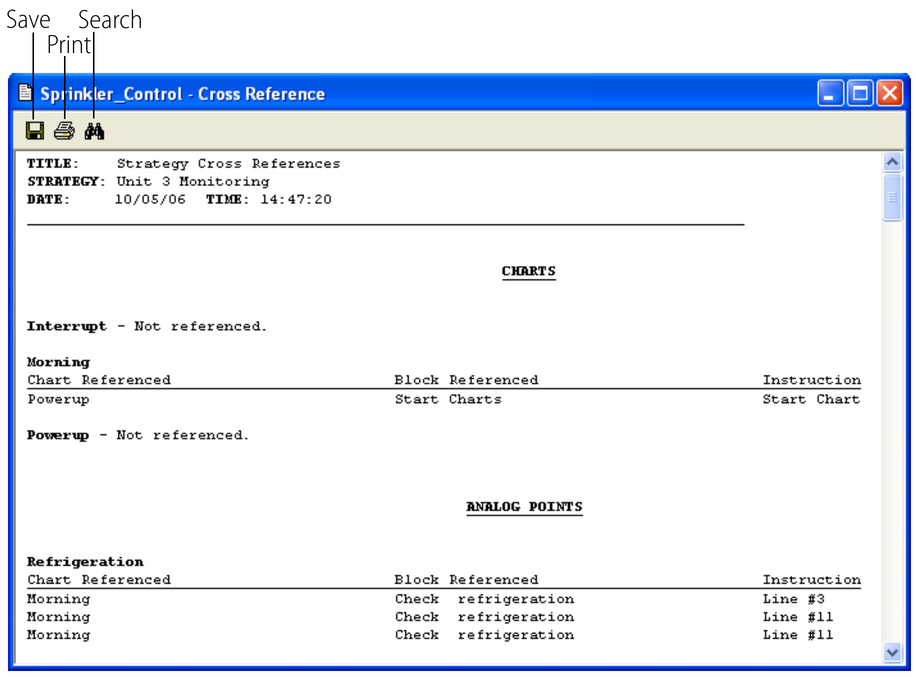
6. To print the data, click the Print icon on the toolbar. To save it to a text file, click the Save icon. To search the data, click the Search icon. When finished, close the window.

Viewing and Printing a Cross Reference

You can view and print a report of every operand in your strategy or subroutine—charts, I/O units, analog points, digital points, communication handles, numeric variables, string variables, pointer variables, numeric tables, string tables, and pointer tables. The operands are cross-referenced to the charts, blocks, and instructions in which they are used.

1. To produce a cross reference for a strategy, open it and select File > View/Print > Cross Reference.
2. To view a similar report for a subroutine, open the subroutine window and select Subroutine > View/Print > Cross Reference.

   PAC Control processes the data and puts it in the Cross Reference window.



```
TITLE:    Strategy Cross References
STRATEGY: Unit 3 Monitoring
DATE:     10/05/06   TIME: 14:47:20

                                     CHARTS

Interrupt - Not referenced.

Morning
Chart Referenced                    Block Referenced                    Instruction
Powerup                             Start Charts                        Start Chart

Powerup - Not referenced.

                                   ANALOG POINTS

Refrigeration
Chart Referenced                    Block Referenced                    Instruction
Morning                             Check  refrigeration                Line #3
Morning                             Check  refrigeration                Line #11
Morning                             Check  refrigeration                Line #11
```

   You may need to resize the window and use the scroll bar to see all the data. Notice that the Instruction column (at right) shows the line number the operand appears in when it is in OptoScript code.

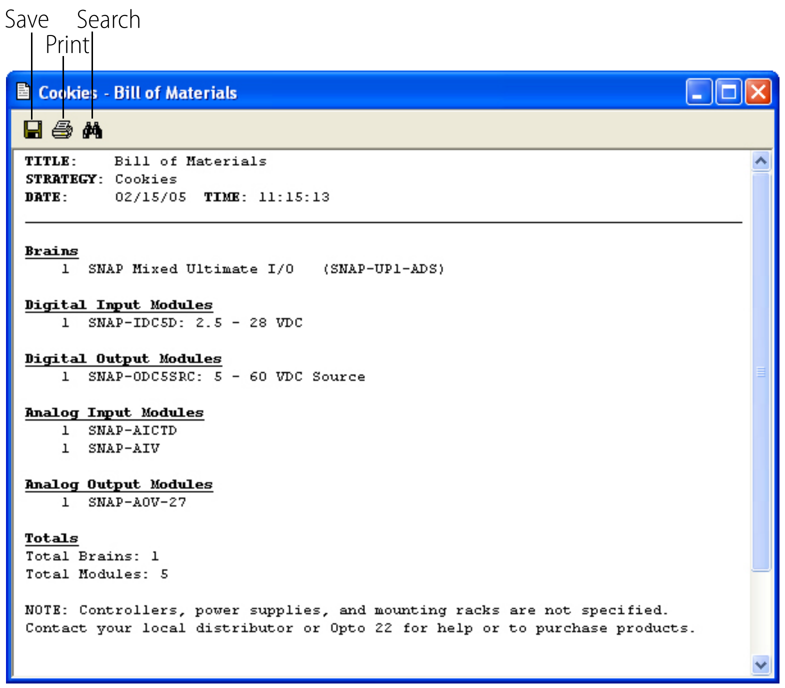
3. To print the data, click the Print icon on the toolbar. To save it to a text file, click the Save icon. To search the data, click the Search icon. When finished, close the window.

Viewing and Printing a Bill of Materials

You can view and print a bill of materials (BOM) that lists all the I/O units and I/O modules (analog and standard digital) required to run the strategy. (Special-purpose modules, such as serial and high-density digital modules, are not included in the BOM.)

1. To produce a BOM for a strategy, open it and select File > View/Print > Bill of Materials.

   PAC Control processes the data and puts it in the Bill of Materials window.



```
TITLE:    Bill of Materials
STRATEGY: Cookies
DATE:     02/15/05  TIME: 11:15:13

Brains
    1  SNAP Mixed Ultimate I/O   (SNAP-UP1-ADS)

Digital Input Modules
    1  SNAP-IDC5D: 2.5 - 28 VDC

Digital Output Modules
    1  SNAP-ODC5SRC: 5 - 60 VDC Source

Analog Input Modules
    1  SNAP-AICTD
    1  SNAP-AIV

Analog Output Modules
    1  SNAP-AOV-27

Totals
Total Brains: 1
Total Modules: 5

NOTE: Controllers, power supplies, and mounting racks are not specified.
Contact your local distributor or Opto 22 for help or to purchase products.
```

   You may need to resize the window and use the scroll bar to see all the data.

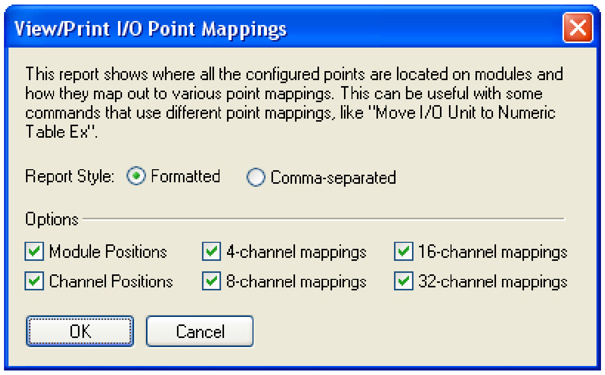
2. To print the data, click the Print icon on the toolbar. To save it to a text file, click the Save icon. To search the data, click the Search icon. When finished, close the window.

Viewing and Printing I/O Point Mappings

You can view and print a report that shows the location and mappings of I/O points. You can use this report as a look-up table for programming purposes. This can be useful with some commands that use different point mappings such as Move I/O Unit to Numeric Table Ex, Move Numeric Table to I/O Unit Ex, and IVAL Move Numeric Table to I/O Unit Ex.

This report basically provides the same information provided in "Table Index Offset Examples," in form 1701, the *PAC Control Command Reference*, but for the actual points in the strategy.

1. To produce an I/O point mappings report for a strategy, open the strategy and select File > View/Print > I/O Point Mappings.

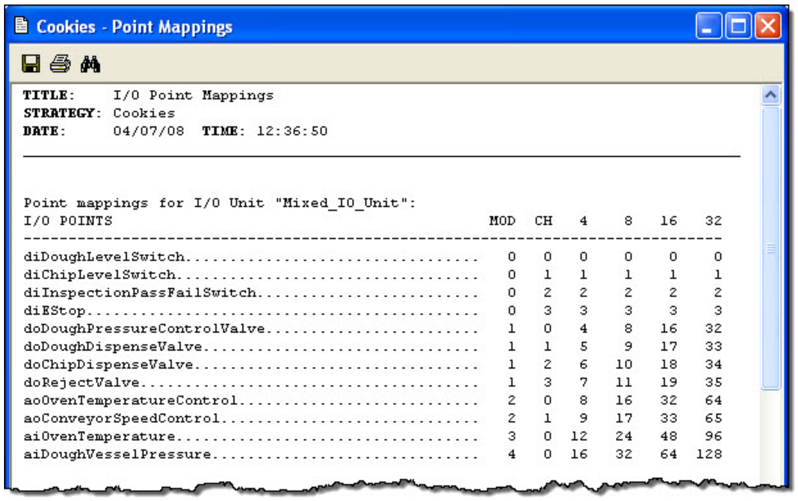


This report shows where all the configured points are located on modules and how they map out to various point mappings. This can be useful with some commands that use different point mappings, like "Move I/O Unit to Numeric Table Ex".

Report Style: [✓ Formatted] Comma-separated

Options
[✓ Module Positions] [✓ 4-channel mappings] [✓ 16-channel mappings]
[✓ Channel Positions] [✓ 8-channel mappings] [✓ 32-channel mappings]

On the View/Print I/O Point Mappings dialog box, select the report style, either Formatted or Comma-separated, and select the position and mappings types, then click OK. A report window appears.



```
TITLE:     I/O Point Mappings
STRATEGY:  Cookies
DATE:      04/07/08  TIME: 12:36:50
________________________________________________________________________________


Point mappings for I/O Unit "Mixed_IO_Unit":
I/O POINTS                                                MOD   CH    4    8   16   32
--------------------------------------------------------------------------------------
diDoughLevelSwitch.................................         0    0    0    0    0    0
diChipLevelSwitch..................................         0    1    1    1    1    1
diInspectionPassFailSwitch.........................         0    2    2    2    2    2
diEStop............................................         0    3    3    3    3    3
doDoughPressureControlValve........................         1    0    4    8   16   32
doDoughDispenseValve...............................         1    1    5    9   17   33
doChipDispenseValve................................         1    2    6   10   18   34
doRejectValve......................................         1    3    7   11   19   35
aoOvenTemperatureControl...........................         2    0    8   16   32   64
aoConveyorSpeedControl.............................         2    1    9   17   33   65
aiOvenTemperature..................................         3    0   12   24   48   96
aiDoughVesselPressure..............................         4    0   16   32   64  128
```

What the columns mean:

Column	Description
MOD	The module number on the rack
CH	The channel number within the module
4	4-channel mapping
8	8-channel mapping

Searching and Replacing

You can search a chart, subroutine, or strategy for missing connections, empty condition blocks, or any command or operand. An operand is anything that can be affected by a command, including charts, I/O units, analog points, digital points, and all kinds of variables. Searching includes OptoScript code within OptoScript blocks.

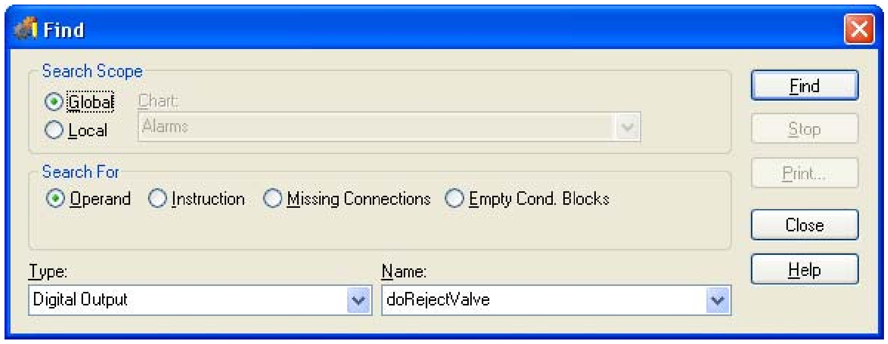
You can also replace instructions or operands with similar items.

Searching

You can search a strategy or one of its charts, or you can search a subroutine.

1. Open the strategy or subroutine and select Edit > Find.

The Find dialog box appears.


Find

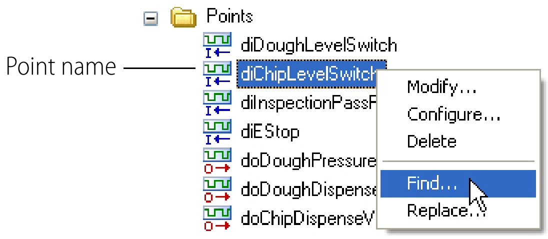
Search Scope: [✓ Global] Local — Chart: Alarms

Search For: [✓ Operand] Instruction Missing Connections Empty Cond. Blocks

Type: Digital Output

Name: doRejectValve

Or, if the item you want to find is listed in the strategy tree, such as a variable, table, or point, right-click the item and select either Find in the pop-up menu.



The item appears in the Find or dialog box automatically.

2. Under Search Scope, to search the entire strategy, click Global. To search one chart only, click Local and choose the chart name from the drop-down list.

If you are searching a subroutine, the search is Local and the subroutine's name is shown.

3. Under Search For, choose one of the following:

– To search for a chart, an I/O unit or point, or a variable, click Operand. In the Type and Name fields, choose the operand you want from the drop-down list.

– To search for an instruction, click Instruction. Click Action or Condition, and choose the instruction you want from the drop-down list.

– To search for blocks that are not connected to other blocks, click Missing Connections.

– To search for condition blocks that have no instructions, click Empty Cond. Blocks.

4. Click Find.

The dialog box expands and the search results appear at the bottom.

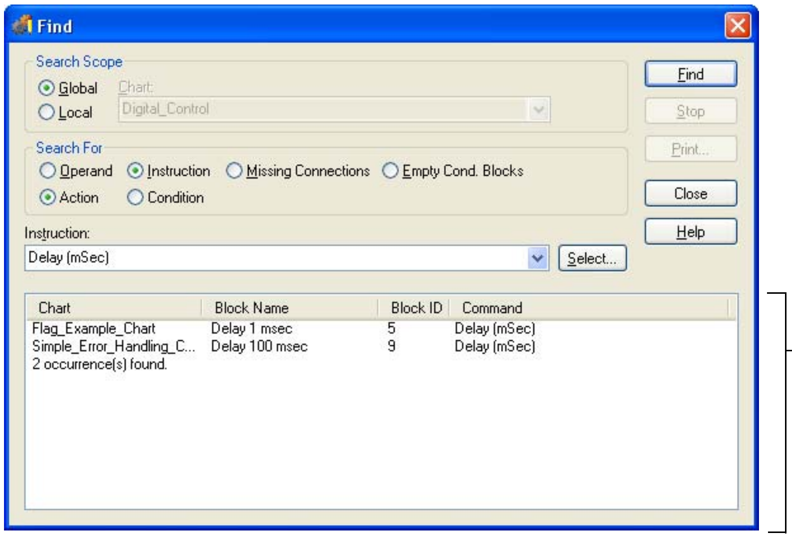


Chart	Block Name	Block ID	Command
Flag_Example_Chart	Delay 1 msec	5	Delay (mSec)
Simple_Error_Handling_C...	Delay 100 msec	9	Delay (mSec)

2 occurrence(s) found.

For more information on any item in the search results, try double-clicking the item.

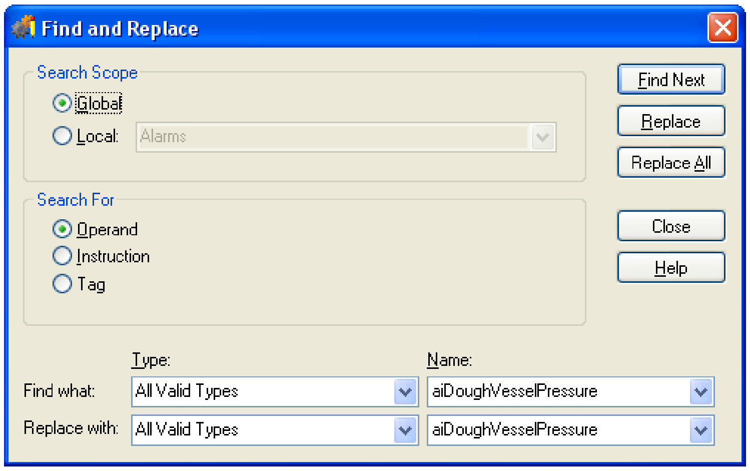
5. To save the search results to a file or to print them, click Print. In the window that opens, click the Save icon to save or the Print icon to print your search results.
6. When you have finished your search, close the Find dialog box.

Replacing

You can also replace any operand, instruction, or tag with a similar item. As in searching, you can replace items in a strategy or one of its charts, or you can replace items in a subroutine.

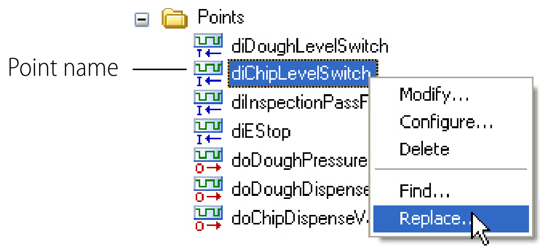
1. Open the strategy or subroutine, and select Edit > Replace.

   The Find and Replace dialog box appears.

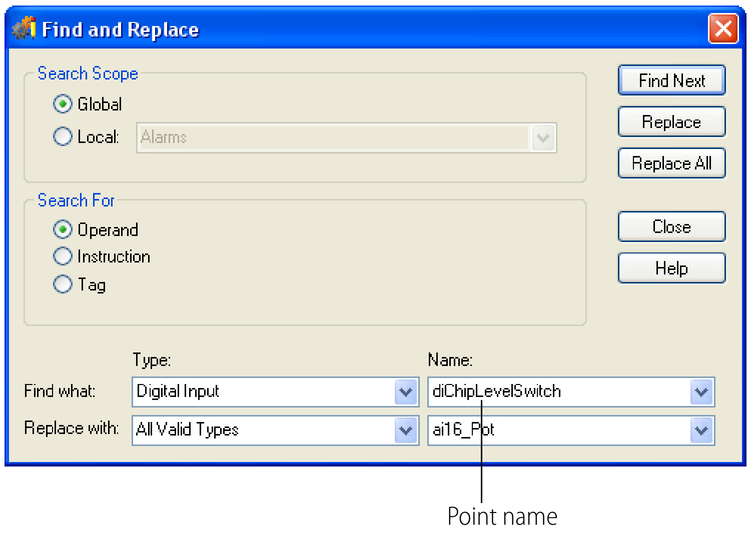


	Type:	Name:
Find what:	All Valid Types	aiDoughVesselPressure
Replace with:	All Valid Types	aiDoughVesselPressure

Or, if the item you want to find is listed in the strategy tree, such as a variable, table, or point, right-click the item and select Replace in the pop-up menu.



The item appears in the Find and Replace dialog box automatically.



Find what: Type: Digital Input; Name: diChipLevelSwitch
Replace with: Type: All Valid Types; Name: ai16_Pot

2. Under Search Scope, to search the entire strategy, click Global. To search one chart only, click Local and choose the chart name from the drop-down list.

   If you are searching a subroutine, the search is Local and the subroutine's name is shown.

3. Under Search For, choose one of the following:

   – To search for a chart, an I/O unit or point, or a variable, click Operand. In the Find Type and Name fields, choose the operand you want to replace from the drop-down list. In the Replace With Type and Name fields, choose the operand you want to use instead.

   – To search for an instruction, click Instruction. Click Action or Condition, and choose the instruction you want to replace from the Find drop-down list. In the Replace With drop-down list, choose the instruction you want to use instead.

   – To search for a substring within tag names, click Tag. In the "Find what" text box under Substrings, enter the substring you want to replace. In the "Replace with" text box, enter the replacement text. For example, if you have several tag names that contain the substring, "Generic," such as *nGenericCount*, *fGenericTemperature,* and *diGenericInput*, you could replace "Generic" with "Station1," and the tagnames would become *nStation1Count*, *fStation1Temperature,* and *diStation1Input.*

4. Click Find Next.

When the first occurrence of the operand or instruction is found, the Instructions dialog box it appears in is displayed.

5. To replace this occurrence, click Replace. To skip it and find the next one, click Find Next.

   If you are replacing operands, you can replace all occurrences at once by clicking Replace All. If you are replacing instructions, you must verify each one.

6. If the Edit Instructions dialog box appears, make any necessary changes and click OK to save them before moving on.

7. When replacements are finished, close the Find and Replace dialog box.

Legacy Options

By default, PAC Control initially shows only SNAP PAC I/O units and the commands used with them. When you're using the SNAP PAC system only, hiding legacy I/O units and commands makes it simpler and less confusing to build your strategy. However, the legacy capabilities can be made visible as needed in a specific strategy.

NOTE: If you enable legacy I/O units and commands in PAC Control, see the legacy versions of the PAC control guides for information on how to use them:

- *Opto 22 form 1710, the* PAC Control User's Guide, Legacy Edition
- *Opto 22 form 1711, the* PAC Control Command Reference, Legacy Edition
- *Opto 22 form 1713, the* PAC Control User's Guide, Legacy Edition

Existing Strategies

When you open an existing ioControl strategy in PAC Control, either Basic or Professional, PAC Control will automatically show the I/O units and commands appropriate to that strategy. For example, if you have been using SNAP Ultimate I/O, SNAP Ultimate I/O units will be visible in the strategy. In addition, commands used with SNAP Ultimate I/O but which are now deprecated (obsolete) because they are not needed with SNAP PACs are also available.

Similarly, if you open an existing strategy that contains *mistic* serial I/O units, all the *mistic* I/O unit types and commands—such as event/reaction and *mistic* PID commands—are automatically shown.

New Strategies

When you create a new strategy in PAC Control, either Basic or Professional, the strategy will show only the SNAP PAC System. This means:

- When you use a command such as *Get I/O Unit as Binary Value*, the only I/O units shown for Argument 1 will be SNAP PAC I/O units (SNAP-PAC-R1, SNAP-PAC-R1-B, SNAP-PAC-R2, SNAP-PAC-EB1, SNAP-PAC-EB2, SNAP-PAC-SB1, and SNAP-PAC-SB2).
- Commands that are used only with legacy hardware—such as *Enable Communication to Mistic PID Loop* or *Get All HDD Module Off-Latches*—won't be listed.

Since some Opto 22 customers work with multiple systems, legacy options can be set individually for each strategy or subroutine. If you are using legacy hardware, you can choose the options that apply to each strategy you're working with, without affecting other strategies.

IMPORTANT*: Once you have enabled a legacy option for a specific strategy or subroutine, you cannot disable it later for the same strategy or subroutine.*

Enabling Legacy Options

Here's how to set legacy options:

1. In PAC Control, open the strategy or subroutine for which you want to change options.
2. Choose File > Strategy Options. Click the Legacy tab.

   Depending on the hardware you're using, you can set one or more of the following options:

   – Enable Ethernet, Ultimate, and Simple I/O units and commands (see page 233).

   – Enable high-density digital commands (see page 233).

   – (PAC Control Pro only) Enable *mistic* I/O units and commands (see page 234).
3. Click the option you want to enable. At the confirmation dialog box, make sure it is correct, and then click Yes. Repeat to enable other options.

   The options are changed for the strategy or subroutine.

Enable Ethernet, Ultimate, and Simple I/O Units and Commands

If you're using any of the I/O unit types shown in the following table, you should enable Ethernet, Ultimate, and Simple I/O units and commands. When you do, both the I/O unit types and the commands shown in the table will become available in the strategy.

I/O Unit Types	Commands
SNAP-ENET-D64 SNAP-B3000-ENET SNAP-ENET-RTC SNAP-UP1-D64 SNAP-UP1-ADS SNAP-UP1-M64 SNAP-ENET-S64	IVAL Set Digital-64 I/O Unit from MOMO Masks [DEPRECATED] IVAL Set Mixed 64 I/O Unit from MOMO Masks [DEPRECATED] IVAL Set Mixed I/O Unit from MOMO Masks [DEPRECATED] IVAL Set Simple 64 I/O Unit from MOMO Masks [DEPRECATED] Set Digital-64 I/O Unit from MOMO Masks [DEPRECATED] Set Mixed 64 I/O Unit from MOMO Masks [DEPRECATED] Set Mixed I/O Unit from MOMO Masks [DEPRECATED] Set Simple 64 I/O Unit from MOMO Masks [DEPRECATED]

Enable High-Density Digital Module Commands

All SNAP high-density digital modules are fully supported by the SNAP PAC System using regular digital point commands. However, three of our older HDD modules can also be used with analog-capable SNAP Ultimate, SNAP Ethernet, and SNAP Simple I/O units, if you use the older "legacy" HDD commands.

If you are using SNAP-ODC-32-SNK, SNAP-ODC-32-SRC, or SNAP-IDC-32 HDD modules with these older I/O units, you should enable High-Density Digital module commands. When you do, the following commands become available in the strategy:

Commands	
Clear HDD Module Off-Latches	Get All HDD Module On-Latches
Clear HDD Module On-Latches	Get All HDD Module States
Get & Clear All HDD Module Off-Latches	Get HDD Module Counters
Get & Clear All HDD Module On-Latches	Get HDD Module Off-Latches
Get & Clear HDD Module Counter	Get HDD Module On-Latches
Get & Clear HDD Module Counters	Get HDD Module States
Get & Clear HDD Module Off-Latches	Set HDD Module from MOMO Masks
Get & Clear HDD Module On-Latches	Turn Off HDD Module Point
Get All HDD Module Off-Latches	Turn On HDD Module Point

Enable *mistic* I/O Units and Commands

If you're using legacy *mistic* hardware with a SNAP PAC S-series controller and PAC Control Professional, you should enable *mistic* I/O units and commands. When you do, the following I/O unit types and commands will become available in the strategy.

I/O Units Types.

I/O Unit Types		Other Types
G4D16R	B3000 or B3000-B (Digital)	Mistic PID Loop
G4D32RS	B3000 or B3000-B (Analog)	Digital Event/Reaction
G4A8R, G4RAX	B100	Analog Event/Reaction
B200	SNAP-BRS	Event/Reaction Group

Commands.

Commands	
Clamp Mistic PID Output Clamp Mistic PID Setpoint Clear All Event Latches Clear Event Latch Convert Mistic I/O Hex String to Float Convert Number to Mistic I/O Hex String Disable Communication to Event/Reaction Disable Communication to Mistic PID Loop Disable Event/Reaction Group Disable Mistic PID Output Tracking in Manual Mode Disable Mistic PID Output Disable Mistic PID Setpoint Tracking in Manual Mode Disable Scanning for All Events Disable Scanning for Event Disable Scanning of Event/Reaction Group Enable Communication to Event/Reaction Enable Communication to Mistic PID Loop Enable Event/Reaction Group Enable Mistic PID Output Tracking in Manual Mode Enable Mistic PID Output Enable Mistic PID Setpoint Tracking in Manual Mode Enable Scanning for All Events Enable Scanning for Event Enable Scanning of Event/Reaction Group Event Occurred? Event Occurring? Event Scanning Disabled? Event Scanning Enabled? Event/Reaction Communication Enabled? Event/Reaction Group Communication Enabled? Get Analog Filtered Value Get Analog Square Root Filtered Value Get Analog Square Root Value Get Analog Totalizer Value Get & Clear Analog Filtered Value Get & Clear Analog Totalizer Value Get & Clear Event Latches Get & Restart Off-Time Totalizer Get & Restart On-Time Totalizer Get & Restart Period	Get Event Latches Get Frequency Get Mistic PID Control Word Get Mistic PID D Term Get Mistic PID I Term Get Mistic PID Input Get Mistic PID Mode Get Mistic PID Output Rate of Change Get Mistic PID Output Get Mistic PID P Term Get Mistic PID Scan Rate Get Mistic PID Setpoint Get Off-Time Totalizer Get On-Time Totalizer Get Period Get Period Measurement Complete Status IVAL Set Analog Filtered Value IVAL Set Digital Binary [DEPRECATED] IVAL Set Frequency IVAL Set Mistic PID Control Word IVAL Set Mistic PID Process Term IVAL Set Off-Totalizer IVAL Set On-Totalizer IVAL Set Period Mistic PID Loop Communication Enabled? Read Event/Reaction Hold Buffer Set Analog Totalizer Rate Set Digital I/O Unit from MOMO Masks [DEPRECATED] Set Mistic PID Control Word Set Mistic PID D Term Set Mistic PID I Term Set Mistic PID Input Set Mistic PID Mode to Auto Set Mistic PID Mode to Manual Set Mistic PID Output Rate of Change Set Mistic PID P Term Set Mistic PID Scan Rate Set Mistic PID Setpoint Transmit/Receive Mistic I/O Hex String

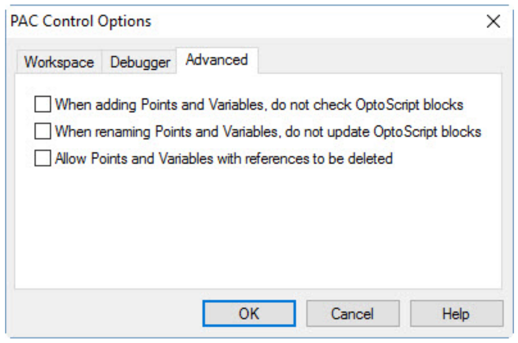
Advanced Options

Very few applications require changes to the “Advanced” options, which involve adding and changing variables and points in the strategy. However, these options can be useful for strategies with a very large number of charts.

Three options are available. Unless you are certain you need these options, do not select them.

1. To see the options, choose Configure > Options. Click the Advanced tab.

By default, these options are not selected:


PAC Control Options

Workspace | Debugger | Advanced

- [ ] When adding Points and Variables, do not check OptoScript blocks
- [ ] When renaming Points and Variables, do not update OptoScript blocks
- [ ] Allow Points and Variables with references to be deleted

2. If you are sure you require one or more options, choose the one(s) you need.

– **When adding Points and Variables, do not check OptoScript blocks:** If you check this box, when you add a point or variable, OptoScript blocks will not be searched to see if it is already being used in the block. That makes it faster to add points and variables in a strategy that has a large number of charts. However, reference counts are not updated during this process. When you have finished adding all the points and variables, choose Tools > Regenerate Reference Counts.

– **When renaming Points and Variables, do not update OptoScript blocks:** If you check this box, you can rename a variable or point in the Strategy Tree, and it will not be renamed in any OptoScript block. Renaming points and variables is faster in a strategy with a large number of charts that have no OptoScript blocks; but if any renamed items exist in OptoScript blocks, the strategy will not compile. You will have to manually edit those OptoScript blocks.

– **Allow Points and Variables with references to be deleted:** Use this feature on variables or points that are used only in OptoScript blocks. If the variable or point is used in an Action or Condition block, the strategy will not compile.

NOTE: If you want to delete a variable or point that shows a reference count greater than zero, but it is not being used in the strategy, don't use this advanced option. Instead, choose Tools > Regenerate Reference Counts to recalculate all reference counts. Then you will be able to delete the variable or point.

8: Working with Flowcharts

OPTO 22

Introduction

This chapter shows you how to work with flowcharts, the building blocks of your strategy. When you create a new strategy, one chart is created for you: the Powerup Chart. You must create all the other charts to do the work of the strategy.

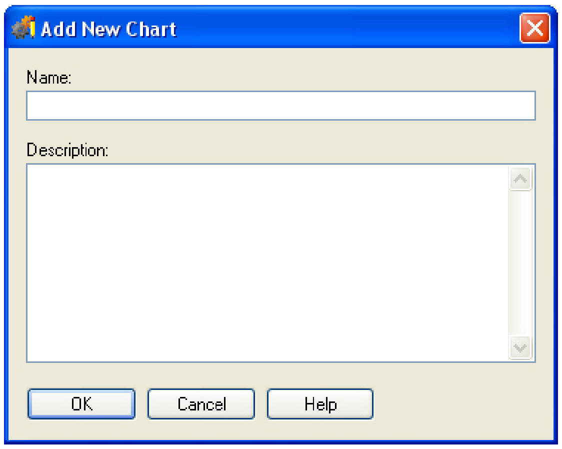
In this Chapter

Creating a New Chart 237
Working with Chart Elements 238
Opening, Saving, and Closing Charts 248
Copying, Renaming, and Deleting Charts 249
Printing Charts 251
Exporting and Importing Charts 251

Creating a New Chart

1. With your strategy open and in Configure mode, select Chart > New, or right-click the Charts folder on the Strategy Tree and select New from the pop-up menu.

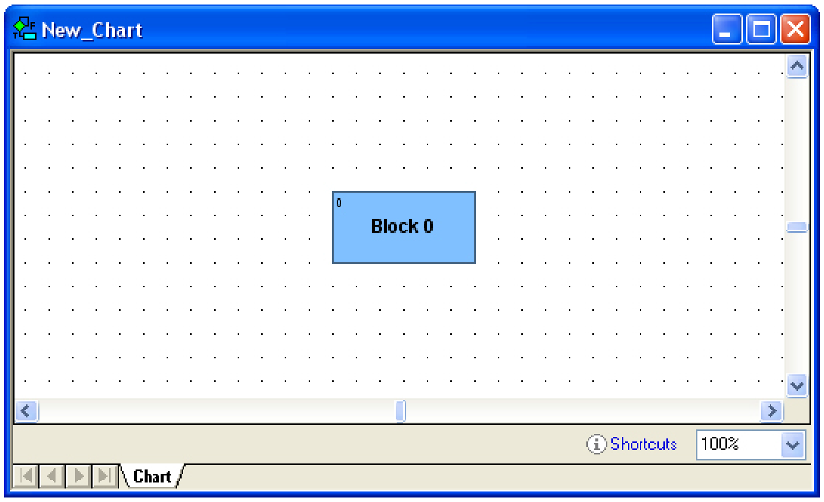
   The Add New Chart dialog box appears.



2. Enter a name for the new chart.

   The name must start with a letter, but may also include numbers and underscores. If you type spaces, they are converted to underscores. All other characters are ignored.
3. (Optional) Type a description.
4. Click OK.

   The new chart is listed on the Strategy Tree under the Charts folder, and the new chart window appears. Block 0, the starting block, is shown automatically. No matter how many other blocks you add or where you place them, block 0 is always the first block to be executed in the chart.



NOTE: Because chart windows show a small portion of a potentially large chart, a small movement with the scroll bar can mean a big change in what you see. If you lose your flowchart in the window, select View > Center on Block and choose the block you want to see in the middle of the screen. Or just right-click inside the flowchart and choose Center on Block from the popup menu.

Working with Chart Elements

Chart Components

Charts can contain five kinds of flowchart blocks, lines connecting the blocks, and text.

Action Blocks are rectangles that contain one or more commands (instructions) that do the work of the strategy, such as turning things on or off, setting variables, and so on. See "9: Using Variables and Commands" for more information. Action blocks can have more than one entrance but only one exit.

Condition Blocks are diamonds containing questions that control the logical flow of a strategy. Condition blocks can have many entrances, but only two exits: True and False.

OptoScript Blocks are hexagons containing OptoScript code, a procedural language you may want to use to simplify certain tasks. See "11: Using OptoScript" for more information. OptoScript blocks can have more than one entrance but only one exit.

Continue Blocks are ovals that contain no commands, but simply route chart logic to a new location, such as to the top of a chart. These blocks help keep charts neat by avoiding awkward connections between two blocks that are far apart.

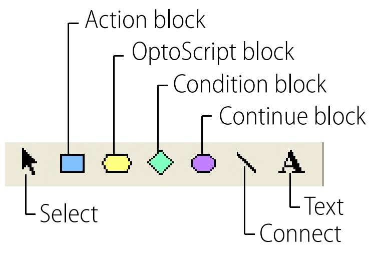
Sync Blocks are used in charts that control systems with controller redundancy. When a sync block is encountered, the redundant controllers are synchronized. For more information, see Opto 22 form 1831, the *SNAP PAC Redundancy Option User's Guide*.

Connections are lines with arrows that connect one block to the next, directing the flow of strategy logic.

Text explains the chart's purpose and elements for anyone who needs to understand them later.

Using the Drawing Toolbar

The drawing toolbar includes tools for each of the elements plus a Select tool, or pointer, for manipulating elements:



Changing the Appearance of Elements in a Chart Window

You can change the background appearance of charts or subroutines, the color and size of blocks and text, and the color of connection lines. Depending on the scope you want to affect, you can change these window properties at three levels:

- Across PAC Control—to change the appearance of all new charts in all new strategies, and all new subroutines.
- Across a strategy—to change the appearance of all new charts in the open strategy.
- For the open chart or subroutine—to change the appearance of all new elements in the open chart or subroutine window.

IMPORTANT*: Note that most changes affect only new charts and their elements. Existing charts, subroutines, and elements are* **not** *changed. To avoid having to go back and change each item individually, make sure you set the defaults the way you want them before you create new charts. Once you have changed the defaults, see "Changing Existing Elements to Match New Defaults" on page 241 to change existing elements to match the new defaults.*

To change the appearance of charts and elements, follow these steps:

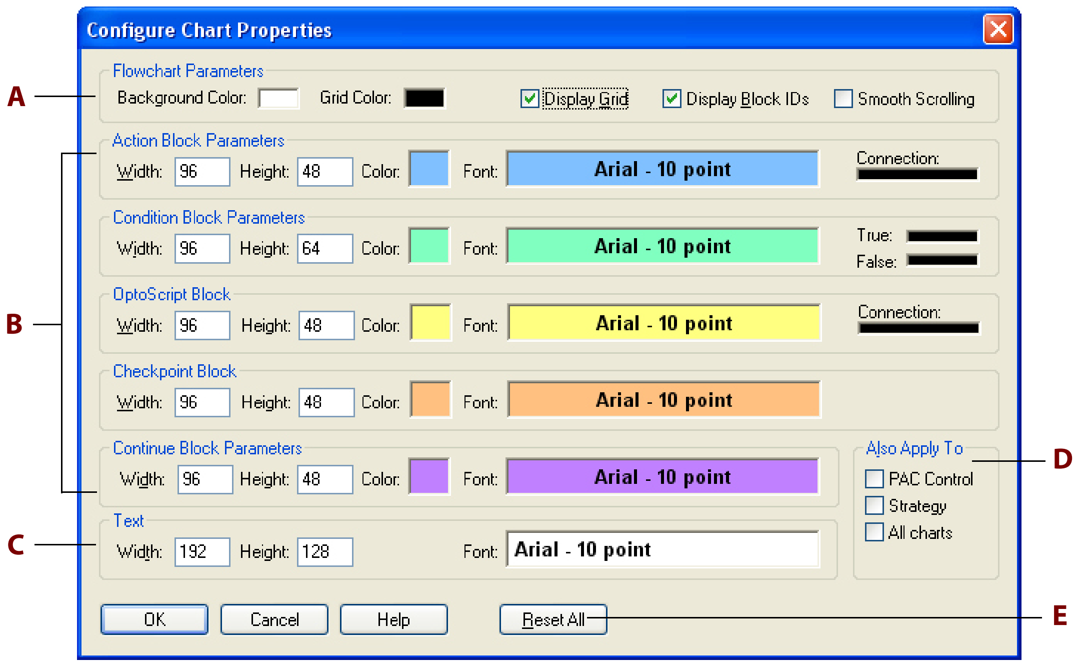
1. Choose one of the following, depending on the scope you want to change:
   - To change all new charts in all new strategies and all new subroutines, choose Configure > Default Properties to open the Configure PAC Control Default Properties dialog box.
   - To change all new charts in the open strategy only, choose File > Strategy Properties to open the Configure Strategy Properties dialog box.

– To change new elements in the open chart or subroutine only, choose Chart > Properties or Subroutine > Properties to open the Configure Chart Properties or Configure Subroutine Properties dialog box.

The dialog box that opens looks like this, except that its title may be different.

2. Complete the fields as described in "Configure Chart Properties Dialog Box" below.
3. When you have made all the changes, click OK.

Configure Chart Properties Dialog Box



Configure Chart Properties

Flowchart Parameters
Background Color: Grid Color: [✓] Display Grid [✓] Display Block IDs [ ] Smooth Scrolling

Action Block Parameters
Width: 96 Height: 48 Color: Font: Arial - 10 point Connection:

Condition Block Parameters
Width: 96 Height: 64 Color: Font: Arial - 10 point True: False:

OptoScript Block
Width: 96 Height: 48 Color: Font: Arial - 10 point Connection:

Checkpoint Block
Width: 96 Height: 48 Color: Font: Arial - 10 point

Continue Block Parameters
Width: 96 Height: 48 Color: Font: Arial - 10 point

Text
Width: 192 Height: 128 Font: Arial - 10 point

Also Apply To
[ ] PAC Control
[ ] Strategy
[ ] All charts

A—*Flowchart Properties.* Specify general chart display in this area. To apply these changes to existing charts as well as to new ones, click All charts in D.

- To change the chart background color from the default of white, click the Background Color box. Choose a new color from the Color dialog box.
- To change the chart's grid color from the default of black, click the Grid Color box.
- The grid and block ID numbers are displayed by default. To remove them, click Display Grid or Display Block ID's to remove the check mark.
- To enable or disable smooth scrolling in a flowchart, click Smooth Scrolling; this option is disabled by default.

B—*Action Block Parameters.* Define the appearance of action blocks, condition blocks, OptoScript blocks, and continue blocks in this area. These changes affect new blocks only, not existing blocks. (To change existing blocks, see "Changing Existing Elements to Match New Defaults" below.)

- In the Width and Height fields, type the block size in pixels. For action and continue blocks, the default width is 96 and the default height is 48; the minimum parameters are 48 (width) and 32 (height). For condition blocks, the default height is 64. (Note that the numbers you enter are rounded down to be evenly divisible by 16; for example, if you enter 81, click OK and then reopen the dialog box, the parameter reads 80.)

- To change the color of the blocks from the default, click the Color box.
- To change block name text formatting, click the Font box. In the Font dialog box, change the font, font style, size, effects, and color. The default font is black 10-point Arial bold.
- To change text alignment, right-click the Font box and choose left, center, or right from the pop-up menu.
- To change the color of connection lines, click a Connection line at the far right. Choose the new color from the Color dialog box.

C—*Text.* Define width, height, and font of text blocks that appear as comments in a chart or subroutine. Default width is 192; default height is 128; the minimum for both is 16. The default font is black 10-point Arial bold.

D—*Also Apply To.* To expand the scope of the changes you've made, click these boxes. Click PAC Control to apply the changes to all new strategies and subroutines in PAC Control. Click Strategy to apply the changes to all new charts in the current strategy. Click All Charts to apply the changes to all new charts and all new graphic objects added to the current strategy.

Depending on which dialog box you are in and what is currently open, one or more of these options may be grayed out. For example, if you are in the Configure PAC Control Default Properties dialog box, it is assumed that the changes are to be applied throughout PAC Control, and that option is therefore grayed out.

E—*Reset All.* To reset all parameters and options to their factory default settings, click Reset All.

Changing Existing Elements to Match New Defaults

Once you have changed the defaults for the way elements appear in a chart, you can update existing blocks, connections, and text to match.

***CAUTION**: When you update existing objects to match, you cannot undo the update.*

1. Right-click on an empty space in the chart whose elements you want to change. Choose Select from the pop-up menu. From the sub-menu, choose the item type you want.

   For example, to select all action blocks, choose Select > Action Blocks.

2. Right-click again in the chart, and choose Properties > Copy from Default from the pop-up menu.

   The color, size, and font of all selected items change to match the flowchart defaults.

Drawing Blocks

Action, condition, OptoScript, and continue blocks are all drawn the same way.

1. With the chart open and the strategy in Configure or Online mode, click the tool you want to use.

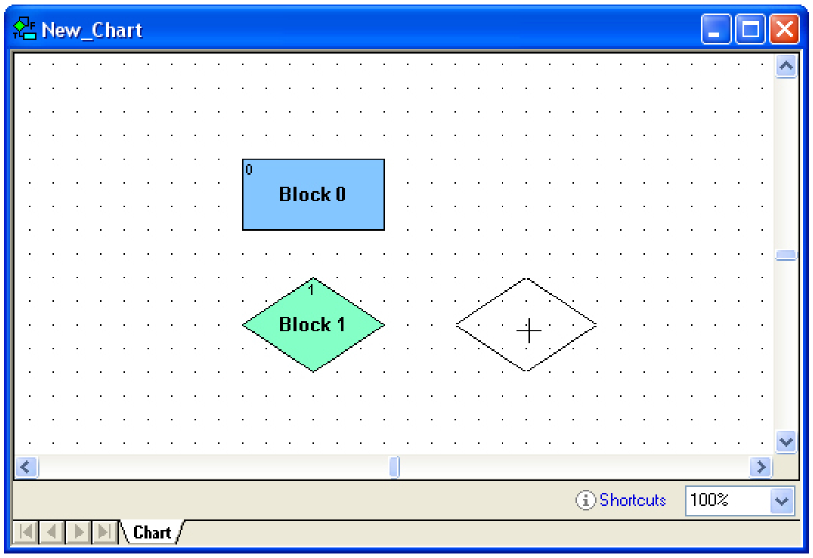
   ▭ for an action block

   ◇ for a condition block

for an OptoScript block
for a continue block.
As you move the mouse into the window, you see an outline representing the block.

2. Click where you want to place the block.

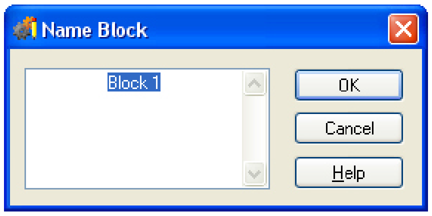
   The new block appears.



3. Click in another location to place other blocks of the same type. When you have finished using the tool, click the right mouse button, click another tool in the toolbar, or press Esc.

Naming Blocks

1. With the chart open and the strategy in Configure or Online mode, click the Select tool and click the block to select it.
2. Right-click the block and choose Name from the pop-up menu.



3. In the Name Block dialog box, type the name and click OK.

Renaming Blocks

1. With the chart open and the strategy in Configure or Online mode, click the Select tool and click the block to select it.
2. Right-click the block and choose Name from the pop-up menu.
3. In the Name Block dialog box, change the name. Click OK.

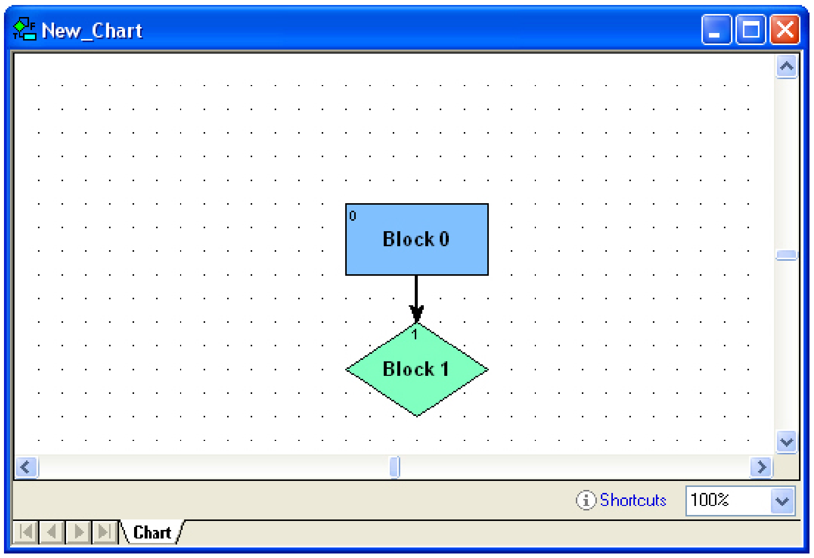
Connecting Blocks

To connect blocks, start with the chart open and the strategy in Configure or Online mode. Remember that action blocks and OptoScript blocks have only one exit, and condition blocks have two.

Action Blocks and OptoScript Blocks

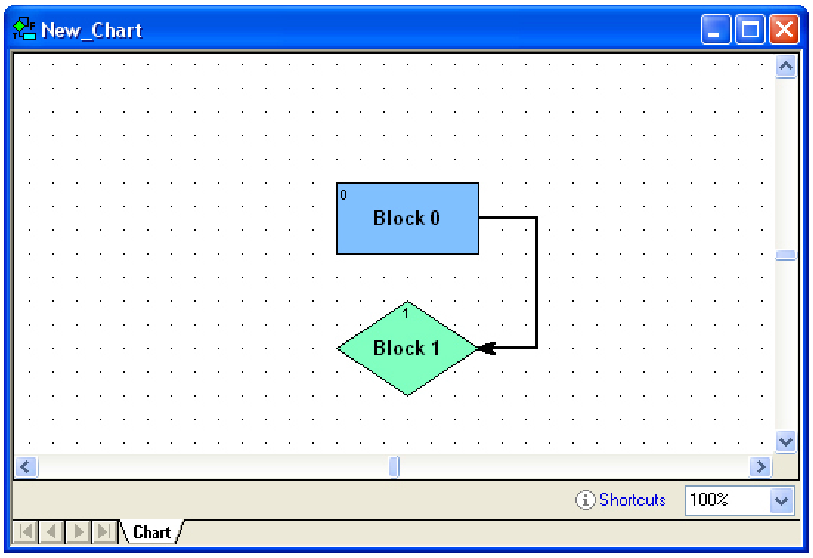
1. To connect an action block or an OptoScript block to the next block in a program sequence, click the Connect tool .
2. First click the source block and then click the destination block.

   Although you can click anywhere inside the blocks to make a connection, the connection is attached at the side closest to where you clicked. In the figure below, Block 0 is the source block and Block 1 is the destination block:



   To keep your charts neat, try to draw the most direct connections possible. To do so, after clicking the source block, move your cursor out of the block at a point closest to its destination.
3. To create a bend or elbow in a connection, click wherever you want the bend while drawing the connection.

For example, to draw the connection in the following figure, we selected the Connect tool, clicked Block 0, moved the cursor out of the block to the right, clicked at point A, clicked again at point B, and then clicked the right side of Block 1:

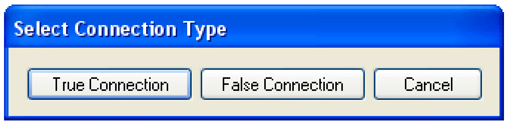


4. While you're still drawing a line, to delete an elbow you don't want, click the right mouse button once to undo it.

   If you created several elbows, you can eliminate them in reverse order with repeated right mouse clicks. If no more elbows remain and you right-click again, you delete the connection. Once you have completed a connection, however, you cannot undo it this way.

Condition Blocks

1. To connect a condition block to the next block in a program sequence, click the Connect tool .
2. Click the source block.

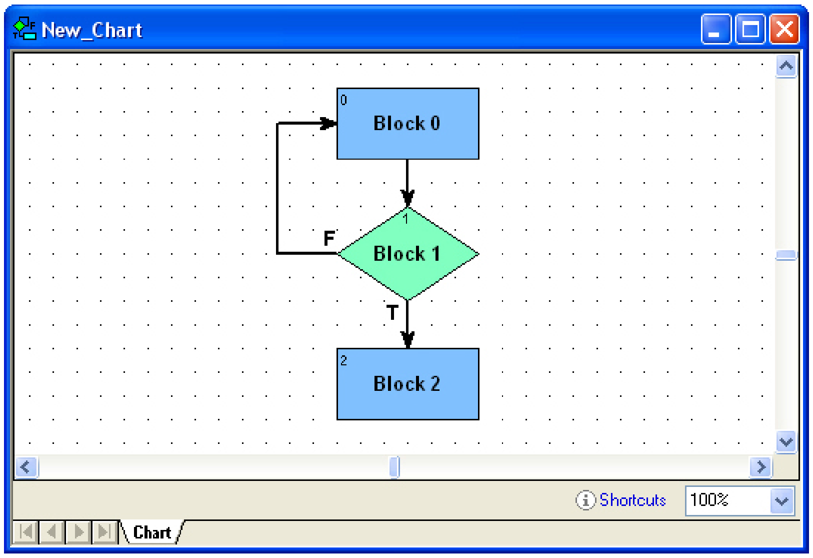


3. Indicate whether you are drawing the True connection or the False connection, and then click OK.
4. Click the destination block you chose (True or False).

   The connection is labeled T or F depending on its type.
5. Draw another connection to the second destination block.

   It is labeled the opposite exit type.

For example, the following figure shows the True and False connections from the condition block, Block 1. If the conditions in Block 1 are true, Block 2 is executed next. If the conditions in Block 1 are false, Block 0 is executed next:



Adding Text

One of the best places to put comments about a strategy is directly on its flowcharts. Start with the chart open and the strategy in Configure or Online mode.

1. To add text to a chart, click the Text tool.

   When you move the mouse onto the chart, a rectangle representing the text area appears.

2. Click the mouse button and type your comments.

   If you type in more text than the text frame holds, it expands in length.

3. When you have finished typing, click anywhere on the chart outside the text frame.

   The frame becomes invisible, and only the text appears. To change the size or shape of the text block, see "Resizing Blocks or Text Blocks" on page 248.

4. Click in another location to create another text frame, or release the tool by right-clicking in the chart or choosing another tool from the toolbar.

Editing Text

1. With the chart open and the strategy in Configure or Online mode, click the Select tool.
2. Double-click the text block you want to change.

   A blinking cursor appears at the beginning of the text.

3. Change the text as needed.

   You can use any standard Windows Ctrl key combinations when editing, including Ctrl+arrow keys and Ctrl+Home or End for navigation. You can also use Ctrl+X (cut), Ctrl+C (copy), and Ctrl+V (paste).

4. When you have finished changing text, click outside the text frame.

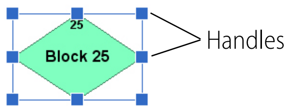
The text block stays the same width but changes length to accommodate additional or deleted text. To change the size or shape of the text block, see "Resizing Blocks or Text Blocks" on page 248.

Selecting Elements

Before you can manipulate most elements, you need to select them. Start with the chart open and the strategy in Configure or Online mode.

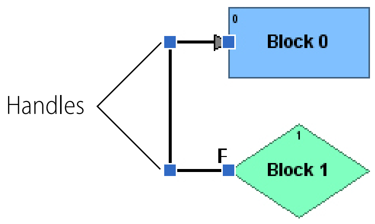
1. Click the Select tool.
2. To select an action, OptoScript, condition, continue, or text block, click the block.

   Handles appear around the block:



3. To select a connection, click it.

   Handles appear at the elbows and end points of the connection:



4. To select all connections entering or exiting a block, click the block, click the right mouse button, and choose Select Connections from the pop-up menu.
5. To select more than one element, do one of the following:
   - Select the first element, hold down the Shift key, and select additional elements.
   - Click and drag the mouse to draw a rectangle completely around the elements you want to select.
   - To select all items of the same type, right-click anywhere in the window and choose Select from the pop-up menu. From the sub-menu, choose the item type you want.

Moving Elements

1. With the chart open and the strategy in Configure or Online mode, click the Select tool.
2. To move any action, OptoScript, condition, continue, or text block, click it. Then click and hold the mouse button anywhere on the selected item except on its handles, and drag it to the position you want.

   You can also use the arrow keys on your keyboard to move a block in any direction. Note that when you move a block, any connections attached to it also move.
3. To move a connection, click it. Then click and drag any handle in any direction.

You can also move an end point from one block to another, as long as the result is a valid connection. A disallowed move is ignored.

4. To move several elements at once, select them, and then click and drag them.

   If elements end up stacked on top of each other, you may need to change their z-order before you can move them. Seethe following section.

Moving Elements in Front of or Behind Other Elements (Changing Z-Order)

If elements are stacked on top of each other, you can select only the one in front. To change their position (z-order), follow these steps:

1. Click the element to select it.
2. Right-click the element. From the pop-up menu, choose Z-order. From the sub-menu, choose the action you want to take:
   - Bring Forward—moves the element one position closer to the front.
   - Bring To Front—moves it all the way to the front.
   - Send Backward—moves the element one position closer to the back.
   - Send To Back—moves it all the way to the back.

Cutting, Copying, and Pasting Elements

You can cut, copy, and paste most chart or subroutine elements. Cut or copied elements are placed on the Windows Clipboard, and they can be pasted in the same chart or subroutine, in a different chart or subroutine, or in a different strategy.

A connection can be cut or copied, but it cannot be pasted unless its original source and destination blocks have also been pasted. Block 0 cannot be cut.

1. With the chart open and the strategy in Configure or Online mode, click the Select tool .
2. To cut or copy element(s), click them. Press Ctrl+X to cut or Ctrl+C to copy.

   You can also select the element(s) and then click Edit > Cut or Edit > Copy, or click the right mouse button and choose Cut or Copy from the pop-up menu.

3. To paste blocks, press Ctrl+V, click Edit > Paste, or right-click anywhere on a chart and then select Paste from the pop-up menu.

   Text blocks are pasted immediately. For action, condition, or continue blocks, a message appears asking if you want to keep the original name of the block being pasted.

   If you paste to a different strategy or to a subroutine, PAC Control checks the referenced variables to make sure they match. Variables that do not exist are created. Variables that exist but are different—for example, a table with the same name but a different table length—are noted in a log file that appears when the paste is complete.

Deleting Elements

1. Make sure the chart is open and the strategy is in Configure or Online mode.
2. Click the Select tool . Click the element(s) to select them.

   CAUTION*: Make sure you have selected the element you want. You cannot undo a deletion!*

3. Press Delete.

You can also select the element(s), right-click them, and select Delete from the pop-up menu. Block 0 cannot be deleted. If you delete a block, you'll be asked to verify the deletion.

Changing Element Color and Size

You can change the colors and sizes of blocks, connections, and text in your chart. To change one element (for example, the color of one block), use the steps in this section. To change more than one at a time, see "Changing the Appearance of Elements in a Chart Window" on page 239.

Start with the chart open and the strategy in Configure or Online mode.

Resizing Blocks or Text Blocks

1. Click the Select tool and click the block to select it.

2. Click one of the handles, then drag it in the direction you want. To resize horizontally and vertically at the same time, drag a corner handle.

Changing Block Colors

1. Click the Select tool and click the block to select it.

2. Right-click the block and choose Color from the pop-up menu.

3. Pick the color you want and click OK.

Changing Text

You can change the size, font, font style, or color of the text in any block.

1. Click the Select tool and click the block to select it.

2. Right-click the block and choose Font from the pop-up menu.

3. In the Font dialog box, make the changes you want. Click OK.

4. To change whether text appears at the left, the center, or the right of a block, select the block and click the right mouse button. From the pop-up menu, choose Justify; from the sub-menu, choose Left, Center, or Right.

Changing an Element Back to the Defaults

1. Select the item and click the right mouse button.

2. From the pop-up menu, choose Properties. From the sub-menu, choose Copy from Default.

To change defaults, see "Changing the Appearance of Elements in a Chart Window" on page 239.

Opening, Saving, and Closing Charts

Opening a Chart

Make sure the strategy is open. In the Strategy Tree, double-click the chart you want to open.

You can also open a chart by selecting Chart > Open, and then double-clicking the chart name in the Open Chart dialog box, which lists all charts that are not currently open.

If a chart is open but not visible on the screen, click the chart's name tab at the bottom of the window to make it visible.

Saving a Chart

1. Make sure the chart is open and is the active window.
2. From the Chart menu, choose Save.

Charts are automatically saved when you choose File > Save All. If you choose File > Save Strategy or click the Save Strategy icon on the toolbar, you can choose which charts to save in the Save Strategy dialog box.

Closing a Chart

To close a chart, click the close box in the upper-right corner of the chart's window (not the PAC Control window). You can also close a chart by pressing Ctrl+F4 when the chart window is active. If you have made changes to the chart, you are prompted to save them.

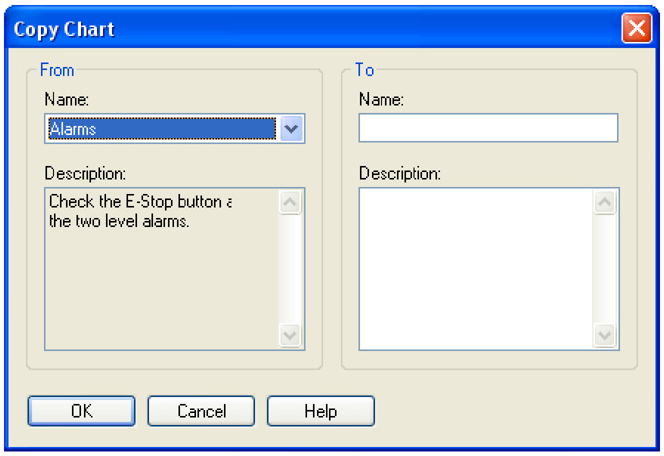
Copying, Renaming, and Deleting Charts

Copying a Chart

If an existing chart is similar to one you want to create, it is easier to copy it than to create a new one from scratch. To copy a chart in the same strategy, follow the steps in this section. To copy a chart to another strategy, see "Exporting and Importing Charts" on page 251.

1. With the strategy open and in Configure mode, select Chart > Copy.

   The Copy Chart dialog box appears.



From Name: Alarms

From Description: Check the E-Stop button a the two level alarms.

2. In the From field, choose the chart you want to copy from the drop-down list.

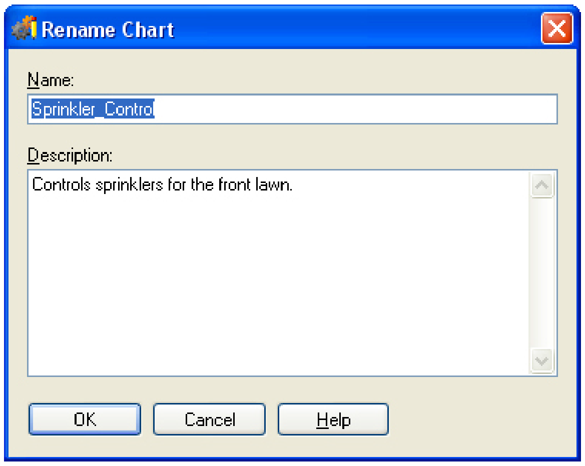
3. In the To field, enter a name for the new chart.

   The name must start with a letter and include only letters, numbers, or underscores. (Spaces are converted to underscores).
4. (Optional) Enter a description for the new chart.
5. Click OK.

   The new chart is created and appears as the active window on the screen.

Renaming a Chart

1. Make sure the strategy is in Configure mode and that the chart you want to rename is the active window.
2. From the Chart menu, choose Rename.



Sprinkler_Control

Controls sprinklers for the front lawn.

3. Enter a new name and description (optional). Click OK.

   The chart is renamed.

Deleting a Chart

You can delete any charts except for the Powerup chart. However, you cannot delete a chart if it is called or used by another chart in your strategy.

1. Make sure the strategy is open and in Configure mode.
2. In the Strategy Tree, right-click the name of the chart you want to delete and choose Delete from the pop-up menu. Or, if the chart is the active window, choose Chart > Delete.
3. At the confirmation message, make sure you are deleting the correct chart.

   ***CAUTION**: You cannot undo a deletion!*
4. Click Yes to delete the chart.

   The chart window disappears (if it was open), the chart is removed from the Strategy Tree, and the strategy is saved.

Printing Charts

You can print any flowchart. To print a chart as it appears on the screen, see "Printing Chart or Subroutine Graphics" on page 221. To print commands (instructions) for the chart, see "Viewing and Printing Strategy or Subroutine Commands" on page 223.

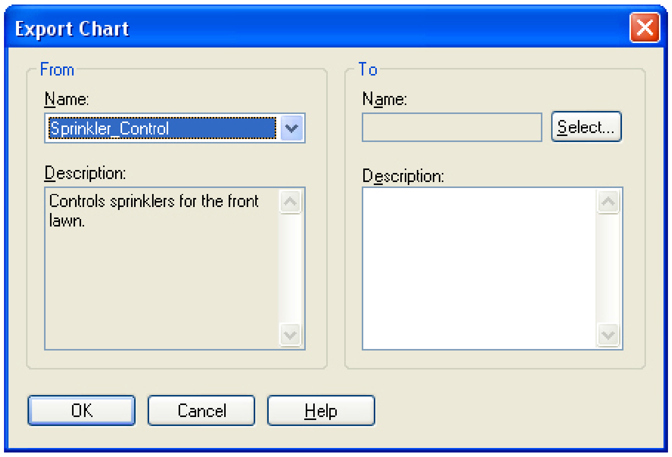
Exporting and Importing Charts

To copy a chart to another strategy, you must export it as a PAC Control chart export file (.cxf file) and then import it into the strategy where you want it.

Exporting a Chart

1. With the strategy open and in Configure or Online mode, choose Chart > Export.

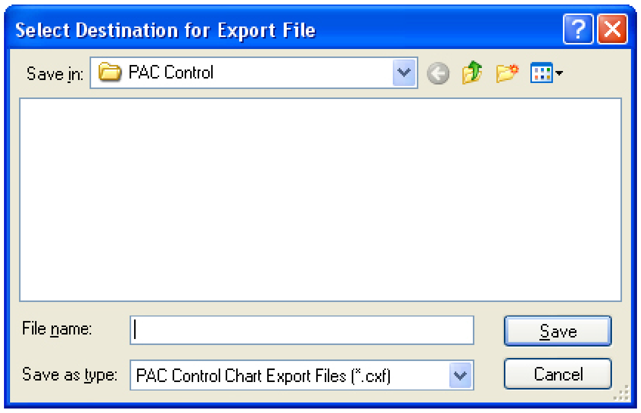
   The Export Chart dialog box appears.



Name: Sprinkler_Control

Description: Controls sprinklers for the front lawn.

2. In the From section of the dialog box, select the chart to be exported from the Name drop-down list.
3. In the To section of the dialog box, click Select.



Save in: PAC Control

Save as type: PAC Control Chart Export Files (*.cxf)

4. Navigate to where you want the exported chart file to be saved. In the File name field, enter a name for the exported chart. Click Save.

   You return to the Export Chart dialog box, which now shows the path and file name in the To section.

5. (Optional) Enter a description for the new chart.
6. Click OK.

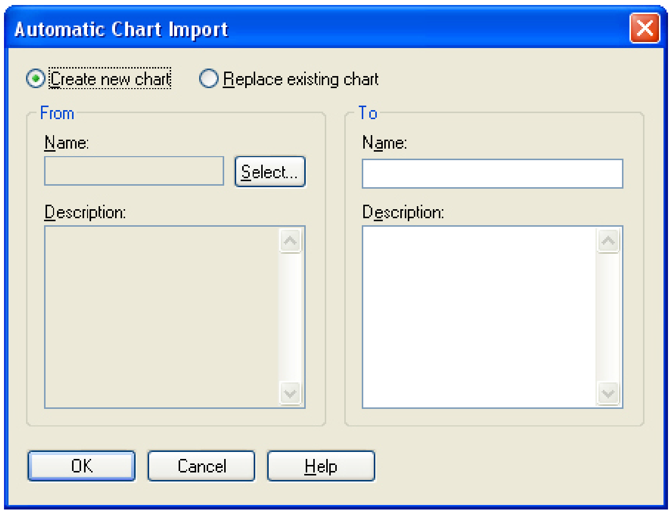
   The exported chart is saved. You can import it into any PAC Control strategy. See the next section for information on importing charts.

Importing a Chart

Use the Automatic Chart Import dialog box to import a chart previously exported. Also see "Exporting a Chart" on page 251.

1. With the strategy open and in Configure mode, choose Chart > Import.

   The Automatic Chart Import dialog box appears.



2. At the top of the dialog box, click Create new chart or Replace existing chart.

   ***CAUTION**: If you choose Replace existing chart, the old chart will be completely overwritten with the chart you are importing.*

3. Click Select. Navigate to the exported chart. Click OK.
4. In the To section of the dialog box, enter a name for the new chart.

   If you wish, enter a description. If you are replacing an existing chart, select it from the list. To copy the imported chart description, select the text in the left description field, copy it, and paste it in the right description field.

5. Click OK.

   The chart is imported. A Chart Import Report window shows you how the tags in the chart match with those already in the strategy. Any tags from the chart that do not already exist in the strategy are created and added.

9: Using Variables and Commands

OPTO 22

Introduction

This chapter discusses the seven types of variables used in PAC Control: numeric, string, pointer, numeric table, pointer table, string table, and communication handle variables.

This chapter also shows you how to use the commands, or instructions, in PAC Control and discusses the mechanics of adding commands to your strategy flowcharts. For command information to help you program your strategy effectively, see Chapter 10. To find out how to use commands in OptoScript code, see Chapter 11. For a list of all standard PAC Control commands and their OptoScript equivalents, see Appendix E.

In this Chapter

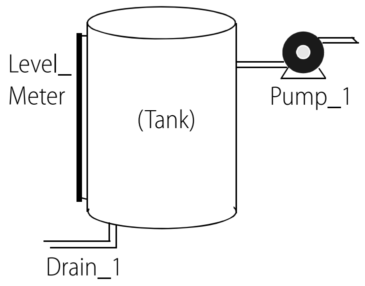
About Variables.......253
Variables in PAC Control.......255
Adding Variables.......257
Adding Tables.......260
Changing a Configured Variable.......265
Viewing Variables in Debug Mode.......266
Adding Commands.......270
Changing a Command.......274
Deleting a Command.......274
Cutting or Copying a Command.......275
Configuring a Continue Block.......276
Viewing and Printing Chart Instructions.......276

About Variables

As Chapter 2 mentions, variables store pieces of information in a strategy. You create a variable for each piece of information in your control process that must be acted upon. These pieces of information might include the name of a chart, the on or off state of a switch, communication parameters for a peer on the network, or a table that holds a series of numbers.

Each variable has a name and a value. You assign the variable's name in plain English, so you know what it is. The variable's value is the current information it represents. As a strategy runs, the variable's name remains the same, but its value may change. For example, the name of the variable Oven_Temperature stays the same, but its value (the temperature of the oven) may change several times while the strategy is running.

To illustrate variables, suppose you are regulating the amount of water in a tank. You must keep the tank filled beyond a minimum level, but you cannot let it get too full.



You've already configured the I/O points:

- Level_Meter is an analog input point that registers the quantity of water in the tank.
- Pump_1 is a digital output point that turns the pump on or off.
- Drain_1 is a digital output point that opens or closes the drain.

Next, you configure variables as places to hold information that these I/O points must work with:

- To avoid constantly polling Level_Meter to find out the quantity of water in the tank, you create a variable called Tank_Level_Reading in which to store the level. The input point Level_Meter is periodically checked and the value of Tank_Level_Reading updated.
- To establish the maximum and minimum levels, you create variables called Tank_Max_Level and Tank_Min_Level. The value in Tank_Level_Reading can be compared to the values in these two variables to determine whether the pump should be turned on or off, or the drain opened or closed, to maintain the proper level. (You could create constant values, called *literals*, for the minimum and maximum levels, but creating them as variables lets you change their values in Debug mode.)

Types of Data in a Variable

A variable stores one of six types of data: floating point, integer, timer, string, pointer, or communication handle. When you create the variable, you designate the type of data it contains. It is always best to choose the most appropriate data type for the information you are storing. PAC Control can store an integer in a floating point variable, but it has to convert the data first. Unnecessary data conversions take up processor time.

- **Numeric** data stores numbers and can be one of the following types:
  - A **floating point** (or *float*) is a numeric value that contains a decimal point, such as 3.14159, 1.0, or 1,234.2. A good example of a float variable is one that stores readings from an analog input such as a thermocouple. PAC Control uses IEEE single-precision floats with rounding errors of no more than one part per million.
  - An **integer** is a whole number with no fractional part. Examples of integer values are -1, 0, 1, 999, or -4,568. The state of a switch, for example, could be stored in an integer variable as 1 (on) or 0 (off).

    Most integers used in PAC Control are 32-bit signed integers, which can range from -2,147,483,648 to 2,147,483,647. These 32-bit integers should be used for general integer use, such as status variables, mathematics, and indexes. If you are using the SNAP-UP1-D64, SNAP-UP1-M64, or SNAP-ENET-D64 brains, which handle digital modules in all 16 module positions, you can use 64-bit integers. 64-bit integers address the entire I/O unit at once but are slower than integer 32 commands.

  - A **timer** stores elapsed time in units of seconds with resolution in milliseconds. Up Timers count up from zero, and Down Timers start from a value you set and count down to zero. Timers can range from 0.001 to 4.611686 x 10^{15}.

- A **string** stores text and any combination of ASCII characters, including control codes and extended characters. For instance, a string variable might be used to send information to a display for an operator to see. A string variable is also used to set parameters for peer-to-peer communication. When defining most string variables, you must specify the width of the string. The width is the maximum number of characters that the variable may hold.

  A string variable can contain numeric characters, but they no longer act as numbers. To use them in calculations, you must convert them into floating point or integer numbers. Conversely, a numeric value to be displayed on a screen must first be converted into a string.

- A **pointer** does not store the value of a variable; instead, it stores the memory address of a variable or some other PAC Control item, such as a chart or an I/O point. You can perform any operation on the pointer that you could perform on the object the pointer points to. Pointers are an advanced programming feature and are very powerful, but they also complicate programming and debugging.

- A **communication handle** stores information used to communicate with other entities, for example other devices on the network or files that store data. The communication handle's value is a string containing the parameters needed to make a connection with a specific entity. For outgoing Ethernet communication, for example, these parameters usually include the protocol, the IP address of the device you are communicating with, and the port number used on the device.

  After the initial connection is made, the communication handle is referenced during communication with the entity, and then used to close communication.

Variables in PAC Control

In PAC Control there are seven types of variables:

- numeric
- numeric table
- string
- string table
- pointer
- pointer table
- communication handle

Numeric, string, and pointer variables contain individual pieces of data. Numeric table, string table, and pointer table variables contain several pieces of related data in the form of a table. Communication handle variables contain parameters used by PAC Control for communicating with other devices and files.

Table Variables

In a table variable, the variable name represents a group of values, not just one. Table variables are one-dimensional arrays, which means that each table is like a numbered list of values. You refer to each item in the list by its number, or *index*. Indexes start at 0, not at 1. Here are two table examples:

In a float table, values are floating point numbers:

Index	Value
0	82.0
1	86.1
2	85.0
3	74.8
4	72.3
5	72.7

In a string table, values are strings:

Index	Value
0	Maria
1	Tom
2	Siu
3	Andre

When you define a table, you must specify its length, which is how many values the table can store. The length of a table is NOT the value of the last index. Since indexes start at 0, a table with a length of 100 contains indexes 0 through 99. Table length is limited only by the amount of memory in the control engine. (For more information on the available memory, see "If You Have Memory Problems" on page 415.)

Numeric tables store either integer values or floating point numbers (but not both in the same table). String tables store strings. Because pointers can point to any data type (for example, an I/O point, a chart, or even another table), pointer tables can store an assortment of data types.

Persistent Data

Most variables can be either *global* or *persistent* in scope. Global variables are set to an initial value (which you specify during configuration) either whenever the strategy is run or whenever it is downloaded.

Persistent variables, however, are initialized only when the strategy is first downloaded. Numeric variables (and tables) are initially set to 0. String variables (and tables), pointer variables (and tables), and communication handles are initialized as blank, or empty. The key point about a persistent variable is that its value is saved in the controller's memory; it does not get initialized when the strategy is run, stopped, or started, and it does not get initialized if the strategy is modified and downloaded again. A persistent variable's value remains the same until one of the following events occurs:

- A strategy with a different name is downloaded.
- The RAM memory on the controller is cleared.
- A new firmware kernel is downloaded to the controller.
- A strategy download is canceled during the download process.

- The persistent object is changed as follows:
  - A persistent table's length (integer, 64-bit integer, float, and string) was modified.
  - A persistent string variable's length was modified.
  - A persistent string table element's string length was modified.
  - A persistent variable's type has changed. For example, a persistent float variable is deleted and a new persistent integer variable is created with the same name.

Persistent data can be very useful in certain situations. For example, suppose you have a PID setpoint that is fine-tuned as your process runs. If the setpoint is configured as persistent, its value won't be lost if you must re-download the strategy after making a program change.

Pointer variables, pointer tables, and timers are not allowed to be persistent; but all other variables and tables can be. Persistent variables cannot be configured on-the-fly.

Literals

A literal is used like a variable, but it is constant data that never changes. A literal has no variable name, only a fixed value, which may be a floating point number, an integer, or a string.

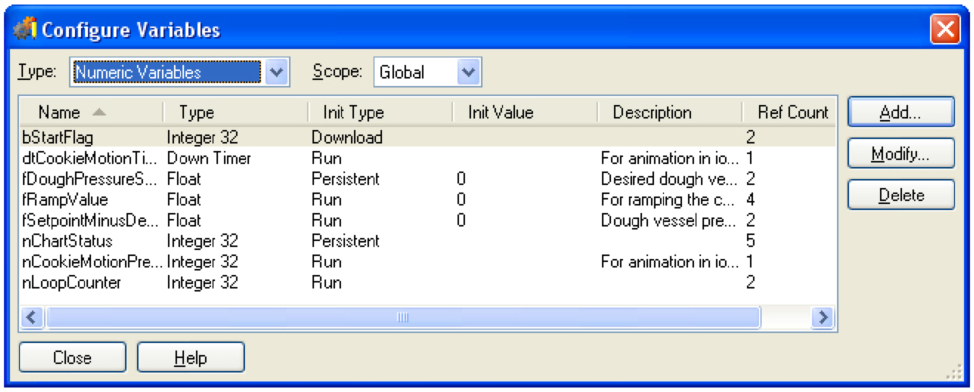
If you are using subroutines, see the information on literals in "Data Types for Subroutines" on page 400.

Adding Variables

This section includes steps for adding numeric, string, pointer, and communication handle variables. (For numeric, string, and pointer tables, see "Adding Tables" on page 260.)

1. With the strategy or subroutine open in Configure mode, choose Configure > Variables.

The Configure Variables dialog box opens.



Type: Numeric Variables
Scope: Global

Name	Type	Init Type	Init Value	Description	Ref Count
bStartFlag	Integer 32	Download			2
dtCookieMotionTi...	Down Timer	Run		For animation in io...	1
fDoughPressureS...	Float	Persistent	0	Desired dough ve...	2
fRampValue	Float	Run	0	For ramping the c...	4
fSetpointMinusDe...	Float	Run	0	Dough vessel pre...	2
nChartStatus	Integer 32	Persistent			5
nCookieMotionPre...	Integer 32	Run		For animation in io...	1
nLoopCounter	Integer 32	Run			2

This dialog box lists all the variables in the strategy or subroutine that are of the type shown in the Type field. You can sort the data in this dialog by clicking any one of the column header buttons.

2. In the Type drop-down list, choose the type of variable you want to configure.

3. If you are adding the variable to a subroutine, select Subroutine in the Scope drop-down list.

4. To add a new variable, click Add.

   The Add Variable dialog box appears.

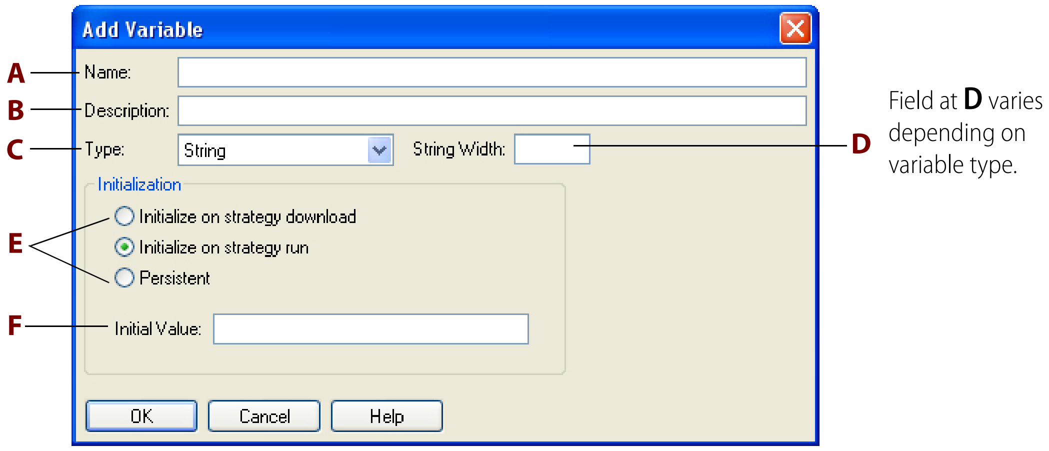
5. Complete the fields as described in "Add Variable Dialog Box" below.

6. Click OK.

   The Add Variable dialog box closes and the new variable appears in the Configure Variables dialog box.

Add Variable Dialog Box

This dialog box varies depending on the type of variable.



The figure above shows the Add Variable dialog box as it appears for string variables. Fields are slightly different for other variables.

A—*Name.* Enter a name for the variable. The name must start with a letter and may contain letters, numbers, and underscores. (Spaces are converted to underscores.)

B—*Description.* (Optional) Enter a description of the variable.

C—*Type.* In the Type drop-down list, select the type of data for the variable. Possible types are shown in the following table. For more information, see "Types of Data in a Variable" on page 254.

Variable	Possible Data Types
Numeric	Integer 32, integer 64, floating point, up or down timer
Numeric table	Integer 32, integer 64, or floating point
String	String
String table	String
Pointer	Pointer to any data type
Pointer table	Pointer to any data type, and the data type may change over time
Communication handle	Communication handle

D—*String Width.* This field varies depending on the variable type.

(**Table variables** only) In the Table Length field, enter an integer between 1 and 1,000,000 representing the number of elements in the table. The greater the number, the more memory required to store the table.

(**String variables** and **string tables** only) In the String Width field, enter the maximum number of characters permitted in the string. The number must be an integer between one and 1024. The greater the number, the more memory required to store the string.

(**Pointer variables** only) From the Pointer to Type drop-down list, select the type the pointer points to. Note that void pointers are not allowed; a pointer must point to a specific type. Also note that you cannot point a pointer to another pointer; PAC Control has only one level of indirection for pointers. If you try to point to a pointer, PAC Control assigns to the new pointer the address of the object being pointed to.

E—*Initialization.* (For all variables except up timers and down timers)

Select one of the following:

Initialize on Strategy Run: Sets the variable to the initial value (F) each time the strategy is run (either manually from the debugger or automatically via the autorun flag).

Initialize on Strategy Download: Sets the variable to the initial value (F) only when a strategy is downloaded. The variable retains its current value when the strategy is stopped and then run again, either through the debugger or autorun. It also retains its value if power is cycled. Note that this choice means the variable is stored in battery-backed RAM, which is limited in size. To keep a variable's current value through both power cycles and strategy download, make the variable a persistent variable (see the Persistent option below and "Persistent Data" on page 256). See "If You Have Memory Problems" on page 415.

Persistent: Data in the variable will be persistent. Pointer variables and timers cannot be persistent. For more information, see "Persistent Data" on page 256.

The following table shows how your choices about variable initialization and persistence affect what happens to variables (applies to all variables and tables except up timers and down timers).

Variable	What happens to the variable's value when...?		
	Strategy stops, then starts (through debugger or autorun)	**Power is cycled**	**Same strategy is downloaded**
Variable initialized on strategy run (default)	Set to initial value	Set to initial value	Set to initial value
Variable initialized on strategy download	Retains current value	Retains current value	Set to initial value
Persistent variable	Retains current value	Retains current value	Retains current value

F—*Initial Value.* (For all variables except pointers, up timers, and down timers) Enter the value to which the variable is to be set initially. You can use either decimal or hexadecimal form. If you leave this field blank, the initial value is set to zero.

(For **pointer variables** only) When you have selected the Pointer to Type, the drop-down list in this field shows all the valid objects for that type that are currently defined in your strategy. Choose one or leave the initial value as NULL. A NULL value means that the pointer is created but does not initially point to anything.

(For **communication handles** only) Enter a string containing communication parameters in the correct format for the type of communication handle you are using. The type (for example, `tcp`, `ftp`, `file`) must be in lowercase letters, and parameters are separated by colons and commas according to the format required. See "Communication Commands" on page 280 for information and examples.

- To communicate with a serial module, use the IP address of the brain the module is attached to, and use the serial module's port number according to its position on the rack, for example: tcp:10.192.55.185:22502. For port numbers, see "Establishing an Ethernet Connection" in form 1191, the *SNAP Serial Communication Module User's Guide*. See also, form 1940, *TCP and UDP Port Assignments*.
- To communicate with another SNAP PAC brain, be aware of port numbers that are reserved for a specific protocol. For more information, see form 1940, *TCP and UDP Port Assignments*.
- For other peers on the Ethernet network, be aware of port numbers they may use for specific purposes. Ports 22000 and 22001 are reserved for the control engine. For a list of standard Ethernet port numbers, refer to http://www.iana.org/assignments/port-numbers. See also, Opto 22 form 1940, *TCP and UDP Port Assignments.*
- If you are using the Accept Incoming Communication command to listen for communication requests, leave out the IP address and use the following format: tcp:port

For tables, each value in the table is set to the initial value. If you need to set individual table elements to differing values initially, you can do so in the Powerup chart. If you need to use an initialization file to set values on strategy download, contact Opto 22 Product Support.

Adding Tables

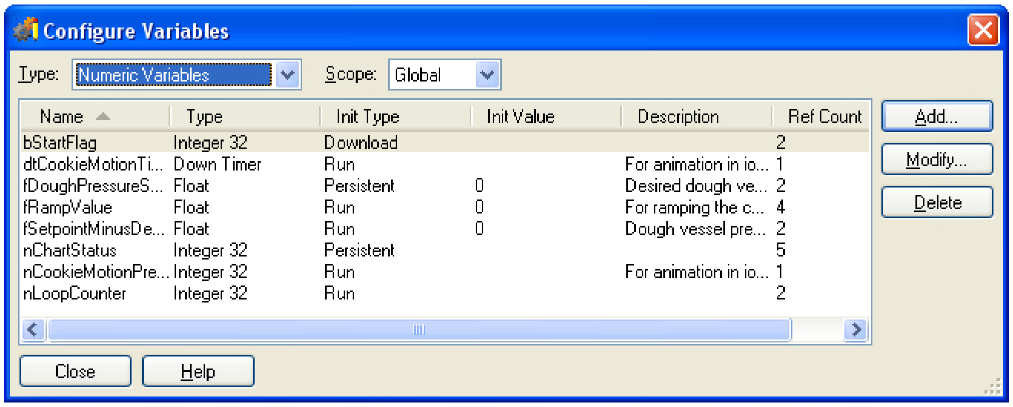
See the following topics to add numeric, string, and pointer tables:

- "Adding Table Variables" (below)
- "Setting Initial Values in Variables and Tables During Strategy Download" on page 261

Adding Table Variables

1. With the strategy or subroutine open in Configure mode, choose Configure > Variables.

   The Configure Variables dialog box opens, listing all variables of one type in the strategy.

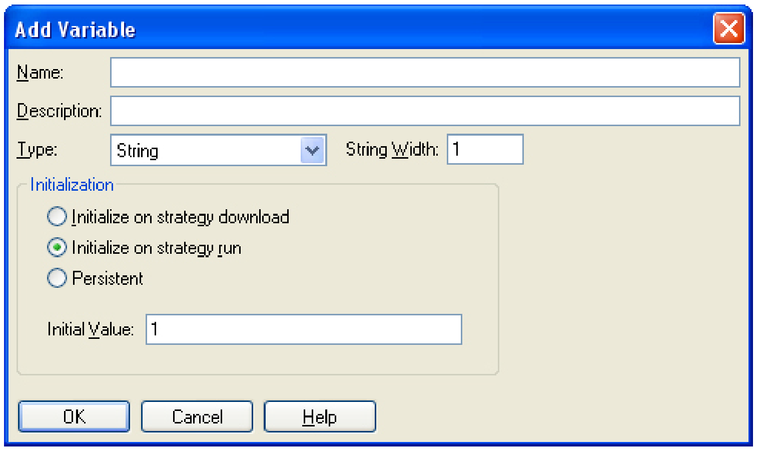


Type: Numeric Variables   Scope: Global

Name	Type	Init Type	Init Value	Description	Ref Count
bStartFlag	Integer 32	Download			2
dtCookieMotionTi...	Down Timer	Run		For animation in io...	1
fDoughPressureS...	Float	Persistent	0	Desired dough ve...	2
fRampValue	Float	Run	0	For ramping the c...	4
fSetpointMinusDe...	Float	Run	0	Dough vessel pre...	2
nChartStatus	Integer 32	Persistent			5
nCookieMotionPre...	Integer 32	Run		For animation in io...	1
nLoopCounter	Integer 32	Run			2

2. In the Type drop-down list, choose the type of table variable you want to add.
3. If you are adding the table variable to a subroutine, select Subroutine in the Scope drop-down list.
4. To add a new table variable, click Add.

   The Add Variable dialog box appears.



Add Variable

Name:

Description:

Type: String

String Width: 1

Initialization

( ) Initialize on strategy download

(●) Initialize on strategy run

( ) Persistent

Initial Value: 1

   The figure above shows the dialog box as it appears for string tables. Fields are slightly different for other table variables.

5. Complete the fields as described in "Add Variable Dialog Box" on page 258.

   NOTE: If you need to set individual table elements to differing values initially, you can do so in the Powerup chart. If you need to use an initialization file to initialize the values on strategy download, see the next section, "Setting Initial Values in Variables and Tables During Strategy Download."

6. To have data in the table be persistent, select Persistent.

   Pointer tables cannot be persistent. For more information, see "Persistent Data" on page 256.

7. Click OK.

   The Add Variable dialog box closes and the new table variable appears in the Configure Variables dialog box.

Setting Initial Values in Variables and Tables During Strategy Download

When you are adding table variables in PAC Control, you can set all table elements to one initial value in the Add Variables dialog box. If you want to set each individual table element to its own value, however, you need to create an initialization file and download it with your PAC Control strategy.

In addition to setting initial values for table elements, sometimes it is easier to initialize all variables during strategy download using an initialization file.

This section shows you how to create an initialization file and download it with your strategy.

Creating the Initialization File

A sample initialization file, Init.txt, is copied to the local hard drive when you install PAC Control. The file's location is C:\Users\Public\Documents\Opto 22\PAC Project <version number>\Control Basic Examples <or Control Pro Examples, if you have PAC Control Professional>\InitVariables.

You can open this file with a text editor to see the proper syntax for each initialization entry, and then modify it as necessary for your strategy.

***IMPORTANT**: Every initialization file should include the following line near the top:*

```
\ ""DOWNLOAD_COMPRESSION_OFF
```

Make sure to enter a backslash and a space at the beginning of the line. Download compression removes the newlines necessary for string initialization. This line tells PAC Control to turn off download compression so strings can be initialized.

***IMPORTANT**: Each file must end with a carriage return so that the last line in the file is blank, and each line within the file must end with a carriage return. If you add comments to the file, they must be preceded by a backslash (\) and a space.*

Text Examples

NOTE: Variable names are case-sensitive and can include both upper- and lower-case letters.

Integer Example. To set an initial value of 123 for the variable INTEGER_VARIABLE, you would include the following text in the initialization file:

```
123 ^INTEGER_VARIABLE @!
```

Float Example. Floating point values must contain a decimal point. To set an initial value of 456 for the variable FLOAT_VARIABLE, you would include the following text (with the decimal point and zero) in the initialization file:

```
456.0 ^FLOAT_VARIABLE @!
```

String Example. To set an initial value of "String Variable Test String" for STRING_VARIABLE, you would include the following text in the initialization file. The spaces and newline are very important:

```
*STRING_VARIABLE $INN
String Variable Test String
```

Communication Handle Example. To add the parameters of a communication handle named COMM_Handle, you would include the following text in the initialization file. *The spaces and newline are very important.*

```
*COMM_Handle $INN
tcp:10.0.0.1:1000
```

Pointer Example. To have the pointer PTR_POINTER_VARIABLE point initially to INTEGER_VARIABLE, you would include the following text in the initialization file:

```
^INTEGER_VARIABLE MoveToPointer PTR_POINTER_VARIABLE
```

Integer 32 Table Example. To set initial values of 10, 20, 30, 40, and 50 for elements zero through four of an integer table named My_Int_32_Table, include the following text:

```
10 0 }My_Int_32_Table TABLE!
20 1 }My_Int_32_Table TABLE!
30 2 }My_Int_32_Table TABLE!
```

```
40 3 }My_Int_32_Table TABLE!
50 4 }My_Int_32_Table TABLE!
```

Integer 64 Table Example. To set initial values of 10, 20, 30, 40, and 50 for elements zero through four of an integer table named My_Int_64_Table, include the following text:

```
10.. 0 }My_Int_64_Table 2TABLE!
20.. 1 }My_Int_64_Table 2TABLE!
30.. 2 }My_Int_64_Table 2TABLE!
40.. 3 }My_Int_64_Table 2TABLE!
50.. 4 }My_Int_64_Table 2TABLE!
```

Float Table Example. For a float table, the initial values must include a decimal point. To set initial values of 1.1, 2.2, 3.3, 4.4, and 5.5 for elements zero through four of a float table named My_Float_Table, include the following text:

```
1.1 0 }My_Float_Table TABLE!
2.2 1 }My_Float_Table TABLE!
3.3 2 }My_Float_Table TABLE!
4.4 3 }My_Float_Table TABLE!
5.5 4 }My_Float_Table TABLE!
```

String Table Example. To set initial values of “zero”, “one”, “two”, “three”, and “four” for elements 0–4 of a string table named My_String_Table, include the following text. Make sure you turn off download compression and use new lines as shown:

```
0 {My_String_Table $TABLE@ $INN
zero
1 {My_String_Table $TABLE@ $INN
one
2 {My_String_Table $TABLE@ $INN
two
3 {My_String_Table $TABLE@ $INN
three
4 {My_String_Table $TABLE@ $INN
four
```

Pointer Table Example. Each index in a pointer table points to another item within the strategy, for example an I/O point, a variable, or a chart. Setting initial values for pointer tables means designating the items the pointer table initially points to. For example, you would include the following text to have a pointer table named My_Ptr_Table initially point to Oven_Temperature (a variable), Alarm_Handler (a chart), Thermocouple (an analog input), Fuel_Pump (an analog output), and Fan_1 (a digital output):

```
^Oven_Temperature 0 PTBL_My_Ptr_Table TABLE!
&Alarm_Handler 1 PTBL_My_Ptr_Table TABLE!
~Thermocouple 2 PTBL_My_Ptr_Table TABLE!
~Fuel_Pump 3 PTBL_My_Ptr_Table TABLE!
~Fan_1 4 PTBL_My_Ptr_Table TABLE!
```

Special Characters in the Initialization File. Note that the initial character on each line of the initialization file is a special character that identifies the object type. Possible characters include the following:

Character	Object Type
^	float, integer, or timer

Character	Object Type
*	string
PTR_	pointer variable
~	I/O point
&	chart
}	float table or integer table
{	string table
PTBL_	pointer table
%	I/O unit

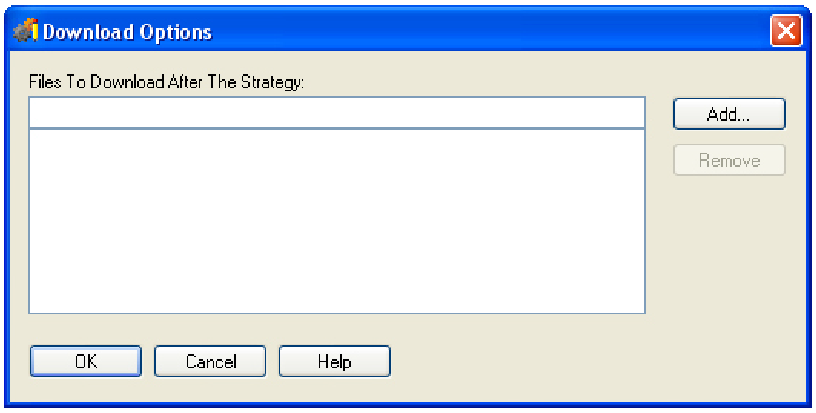
Saving the Initialization File. When you have finished modifying the file, save it as a text file (.txt file extension) to your strategy directory. You can name the file anything you like. (You can also save it to any directory you like, but it is good practice to save each initialization file to the directory of the strategy that references it.)

Downloading the Initialization File

To use the file, you need to download it with your strategy.

1. With the strategy open in Configure or Online mode, choose Configure > Control Engines, or double-click the control engine's name.
2. Highlight the control engine and click the Download Options button.

   The Download Options dialog box appears. The initialization file must be downloaded immediately after your strategy is downloaded.



3. Click Add.
4. Navigate to the initialization file you created. Double-click the file name.

   The file appears in the Download Options dialog box.
5. Click OK.

   The initialization file sets values for the variables and table elements immediately after your next full strategy download.

***IMPORTANT**: When you create each variable in the strategy, you must check "Initialize on Strategy Download" in the Add Variable dialog box.*

Since download options are specific to each control engine, make sure you set the initialization file as a download option for every control engine on which the strategy will run. Because control engines have separate initialization files, you can use the same master strategy for two or more control engines and configure differences in variables by customizing the initialization files.

Starting the Strategy Automatically

If you want your strategy to start automatically as soon as the strategy finishes downloading to the controller using PAC Control Debug mode, modify the Init.txt file as follows:

1. Add the following to the end of the Init.txt file:

   `_RUN`

2. Add 1 or more blank lines after the `_RUN` line

   This is required; it won't work without at least 1 blank line at the end of the file.

If you are not using the other features of the Init.txt file to initialize variables after strategy download, you can delete the rest of the content of the file.

Changing a Configured Variable

You can change a variable, but you cannot change a variable's type.

1. To change a configured variable, make sure the strategy or subroutine is open and in Configure or Online mode.
2. If you are working in a strategy, on the Strategy Tree, expand the Variables folder until you see the name of the variable you want to change. Double-click its name to open the Edit Variable dialog box.
3. If you are working in a subroutine, choose Configure > Variables. In the Configure Variables dialog box, double-click the name of the variable you want to change.
4. In the Edit Variable dialog box, make the necessary changes and click OK.

   If you need help, see "Adding Variables" on page 257 or "Adding Tables" on page 260.

You can also change a variable from the Configure Variables dialog box by double-clicking the variable's name or by highlighting it and clicking Modify.

Deleting a Variable

You cannot delete a variable that is referenced within the strategy or subroutine. *Be careful when deleting variables, since you cannot undo a deletion.*

1. Make sure the strategy is open and in Configure mode.
2. On the Strategy Tree, expand the Variables folder until you see the name of the variable you want to change. Right-click the name and choose Delete from the pop-up menu.

   The variable is deleted.

You can also delete a variable from the Configure Variables dialog box by highlighting the variable's name and clicking Delete.

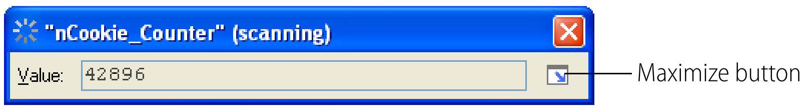
Viewing Variables in Debug Mode

While the strategy is running in Debug mode, you can view its variables and modify the value of a variable or of entries in a table. You can also view several variables at once—as well as other strategy elements—by putting them into a watch window.

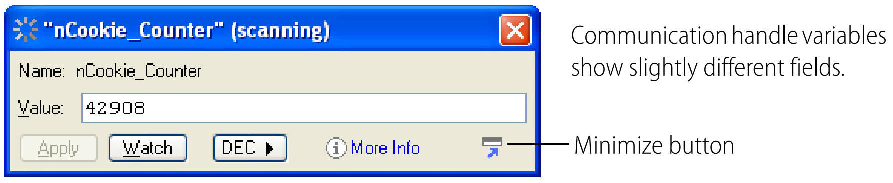
Viewing Numeric, String, and Communication Handle Variables

1. Make sure the strategy is running in Debug mode. On the Strategy Tree, double-click the variable you want to view.

The Inspect Variables dialog box opens. The animated icon at the upper left assures you that the data is fresh. The title bar includes the name of the variable and indicates whether scanning is occurring.



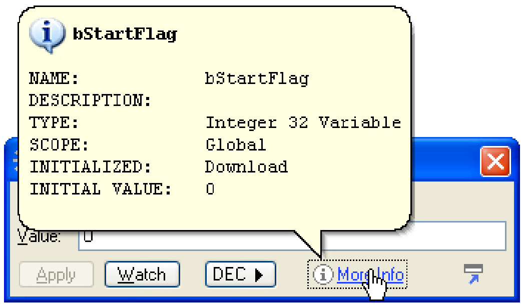
2. To view more options, click the Maximize button.


"nCookie_Counter" (scanning)
Name: nCookie_Counter
Value: 42908
Apply
Watch
DEC
More Info
Communication handle variables show slightly different fields.
Minimize button

Scanning stops whenever you click a changeable field. It resumes once you click Apply, another button, or an unchangeable field. If scanning resumes before you click Apply, any changes you made are lost.

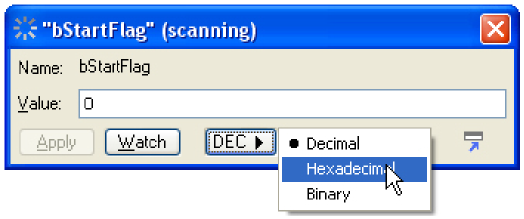
If you do not want to change the value of the variable, you can click the Minimize button to shrink the dialog box back to its original size.

3. To view the configuration information about the variable, click More Info.



```
bStartFlag

NAME:            bStartFlag
DESCRIPTION:
TYPE:            Integer 32 Variable
SCOPE:           Global
INITIALIZED:     Download
INITIAL VALUE:   0
```

4. To select the display format, click the button to the right of the Watch button.



A numeric variable can appear as decimal, hexadecimal, or binary. A string variable can appear as ASCII or hexadecimal.

5. To change the value of the variable, type the new value in the Value field and click Apply.

The field turns magenta until you click Apply.

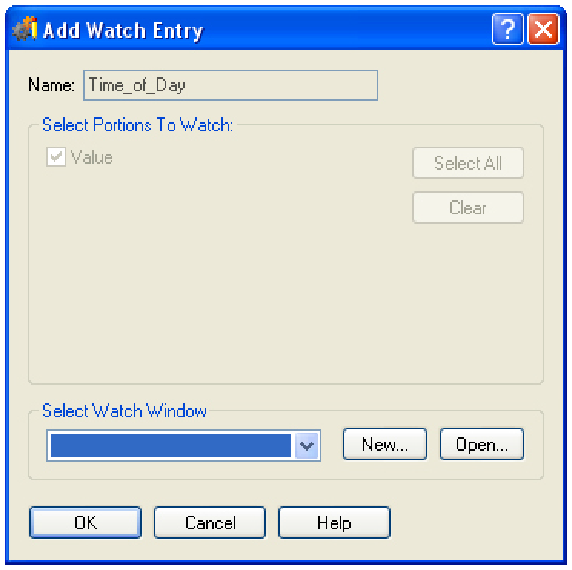
For a string variable, if your change lengthens the string beyond its maximum width, the string is truncated to fit.

For a communication handle variable, changing the value of the variable here has the same effect as using a Set Communication Handle Variable command. If the communication handle is currently open, the value will be changed *but will not affect the connection*.

6. To monitor the variable in a watch window, click Watch.

If you have only one watch window and it is already open, the variable appears immediately in the window for monitoring.

Otherwise, the Add Watch Entry dialog box appears.



7. Check the items you want to watch.

Items to watch vary depending on the variable type.

8. In the Add Watch Entry dialog box, do one of the following:

– If the watch window you want to use to monitor the variable is open, choose it from the Select Watch Window drop-down list.

- If the watch window you want is not open, click Open. Navigate to it and double-click it to open it.
- If you want to monitor the variable in a new watch window, click New. (For help, see "Creating a Watch Window" on page 193.)

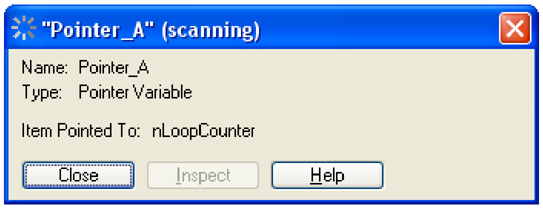
9. When the Add Watch Entry dialog box shows the correct items to watch and the watch window you want, click OK.

The variable appears in the watch window.

Viewing Pointer Variables

1. Make sure the strategy is running in Debug mode. On the Strategy Tree, double-click the pointer variable you want to view.

The View Pointer dialog box appears, showing the pointer's name, type, and item pointed to.


"Pointer_A" (scanning)

Name: Pointer_A
Type: Pointer Variable

Item Pointed To: nLoopCounter

2. To view the status or value of the item pointed to, click the Inspect icon.

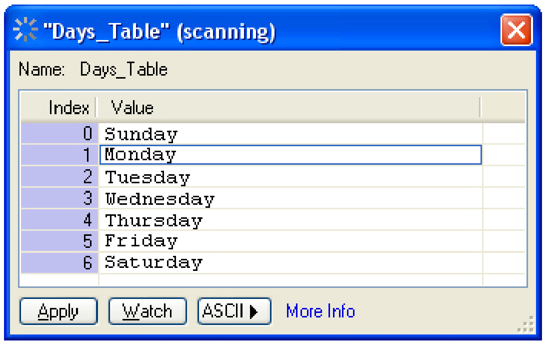
If you need help, follow the steps in "Viewing Numeric, String, and Communication Handle Variables" on page 266.

Viewing Numeric and String Tables

1. Make sure the strategy is running in Debug mode. On the Strategy Tree, double-click the table variable you want to view.

The View Table dialog box appears, showing the table's name, length (maximum number of entries), width for a string table, initialization method, and security level. It also lists the index and value of each table entry. The title bar includes the name of the variable and indicates whether scanning is occurring.

Here's an example for a string table.

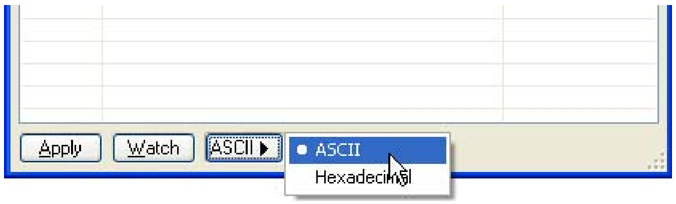

"Days_Table" (scanning)

Name: Days_Table

Index	Value
0	Sunday
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday

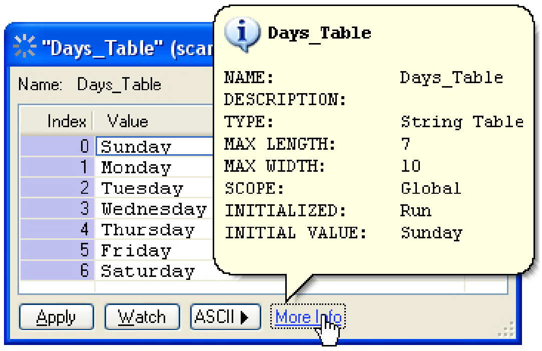
Scanning for an individual table element stops whenever you select an element in the table. It resumes for that element if no changes are made and another table element is selected, or when you click Apply. A magenta background indicates that scanning is stopped.

2. To change the format of the elements, click the button to the right of the Watch button.



For string tables, you can select either ASCII or Hexadecimal. For numeric tables, you can select Decimal, Hexadecimal, or Binary.

3. To view the configuration information, click More Info.



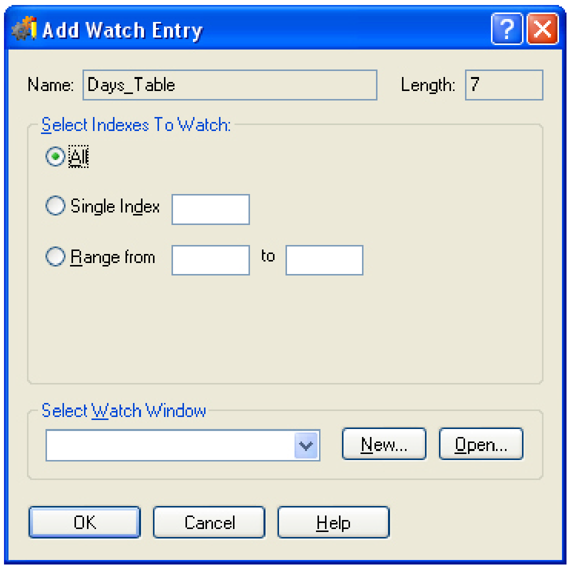
Index	Value
0	Sunday
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday

```
Days_Table

NAME:            Days_Table
DESCRIPTION:
TYPE:            String Table
MAX LENGTH:      7
MAX WIDTH:       10
SCOPE:           Global
INITIALIZED:     Run
INITIAL VALUE:   Sunday
```

4. To change a table entry, click its index number, highlight the value, and type in a new value. Click Apply.

5. To monitor the table in a watch window, click Watch.

The Add Watch Entry dialog box appears.



Name: Days_Table
Length: 7
[✓ All]

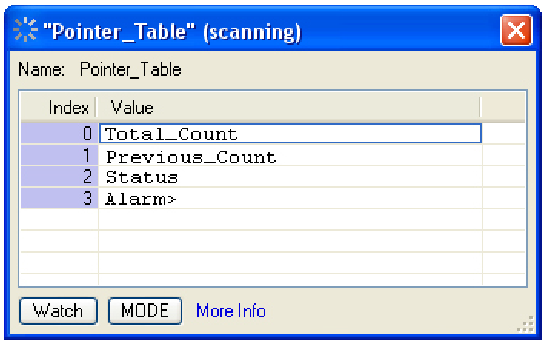
6. In the Add Watch Entry dialog box, do one of the following:
   – If the watch window you want to use to monitor the table variable is open, choose it from the Select Watch window drop-down list.
   – If the watch window you want is not open, click Open. Navigate to it and double-click it to open it.
   – If you want to monitor the variable in a new watch window, click New. (For help, see "Creating a Watch Window" on page 193.)
7. Select the indexes you want to watch.
8. When the Add Watch Entry dialog box shows the correct items to watch and the watch window you want, click OK.

   The table variable appears in the watch window.

Viewing Pointer Tables

1. Make sure the strategy is running in Debug mode. On the Strategy Tree, double-click the pointer table you want to view.

   The View Table dialog box appears, showing the pointer table's name, length, and the items pointed to. You cannot change a pointer table entry in this dialog box.



Name: Pointer_Table

Index	Value
0	Total_Count
1	Previous_Count
2	Status
3	Alarm>

2. To view the status or value of the item pointed to, highlight it in the table and click More Info.

   If you need help, follow the steps in "Viewing Numeric and String Tables" on page 268.

Adding Commands

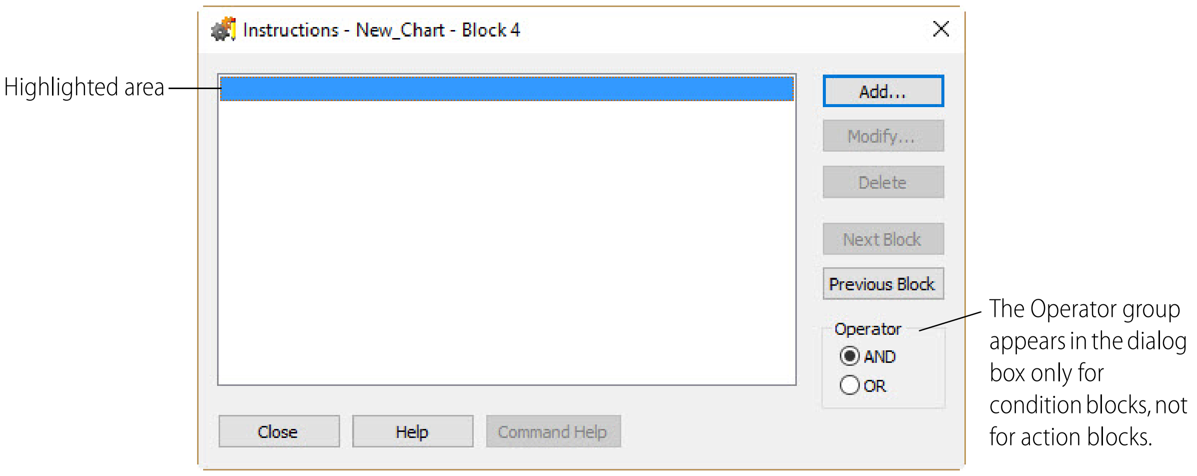
To make a block in a strategy flowchart do the work it's intended to do, you add one or more commands. Commands use the I/O points and variables you've already configured, as well as other elements in your strategy. A command, for example, might turn on a digital point, move a value to a variable, or check to see whether a chart is running. PAC Control contains more than 500 commands you can use. A command in PAC Control is often called an instruction.

You can add commands to action blocks, condition blocks, and OptoScript blocks. Continue blocks just move flowchart logic to another block. (See "Configuring a Continue Block" on page 276.)

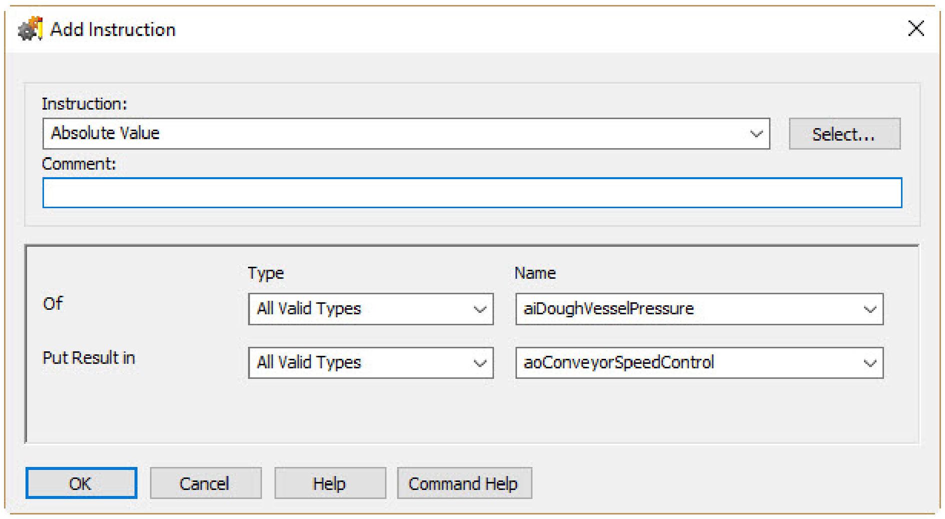
To add commands to an action or condition block, follow the steps below. (To add commands to an OptoScript block, see "OptoScript Functions and Commands" on page 378 and "Using the OptoScript Editor" on page 392.)

1. With the strategy open in Configure or Online mode and the flowchart open, double-click the block to which you want to add a command.

   The Instructions dialog box appears. The highlighted area shows where the new command will be added.


Instructions - New_Chart - Block 4
Highlighted area
Add...
Modify...
Delete
Next Block
Previous Block
Operator
AND
OR
Close
Help
Command Help
The Operator group appears in the dialog box only for condition blocks, not for action blocks.

2. Click Add to open the Add Instruction dialog box.



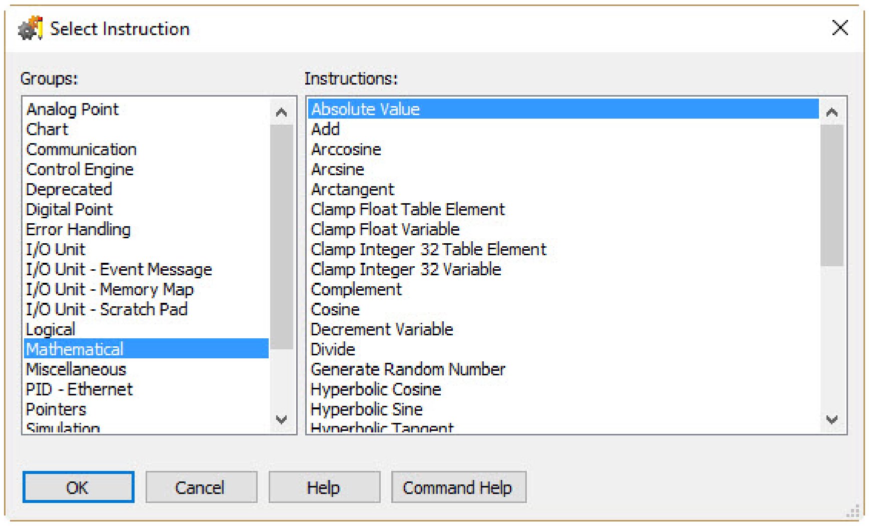
Instruction: Absolute Value

Comment:

	Type	Name
Of	All Valid Types	aiDoughVesselPressure
Put Result in	All Valid Types	aoConveyorSpeedControl

3. If you know the command you want, enter its name in the Instruction field by typing it in or by choosing it from the drop-down list. Skip to step 7.

If you don't know the command name, click Select.



4. Click the name of a command group in the left column to display all the commands in that group. In the right column, click the command you want.

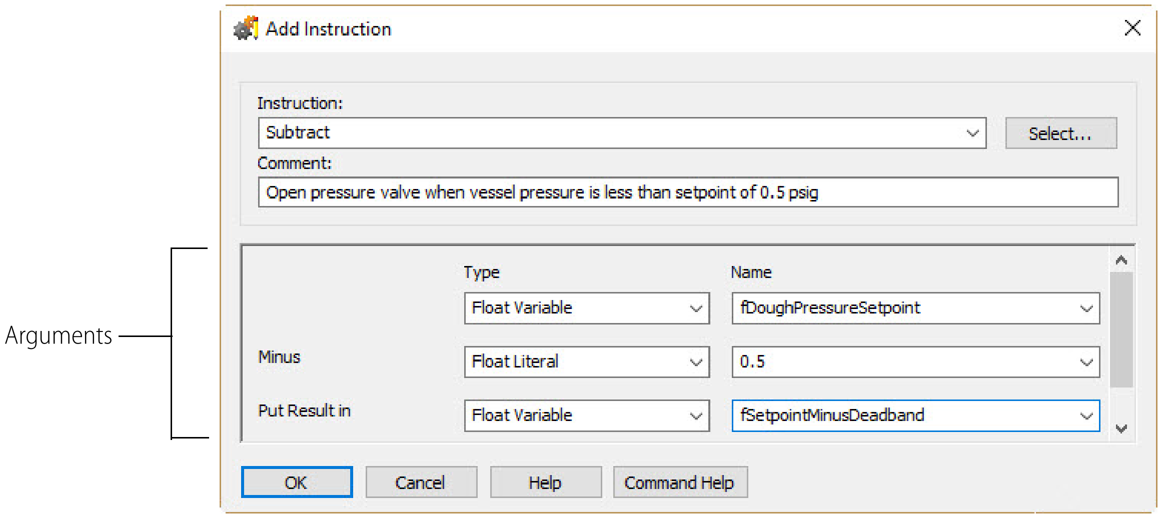
   For more information on any command, click the command name and click Command Help to open help. You can also look up the command in the *PAC Control Command Reference*.

5. When the command you want is highlighted, click OK to return to the Add Instruction dialog box.

6. (Optional) Enter a comment about the purpose of the instruction.

   Comments help explain the command's purpose in this block and are helpful to anyone who debugs or updates the strategy later.

7. Complete each argument for the command by typing in the Type and Name fields or by choosing from the drop-down lists.



Instruction: Subtract

Comment: Open pressure valve when vessel pressure is less than setpoint of 0.5 psig

	Type	Name
	Float Variable	fDoughPressureSetpoint
Minus	Float Literal	0.5
Put Result in	Float Variable	fSetpointMinusDeadband

If the argument type is a literal (or constant), you must type it in. You can use either decimal or hexadecimal form.

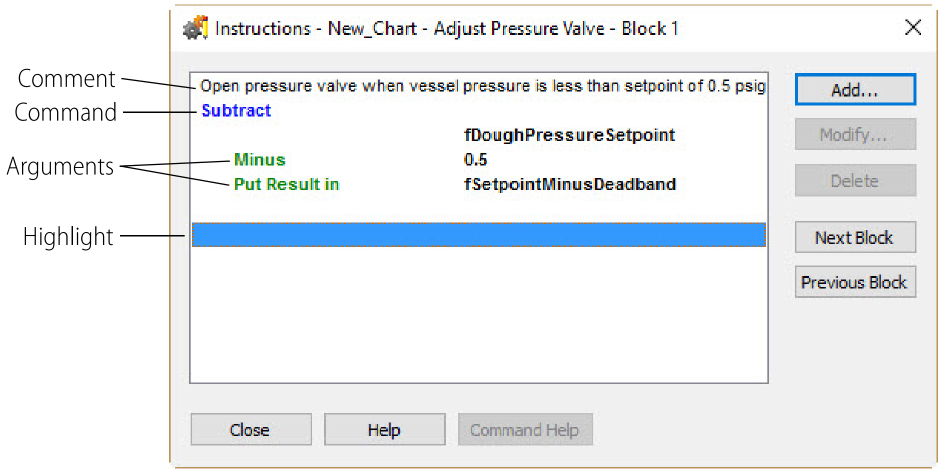
If you type in the name of an item that doesn't exist, for example a variable or I/O point, you are prompted to add it to the strategy.

Each command requires a certain number of arguments, from zero to eight. For help in completing arguments, see the *PAC Control Command Reference* or the online help for the specific command you're using.

8. When the arguments are complete, click OK.

You return to the Instructions dialog box, which now shows the command you just added. Notice that the comment you entered appears just above the command. The arguments you entered appear as part of the instruction. The highlight shows where the next command will be placed if you add another command to this block.

Here is an example of an Instructions dialog box for an action block:



```
Open pressure valve when vessel pressure is less than setpoint of 0.5 psig
Subtract
                              fDoughPressureSetpoint
      Minus                   0.5
      Put Result in           fSetpointMinusDeadband
```

9. To add another command, click to place the highlight where you want the command to appear. Click Add and repeat the steps in this section.

If you highlight a command that's already in the dialog box, the new command will appear just before it.

TIP: *If you are adding commands to several blocks at once, you can quickly move from block to block in the chart by clicking the Next Block or Previous Block buttons in the Instructions dialog box.*

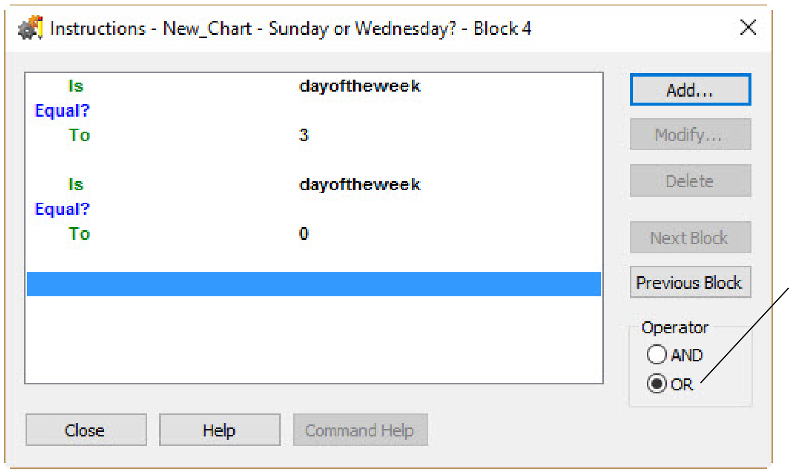
If you're working with a condition block, clicking Next Block opens a dialog box so you can select which block to choose, since condition blocks have two exits. The same dialog box appears if you click Previous Block and the block you're working with has connections coming from more than one block.

10. If you put more than one command in a condition block, complete the Operator group as follows:

– If both commands must be true to exit the block true, click AND.

– If only one of the commands must be true to exit the block true, click OR.

Here is an example of an Instructions dialog box for a condition block with two commands. In this case, the block will exit true if either of the commands is true.



Instructions - New_Chart - Sunday or Wednesday? - Block 4

```
    Is          dayoftheweek
Equal?
    To          3

    Is          dayoftheweek
Equal?
    To          0
```

Operator: AND [✓ OR]

Either the first command OR the second command can be true for this condition block to exit true.

Changing a Command

1. With the strategy open in Configure or Online mode and the flowchart open, double-click the block containing the command you want to change.

   NOTE: To change commands in OptoScript blocks, see "Using the OptoScript Editor" on page 392.

2. In the Instructions dialog box, double-click any line of the command you want to change.

   You can also click the command to highlight it and click Modify.

3. In the Edit Instruction dialog box, make the necessary changes.

   For help, see "Adding Commands" on page 270.

4. Click OK to return to the Instructions dialog box, where you can see the changed command.

Deleting a Command

You can delete a command permanently, or you can comment out a command so it is temporarily skipped, usually for debugging purposes.

Permanently Deleting a Command

1. With the strategy in Configure or Online mode and the flowchart open, double-click the block containing the command you want to delete.

   NOTE: To delete commands in OptoScript blocks, see "Using the OptoScript Editor" on page 392.

2. In the Instructions dialog box, click any line of the command you want to delete.

   ***CAUTION**: Make sure you select the correct command. You cannot undo a deletion!*

3. Click Delete or press Delete on the keyboard.

Commenting Out a Command

You can mark certain commands so that the strategy temporarily ignores them. Commenting out one or more commands can help you pinpoint problems in a strategy.

1. With the strategy in Configure or Online mode and the flowchart open, double-click the block containing the command(s) you want to comment out.

   NOTE: To comment out commands in OptoScript blocks, see "Using the OptoScript Editor" on page 392.

2. In the Instructions dialog box, click the first command you want to comment out. Click Add.
3. In the Add Instructions dialog box, choose the instruction Comment (Block). Click OK.

   You return to the Instructions dialog box, and all the instructions in the block from that point on are grayed out, indicating that they will be ignored when the strategy runs.

4. Click just beyond the last command you want to comment out. Add another Comment (Block) instruction.

   When you return to the Instructions dialog box, all commands between the two Comment (Block) instructions are grayed out.

5. When you no longer want the strategy to ignore the command(s), delete the two Comment (Block) instructions.

   NOTE: The Comment (Block) command is used for this purpose. The Comment (Instruction) command just places an explanatory comment in the Instructions dialog box. It does not affect any commands.

Cutting or Copying a Command

For commands in action, condition, and continue blocks you can cut or copy commands to the Windows clipboard and then paste them in the same block or in another block, or even in a block of another chart within the same strategy. For OptoScript blocks, see "Using the OptoScript Editor" on page 392.

1. With the strategy in Configure or Online mode and the flowchart open, double-click the block containing the command you want to cut or copy.
2. In the Instructions dialog box, click any line of the command you want to cut or copy.

   To cut or copy more than one command, hold down the Shift key while you click all the commands to be cut or copied at once.

   NOTE: To cut and paste all commands in a block, copy and paste the entire block, and then change its name if necessary.

3. Press Ctrl+X to cut the command(s) or Ctrl+C to copy them.

   You can also click the right mouse button and choose Cut or Copy from the pop-up menu.

   The command is cut or copied to the Windows clipboard.

Pasting a Command

Once you have cut or copied a command, you can paste it into any block in the same strategy.

Choose one of the following, and then press Ctrl+V:

- To paste the command in the same block, click where you want to insert the command.
- To paste the command at the end of the instruction block, move the cursor just below the last instruction and click to highlight the empty space.
- To paste the command to another block, click Close to exit the current Instructions dialog box and double-click the block where you want to place the command. Click where you want to insert the command.

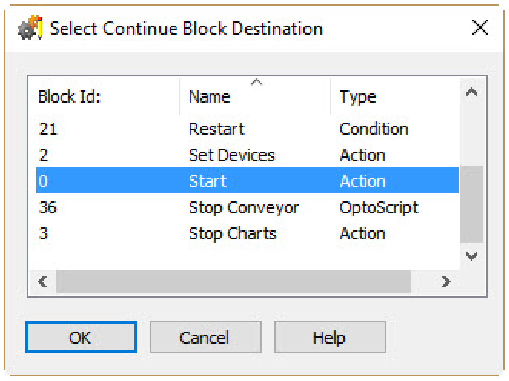
Configuring a Continue Block

Continue blocks do only one thing: jump to another block. Thus, the only information you need to provide in a continue block is the name of the block to jump to.

1. With the strategy in Configure or Online mode and the flowchart open, double-click the continue block you want to configure.

   You can also click the block, click the right mouse button, and choose Detail from the pop-up menu.

   The Select Continue Block Destination dialog box appears, listing all blocks in the chart.



Block Id:	Name	Type
21	Restart	Condition
2	Set Devices	Action
0	Start	Action
36	Stop Conveyor	OptoScript
3	Stop Charts	Action

2. Click the destination block and click OK.

Viewing and Printing Chart Instructions

To view or print commands in a chart, see "Viewing and Printing Strategy or Subroutine Commands" on page 223.

10: Programming with Commands

Introduction

Commands (or instructions) in PAC Control are roughly grouped by function. This chapter tells you what you need to know about each group in order to program your PAC Control strategy effectively. For detailed information on using a command, see Opto 22 form 1701, the PAC Control Command Reference or form 1711, the PAC Control Command Reference, Legacy Edition.

In this Chapter

Analog Point Commands....278
Chart Commands....279
Communication Commands....280
Control Engine Commands....327
Digital Point Commands....327
Error Handling Commands....331
Event/Reaction Commands....332
I/O Unit Commands....334
I/O Unit—Event Message Commands....336
I/O Unit—Memory Map Commands....337
I/O Unit—Scratch Pad Commands....338
Logical Commands....340
Mathematical Commands....342
Miscellaneous Commands....345
PID—Ethernet Commands....345
PID—mistic Commands....349
Pointer Commands....355
Simulation Commands....357
String Commands....358
Time/Date Commands....366
Timing Commands....367

Analog Point Commands

The following commands are used with analog points:

Offset and Gain	**Totalizers**
Calculate & Set Analog Gain	Get Analog Totalizer Value
Calculate & Set Analog Offset	Get & Clear Analog Totalizer Value
Set Analog Gain	Set Analog Totalizer Rate
Set Analog Offset	
	Others
Minimum/Maximum Values	Get HART Unique Address
Get Analog Maximum Value	Receive HART Response
Get Analog Minimum Value	Receive HART Burst Response
Get & Clear Analog Maximum Value	Send/Receive HART Command
Get & Clear Analog Minimum Value	Set Analog Filter Weight
	Set Analog Load Cell Fast Settle Level
	Set Analog Load Cell Filter Weight
	Set Analog TPO Period

Offset and Gain	**Others**
Calculate & Set Analog Gain	Get & Clear Analog Filtered Value[1, 2]
Calculate & Set Analog Offset	Get & Clear Analog Totalizer Value
Set Analog Gain	Get Analog Filtered Value[1, 2]
Set Analog Offset	Get Analog Square Root Filtered Value[1, 2]
	Get Analog Square Root Value[1, 2]
Minimum/Maximum Values	Get Analog Totalizer Value
Get Analog Maximum Value	Get HART Unique Address
Get Analog Minimum Value	Ramp Analog Output
Get & Clear Analog Maximum Value	Receive HART Response
Get & Clear Analog Minimum Value	Receive HART Burst Response
	Send/Receive HART Command
	Set Analog Filter Weight
	Set Analog Load Cell Fast Settle Level
	Set Analog Load Cell Filter Weight
	Set Analog Totalizer Rate
	Set Analog TPO Period

[1] Applies to PAC Control Professional only
[2] Applies to *mistic* I/O units only

Offset and Gain Commands

Although you normally calibrate analog points in PAC Manager, PAC Control provides commands to set the values for a point's offset and gain. (For more information about calibrating offset and gain, see "Reading and Writing to Analog Points" in Opto 22 form 1704, the *PAC Manager User's Guide*.)

If you already know a point's offset and gain, you can set the values in PAC Control by using the Set Analog Offset and Set Analog Gain commands. If you don't know the offset and gain, you can use the Calculate & Set Analog Offset command, and then use Calculate & Set Analog Gain to have the brain calculate the offset and the gain for you—but again, calibration is typically performed in PAC Manager, and not within a strategy's logic.

Analog Totalizers

Analog totalizers are used to track total volume or quantity. For example, if an analog point measures gallons per minute, you could use an analog totalizer to determine the total number of gallons moved over a period of time.

To read the value and leave the totalizer running, use the command Get Analog Totalizer Value. To read the value and set the totalizer back to zero, use the command Get & Clear Analog Totalizer Value.

Minimum/Maximum Values

The Opto 22 brain automatically keeps track of minimum and maximum values for analog input points. Min/max values are often used to monitor pressure or temperature.

To read the minimum or maximum value and leave it as is, use Get Analog Minimum Value or Get Analog Maximum Value. To read the minimum or maximum value and clear it—for example, to record the minimum pressure in each 24-hour period—use Get & Clear Analog Minimum Value or Get & Clear Analog Maximum Value.

Analog Totalizers

Analog totalizers are used to track total volume or quantity. For example, if an analog point measures gallons per minute, you could use an analog totalizer to determine the total number of gallons moved over a period of time.

To read the value and leave the totalizer running, use the command Get Analog Totalizer Value. To read the value and set the totalizer back to zero, use the command Get & Clear Analog Totalizer Value.

Analog Points and OptoScript Code

In OptoScript code, an analog I/O point can be used directly, wherever a float variable can be used. For example, you can assign an analog point a value, or use points directly in mathematical expressions and control structures. For more information, see "Using I/O in OptoScript" on page 378.

Reading or Writing Analog Points

The Move command is the main command used to read or write analog points.

Chart Commands

The following commands control charts in the strategy:

Call Chart	Continue Calling Chart

Calling Chart Running?	Continue Chart
Calling Chart Stopped?	Get Chart Status
Calling Chart Suspended?	Start Chart
Chart Running?	Stop Chart
Chart Stopped?	Suspend Chart
Chart Suspended?	

For information about charts in a PAC Control strategy, see "PAC Control Terminology" on page 49.

About the Task Queue

How do subroutines fit into the task queue?

Whenever a chart calls a subroutine, the subroutine temporarily inherits the task in use by the calling chart along with its priority.

Does a task always use all of its allocated time?

Not always. If a chart or subroutine runs in a loop, all allocated time is used. If a chart or subroutine does not need all of its allocated time to complete its job, all remaining time (including any portion of a time slice) is given up.

The following conditions cause a chart to use less than a full time slice:

- The chart or subroutine stops.
- The chart or subroutine is suspended.
- A Delay command is used.

Using the command Delay (mSec) with a value of 1 millisecond is a handy way to give up the time slice while waiting in a loop for a condition to become True. For more information, see form 1776, Optimizing PAC Project System Performance.

When does the requested change to a chart or task status take effect?

Not immediately. In any multitasking system, timing and synchronization issues are always a concern. The time required for a particular request to be implemented depends on the number of tasks currently running and the specified chart's location in the task queue. We recommend using commands such as Calling Chart Suspended? and Chart Running? to determine the status of a chart.

Communication Commands

The following commands refer to moving data among entities that store and transfer data:

Accept Incoming Communication	Receive Pointer Table
Clear Communication Receive Buffer	Receive String
Close Communication	Receive String Table
Communication Open?	Send Communication Handle Command
Get Active Interrupt Mask[1]	Send Email
Get Communication Handle Value	Send Email with Attachments
Get End-Of-Message Terminator	Set Communication Handle Value

Get Number of Characters Waiting	Set End-Of-Message Terminator
HTTP Get	Transfer N Characters
HTTP Post Calculate Content Length	Transmit Character
HTTP Post From String Table	Transmit NewLine
Listen for Incoming Communication	Transmit Numeric Table
Open Outgoing Communication	Transmit Pointer Table
Receive Character	Transmit String
Receive N Characters	Transmit String Table
Receive Numeric Table	Transmit/Receive Mistic I/O Hex String[1]
Receive Numeric Table Ex	Transmit/Receive String
Receive Numeric Variable	

[1] PAC Control Professional only; *mistic* I/O units only

Accept Incoming Communication	Receive Pointer Table
Clear Communication Receive Buffer	Receive String
Close Communication	Receive String Table
Communication Open?	Send Communication Handle Command
Get Communication Handle Value	Set Communication Handle Value
Get End-Of-Message Terminator	Send Email
Get Number of Characters Waiting	Send Email with Attachments
HTTP Get	Set End-Of-Message Terminator
HTTP Post Calculate Content Length	Transfer N Characters
HTTP Post From String Table	Transmit Character
Listen for Incoming Communication	Transmit NewLine
Open Outgoing Communication	Transmit Numeric Table
Receive Character	Transmit Pointer Table
Receive N Characters	Transmit String
Receive Numeric Table	Transmit String Table
Receive Numeric Table Ex	Transmit/Receive String
Receive Numeric Variable	

Communication Handles

A communication handle (also known as comm handle or COM handle) is a variable in PAC Control that stores the parameters needed to connect to a specific entity for the purpose of exchanging data. The entity may be another device on the network, a file located on the control engine, or some other thing that stores or transfers data. The value of a communication handle variable is a string consisting of communication parameters, separated by colons.

Communication handles are used to open and close communication channels, and to transmit and receive data. You can use communication handles to:

- Communicate with devices (including serial communication modules) via an Ethernet connection
- Communicate with devices via a physical serial port on a SNAP PAC controller
- Create, read, and write data to local and remote files and tables

For example, you might use a UDP communication handle to communicate with another device on the network via a UDP/IP connection, and use an FTP communication handle to send data from a controller to a file on a PC.

NOTE: A communication handle is not required to use the SNAP PAC REST API. To configure HTTPS access to your PAC's RESTful server and learn how to call the API, visit developer.opto22.com.

In many cases you'll define a communication handle variable and never change it, because the parameters rarely change. However, in some cases it might be useful to change it. (Just remember that if you change parameters while a communication handle is open, you must close the comm handle and reopen it for the new parameters to take effect.)

PAC Control provides these types of communication handles:

TCP	For Ethernet communication via TCP/IP, or with SNAP PAC serial modules. Note that outgoing (client) TCP communication handles in PAC firmware R9.5a (or higher) support secure data exchange via SSL/TLS. Currently, SSL is not supported in incoming (server) TCP communication handles.	See page 283.
UDP	For lightweight Ethernet communication via UDP/IP.	See page 283.
File	For creating files to be stored on a control engine, and for reading or writing to files on a control engine.	See page 301.
FTP	For accessing files on a local file system (like a network drive), or on an FTP server. Also used to transfer files between two servers.	See page 308.
Serial	For communication with serial devices that are physically connected via an RS-232 or RS-485 connector on a SNAP PAC R-series or S-series controller. (Not used for serial communication modules or serial-based I/O units.)	See page 312.

Communication Timeout Values

Each type of communication handle has a predefined default timeout value (the amount of time the system waits for a command to complete). Timeouts prevent systems from wasting resources when something is wrong.

Communication Handle Type	Default Timeout Value
TCP	10 seconds
UDP	10 seconds
File	1 second
FTP	30 seconds
Serial	1 second

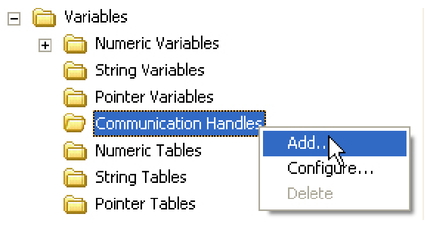
For the TCP, UDP, and Serial communication handles, you can set a different timeout value and read the current timeout value for an already open communication handle. For TCP and UDP only, you can also set the timeout values for receiving data, and for opening both incoming and outgoing communications. For details, see the "Communication Commands" section in Opto 22 form 1701, *PAC Control Command Reference*.

Adding Communication Handles

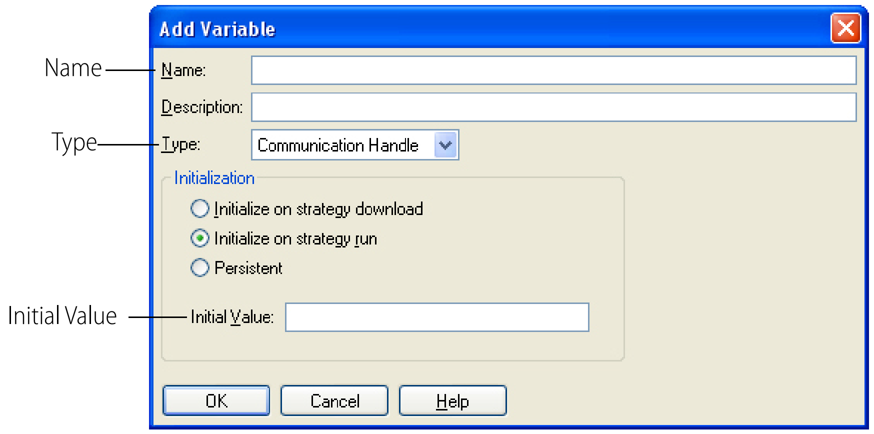
Using the Add Variable dialog box, you can create variables for each type of communication handle. For more information about variables, see "Adding Variables" on page 257.

To add a communication handle:

1. In the Strategy Tree, under Variables, right-click Communication Handles, and then choose Add to open the Add Variable dialog box.



2. At the minimum, complete the following fields:
   - **Name**: Create an appropriate name, such as TCP_Handle.
   - **Type**: Select Communication Handle.
   - **Initial Value**: Varies with type.
     - For TCP and UDP, see page 283.
     - For File, see page 301.
     - For FTP, see page 308.
     - For Serial, see page 312.



Using TCP and UDP Communication Handles

TCP and UDP communication handles exchange data with accessible devices anywhere on a network and even over the Internet.

- Use TCP when you need reliable data transmission with error handling and message retries.
- Use UDP when bandwidth is a concern and possible data loss is not an issue.

NOTE: If the two devices that are sharing data are both control engines, it may be easier to use the Scratch Pad areas on each to transfer data, rather than using communication commands. See "I/O Unit—Scratch Pad Commands" on page 338 *for more information.*

Because SNAP serial communication modules are IP-addressable, you communicate with them by using a TCP or UDP communication handle (not a Serial communication handle). This way, a strategy running on a PC or SNAP PAC device can access serial devices connected to modules on a remote SNAP PAC R-series or EB-series device (which in this case, acts as an Ethernet-to-Serial converter).

NOTE: Be sure to use a separate TCP (or UDP) communication handle for each chart in a strategy. If two charts were to run simultaneously while sharing an open communication handle, each chart would be able to read and write data from the communication handle as if the other running chart didn't exist. Because these reads and writes are not synchronized between the charts, it is possible for one chart to read the other chart's data.

TCP Connections

TCP is a connection-oriented protocol, which means a connection must be established between the *client* (the device requesting the data) and the *server* (the device providing the data) before the devices exchange data. This connection is typically called a *handshake*. TCP ensures reliable data transmission, but the handshake and reliable data transmission require more bandwidth than UDP.

Outgoing (client) TCP communication handles in PAC firmware R9.5a (or higher) support secure data exchange via SSL/TLS—two similar types of Internet protocols that provide authentication (that is, established proof of identification) over an encrypted network channel. Lower versions of PAC Control and PAC firmware support SSL over TCP only with the HTTP Get, HTTP Post, Send Email, and Send Email With Attachment commands. (UDP protocol does not support SSL/TLS.) For more information about SSL/TLS, see "Installing CA Root Certificates" on page 316.

Incoming and Outgoing Communication

TCP and UDP communication is normally requested by the device that needs the data.

For example, when a PAC Control strategy running on a SNAP PAC I/O unit needs the value of a set of variables from another strategy running on a different SNAP PAC controller:

- The controller (the client) requests communication using the command, Open Outgoing Communication.
- The second controller (the server) uses the Listen for Incoming Communication command, and then the Accept Incoming Communication command to establish the connection.

Once connected, data is transmitted and received in both directions.

Depending on the situation, you may want to have both devices request communication and establish two connections between them. That way if one device goes offline and then comes online, you can have the flowchart logic set up to immediately open communication again.

In some cases, however, one of the devices may be incapable of requesting communication; for example, a serial device such as a barcode reader attached to a serial communication module on a controller. The serial device is incapable of requesting Ethernet communication; essentially it can only listen and provide data when asked. In cases like this, the controller must use Open Outgoing Communication to initiate communication. (For serial devices attached directly to a serial port on a SNAP PAC controller or brain, SNAP-LCE controller, or SNAP Ultimate brain, see "Using Serial Communication Handles to Communicate with Serial Devices" on page 312.)

Outgoing Communication. For *outgoing* communication (that is, communication initiated by the client), the communication handle's value is in the format:

UDP	`udp:`<Server's IP address or hostname>`:`<port that is listening for requests>
Basic TCP	`tcp:`<Server's IP address or hostname>`:`<port that is listening for requests>
SSL/TLS over TCP	`tcp:`<Secure server's IP address or hostname>`:`<port that is listening for requests>,`ssl[]` Note the comma after the port number and the empty square brackets after "ssl".

NOTES:

- "tcp" and "udp" are lowercase.
- A hostname is user-friendly name for the IP address. Hostnames are usually created by a company's IT department. (Before you substitute a hostname for a controller's IP address, use PAC Manager to configure the DSN parameter for the controller. For details, see Opto 22 form 1704, the *PAC Manager User's Guide*.)
- For information about ports, see page 286.
- Only TCP supports outgoing (client) communication with SSL/TLS authentication and encryption. To use SSL/TLS, your controller must have an installed *security certificate*. (For information about security certificates, see "Installing CA Root Certificates" on page 316.)

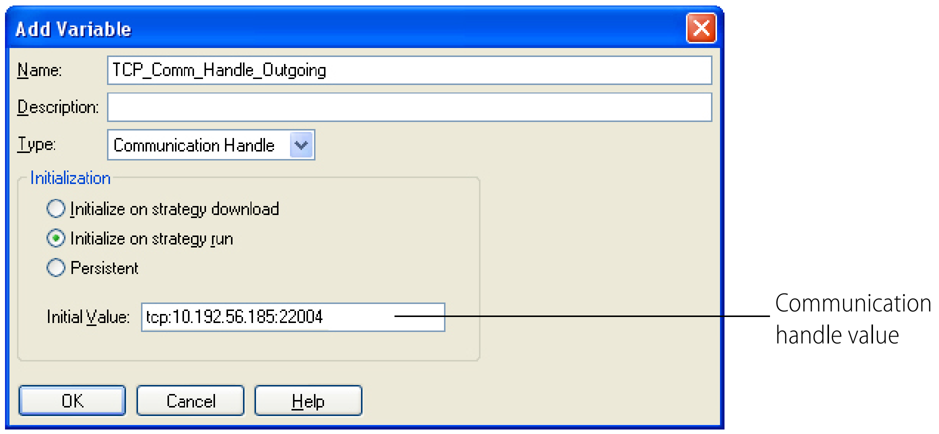
The following table shows examples of TCP and UDP communication handles for outgoing communication.

Outgoing Communication	Communication Handle Value
	Protocol:Server's IP Address or Hostname:Port,
Ethernet TCP - to a serial communication module on another rack	`tcp:10.192.54.10:22506`
Ethernet TCP - to a remote server (using the server's hostname)	`tcp:my_server:255`
Ethernet TCP (with secure SSL/TLS) - to a remote server	`tcp:10.192.56.185:443,ssl[]`

Ethernet TCP - to a serial communication module on the same rack (uses SNAP PAC R-series or SNAP Ultimate brain's IP address or loopback IP address)	`tcp:10.192.55.90:22511`
	or
	`tcp:127.0.0.1:22511`

Ethernet UDP - to a serial communication module on another rack	`udp:10.192.54.10:22506`

To add a communication handle to a PAC Control strategy, see the steps starting on page 283. In the Add Variable dialog box, enter the communication handle value in the Initial Value field.



Add Variable

Name: TCP_Comm_Handle_Outgoing

Description:

Type: Communication Handle

Initialization

- Initialize on strategy download
- [✓ Initialize on strategy run]
- Persistent

Initial Value: tcp:10.192.56.185:22004

Port Numbers. To communicate with other control engines or other devices on a network, you can use any port number that is not already in use by that device. For example, you can safely use ports 22004 and 22005, which have been assigned to Opto 22 by the Internet Assigned Numbers Authority (IANA). Opto 22 has allocated these Ethernet port numbers for customer use. In general, anything below 1024 is already predefined and shouldn't be used unless you are implementing a predefined protocol that already has a reserved port.

If you do not know what port number to use, ask your network administrator or check the list of standard reserved Ethernet port numbers at www.iana.org/assignments/port-numbers to see ports that may apply to your devices on your network.

The following port numbers are the defaults, reserved by Opto 22 devices for the purposes shown.

Port #	Purpose
20	FTP (file transfer protocol) for get, send, and dir functions
21	FTP (file transfer protocol)
25	SMTP (simple mail transfer protocol)
161	SNMP-based enterprise management systems
502	Modbus/TCP
2001	Main command processor, OptoMMP-based (also used by SNAP PAC controllers and brains, SNAP Ethernet I/O, and E1/E2 brains)
2222, 44818	Ethernet/IP
22000	Opto 22 controllers and brains
22001	Host port (for information on configuring the host port, see Opto 22 form 1704, the PAC Manager User's Guide)
22002	Opto 22 controllers and brains
22003	Opto 22 controllers and brains

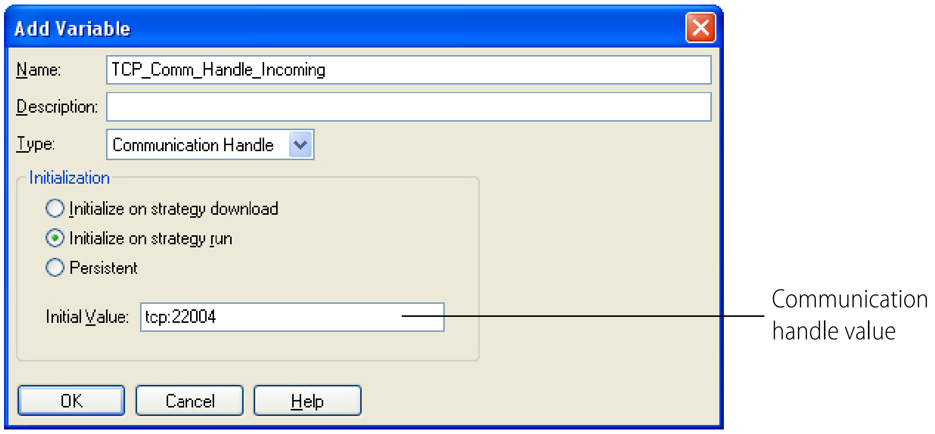
Port #	Purpose
22500–22531	Serial communication modules
50000–50999	Internal messaging

Note that port numbers on the SNAP Ultimate brains, SNAP-LCE controllers, and SNAP PAC controllers and brains are the default ports; some may have been changed for security reasons. (For more information, see the section on Security in Opto 22 form 1704, the PAC Manager User's Guide.)

Incoming Communication. For *incoming* communication (that is, communication requested by another device), the communication handle value includes just the protocol and the port number: `Protocol:Port`. TCP automatically tracks senders so there is no mix-up in the data sent and received. Here are a couple of examples of communication handles for incoming communication:

Incoming Communication	**Communication Handle Value**
	Protocol:Port
Ethernet TCP (for example, to use a controller as an HTTP server)	`tcp:80`
Ethernet UDP (for example, to listen for packets being streamed from a remote I/O unit)	`udp:5001`

To add a communication handle, see the steps starting on page 283. In the Add Variable dialog box, enter the communication handle value in the Initial Value field.



Name: TCP_Comm_Handle_Incoming
Description:
Type: Communication Handle
Initialize on strategy download
[✓ Initialize on strategy run]
Persistent
Initial Value: tcp:22004

Opening and Closing TCP Communication Handles

When using TCP communication handles, keep in mind that the controller has a limited amount of TCP/IP resources, so you need to be careful about how frequently these communication handles are opened and closed. TCP communication handles are meant to be left open as long as possible.

Per the TCP/IP specification, each time one of these resources is closed, the closed resource unavailable for 120 seconds. Rapidly attempting to open and close TCP communication handles quickly places all available resources in the unavailable condition.

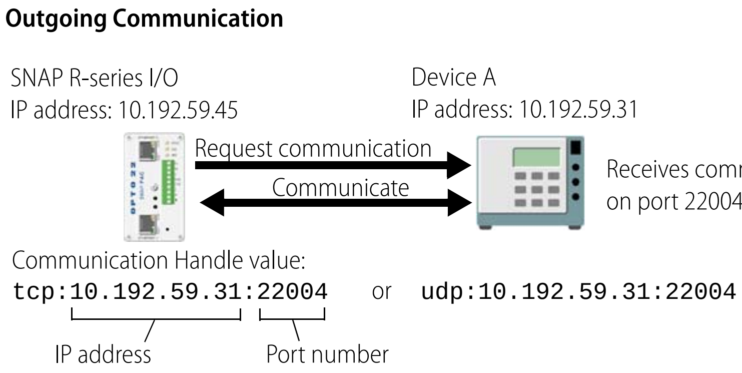
Once this unavailable state occurs, a communication handle will be unable to gain a resource upon the next Open Outgoing Communication command. This effectively causes a lock out of any new TCP/IP connections to be made with the controller (whether from the strategy or externally).

The lock out affects applications such as PAC Control, PAC Display, PAC Manager, OptoOPCServer, and PAC Terminal. These applications will either have intermittent or no connectivity with the controller.

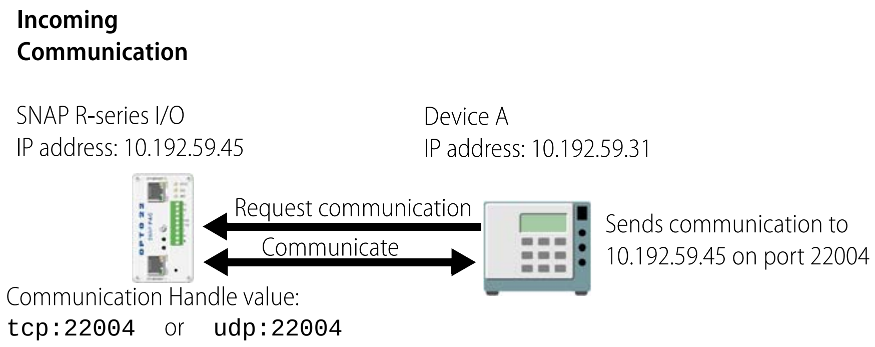
The solution to this dilemma is to control the rate at which TCP communication handles are opened and closed. When a TCP communication handle is closed (whether a result of a prior successful or unsuccessful opening), it should be reopened no sooner than 3 seconds. If this reopening fails to succeed, double the delay to 6 seconds. If this subsequent reopening fails, double the delay to 12 seconds. And thereafter, subsequent reconnections should be performed no shorter than every 24 seconds. This throttling of the opening allows several TCP/IP resources to expire their closing timer.

TCP and UDP Communication Handle Examples

The following diagram shows an example of a communication handle value for outgoing communication:



For incoming communication, the communication handle value could be:

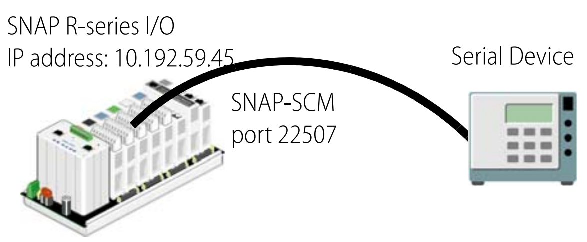


Communication Handles for Serial Communication Modules. For communication with serial devices through a serial communication module, the communication handle value consists of the IP address of the brain the module is attached to, plus the serial module's port number according to its position on the rack.

NOTE: For port number information, see "Establish an Ethernet Connection" in Opto 22 form 1191, the SNAP Serial Communication Module User's Guide.

Here are two examples of communication handles for communicating with serial modules, the first showing communication through a module on the same rack as a SNAP PAC R-series controller, and the second showing communication through a module on a different rack.

To a Serial Module on the Same Rack



For a serial module on the same rack, use the loopback IP address 127.0.0.1, which tells the brain to talk to its own rack. You can also use the brain's own IP address (in this example, 10.192.59.45), but the loopback address allows you to change the IP address of the brain without having to change the communication handle.

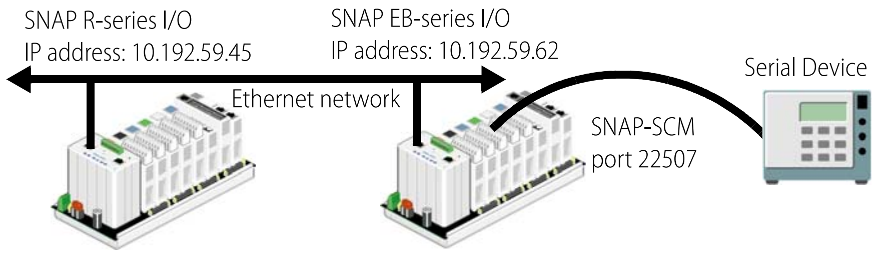
Communication Handle value:

`tcp:127.0.0.1:22507` or `udp:127.0.0.1:22507`

or $\texttt{tcp:}\underbrace{\texttt{10.192.59.45}}_{\text{IP address}}\texttt{:}\underbrace{\texttt{22507}}_{\text{Port number}}$ or `udp:10.192.59.45:22507`

NOTE: For port number information, see "Establish an Ethernet Connection" in form 1191, the SNAP Serial Communication Module User's Guide.

To a Serial Module on a Different Rack



Communication Handle value:

$\texttt{tcp:}\underbrace{\texttt{10.192.59.62}}_{\text{IP address}}\texttt{:}\underbrace{\texttt{22507}}_{\text{Port number}}$ or `udp:10.192.59.62:22507`

NOTE: For port number information, see "Establish an Ethernet Connection" in form 1191, the SNAP Serial Communication Module User's Guide.

Using Flowcharts to Control TCP or UDP Communication

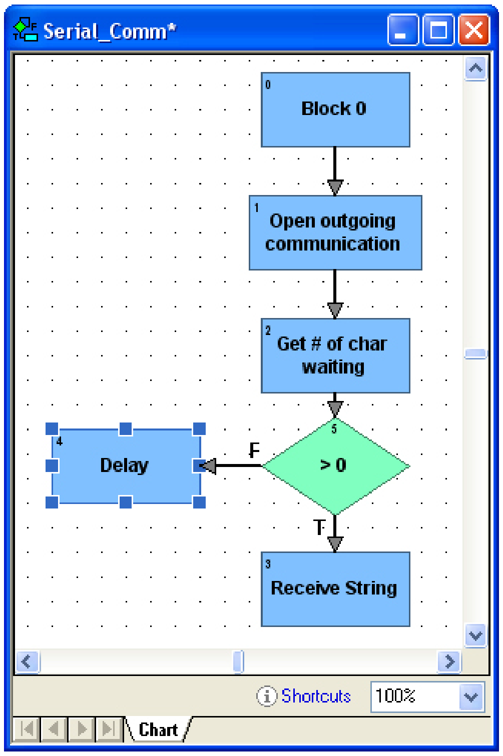
When a control engine is communicating with another device using TCP or UDP, it runs a PAC Control flowchart or charts to control communication.

- **For outgoing communication on controller COM ports and serial modules,** the flowchart uses the command Open Outgoing Communication to request a connection.
- **For incoming communication on controller COM ports,** which is requested by another device, the flowchart must first use Listen for Incoming Communication and then use Accept Incoming Communication to establish a connection.

- **For incoming communication on serial modules,** which is requested by another device, the flowchart uses the Open Outgoing Communication and Communication Open? commands to establish a connection.

The control engine's TCP/IP flowcharts should open communication once and then continue to transmit or receive using the communication handle. Constantly opening and closing communication for each transaction wastes time and is inefficient.

As a simple example, the flowchart at right is designed to receive data from a serial device, such as a barcode reader, through a serial communication module on the same rack as a SNAP PAC R-series I/O unit.



Communication with serial modules in this example is done via TCP. It can be done from the SNAP PAC R-series system itself, as in this example, or from another TCP/IP device on the network.

In this example, the R-series I/O unit establishes communication through the serial module using the communication handle:
`tcp:127.0.0.1:22507`
The loopback IP address is used, since the serial module is on the same rack.

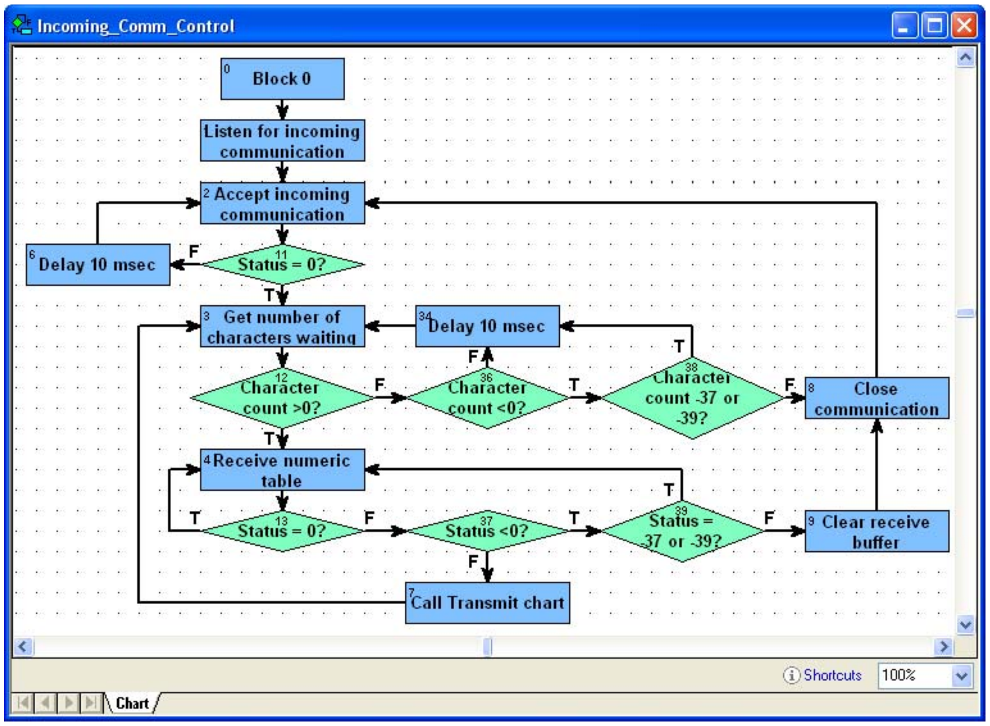
When characters are detected in the receive buffer, the I/O unit receives the string and processes it, and then after a short delay checks for another message. Communication remains open for additional messages.

This simple flowchart illustrates the basics of handling communication, without any error checking. However, while receiving and transmitting, TCP/IP control charts should also monitor the status value returned by commands such as Transmit Numeric Table and Receive Numeric Table, and close communication if there are errors. If your PAC Control application opens sessions but does not close unused or bad sessions, the maximum number of sessions could be used up. Note that communication should not be closed for timeout errors from Receive commands (error numbers –37 and –39), however, because these errors simply mean that there was no data waiting in the receive buffer for the specified session.

To save time, before using a Receive command (such as Receive String or Receive Numeric Table), TCP/IP charts should use the command Get Number of Characters Waiting. If there are zero characters waiting, then there is no reason to use the Receive command. It is also a good idea to start a down timer, and then loop on the Get Number of Characters Waiting command until either there are more than zero characters waiting, or the timer expires.

The following chart is also an example of TCP/IP communication on controller COM ports, but with three differences: in this case another device is requesting communication, the peer will both transmit and receive, and error checking is shown for the commands Get Number of Characters Waiting and Receive Numeric Table.

Similar error checking should be used for transmit commands, which in this strategy are in another chart.



If the devices communicating with each other are all SNAP R-series or Ultimate I/O systems, be careful not to have one of them send information to another faster than the receiving system can pull it out of its receive buffer. When a control engine receives data from the Ethernet network, it holds the data in memory (the buffer) until the PAC Control flowchart removes it with a Receive command. If the data is not removed fast enough, the receive buffer will fill up. You can prevent this problem by transmitting the information less frequently. If the same chart is transmitting and receiving, you can alter the chart so that it receives more often than it transmits.

Ethernet Connections and Ports

The number of Ethernet connections available varies by hardware, type, and in some cases firmware version, such as 80 on current SNAP PAC R-series I/O systems.

The host connection is on port 22000 or 22001 and is used to communicate with the PC that runs PAC Control. These port numbers are reserved for this purpose, but they can be configured. For information see Opto 22 form 1704, the PAC Manager User's Guide.

Peer-to-peer connections can be on any other port number that is not already used on the network; these connections are used to communicate with other control engines or other devices on the network. When you assign port numbers for peer-to-peer connections, make sure you do not assign port numbers that devices on your network may use for specific purposes. For a list of standard Ethernet port numbers, refer to: www.iana.org/assignments/port-numbers

***IMPORTANT**: If you are using TCP/IP or UDP connections, consult commercially available texts on TCP/IP or UDP that discuss client/server architectures, so you'll understand how the protocol works and what to expect during communication.*

UDP Communication Handles

The User Datagram Protocol (UDP) is a fast, lightweight communications protocol that is sometimes used as an alternative to TCP (see page 283). UDP is used to transmit messages—called *datagrams*—across an IP network from a source IP address and source UDP port to a destination IP address and destination UDP port. UDP communication handles most often used when bandwidth is a concern and reliable data transmission is not required.

Important Characteristics of UDP

- Uses port numbers to help distinguish different types of requests.
- Has an optional checksum function to ensure the datagram is delivered intact
- Uses a connectionless transmission model, which means it doesn't do a "handshake" or otherwise ensure that the receiving IP address is available and ready to receive.
- Does not guarantee datagram delivery. If a delivery fails, it does not perform retries.
- Does not preserve datagram sequencing. Datagrams may arrive at the destination in a different order than they were sent.
- Is unaware of congestion at the receiving IP address. If the receiver's queue becomes full, subsequent datagrams are dropped.

The following table describes the available categories of PAC Control UDP communication handles:

Handle Category	Local Characteristics	Remote Characteristics	Communication Handle Value
Local Bound	The local UDP port is specified by the communication handle.	May be used to communicate with multiple remote IP addresses and ports by using the command "Send Communication Handle Command, "set.dst:<ip address>:<port>" or "set.dst:<hostname with fully qualified domain name>:<port>.	udp:<local port>
Remote Bound	Local UDP port is chosen by the operating system.	Communicates with the communication handle's specified remote IP address and port.	udp:<remote IP address>:<remote port>
Unbound	Local UDP port is chosen by the operating system.	Can be used to communicate with multiple remote IP addresses and ports by using the command "Send Communication Handle Command" with a destination string like "set.dst:<ip address>:<port>" or "set.dst:<hostname with fully qualified domain name>:<port>.	udp:

NOTE: Be sure to use a separate UDP (or TCP) communication handle for each chart in a strategy. If two charts were to run simultaneously while sharing an open communication handle, each chart would be able to read and write data from the communication handle as if the other running chart didn't exist. Because these reads and writes are not synchronized between the charts, it is possible for one chart to read the other chart's data.

Configuring a Local Bound UDP Communication Handle

A local bound communication handle is used to receive UDP datagrams through a known UDP port. You can also configure the communication handle to transmit data to a specific destination. For details, see the "Set Communication Handle Value" command.

To add a communication handle to a PAC Control strategy, see the steps starting on page 283. In the Add Variable dialog box, enter the communication handle value in the Initial Value field.

The local bound communication handle value has the following format:

udp:<local bound port number>

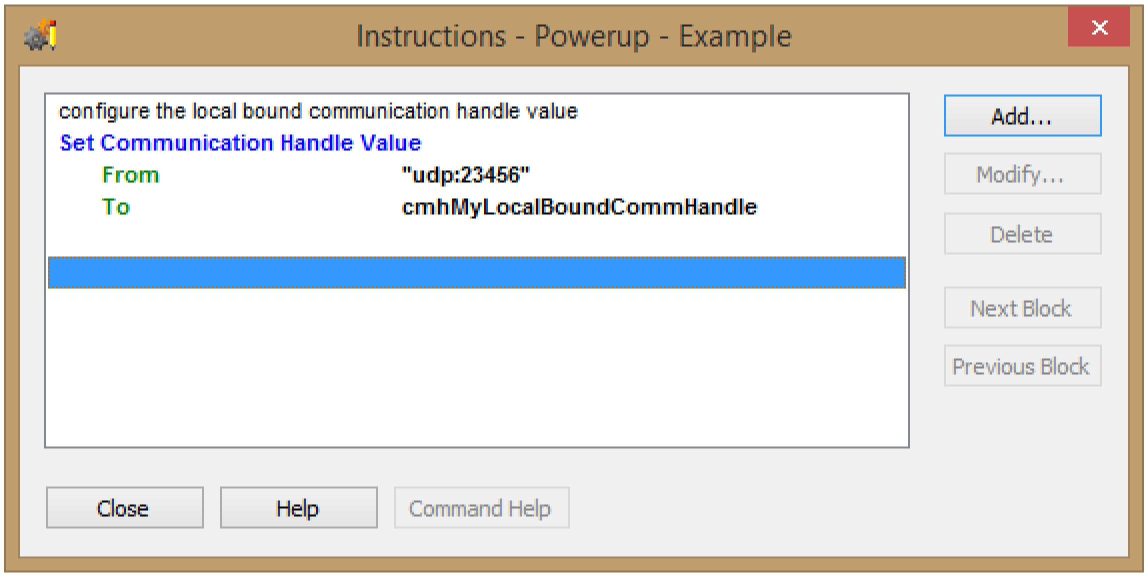
Example:

```
udp:23456
```

OptoScript example:

```
SetCommunicationHandleValue("udp:23456", cmhMyLocalBoundCommHandle);
```

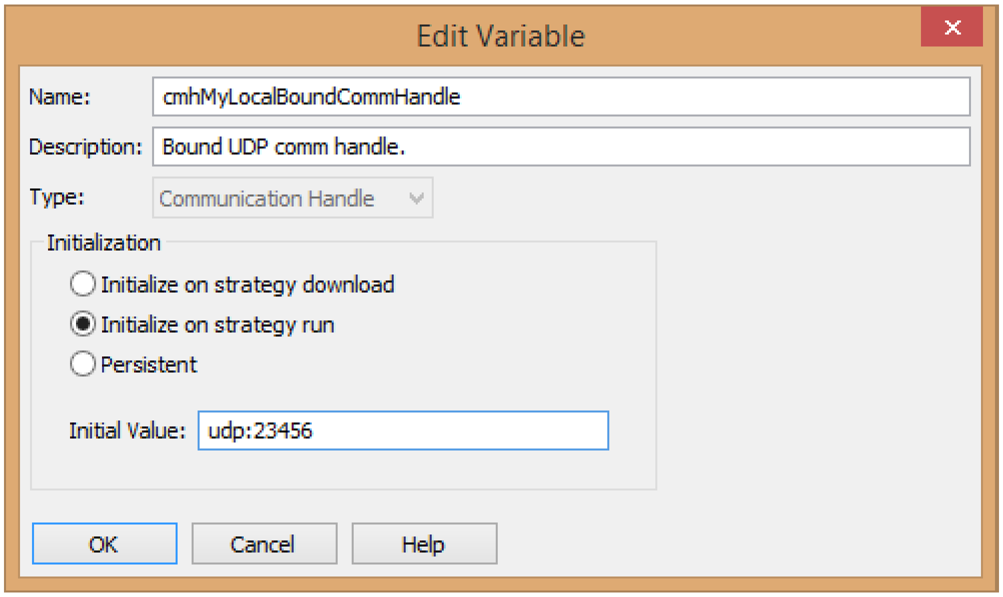
Action block example:



```
configure the local bound communication handle value
Set Communication Handle Value
    From                       "udp:23456"
    To                         cmhMyLocalBoundCommHandle
```

In this example, any data sent via UDP to UDP port 23456 will be received by the PAC Control strategy. UDP datagrams sent to the bound port will be received on all active Ethernet or Wireless Ethernet interfaces.

The communication handle value may be configured when defining the communication handle in the strategy tree. Set the "Initial Value" field when the communication handle is set to be "Initialize on strategy download" or "Initialize on strategy run."



Name: cmhMyLocalBoundCommHandle

Description: Bound UDP comm handle.

Type: Communication Handle

Initialization

( ) Initialize on strategy download
[✓ Initialize on strategy run]
( ) Persistent

Initial Value: udp:23456

For more information about defining variables in the strategy, see "Adding Variables" on page 257.

Configuring a Remote Bound UDP Communication Handle

A remote bound UDP communication handle is typically used to communicate to another device that receives UDP datagrams on a bound (preset) port number. The remote port number allows transmitters to direct the datagram's destination.

To add a communication handle to a PAC Control strategy, see the steps starting on page 283. In the Add Variable dialog box, enter the communication handle value in the Initial Value field

The remote bound communication handle value has the following format:

udp:<remote IP Address>:<remote port number>

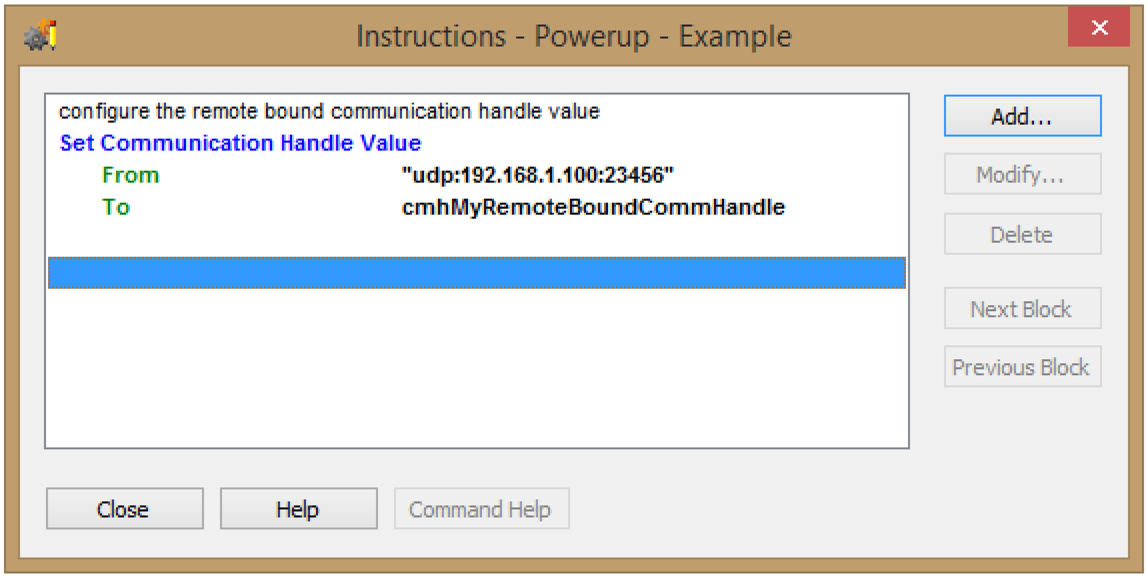
Example:

```
udp:192.168.1.100:23456
```

OptoScript example:

```
SetCommunicationHandleValue("udp:192.168.1.100:23456",
cmhMyRemoteBoundCommHandle);
```

Action block example:



```
configure the remote bound communication handle value
Set Communication Handle Value
    From                    "udp:192.168.1.100:23456"
    To                      cmhMyRemoteBoundCommHandle
```

The example configures a communication handle that transmits data to the destination host IP address 192.168.1.100, UDP port 23456.

Configuring an Unbound UDP Communication Handle

The unbound communication handle is very similar to the remote bound category. The only difference is the communication handle value does not specify the local port.

To add a communication handle to a PAC Control strategy, see the steps starting on page 283. In the Add Variable dialog box, enter the communication handle value in the Initial Value field.

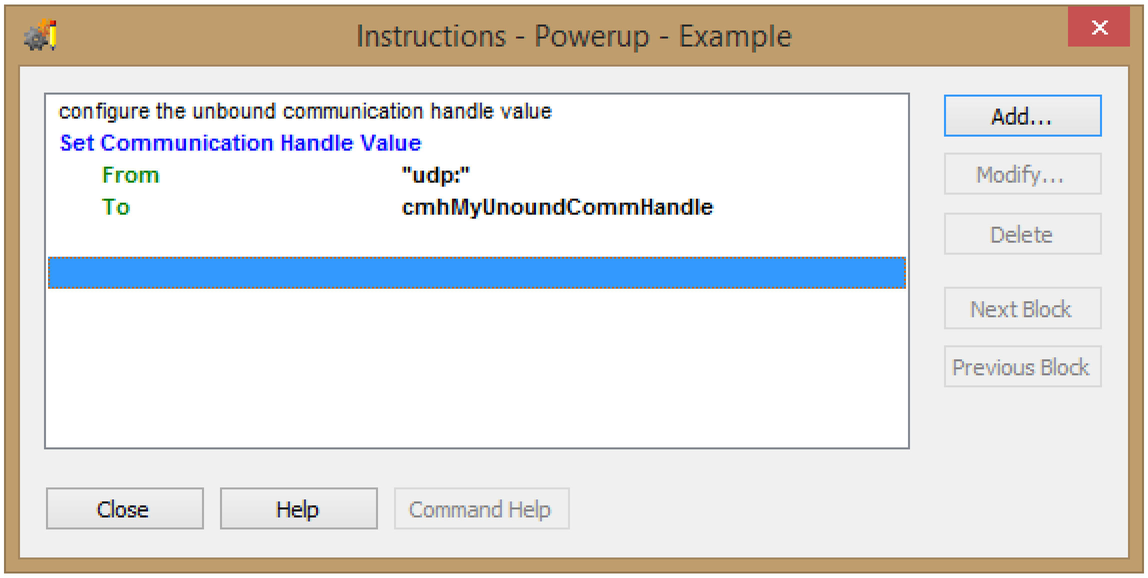
The unbound UDP communication handle value has this format:

udp:

OptoScript example:

```
SetCommunicationHandleValue(“udp:”, cmhMyUnboundCommHandle);
```

Action block example:



```
configure the unbound communication handle value
Set Communication Handle Value
    From                  "udp:"
    To                    cmhMyUnoundCommHandle
```

Opening the UDP Communication Handle

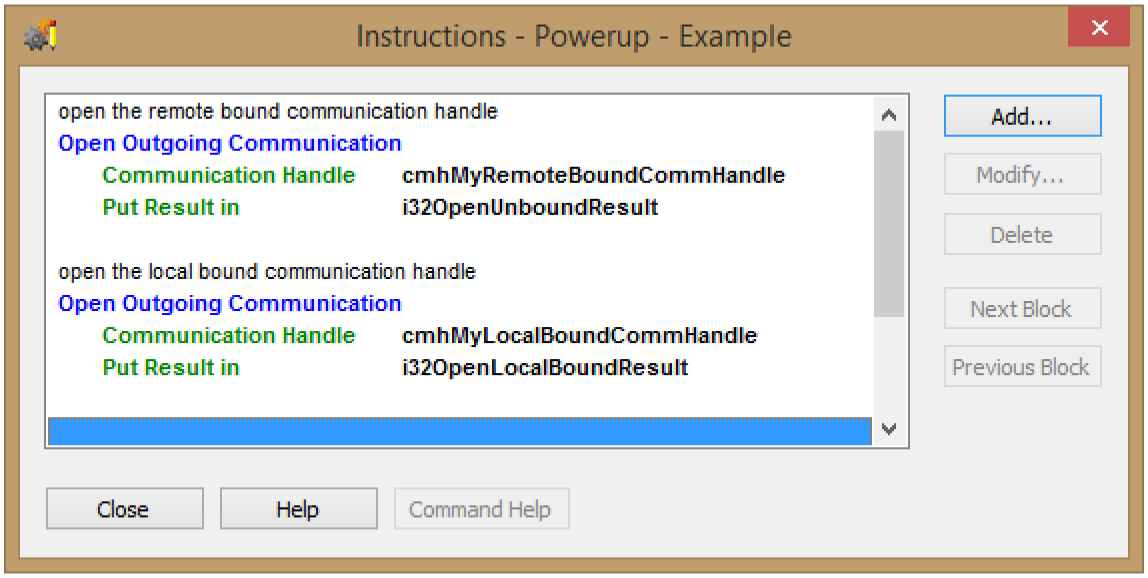
To open either an unbound or bound communication handle, use the OpenOutgoingCommunication command.

OptoScript example:

```
i32OpenUnboundResult =
OpenOutgoingCommunication(cmhMyRemoteBoundCommHandle);
```

```
i32OpenBoundResult =
OpenOutgoingCommunication(cmhMyLocalBoundCommHandle);
```

Action block example:



```
open the remote bound communication handle
Open Outgoing Communication
    Communication Handle  cmhMyRemoteBoundCommHandle
    Put Result in         i32OpenUnboundResult

open the local bound communication handle
Open Outgoing Communication
    Communication Handle  cmhMyLocalBoundCommHandle
    Put Result in         i32OpenLocalBoundResult
```

To find out if the handle opened successfully, inspect the result from the OpenOutgoingCommunication command. A zero result means the opening succeeded. A negative value indicates the communication handle was not opened.

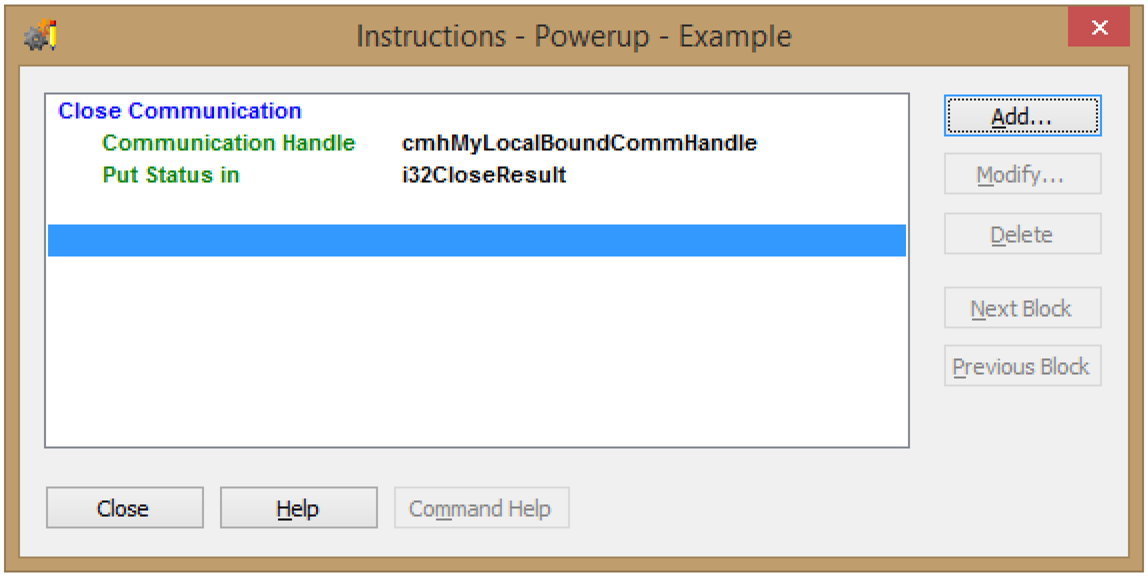
Closing the UDP Communication Handle

To close any communication handle, use the command "CloseCommunication."

OptoScript example:

```
CloseCommunication(cmhMyRemoteBoundCommHandle);
```

Action block example:



Close Communication
Communication Handle cmhMyLocalBoundCommHandle
Put Status in i32CloseResult

Many applications never close a communication handle once they are opened. If a communication fault were to occur on an unbound communication handle, the handle continues to remain open and may still be used to resume communications. If the communication handle was bound, it too remains open and datagrams will be received once the network problem is corrected.

Transmitting Datagrams with a Remote Bound Communication Handle

The following commands may be used to transmit data with an open UDP communication handle.

- Transmit Character*
- Transmit Numeric Table
- Transmit Pointer Table
- Transmit String
- Transmit String Table
- Transmit/Receive String

*If transmitting several sequential characters of a string, use Transmit String instead.

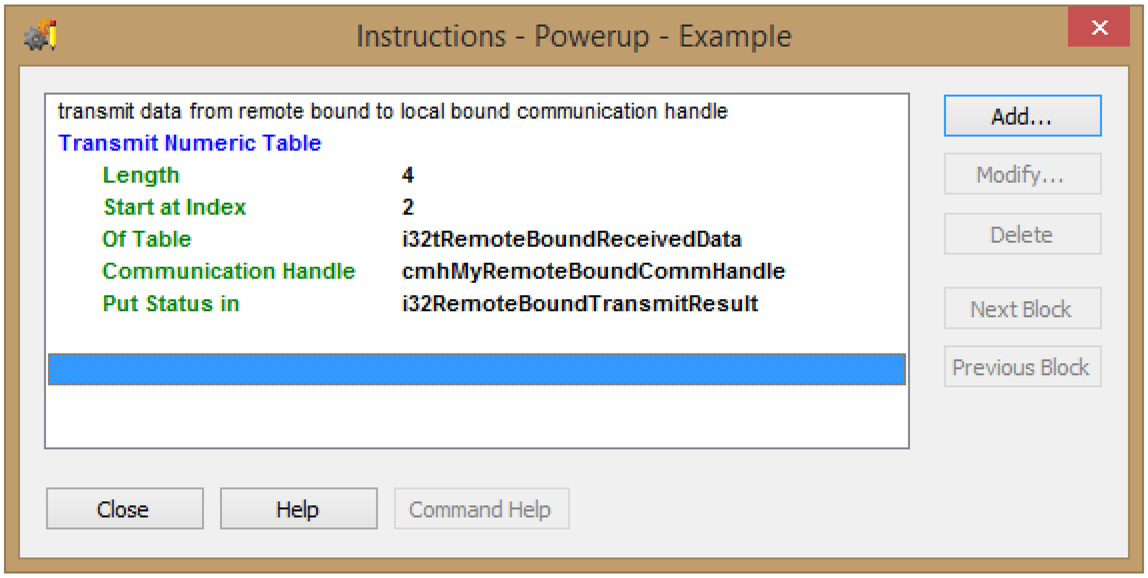
When using these commands, the data may be transmitted to the IP address and UDP port specified when the UDP communication handle was opened.

To add a communication handle to a PAC Control strategy, see the steps starting on page 283. In the Add Variable dialog box, enter the communication handle value in the Initial Value field.

OptoScript example:

```
i32TransmitResult = TransmitNumTable(4, 2, i32tMyDataTable,
cmhMyRemoteBoundCommHandle);
```

Action block example:



transmit data from remote bound to local bound communication handle
Transmit Numeric Table

Length	4
Start at Index	2
Of Table	i32tRemoteBoundReceivedData
Communication Handle	cmhMyRemoteBoundCommHandle
Put Status in	i32RemoteBoundTransmitResult

This example transmits elements 2, 3, 4, and 5 of table i32tMyDataTable to the IP address 192.168.1.100, UDP port 23456. All four elements are transmitted in a single datagram over the network.

Transmitting Datagrams with a Local Bound Communication Handle

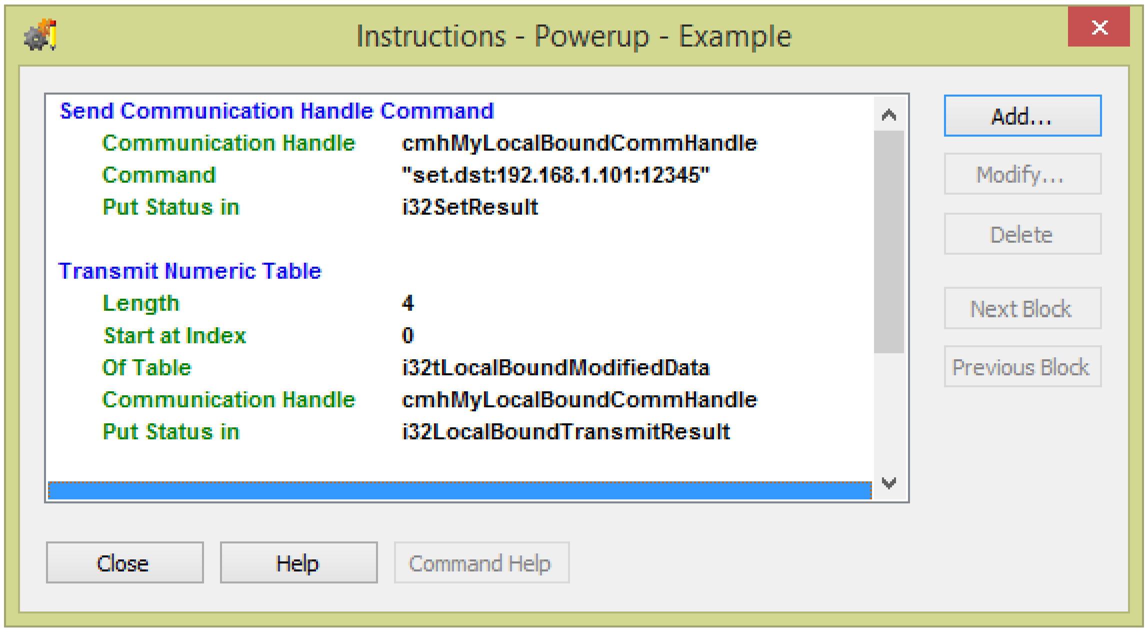
Transmitting with a bound communication handle requires the communication handle's destination (the IP address and port) to be set. Use the command "Send Communication Handle Command" to configure these destination parameters. Once the destination is set, the same data transmit commands may be used.

To add a communication handle to a PAC Control strategy, see the steps starting on page 283. In the Add Variable dialog box, enter the communication handle value in the Initial Value field.

OptoScript example:

```
i32SendResult = SendCommunicationHandleValue(cmhMyLocalBoundCommHandle,
"set.dst:192.168.1.101:12345");
i32TransmitResult = TransmitNumTable(4, 2, i32tMyDataTable,
cmhMyLocalBoundCommHandle);
```

Action block example:



```
Send Communication Handle Command
      Communication Handle       cmhMyLocalBoundCommHandle
      Command                    "set.dst:192.168.1.101:12345"
      Put Status in              i32SetResult

Transmit Numeric Table
      Length                     4
      Start at Index             0
      Of Table                   i32tLocalBoundModifiedData
      Communication Handle       cmhMyLocalBoundCommHandle
      Put Status in              i32LocalBoundTransmitResult
```

This example first configures the local bound UDP communication handle to transmit the data to the IP address 192.168.1.101, UDP port 12345. Subsequent data transmissions will be sent to the same IP address and UDP port until a SendCommunicationHandle with a new destination IP address or UDP port is commanded.

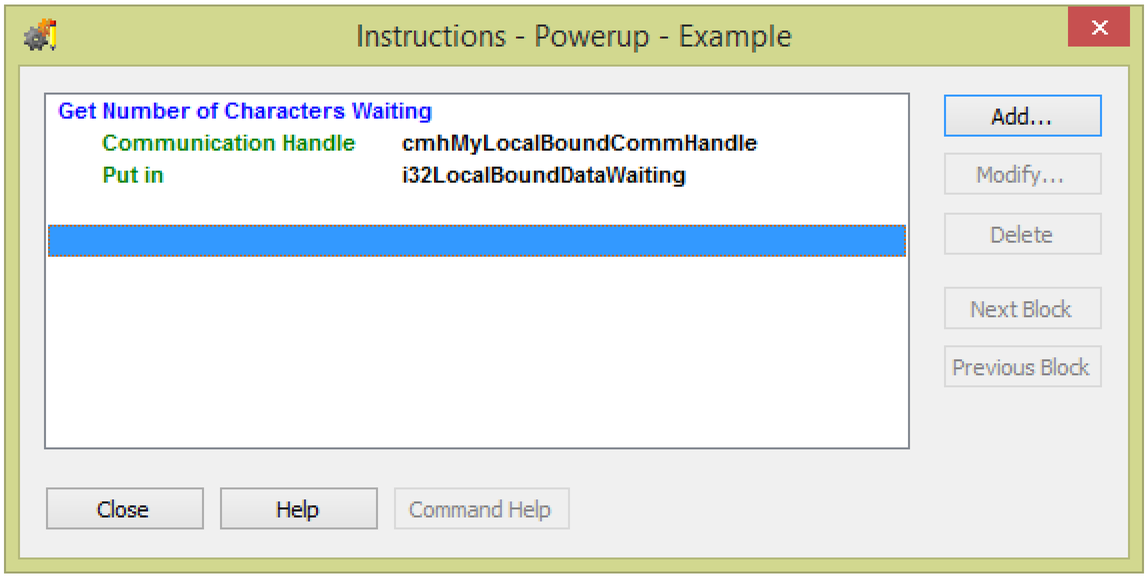
Determining when a Datagram Has Arrived

Use the command "Get Number of Characters Waiting" to determine if datagrams have arrived on the UDP communication handle.

OptoScript example:

```
i32BytesWaiting = GetNumberOfCharsWaiting(cmhMyLocalBoundCommHandle);
```

Action block example:



```
Get Number of Characters Waiting
    Communication Handle    cmhMyLocalBoundCommHandle
    Put in                  i32LocalBoundDataWaiting
```

The strategy should inspect the result value from the command. When positive, this indicates the amount of data pending. When receiving a positive result, the value indicates the amount of data contained in the current datagram.

In some cases, a timer is used to limit how long the chart will wait for data to arrive. This is usually configured as a loop and includes either an up- or down-timer to monitor the elapsed time. Within the loop, checking for the number of characters waiting, include a "Delay Millisecond" command with a reasonable delay time to allow the chart to release its timeslice.

Receiving Datagrams

Use the following commands to receive datagrams from a bound or unbound UDP communication handle.

- Receive Character
- Receive N Characters
- Receive Numeric Table
- Receive Numeric Table Ex
- Receive Numeric Variable
- Receive Pointer Table
- Receive String
- Receive String Table

When the chart logic has determined that a datagram is pending on the communication handle, the data may be received.

When receiving the datagram, ensure that all of the bytes contained within the current datagram reported by the command "Get Number Of Characters Waiting" are received. Once all the data of the current UDP datagram is received, the next datagram may be received. The reading of the entire datagram within PAC Control's implementation of UDP messaging differs from typical programming environment "socket" implementations.

Retrieving The Received Datagram's Source IP address And Source UDP port

When a datagram is received, the identity of the transmitter can be retrieved using the "Send Communication Handle Command" command. The command returns the values in a numeric format. To set a destination address and port so that a reply can be sent on the same communication handle, these values must be converted to strings and formatted into a command string understandable by Send Communication Handle Command.

OptoScript example:

```
// retrieve the datagram’s source IP address and source port
i32SourceIpAddress = SendCommunicationHandleCommand(cmhMyBoundCommHandle,
"get.src");
i32SourcePort = SendCommunicationHandleCommand(cmhMyBoundCommHandle,
"get.srcport");


// convert numeric source values to strings
Int32ToIpAddressString(i32SourceIpAddress, sSourceIpAddress);
NumberToString(i32SourcePort, sSourcePort);


// create a command string for the “Send Communication Handle Command”
sReplyAddress = “set.dst:” + sSourceIpAddress + ":" + sSourcePort;
```

The result stored in sReplyAddress (for example "set.dst:192.168.1.100:54321") is used as the command string to set the next transmission destination to send a datagram to IP address "192.168.1.100," port 54321.

Using the Control Engine's File System

The memory available for file storage is about 4 MB on a PAC S-series controller, or 2 MB on a PAC R-series controller or SNAP Ultimate brain, and 1 MB on a SNAP-LCE. File storage on a SoftPAC controller is limited only by the size of your hard drive and the volumes available on your network. Any types of files can be stored there, and files can be sorted into directories or folders just as they can on a PC. These stored files are then available for use; for example, the control engine can read them, add data to them, and even send data from them via FTP to another device on the network. SNAP PAC controllers also allow you to store data or files using a microSD card. For more information about using a microSD card, see the user's guide for the controller.

NOTE: Certain FTP commands may also be useful when dealing with files, even if the files are all local. For example, the dir command is available with comm handles.

The file communication handle is used to create, write to, and read from stored files on the control engine. The format for the handle's value is: `file:<open mode>,<filename>`

Note that `file` is all lowercase. Open modes are:

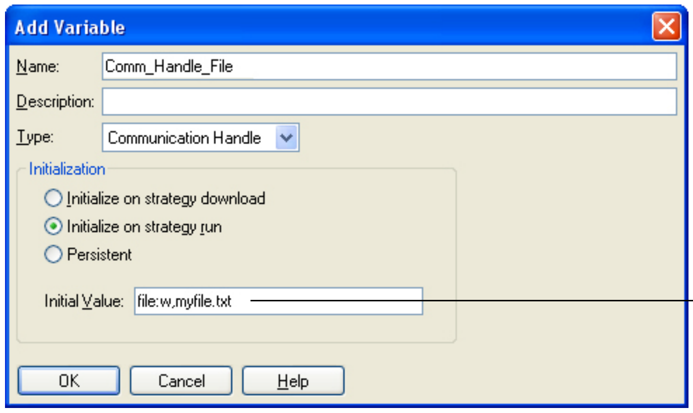
Open mode	Description
r	Opens a file for reading. If the file does not exist or cannot be found, the open call fails.

Open mode	Description
w	Creates a new file and opens it for writing. If the file already exists, its contents are destroyed.
a	Opens a file for writing at the end of the file (appending). If the file doesn't exist, it is created.

Here are some examples of file communication handle values:

Creates the file myfile.txt and opens it for writing.	`file:w,myfile.txt`
Opens the existing file myfile.txt so data can be appended to it.	`file:a,myfile.txt`
Creates the file Temperature data. txt in the directory Data_files and opens it for writing. If the directory doesn't exist, it is created.	`file:w,/Data_files/Temperature data.txt`
Opens the file Temperature data.txt in the directory Data_files for reading.	`file:r,/Data_files/Temperature data.txt`

To add a communication handle, see the steps starting on page 283. In the Add Variable dialog box, enter the communication handle value in the Initial Value field.



Name: Comm_Handle_File
Description:
Type: Communication Handle
Initialize on strategy download [✓ Initialize on strategy run] Persistent
Initial Value: file:w,myfile.txt

Keep the following limitations in mind as you use the file communication handle with the controller's file system (see below for microSD card):

Maximum length for file names and directory names	127 characters
Filename characters allowed	All ASCII characters except *, ?, null, and /
Path name component separator	/
Maximum number of files and directories that can be open simultaneously	16
Maximum directory depth	Limited only by available memory
Maximum number of files	Limited only by available memory. Each file uses 516 bytes of overhead plus its number of bytes rounded up to the nearest multiple of 516 bytes.

Maximum number of directories	Limited only by available memory. Each directory uses 516 bytes.
Maximum amount of memory available in the control engine's file system	Approximately 4 MB on a SNAP-PAC-S1, or 2 MB on a SNAP PAC R-series controller or SNAP Ultimate brain, or 1 MB on a SNAP-LCE controller (varies slightly depending on the control engine firmware version)

If power to the control engine is turned off, files are destroyed unless they have been saved to flash memory. See "Commands Relating to Permanent Storage" on page 327 for information on saving files to flash using PAC Control commands.

Using a File on a microSD Card

If your SNAP PAC controller has a microSD card slot in the top of the controller's case, you can use PAC Control commands with files on a microSD card just as you would with any other file in the controller's file system. Just remember two things:

- Include the card's directory name in the path in the file communication handle, for example: `file:a,/sdcard0/VoltLog.txt`
- Filename length: See your controller's user's guide for card compatibility information. Filenames up to 127 characters can be used on higher-capacity microSDHC cards. However, on lower-capacity standard microSD cards (2GB or less, formatted with FAT16), all files you store on the card must be named with a maximum of eight characters in the name plus three characters in the extension (8 dot 3 format), for example: `datafile.txt`

For more information on using the microSD card, see the controller's user guide.

Working with Files in Your Strategy

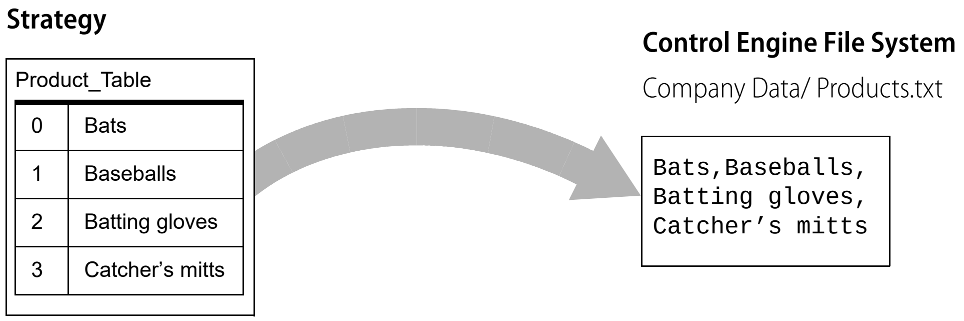
The commands you use with a communication handle vary according to the action (the open mode) defined in the handle's value. To change actions—from read to write, for example—you can use Set Communication Handle Value to change the handle's value.

Commands to use with any file communication handle	**Commands to use with file comm handles in Read mode**	**Commands to use with file comm handles in Write or Append mode**
Close Communication Communication Open? Get Available File Space Get Number of Characters Waiting Open Outgoing Communication Send Communication Handle Command *(delete, getpos, or setpos)* Set Communication Handle Value Set End-of-Message Terminator	Receive Numeric Table Receive Pointer Table Receive String Receive String Table Send Communication Handle Command (*find*)	Transfer N Characters Transmit Character Transmit NewLine Transmit Numeric Table Transmit Pointer Table Transmit String Transmit String Table

To work with files in your strategy, first use the command Open Outgoing Communication.

Writing to a File. For example, suppose you have data in your strategy in a string table (Product_Table) that you want to write to a file (Products.txt) in a directory (Company Data) on the control engine. Here's how you would do it:

1. Use Open Outgoing Communication to open a file communication handle. The value of the handle would be: `file:w,/Company Data/Products.txt`
2. Use the condition Communication Open? to make sure the communication handle opened.
3. To put the data from the string table into a comma-delimited file (which is easy to open in database software), use the command Set End-Of-Message Terminator to indicate that a comma should be used as the delimiting character. Or, if you want each line of your data to end with a LF (line feed), set the EOM (end-of-message) terminator to 10 (decimal), which is the ASCII code for LF. See "ASCII Table" on page 366 for other EOM characters.
4. Use the command Transmit String Table to transmit data from Product_Table directly into the Products.txt file. Items from the string table are separated by commas in the file.



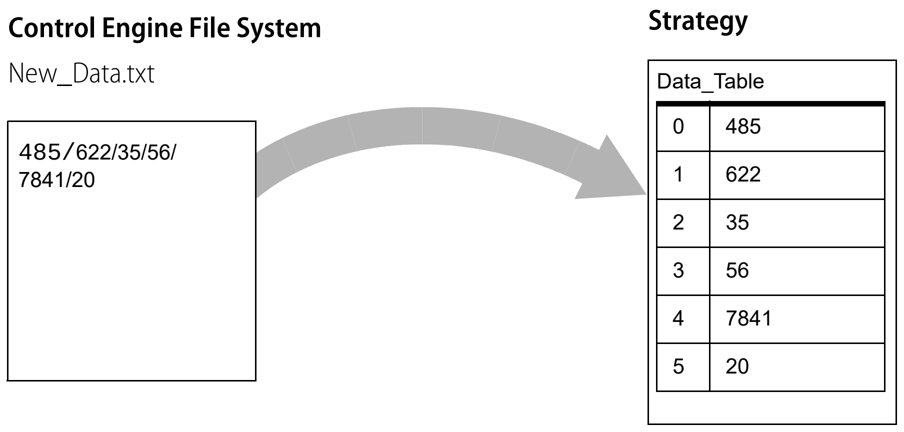
5. Use the command Close Communication to close the communication handle.

Reading a File. As another example, suppose you have a file on the control engine that was placed there via FTP. (See the next section for details on using the FTP communication handle.) You want to read this file (New_Data.txt) and place the data in it into a string table (Data_Table) in your strategy.

1. Use Open Outgoing Communication to open a file communication handle. The value of the handle would be: `file:r,New_Data.txt`
2. Use the condition Communication Open? to make sure the handle opened.
3. Use the command Set End-Of-Message Terminator to indicate what character in the New_Data.txt file should be read as the delimiting character. (In the example shown below, it's a slash.)

   TIP: If the lines of data in the text file end with LF (line feed), set the EOM (end-of-message) terminator to 10 (decimal), which is the ASCII code for LF.

4. Use the command Receive String Table to receive the data from New_Data.txt directly into Data_Table. To read the whole file, use a value of -1 for the Length parameter.



Notice the numbers used in this example. These are numbers represented as strings. For the purpose of storing and sending data, this is the simplest way to represent them. If you need to use them in calculations, however, you must first convert them to numeric values. You can do so in your PAC Control strategy by using a command such as Convert String to Float or Convert String to Integer 32. (See "String Commands" on page 358 for more information.)

5. It might also be helpful to use the command Send Communication Handle Command which provides the following commands that you can use when accessing files.

find: locates data in the file if each line has a unique identifier.

getpos returns an integer that indicates the current position in the file.

setpos:<position> jumps to the specified position within the file.

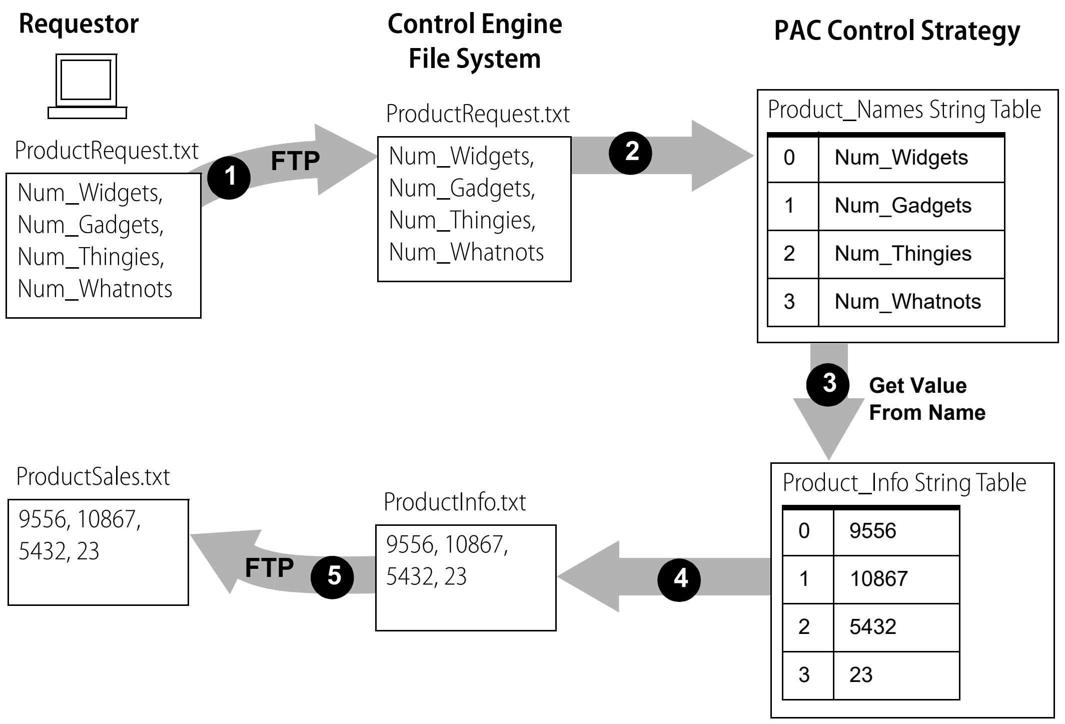
find:<mystring> (strings only) searches for the string within the file and returns its location as an offset from the current position in the file. The file must have been opened in r (read) mode."

6. Finally, close the communication handle by using the command Close Communication.

A More Complex Example. Here's a more complex example which shows the actual OptoScript code. In this example, someone on the network needs the value of several variables in the PAC Control strategy running on the control engine. This person has sent to the control engine via FTP a comma-delimited text file (ProductRequest.txt) containing the names of the variables. (See page 308 for information on FTP.)

This section of the PAC Control strategy reads the variable names from the text file in the control engine's file system and places the names in a string table (Product_Names). Next, the strategy uses the command Get Value From Name to place the *values* of the variables into another table (Product_Info). The data from this table is then written to another text file (ProductInfo.txt) on the control engine, which can later be sent via FTP to the person who requested the data.

The whole operation (including the FTP portions, which are not covered in this code example), might look like this:



The OptoScript code in this example also makes use of the command Set Communication Handle Value to change the value of a specific communication handle during the operation.

Sets the value for the communication handle chAFile.	`SetCommunicationHandleValue("file:r,ProductRequest.txt",chAFile);`

Opens the communication handle and checks to make sure it opened.	`status = OpenOutgoingCommunication(chAFile);`
	`if (status == 0) then`

Sets the end-of-message terminator to a comma because the file to be read is comma-delimited.	` SetEndOfMessageTerminator(chAFile, ',');`
Reads the contents of the file Product_Names into a string table.	` status = ReceiveStrTable(4,0,Product_Names,chAFile);`

Loops through the items in Product_Names table and places the values they represent into another string table, Product_Info. (Note that the "numbers" in the Product_Info table are not true numbers, but string representations of numbers.)	` index = 0;`
	`  while ((index < 4) and (status == 0))`
	`    status = GetValueFromName(Product_Names[index], Pro-`
	`duct_Info[index]);`
	`    index = index + 1;`
	`  wend`

Closes communication.	` status = CloseCommunication(chAFile);`
Changes the value of the communication handle; it is now set to write to the file ProductInfo.txt on the control engine.	` SetCommunicationHandleValue("file:w,ProductInfo.txt",chAFile);`

Opens the communication handle and checks to make sure it opened.	` status = OpenOutgoingCommunication(chAFile);`
	` if (status == 0) then`

Makes the ProductInfo.txt file comma-delimited, too.	`  SetEndOfMessageTerminator(chAFile, ',');`

Writes the data from the Product_Info string table into the ProductInfo.txt file on the control engine.	`  status = TransmitStrTable(4,0,Product_Info,chAFile);`
	` endif`
	`endif`

Deleting Files and Moving Within Them. Another command you'll find useful with file communication handles is Send Communication Handle Command. Using these commands, you can delete files, find a position within the file, and jump to a specific position within the file. See the *PAC Control Command Reference* or online Help for details.

For example, in order to find specific data in a file, you can use following commands within the command Send Communication Handle Command.

- **find:<mystring>**

  (for strings only) Use this command to search for a string within the file and return its location as an offset from the current position in the file. The file must have been opened in r (read) mode.
- **getpos**

  Returns an integer that indicates the current position in the file.
- **setpos:position**

  Jumps to the specified position within the file.

Moving Files via FTP

As explained in the previous section ("Using the Control Engine's File System," starting on page 301) the control engine's memory includes a substantial area available for file storage. You can move files to and from file storage using the File Transfer Protocol (FTP):

- **To move files to and from file storage using another device such as a PC,** use any standard FTP software. See instructions in Opto 22 form 1704, the PAC Manager User's Guide. A maximum of five devices can FTP files to a control engine simultaneously.
- **To move data to and from file storage programmatically,** a strategy can request or send a file using an FTP communication handle, explained in this section. A maximum of 16 communication handles can be used simultaneously to move data via FTP.

FTP also allows you to get a directory listing of files on a remote server or in the local file storage area. In addition, if your SNAP PAC controller has a microSD card slot in the top of the controller's case, you can use the FTP communication handle to manipulate files on a microSD card.

FTP Communication Handle Examples

The value for the FTP communication handle is in the format:

```
ftp:<IP address>:<port>,<username>,<password>,<optional timeout>
```

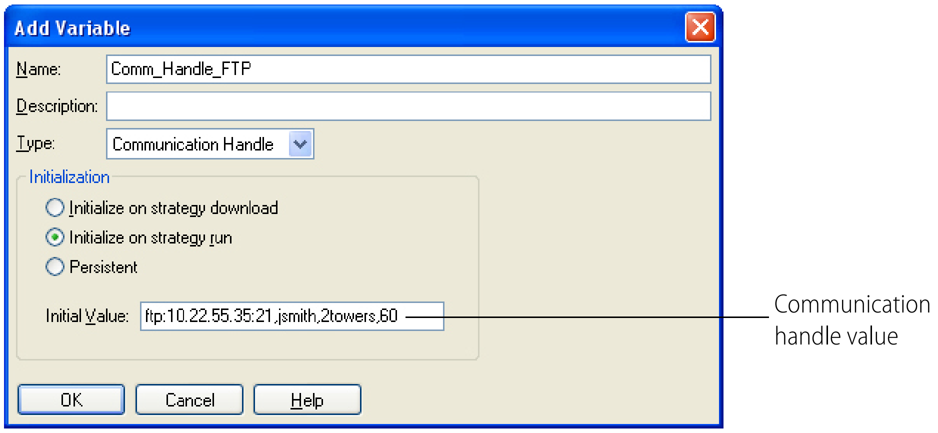
`ftp`	Use all lowercase letters.
`IP address`	IP address (or URL) of the destination device (where the file will go).
`port`	Default port is 21, with 20 for the data port.
`username` `password`	Enter the username and password set up for the destination device. If the destination device is another control engine or the local server, use anything except an empty string.
`optional timeout`	Specify a communication timeout value in seconds, or leave this parameter out to use the default of 30 seconds.

Here are two examples of FTP communication handle values:

`ftp:10.22.55.35:21,jsmith,2towers,60` (timeout increased to 60 seconds)

`ftp:10.192.54.195:21,m,m` (no timeout specified)

To add a communication handle, see the steps starting on page 283. In the Add Variable dialog box, enter the communication handle value in the Initial Value field.



Name: Comm_Handle_FTP
Description:
Type: Communication Handle
Initialize on strategy download
Initialize on strategy run [✓]
Persistent
Initial Value: ftp:10.22.55.35:21,jsmith,2towers,60

Setting the FTP Mode

If you are connecting to an FTP site behind a firewall, you may need to set a mode for the session. After the communication handle is open, you can set the FTP mode by issuing one of the following commands:

- For Active FTP:
  `i32ResultCode = SendCommunicationHandleCommand(cmh, "active");`
- For Passive FTP:
  `i32ResultCode = SendCommunicationHandleCommand(cmh, "passive");`

Using FTP Communication Handles in Your Strategy

Suppose you have data you need to send via FTP to a device on the network whose IP address is 10.192.56.45. Your user name on this device is JoeG and your password is hello. You expect the default timeout of 30 seconds to be adequate.

There are two ways you can send the data: all at once, or in pieces over time. If your data file is less than 1 MB in size, you can send it all at once. If it is larger, or if you want to append additional data to a file that already exists, you send it in pieces.

Sending or Retrieving the Data All At Once. This method is better for small data files; files larger than 1MB may take a long time to transfer, and the control engine may become unresponsive during the process.

For Sending: Suppose you want to send the data currently in the file SendThisData.txt on the control engine to a file HunkoData.txt on the device.

Here’s how you would send the data in your PAC Control strategy.

1. Use the command Open Outgoing Communication to open an FTP communication handle. The value of the handle would be: `ftp:10.192.56.45:21,JoeG,hello`
2. Use the condition Communication Open? to make sure the communication handle opened.
3. Use the command Send Communication Handle Command to specify the file name and send the file in one step.

Use the communication handle command **send:<local filename>, <remote filename>** as follows:

```
send:SendThisData.txt,HunkoData.txt
```

If the remote file name already exists, it is overwritten.

4. When you have finished sending data to the file, use the command Close Communication to close the communication handle.

For Retrieving: Suppose you want to get the data from the file GetThisData.txt on the device, and retrieve it to the file HunkoData.txt on the control engine.

Here's how you would retrieve the data in your PAC Control strategy.

1. Use the command Open Outgoing Communication to open an FTP communication handle. The value of the handle would be: `ftp:10.192.56.45:21,JoeG,hello`
2. Use the condition Communication Open? to make sure the communication handle opened.
3. Use the command Send Communication Handle Command to specify the file name and retrieve the file in one step.

   Use the communication handle command **get:<remote filename>,<local filename>** as follows:

   ```
   get:GetThisData.txt,HunkoData.txt
   ```

   If the local file name already exists, it is overwritten.
4. When you have finished retrieving data from the file, use the command Close Communication to close the communication handle.

Sending the Data in Pieces. To append data to an existing file, or to send a very large file (larger than 1 MB), you can send data in pieces.

Suppose you want to append data from a strategy variable or table to an existing file named HunkoData.txt on the device. Here's how you would send the data:

1. Use the command Open Outgoing Communication to open an FTP communication handle. The value of the handle would be: `ftp:10.192.56.45:21,JoeG,hello`
2. Use the condition Communication Open? to make sure the communication handle opened.
3. To specify the file name on the remote server, use the command Send Communication Handle Command. The communication handle command you send is:

   ```
   dest:HunkoData.txt
   ```
4. Use a Transmit command (such as Transmit String or Transmit Numeric Table) to send the data. In the command, use the name of the FTP communication handle you just opened.

   The data is appended to the file. (If the remote file name is new, the file is created and the data placed in it.)
5. When you have finished sending data, use the command Close Communication to close the communication handle.

If you are sending a large file from the control engine to the device, you would need to open up two communication handles: an FTP handle just like the one in the example above, and a File handle for the file on the control engine. Then use the Transfer N Characters command to send the file in chunks.

For another example, see the diagram in "A More Complex Example" on page 305, which shows how FTP communication handles and file communication handles might be used together. See the *PAC Control Command Reference* or online Help for detailed information on commands.

Retrieving a Directory Listing

The OptoScript code in this example makes use of the command Set Communication Handle Value and the dir option to retrieve a directory listing.

Sets the value for the communication handle chAFile.	`// Configure the chFTP comm handle to log into itself. The username and`
	`// password don't matter--they just can't be left empty. We'll use the`
	`// loopback address of 127.0.0.1 so this code is more portable.`
	`SetCommunicationHandleValue("ftp:127.0.0.1:21,noimporta,whocares", chFTP`
	` );`

Opens the communication handle and checks to make sure it opened.	`// Open the communication handle (log in to the local server)`
	`nStatus = OpenOutgoingCommunication( chFTP );`
	`if (nStatus == 0) then`

Requests directory listing, returns number of listings.	`  nStatus = SendCommunicationHandleCommand( chFTP, "dir" );`

Makes sure "dir" worked—firmware must be version 7.1 or greater.	`  if (nStatus >= 0) then`
	`    nFileCount = nStatus;`

Reads in listings to stList.
 	
NOTE: Each listing from an Opto 22 device server ends with these hex bytes: 0D 0A and is delineated from the next by a 00.
	`    // Set the EOM character to 00`
	`    SetEndOfMessageTerminator( chFTP, 0x00 );`
	`    nStatus = ReceiveStrTable( nStatus, 0, stList, chFTP );`
	
	`    if ( nStatus == 0 ) then`

Optional: parses each listing into separated tables for:
1. date/time stamp (first 17 characters)
2. file size (next 21 characters)
3. file name (remaining characters) and removes the 0D 0A left over on the end of the file name.
	`      for nIndex = 0 to (nFileCount - 1) step 1`
	`        GetSubstring( stList[nIndex], 0, 17, stDateTimeStamps[nIndex] );`
	`        GetSubstring( stList[nIndex], 18, 21, sTemp );`
	`        ntSizes[nIndex] = StringToInt32( sTemp );`
	`        GetSubstring( stList[nIndex], 39, 1000, stFilenames[nIndex] );`
	
	`        nLength = GetStringLength( stFilenames[nIndex] );`
	`        GetSubstring( stFilenames[nIndex], 0, nLength - 2,`
	`                      stFilenames[nIndex] );`
	`      next`

Further parsing could be done on the date/time stamps, depending on how those values will be used.	`    endif`
	`  endif`
	`endif`
	`CloseCommunication( chFTP );`

Using Serial Communication Handles to Communicate with Serial Devices

Serial connectors are located on the top of SNAP PAC controllers and brains, SNAP-LCE controllers, and SNAP Ultimate controller brains. These ports can be used for maintenance, such as loading new firmware, or for Point-to-Point Protocol (PPP) communication via a modem. They can also be used to send or receive data directly from a serial device, such as barcode readers, weigh scales, I/O, or any intelligent device. Serial communication handles are used to communicate with these intelligent devices (see examples below).

When using the (faster) on-board serial connections available on PAC R- and PAC S-series controllers, the communication handle value start with "ser:" followed by the baud settings. If you're using a PAC R-series controller, you'll need to configure the onboard Port 0 to be None rather than the default of PPP; otherwise, the Open command will fail with a -20 (Resource busy) error.

To set the serial port to None in PAC Manager:

1. Open the Inspect window in PAC Manager and enter the IP address for your device.
2. Click Communications and choose Communication Port Control.
3. In the Value field for the Control Function of the serial port you want to use, select None from the drop-down menu.

   NOTE: For a SNAP-PAC-S2, you can configure any of the serial ports as a either RS-232 or RS-485 by changing the Value field for the Mode for Communication parameter.

For more detailed information on setting serial communications, see Opto 22 form 1704, the *PAC Manager User's Guide*. On a SNAP-LCE or SNAP Ultimate brain, firmware version 5.1c or newer is required.

IMPORTANT: *Serial communication handles are used only for direct connection to serial devices. If you are connecting to serial devices through serial communication modules on the I/O unit, use a TCP communication handle instead. See* page 283.

NOTE: Serial I/O units are wired to an RS-485 port on the controller and do not require serial communication handles. Communication is defined when the I/O unit is configured.

NOTE: Be sure to use a single communication handle for each of a controller's physical serial ports used with the subroutines. If two charts were to run simultaneously while sharing an open communication handle, it is possible for one serial port to transmit data (on the wire) at the same time the remote device is transmitting a response.

Serial Communication Handle Examples

To enter a serial communication handle, you can use either of the following two methods for SNAP PAC and SNAP-LCE controllers. For SNAP Ultimate controllers, use Method 1 only.

Method 1. With this method, the value for the serial communication handle is in the format:

```
ser:<port number>,<baud rate>,<parity>,<data bits>,<stop bits>
```

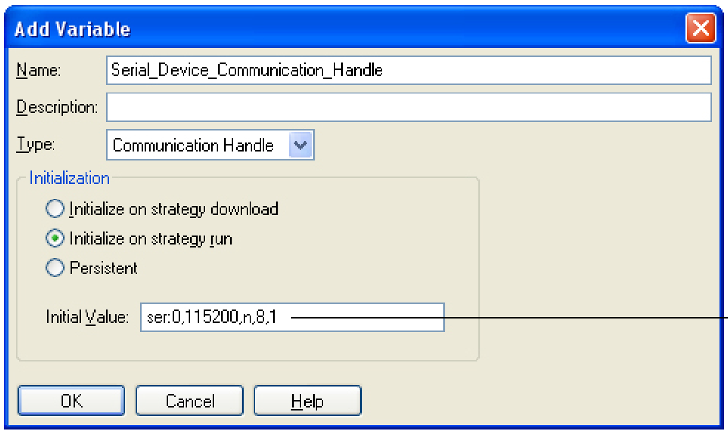
Setting	Valid Values
port number	UIO: 0 or 1 SNAP-PAC-S1: 0, 1, or 2 SNAP-PAC-S2: 0, 1, 2, or 3 SNAP-PAC-R1, R1-B, and R2: 0
baud rate	230400*, 115200, 76800, 57600, 38400, 19200, 9600, 4800, 2400, 1200, or 300
parity	n, o, or e (none, odd, or even)
data bits	8 or 7
stop bits	1 or 2

*230,400 kBd is not available on SNAP Ultimate controllers.

Here is an example of a serial communication handle value:

`ser:0,115200,n,8,1` (port 0, baud rate of 115,200, no parity, 8 data bits, and 1 stop bit)

To add a communication handle, see the steps starting on page 283. In the Add Variable dialog box, enter the communication handle value in the Initial Value field.

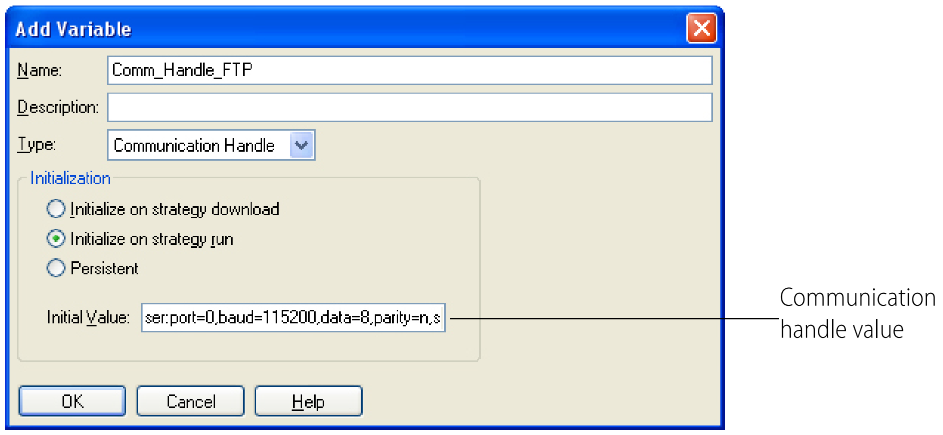


Name: Serial_Device_Communication_Handle
Description:
Type: Communication Handle
Initialization
( ) Initialize on strategy download
(•) Initialize on strategy run
( ) Persistent
Initial Value: ser:0,115200,n,8,1

Method 2. (SNAP PAC only) Using keywords, this method offers more flexibility. For example, use this method if you want to specify RTS / CTS on a serial port on a SNAP S-series or R-series controller. Keywords may be used in any order. Each keyword must be lowercase and followed by an equal sign and a value, with no spaces (even after commas), as shown in the following examples:

```
ser:port=0,baud=115200,data=8,parity=n,stop=1,rts-cts=0
ser:baud=115200,data=8,parity=n,rts-cts=1,stop=1,port=0,timeout=1.5
```

In the Add Variable dialog box, enter the communication handle value in the Initial Value field. (Due to the long value, part of the value is hidden.)


Add Variable

Name: Comm_Handle_FTP

Description:

Type: Communication Handle

Initialization

- Initialize on strategy download
- [✓ Initialize on strategy run]
- Persistent

Initial Value: ser:port=0,baud=115200,data=8,parity=n,s

NOTE: You must include at the minimum *the port number and one parameter. Otherwise, it will not work.*

The keywords and possible values are:

Keyword	Valid Values	Comments
port	0, 1, 2, 3	The ports available depends on the controller type. For a list of available ports, see the documentation that came with your controller.
baud	230400*, 115200, 76800, 57600, 38400, 19200, 9600, 4800, 2400, 1200, or 300	* 230400 is not available on SNAP Ultimate controllers.
parity	E, e, O, o, N ,n, 9*	* A value of 9 is used for RS-485 multi-drop communications.
data	8, 7	The number of data bits
stop	1, 2	The number of stop bits
mode	0, 522*	* 522 enables I/O mode. 0 should be used for everything else.
rts-cts	0, 1, 2	0 turns rts/cts off. 1 enables symmetrical (full-duplex) rts-cts flow control. 2 enables asymmetrical (half-duplex) rts-cts flow control. By default the system waits up to one second for cts to be asserted.
timeout	float, seconds	

If a parameter is not specified, the current value is used, so you should include any parameter whose current value you are not sure of.

IMPORTANT*: As described earlier in this section, be sure to use PAC Manager to set the control function of any port you want to use to None. Otherwise, you will get a -20, (Resource Busy), error. For more information, see Opto 22 form 1704, the* PAC Manager User's Guide.

Using Serial Communication Handles in Your Strategy

To use a serial communication handle in your strategy, first use the command Open Outgoing Communication. Verify that the communication handle opened by using the condition

Communication Open? Then use Transmit, Transfer, and Receive commands to send or receive data as necessary. Remember to check for any errors. When you have finished sending and receiving data, use the command Close Communication to close the communication handle.

Make sure your strategy receives data promptly. Incoming serial communication is buffered up to 127 characters. If more than 127 characters come in before the strategy receives them, the additional characters are lost.

See Opto 22 form 1701, the PAC Control Command Reference, or online Help for details on using specific commands.

Sending an Email

Use the email commands to have a strategy send an email automatically from your control engine to recipients via SMTP.

- **Send Email** sends a text-only email.
- **Send Email with Attachments** sends text and one or more attached files such as log files, status reports, and images.

The following examples are just a few of the many possible uses for these commands:

- Send out updates of a system's status.
- Have a critical system send hourly status emails; if no email arrives after an hour, it means something is wrong.
- Send an email in case of an alarm condition.
- Send a daily status report.

For more information on these commands, see Opto 22 form 1701, the PAC Control Command Reference.

A sample chart to send email is available on our website, www.opto22.com. Go to Support > Downloads and then search for Sample PAC Control Basic Chart to Send Email in Samples and Freeware.

Before you begin:

- Make sure to assign to the controller an IP address as well as the appropriate Subnet Mask, DNS, and Gateway for the network. For more information, see Opto 22 form 1704, the PAC Manager User's Guide.
- The Send Email commands require you to use an existing email account. You may be able to use a corporate email account, but check with your network administrator first. Your SMTP server must support either AUTH LOGIN or AUTH PLAIN authentication (neither AUTH MD5 nor AUTH CRAM-MD5 can be used with the Send Email commands). Or, you can use a free email account such as Gmail™ or Yahoo!® Mail.

  If you will be using a *corporate email account*, ask your network administrator for the following information about the SMTP server:

  – **Outgoing SMTP server name or IP address**

  – **Port**

  – **Security type**: Choices are *ssl*, *tls*, or *none*

    SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are standard security technologies for establishing an encrypted link between a web server and a browser.

If no root certificate is required for the connection, select "None" and the connection will not be authenticated. If the target mail server doesn't require SSL or TLS, the login will succeed. However, if it requires SSL or TLS, the login will fail. Most servers require encryption, so unless you're using an in-house server, you'll probably need a root certificate.

– **Root certificate** (if applicable): A root certificate, which is used to validate an SSL or TLS secure connection, may be required for your SMTP server. If so, you must obtain the root certificate and then install it on your SNAP PAC controller as described below.

Installing CA Root Certificates

NOTE: You do not need to install a root certificate:

– *To send email using a SoftPAC controller. In this case, the necessary certificates are preinstalled with Windows.*

– *When your controller has PAC firmware R9.5a (or higher) and the server you are communicating with uses a certificate derived from an Equifax or Digicert certificate. In this case, the PAC firmware includes built-in certificates for Equifax and Digicert.*

An installed *security certificate* is required if you want to have secure network communications using SSL/TLS Internet protocols. A security certificate is a digital tool used by Web browsers and other network services to validate the identity of the server your computer is connecting to. For example, before you access email or make a payment over an HTTPS Internet connection, the server and your computer verify their identities by exchanging information about their security certificates. If something is wrong with the certificates, your browser typically displays a warning that the connection is not secure.

A *CA certificate* is a type of security certificate that is issued by a Certificate Authority, which is a trusted entity (for example, Digicert® and GoDaddy®) that issues legitimate digital security certificates. A *root* certificate is a kind of security certificate that ensures the certificate's authenticity by identifying the root CA (that is, the original source of the certificate's verification). Root certificates use a special kind of cryptography known as *public key infrastructure*.

PAC firmware R9.5a and higher include certificates for Equifax and Digicert (two of the most commonly used CA root certificates for email service providers, including Google Gmail™ and Yahoo! Mail®). This means that if you are using PAC firmware R9.5a (or higher) and PAC Manager R9.5a (or higher), you don't have to install Gmail's or Yahoo! Mail's CA root certificates to send emails to these two email service providers. However, to use a SNAP PAC device (as a client) to make a connection to other secure devices via a TCP communication handle, or to use an HTTPS Get or HTTPS Post communication handle, you can easily install a CA root certificate using PAC Manager.

Before you can install a CA root certificate, you must:

1. Have a CA root certificate that uses Base64 ASCII format. (Your company's IT department should be able to provide this, as they will know the server it needs to match. For example, if the certificate is for email, it must match the email server.)

   – You can tell a certificate file is Base64 ASCII format if:

     – The file has a .PEM extension; or

     – You open the file in a text editor (such as Notepad) and you see the line:

   `"-----BEGIN CERTIFICATE-----"`

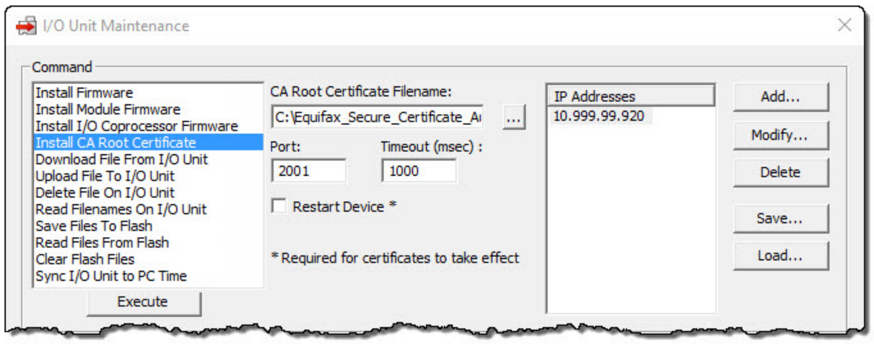
SNAP PAC controller can accept only one certificate per file. You can tell whether there is only one certificate in a file by opening the file in a text editor and verifying there is only one line that says, “`-----BEGIN CERTIFICATE-----`”

2. Know where to locate the certificate on your network (that is, its path and filename).

Install a CA root certificate

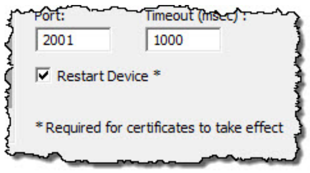
To use these instructions, you must have PAC firmware R9.5a (or higher) and PAC Manager R9.5a (or higher). If you don't, see page 317.

1. If PAC Manager isn't already open, start it:
   - In Windows 7 and Windows Vista, press the Windows Start key, and then click Programs > Opto 22 > PAC Project 9.6 > PAC Manager.
   - In Windows 10 and Windows 8.1, press the Windows Start key, type `PAC Manager 9.6` and then press the Enter key.
2. In the PAC Manager menu bar, click Maintenance .
3. In the Command list, select Install CA Root Certificate.
4. Click the Browse button and browse to the location of the certificate. Select the certificate, and then click OK.



CA Root Certificate Filename: C:\Equifax_Secure_Certificate_Au
Port: 2001
Timeout (msec) : 1000
Restart Device *
* Required for certificates to take effect
IP Addresses
10.999.99.920

5. Select the IP address of the SNAP PAC device that you want to install the certificate on.
6. (Optional) Change the Port number and the Timeout value.
7. Select Restart Device, and then click Execute to install the certificate, reboot the device, and activate the certificate.



Port: 2001
Timeout (msec) : 1000
☑ Restart Device *
* Required for certificates to take effect

Repeat steps 3 through 7 if you have additional certificates to install.

Install a CA root certificate (PAC Firmware R9.4c and lower / PAC Manager R9.4c and lower)

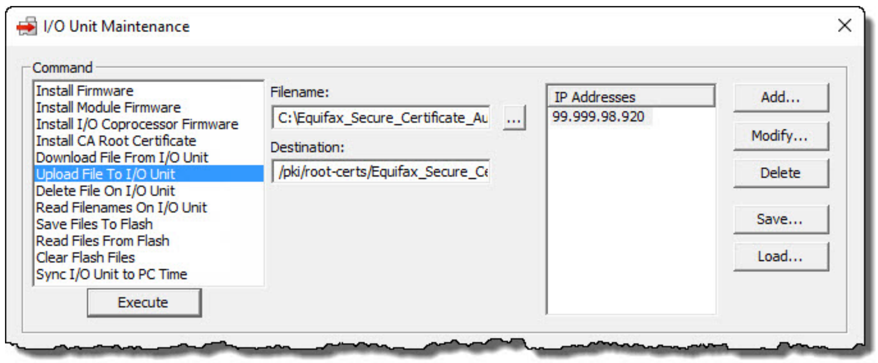
To place a root certificate in the /pki/root-certs folder on your controller:

1. If PAC Manager isn't already open, start it:

- In Windows 7 and Windows Vista, press the Windows Start key, and then click Programs > Opto 22 > PAC Project 9.6 > PAC Manager.
- In Windows 10 and Windows 8.1, press the Windows Start key, type `PAC Manager 9.6` and then press the Enter key.

2. In the PAC Manager menu bar, click Maintenance.
3. In the Command list, select Upload File To I/O Unit.
4. Click the Browse button and browse to the location of the certificate. Select the certificate, and then click OK.
5. In the Destination field, type /pki/root-certs/ and the name you want the certificate to have on the controller; for example:

   `/pki/root-certs/Equifax_Secure_Certificate_Authority.cer`
6. Select the IP address of the SNAP PAC device that you want to install the certificate on.



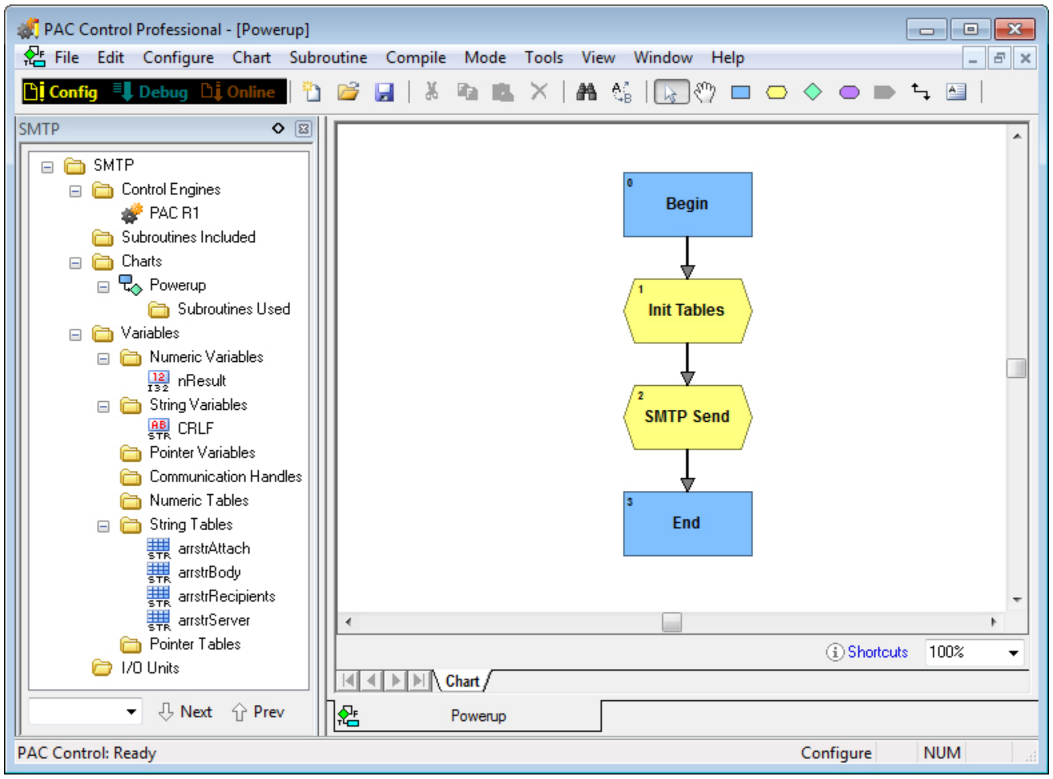
Filename: C:\Equifax_Secure_Certificate_Au
Destination: /pki/root-certs/Equifax_Secure_Ce
IP Addresses
99.999.98.920

7. Click Execute.
8. Repeat steps 2–6 if you have additional certificates to install.
9. Select the Save Files To Flash command, and then click Execute.
10. Restart the SNAP PAC device to activate the certificates.

Send Email with Attachments Example

The simple example strategy below demonstrates how to use OptoScript to create and send an email using the Send Email with Attachments command. You can of course use the PAC Control versions of the Send Email commands. However, with OptoScript you can more easily see and edit each of the email components—the Server information, Recipients, Message Body, and Attachment File Names. For more information on these commands, see Opto 22 form 1701, the PAC Control Command Reference.

A sample chart to send email is available on our website, www.opto22.com. Go to Support > Downloads and then search for Sample PAC Control Basic Chart to Send Email in Samples and Freeware.



The strategy is comprised of only one chart, the Powerup chart, which has two OptoScript blocks, Init Tables and SMTP Send. The first OptoScript block, Init Tables, contains the email body. The second one, SMTP Send, uses the Send Email with Attachments command to send the email and attachments.

To send an email and attachments using the example:

1. Create a new strategy and then choose Configure > Control Engine to configure a control engine. For more information, see "Associating the Control Engine with Your Strategy" on page 103.
2. Create a numeric variable and a string variable as follows. For more information, see "Adding Variables" on page 257.

Name	Type
nResult	Integer32

3. Create a string variable as follows.

Name	Width
CRLF	256

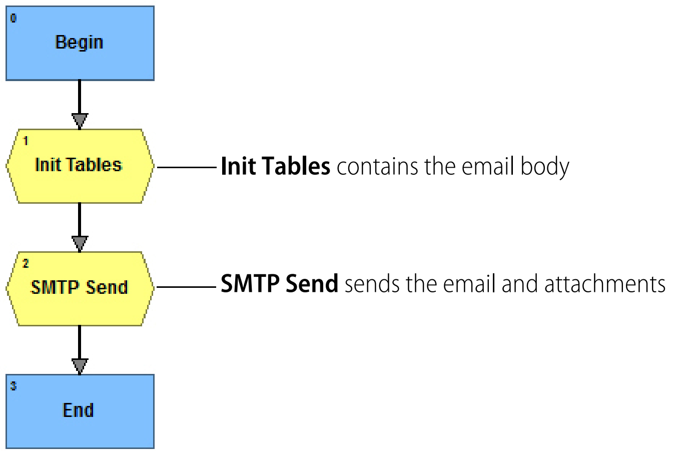
4. Create the following string tables. For more information, see "Adding Tables" on page 260.

Name	Table Length	String Width
arrstrAttach	10	256
arrstrBody	100	1024
arrstrRecipients	10	256
arrstrServer	10	256

5. To the Powerup chart, add an Action Block followed by two OptoScript Blocks and then one more Action Block.

Position	Name	Type	Function
First	Init Tables	Action	Start
Second	Init Tables	OptoScript	Contains the email body
Third	SMTP Send	OptoScript	Sends the email and attachments
Fourth	End	Action	End

6. Connect the blocks starting with the first Action Block.

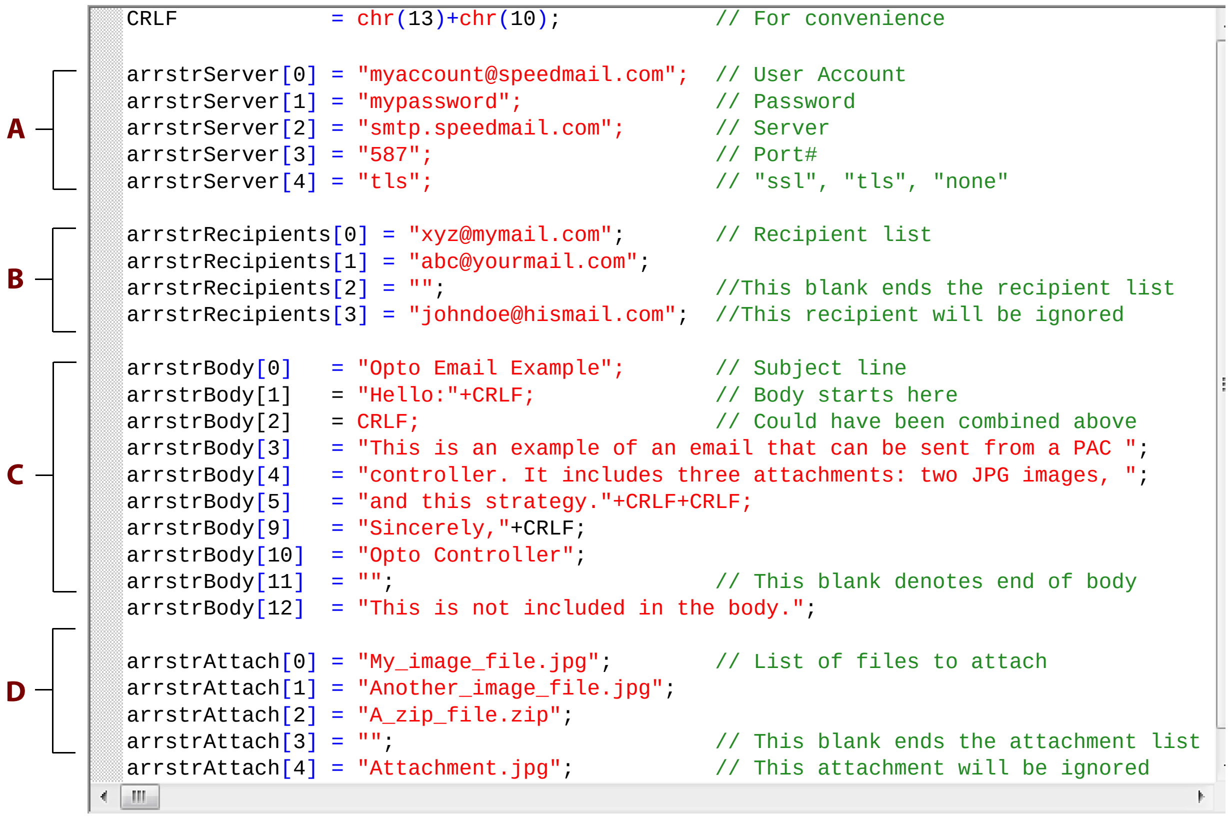

0
Begin
1
Init Tables
Init Tables contains the email body
2
SMTP Send
SMTP Send sends the email and attachments
3
End

7. Add code to the InitTables OptoScript Block.

   This block creates the contents of the string tables which includes server information, a recipient list, the body of the email, and a list of attached files.

   The first line in the block defines CRLF as a carriage-return / linefeed combination, chr(13)+chr(10), which starts a new line in the email body. Defining this value allows you to use CRLF instead of having to spell out the chr(13)+chr(10) combination each time. For a double carriage-return / linefeed combination, use CRLF+CRLF.

   NOTE: If multiple emails with the similar subject and body text are sent through the same email account, the emails may be flagged as SPAM by the email service provider and not be delivered. To *avoid this, you may need to add code to the subject and body text that changes the text dynamically with each new email.*



```
CRLF              = chr(13)+chr(10);            // For convenience

arrstrServer[0] = "myaccount@speedmail.com";  // User Account
arrstrServer[1] = "mypassword";               // Password
arrstrServer[2] = "smtp.speedmail.com";       // Server
arrstrServer[3] = "587";                      // Port#
arrstrServer[4] = "tls";                      // "ssl", "tls", "none"

arrstrRecipients[0] = "xyz@mymail.com";       // Recipient list
arrstrRecipients[1] = "abc@yourmail.com";
arrstrRecipients[2] = "";                     //This blank ends the recipient list
arrstrRecipients[3] = "johndoe@hismail.com";  //This recipient will be ignored

arrstrBody[0]   = "Opto Email Example";       // Subject line
arrstrBody[1]   = "Hello:"+CRLF;              // Body starts here
arrstrBody[2]   = CRLF;                       // Could have been combined above
arrstrBody[3]   = "This is an example of an email that can be sent from a PAC ";
arrstrBody[4]   = "controller. It includes three attachments: two JPG images, ";
arrstrBody[5]   = "and this strategy."+CRLF+CRLF;
arrstrBody[9]   = "Sincerely,"+CRLF;
arrstrBody[10]  = "Opto Controller";
arrstrBody[11]  = "";                         // This blank denotes end of body
arrstrBody[12]  = "This is not included in the body.";

arrstrAttach[0] = "My_image_file.jpg";        // List of files to attach
arrstrAttach[1] = "Another_image_file.jpg";
arrstrAttach[2] = "A_zip_file.zip";
arrstrAttach[3] = "";                         // This blank ends the attachment list
arrstrAttach[4] = "Attachment.jpg";           // This attachment will be ignored
```

A—Server Information. The server table *must* have each of these five entries in the order shown in the example. Make sure all server information table elements are filled in.

Here are the ways that you can specify the account and "from" names in element 0:

If you want this:	Use this:
No user account and no "from"	`arrstrServer[0] = "";`
User Account only, ("from" uses the account name)	`arrstrServer[0] = "MyAccount";`
Account only (no "from")	`arrstrServer[0] = "MyAccount:";`
"from" only (no user account)	`arrstrServer[0] = ":FromName";`
Separate account and "from" names	`arrstrServer[0] = "MyAccount:FromName";`

If you want to change the **timeouts** for transmitting, receiving, and connecting, you need to add a sixth element [5]. The default is 10,000 milliseconds (10 seconds) for each. For more information, see "Setting Timeouts for the Send Email Commands" on page 323.

For a **corporate email** account, use the server, port, and security values for the SMTP server you obtained previously from your network administrator.

If no root certificate is required for the connection, use "none" for element 4 of the server information table and the connection will not be authenticated. If the target mail server doesn't require SSL or TLS, the login will succeed. However, if it requires SSL or TLS, the login will fail. Most servers require encryption, so unless you're using an in-house server, you'll probably need a root certificate.

PAC firmware R9.5a and higher includes root certificates for both Gmail and Yahoo! Mail. For details, see "Installing CA Root Certificates" on page 316.

If you aren't using PAC firmware R9.5a (or higher) and PAC Manager R9.5a (or higher), you can use these values for Gmail and Yahoo! Mail:

– For a **Gmail** account, use these values:

Server	smtp.gmail.com
Port	587
Security	tls

– For a **Yahoo! Mail** account, use these values:

Server	smtp.mail.yahoo.com
Port	465
Security	ssl

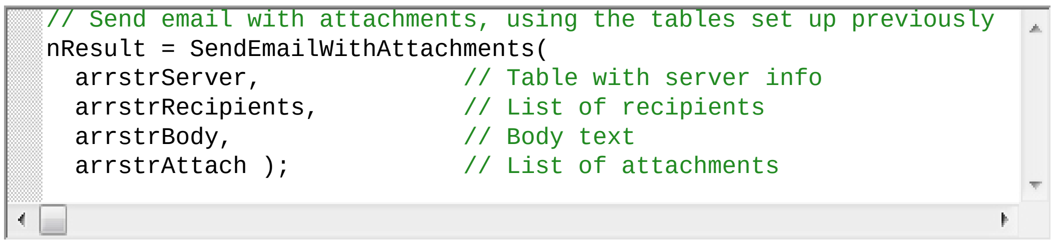
B— Recipients. Enter the email address for each recipient. Make sure the recipients list contains at least one valid entry.

C—Message Body. Element zero is always the subject line. Each line is entered as a separate string. However, they will be concatenated in the body of the email. The email body starts with element one, and continues to the first empty element or the end of the table, whichever comes first. Note that a blank line is not the same as an empty element.

D—Attachment File Names. Files specified in the attachments table must exist on the controller's file system, (RAM, flash, or SD card), and must include the full directory specification. For example, if a file is in the root directory, just use the file name. However, if it is in the temp directory of the SD card, you must put /sdcard0/temp/ in front of the file name. Any kind of file may be attached, so long as there is room in the file system. However, be cautious about file size, many Internet service providers still have a 10 MB attachment limit.

8. Add code to the SMTP Send OptoScript block.

This block invokes the Send Email with Attachments command to send the email with attachments:



```
// Send email with attachments, using the tables set up previously
nResult = SendEmailWithAttachments(
  arrstrServer,              // Table with server info
  arrstrRecipients,          // List of recipients
  arrstrBody,                // Body text
  arrstrAttach );            // List of attachments
```

9. Click Debug, and then click Run Strategy.

The email and attachments are sent to the recipient(s) in your list.

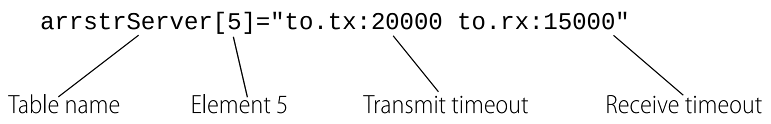
Setting Timeouts for the Send Email Commands

The default Send Mail commands timeout for transmitting, receiving, and connecting is 10,000 milliseconds (10 seconds) when you have filled in elements 0–4 of the Server Information table (see page 321). If you want to change one or more timeouts from the default, add an element 5 and specify the value as follows:

To do this:	Use this command where "nnnn" is the timeout in milliseconds
Set transmit timeout	"to.tx:nnnn"
Set receive timeout	"to.rx:nnnn"
Set connect timeout	"to.cx:nnnn"

Each timeout command is independent of the others so you can use one, two, or all three of the commands in any order. If you use more than one command, put a space between each one.

The following example sets the transmit timeout to 20 seconds and the receive timeout to 15 seconds. The connect timeout is not changed.



```
arrstrServer[5]="to.tx:20000 to.rx:15000"
```

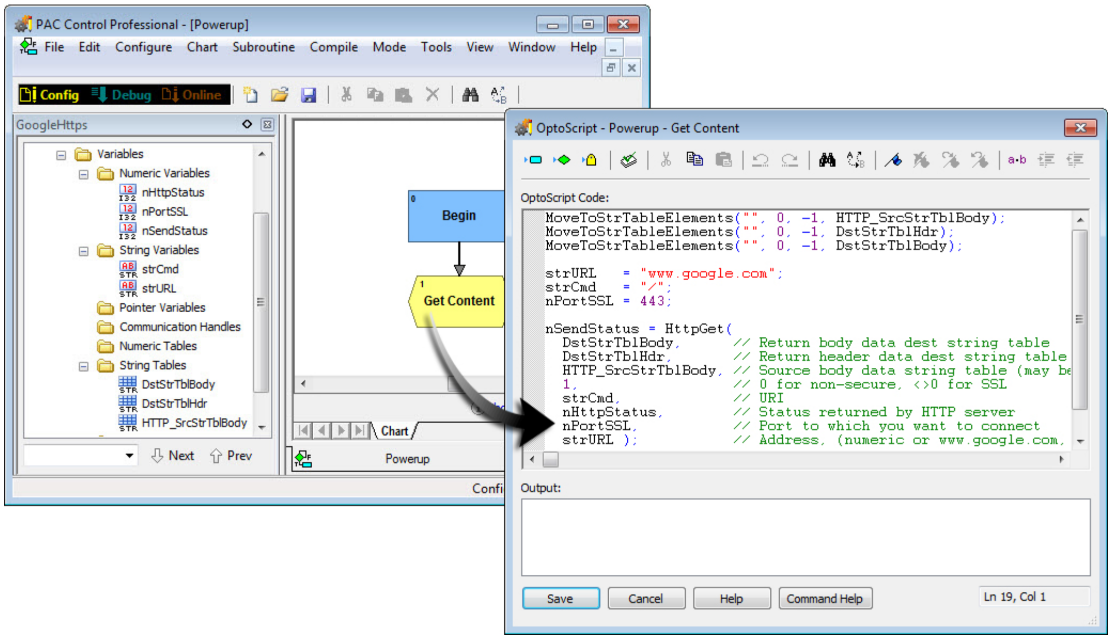
Retrieving a Webpage

Use the HTTP Get command to retrieve non-secure (HTTP) or secure (HTTPS) content from a web page. The example provided here uses the HTTP Get command in OptoScript to retrieve secure content from www.google.com. For more information on this command, see Opto 22 form 1701, the PAC Control Command Reference.

To retrieve HTTPS content from a website, you will first need to:

- Obtain the root certificate used by the target website to validate an SSL secure connection. (Your company's IT department should be able to provide this, as they will know the server it needs to match.)
- Install the root certificate on your SNAP PAC controller. For instructions, see "Installing CA Root Certificates" on page 316.

NOTE: If you are using a SoftPAC controller, you don't need additional root certificates to use HTTP Get— the necessary certificates are preinstalled with Windows.



```
MoveToStrTableElements("", 0, -1, HTTP_SrcStrTblBody);
MoveToStrTableElements("", 0, -1, DstStrTblHdr);
MoveToStrTableElements("", 0, -1, DstStrTblBody);

strURL   = "www.google.com";
strCmd   = "/";
nPortSSL = 443;

nSendStatus = HttpGet(
  DstStrTblBody,       // Return body data dest string table
  DstStrTblHdr,        // Return header data dest string table
  HTTP_SrcStrTblBody,  // Source body data string table (may be
  1,                   // 0 for non-secure, <>0 for SSL
  strCmd,              // URI
  nHttpStatus,         // Status returned by HTTP server
  nPortSSL,            // Port to which you want to connect
  strURL );            // Address, (numeric or www.google.com,
```

For this example, you'll need to install the Equifax Secure Certificate Authority (DER encoded X.509) root certificate, which is available on our website, www.opto22.com. (Go to Support > Downloads and then search for Equifax Root Certificate in Samples and Freeware.) For instructions to install the certificate, see page 316.

1. Make sure the controller is assigned an IP address as well as the appropriate Subnet Mask, DNS, and Gateway for the network. For more information on how do this, see Opto 22 form 1704, the PAC Manager User's Guide.
2. In PAC Control, create a new strategy.
3. Create the following numeric (Integer 32) variables set to "Initialize on strategy run." For more information, see "Adding Variables" on page 257.

Variable Name	Type
nHttpStatus	Integer32
nPortSSL	Integer32
nSendStatus	Integer32

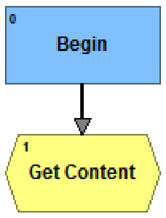
4. Create the following string variables.

Variable Name	Type	Width
strCmd	String	100
strURL	String	256

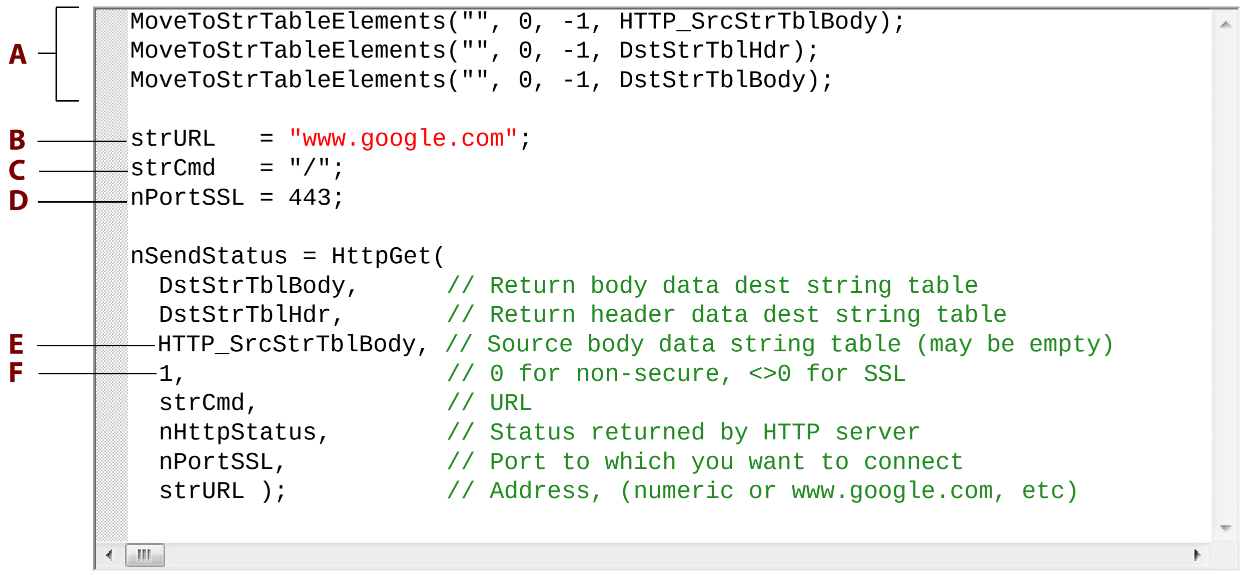
5. Create the following string tables. For more information, see "Adding Tables" on page 260.

Table Name	Type	Length	Width
DstStrTblBody	String	100	1024
DstStrTblHdr	String	100	1024
HTTP_SrcStrTblBody	String	10	256

6. Add an Action Block named Begin, and an OptoScript Block named Get Content to the Powerup chart.
7. Connect the Begin block to the Get Content block.



8. Add the following code to the OptoScript Block.



```
MoveToStrTableElements("", 0, -1, HTTP_SrcStrTblBody);
MoveToStrTableElements("", 0, -1, DstStrTblHdr);
MoveToStrTableElements("", 0, -1, DstStrTblBody);

strURL   = "www.google.com";
strCmd   = "/";
nPortSSL = 443;

nSendStatus = HttpGet(
  DstStrTblBody,      // Return body data dest string table
  DstStrTblHdr,       // Return header data dest string table
  HTTP_SrcStrTblBody, // Source body data string table (may be empty)
  1,                  // 0 for non-secure, <>0 for SSL
  strCmd,             // URL
  nHttpStatus,        // Status returned by HTTP server
  nPortSSL,           // Port to which you want to connect
  strURL );           // Address, (numeric or www.google.com, etc)
```

A—MoveToStrTableElements. These lines ensure that the tables are empty.

B— Web address. The text string used for argument 7. For example "www.google.com".

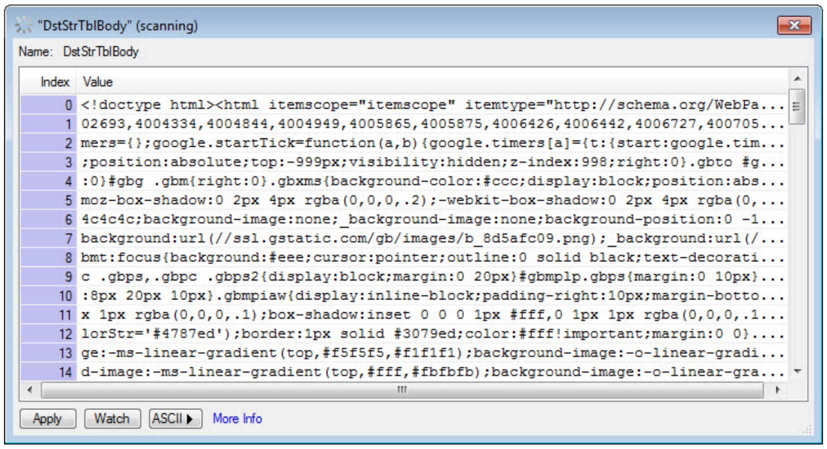
C—Sub-page. A forward slash (/) indicates the root level.

D—Port number. 80 is the standard port if you want to retrieve HTTP content. 443 is the standard port if you want to retrieve HTTPS content.

E—Timeouts. Use element [0] of the table for Argument 2 to change the timeouts for transmitting, receiving, and connecting. For details, see "Setting Timeouts for the HTTP Get and HTTP Post from String Table Commands" on page 326.

F—Security mode. Use 0 to retrieve non-secure connection HTTP content. Use a number other than zero (such as 1) to retrieve secure HTTPS content.

9. Click Debug, and then click Run Strategy.
10. Double-click the DstStrTblBody table to see whether the webpage content was retrieved. If content was retrieved, you will see HTML code.



Index	Value
0	`<!doctype html><html itemscope="itemscope" itemtype="http://schema.org/WebPa...`
1	`02693,4004334,4004844,4004949,4005865,4005875,4006426,4006442,4006727,400705...`
2	`mers={};google.startTick=function(a,b){google.timers[a]={t:{start:google.tim...`
3	`;position:absolute;top:-999px;visibility:hidden;z-index:998;right:0}.gbto #g...`
4	`:0}#gbg .gbm{right:0}.gbxms{background-color:#ccc;display:block;position:abs...`
5	`moz-box-shadow:0 2px 4px rgba(0,0,0,.2);-webkit-box-shadow:0 2px 4px rgba(0,...`
6	`4c4c4c;background-image:none;_background-image:none;background-position:0 -1...`
7	`background:url(//ssl.gstatic.com/gb/images/b_8d5afc09.png);_background:url(/...`
8	`bmt:focus{background:#eee;cursor:pointer;outline:0 solid black;text-decorati...`
9	`c .gbps,.gbpc .gbps2{display:block;margin:0 20px}#gbmplp.gbps{margin:0 10px}...`
10	`:8px 20px 10px}.gbmpiaw{display:inline-block;padding-right:10px;margin-botto...`
11	`x 1px rgba(0,0,0,.1);box-shadow:inset 0 0 0 1px #fff,0 1px 1px rgba(0,0,0,.1...`
12	`lorStr='#4787ed');border:1px solid #3079ed;color:#fff!important;margin:0 0}....`
13	`ge:-ms-linear-gradient(top,#f5f5f5,#f1f1f1);background-image:-o-linear-gradi...`
14	`d-image:-ms-linear-gradient(top,#fff,#fbfbfb);background-image:-o-linear-gra...`

Setting Timeouts for the HTTP Get and HTTP Post from String Table Commands

The default timeout for transmitting, receiving, and connecting is 10,000 milliseconds (10 seconds). If you want to change one or more timeouts from the default, specify the value in element [0] of the table for Argument 2, as follows:

To do this:	Use this command where "nnnn" is the timeout in milliseconds
Set transmit timeout	"to.tx:nnnn"
Set receive timeout	"to.rx:nnnn"
Set connect timeout	"to.cx:nnnn"

Each timeout command is independent of the others so you can use one, two, or all three of the commands in any order. If you use more than one command, put a space between each one.

The following example sets the transmit timeout to 20 seconds and the receive timeout to 15 seconds. The connect timeout is not changed.

```
HTTP_SrcStrTblBody[0]="to.tx:20000 to.rx:15000"
```

Table name   Element 0   Transmit timeout   Receive timeout

Control Engine Commands

The following commands refer to the control engine:

Calculate Strategy CRC	Get Firmware Version
Erase Files In Permanent Storage	Load Files From Permanent Storage
Get Available File Space	Retrieve Strategy CRC
Get Control Engine Address	Save Files To Permanent Storage
Get Control Engine Type	Start Alternate Host Task

Commands Relating to Permanent Storage

The term "Permanent Storage" in three of the commands listed above refers to the control engine's flash memory. Files that are saved to flash memory remain in the control engine even when power to it is turned off. On a SoftPAC controller, files are saved to the PC's hard drive.

These commands do NOT affect firmware files, configuration data, or strategy files saved to flash; they affect only files at the root of the control engine's file system. For more information on the file system, see "Using the Control Engine's File System" on page 301. For the specifics on individual commands, see online help or the *PAC Control Command Reference*.

***CAUTION**: Since these commands write to flash memory, use them sparingly within your strategy and make sure they do not end up in a loop. You can literally wear out flash memory if you save to it or erase it too many times.*

Digital Point Commands

The following commands are used with digital points.

Basic Commands
Off?
On?
Turn Off
Turn On

Totalizers
Get Off-Time Totalizer
Get On-Time Totalizer
Get & Restart Off-Time Totalizer
Get & Restart On-Time Totalizer

Counters[1]
Clear Counter
Get Counter
Get & Clear Counter
Start Counter
Stop Counter

Latches
Clear All Latches
Clear Off-Latch
Clear On-Latch
Get Off-Latch
Get On-Latch
Get & Clear Off-Latch
Get & Clear On-Latch
Off-Latch Set?
On-Latch Set?

Pulses
Generate N Pulses
Get Off-Pulse Measurement
Get Off-Pulse Measurement Complete Status
Get & Restart Off-Pulse Measurement
Get On-Pulse Measurement
Get On-Pulse Measurement Complete Status

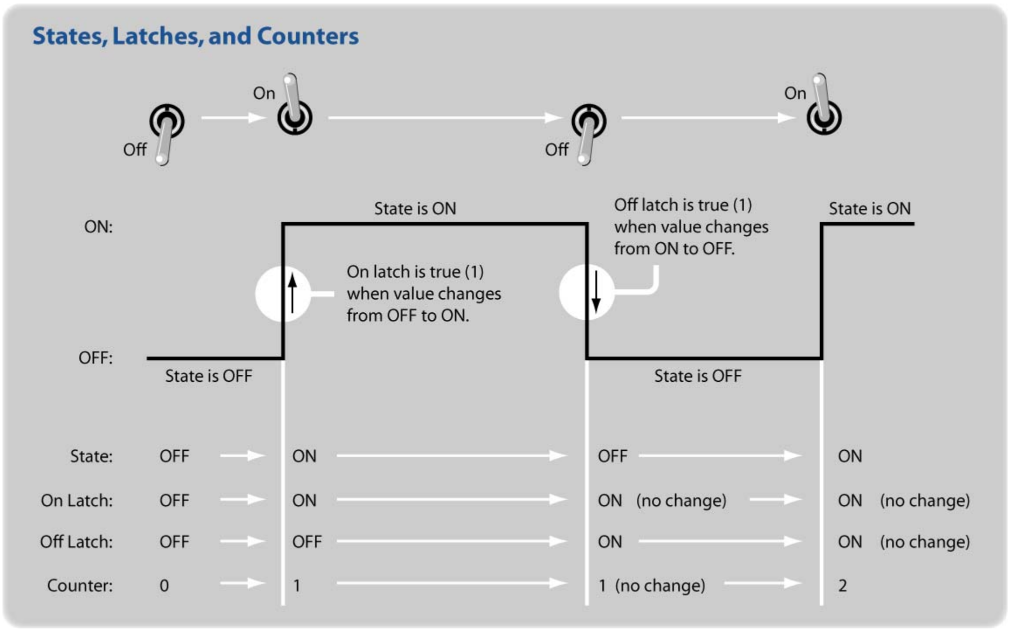
Period and Frequency
Get Frequency
Get Period
Get Period Measurement Complete Status
Get & Restart Period
Set TPO Percent
Set TPO Period

Get & Restart On-Pulse Measurement
Start Continuous Square Wave
Start Off-Pulse
Start On-Pulse

[1] Some digital point commands are available only in PAC Control Professional. Some commands are available only on some I/O units. For details, see specific information for each command in Opto 22 form 1700, the PAC Control Command Reference, or in Online Help.

States, Latches, and Counters

The following diagram illustrates states, latches, and counters. While states and latches apply to digital points on all I/O units, counters depend on the capability of the brain. See the brain's data sheet for specifications.


States, Latches, and Counters
Off
On
Off
On
ON:
OFF:
State is OFF
State is ON
On latch is true (1) when value changes from OFF to ON.
Off latch is true (1) when value changes from ON to OFF.
State is OFF
State is ON

State:	OFF	ON	OFF	ON
On Latch:	OFF	ON	ON (no change)	ON (no change)
Off Latch:	OFF	OFF	ON	ON (no change)
Counter:	0	1	1 (no change)	2

Latches

Latches are an extremely high-speed digital function. Both on-latches and off-latches are available. Latches are automatic and do not have to be configured.

When the value of a digital input point changes from off to on, an on-latch is automatically set. While the value of the point may return to off, the on-latch remains set until cleared, as a record of the change. Similarly, an off-latch is set when the value of a digital point changes from on to off, and it remains set until cleared.

To read a latch and clear it at the same time, use the command Get & Clear On-Latch or Get & Clear Off-Latch.

Counters

Most standard digital inputs can be used as counters to count the number of times the input changes from off to on. The availability of counters depends on the brain's capabilities, and the speed of counters depends on the module.

For information on brain capabilities, see form 1689, the SNAP PAC Brains Data Sheet.

For more information on module counting speed, see the module's specifications on the Opto 22 website, www.Opto22.com. On the website, select Products > SNAP PAC System > Brains and I/O > Digital I/O.

NOTE: You can use counters on high-density modules with SNAP-PAC-R1 and SNAP-PAC-R1-B controllers, and SNAP-PAC-EB1 and SNAP-PAC-SB1 brains with firmware 8.1 or newer.

Before using a counter, you must configure the point as a counter. (See "Adding a Digital I/O Point" on page 136 or use PAC Manager.) Counters on Ethernet and SNAP PAC I/O units do not need to be started, and they cannot be stopped. Therefore, do not use the Start Counter and Stop Counter commands with SNAP PAC I/O units. However, you can use Get Counter and Get & Clear Counter.

Counters on *mistic* I/O units must first be started using the Start Counter command. In addition to Start Counter, you can use the following commands on *mistic* I/O units: Stop Counter, Get Counter, Get & Clear Counter, and Clear Counter.

Quadrature Counters

Quadrature counters require a special module and configuration, but once they are configured, you use the same commands (such as Start Counter and Clear Counter) for them as for regular counters.

NOTE: You can use counters on high-density modules with SNAP-PAC-R1 and SNAP-PAC-R1-B controllers, and SNAP-PAC-EB1 and SNAP-PAC-SB1 brains with firmware 8.1 or newer.

Before using a counter, you must configure the point as a counter. (See "Adding a Digital I/O Point" on page 136 or use PAC Manager.) Counters on Ethernet and SNAP PAC I/O units do not need to be started, and they cannot be stopped. Therefore, do not use the Start Counter and Stop Counter commands with SNAP PAC I/O units. However, you can use Get Counter and Get & Clear Counter. A positive value means that phase A leads phase B. See additional details in the *PAC Control Command Reference* or online Help.

Counters on *mistic* I/O units must first be started using the Start Counter command. In addition to Start Counter, you can use the following commands on *mistic* I/O units: Stop Counter, Get Counter, Get & Clear Counter, and Clear Counter. A positive value means that phase B leads phase A.

Totalizers

Digital totalizers track the total time a specific input point has been on or off. For example, you could track how long a pump, fan, or motor has been on. Digital totalizers are useful for periodic maintenance. Before using a totalizer, you must configure the point with this feature. (See "Adding I/O Points" on page 136 for help.) The availability of totalizers depends on the brain; see the brain's data sheet for more information.

To check total time and leave the totalizer running, use Get Off-Time Totalizer or Get On-Time Totalizer. To check total time and reset the totalizer to zero, use Get & Restart Off-Time Totalizer or Get & Restart On-Time Totalizer.

Pulses

Pulsing commands send on- and off-pulses to an output point. The availability of pulsing depends on the brain; see the brain's data sheet for specifications.

Generate N Pulses. The command Generate N Pulses is frequently used to flash a light or sound an alarm. For example, you could sound a two-second alarm four times. In the arguments, you set the number of times the on-pulse is sent, the length of the on-pulse, and the length of the off-pulse. Generate N Pulses always starts with an off-pulse. If you resend this command, make sure to leave sufficient time in between so it does not interfere with itself.

Start On Pulse and Start Off Pulse. The commands Start On Pulse and Start Off Pulse send a single pulse cycle:

- Start On Pulse starts with an on-pulse of a length you determine, and ends with an off-pulse.
- Start Off Pulse starts with an off-pulse of a length you determine, and ends with an on-pulse.

Both of these commands can be used as time delays. For example, if a light is on and you want to turn it off after 30 seconds, you can send a Start On Pulse command, setting the on-pulse to be 30 seconds long. At the end of that time, the off-pulse is sent to turn off the light.

You can also use this type of command in a loop to turn a digital point on or off for short intervals. For example, you could create a loop that checks the level of liquid in a tank and pulses on a drain if the level is too high. The advantages of using a pulse command are that the point does not have to be turned off, and if communication is lost to the point, the point does not remain on.

Pulse Measurement commands measure pulses on digital input points. For details, see the specific command in the *PAC Control Command Reference* or online Help.

IVAL and XVAL

All I/O points have two associated values: XVAL and IVAL. If you are using PAC Control in Debug mode to manipulate I/O values or to disable an I/O point or I/O unit, you need to understand these values.

XVAL. The external value, or XVAL, is the "real" or hardware value as seen by the I/O unit. This value is external to the control engine and strategy logic.

IVAL. The internal value, or IVAL, is a logical or software copy of the XVAL that is in the control engine. The IVAL may or may not be current, since it is updated to match the XVAL when strategy logic accesses the I/O point.

If the IVAL does not match the XVAL, a mismatch just means that your strategy logic is not reading from or writing to the I/O point in question at the moment.

Simulation and Test: The "Real" Use for XVAL and IVAL

To test output performance, you may want to force an XVAL for a specific output to a particular value. If the program is actively writing to the output, you need to disable the output to do so. If the program is stopped, there is no need to disable it.

To test program logic, you may want to force an IVAL for a specific input to a particular value. To do so, you must disable the input first.

You can disable an I/O point or unit in two ways. The more common way is from within Debug mode, by double-clicking a point on the Strategy Tree and modifying the point's settings and values through the Inspection dialog box. The second way is from within the strategy, using commands such as Disable Communication to Digital Point, Disable Communication to Analog Point, or Disable Communication to I/O Unit. (See "Simulation Commands" on page 357.)

Additional Commands to Use with Standard Digital Points

Although not listed under Digital Point commands, several other commands can be used for digital operations:

- Use **Move** to cause an output to assume the state of another input or output. A digital input or output that is on returns a True (non-zero). A True (non-zero) sent to a digital output turns it on.
- Use **NOT** to cause an output on one I/O unit to assume the opposite state of an input on another I/O unit.
- Use **Get I/O Unit as Binary Value** to get the state of all points at once. Then use **Bit Test** to determine the state of individual points. This method is much faster than reading each point individually.
- Use **Set I/O Unit From MOMO Masks** to control all outputs at once.

Standard Digital Points and OptoScript Code

In OptoScript code, a standard digital I/O point can be used directly, wherever a numeric variable can be used. For example, you can turn a point off by assigning it a value of zero, or turn it on by assigning it a non-zero value. You can also use standard digital points directly in mathematical expressions and control structures. For more information, see "Using I/O in OptoScript" on page 378.

Reading or Writing Digital Points

The Move command is the main command used to read or write digital points.

Error Handling Commands

The following commands refer to handling errors:

Add Message to Queue	Get Error Code of Current Error
Add User Error to Queue	Get Error Count
Add User I/O Unit Error to Queue	Get ID of Block Causing Current Error
Caused a Chart Error?	Get Line Causing Current Error
Caused an I/O Unit Error?	Get Name of Chart Causing Current Error
Clear All Errors	Get Name of I/O Unit Causing Current Error
Copy Current Error to String	Get Severity of Current Error
Disable I/O Unit Causing Current Error	Remove Current Error and Point to Next Error
Enable I/O Unit Causing Current Error	Stop Chart on Error
Error on I/O Unit?	Suspend Chart on Error
Error?	

All good programmers must deal with errors. These error handling commands are used to monitor errors, figure out which I/O unit caused an error, disable or re-enable communication to the I/O unit, and clear errors and other messages from the message queue. For a simple example of an error handler chart, see page 90. For more on the message queue, see "Queue Messages" on page 424.

You can use the command Add User Error to Queue to add your own information, warning, or error message to the queue. This command can be helpful in troubleshooting.

IO Enabler Sample Strategies

Sample "IO Enabler" strategies are available on the Opto 22 website. The logic in these sample strategies is designed to automatically recover communications to any I/O unit that temporarily goes offline (that is, has communications disabled) for any reason. Choose a sample that is compatible with your system configuration. Each sample has the same basic code:

www.opto22.com/site/downloads/dl_drilldown.aspx?aid=3605

www.opto22.com/site/downloads/dl_drilldown.aspx?aid=3603

www.opto22.com/site/downloads/dl_drilldown.aspx?aid=3604

Event/Reaction Commands

Pro

The following commands cannot be used with Ethernet-based I/O units. They are used with event/reactions on *mistic* I/O units, and are available only in PAC Control Professional when legacy I/O units and commands are enabled. To enable legacy I/O units and commands, see "Legacy Options" on page 232.

NOTE: Similar events and reactions can be configured on a SNAP Ethernet-based I/O unit using PAC Manager, but they can interfere with PAC Control strategy logic unless you are very careful. For more information, see Opto 22 form 1704, the PAC Manager User's Guide.

Clear Event Latch
Clear All Event Latches
Clear I/O Unit Interrupt
Disable Interrupt on Event
Disable Scanning for All Events
Disable Scanning for Event
Disable Scanning of Event/Reaction Group
Enable Interrupt on Event
Event Occurred?
Event Occurring?
Enable Scanning for All Events
Enable Scanning for Event
Enable Scanning of Event/Reaction Group
Event Scanning Disabled?
Event Scanning Enabled?
Generating Interrupt?
Get Event Latches
Get & Clear Event Latches
Interrupt Disabled on Event?
Interrupt Enabled on Event?
Read Event/Reaction Hold Buffer

Pro

Understanding Event/Reactions (*mistic* Only)

An event/reaction lets you distribute control logic to an I/O unit, so that some of the logic in a strategy can be run on the I/O unit independently of the controller. Event/reactions are supported by most *mistic* protocol brains. To verify, check the data sheet for the brain you are using.

As the name suggests, an event/reaction consists of an event and a corresponding reaction. The event is a state you define that the I/O unit can recognize. The defined state can be a combination of values, inputs, and outputs. Each time the event becomes true, its corresponding reaction is executed once.

On a digital multifunction I/O unit, for example, any pattern of input and output states (on and off) can constitute an event. On an analog I/O unit, an event could occur when an input channel attains a reading greater than a selected value. Examples of reactions include turning on or off a set of outputs, ramping an analog output, or enabling or disabling other event/reactions.

Event/reactions are stored in each I/O unit. As soon as power is applied to the I/O unit, all event/reactions for which scanning is enabled are scanned continuously in the same order in which they were configured in PAC Control. Since each *mistic* I/O unit can be configured with up to 256 event/reactions, complex tasks and sequences can be performed on an I/O unit.

For step-by-step instructions on configuring event/reactions, see "Configuring Event/Reactions" on page 162.

Why Use Event/Reactions?

- To reduce communication overhead between the I/O unit and the controller.
- To distribute control logic sequences to the I/O unit rather than concentrating them in the controller.
- To handle high-speed logic functions more efficiently by distributing them to an I/O unit.
- To increase the execution speed of a strategy in the controller.

Typical Applications for Event/Reactions

- Motor-starting logic
- Drum sequencers
- Alarms
- Analog biasing
- Power-up sequencing

Using an Interrupt Chart with the Generating Interrupt? Command (*mistic* Only)

There are two reasons to use an Interrupt chart:

- To be promptly notified of critical events that occur on the I/O unit.
- To allow an event on one I/O unit to quickly cause a reaction on another I/O unit using logic in the Interrupt chart as the gateway. (For maximum speed and efficiency, configure reactions to occur at the I/O unit whenever possible.)

To use an Interrupt chart, follow these steps:

1. Wire the interrupt lines from *mistic* I/O units to the controller.
2. Use the Generating Interrupt? command to determine which I/O units are generating an interrupt.

3. For each I/O unit that is generating an interrupt, do the following steps:
   a. Use the command Clear I/O Unit Interrupt.
   b. Use the command Event Occurred? to determine which event/reaction(s) caused the interrupt.
   c. For each event that occurred, use the command Clear Event Latch.
   d. React to each event as desired.

   IMPORTANT: *Check every event that may have caused the interrupt. There may have been multiple events.*

See "Using an Interrupt Chart for Event/Reactions (mistic only)" on page 88 for a simple example.

I/O Unit Commands

The following commands are used to communicate with an I/O unit, which controls a group of I/O points:

Clear I/O Unit Configured Flag	Move I/O Unit to Numeric Table Ex
Get I/O Unit as Binary Value	Move Numeric Table to I/O Unit
Get I/O Unit as Binary Value 64	Move Numeric Table to I/O Unit Ex
Get Target Address State[1]	Set All Target Address States[1]
I/O Unit Ready?	Set I/O Unit Configured Flag
IVAL Move Numeric Table to I/O Unit	Set I/O Unit from MOMO Masks
IVAL Move Numeric Table to I/O Unit Ex	Set Target Address State[1]
Move I/O Unit to Numeric Table	Write I/O Unit Configuration to EEPROM

[1] PAC Control Professional only

Clear I/O Unit Configured Flag	Move I/O Unit to Numeric Table Ex
Get I/O Unit as Binary Value	Move Numeric Table to I/O Unit
Get I/O Unit as Binary Value 64	Move Numeric Table to I/O Unit Ex
Get Target Address State[1]	Set All Target Address States[1]
I/O Unit Ready?	Set I/O Unit Configured Flag
IVAL Move Numeric Table to I/O Unit	Set I/O Unit from MOMO Masks
IVAL Move Numeric Table to I/O Unit Ex	Set Target Address State[1]
Move I/O Unit to Numeric Table	Write I/O Unit Configuration to EEPROM

[1] PAC Control Professional only

CAUTION*: Write I/O Unit Configuration to EEPROM is not the recommended method for saving configuration to flash memory. If it is used too often or is in a loop within a strategy, flash memory can literally wear out. Instead of using this command in the strategy, it is better to store configurations to flash manually using either PAC Manager (see Opto 22 form 1704, the* PAC Manager User's Guide, *for instructions) or PAC Control in Debug mode (see* "Inspecting I/O Units and Saving Settings to Flash" on page 172*).*

Commands for Ethernet Link Redundancy

The three target address commands (Get Target Address State, Set Target Address State, and Set All Target Address States) are used to manually change the path of communication between the controller and the I/O unit(s), based on the IP address used for the I/O unit. These commands let you switch communication from a primary to a secondary IP address (or vice versa) or enable or disable communication to the primary or secondary address.

Ethernet link redundancy to I/O units is available only in PAC Control Professional, only from a SNAP PAC controller, and only to Ethernet-based I/O units. The secondary IP address for an I/O unit may be for the second Ethernet network interface on a SNAP PAC R-series controller, or it may be for a separate I/O unit. If it is a separate unit, the primary and secondary I/O units must be the same type (for example, SNAP-PAC-EB1) and have exactly the same points, because they are configured together under one name.

One or both target addresses (primary and secondary) can be enabled, but only one address (the *active* address) will be used by the controller at one time. For link redundancy, both the primary and secondary addresses must be enabled. When an I/O unit is configured with two IP addresses, the default is for both to be enabled and the primary address to be active.

One address is always active. If communication fails through the primary address, the control engine automatically switches to the secondary address. It continues to use the secondary address until communication fails through the secondary address or until you change the active address using Set Target Address State (for one I/O unit) or Set All Target Address States (for all I/O units on the control engine).

You may also want to use these commands to disable one address, for example if you are doing maintenance or repair on a network segment and need to switch communication to another segment temporarily. Disabling one address, of course, means that you no longer have link redundancy.

If both addresses are disabled or unavailable, then communication is not possible and communication to the I/O unit becomes disabled. If both addresses are disabled, and you want to bring the I/O unit back online, you must first enable one or both of the addresses and then enable communication to the I/O unit.

You can find out which addresses are enabled for an I/O unit and which address is currently active by using Get Target Address State. If an address is enabled but is not functional when a strategy is started or when communication for an I/O unit is changed from Disabled to Enabled, an error is posted in the controller's error queue.

To use these commands, you must have already designated primary and secondary IP addresses when configuring I/O units. For steps, see "Adding an I/O Unit" on page 130. For additional information about link redundancy, see "Using Ethernet Link Redundancy in PAC Control" on page 104.

Table Commands

The table commands for I/O units affect the states or values of all points on the I/O unit at once. For example, you can use the command Move I/O Unit to Numeric Table to read the states of all digital points and the values of all analog points on one I/O unit and place them into a table for easy retrieval. Table commands move data very quickly for faster throughput.

Three of the table commands—IVAL Move Numeric Table to I/O Unit Ex, Move I/O Unit to Numeric Table Ex, and Move Numeric Table to I/O Unit Ex—provide support for high-density points. A Points per Module parameter lets you specify the number of points required. For example, if you are using 32-channel modules, you would pass a 32. But if you are only using 8-channel modules, you could pass an 8 instead. This allows table sizes to be as small as possible.

Other commands relating to tables can be found in the following topics:

"Miscellaneous Commands" on page 345
"Logical Commands" on page 340
"Mathematical Commands" on page 342
"Pointer Commands" on page 355
"String Commands" on page 358.

I/O Unit—Event Message Commands

The following commands refer to event messages sent from SNAP PAC, SNAP Ultimate or SNAP Ethernet I/O units. They do not apply to *mistic* event/reactions.

Get I/O Unit Event Message State
Get I/O Unit Event Message Text
Set I/O Unit Event Message State
Set I/O Unit Event Message Text

A SNAP PAC, SNAP Ultimate, or Ethernet I/O system can send a message as a response to an event that occurs within strategy logic. For example, if pressure in a pipe reaches a certain level, the system can send a warning email message to a technician. Or data about a process can be streamed to a computer every 30 seconds for monitoring.

CAUTION*: Events and reactions (including event messages that are processed separately from your strategy) can conflict with strategy logic. If you are using PAC Control, use strategy logic instead of local events and reactions on the I/O unit, or be very careful that local events and reactions do not conflict.*

Use PAC Manager to configure event messages, following the steps in Opto 22 form 1704, the PAC Manager User's Guide. You can configure the following types of event messages:

- Email or paging messages sent to a person
- Simple Network Management Protocol (SNMP) traps sent to an enterprise management system
- Serial messages sent through a serial communication module to a serial device
- Streamed data sent to a device for processing by a software application.

I/O Unit—Event Message commands are commonly used to find out the state or text of an event message, or to set its state. Event messages can be in the following states:

- **Active**—The message has been triggered by the event. If it was configured to be sent just once, it has been sent. If it was configured to be sent at intervals, it is continuing to be sent.
- **Inactive**—The message is not currently triggered by the event.
- **Acknowledged**—The message has been triggered by the event but has been acknowledged by the receiver so it will not be sent again. (Acknowledgments occur only if the receiving

application writes them to the brain's memory map.) Acknowledged is functionally equivalent to Inactive, but can be useful in some cases to determine whether the receiver has received the message.

The command Set I/O Unit Event Message Text is used to dynamically change a message or to "recycle" a message if you run out of event messages on an I/O unit (128 event messages are available for each I/O unit). For more information, see individual commands in the *PAC Control Command Reference* or online help.

I/O Unit—Memory Map Commands

The following commands refer to the memory map in an Opto 22 memory-mapped device, either a controller or an I/O unit. In the case of a controller/brain, they can refer to the device's own memory map or a memory map on another device.

Read Number from I/O Unit Memory Map
Read Numeric Table from I/O Unit Memory Map
Read String from I/O Unit Memory Map
Read String Table from I/O Unit Memory Map
Write Number to I/O Unit Memory Map
Write Numeric Table to I/O Unit Memory Map
Write String Table to I/O Unit Memory Map
Write String to I/O Unit Memory Map

Memory map commands make it possible for advanced users to read from or write to any Opto 22 memory-mapped device, such as a SNAP PAC controller or brain, or a SNAP Ultimate or Ethernet I/O unit. You can use these commands to read or write to any address within the memory map. The commands are especially useful for reading data from a SNAP device using newer features that may be available in the memory map but are not yet incorporated into PAC Control.

NOTE: If you are reading or writing to the device's Scratch Pad area, use the I/O Unit—Scratch Pad commands instead (see page 338). If you are changing event messages, use the I/O Unit—Event Message commands instead (page 336).

In order to use these commands with a SNAP PAC S-series, or SoftPAC, or SNAP-LCE controller, first you need to create an I/O Unit of the type Generic OptoMMP Device to represent the controller. Use the controller's IP address or the loopback address, 127.0.0.1.

When you use these commands, make sure that you read or write the correct type of data (integer, float, string) to match the specified memory map address. The control engine doesn't know what type of data is in any particular address, so it cannot convert the data type.

Since these are I/O unit commands, remember to check all return values and errors to make sure the command was successful. If a command variable contains a value that is obviously wrong—for example, a memory map address in an incorrect format—communication to the I/O unit will be automatically disabled.

See Opto 22 form 1465, the OptoMMP Protocol Guide, to determine the memory map addresses and data types you need to use.

I/O Unit—Scratch Pad Commands

The following commands are used for peer-to-peer communication for sharing strategy data with other Opto 22 memory-mapped controllers on the network. These commands are used by PAC Control to read or write to the Scratch Pad area in the memory map of a SNAP Ultimate I/O unit or a SNAP PAC or SNAP-LCE controller.

Since each read (get) or write (set) command is completed before another occurs, commands cannot interfere with each other. For example, a get command won't read a partial string while a set command is writing the string.

Get I/O Unit Scratch Pad Bits	Set I/O Unit Scratch Pad Bits from MOMO Mask
Get I/O Unit Scratch Pad Float Element	Set I/O Unit Scratch Pad Float Element
Get I/O Unit Scratch Pad Float Table	Set I/O Unit Scratch Pad Float Table
Get I/O Unit Scratch Pad Integer 32 Element	Set I/O Unit Scratch Pad Integer 32 Element
Get I/O Unit Scratch Pad Integer 32 Table	Set I/O Unit Scratch Pad Integer 32 Table
Get I/O Unit Scratch Pad String Element	Set I/O Unit Scratch Pad String Element
Get I/O Unit Scratch Pad String Table	Set I/O Unit Scratch Pad String Table

Because these are I/O unit commands, in order to use these commands with a SNAP PAC S-series, or SoftPAC, or SNAP-LCE controller, first you need to create an I/O Unit of the type Generic OptoMMP Device to represent the controller. Use the controller's IP address or the loopback address, 127.0.0.1.

Also because these are I/O unit commands, remember to check all return values and errors to make sure the command was successful. If a command variable contains a value that is obviously wrong—for example, a memory map address in an incorrect format—communication to the I/O unit will be automatically disabled.

Each controller or SNAP Ultimate I/O unit running a PAC Control strategy can place data in its own or another's Scratch Pad area, and each can retrieve data that has been placed in the Scratch Pad area by other devices using other applications. Using these commands eliminates the need to open communication handles (see "Communication Commands" on page 280), thus speeding up peer-to-peer communication.

The memory map Scratch Pad area supports four data types: bits, integer 32s, floats, and strings.

- For details on the Scratch Pad area, see Opto 22 form 1704, the PAC Manager User's Guide.
- For the complete memory map, see Opto 22 form 1465, the OptoMMP Protocol Guide.
- For details on using the Scratch Pad for peer-to-peer communication with a controller, see the controller's user's guide.

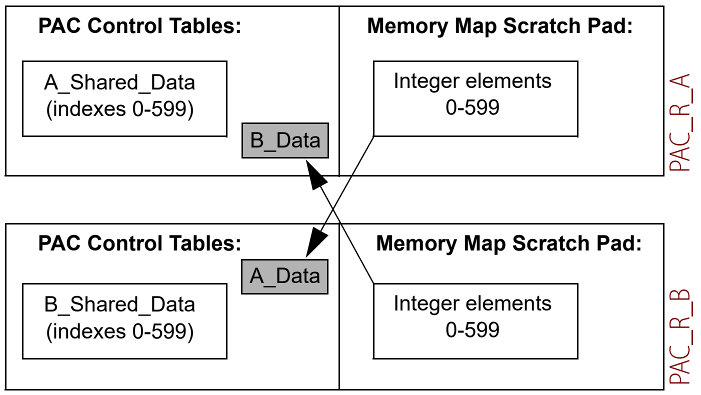
The following page shows a simple example of how Scratch Pad area data exchange would work between two SNAP R-series I/O systems.

Create two tables for PAC_R_A, one for its own data that will be shared (A_Shared_Data) and another for data it will read from PAC_R_B (B_Data). Also create two tables for PAC_R_B, one for its own data (B_Shared_Data) and one for PAC_R_A's data (A_Data).

Suppose PAC_R_A and PAC_R_B are sharing 600 integer elements of the 10,240 integer elements available in the Scratch Pad

PAC_R_A writes data from its own PAC Control strategy table to its own Scratch Pad area, which SPAC_R_B can then read.

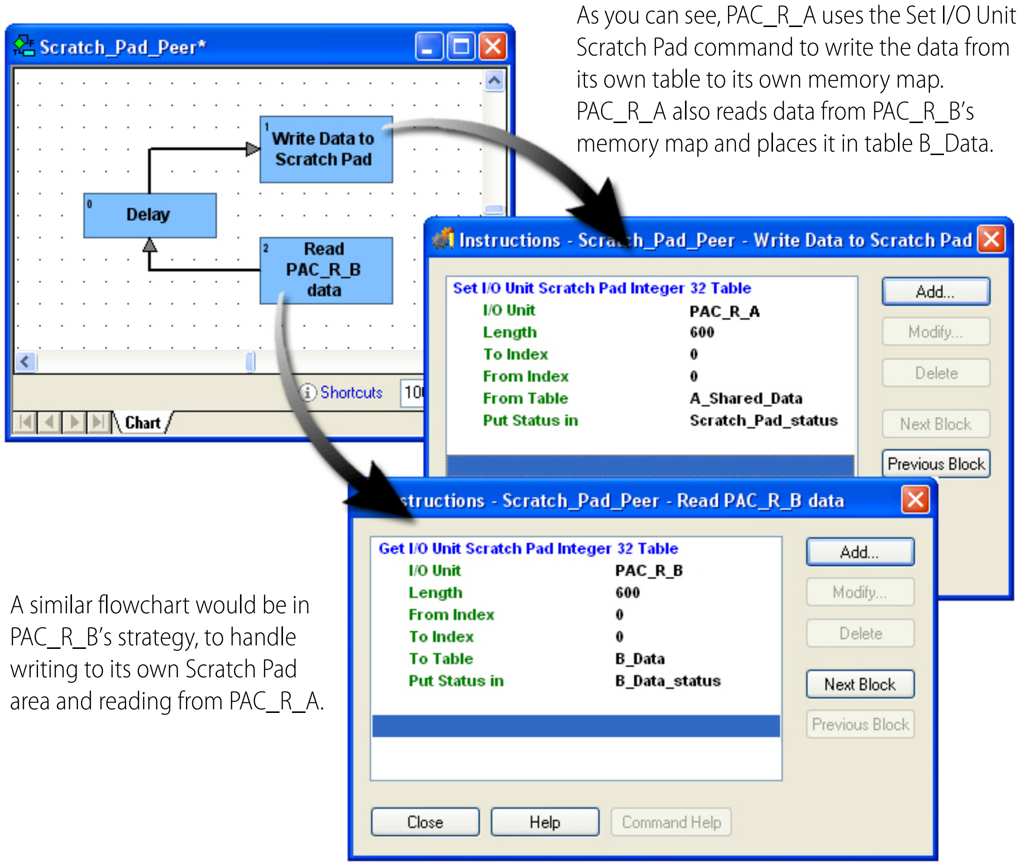
Meanwhile, PAC_R_B writes data from its B_Shared_Data table to its Scratch Pad area, which PAC_R_A reads.



NOTE: When the SNAP R-series I/O unit is writing to its own Scratch Pad, use the loopback IP address. 127.0.0.1

This portion of the flowchart in PAC_R_A (without error checking) might look like this:

As you can see, PAC_R_A uses the Set I/O Unit Scratch Pad command to write the data from its own table to its own memory map. PAC_R_A also reads data from PAC_R_B's memory map and places it in table B_Data.


Scratch_Pad_Peer*
Write Data to Scratch Pad
Delay
Read PAC_R_B data
Shortcuts
Chart
Instructions - Scratch_Pad_Peer - Write Data to Scratch Pad
Add...
Modify...
Delete
Next Block
Previous Block
structions - Scratch_Pad_Peer - Read PAC_R_B data
Close
Help
Command Help

Set I/O Unit Scratch Pad Integer 32 Table

I/O Unit	PAC_R_A
Length	600
To Index	0
From Index	0
From Table	A_Shared_Data
Put Status in	Scratch_Pad_status

Get I/O Unit Scratch Pad Integer 32 Table

I/O Unit	PAC_R_B
Length	600
From Index	0
To Index	0
To Table	B_Data
Put Status in	B_Data_status

A similar flowchart would be in PAC_R_B's strategy, to handle writing to its own Scratch Pad area and reading from PAC_R_A.

Logical Commands

The following commands perform logical functions.

Commands ending a question mark (for example, AND? and OR?) indicate an instruction for a condition block.

AND	Int32 to Float Bits
AND?	Less Than Numeric Table Element?
Bit AND	Less Than or Equal to Numeric Table Element?
Bit AND?	Less Than or Equal?
Bit Change	Less?
Bit Clear	Make Integer 64
Bit Copy	Move 32 bits
Bit NOT	NOT
Bit NOT?	Not Equal to Numeric Table Element?
Bit Off in Numeric Table Element?	Not Equal?
Bit Off?	NOT?
Bit On in Numeric Table Element?	Numeric Table Element Bit Clear
Bit On?	Numeric Table Element Bit Set

Bit OR
Bit OR?
Bit Rotate
Bit Set
Bit Shift
Bit Test
Bit XOR
Bit XOR?
Equal to Numeric Table Element?
Equal?
Flip Flop JK
Float to Int32 Bits
Get High Bits of Integer 64
Get Low Bits of Integer 64
Greater Than Numeric Table Element?
Greater Than or Equal to Numeric Table Element?
Greater Than or Equal?
Greater?
Numeric Table Element Bit Test
OR
OR?
Set Variable False
Set Variable True
Test Equal
Test Greater
Test Greater or Equal
Test Less
Test Less or Equal
Test Not Equal
Test Within Limits
Variable False?
Variable True?
Within Limits?
XOR
XOR?

Understanding Logical Commands

For condition blocks, the Instructions dialog box provides options to designate AND or OR for multiple commands. If you have more than one command in the same condition block and you choose the AND option, all of the commands must evaluate true for the block to exit true. If you have more than one command in a condition block and choose the OR option, the block exits true if any of its commands evaluates true.

Logical actions and conditions work with integers, individual bits within an integer, a single digital I/O point, or a digital I/O unit. These values are treated as Boolean; that is, they are either True or False.

For complex logical operations, you may find OptoScript code easier to use than standard PAC Control commands. See "Using Logical Operators" on page 387 for more information.

Logical True and Logical False

PAC Control always returns a value of +1 to indicate True in an integer variable.

A digital input or output that is on also returns a True (+1). Any non-zero value sent to a digital output turns it on. False is defined as zero (0).

For individual bits within an integer variable, bits that are set (1) indicate on. Bits that are cleared (0) indicate off.

While floats can be used in logic, integers are strongly recommended whenever any bits are referenced. Since PAC Control does not permit bits in a float value to be altered, float values must be converted to integers before bits can be evaluated. See "Mathematical Commands" on page 342 for further information on integers and floats.

Mathematical Commands

The following commands perform mathematical functions:

Absolute Value	Raise to Power
Add	Round
Clamp Float Table Element	Seed Random Number
Clamp Float Variable	Square Root
Clamp Integer 32 Table Element	Subtract
Clamp Integer 32 Variable	Truncate
Complement	
Decrement Variable	**Trigonometry**
Divide	Arcsine
Generate Random Number	Arccosine
Increment Variable	Arctangent
Maximum	Cosine
Minimum	Hyperbolic Cosine
Modulo	Hyperbolic Sine
Multiply	Hyperbolic Tangent
Natural Log	Sine
Raise e to Power	Tangent

Using Integers

In PAC Control, an integer 32 is a 32-bit signed number ranging from -2,147,483,648 to 2,147,483,647 (±2 billion). An integer 64 ranges from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.

An integer can only be a whole number (-1, 0, 1, 2, 3, etc.). In other words, integers do not include a decimal point. The result of an integer operation is always an integer, even if it is placed in a float variable. For example, if 9 is divided by 10, the result is zero (0.9 truncated to an integer). To receive a float result, at least one of the operators would have to be a float. Use the Truncate command to round down to the nearest whole number.

Controlling Rounding

Use the Round command to round up or down to the nearest whole number. Note that 1.50 rounds up to 2.0, and 1.49 rounds down to 10.

Using Floats

While computers, CPUs, and electronic devices (such as Opto 22 controllers) store numbers in binary format, most often they represent real numbers as floating point numbers, or *floats*. For example, in industrial automation applications, all analog values read from an I/O unit are floats. Floats represent real numbers in scientific notation: as a base number and an exponent.

The IEEE Standard for Binary Floating-Point Arithmetic (IEEE 754) is the most widely used standard for floating-point computation. It defines how to store real numbers in binary format and how to convert between binary and float notations.

Opto 22's SNAP PAC System uses IEEE single-precision floats, which have 32 binary digits (bits). The IEEE 754 32-bit float format is as follows:

1 bit	8 bits	23 bits
x	xxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxx
Sign	Exponent	Significand

Float calculation: $(-1)^{\text{Sign}} \times [1 + \text{Significand}/2^{23}] \times 2^{(\text{Exponent}-127)}$

While this is an excellent standard for the purpose, it has limitations that could cause issues if you're not aware of them. Squeezing infinitely many real numbers into a finite number of bits requires an approximate representation. Most floats cannot be exactly represented using this fixed number of bits in a 32-bit IEEE float. Because of this, rounding error is inherent in floating-point computation.

In PAC Control (and in PAC Manager and the OptoMMP protocol), a float is a 32-bit IEEE single-precision number ranging from $\pm 3.402824 \times 10^{-38}$ to $\pm 3.402824 \times 10^{+38}$. These single-precision floats give rounding errors of less than one part per million (1 PPM). You can determine the limit of the rounding error for a particular float value by dividing the value by 1,000,000.

This format guarantees about six and a half significant digits. Therefore, mathematical actions involving floats with seven or more significant digits may incur errors after the sixth significant digit. For example, if the integer 555444333 is converted to a float, the conversion yields $5.554444e^{+8}$ (note the error in the 7th digit). Also, converting $5.554444e^{+8}$ back to an integer yields 555444352 (note the error starting in the 7th digit).

Float Issues and Examples

Accumulation of Relatively Small Floating-point Values. When adding float values, the *relative size* of the two values is important. For example, if you add 1.0 to a float variable repeatedly, the value of the float variable will correctly increase in increments of 1.0 until it reaches $1.677722e^{+7}$ (16,777,220).

Then the value will no longer change, because 1.0 is too small relative to $1.677722e^{+7}$ to make a difference in the significant digits. The same thing will occur if you add 0.0001 to 2,048.0, or add 1,000.0 to $1.717987e^{+10}$. The key is the relative size of the numbers.

Here's another way to think of it. Suppose your bank could keep track only of seven digits. If you were fortunate enough to have one million dollars ($1,000,000) in your account and tried to add 10 cents ($0.10) to it, you would not be able to, because the 10 cents is not big enough relative to the total to be significant. Since the bank has only seven digits to keep track of your money (in this example), one digit has to fall off the end: either the 10 cents falls off the right side or the million digit falls off the left side. Which would you rather see in your bank account?

Seven digits available:	1	2	3	4	5	6	7	
Amount in account:	1	0	0	0	0	0	0	
Add 10 cents (0.10)? (digit falls off on right)	1	0	0	0	0	0	0	
Add 10 cents (0.10)? (digit falls off on left)	0	0	0	0	0	0.	1	Oops!

Note that moving the point indicator doesn't help, because the exponent is separate. If the seven digits for the account represent millions of dollars (1.000000) rather than dollars (1,000,000), the 10 cents would be 0.0000001—still too small to be represented by the seven digits:

Seven digits available:	1	2	3	4	5	6	7	
Amount in account	1.	0	0	0	0	0	0	
Add 10 cents (0.0000001)? (digit falls off on right)	1.	0	0	0	0	0	0	
Add 10 cents (0.0000001)? (digit falls off on left)	.0	0	0	0	0	0	1	Oops again!

The key is that it is not the size of the numbers that matter, but rather their *relative* size.

So if you are accumulating relatively small values in a float variable over a long period of time, at some point, the float value will stop increasing even though you continue to try to add to it.

Comparing Floating-point Values for Equality. Due to rounding errors and the way floating-point calculations are performed, comparing two floats for equality can yield inaccurate results. The precision of comparisons depends on the relative size of the float values as compared to the difference between them.

For example, if 2,097,151.0 is compared for equality with 2,097,152.0, the result will indicate that the two floats are equal, even though it's obvious they are not. The reason is that the difference between the two values is 1.0, and 1.0 compared to one of the compared values (2,097,151.0) is too small; it is less than one part per million.

In this case, 2,097,152.0 divided by 1,000,000 is 2.1. If the difference between the two values is at least 2.1, then the equality comparison is guaranteed to be correct. So if 2,097,152.0 and 2,097,149.0 were compared for equality, the result will indicate they are not equal, because the difference (3.0) is greater than one part per million (2.1). Any time the difference is *at least* one part per million, the result is guaranteed to be accurate. If the difference is less than 1 PPM, it may or may not be accurate.

One method that programmers use to work around this issue is to subtract one float from the other and then compare the absolute value of the result to a limit.

For example:

```
Float_Diff = Float1 - Float2;
If (AbsoluteValue(Float_Diff) < 1.0 ) then
  SetVariableTrue(EqualityFlag);
Else
  SetVariableFalse(EqualityFlag);
Endif
```

Helpful Links for More Information

From Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/IEEE_754

Real numbers: http://en.wikipedia.org/wiki/Real_number

Good example: http://en.wikipedia.org/wiki/Single_Precision

Rounding error: http://docs.sun.com/source/806-3568/ncg_goldberg.html

Comparing floating point numbers:
http://www.cygnus-software.com/papers/comparingfloats/comparingfloats.htm

Mixing and Converting Integers and Floats

An analog value read from an I/O unit and put into an integer is converted from float to integer automatically.

To maintain the integrity and accuracy of a numeric type (float or integer), keep all item types the same. For example, use the Move command to copy an integer value to a variable float when you want float calculations.

Miscellaneous Commands

The following commands are used with tables and for other purposes. Many of them are commonly used.

Comment (Block)	Get Value From Name
Comment (Single Line)	Move
Flag Lock	Move from Numeric Table Element
Flag Unlock	Move Numeric Table Element to Numeric Table
Float Valid?	Move Numeric Table to Numeric Table
Generate Reverse CRC-16 on Table (32 bit)	Move to Numeric Table Element
Get Length of Table	Move to Numeric Table Elements
Get Type From Name	Shift Numeric Table Elements

Comment Commands

The comment commands listed above are used with standard PAC Control commands in action blocks and condition blocks. For information on using comments in OptoScript blocks, see step 10 in "Using the OptoScript Editor" on page 392.

Comment (Single Line) and Comment (Block) commands are used in two entirely different ways:

- **Comment (Single Line)** enters a comment to help explain a block or an instruction within a block. Usually block names and comments within instructions are sufficient, but you can use Comment (Single Line) if you need more room for explanations.
- **Comment (Block)** comments *out* instructions. In other words, it tells the strategy to temporarily ignore certain instructions within a block. It can be useful in debugging or for saving work when a strategy is temporarily changed.

  To use it, place one Comment (Block) command at the beginning of the area you want to ignore, *and place another Comment (Block) command at the end of the area.* If you do not place the second Comment (Block) command, all the remaining instructions in that block are ignored. Areas that are commented out appear in the Instructions dialog box as gray.

PID—Ethernet Commands

The following commands are used with PID loops on SNAP PAC and SNAP Ethernet-based I/O units. For *mistic* I/O units, see "PID—mistic Commands" on page 349. Since PID loops are configured and tuned in Configure Mode while creating your PAC Control strategy, you may not need to use these commands. PID commands are typically used to change input or output location (for example, if the input is on another I/O unit) or to change tuning parameters based on a change that occurs while the strategy is running (such as a recipe change).

For steps to configure and tune PIDs, see "Configuring PID Loops" on page 150. For more information about PID loops on Ethernet-based I/O units, how to tune them, and how to use them in PAC Control, see Opto 22 form 1641, OptoTutorial: SNAP PAC PID.

Get PID Configuration Flags
Get PID Current Input
Get PID Current Setpoint
Get PID Feed Forward
Get PID Feed Forward Gain
Get PID Forced Output When Input Over Range
Get PID Forced Output When Input Under Range
Get PID Gain
Get PID Input
Get PID Input High Range
Get PID Input Low Range
Get PID Max Output Change
Get PID Min Output Change
Get PID Mode
Get PID Output
Get PID Output High Clamp
Get PID Output Low Clamp
Get PID Scan Time
Get PID Setpoint
Get PID Status Flags
Get PID Tune Derivative
Get PID Tune Integral
Set PID Configuration Flags
Set PID Feed Forward
Set PID Feed Forward Gain
Set PID Forced Output When Input Over Range
Set PID Forced Output When Input Under Range
Set PID Gain
Set PID Input
Set PID Input High Range
Set PID Input Low Range
Set PID Max Output Change
Set PID Min Output Change
Set PID Mode
Set PID Output
Set PID Output High Clamp
Set PID Output Low Clamp
Set PID Scan Time
Set PID Setpoint
Set PID Tune Derivative
Set PID Tune Integral

What is a PID?

A proportional integral derivative (PID) control system (often referred to as a PID loop) monitors a process variable, compares the variable's current value to a desired value (a setpoint), and calculates an output to correct error between the setpoint and the variable. Because the calculation is complex, it is done by a mathematical formula that is adjusted (tuned) for each PID loop. The mathematical formulas vary, but all PID systems share these fundamental concepts:

- They evaluate a process variable against its setpoint.
- They control an output to correct the process variable.
- The output comprises proportional, integral, and derivative calculations.
- The effect of proportional, integral, and derivative calculations is modified by user-determined P, I, and D constants.
- The P, I, and D constants need to be tuned for each system.

PID Loops on I/O Units

SNAP PAC controllers and brains provide 96 PID loops per I/O unit. Analog/digital SNAP Ultimate I/O units provide 32 PID loops per I/O unit; SNAP Ethernet I/O units provide 16 PID loops per unit. Because PIDs run on the I/O side of the SNAP PAC R-series controller and Ultimate brain, not on the control side, these PIDs will keep running on the SNAP PAC R-series or Ultimate I/O unit even if the PAC Control strategy stops.

In PAC Control, you can configure each of the PID loops with unique settings for a large number of parameters. For a simple PID loop, you must configure at least the following:

- Input (the process variable being monitored)
- Setpoint (the desired value)
- Output (the I/O point that effects change in the system)
- Scan time (how often the input is sampled)
- PID algorithm used (four algorithms are available; see "Algorithm Choices —Ethernet" on page 347.
- Valid range for input
- Upper and lower clamps for output
- Minimum and maximum change for output

You can also configure the following parameters if necessary:

- Forced output value or use of manual mode if input goes out of range
- Feed forward gain
- Square root of input

In the SNAP PAC, SNAP Ultimate, and SNAP Ethernet PIDs, the derivative is applied only to the process variable (the input) and not to the setpoint. This means you can change the setpoint without causing spikes in the derivative term. These PIDs also prevent integral windup by back calculating the integral without the derivative term. The feed forward term ("bias") is added before output clamping and has a tuning factor.

If desired, you can cascade PIDs by simply using the output point of one PID loop as the input point for another.

Algorithm Choices —Ethernet

When you configure a PID loop, choose one of these algorithms[1]:

- Velocity (Type C)
- ISA

[1]The following obsolete algorithms support PID loops configured before PAC Project R9.5 (for details, see OptoKnowledgeBase article KB82058):
- Velocity (Type B) Obsolete
- ISA (Obsolete)
- Parallel (Obsolete)
- Interacting (Obsolete)

You can continue to use these obsolete algorithms, but Opto 22 recommends you use the new algorithms when configuring new PID loops.

- Parallel
- Interacting

Velocity (Type C) is typically used to perform velocity control. The ISA, Parallel, and Interacting algorithms are derived from the article "A Comparison of PID Control Algorithms" by John P. Gerry in *Control Engineering* (March 1987). These three equations are the same except for the tuning coefficients; converting from one equation to another is merely a matter of converting the tuning coefficients.

Key to Terms Used in Equations

`PV`	Process variable; the input to the PID
`SP`	Setpoint
`InLo, InHi`	Range of the input
`OutLo, OutHi`	Range of the output
`Gain`	Proportional tuning parameter. Unitless. May be negative.
`TuneI`	Integral tuning parameter. In units of seconds. Increasing magnitude increases influence on output.

`TuneD`	Derivative tuning parameter. In units of seconds. Increasing magnitude increases influence on output.
`Output`	Output from the PID
`Err_1`	The Error (PV – SP) from the previous scan
`Integral`	Integrator. Anti-windup is applied after the output is determined to be within bounds.
`PV1, PV2`	PV from the previous scan and the scan before that.
`ScanTime`	Actual scan time (time since previous scan)

Equations Common to All Algorithms

```
Err = PV - SP
Span = (OutHi - OutLo) / (InHi - InLo)
Output = Output + FeedForward * TuneFF
```

Equations Common to ISA, Parallel, and Interacting

```
Integral = Integral + Err
TermP = Err
TermI = TuneI * ScanTime * Integral
TermD = (TuneD / ScanTime) * ( PV - PV1 )
```

Velocity (Type C) Algorithm

```
ΔTermP = ( PV - PV1 )
ΔTermI = TuneI * ScanTime * Err
```

*In this part of the formula, you adjust **TuneI**.*

```
ΔTermD = TuneD / ScanTime * ( PV - 2 * PV1 + PV2 )
```

*In this part of the formula, you adjust **TuneD**.*

```
ΔOutput = Span * Gain * ( ΔTermP + ΔTermI + ΔTermD )
```

*In this part of the formula, you adjust **Gain**.*

ISA (or "Ideal") Algorithm

```
Output = Span * Gain * ( TermP + TermI + TermD )
```

Parallel (or "Independent") Algorithm

```
Output = Span * ( Gain * TermP + TermI + TermD )
```

Interacting (or "Classic") Algorithm

```
Output = Span * Gain * ( TermP + TermI ) * ( 1 + TermD )
```

PID—*mistic* Commands

The following commands are available only in PAC Control Professional, and are used for PID loops running on *mistic* I/O units only. To enable legacy I/O units and commands in PAC Control, see "Legacy Options" on page 232.

For PID loops on Ethernet-based I/O units, see "PID—Ethernet Commands" on page 345.

Clamp Mistic PID Output	Get Mistic PID Output Rate of Change
Clamp Mistic PID Setpoint	Get Mistic PID P Term
Disable Mistic PID Output	Get Mistic PID Scan Rate
Disable Mistic PID Output Tracking in Manual Mode	Get Mistic PID Setpoint
Disable Mistic PID Setpoint Tracking in Manual Mode	Set Mistic PID Control Word
Enable Mistic PID Output	Set Mistic PID D Term
Enable Mistic PID Output Tracking in Manual Mode	Set Mistic PID I Term
Enable Mistic PID Setpoint Tracking in Manual Mode	Set Mistic PID Input
Get Mistic PID Control Word	Set Mistic PID Mode to Auto
Get Mistic PID D Term	Set Mistic PID Mode to Manual
Get Mistic PID I Term	Set Mistic PID Output Rate of Change
Get Mistic PID Input	Set Mistic PID P Term
Get Mistic PID Mode	Set Mistic PID Scan Rate
Get Mistic PID Output	Set Mistic PID Setpoint

What is a PID?

A proportional integral derivative (PID) control system (often referred to as a PID loop) monitors a process variable, compares the variable's current value to a desired value (a setpoint), and calculates an output to correct error between the setpoint and the variable. Because the calculation is complex, it is done by a mathematical formula that is adjusted (tuned) for each PID loop. The mathematical formulas vary, but all PID systems share these fundamental concepts:

- They evaluate a process variable against its setpoint.
- They control an output to correct the process variable.
- The output comprises proportional, integral, and derivative calculations.
- The effect of proportional, integral, and derivative calculations is modified by user-determined P, I, and D constants.
- The P, I, and D constants need to be tuned for each system.

Using PIDs on *mistic* I/O Units

Eight PID loops are available per I/O unit. The PID algorithm used with *mistic* protocol brains, such as the serial B3000 or B3000-B and the G4A8R, is the velocity PID algorithm. It is an interacting type with a reverse output.

- *Interacting* means that the gain is distributed to each term in the equation. Therefore, if you double the gain, you also double the integral and derivative terms.

- *Reverse output* means that the output increases as the input decreases. The reverse output mode is used for *pump-up* control, such as maintaining level, pressure, and flow as well as heating.

For cooling or *pump-down* control, direct output is required. To switch to direct, simply reverse the sign of the gain. For example, a gain of 1.28 would become -1.28. Note that this is not negative gain. The minus sign serves only to change the type of PID output from reverse to direct.

This velocity PID algorithm (also referred to as the incremental PID algorithm) is inherently *anti-windup* since it has no summation in the integral term to saturate. The algorithm is described on pages 160–162 of the book *Microprocessors in Instruments and Control* by Robert J. Bibbero, published by John Wiley and Sons.

Velocity PID Equation (PID—*mistic*)

Change in output = **Gain** *
[(Error – Last Error) +
(**Integral** * Time * Error) +
{(**Derivative**/Time) * (Error – (2 * Last Error) + Oldest Error)}]

where:

- Error is (Setpoint – Input) in Engineering Units
- Time is (Scan Rate/60), which results in time in minutes

All values are in Engineering Units.

The change in output calculated by this formula is added to the existing PID output. If the input span and the output span are different, the change is normalized and then added to the output. This is accomplished by converting the change to a percentage of the input span. The same percentage of output span is then added to the output.

Gain (P)

For those familiar with the term "proportional band," gain is simply the inverse. Gain acts directly on the change in error since the last scan. (Error is the setpoint minus the input value in engineering units.) Therefore, in the case of steady-state error (that is, change in error = 0), gain alone has no effect on the output. For this reason, gain cannot be used alone. Gain is also used as a multiplier on the integral and derivative.

The velocity PID algorithm uses gain much as it is used in the Honeywell "type A" PID and the Bailey "error input" type PID. Higher gain results in increased output change. Too much gain results in output oscillation. Too little gain results in very slow performance.

Integral (I)

This term acts only on the current error. It is used to reduce the current error to zero. Note that during steady-state conditions, integral multiplied by current error multiplied by gain is the only thing affecting the output. The larger the integral value, the larger the change in output.

A positive integral value is required. Integral that is too low will result in undershoot. Integral that is too high will result in overshoot.

Derivative (D)

This term acts only on the change in slope of the input signal. Its purpose is to anticipate where the input will be on the next scan based on a change in the rate of change of the input value. In other words, it changes the output as the input gets near the setpoint to prevent overshooting or undershooting.

Derivative is used in "feed forward" applications and in systems where the loop dead time is long. Its action type is unlimited (that is, it has no filtering). If the input signal is noisy and the derivative value is greater than zero, the input value must be filtered. See "Input Filtering" on page 352 for details. If the slope of the input signal has remained unchanged for the last two scans, the derivative has no effect.

Judging by the change in direction of the input, the derivative contributes an appropriate value to the output that is consistent with where the input will be at the next scan if it continues at its current rate of change.

The derivative is very useful in loops with a long dead time and long time constants. To disable it, set it to zero.

Integral-Derivative Interaction

Integral and derivative can try to move the output in opposite directions. When this is the case, the derivative should be large enough to overcome the integral. Since the derivative is "looking ahead" based on the change in slope, it has a bigger picture than the integral does.

This interaction can be observed when the input is below the setpoint and is rising fast. The integral tries to increase the output (which only makes things worse), while the derivative tries to decrease the output. The derivative does this because at the current rate of change of the input, there will be an input overshoot if the output is increased. Therefore, the derivative needs to be large enough to counteract the integral when necessary.

Configuration Tips (PID—*mistic*)

Gain. The gain value must not be zero. If input engineering units are negative, the output may move in the direction opposite from the one desired. If so, reverse the sign of the gain.

Integral. The integral is required and must be greater than zero.

Input. The input must not be bipolar. An input range of -10 to +10, for example, will not work. Values such as -300 to -100, -100 to 0, and 0 to 100 are acceptable. If an application has a bipolar input range, it will have to be rescaled to a generic range, such as 0 to 100. The setpoint range will then have to match this generic input range.

Setting the output lower and upper clamps. Setting clamps is particularly important if the device controlled by the output signal has "dead areas" at either end. For example, say the output is scaled 0–10. It is connected to a valve that begins to open at 1.25 and is "effectively" fully open at 5.75 (even though it may only be 70% open). Set Lower Clamp to 1.2 (valve closed) and Upper Clamp to 5.75 (valve effectively fully open). This prevents reset windup, potentially resulting in dramatically improved control when the output value has reached either limit and has to suddenly reverse direction.

Setting the maximum change rate of the output. The Max Change Rate can be ignored, since it defaults to 100% per scan. To limit the output rate of change, set Max Change Rate to 10%to start. This setting would limit the output rate of change to 100% in 10 scan-rate periods.

Output. The output can be preset or changed at any time by an operator or by the program. For example, if the output should start at 40% whenever the system is activated, simply set the PID output (or the analog channel output) to this value under program control.

Manual Mode. The factory default causes the setpoint to track the input when the PID is in manual mode, which means that the setpoint will be altered when in manual mode. If you don't want the setpoint to be altered when in manual mode, disable the setpoint track output feature so that when the PID is in manual mode, the setpoint will not be changed.

Input Filtering. If the input signal is noisy, you may want to filter it. To do so, follow these steps:

1. Use the command Set Analog Filter Weight, specifying the appropriate analog input channel. Use a filter weight value of less than 10 times the scan rate. Otherwise, the loop cannot be tuned.
2. Configure the PID loop to use the average/filtered value.
3. You can store the configuration to EEPROM or Flash memory to save the filter weight and the input type (current or average). This can be helpful when re-enabling an I/O unit after a loss of communication.

Tuning Guidelines (PID—*mistic*)

Setting the Scan Rate

The scan rate should be set as fast as possible or as fast as the controlled equipment will allow, unless there are rare and unusual circumstances. Setting the scan rate to be longer than the dead time will result in a PID controller that is so sluggish that it cannot adequately respond to disturbances, and setpoint changes will be extremely slow.

There are, however, exceptions to this rule. If the output of the PID is implemented on equipment that cannot handle fast scan changes, set the PID scan loop to as fast as the equipment can handle. Most control valves, which are very commonly controlled by PID loops, can handle 0.1 second scan rates just fine. However, there are definitely types of equipment that can't handle this. One example would be big gates at big dams. Moving one of these gates is a major event that takes a long time, so one of the control goals is to minimize gate movement. For the flow controllers on these gates, it may be necessary to set the scan time several times longer than the dead time. This allows reaching the setpoint with fewer moves of the gate.

The terms "aggressive" and "conservative" are extremely subjective and depend on the process. "Aggressive" tuning can be thought of as tuning where the speed of reaching the setpoint is the primary concern, and overshoot is tolerated to gain fast response. "Conservative" tuning can be thought of as tuning required in a system where overshoot is not tolerated, and speed of response is sacrificed to prevent overshoot.

The tuning suggestions given here are to achieve the fastest response with no overshoot. This would be on the "aggressive" side of "conservative" tuning. Tuning rules found in control textbooks such as Ziegler-Nichols typically advocate "quarter amplitude decay," which means an overshoot that results in an undershoot that is 1/4 the amplitude of the overshoot and then an overshoot that is 1/4 the amplitude of the undershoot, etc. until it stabilizes on the setpoint. This kind of tuning could be considered "very aggressive."

Determining the Loop Dead Time

To determine the dead time, put the PID output in manual mode, then set the output somewhere around midrange. After the loop has achieved a steady state, change the output by at least 10% of its span. Measure the time (in seconds) that it takes the input to start responding to the change. This is the dead time.

Tuning

The tuning guidelines below are followed by a series of graphs showing the effects of implementing various multiples of the "optimal" gain and integral. The "optimal" gain and integral are multiplied by 2 (too high) and 0.5 (too low), and every combination of these tuning parameters is shown on the graphs. Comparing actual PID loop performance with these graphs can usually identify what adjustments are necessary to improve the tuning of the actual PID loop. It is important to use a graphical tool, like PAC Display, to assist in the tuning process. (Note that the graphical PID tuner in PAC Control cannot be used for tuning *mistic* PID loops.)

These graphs and guidelines are just generalizations. They won't be valid in all possible cases; they are just a guide to help.

These tuning guidelines can be used both to solve tuning problems in an existing loop or as a help to start tuning a new loop.

***IMPORTANT**: Textbook tuning rules, such as Ziegler-Nichols tuning methods, DO NOT work for tuning the velocity PID algorithm.*

Solving Tuning Problems

Oscillations. Oscillations can be caused either by gain that is too high or integral that is too high. If the process variable oscillates below the setpoint, it is probably caused by the gain being too high. If it oscillates at the setpoint, it is not possible to know by looking at the graphs which tuning parameter is causing the problem. Try cutting either the gain or integral, but not both at the same time, to find out which one is causing the problem.

Overshoot. Overshoot is usually caused by the integral being too high. Gain that is too high can also cause overshoot, but that is usually in conjunction with the integral being too high.

Any PID loop can be made to not overshoot and not oscillate if the gain and integral are set low enough, but the response will be slow.

Performance. There is a limit on how fast a good, stable response can be. The middle chart is the best that can be done with no overshoot and no oscillation. The ones with the gain and integral too high move toward the setpoint faster, but they overshoot and oscillate. There will be a point that is the best the PID loop can be tuned, and it will not be possible to get a faster stable response. There are trade-offs between having a fast response and having a stable PID loop that does not overshoot the setpoint.

Starting the Tuning Process for a New PID Loop

A simple and safe method is to start out at a very low gain and integral as shown on page 355 in the lower-left chart. Increase the gain without changing the integral until the fastest response is achieved without oscillation. It still won't reach the setpoint, but the key here is getting a fast response that doesn't oscillate (middle left graph). Now leave the gain alone and increase the integral until it reaches the setpoint without oscillating (middle graph). This completes the tuning.

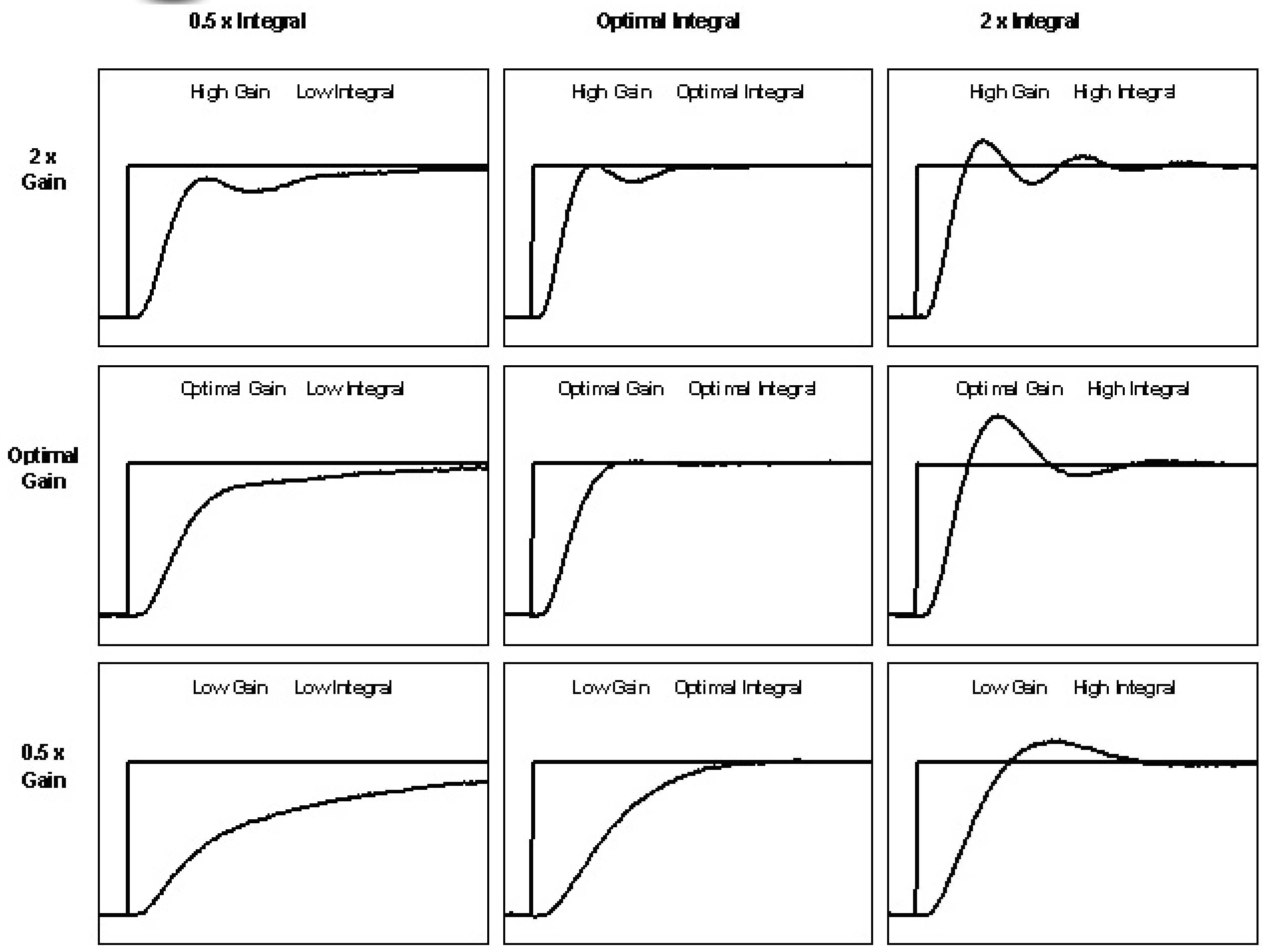
When increasing the gain and integral, it is fastest to just keep doubling them. When the process variable starts oscillating, then make smaller gain and integral changes.

For example, start out with a gain of 0.1 and an integral of 0.1. Next try a gain of 0.2 while keeping the integral at 0.1; then a gain of 0.4, then 0.8, then 1.6, then 3.2. If the PID loop starts oscillating with a gain of 3.2, then try a gain of 2.0 or something in the middle between 1.6 and 3.2. Then make smaller changes until the best gain is found. Suppose the best gain was 2.3; the next step is to keep the gain at 2.3, and then change the integral to 0.2, to 0.4, and then to 0.8, and so on, until the best integral is found.

Derivative

Tuning the derivative term is not addressed in these graphs because most PID loops are fine without it. Derivative in the Opto 22 implementation of the velocity PID algorithm works fine for disturbance rejection; the derivative should be kept very, very low. However, using derivative does not work well for setpoint changes because it will cause spikes in the output. This is because the derivative term of the velocity PID algorithm is calculated based on the error, which is the difference between the setpoint and the process variable. If the setpoint changes, then instantly a jump in error occurs and results in a jump in output. Therefore, it is best to use a derivative term of 0 when cascading PID loops.

Tuning Graphs (PID—*mistic*)



Pointer Commands

The following commands are used with pointers:

Clear Pointer	Move to Pointer
Clear Pointer Table Element	Move to Pointer Table Element
Get Pointer From Name	Pointer Equal to NULL?
Move from Pointer Table Element	Pointer Table Element Equal to NULL?

See also:

"Understanding Pointers" on page 356

"Advantages of Using Pointers" on page 356

"Referencing Objects with Pointers" on page 356

Understanding Pointers

Like integer and float variables, a pointer variable stores a specific number. However, the number is not data—it is the memory location (address) of data. The pointer "points" to data rather than containing the data.

A pointer in PAC Control can point to many different types of objects:

- Another variable
- A digital point or object
- An analog point or object
- An I/O unit
- A chart

Pointers cannot point to other pointers, however. If you try to move a pointer to a pointer, PAC Control just duplicates the existing pointer.

The following table lists the objects that pointers can point to:

Digital Objects	Analog Objects	I/O Units	Variables	Other Objects
Digital Input Digital Output Counter Quadrature Counter	Analog Input Analog Output	SNAP-PAC-R1 SNAP-PAC-R1-B SNAP-PAC-R2 SNAP-PAC-EB1 SNAP-PAC-EB2 SNAP-PAC-SB1 SNAP-PAC-SB2 SNAP-ENET-D64 SNAP-UP1-D64 SNAP-B3000-ENET, SNAP-ENET-RTC SNAP-UP1-ADS SNAP-UP1-M64 SNAP-ENET-S64	Integer Variable Float Variable String Variable Pointer Variable Down Timer Variable Up Timer Variable Integer Table Float Table String Table Communication Handle	Chart

Advantages of Using Pointers

For certain types of operations, pointers can speed up programming and make the strategy more efficient. Pointers are usually recommended only for experienced programmers, however, because their misuse can result in unpredictable behavior. They also complicate strategy debugging. If you use too many pointers, it's easy to lose track of what's pointing to what.

If you choose to use pointers, be sure you use the text tool to document your charts in detail.

Referencing Objects with Pointers

There are two types of pointers—pointer variables and pointer tables.

Pointer Variables. A pointer variable contains a single pointer to a single object. You can set the initial value for a pointer variable when you configure it, or you can set it later by using the command Move to Pointer.

Once the initial value is set, you can reference it using any command you would use for that type of object. For example, if the pointer points to a string variable, you can use any command for the pointer that you would normally use for a string variable, such as Append String to String or Convert String to Float.

Pointer Tables. A pointer table contains a list of objects of different types, each of which can be pointed to. For example, the object at index 0 could be a chart, the object at index 1 a digital point, and the object at index 2 a string variable. An example of using a pointer table for indexing is shown on page 96.

When you create a pointer table, no initial values are set. You can use the Move to Pointer Table command to set individual values in the table.

A pointer table element cannot be directly referenced. It must be copied to a pointer variable first, using the command Move From Pointer Table Element. Once it is in the pointer variable, you can reference it as you would any object of that type. For example, if index 3 in a pointer table points to a counter input, use Move From Pointer Table Element to put the counter input address in a pointer variable. Then you can use any command for the pointer variable that you would normally use with a counter input, such as Start Counter or Clear Counter.

Simulation Commands

The following commands are used for simulation and program testing:

Communication to All I/O Points Enabled?	IVAL Set Analog Point
Communication to All I/O Units Enabled?	IVAL Set Counter
Disable Communication to All I/O Points	IVAL Set Frequency
Disable Communication to All I/O Units	IVAL Set I/O Unit from MOMO Masks
Disable Communication to I/O Unit	IVAL Set Off-Latch
Disable Communication to PID Loop	IVAL Set Off-Pulse
Disable Communication to Point	IVAL Set Off-Totalizer
Enable Communication to All I/O Points	IVAL Set On-Latch
Enable Communication to All I/O Units	IVAL Set On-Pulse
Enable Communication to I/O Unit	IVAL Set On-Totalizer
Enable Communication to PID Loop	IVAL Set Period
Enable Communication to Point	IVAL Set TPO Percent
I/O Point Communication Enabled?	IVAL Set TPO Period
I/O Unit Communication Enabled?	IVAL Turn Off
IVAL Set Analog Maximum Value	IVAL Turn On
IVAL Set Analog Minimum Value	PID Loop Communication Enabled?

Communication to All I/O Points Enabled?	IVAL Set Counter
Communication to All I/O Units Enabled?	IVAL Set Analog Filtered Value
Disable Communication to All I/O Points	IVAL Set Analog Maximum Value
Disable Communication to All I/O Units	IVAL Set Analog Minimum Value
Disable Communication to Event/Reaction[1, 2]	IVAL Set Analog Point
Disable Communication to I/O Unit	IVAL Set Frequency [2]
Disable Communication to Mistic PID Loop[1, 2]	IVAL Set I/O Unit from MOMO Masks
Disable Communication to PID Loop	IVAL Set Mistic PID Control Word[1, 2]
Disable Communication to Point	IVAL Set Mistic PID Process Term[1, 2]
Disable Event/Reaction Group[1, 2]	IVAL Set Off-Latch

- Enable Communication to All I/O Points
- Enable Communication to All I/O Units
- Enable Communication to Event/Reaction[1, 2]
- Enable Communication to I/O Unit
- Enable Communication to Mistic PID Loop[1, 2]
- Enable Communication to PID Loop
- Enable Communication to Point
- Enable Event/Reaction Group[1, 2]
- Event/Reaction Communication Enabled?[1, 2]
- Event/Reaction Group Communication Enabled?[1, 2]
- I/O Point Communication Enabled?
- I/O Unit Communication Enabled?
- IVAL Set Off-Pulse
- IVAL Set Off-Totalizer[2]
- IVAL Set On-Latch
- IVAL Set On-Pulse
- IVAL Set On-Totalizer[2]
- IVAL Set Period[2]
- IVAL Set TPO Percent
- IVAL Set TPO Period
- IVAL Turn Off
- IVAL Turn On
- Mistic PID Loop Communication Enabled?[1, 2]
- PID Loop Communication Enabled?

[1] PAC Control Professional only

[2] *mistic* I/O units only

The Disable commands disconnect the strategy from the real-world device, so that it can be tested without affecting field devices. While the real-world devices are disabled (or if they don't exist) the IVAL commands can be used for testing and simulation. For details on individual commands, see the Opto 22 form 1701,the PAC Control Command Reference, or Online Help.

String Commands

The following commands are used with strings:

- Append Character to String
- Append String to String
- Compare Strings
- Convert Float to String
- Convert Hex String to Number
- Convert IEEE Hex String to Number
- Convert Integer 32 to IP Address String
- Convert IP Address String to Integer 32
- Convert Number to Formatted Hex String
- Convert Number to Hex String
- Convert Number to String
- Convert Number to String Field
- Convert String to Float
- Convert String to Integer 32
- Convert String to Integer 64
- Convert String to Lower Case
- Convert String to Upper Case
- Find Character in String
- Find Substring in String
- Generate Checksum on String
- Generate Forward CCITT on String
- Generate Forward CRC-16 on String
- Generate Reverse CRC-16 on String
- Get Nth Character
- Get String Length
- Get Substring
- Move from String Table Element
- Move String
- Move to String Table Element
- Move to String Table Elements
- Pack Float into String
- Pack Integer 32 into String
- Pack Integer 64 into String
- Pack String into String
- Set Nth Character
- String Equal to String Table Element?
- String Equal?
- Test Equal Strings
- Trim String
- Unpack String
- Verify Checksum on String
- Verify Forward CCITT on String
- Verify Forward CRC-16 on String
- Verify Reverse CCITT on String

Generate Reverse CCITT on String	Verify Reverse CRC-16 on String

Append Character to String	Generate Reverse CCITT on String
Append String to String	Generate Reverse CRC-16 on String
Compare Strings	Get Nth Character
Convert Float to String	Get String Length
Convert Hex String to Number	Get Substring
Convert IEEE Hex String to Number	Move from String Table Element
Convert Integer 32 to IP Address String	Move String
Convert IP Address String to Integer 32	Move to String Table Element
Convert Mistic I/O Hex String to Float[1, 2]	Move to String Table Elements
Convert Number to Formatted Hex String	Pack Float into String
Convert Number to Hex String	Pack Integer 32 into String
Convert Number to Mistic I/O Hex String[1, 2]	Pack Integer 64 into String
Convert Number to String	Pack String into String
Convert Number to String Field	Set Nth Character
Convert String to Float	String Equal to String Table Element?
Convert String to Integer 32	String Equal?
Convert String to Integer 64	Test Equal Strings
Convert String to Lower Case	Trim String
Convert String to Upper Case	Unpack String
Find Character in String	Verify Checksum on String
Find Substring in String	Verify Forward CCITT on String
Generate Checksum on String	Verify Forward CRC-16 on String
Generate Forward CCITT on String	Verify Reverse CCITT on String
Generate Forward CRC-16 on String	Verify Reverse CRC-16 on String

[1] PAC Control Professional only
[2] *mistic* I/O units only

Using Strings

NOTE: All numbers in this discussion of strings are decimal unless otherwise stated.

A PAC Control string is a sequence of characters that can be grouped together. Characteristics of strings include the following:

- Strings are always referred to by name (and, if in a table, by index).
- Each character is represented by one byte.
- Each character is represented by its ASCII code (0 to 255).
- A string containing no characters is referred to as an *empty string*.
- Strings are frequently used in serial communication as a container for moving numeric characters from one device to another.
- Although a string may appear to contain numeric values, it does not. Digits "0" through "9" are characters just as much as "A" through "Z"; they do not represent numeric values.

To illustrate, let's look at the number 22. This is a decimal number representing a quantity of 22. The number 22 can be represented in a string in several ways; here are two of them:

– As "22": two character 50's (The ASCII code for 2 is 50.)

– As "16": a character 49 ("1") and a character 54 ("6") (The hex value of 22 is 16.)

Note that the string representation of the number 22 is no longer a number. It is simply one or two ASCII characters. The string representation of a number must be converted to a numeric value if it is to be used in calculations. Several Convert commands are available for this purpose.

- In standard PAC Control commands, do not use double quotes around string literals. You can use single quotes, but they are not required.
- In OptoScript code, you must use double quotes for string literals. See "11: Using OptoScript" for more information.

String Length and Width

The width of a string is the maximum length a string can be; length is the actual number of characters contained in the string. A string with a width of 100 may currently be empty, which means its length is zero. A string with a width of 10 containing the characters "Hello " has a length of six (five for "Hello" and one for the space after the "o"). Although a string's length may change dynamically as the string is modified by the program, its width remains constant.

When you configure a string variable or string table, you set the width of the string. All the strings in a PAC Control string table must be of the same width.

PAC Control supports a maximum string width of 1024. For applications requiring wider strings, you can use several strings to hold the data, use string tables, or use numeric tables, as described in the next section.

Using Numeric Tables as an Alternative to Strings

Since a string is nothing more than a sequence of characters, you can store a string in a numeric table, with each table element holding a character. The advantage of using numeric tables for strings is that a numeric table can store strings of any size. The disadvantages are:

- Memory usage is much greater.
- No string conversion functions are available for numeric tables. An intermediate temporary string would be required to use string commands for these tables.

Strings in PAC Control can have a width of up to 1024.

Strings and Multitasking

Although string commands are completed before the current task loses its time slice, it is important to note that a string that is constructed in more than one step may require more than one time slice to complete.

For example, if a string is being constructed in two steps (such as Move String "Hello" and Append String to String " World"), after the first step a task switch could occur, and another chart looking at the resulting string might see "Hello" rather than "Hello World."

If another chart is relying on a completed string, you can use a temporary string for building the string, and then move it to the final string. This idea is illustrated in the following example, where a string variable named MSG_String is built in two steps using a temporary string:

1. Move the string literal "The pressure is " to a temporary variable named sTemp.
2. Append a string variable, sPressure, to sTemp.
3. With the complete string now built, move sTemp to MSG_String.

Adding Control Characters to a String

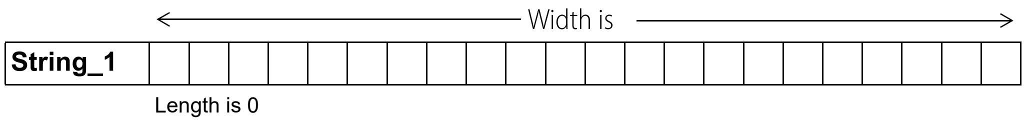
You can input most control characters in a string by typing a backslash (\ ) followed by the two-character hex value of the character. For example, to add an ACK (Ctrl+F) character, enter \06 as part of the string.

This technique works for all control characters except null (\00), carriage return (\0D), line feed (\0A), backspace (\08), and Ctrl+Z (\1A). To add these characters to a string, you must use the Append Character command.

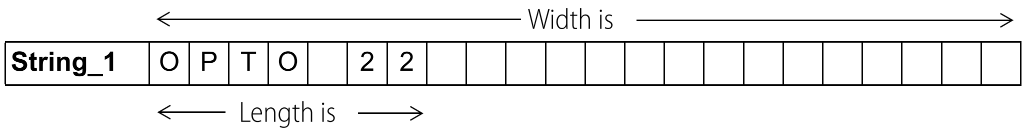
To input a single backslash in a string, type in a double backslash (\\ ).

Sample String Variable

- Declared Name: String_1
- Declared Width: 22
- Maximum Possible Width: 1024
- Bytes of Memory Required: Declared Width + 4 = 22 + 4 = 26



A string is referred to by its name. Initially the previous string is empty, giving it a length of zero. Later, during program execution, seven characters are added to String_1, increasing its length to seven:



Sample String Table

- Declared Name: Promo_Messages
- Declared Width: 26
- Maximum Possible Width: 1024
- Declared Length (Number of indexes, or items, in table): 5
- Maximum Possible Length (Size): 1,000,000

- Bytes of Memory Required: (Declared Width + 4) x Declared Length = (26 + 4) x 5 = 150

← Width is →

Index 0	O	P	T	O		2	2		S	N	A	P		P	A	C		S	Y	S	T	E	M	S		
Index 1	I	n	n	o	v	a	t	i	v	e		I	/	O												
Index 2	D	e	l	i	v	e	r	s		c	o	n	t	r	o	l	,									
Index 3		p	r	o	g	r	a	m	m	a	b	i	l	i	t	y	,		a	n	d					
Index 4	e	n	t	e	r	p	r	i	s	e		c	o	n	n	e	c	t	i	v	i	t	y	!		

A string table is a collection of strings. Each string is referred to by the name of the table it is in and the index where it can be found. The length of the table is the number of strings it can hold. Because string table indexes start with zero, indexes can range from zero to the table length minus one.

The width of each string in the table is the same. The length of each string can vary from zero to the configured width of the table.

String Data Extraction Examples

To extract various pieces of information from a string, use the command Find Substring in String. Consider the following example:

	0	1	2	3	4	5	6															
String_1	O	P	T	O		2	2															

One way to get two separate pieces of information from this string is to get characters 0–3 and then get characters 5 and 6, as shown in the following examples.

Find Substring in String: Example 1

	String_1	*string variable*
Start At	0	*integer literal*
Number Of	4	*integer literal*
Move To	Sub_String1	*string variable (width = 5)*

Results in:

	0	1	2	3	4
Sub_String1	O	P	T	O	

Find Substring in String: Example 2

	String_1	*string variable*
Start At	5	*integer literal*
Number Of	2	*integer literal*
Move To	Sub_String2	*string variable (width = 5)*

Results in:

	0	1	2	3	4
Sub_String2	2	2			

String Building Example

Strings are assembled using commands Move String, Append Character to String, and Append String to String. Consider the following original string and the examples that follow:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
String_1	S	N	A	P		P	A	C														

Move String

From	OPTO	*string literal*
To	String_1	*string variable*

Results in (note that Move String erased the previous contents of the string):

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
String_1	O	P	T	O																		

← Length is →

Append Character to String

From	32	*integer literal (represents a space)*
To	String_1	*string variable*

Results in (note the space character in position 4):

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
String_1	O	P	T	O																		

← Length is →

Append String to String

From	22	*string literal*
To	String_1	*string variable*

Results in:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
String_1	O	P	T	O		2	2															

← Length is →

Append Character to String

From	13	*integer literal (carriage return)*
To	String_1	*string variable*

Results in:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
String_1	O	P	T	O		2	2	¶														

← Length is →

Comparison to Visual Basic and C

The following table lists PAC Control string commands and their equivalents in Microsoft Visual Basic® and C. If you are using OptoScript, see "E: OptoScript Language Reference," for additional comparisons.

PAC Control Command	Visual Basic	C

Append Character to String	S$ = S$ + Chr$(MyChar%)	i = strlen(str);
		str[i] = MyChar;
		str[i + 1] = 0;

Append String to String	S$ = S$ + "Hello"	strcat(str, "Hello");
Convert Hex String Number	1% = "&h" + S$	sscanf(str,"%x",&iNum);
Convert Number to Formatted Hex String	S$ = Hex$(1%)	sprintf(str,"%x",iNum);

Convert Number to String	S$ = CStr(1%)	sprintf(str,"%d",iNum);
		sprintf(str,"%f",fNum);

Convert String to Float	F = CSng(S$)	sscanf(str,"%f",&fNum);
		fNum = atof(str);

Convert String to Integer 32	I% = CInt(S$)	sscanf(str,"%d",&iNum);
		iNum = atoi(str);

Get Nth Character	MyByte% = ASC(MID$(Str$,n%,1))	MyByte = str[n];
Get String Length	MyLENGTH% = LEN(Str$)	iLEN = strlen(str);

Get Substring	SubStr$ = MID$(Str$,i,n)	strncpy(subStr,&str[i],n);
		subStr[n] = '\0';

Move String	STR$ = "Hello"	strcpy(strDest,"Hello");
Test Equal Strings	Equal% = (STR$ = "Hello")	i = strcmp(str1,"Hello");
String Equal?	if STR$ = "Hi" then...	if(!strcmp(str1,"Hi"))
String Equal to String Table Element?	if STR$(n%) = "Hi" then...	if(!strcmp(str1[n],"Hi"))

Convert-to-String Commands

The five convert-to-string commands are typically used when printing a number to a port. The ASCII table on the following page shows how various parameters affect the string as it is converted. Note the following:

- Some commands add leading spaces to achieve the specified length. These spaces are indicated with underscores (_).
- Floats (if used) are automatically rounded to integers before conversion except when using the command Convert Number to Formatted Hex String.

	Command Parameters			**Convert-to-String Commands**				
	Numeric value to be converted	**Number of digits right of decimal point**	**Length**	**Convert Number to Formatted Hex String (Length 8 required for floats)**	**Convert Float to String**	**Convert Number to Hex String**	**Convert Number to String Field**	**Convert Number to String**
Floats	16.0	1	4	41800000	16.0	10	1.6e+01	1.6e+01
	16.0	2	4	41800000	****	10	1.6e+01	1.6e+01
	-16.0	1	4	C1800000	****	FFFFFFF0	-1.6e+01	-1.6e+01
	1.23	1	4	3F9D70A4	_1.2	1	1.23e+00	1.23e+00
	12.3	1	4	4144CCCD	12.3	C	1.23e+01	1.23e+01
	0.0	1	4	00000000	_0.0	0	0.0e+00	0.0e+00
Integers	16	1	4	0010	16.0	10	_ _16	16
	16	2	4	0010	****	10	_ _16	16
	-16	1	4	FFF0	****	FFFFFFF0	_-16	-16
	0	1	4	0000	0.0	0	_ _ _0	0
	1000	1	2	**	**	3E8	1000	1000

**** Indicates an overflow. The whole-number portion of the resulting string is too long for its space.

ASCII Table

The following table shows ASCII characters with their decimal and hex values. For characters 0–31, equivalent control codes are also listed; for example, a carriage return (character 13) is equivalent to CTRL+M (shown in the table as ^M).

Dec	Hex	CC	Char
0	00	^@	NUL
1	01	^A	SOH
2	02	^B	STX
3	03	^C	ETX
4	04	^D	EOT
5	05	^E	ENQ
6	06	^F	ACK
7	07	^G	BEL
8	08	^H	BS
9	09	^I	HT
10	0A	^J	LF
11	0B	^K	VT
12	0C	^L	FF
13	0D	^M	CR
14	0E	^N	SO
15	0F	^O	SI
16	10	^P	DLE
17	11	^Q	DC1
18	12	^R	DC2
19	13	^S	DC3
20	14	^T	DC4
21	15	^U	NAK
22	16	^V	SYN
23	17	^W	ETB
24	18	^X	CAN
25	19	^Y	EM
26	1A	^Z	SUB
27	1B	^[	ESC
28	1C	^\	FS
29	1D	^]	GS
30	1E	^^	RS
31	1F	^_	US

Dec	Hex	Char
32	20	Space
33	21	!
34	22	“
35	23	#
36	24	$
37	25	%
38	26	&
39	27	’
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	‘
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	DEL

Time/Date Commands

The following commands refer to time, dates, and days:

Copy Date to String (DD/MM/YYYY)
Copy Date to String (MM/DD/YYYY)
Get System Time
Get Time Zone Description

Copy Time to String	Get Time Zone Offset
Convert Date & Time to NTP Timestamp	Get Year
Convert NTP Timestamp to Date & Time	Set Date
Get Date & Time	Set Day
Get Day	Set Hours
Get Day of Week	Set Minutes
Get Hours	Set Month
Get Julian Day	Set Seconds
Get Minutes	Set Time
Get Month	Set Time Zone Configuration
Get Seconds	Set Year
Get Seconds Since Midnight	Synchronize Clock SNTP

These commands can be used for timing a process or for making sure things happen according to a set schedule. For example, you could use the command Get Seconds Since Midnight at the beginning of a process and again at the end of the process, and then subtract the two numbers to find out how long the process took.

You can set the time and date on the control engine by synchronizing it with the PC; in PAC Control Debug mode, choose this option while viewing the control engine (Control Engine > Inspect). You can also use these commands to set the time and date on the control engine. Commands with NTP in the name refer to the Network Time Protocol.

Timing Commands

The following commands are used for timers and delays in a strategy.

Timers	Stop Timer
Continue Timer	Timer Expired?
Down Timer Expired?	Up Timer Target Time Reached?
Get & Restart Timer	
Pause Timer	**Delays**
Set Down Timer Preset Value	Delay (mSec)
Set Up Timer Target Value	Delay (Sec)
Start Timer	

Delay Commands

Delay commands are used frequently in strategies to pause the logic. Here are two reasons to use Delay (mSec) or Delay (Sec):

- To allow time for the state of an input to change before it is checked again. For example, a delay could give an operator time to release a button before the state of the button is rechecked, or allow time for an alarm state to change before rechecking.
- To let a chart give up the remainder of its time slice, when its logic does not need to run constantly. For more information on using delays in this way, see form 1776, Optimizing PAC Project System Performance.

Using Timers

Timers are a special type of numeric variable. A PAC Control timer stores elapsed time in units of seconds with resolution of milliseconds. Down timers continuously count down to zero, and up timers continuously count up from zero. Timers can be paused and continued.

To create a timer in PAC Control, configure a numeric variable and select the type Up Timer or Down Timer. You can use any PAC Control command (for example, Move) that references a numeric variable to access a timer. You can view the current value of a timer at any time in PAC Control Debug mode.

Since the timer is independent from the control engine's clock, over thousands of seconds, the timer and the control engine's clock will not match. Timers do not place any additional load on the CPU.

Down Timer Operation

The Set Down Timer Preset Value command sets the time the down timer will start from, but does not start the timer. Use the Start Timer command to start the timer counting down to zero. (Since the default preset value for a down timer is zero, nothing will happen if you use the Start Timer command before setting a value.)

Alternatively, you can use the Move command to set the time the down timer will start from. If you use Move, the down timer begins counting down immediately. If program execution speed is a priority, use the Move command and put an integer value rather than a float into the timer. This action eliminates the float-to-integer conversion time.

Note that if you use the Move command, any value you set using Set Down Timer Preset Value is overwritten, and subsequent Start Timer commands start the timer from the value last sent by the Move command.

To determine if the timer is finished, use the condition Down Timer Expired? This condition is true any time the down timer has a value of zero. Down Timer Expired? is much faster than using the condition Equal? to compare the timer to a value of zero.

The Stop Timer command forces the timer to stop *and puts its value at zero*. If you want to halt the timer and have it maintain its value at the time it was stopped, use the Pause Timer command instead. When you use Pause Timer, you can move the timer's value at the time it was stopped to a variable. You can also use the Continue Timer command to resume the timer where it left off.

Up Timer Operation

The Set Up Timer Target Value command sets the time for the Up Timer Target Time Reached? condition. It does not start the timer, however, and the timer does not stop when it reaches the target value. You must start the up timer from zero by using the Start Timer command.

If you use the Move command to move a value to an up timer, the value you moved becomes the target value and the up timer starts timing immediately. (Note that the timer does not start from the value you moved; it always starts at zero.)

The up timer does not stop when it reaches the target value. To determine if the timer has reached its target value, use the condition Up Timer Target Time Reached? This condition tests the timer to see if it is greater than or equal to the target time.

The Stop Timer command forces the timer to stop *and resets it to zero*. If you want to halt the timer and have it maintain its value at the time it was stopped, use the Pause Timer command instead. After you use Pause Timer, you can then move the timer's value at the time it was stopped to a variable. You can also use the Continue Timer command to resume the timer where it left off.

11: Using OptoScript

Introduction

This chapter shows you how to create and use OptoScript, an optional programming language that can simplify certain types of operations in PAC Control. Modeled after computer languages such as C and Pascal, OptoScript code gives you an alternative to using standard PAC Control commands.

In addition to the examples provided here, see the Control Basic Examples directory on your hard drive for a fully annotated example strategy. OptoScript information is also provided in Opto 22 form 1638, the *SNAP PAC Learning Center User's Guide*.

You will find OptoScript easy to use if you already have computer programming experience. Beginning programmers may also want to try it for control operations involving extensive math calculations, string handling, or complex loops and conditions.

This chapter assumes that you have some programming experience. Experienced programmers may want to see "Notes to Experienced Programmers" on page 444.

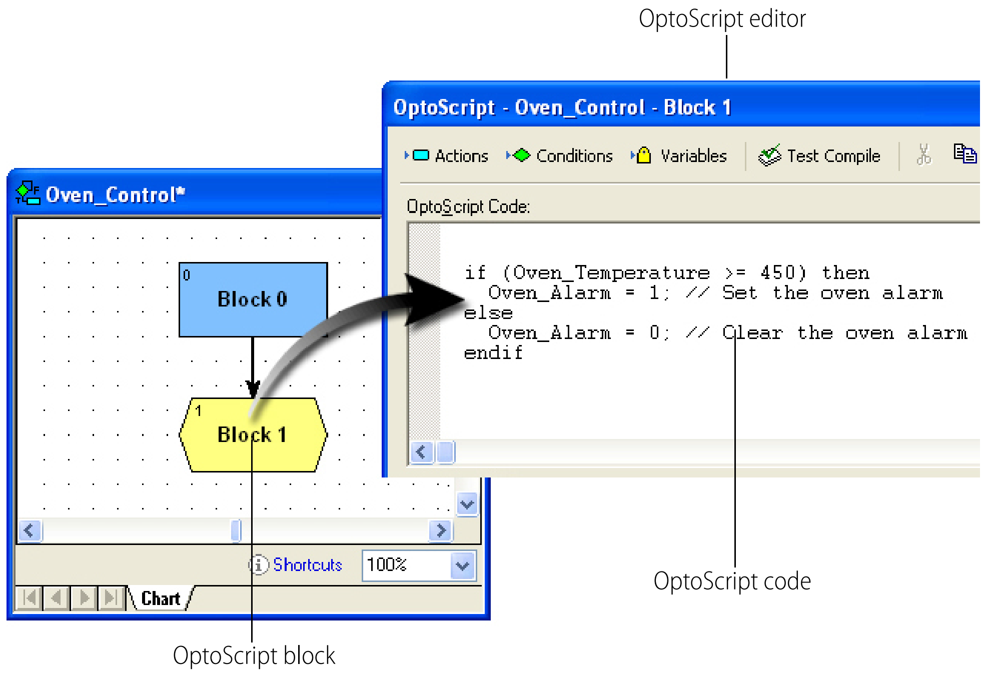
In this Chapter

About OptoScript 369
When to Use OptoScript 370
OptoScript Functions and Commands 378
OptoScript Syntax 379
OptoScript Data Types and Variables 380
OptoScript Expressions and Operators 385
OptoScript Control Structures 388
Using the OptoScript Editor 392
Troubleshooting "Unable To Find" Errors 396
Troubleshooting Syntax Errors 396
Debugging Strategies with OptoScript 398

About OptoScript

OptoScript is a procedural type of computer language similar to Pascal, C, or BASIC. It can be used within any PAC Control strategy or subroutine to replace or supplement standard PAC Control commands. It does not add new functions, but offers an alternative method within PAC Control's flowcharting environment to simplify some common programming tasks.

OptoScript code cannot be mixed with commands in action or condition blocks; it is used in its own hexagonal flowchart block. The following figure shows an example of an OptoScript flowchart block and its contents:



```
if (Oven_Temperature >= 450) then
  Oven_Alarm = 1; // Set the oven alarm
else
  Oven_Alarm = 0; // Clear the oven alarm
endif
```

When to Use OptoScript

You'll want to use OptoScript for some common programming tasks that can be more difficult to do using standard PAC Control commands than using a procedural language. Extensive math calculations or complex loops, for example, can be done with standard commands but take up a lot of space on a flowchart.

When you use OptoScript, however, be aware that it is not self-documenting. Make sure you frequently use comments to explain what the code does, so that when you come back to it a year later—or when someone who is not as familiar with the code or the strategy must change it—it can be easily interpreted.

This section shows examples of using OptoScript for:

- Math expressions
- String handling
- Complex loops
- Case statements
- Conditions
- Combining math expressions, loops, and conditions

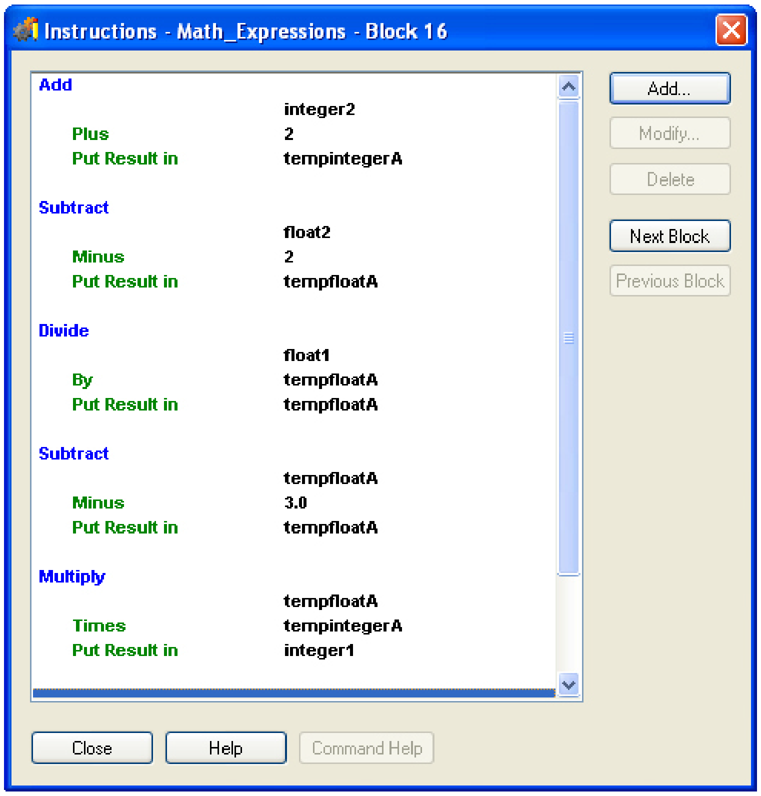
For Math Expressions

OptoScript is especially useful for mathematical computations. Math expressions are simpler and easier, and many of them are built right into the language, instead of requiring commands such as Add or Multiply. OptoScript has no limitations on the number of parentheses you can use in math expressions.

Here's an example of a mathematical expression in OptoScript:

```
integer1 = (integer2 + 2) * (float1 / (float2 - 2) - 3);
```

To accomplish the same computation using standard PAC Control commands, you would need to create at least two intermediate variables and use five instructions, as shown below.



```
Add
                                integer2
    Plus                        2
    Put Result in               tempintegerA

Subtract
                                float2
    Minus                       2
    Put Result in               tempfloatA

Divide
                                float1
    By                          tempfloatA
    Put Result in               tempfloatA

Subtract
                                tempfloatA
    Minus                       3.0
    Put Result in               tempfloatA

Multiply
                                tempfloatA
    Times                       tempintegerA
    Put Result in               integer1
```

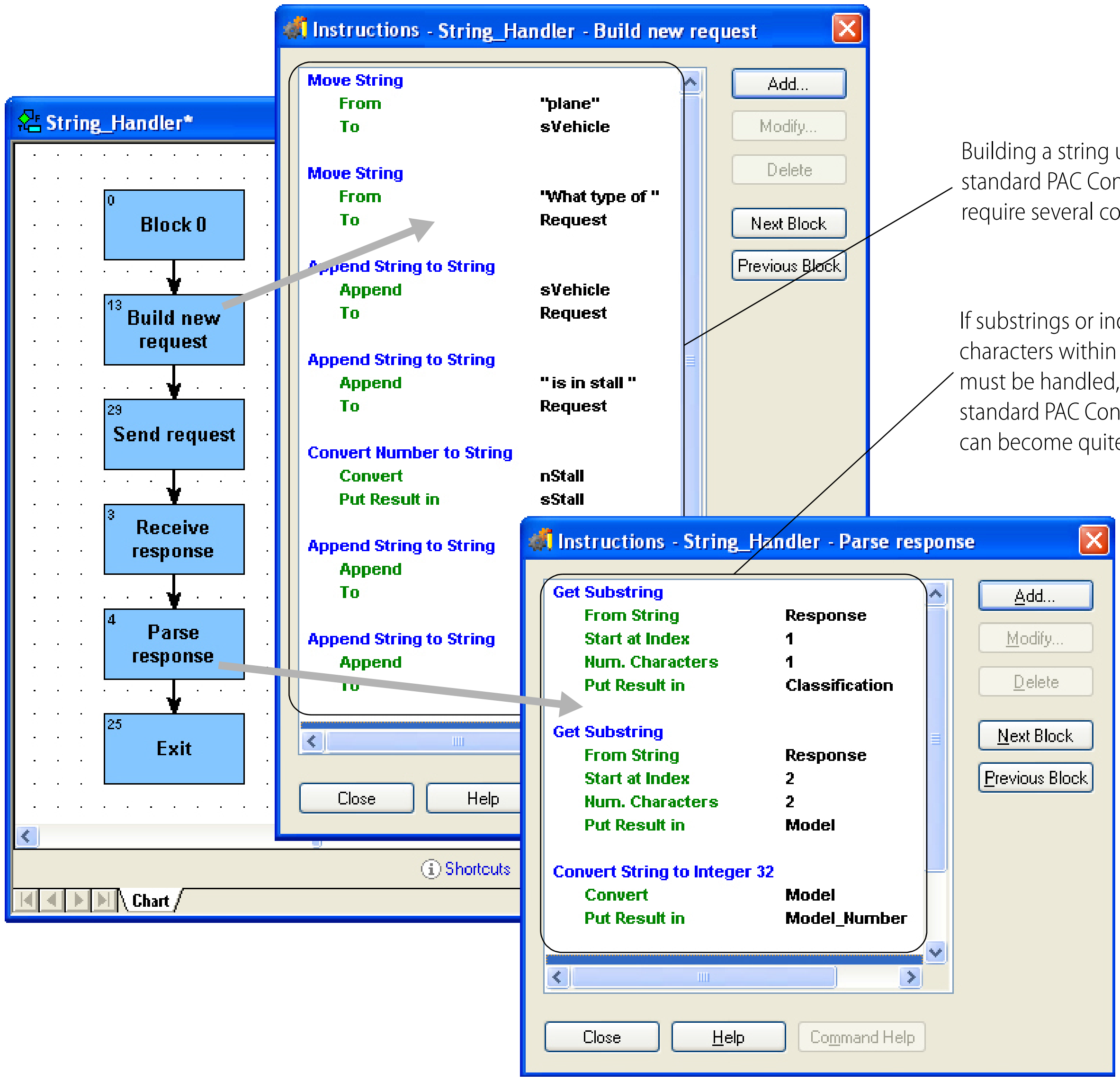
As you can see, the OptoScript version of this math expression is not only simpler to create, but also easier to understand once created.

For String Handling

If your strategy transmits and receives serial data, you will want to try using OptoScript code. In standard PAC Control, forming and parsing (decoding) serial data can take several blocks. In OptoScript, string handling can be easier.

The following figure shows a flowchart designed to send the string request, "What type of plane?" and parse the response, "F14," into a classification (F) and a model number (14).

Compare these blocks and instructions with the ones on the following page, done in OptoScript.



```
Move String
    From                    "plane"
    To                      sVehicle

Move String
    From                    "What type of "
    To                      Request

Append String to String
    Append                  sVehicle
    To                      Request

Append String to String
    Append                  " is in stall "
    To                      Request

Convert Number to String
    Convert                 nStall
    Put Result in           sStall

Append String to String
    Append
    To

Append String to String
    Append
    To
```

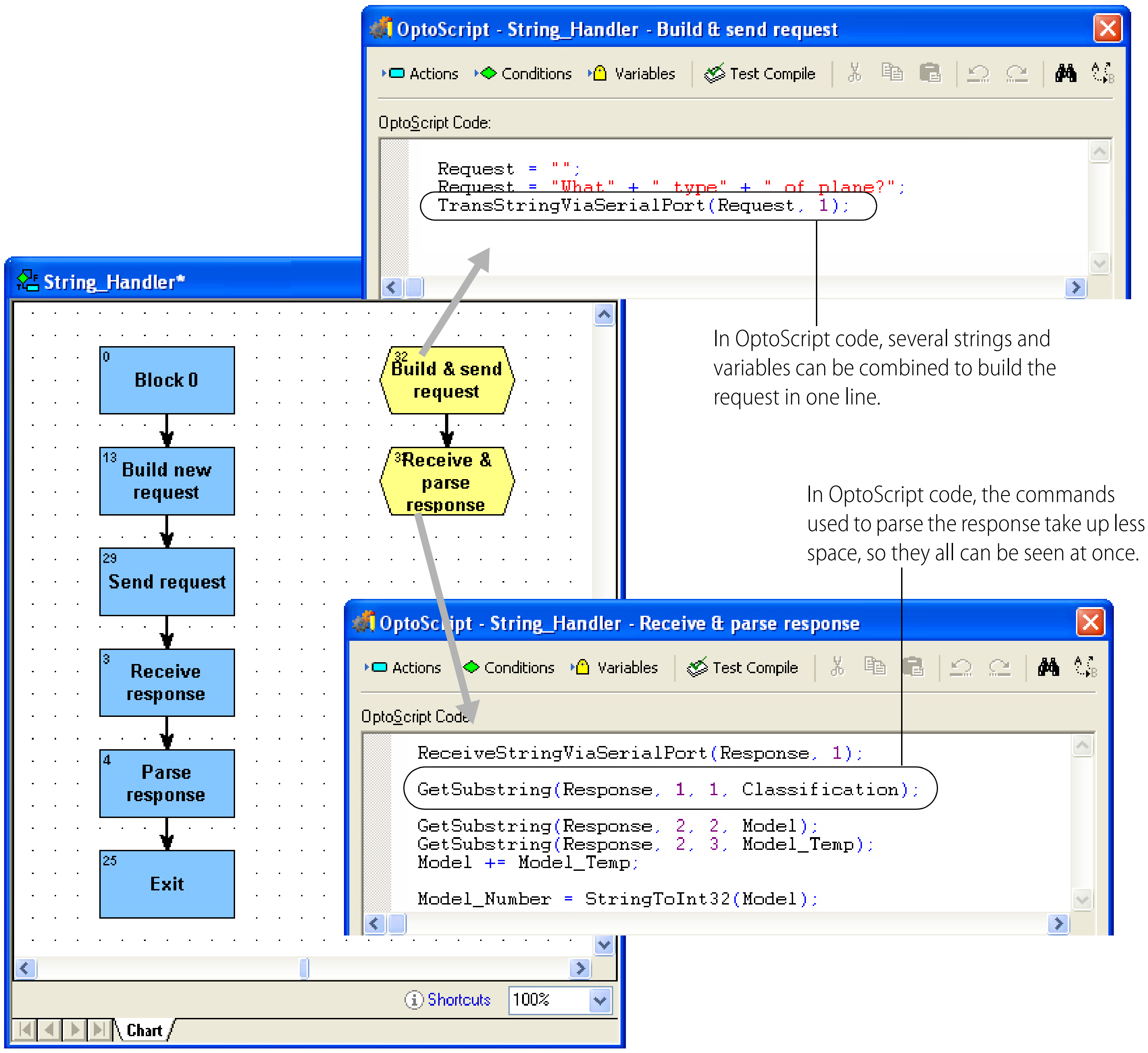
```
Get Substring
    From String             Response
    Start at Index          1
    Num. Characters         1
    Put Result in           Classification

Get Substring
    From String             Response
    Start at Index          2
    Num. Characters         2
    Put Result in           Model

Convert String to Integer 32
    Convert                 Model
    Put Result in           Model_Number
```

Building a string using standard PAC Control can require several commands.

If substrings or individual characters within a string must be handled, a standard PAC Control block can become quite large.

The OptoScript version of the String_Handler flowchart is more compact. The string request can be built more easily, and parsing the response takes up much less space.

If you handle more complex serial data than in the String_Handler example, you will find OptoScript code even more useful.



```
Request = "";
Request = "What" + " type" + " of plane?";
TransStringViaSerialPort(Request, 1);
```

```
ReceiveStringViaSerialPort(Response, 1);

GetSubstring(Response, 1, 1, Classification);

GetSubstring(Response, 2, 2, Model);
GetSubstring(Response, 2, 3, Model_Temp);
Model += Model_Temp;

Model_Number = StringToInt32(Model);
```

In OptoScript code, several strings and variables can be combined to build the request in one line.

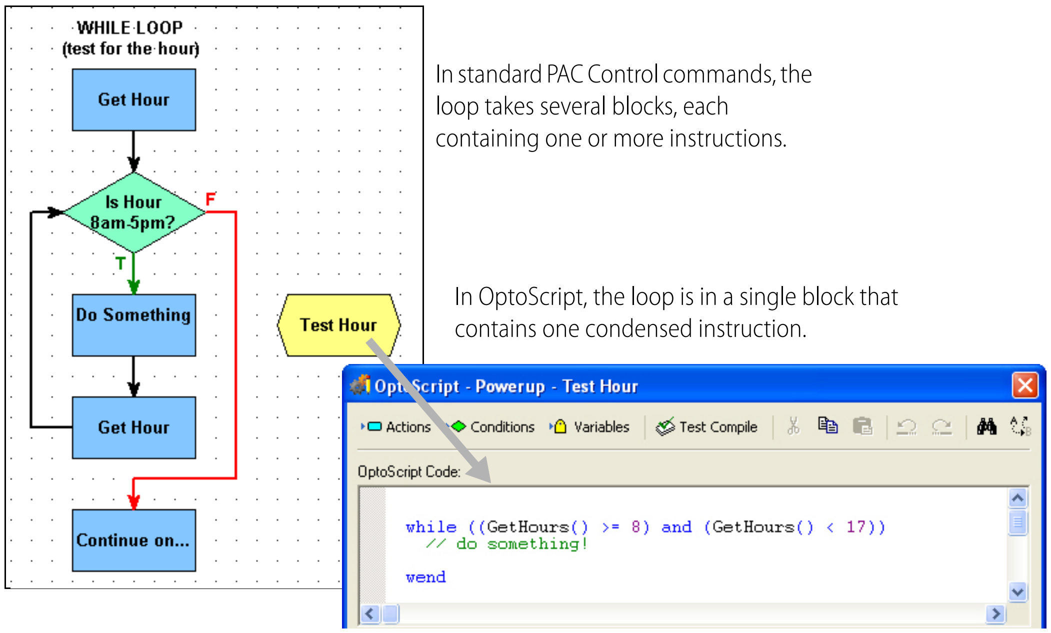
In OptoScript code, the commands used to parse the response take up less space, so they all can be seen at once.

For Complex Loops

Strategies that use complex loops—for example, to repeat an operation while a condition remains true—are easier to create and take up less space in a flowchart when done in OptoScript. *While* loops, *repeat* loops, and *for* loops are all available.

- **While loops** repeat a process while a test is true (the test comes at the beginning of the process).
- **Repeat loops** repeat a process until a test is false (the test comes at the end of the process). This kind of loop is guaranteed to execute at least once.
- **For loops** repeat a process for a specified number of times.

Below is an example of a *while* loop as it would appear in standard flowchart commands, contrasted with the way it could be handled in an OptoScript block.



In standard PAC Control commands, the loop takes several blocks, each containing one or more instructions.

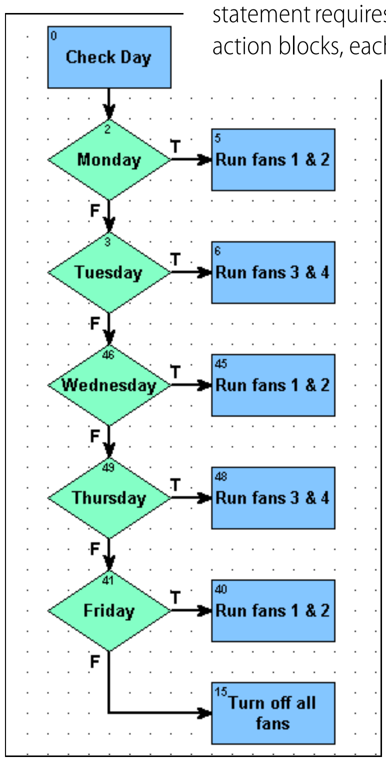
In OptoScript, the loop is in a single block that contains one condensed instruction.

```
while ((GetHours() >= 8) and (GetHours() < 17))
  // do something!

wend
```

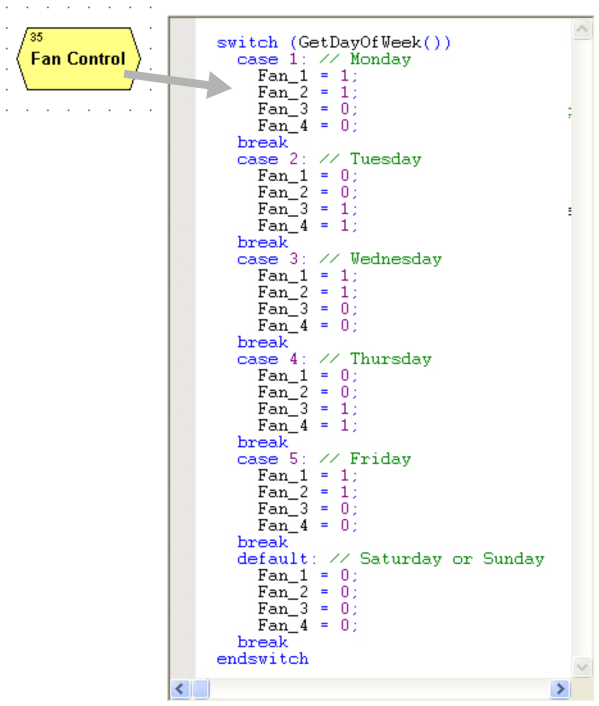
For Case Statements

Case or *switch* statements create multiple decision points. They can also be easier to do using OptoScript. Here is an example of a case statement:

In standard PAC Control commands, the case statement requires several sets of condition and action blocks, each containing commands.



In OptoScript, the code is all in one block.

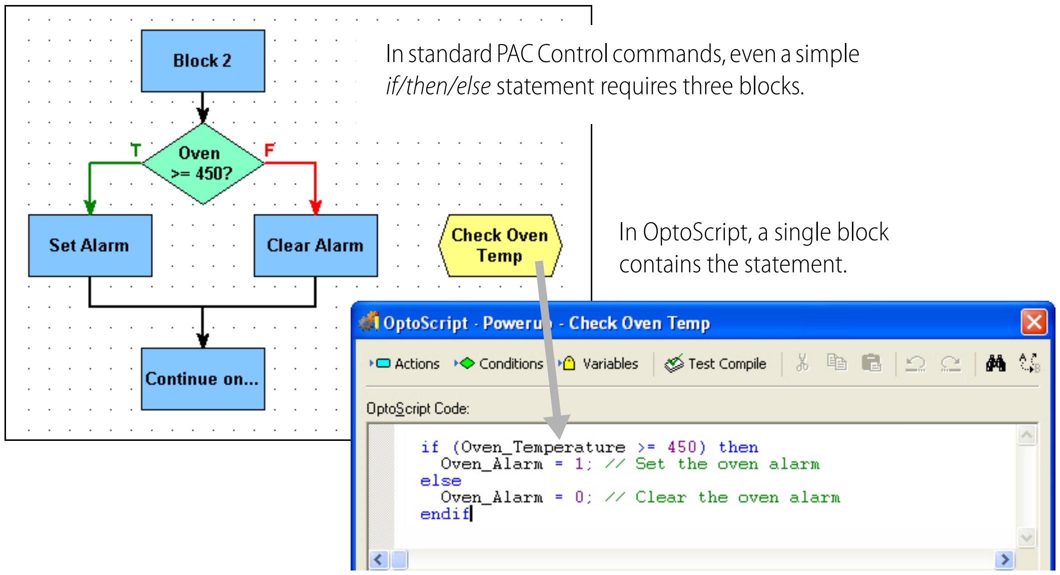


```
switch (GetDayOfWeek())
  case 1: // Monday
    Fan_1 = 1;
    Fan_2 = 1;
    Fan_3 = 0;
    Fan_4 = 0;
  break
  case 2: // Tuesday
    Fan_1 = 0;
    Fan_2 = 0;
    Fan_3 = 1;
    Fan_4 = 1;
  break
  case 3: // Wednesday
    Fan_1 = 1;
    Fan_2 = 1;
    Fan_3 = 0;
    Fan_4 = 0;
  break
  case 4: // Thursday
    Fan_1 = 0;
    Fan_2 = 0;
    Fan_3 = 1;
    Fan_4 = 1;
  break
  case 5: // Friday
    Fan_1 = 1;
    Fan_2 = 1;
    Fan_3 = 0;
    Fan_4 = 0;
  break
  default: // Saturday or Sunday
    Fan_1 = 0;
    Fan_2 = 0;
    Fan_3 = 0;
    Fan_4 = 0;
  break
endswitch
```

Using OptoScript for case statements saves space in the flowchart and lets you see all the possible cases in one dialog box.

For Conditions

Like loops and case statements, conditions can be simpler when done in OptoScript code. *If/then*, *if/then/else*, and *if/then/elseif* statements can all be mixed and nested as needed. Here's an example of a simple *if/then/else* statement as it could be done in standard PAC Control commands and in OptoScript:

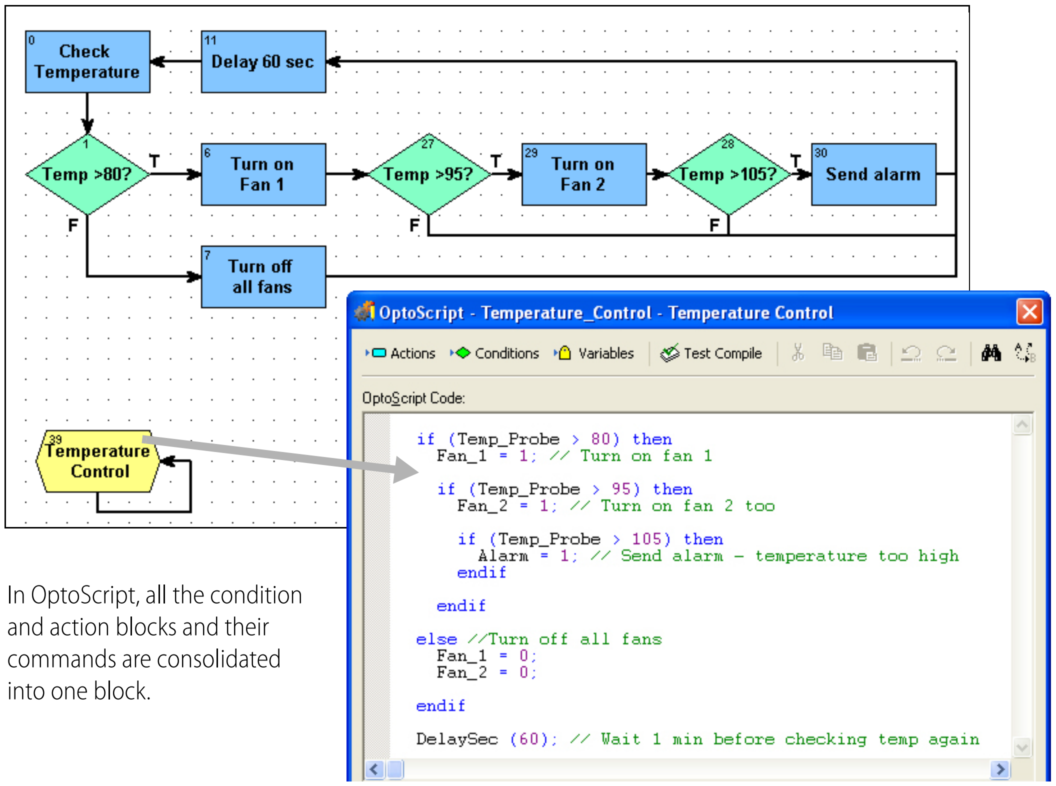


In standard PAC Control commands, even a simple *if/then/else* statement requires three blocks.

In OptoScript, a single block contains the statement.

```
if (Oven_Temperature >= 450) then
  Oven_Alarm = 1; // Set the oven alarm
else
  Oven_Alarm = 0; // Clear the oven alarm
endif
```

OptoScript is even more useful for more complex conditions, such as the following:

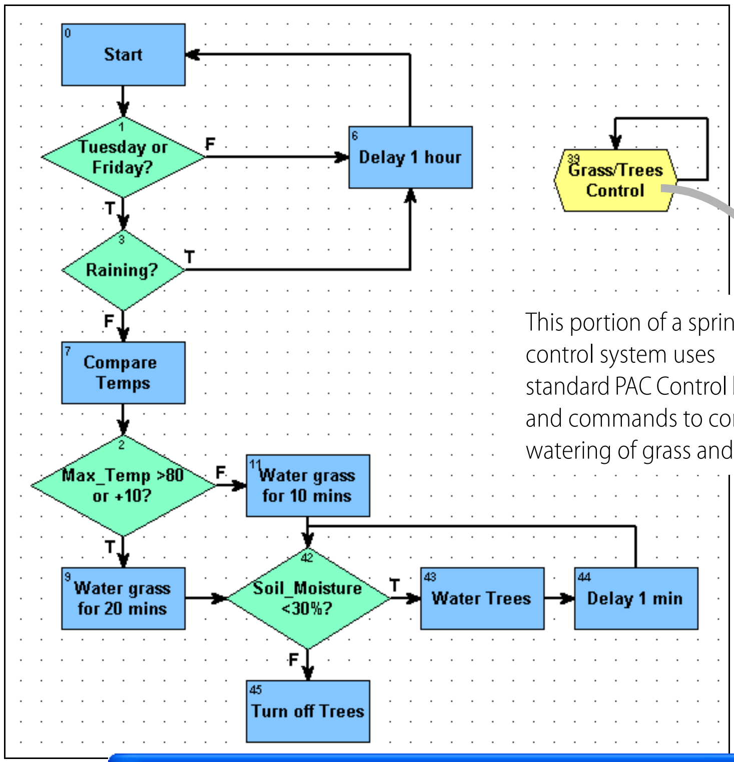


```
if (Temp_Probe > 80) then
  Fan_1 = 1; // Turn on fan 1

  if (Temp_Probe > 95) then
    Fan_2 = 1; // Turn on fan 2 too

    if (Temp_Probe > 105) then
      Alarm = 1; // Send alarm - temperature too high
    endif

  endif

else //Turn off all fans
  Fan_1 = 0;
  Fan_2 = 0;
endif

DelaySec (60); // Wait 1 min before checking temp again
```

In OptoScript, all the condition and action blocks and their commands are consolidated into one block.

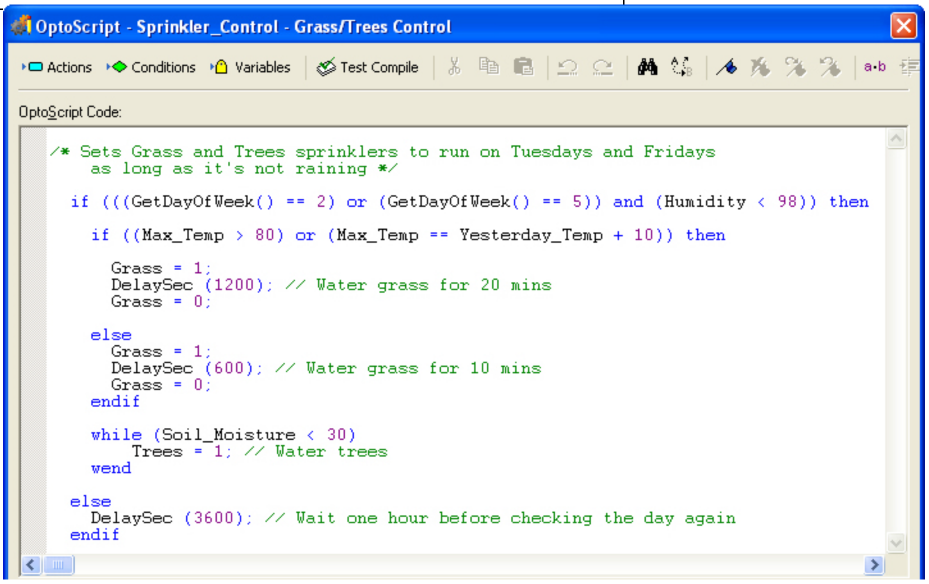
For Combining Expressions, Operators, and Conditions

The real power of OptoScript can be seen in complex operations.



This portion of a sprinkler control system uses standard PAC Control blocks and commands to control watering of grass and trees.

The OptoScript version of Grass/Trees Control handles the loops, conditions, and operators easily in a single block.



```
/* Sets Grass and Trees sprinklers to run on Tuesdays and Fridays
   as long as it's not raining */

  if (((GetDayOfWeek() == 2) or (GetDayOfWeek() == 5)) and (Humidity < 98)) then

    if ((Max_Temp > 80) or (Max_Temp == Yesterday_Temp + 10)) then

      Grass = 1;
      DelaySec (1200); // Water grass for 20 mins
      Grass = 0;

    else
      Grass = 1;
      DelaySec (600); // Water grass for 10 mins
      Grass = 0;
    endif

    while (Soil_Moisture < 30)
        Trees = 1; // Water trees
    wend

  else
    DelaySec (3600); // Wait one hour before checking the day again
  endif
```

Generally speaking, the more complex the combination of math expressions, logical and comparison operators, loops, and conditions, the more convenient it is to use OptoScript code rather than standard blocks and commands.

OptoScript Functions and Commands

Since functions in OptoScript are provided by commands almost identical to the standard commands in PAC Control, you have the same complete range of functions. There are no additional functions for OptoScript code, and you cannot make your own functions.

Standard and OptoScript Commands

In many cases you can easily recognize OptoScript commands, because they are almost the same as standard PAC Control commands. All spaces are removed from the OptoScript commands, however, and in some cases words in the command are abbreviated or left out. Commands are case sensitive. Here are some examples of the same commands in PAC Control and in OptoScript:

PAC Control Command	OptoScript Command
Get Counter	GetCounter
Set Down Timer Preset Value	SetDownTimerPreset
Delay (mSec)	DelayMsec
Convert Float to String	FloatToString
Get Number of Characters Waiting	GetNumCharsWaiting

Some commands are built into OptoScript functionality. Some of these have OptoScript commands and some do not; you can use either the built-in functionality or the OptoScript command, if it exists. Here are some examples:

PAC Control Command	OptoScript Command	Built-In Equivalent	Example
Move		=	item1 = value
Add		+	1 + 2
Less?		<	value1 < value2
Turn On	TurnOn	= [non-zero]	digital3 = 1
Turn Off	TurnOff	= 0	digital3 = 0
Comment (Single Line)		//	// comment
Set Nth Character	SetNthCharacter		s1[5] = 'c'

See Appendix E for a table of all PAC Control commands and their OptoScript equivalents. In addition, OptoScript equivalents for each command are shown in the *PAC Control Command Reference* and in the online command help.

Using I/O in OptoScript

One advantage of OptoScript is that any named I/O point can be used directly, wherever a numeric variable can be used, rather than requiring a variable. Digital points behave like integer variables that have only two possible states: zero (off) or non-zero (on). Analog points behave like float variables.

For example, you can turn a digital point off by simply assigning it a value of zero:

```
Light_Switch = 0;
```

You can turn a digital point on by assigning it any value other than zero:

```
Light_Switch = 1;
Light_Switch = -1;
Light_Switch = 486;
```

You can use I/O points directly in mathematical expressions:

```
fLimit = Pressure_Input + 50;
```

Or use them directly in control structures, for example to turn off the light if the door is closed:

```
if (not Door) then
  Light_Switch = 0;
endif
```

You can set an output based on the value of an input or a variable:

```
LED01 = Switch_A;
Proportional_Valve = fPressure_Control
```

You can use a point directly with a command:

```
fRange = GetAnalogMaxValue(Temp_Input) - GetAnalogMinValue(Temp_Input);
TurnOn(Fan_A);
IsOn(Fan_A);
```

OptoScript Syntax

Here is a sample section of OptoScript code to illustrate syntax.

```
fPressure = 300.0;

nTotal = ntTable[0] + ntTable[1] + ntTable[2];

while ((GetHours() >= 8) and (GetHours() < 17))
  Fan_A = 1;
wend

// Send alarm if oven temperature too hot.
if (Oven_Temperature >= 450) then
  Oven_Alarm = 1; // Set the oven alarm
else
  Oven_Alarm = 0; // Clear the oven alarm
endif

nCheck = GenerateChecksumOnString (0, sMessage);
nError_Block = GetIdOfBlockCausingCurrentError();
RemoveCurrentError();

sGreeting = “Hello, world!”;
nPos = FindCharacterInString('!', 0, sGreeting);
```

Each statement is followed by a semicolon.

Table elements are put in square brackets next to the table name.

Parentheses are used as separators for expressions and operators. You can use an unlimited number of parentheses.

Line comments appear on a separate line or after a statement. They are preceded by two slashes and a space Block comments (not illustrated) are preceded by /* and followed by */.

Parameters (arguments) for a command are listed in order within parentheses following the command. Commands that have no arguments must still include the parentheses.

An individual character can be in single quotes or in double quotes, depending on its type. A string must be in double quotes.

NOTE: Each block has only one exit point. It is not possible to use `return` *to jump out of the current block.*

More About Syntax with Commands

As noted in the previous sample, arguments for a command are listed in the parentheses following the command. Arguments are listed in order beginning with argument 1. To find out the arguments for any command, see the *PAC Control Command Reference* or online command help.

```
SetDownTimerPresetValue (60.0, Minute_Timer)

            command              (argument 1, argument 2)


EnableIOUnitCausingCurrentError   ()

                 command                     (no arguments)
```

There are two *types* of OptoScript commands: procedure commands and function commands.

Procedure commands accomplish an action and return no value. Here are some examples:

```
RemoveCurrentError();
ClampInt32TableElement(10, 0, 5, x1);
```

Function commands return a value from their action, so the value can be placed somewhere. In the following examples, the value is placed in the variable at the beginning of the statement:

```
nMonth = GetMonth();
fSquare_Root = SquareRoot(99);
nPosition = FindCharacterInString('S', 0, sName);
```

When you compare these examples to the identical commands in standard PAC Control, you'll notice that the returned value for the standard PAC Control command is an argument. In OptoScript the returned value is not an argument, thus reducing the number of arguments by one. In the first example, the standard command Get Month has one argument, which is where the result is placed. The OptoScript command equivalent, GetMonth, has no arguments and places the result in the variable.

In most cases you will use the value a function command returns by placing it in a variable, a control structure, or a mathematical expression. Occasionally, however, you may not need to use the result. For example, the command StartChart returns a status. If you do not need to track the status, you can ignore it by not placing the result anywhere, for example:

```
StartChart(Fan_Control);
```

OptoScript Data Types and Variables

Unlike most procedural languages, PAC Control maintains a database of all declared variables, which is shared with PAC Display. Variables are not declared in OptoScript code, but are created (declared) within PAC Control. (See "9: Using Variables and Commands.") Variables are not declared in OptoScript because local variables are not allowed. All variables are global for the strategy (or global within a subroutine).

If you use a variable in OptoScript code that does not currently exist in the strategy, you'll receive an error message when you test compile the code and can add the variable then.

Variable Name Conventions

With OptoScript and in PAC Control generally, it's a good idea to get into the habit of indicating the variable type in each variable's name. Some variable types may be obvious in the name itself, but others are not. For more information, see "Naming Conventions" on page 85.

Using Numeric Literals

Here are examples of how to use numeric literals in OptoScript. Formats are automatically converted if they don't match the variable type. For example, if a value of 300.2 were assigned to an integer 32, the value would be converted to 300.

Decimal Integer 32 Literals assigned to variables:

```
nVariable1 = 0;
nVariable2 = 10;
nVariable3 = -123;
```

Decimal Integer 64 Literals assigned to variables. Integer 64s have an i64 at the end:

```
dVariable1 = 0i64;
dVariable2 = 10i64;
dVariable3 = -123i64;
```

Hexadecimal Integer 32 Literals assigned to variables. Hex notation starts with 0x. Digits A–F may be upper or lower case:

```
nVariable1 = 0x0;
nVariable2 = 0x10;
nVariable3 = 0x12AB34CD;
nVariable3 = 0x12ab34cd;
```

Hexadecimal Integer 64 Literals assigned to variables:

```
dVariable1 = 0x0i64;
dVariable2 = 0x10i64;
dVariable3 = 0x1234567890ABCDEFi64;
```

Float Literals assigned to variables (Float literals may use scientific notation):

```
fVariable1 = 0.0;
fVariable2 = 12.3;
fVariable3 = -123.456;
fVariable3 = -1.23456e2;
fVariable3 = -12345.6e-2;
```

Making Assignments to Numeric Variables

Values are easily assigned to variables.

Simple Integer 32 assignments:

```
n1 = 1;
n2 = n1;
```

Simple Integer 64 assignments:

```
nn1 = 2i64;
nn2 = nn1;
```

Simple Float assignments:

```
f1 = 3.0;
f2 = f1
```

Simple assignments between different data types (Types will be automatically converted to match):

```
n1 = 4.0;
nn1 = n1;
f1 = n1;
```

Using Strings

As noted in the section on syntax, a string in OptoScript must be in double quotes. An individual character can be used either as a string (in double quotes) or as an integer value representing that character in ASCII (in single quotes). When you assign a single character to a string, use double quotes to avoid a syntax error:

```
sString = "a";
```

To change a single-character integer into a string, use the `Chr()` keyword as shown below:

```
sString = Chr('a');
sString = Chr(97);
```

```
n = 97;
sString = Chr(97)
```

NOTE: If you intend to retrieve the string data in either PAC Display or OptoOPCServer, to avoid an error do not use the Chr( ) keyword character to insert an embedded null, e.g. Chr(0).

Strings can be used in the following ways.

String literals (must be all on one line):

```
sGreeting = "Hello, world!"
```

When you use the Chr() keyword to assign a character value to a string variable, you can either quote a character or give its ASCII value. For example, the following two statements are equivalent

```
sString1 = Chr('A');
sString1 = Chr(65);
```

String variables:

```
sOutgoing = sIncoming;
```

```
A string can be thought of as a table of
characters. The number in square
brackets is the character’s index.
(Note that the index starts with the
number zero.) The following code would
result in sGreeting equaling “Hello!!!”
sGreeting = "Hello...";
sGreeting[5] = '!';
sGreeting[6] = '!';
sGreeting[7] = sGreeting[6];
```

A character element of a string variable may be treated like an Integer 32 value:

```
nNumber = sString2[1] * sString2[2];
```

Clear a string using empty quotation marks:

```
sString1 = "";
```

The + operator is used to paste strings together. There is no limit to the number of + operators you can use on a line. The + operator must be used in an assignment statement:

```
sString1 = "Hello ";
sString2 = "world";
sString3 = "!";
After the three lines above, the following two lines would produce the
same result:
sString4 = sString1 + sString2 + sString3;
sString4 = sString1 + "world" + sString3;
```

Use the += operator to append one string to another and change the value of one of them into the result. In the following example, the value of sName would change to “Smith, John”:

```
sName = "Smith, ";
sFirstName = "John";
sName += sFirstName;
```

The Chr() keyword can be used to convert a numeric value into a one-element string:

```
sString5 = sString1 + sString2 + Chr('!');
sString5 = sString1 + sString2 + Chr(33);
```

Working with Pointers

The use of pointers is an advanced programming technique that is very powerful. For more information on using pointers, see "Pointer Commands" on page 355.

NOTE: Except for the equal operator (==), the comparison operators (see page 386) can be used only with a numeric expression. They cannot be used with a null which is not considered to be a number.

For the following examples, assume that:

```
n1 = 5;
f1 = 9.2;
s1 = "test 123";
```

Using the "&" Character to Point to a Tag's Address

A pointer variable or pointer table element contains the address of the object (tag) it is pointing to; this is different than the tag itself. In OptoScript, the "&" means *address of*. It is used as a prefix to a tag in OptoScript when moving the address of a tag into a pointer. PAC Control commands were designed to act on tags and not just the address of the tags. The "&" prefix tells the controller to use the address of the tag and not the tag itself.

Set the pointer. The types must match or the control engine will generate an error.

```
pn1   = null;
pn1   = &n1;
pf1   = &f1;
ps1   = &s1;
pcht1 = &Powerup;
```

Using the "&" Character to Move a Pointer

When moving a pointer to pointer, use the "&*" prefix.

To move a value from a pointer to another pointer:

pVariable0 = &*pVariable1;

To move a value from a pointer to a pointer table:

ptTable[0] = &*pVariable4;

Using the "*" Character to De-reference a Pointer

Use "*" to de-reference a pointer. It will then behave just like the variable to which it is pointing. The following two statements are equivalent:

n2 = *pn1 + *pf1

n2 = n1 + f1;

The "*" character has to be used as a prefix to a pointer in OptoScript so that the tag that is pointed to will be used instead of the address of the tag. When using a pointer with normal PAC Control commands you need to de-reference the pointer because PAC Control commands are designed to use tags instead of the address of the tags.

Using the Comparison Operator "=="

To see if a pointer is pointing to something, use the comparison operator == (see page 386) to compare it to null. This use is similar to standard PAC Control condition commands such as Pointer Equal to NULL?

For example:
```
n2 = pn1 == null;
n2 = null == pn1;
if (pt1[0] == null) then
```

Determining Which Variable to Use

Pointers are very useful when you don't know what variables need to be used until runtime. For instance, the next example uses a switch statement (see page 389) to determine which variable to use based on the day of the week. It then uses a pointer to perform a calculation using the correct variable.

```
switch (GetDayOfWeek())
   case 0:  // Sunday
      pn1 = n2;
      break
   case 6:  // Saturday
      pn1 = n3;
      break
   default: // Monday-Friday
      pn1 =5 n4;
      break
endswitch
```

Use the pointer to set the chosen variable.
```
*pn1 = n5 * f1 - 5;
```

For More Information

For more information and examples of using pointers in OptoScript, see the following:

- Pages "Pointer Commands" on page 355 and "Pointers and Indexing" on page 96.
- See *Appendix E: OptoScript Language Reference,*
- Download the OptoScript Examples. Either click here, or go to the Opto 22 website and search for "OptoScript Examples."

Working with Tables

Following are some examples for using numeric, string, and pointer tables.

Numeric tables:
```
ntTable1[0] = 1;
ntTable1[1] = 2.0;
ntTable1[2] = nVar1;
ntTable1[3] = ntTable1[2];
ntTable1[4] = ntTable1[ntTable1[0]];
ntTable1[5] = nVar1 + ntTable1[2] * 3.1;
nVar1 = ntTable1[0];
nVar1 = (ntTable1[0] + ntTable1[1]) * ntTable1[2];
```

String tables:
```
stStrT1[0] = "Hello, ";
stStrT1[1] = "world";
stStrT1[2] = stStrT1[0] + " " + stStrT1[1] + Chr('!');
sString1 = stStrT1[2];
```

Pointer tables:
```
ptTable6[0] = &*pVariable2;
ptTable6[1] = &nVar1;
ptTable6[2] = &stStrT1[0];
stStrT1[0] = *ptTable6[2];
```

Pointer tables. Note that types are not checked when putting pointers into a pointer table. However, when a pointer is moved from a pointer table element into a pointer variable, the types are checked at runtime by the control engine and must match. For example, assume that the following elements have been placed in table ptPointT:
```
ptPointT[0] = null;
ptPointT[1] = &nLED_A;
ptPointT[2] = &fTemp;
ptPointT[3] = &sString1;
ptPointT[4] = &Powerup;
```
Based on this information, the first two of the following statements are good. The third one is bad and will cause a control engine error, because the element at ptPointT[3] is a string and therefore does not match the variable pntl, which is defined as an integer 32:
```
pn1 = ptPointT[1];
pf1 = ptPointT[2];
pn1 = ptPointT[3];
```

OptoScript Expressions and Operators

OptoScript includes mathematical expressions as well as comparison, logical, and bitwise operators. Because expressions and operators are built into the OptoScript language, several standard PAC Control commands such as Multiply, Bit Shift, and Greater Than or Equal? are not used.

Using Mathematical Expressions

Addition
```
nCount = nLast_Count + 2;
fPressure = 1.5 + fReading;
nTotal = nMonday + nTuesday + 10;
```

Subtraction
```
nNumber_A = nNumber_B - 250;
fRange = fMax_Temp - fMin_Temp;
```

Multiplication
```
nQuantity = nBoxes * 12;
nHours = nSeconds * 60 * 60;
fMax_Speed = fSpeed * 16.52;
```

Division
```
nBoxes = nCount / 6;
fConversion = fLimit / 2.0;
```

Modulo division. If any argument is a float, it is rounded to an integer before the division occurs.
```
nVar1 = nVar2 % 2;
nVar1 = 2 % nVar2 % nVar3;
fFloat1 = fFloat2 % 2.5;
```

Mixture of operators.
```
nAvg = (nHrs_A + nHrs_B) / 2;
nVar1 = fFloat2 + nVar3 * 4;
```

Use parentheses to clarify groupings and meaning. You can use an unlimited number of parentheses.
```
nVar1 = nVar2 * (fFloat2 - 2.0);
nVar1 = (nVar2 + 2) * (nVar3 + (fFloat1 / (fFloat2 - 2)) - 3);
```

The *, /, and % operators have greater precedence than + and -. (See Opto 22 page 447for the order of precedence.) In the following lines, line #1 is equivalent to line #3, not to #2.
```
n1 = n2 + n3 * n4;
n1 = (n2 + n3) * n4;
n1 = n2 + (n3 * n4);
```

Using Comparison Operators

All OptoScript comparison operators return an Integer 32 value of zero (false) or of non-zero (true). OptoScript supports the following comparison operators for comparing two numeric values:

NOTE: Except for the equal operator (==), the comparison operators can be used only with a numeric expression. They cannot be used with a null which is not considered to be a number.

Operator and Meaning		**Example**
==	equal	`nVar1 = nVar2 == fFloat3;`
<>	not equal	`nVar1 = nVar2 <> fFloat3;`
<	less than	`nVar1 = nVar2 < fFloat3;`
<=	less than or equal	`nVar1 = nVar2 <= fFloat3;`
>	greater than	`nVar1 = nVar2 > fFloat3;`
>=	greater than or equal	`nVar1 = nVar2 >= fFloat3;`

More complex examples:
```
nVar1 = (nVar2 * 2) == (fFloat3 / 9.5);
nVar1 = (nVar2 * 2) < (fFloat3 / 9.5);
```

You can also use a comparison operator to test whether two strings are equal. For example:

```
nVar1 = sString1 == sString2;
nVar1 = sString1 == "abc";
nVar1 = sString1 == stStrT1[0];
nVar1 = stStrT1[0] == stStrT1[1];
```

When you use a comparison operator in an *if* statement, it isn't necessary to put the result in a variable because the result is used (consumed) by the *if*:

```
if (fICTD_Input <= Avg_Temp) then
  Fan_A = 0;
endif
```

Using Logical Operators

All OptoScript logical operators return an Integer 32 value of zero (false) or of non-zero (true). OptoScript supports the following logical operators for numeric values:

Operator and Meaning		Example
`and`	Result is true if both values are true	`nVar1 = nVar2 and nVar3;`
`or`	Result is true if at least one value is true	`nVar1 = nVar2 or nVar3;`
`xor`	Result is true if only one value is true	`nVar1 = nVar2 xor nVar3;`
`not`	invert the logical value	`nVar1 = not nVar2;`

Any number of logical operators can be chained together:

```
nVar1 = nVar2 and nVar3 and nVar4;
nVar1 = nVar2 and nVar3 or nVar4;
```

Logical operators are left-associative. For example, these two lines are equivalent:

```
nVar1 = nVar2 and nVar3 or
nVar4;
nVar1 = (nVar2 and nVar3) or
nVar4;
```

The *not* operator precedes a value (it takes a value only on its right-hand side):

```
nVar1 = not nVar2;
```

The following two lines are equivalent:

```
nVar1 = not nVar1 and not nVar2;
nVar1 = (not nVar1) and (not
nVar2);
```

Logical operators can be combined with comparison operators to create complex logical expressions:

```
nVar1 = (nVar2 < 1) and (nVar3 == 6.5);
nVar1 = (nVar2 < 1) and (sString1 == "abc");
nVar1 = ((nVar2 < 1) and (nVar4 xor nVar5) or (not (fFloat1 == fFloat2));
nVar1 = not (nVar2 < 5); // same as  "nVar1 = nVar2 >= 5;"
```

When you use a logical operator in an *if* statement, it isn't necessary to put the result in a variable because the result is used (consumed) by the *if*:

```
if (Motor_1 or Motor_2) then
  Motor_3 = 0;
endif
```

Using Bitwise Operators

All OptoScript bitwise operators operate on integer values. OptoScript supports the following bitwise operators:

`bitand`	(bitwise and)
`bitor`	(bitwise or)
`bitxor`	(bitwise xor)
`bitnot`	(bitwise not)
`<<`	(left shift)
`>>`	(right shift)

Use the *bitwise and* operator to *and* together the two values bit by bit:

```
n1 = n2 bitand 2;
n1 = n2 bitand n3;
```

Hex literals can be convenient:

```
n1 = n2 bitand 0x0002;
```

Use the *bitwise or* operator to *or* together the two values bit by bit:

```
n1 = n2 bitor 2;
n1 = n2 bitor 0x0002;
n1 = n2 bitor n3;
```

Use the *bitwise xor* operator to *xor* together the two values bit by bit:

```
n1 = n2 bitxor 2;
n1 = n2 bitxor 0x0002;
n1 = n2 bitxor n3;
```

The *left-shift* operator shifts the left value's bits to the left by the right value:

```
n1 = n2 << 2;      // left shift n2's value by 2
n1 = n2 << n3;     // left shift n2's value by n3
```

The *right-shift* operator shifts the left value's bits to the right by the right value:

```
n1 = n2 >> 2;      // right shift n2's value by 2
n1 = n2 >> n3;    // right shift n2's value by n3
```

Precedence

For a list of operators from highest to lowest precedence, see "Operators" on page 447.

OptoScript Control Structures

OptoScript provides the following structures to control the flow of logic in the code:

- "If Statements" (below)
- "Switch or Case Statements" on page 389
- "While Loops" on page 390
- "Repeat Loops" on page 391
- "For Loops" on page 391

If Statements

If statements offer branching in logic: if statement A is true, then one action is taken; if statement A is false (or statement B is true), a different action is taken. *If* statements are very flexible; here are several examples of ways you can use them.

Any numeric value can be tested by the *if* statement (NOTE: The value is first converted to an Int32):

```
if (n1) then
  f1 = 2.0;
endif
```

Since a comparison operator returns an Integer 32 value, it can be used as the test value:

```
if (n1 > 3) then
  f1 = 2.0;
  f2 = 6.5;
endif
```

Complex logical operations can also be used:

```
if ((n1 > 3) and (not n1 == 6)) then
  f1 = 2.0;
  f2 = 6.5;
endif
```

An optional `else` statement can be added:

```
if (n1 > 3) then
  f1 = 2.0;
  f2 = 6.5;
else
  f3 = 8.8;
endif
```

Multiple `elseif` statements can be used to chain together several tests. The `else` statement is still allowed at the end.

```
if (n1 > 3) then
  f1 = 2.0;
  f2 = 6.5;
elseif (n1 < -3) then
  f3 = 8.8;
elseif (n1 == 0) then
  f3 = f1 * f2;
else
  f1 = 0;
  f2 = 0;
  f3 = 0;
endif
```

If statements can be nested. Each `if` requires an `endif`:

```
if (n1 > 3) then
  f1 = 2.0;
  f2 = 6.5;

  if (n1 % 10) then
    f1 = f1 * 2;
    f2 = f2 * 3;
  else
    f3 = 0;
  endif
endif
```

Switch or Case Statements

A *switch* or *case* statement also offers branching logic and can be used in place of *if* statements when the expression can match one of a number of numeric values. The value for each case can be a numeric constant or a mathematical expression only. Comparisons and logical operators cannot be used in cases, nor can strings. If a case involves a float, the float is converted to an integer before use. Notice that only one case can be tested at a time.

Here's an example of a switch statement.

```
switch (nNumber)
  case 1:
    f1 = 10;
    break
  case 2:
    f1 = 15;
    break
  case (n2 * 2):
    f1 = 20;
    break
  default:
    f1 = 0;
    f2 = -1;
    break
endswitch
```

The value of the expression in parentheses, nNumber, is compared to each of the cases. If the case matches the value of nNumber, the action is taken.

Make sure you use a colon (:) after each case.

If a case matches the value of nNumber, the break statement after citation immediately exits the switch. Notice that a semicolon is not used after break.

You can use a mathematical expression as a case.

If no case matches, the default action is taken. Using a default is optional; if you use it, it must be at the end of the list.

A switch statement must be followed by `endswitch`.

While Loops

The *while* loop is used to execute a list of statements while a given condition is true. The condition is tested at the beginning of each loop.

For example, this loop sets the first five elements (elements 0 through 4) of a table (ntTable) to a value of 10:

```
nIndex = 0;
while (nIndex < 5)
  ntTable[nIndex] = 10;
  nIndex = nIndex + 1;
wend
```

Initialize the counter.
Execute loop if condition is true.
Set the table element.
Increment the counter.

While loops can be nested and can contain other kinds of program statements. Each `while` needs a matching `wend` at the end. For example:

```
n1 = 0;
 while (n1 < 100)
   while ((n1 > 50) and (n1 < 60))
     nt1[n1] = n1 * 100;
     n1 = n1 + 1;
   wend
   nt1[n1] = n1;
   n1 = n1 + 1;
 wend
```

Repeat Loops

Repeat loops, in contrast to *while* loops, are used to execute a list of statements until a given condition is true. Because the condition is tested at the end of each loop, the content of the loop will always be executed at least once.

This example sets the first five elements of ntTable to 10. Compare this example to the example for *while* loops to see the difference.

```
nIndex = 0;
repeat
  ntTable[nIndex] = 10;
  nIndex = nIndex + 1;
until (nIndex >= 5);
```

Initialize the counter.
Set the table element.
Increment the counter.
Exit loop if condition is true.

Repeat loops can be nested and can contain other kinds of program statements. Each `repeat` statement needs a matching `until` statement at the end.

For Loops

For loops can be used to execute a list of statements a certain number of times.

The `for` line sets up a predefined initial value and a predefined final value for the counter that counts the repetitions. The line also includes the steps by which the counter gets from its initial value to its final value (step 1 counts by ones; step 2 counts by twos, and so on). The `step` is required. The counter can be any numeric variable or I/O point, but its value will always be a whole number. The initial value, final value, and step can be any numeric expression; they are converted to integer 32s.

***CAUTION**: A step value of zero creates an infinite loop. A float step value between –0.5 and 0.5 also creates an infinite loop, since it is rounded to zero when converted to an integer 32.*

This example results in nVariable equaling 6:

```
nVariable = 1;
 for nCounter = 0 to 4 step 1
   nVariable = nVariable + 1;
 next
```

The counter starts at zero, and its final value is 4. It will count up one step at a time.

The `for` loop must end with `next`.

The *for* loop counter can be used in the loop. This example sets the first five elements of table ntTable to 10:

```
for nIndex = 0 to 4 step 1
  ntTable[nIndex] = 10;
next
```

Other step amounts can be used, including negative steps. Do not use a zero step, which creates an infinite loop. This example sets elements 0, 2, and 4 of ntTable to 20:

```
for nIndex = 0 to 4 step 2
  ntTable[nIndex] = 20;
next
```

Predefined values can be a numeric expression, but they are evaluated only at the beginning of the loop. For instance, the following example will loop 0 to 15 because the upper limit of nSide*3 is evaluated only at the beginning of the loop, not each time through the loop:

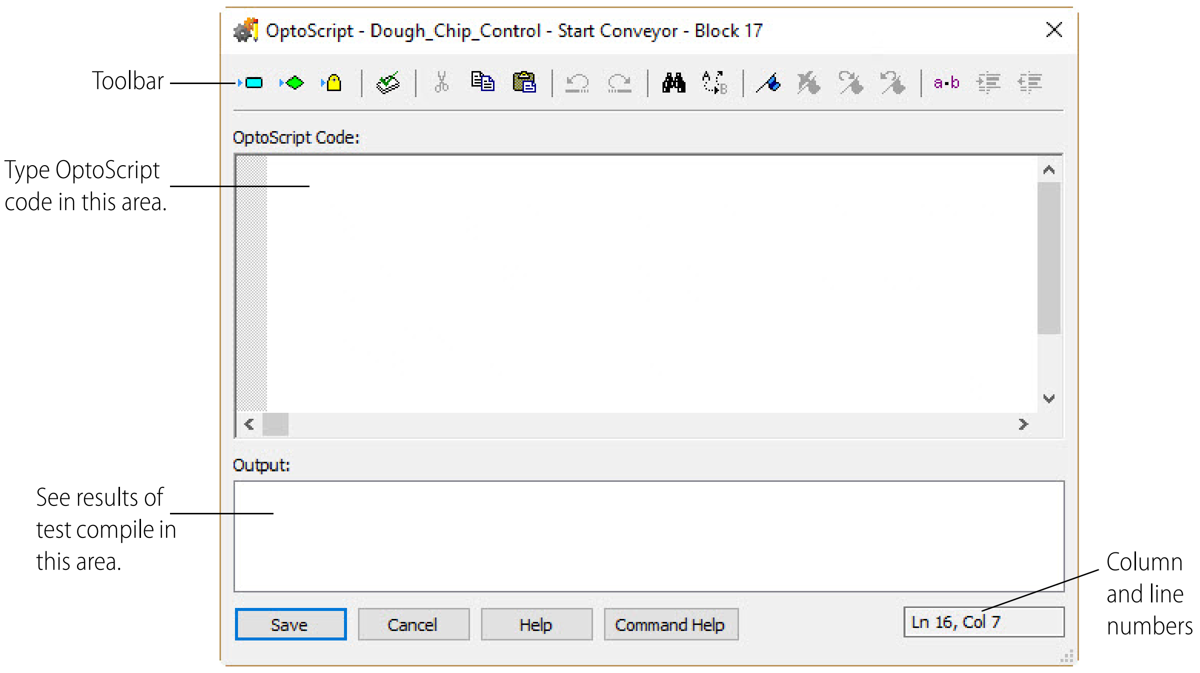
```
nSide = 5;
for nLength = 0 to (nSide * 3) step 1
  nSide = 1;
next
```

For loops can be nested and can contain other types of statements. Each `for` requires a `next` at the end.

Using the OptoScript Editor

1. To use the editor, create an OptoScript block in the flowchart where you want the code to appear. (For more information on creating charts and blocks, see "8: Working with Flowcharts.") Double-click the OptoScript block to open the editor.

   The editor is similar to the editor for Microsoft Visual Basic®. You can resize the editor window as needed to see the code.


Toolbar
Type OptoScript code in this area.
See results of test compile in this area.
Column and line numbers
OptoScript - Dough_Chip_Control - Start Conveyor - Block 17
OptoScript Code:
Output:
Save
Cancel
Help
Command Help
Ln 16, Col 7

The following tools are provided in the OptoScript toolbar:

Button	Description
Actions	Insert Action Command
Conditions	Insert Conditions Command
Variables	Insert Conditions Command
Test Compile	Test Compile
✂	Cut
	Copy
	Paste
	Undo
	Redo

Button	Description
	Find
A B	Replace
	Toggle Bookmark
	Clear All Bookmarks
	Next Bookmark
	Previous Bookmark
a·b	Toggle Whitespace
	Increase Indent
	Decrease Indent

2. Begin typing OptoScript code in the top area.

You'll notice that what you type is automatically color-coded to help you:

– Blue—operators and control structures

– Purple—values

– Green—comments

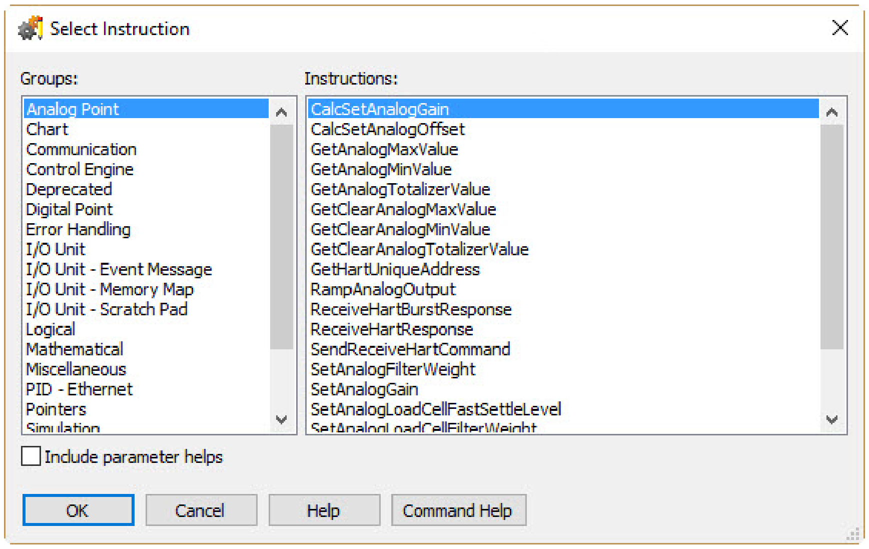
– Black—commands and names of variables, I/O points, charts, and other items

– Red—string literals.

If you want to see white-space marks to help line up code, click the Toggle Whitespace button in the toolbar. To hide the marks, click the button again.

3. To use a command, place your cursor in the OptoScript code where you want the command to appear.

Click the Insert Actions or Conditions button in the toolbar.


Select Instruction

Groups:

Analog Point
Chart
Communication
Control Engine
Deprecated
Digital Point
Error Handling
I/O Unit
I/O Unit - Event Message
I/O Unit - Memory Map
I/O Unit - Scratch Pad
Logical
Mathematical
Miscellaneous
PID - Ethernet
Pointers
Simulation

Instructions:

CalcSetAnalogGain
CalcSetAnalogOffset
GetAnalogMaxValue
GetAnalogMinValue
GetAnalogTotalizerValue
GetClearAnalogMaxValue
GetClearAnalogMinValue
GetClearAnalogTotalizerValue
GetHartUniqueAddress
RampAnalogOutput
ReceiveHartBurstResponse
ReceiveHartResponse
SendReceiveHartCommand
SetAnalogFilterWeight
SetAnalogGain
SetAnalogLoadCellFastSettleLevel
SetAnalogLoadCellFilterWeight

☐ Include parameter helps

4. In the Select Instruction dialog box, select the command group from the left-hand column, and then select the command name from the right-hand column.

For information on any command, highlight it and click Command Help, or just double-click the command name.

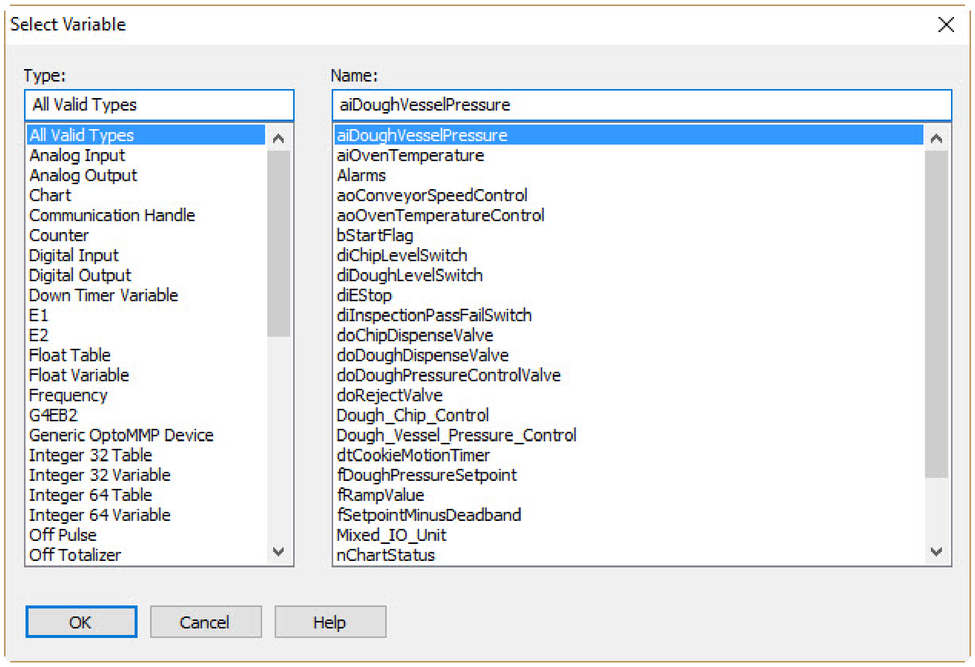
NOTE: If you know the command name, you can just type it into the OptoScript code. Remember that OptoScript command names may be different from standard PAC Control commands. See "E: OptoScript Language Reference," Opto 22 form 1700, for more information.

5. To include information about the instruction in the OptoScript code, select "Include parameter helps."

6. Click OK.

The command appears in the OptoScript code.

7. To use a variable, table, I/O unit or point, chart, counter, timer, or similar item, place your cursor where you want the item to appear in the code. If you know the item's exact name, enter it and skip to step 9. If you're not sure of the item's name, locate it in the Strategy Tree, enter it, and skip to step 9. Or for a variable, click the Add Variable icon in the toolbar.



Type	Name
All Valid Types	aiDoughVesselPressure
Analog Input	aiOvenTemperature
Analog Output	Alarms
Chart	aoConveyorSpeedControl
Communication Handle	aoOvenTemperatureControl
Counter	bStartFlag
Digital Input	diChipLevelSwitch
Digital Output	diDoughLevelSwitch
Down Timer Variable	diEStop
E1	diInspectionPassFailSwitch
E2	doChipDispenseValve
Float Table	doDoughDispenseValve
Float Variable	doDoughPressureControlValve
Frequency	doRejectValve
G4EB2	Dough_Chip_Control
Generic OptoMMP Device	Dough_Vessel_Pressure_Control
Integer 32 Table	dtCookieMotionTimer
Integer 32 Variable	fDoughPressureSetpoint
Integer 64 Table	fRampValue
Integer 64 Variable	fSetpointMinusDeadband
Off Pulse	Mixed_IO_Unit
Off Totalizer	nChartStatus

8. In the Select Variable dialog box, select the variable type from the left-hand column, and then select the variable name from the right-hand column.

The item appears in the code.

9. Use the Tab key on the keyboard as you type to indent lines as needed. To increase or decrease indentation for a line of code you've already typed, highlight the line and click the Increase Indent or Decrease Indent button in the toolbar.

10. Enter comments to document what the code does, so anyone who must debug or maintain the code can clearly see your intentions.

Comments appear in green. Line comments must be preceded by two slashes, for example:

```
// This is a line comment.
```

Block comments must be preceded by one slash and an asterisk, and be followed by the same two elements in reverse. For example:

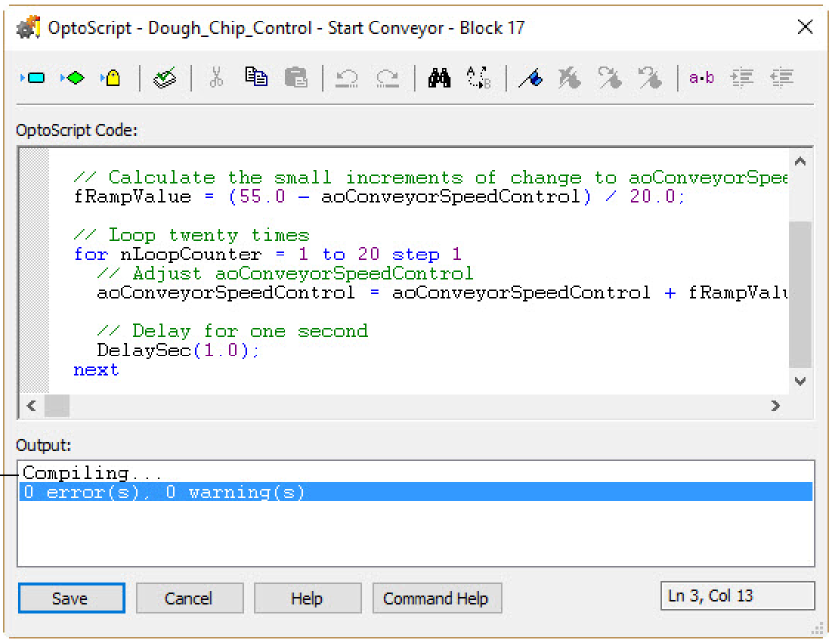
```
/* This is a block comment that goes
beyond one line. */
```

11. Use the Bookmark buttons in the toolbar as needed to set or clear temporary bookmarks within the code and to move between them.

Bookmarks mark lines of code so you can easily find them and jump from one bookmark to the next. Bookmarks remain only while the editor is open; they are not saved when the dialog box is closed.

12. When you have finished entering all the code for an OptoScript block, click the Test Compile button in the toolbar to compile the code for this block.

The code is compiled, and the results appear in the bottom part of the OptoScript window:


OptoScript - Dough_Chip_Control - Start Conveyor - Block 17
OptoScript Code:
Output:
Save
Cancel
Help
Command Help
Ln 3, Col 13
Results after code is compiled

```
// Calculate the small increments of change to aoConveyorSpee
fRampValue = (55.0 - aoConveyorSpeedControl) / 20.0;

// Loop twenty times
for nLoopCounter = 1 to 20 step 1
  // Adjust aoConveyorSpeedControl
  aoConveyorSpeedControl = aoConveyorSpeedControl + fRampValu

  // Delay for one second
  DelaySec(1.0);
next
```

```
Compiling...
0 error(s), 0 warning(s)
```

NOTE: The next time the chart is compiled, all OptoScript code within the chart will be compiled again.

If errors are found, you can fix them now or later. Begin with the first one (the one on the lowest-numbered line), since later errors are often a result of earlier errors. To check a command, place the cursor anywhere in the command name and click the Command Help button. If you need to add variables or other items that don't exist in the strategy, do so after step 12.

13. When you have finished with the code in this OptoScript block, click OK to save your work and close the editor.

You return to the flowchart.

Troubleshooting "Unable To Find" Errors

Also see "Troubleshooting Syntax Errors" below.

If you test compile an OptoScript block and receive "unable to find" errors, try the following suggestions.

For Commands

Check the exact spelling of the command, including upper and lower case. OptoScript commands are similar to standard PAC Control commands, but contain no spaces and some abbreviations.

Also check that the command is necessary in OptoScript. Some common commands, including comparison commands such as Less? and mathematical commands such as Add, are replaced with operators built into the OptoScript language. Check Appendix E: OptoScript Language Reference, for equivalent OptoScript commands.

The easiest way to make sure the command you enter is valid is to enter it by clicking one of the Insert Command buttons in the OptoScript Editor and choosing the command from the Select Instruction dialog box.

For Variables or Other Configured Items

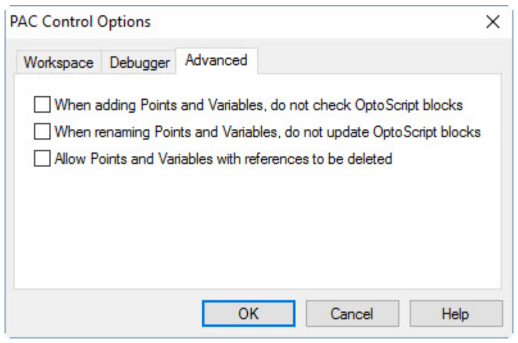
Variables, I/O units and points, counters, and other configured items in your strategy—as well as charts—have usually been created before you use them in OptoScript code. Check their exact spelling, including underscores and upper and lower case, to make sure they are correct in the code. The easiest way to make sure spelling is correct is to enter the variable or other item by clicking the Insert Variable button in the OptoScript Editor and choosing the item from the drop-down lists.

If the item has not yet been configured or created, use the normal PAC Control methods to do so. For help, see the chapters in this guide on configuring I/O and using variables.

IMPORTANT: In nearly all applications, you want points and variables to be identical in the Strategy Tree and in OptoScript blocks. For example, if you change the name of a variable or point in the Strategy Tree, you want the name changed in OptoScript blocks as well.

However, it is possible to turn off some of these automatic updates and cross-checks. To check whether they've been turned off:

1. With the strategy open in Configure mode, choose Configure > Options.
2. Click the Advanced tab.



- [ ] When adding Points and Variables, do not check OptoScript blocks
- [ ] When renaming Points and Variables, do not update OptoScript blocks
- [ ] Allow Points and Variables with references to be deleted

3. Look to see if any boxes are checked. If they are, you may need to make manual adjustments. (See more information in "Advanced Options" on page 235.)

Troubleshooting Syntax Errors

Check for the following common syntax errors:

Missing Code

Check for obvious errors first. For example, make sure nothing essential has been left out of (or unnecessarily added to) a statement:

Sample Statement	Should Be	Missing Code
`iTotal = x + y + ;`	`iTotal = x + y + z;`	Last operator missing a variable
`iTotal = x + y + z`	`iTotal = x + y + z;`	Semicolon missing
`sGreeting = Hello!"`	`sGreeting = "Hello!"`	First quotation mark missing on the string
`iTime = Get Hours;`	`iTime = GetHours();`	Extra space in command name; parentheses missing after the command
`x = (1 + (x - y);`	`x = (1 + (x - y));`	Parentheses mismatched (last half missing)

Check to make sure operators are used correctly. You may want to review "OptoScript Expressions and Operators" on page 385.

If you are using control structures such as loops or *if* statements, especially if they are nested, make sure all required elements are present. For example, every `if` must have a `then` and an `endif`. See "OptoScript Control Structures" on page 388 for more information.

Type Conflicts

Type conflicts are caused when different data types are incorrectly mixed. For example, you cannot assign an integer to a string. Make sure data types are correct. It is easier to keep track of data types if you use Hungarian notation when naming variables. See "Variable Name Conventions" on page 381 for help.

Debugging Strategies with OptoScript

Before trying to debug strategies containing OptoScript code, make sure the code has been compiled within each block, or choose Compile All to do all blocks at once.

When you begin debugging the strategy, start by stepping through whole blocks. If you run across a problem, then step within that block. Stepping within the block is discussed in "Choosing Debug Level" on page 211.

12: Using Subroutines

OPTO 22

Introduction

This chapter shows you how to create and use subroutines.

In this Chapter

About Subroutines 399
Creating Subroutines 401
Using Subroutines 407
Viewing Subroutines 411
Printing Subroutines 412

About Subroutines

A subroutine is a custom command that represents a series of commands. Subroutines are useful anytime you have a group of commands that is repeated in a strategy or used in more than one strategy. Subroutines are built using the same tools and logic used to create charts. Once built, you can call them at any time from any chart in any strategy or from another subroutine. (You cannot call subroutines recursively, though; that is, a subroutine cannot call itself, nor can a subroutine called by a subroutine call the original subroutine.)

NOTE: To call a subroutine from another subroutine, you need SNAP PAC firmware R9.5a or higher.

Like charts, subroutines start at block 0 and move sequentially through command blocks to the end. They use variables, inputs, and outputs. They can use OptoScript code. Each subroutine is displayed in its own window, and you can open and view several subroutine windows at once.

Unlike charts, however, subroutines are independent from a strategy. You don't need to have a strategy open to create or change a subroutine. And if you do have a strategy open, creating a subroutine has no effect on the open strategy unless you specifically link them together. (Debugging a subroutine, however, requires that it be called from a strategy.)

A second important difference between subroutines and charts is that subroutines offer two ways to work with variables and other logical elements: they can be passed in or they can be local to the subroutine.

- **Passed-in items** are passed into the subroutine by the strategy. They are referenced when the subroutine is executed. If a parameter is *passed by reference*, the subroutine will pass the value back to the strategy. For example, you could create a subroutine to add 3.0 to a passed-in float variable. If they are *passed by value*, the updated value is not passed back to the strategy by the subroutine. For more information, see the next section, "Data Types for Subroutines."

  When the subroutine ends, the float variable will contain a new value. Passed-in items are called *subroutine parameters*, and you can use up to 12 of them in a subroutine.
- **Local items** are created when a subroutine begins, and they are destroyed when it ends. For example, a subroutine could take a passed-in item, copy it to a local variable, and add 3.0 to that local variable for use within the subroutine. The local variable is created when the subroutine is called, and it disappears when the subroutine ends.

Data Types for Subroutines

The following data types may be used in subroutines for both passed-in items and local items:

- Numeric variables (integers, floats, and timers)
- Numeric literals (integers and floats). If you are familiar with other programming languages, literals are similar to "passed by value" parameters, while variables are like "passed by reference" parameters. Other types besides numeric can be passed into the subroutine through literals, however; see below for more information.
- Numeric tables
- String variables
- String literals
- String tables
- Communication handles

The following data types are also supported:

- For passed-in items: I/O points, I/O units, and pointer tables
- For local items: pointer variables

The following data types are not supported in subroutines: PID loops, event/reactions, and charts.

Although most variables passed in and out of a subroutine must be of a specific type, *literals* that are passed *into* subroutines can take several types. Using a string literal, you can pass in either a string literal or a string variable. Using a numeric literal, you can pass in an analog point, a digital point, an integer variable, a float variable, or a timer variable.

This flexibility in using literals makes it easier to use a subroutine in multiple strategies. For example, a literal passed into a subroutine from two strategies might be a float value in one strategy and an analog point in the other.

Creating Subroutines

Tips for Subroutines

As a general rule, keep subroutines as small as possible to do the job they're intended for. Extra variables and unnecessarily large table sizes can affect the memory available for running subroutines.

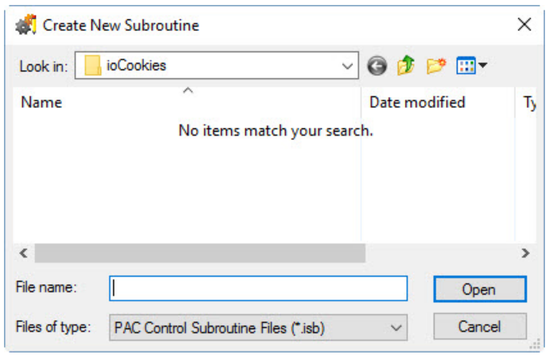
A Put Status In parameter appears automatically in every subroutine. This parameter is used to let you know whether the subroutine was called successfully, in the same way that function commands return a status. Make sure that you always check the status code after calling a subroutine. Subroutine status codes are:

0	Success
-67	Out of memory
-69	Null object error. Make sure you are not passing a pointer that points to null.
-72	Nesting too deep
-73	Invalid subroutine or parameters. In configure mode, choose Edit > Find, select Global and Operand, and then click Find to search for objects with the same name as the subroutine. Rename any objects that you find.

Drawing the Flowchart

1. In the PAC Control main window (with or without a strategy open), choose Subroutine > New.

   The Create New Subroutine dialog box appears.


Create New Subroutine
Look in: ioCookies
Name
Date modified
Ty
No items match your search.
File name:
Files of type: PAC Control Subroutine Files (*.isb)
Open
Cancel

2. Enter a subroutine name.

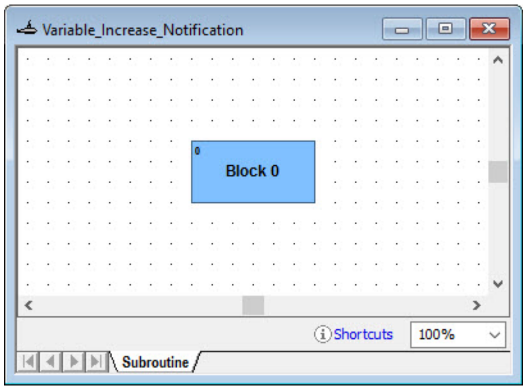
   The name must start with a letter, but may also include numbers and underscores. It cannot include spaces or other characters.

   The subroutine name will become a command (instruction) in PAC Control. It's a good idea to make it a descriptive name indicating the purpose of the subroutine, for example, "Variable_Increase_Notification." You cannot use the name of any existing command (for example, "Add").

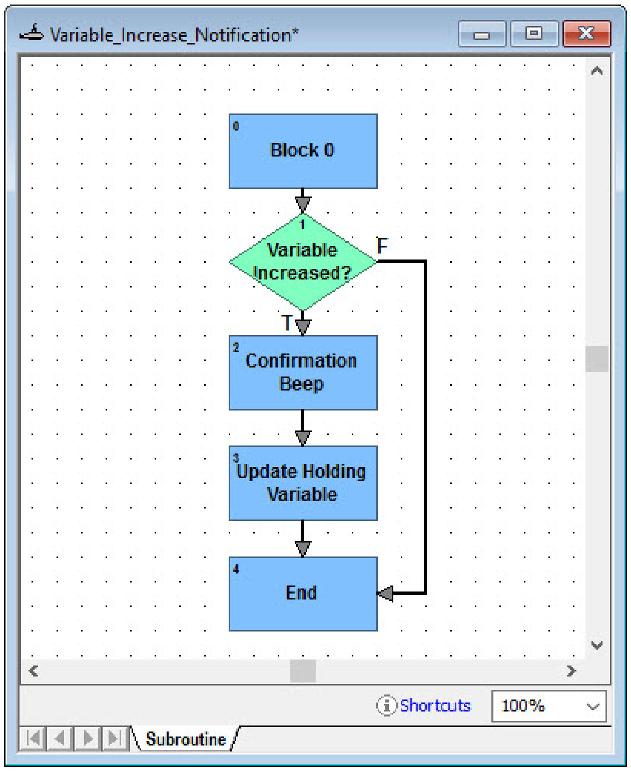
3. Navigate to the directory where you want to store the subroutine and click Open.

Unlike strategies, multiple subroutines can be saved to the same directory.

A new subroutine window is created.



4. Add blocks and connections and name the blocks as you would in a chart, as shown in the example below.



You can also copy existing flowchart blocks from another subroutine or chart and paste them into the new subroutine. See "Cutting, Copying, and Pasting Elements" on page 247.

5. Save the subroutine by selecting Subroutine > Save.

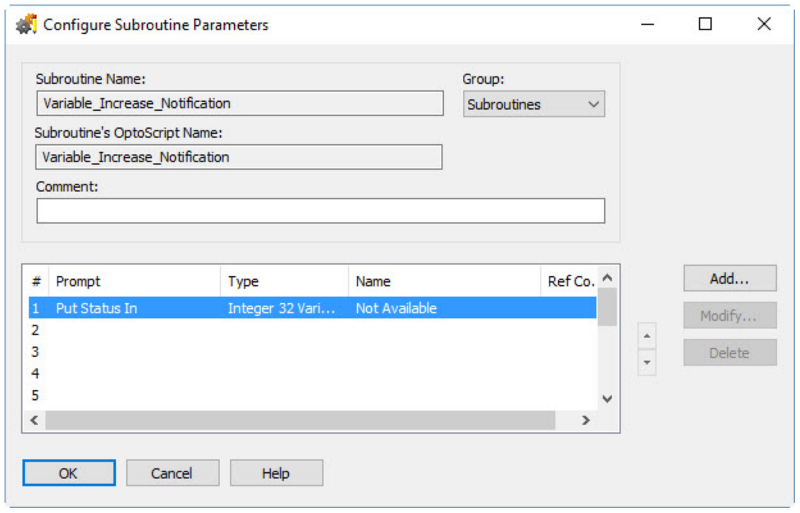
Configuring Subroutine Parameters

Before you can call a subroutine from a strategy, you must configure the variables and other logical items that are passed into it. These passed-in items, called subroutine parameters, are the only information that is shared between a subroutine and the calling strategy. Twelve parameters can be passed into a subroutine, and since a table can be a parameter, those 12 parameters can include a large amount of data.

An item passed into a subroutine may be called by one name in the strategy and by another name in the subroutine. In fact, if a subroutine is used for more than one strategy, it is good practice to select generic names in the subroutine. For example, if you create a subroutine to average values in any float table, the table might be named Float_Table in the subroutine. You could use this subroutine to average pressures in a table named Pressure_Values from a strategy, and Pressure_Values would be referred to as Float_Table in the subroutine.

1. With the subroutine open, select Subroutine > Configure Parameters.

   The Configure Subroutine Parameters dialog box appears.



Subroutine Name: Variable_Increase_Notification
Group: Subroutines
Subroutine's OptoScript Name: Variable_Increase_Notification
Comment:

#	Prompt	Type	Name	Ref Co.
1	Put Status In	Integer 32 Vari...	Not Available	
2				
3				
4				
5				

   In this dialog box you determine the way the subroutine is called from the strategy.

2. From the Group drop-down list, choose the command group you want the subroutine to appear in.

   For example, if you create a subroutine to find the average value of several variables, you could choose Mathematical as the command group. The default group is Subroutines.

3. (Optional) Enter a comment to explain the purpose of the subroutine.
4. Notice that one parameter, Put Status In, has been automatically entered for you.

   This parameter is used to return status information on the subroutine, and it always appears at the bottom of the parameter list. A subroutine essentially becomes a command within a chart or another subroutine, and subroutines are similar to function commands that return a status. Since the system itself returns this status parameter, the name of the status parameter is not available in the subroutine. When you add the subroutine to a strategy, you choose the variable the status will be placed in.

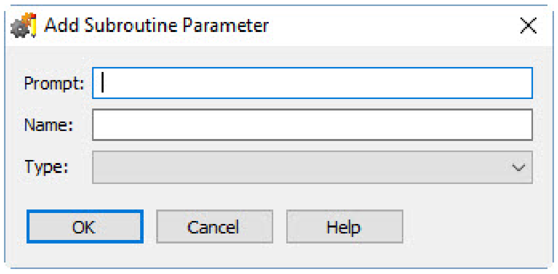
***IMPORTANT**: Make sure you always check returned status codes for all subroutines. Subroutine status codes and their meanings are:*

0	Success
-67	Out of memory
-69	Null object error. Make sure you are not passing a pointer that points to null.
-72	Nesting too deep
-73	Invalid subroutine or parameters. In configure mode, choose Edit > Find, select Global and Operand, and then click Find to search for objects with the same name as the subroutine. Rename any objects that you find.

5. For each parameter you add, do the following steps.

NOTE: What you enter here appears in the Add Instruction dialog box when the subroutine is called from within the strategy. See page 405 for an example.

a. Highlight the first empty line (below the Put Status In parameter) and click Add to open the Add Subroutine Parameter dialog box.

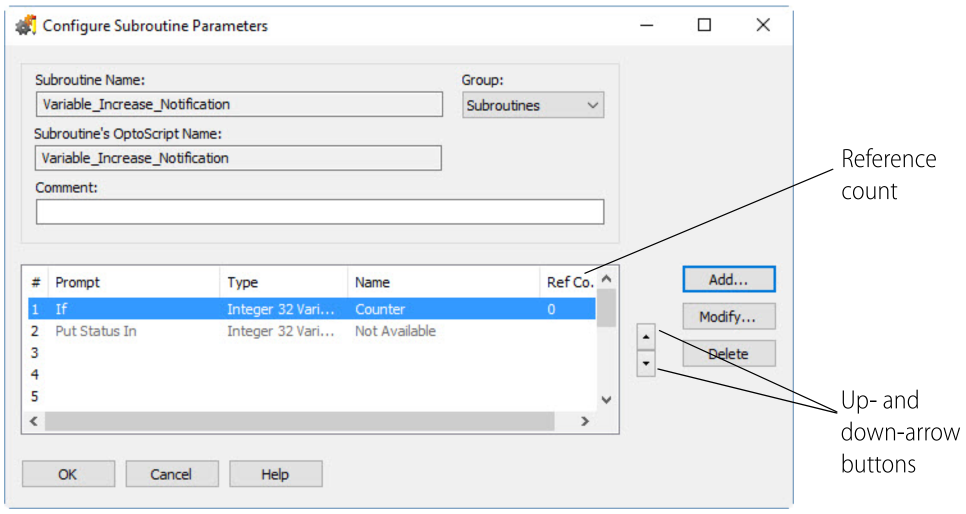


b. In the Prompt field, enter the prompt text you want to show in the Add Instruction dialog box in the strategy. This field is limited to 16 characters.

c. In the Name field, enter the name of the parameter (the argument) as it will be referred to in the subroutine. This name is used within the subroutine only.

d. From the Type drop-down list, choose the type of item to be passed into the subroutine. Use variables for values the subroutine changes (passed by reference); use literals for values the subroutine uses but does not change (read-only or passed by value).

e. Click OK. The parameter appears in the Configure Subroutine Parameters dialog box, above the Put Status In parameter. (Scroll up to see it if necessary.)



Subroutine Name: Variable_Increase_Notification
Group: Subroutines
Subroutine's OptoScript Name: Variable_Increase_Notification
Comment:

#	Prompt	Type	Name	Ref Co.
1	If	Integer 32 Vari...	Counter	0
2	Put Status In	Integer 32 Vari...	Not Available	
3				
4				
5				

6. Repeat step 5 for each parameter. To change a parameter, highlight it and click Modify. To change the order of a parameter in the list, highlight it and click the up- or down-arrow button in the dialog box. To delete a parameter, highlight it and click Delete.

NOTE: You cannot delete the Put Status In parameter or change its order in the list, and you cannot delete a parameter that has a reference count greater than zero (indicating that it is used in the subroutine). Also, if you add or delete parameters after including a subroutine in a strategy, you may receive an error and will need to add the subroutine to the strategy again.

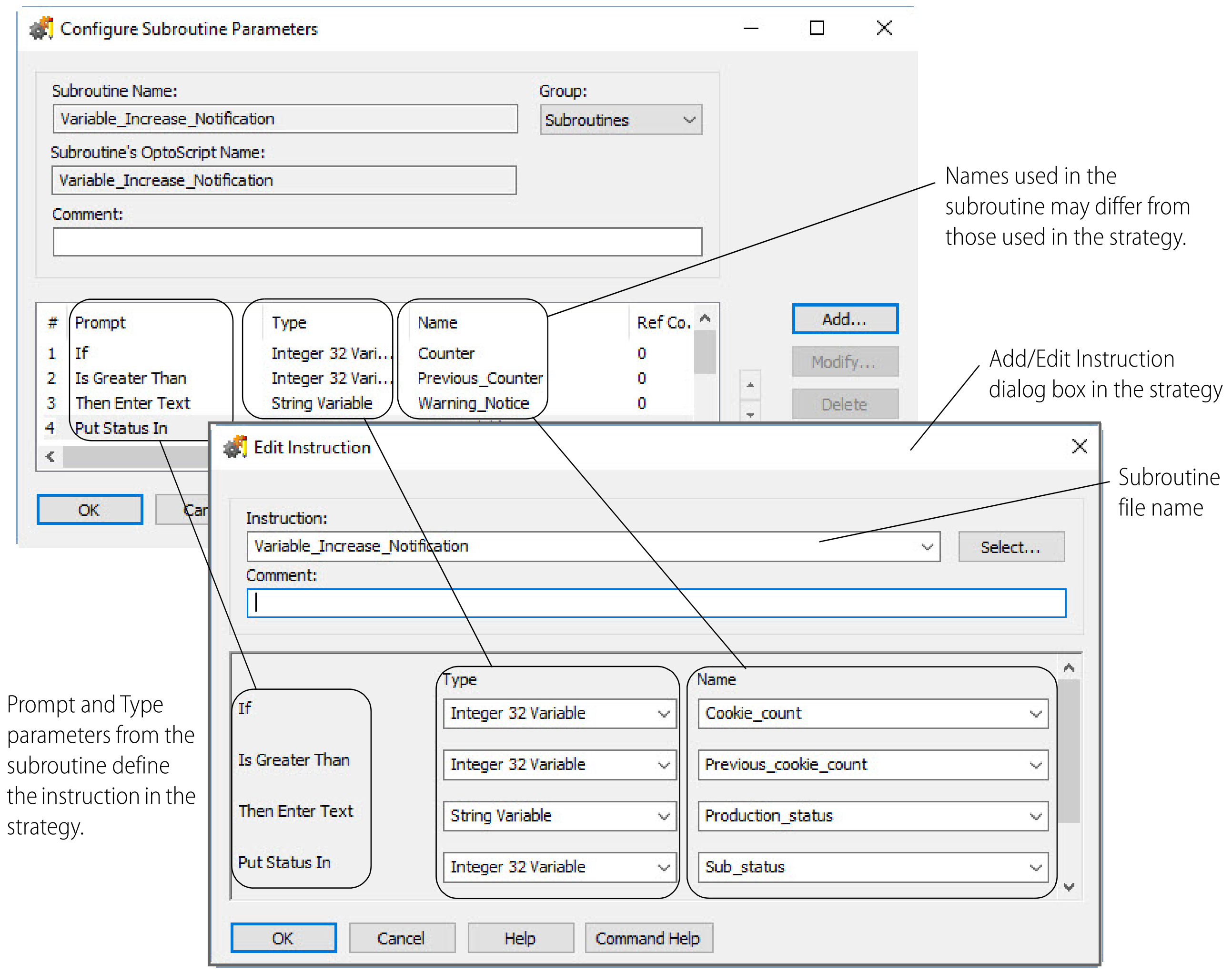
7. When the parameters appear the way you want them in the list, click OK.

The parameters you have named can now be used in the subroutine's commands.

Configured Parameters Example

Here's an example of a completed Configure Subroutine Parameters dialog box, showing three parameters to be passed into the subroutine.

When the subroutine is called from the strategy, these parameters appear in the Add Instruction dialog box:


Names used in the subroutine may differ from those used in the strategy.
Add/Edit Instruction dialog box in the strategy
Subroutine file name
Prompt and Type parameters from the subroutine define the instruction in the strategy.
Configure Subroutine Parameters
Edit Instruction
Add...
Modify...
Delete
OK
Cancel
Help
Command Help
Select...

Subroutine Name: Variable_Increase_Notification
Group: Subroutines
Subroutine's OptoScript Name: Variable_Increase_Notification
Comment:

#	Prompt	Type	Name	Ref Co.
1	If	Integer 32 Vari...	Counter	0
2	Is Greater Than	Integer 32 Vari...	Previous_Counter	0
3	Then Enter Text	String Variable	Warning_Notice	0
4	Put Status In			

Instruction: Variable_Increase_Notification
Comment:

	Type	Name
If	Integer 32 Variable	Cookie_count
Is Greater Than	Integer 32 Variable	Previous_cookie_count
Then Enter Text	String Variable	Production_status
Put Status In	Integer 32 Variable	Sub_status

Adding Commands and Local Variables

Adding commands (instructions) to subroutines is exactly like adding instructions to charts. For help, see "Adding Commands" on page 270. If you are using OptoScript code within a subroutine, see Chapter 11 for help on creating code. You can copy and paste instructions from one block to another within the same subroutine; if you have trouble copying and pasting instructions from one chart or subroutine into another, simply paste the entire block and then modify it.

You may also need to add local items to be used in the subroutine only and discarded when the subroutine is finished running. Adding variables to subroutines is also like adding variables to charts. For help, see "Adding Variables" on page 257.

Remember that the subroutine is separate from any strategy and can be called by any strategy. I/O units and points are specific to a strategy, so they cannot be added to a subroutine. Also, note that commands and OptoScript code within a subroutine can use only the passed-in and local items in the subroutine, not items in the strategy that calls the subroutine.

Compiling and Saving the Subroutine

1. With the subroutine open, select Compile > Compile Subroutine.

   When the subroutine has finished compiling, the cursor returns to its normal form.

2. Select Subroutine > Save.

Using Subroutines

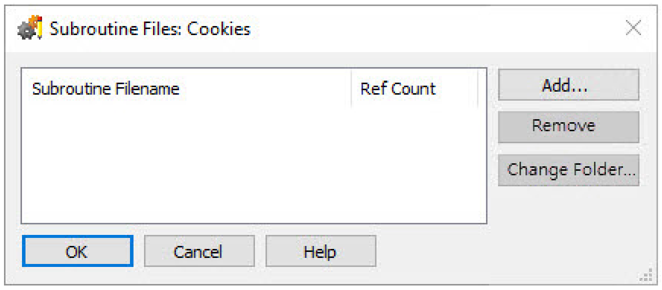
To use a subroutine in a strategy, first add the subroutine to the strategy. Then, you can use it just like a command in the logic within the strategy's flowcharts, or within another subroutine that has also been added to the strategy.

Adding a Subroutine to a Strategy

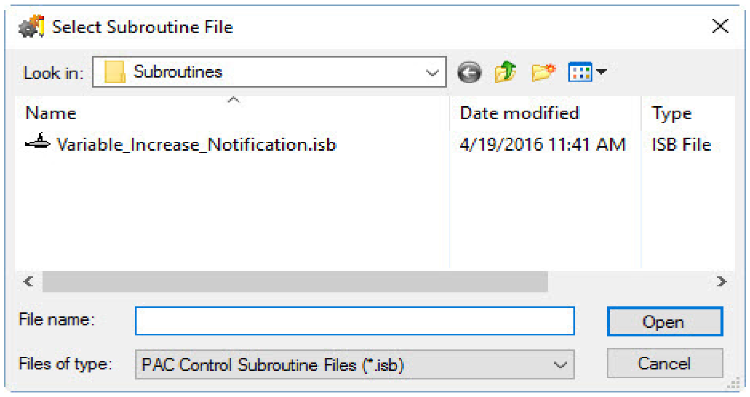
Since subroutines are independent of strategies, you must add the subroutine to the strategy before you can use it.

1. With the strategy open in Configure mode, double-click the Subroutines Included folder on the Strategy Tree (or in the menu bar, click Configure > Subroutine Included).

   The Subroutine Files dialog box appears, listing all subroutines currently included in the strategy. The example below shows no subroutines currently included.



2. Click Add.
3. In the Select Subroutine File dialog box, navigate to the folder containing the subroutine you want to add.



Name	Date modified	Type
Variable_Increase_Notification.isb	4/19/2016 11:41 AM	ISB File

Files of type: PAC Control Subroutine Files (*.isb)

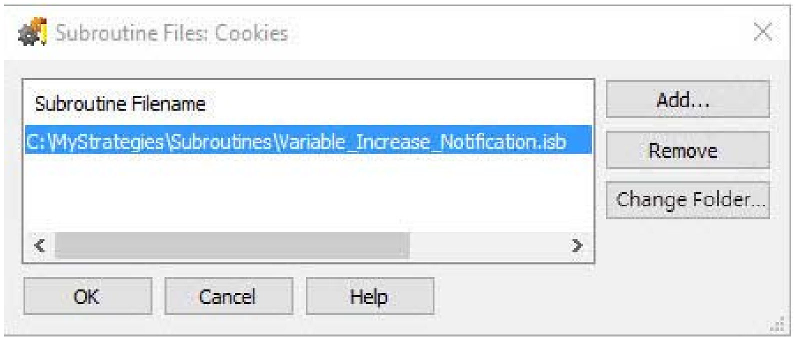
4. Click a subroutine to select it, and then click Open.
5. The new subroutine's name appears in the Subroutine Files dialog box. Click OK to save the change and close the dialog box.

   The new subroutine appears in the Strategy Tree in the Subroutines Included folder.

Removing a Subroutine

If the Remove button is grayed out when you select a subroutine in the Subroutine File dialog box, it means the subroutine is used in the strategy. Before you can remove the subroutine, you must first delete any references to it in the strategy.

1. Open the Subroutine Files dialog box, and then click the subroutine to remove.



C:\MyStrategies\Subroutines\Variable_Increase_Notification.isb

2. Click Remove, and then click OK to save your changes and close the dialog box.

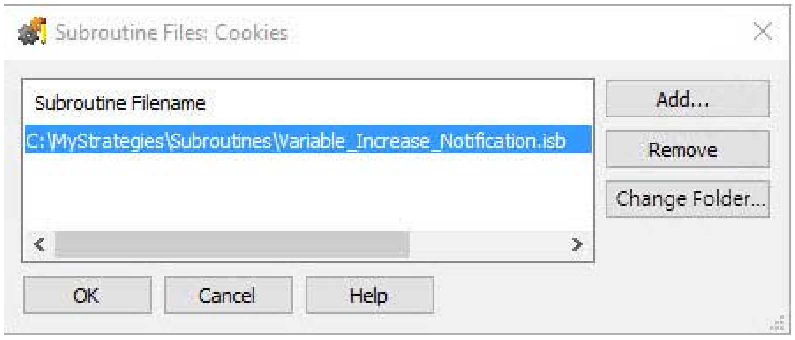
Changing a Subroutine's Folder

The Change Folder button helps when:

- A subroutine is saved in a read-only folder (which can cause errors when you attempt to compile the strategy)
- You simply want to change the location of a subroutine that is being used by a strategy.

To change a subroutine's folder:

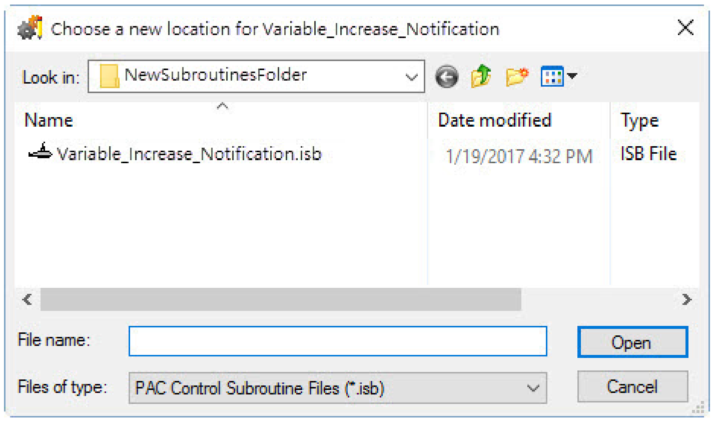
1. Open the Subroutine Files dialog box. The subroutine's original folder and filename appear.



C:\MyStrategies\Subroutines\Variable_Increase_Notification.isb

2. Click the subroutine to select it, and then click Change Folder.

3. In the "Choose a new location for" dialog box, navigate to the folder containing the subroutine.


Choose a new location for Variable_Increase_Notification
Look in:
NewSubroutinesFolder
Name
Date modified
Type
Variable_Increase_Notification.isb
1/19/2017 4:32 PM
ISB File
File name:
Files of type:
PAC Control Subroutine Files (*.isb)
Open
Cancel

4. Click the subroutine to select it, and then click Open.
5. The subroutine's name appears in the Subroutine Files dialog box. Click OK to save the change and close the dialog box.

Using a Subroutine in Strategy Logic

You use a subroutine just like a PAC Control command: by adding the subroutine as a command to a block in a chart or subroutine.

NOTE: Subroutines are located in the Subroutine group by default. If you want subroutines in a different group, you must change the group when you configure the subroutine's parameters. See "Configuring Subroutine Parameters" on page 403.

1. With the strategy open in Configure mode, open the chart or subroutine that will call the subroutine.
2. Double-click the block that will call the subroutine.

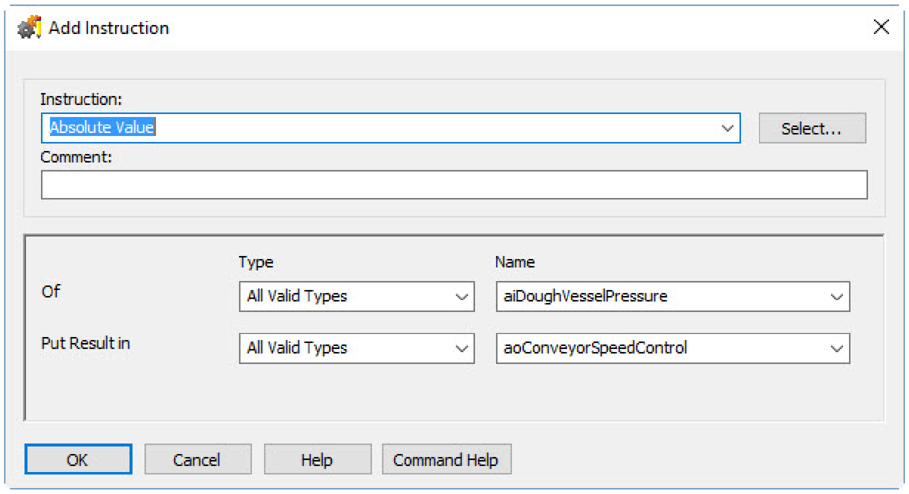
   If it is an OptoScript block, list parameters (arguments) in order within parentheses, the same way you enter parameters for PAC Control commands (see "OptoScript Syntax" on page 379). Be sure to use the subroutine's OptoScript name, using underscores instead of spaces. The Put Status In return value can be consumed by a variable (as shown below), a mathematical expression, or a control structure.

```
nStatus = Variable_Increase_Notification( bCondition, nValue, Output1 );
```

   For more information on entering a command (instruction) in OptoScript, see "Using the OptoScript Editor" on page 392.

3. In the Instructions dialog box, click where you want the instruction to be placed, and then click Add.

The Add Instruction dialog box appears.



Add Instruction

Instruction: Absolute Value

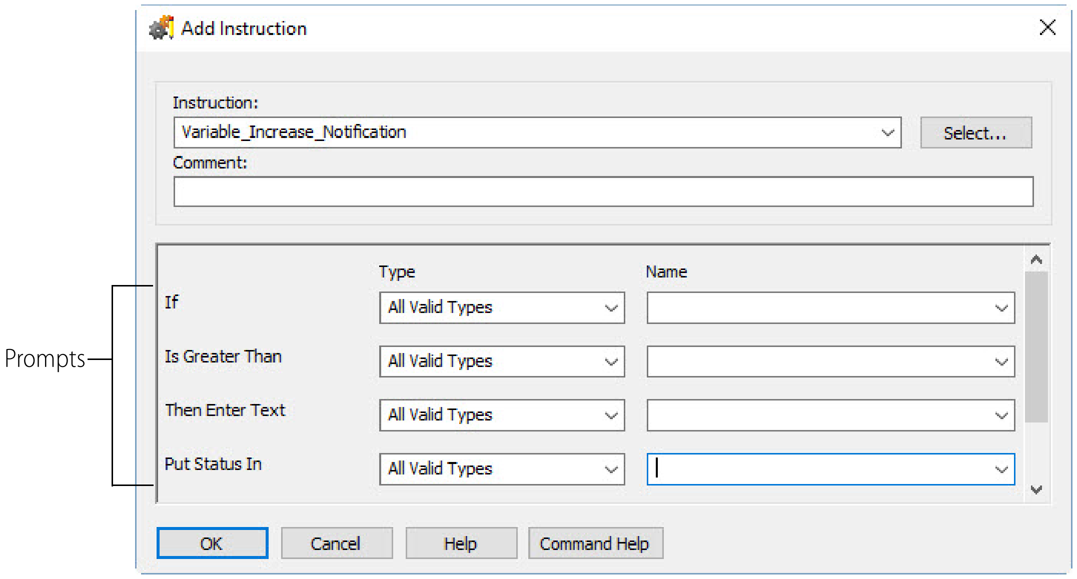
Comment:

	Type	Name
Of	All Valid Types	aiDoughVesselPressure
Put Result in	All Valid Types	aoConveyorSpeedControl

4. In the highlighted Instruction field, enter the subroutine name (or choose it using the drop-down list, or click the Select button and locate it in the command group you chose).

NOTE: If you click Select, you will find subroutines located in the Subroutine group by default. If you want subroutines in a different group, you can change the group when you configure the Subroutine's parameters. See "Configuring Subroutine Parameters" on page 403.

The subroutine command appears in the dialog box, just as any command would, and the prompts you entered when you configured the parameters appear also.



Add Instruction

Instruction: Variable_Increase_Notification

Comment:

	Type	Name
If	All Valid Types	
Is Greater Than	All Valid Types	
Then Enter Text	All Valid Types	
Put Status In	All Valid Types	

5. Choose the Type and Name for each prompt from the drop-down lists.

You can configure variables on the fly as you would with any command. Remember that the Type was chosen when the parameters for the command were configured, so your Type choices may be limited.

6. When the Add Instruction dialog box is completed, click OK.

7. Click Close to close the Instructions dialog box and return to the chart.

   The chart or subroutine is now set up to call the subroutine.

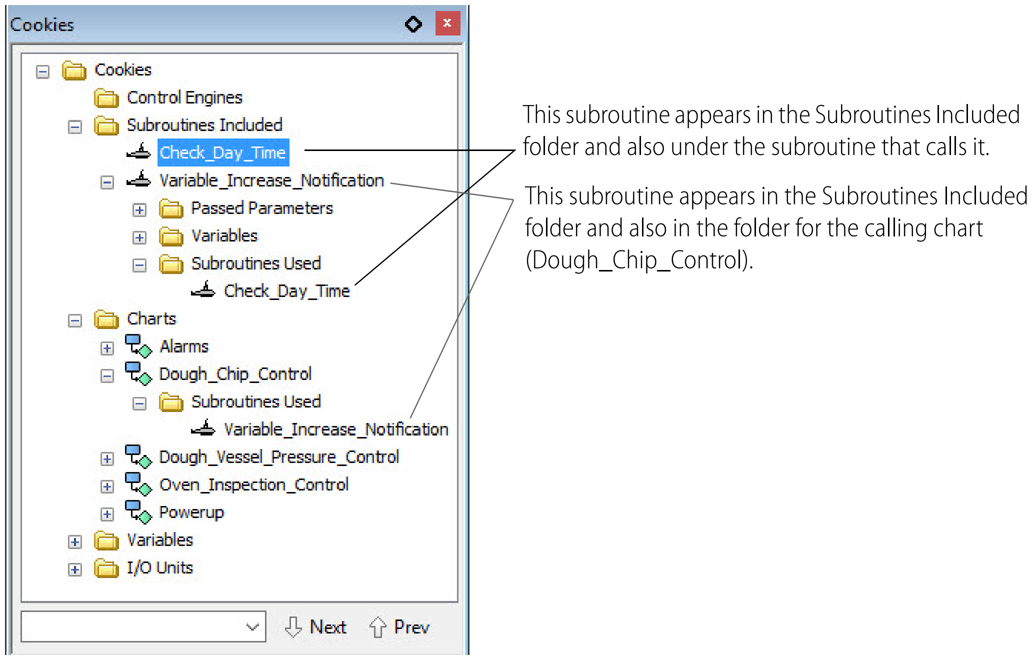
Debugging Subroutines

Debugging a subroutine is just like debugging a flowchart. When you are debugging a strategy that calls the subroutine, make sure the debug level is Full Debug (see page 211). Then use the Step Into button to step inside the block that calls the subroutine. The subroutine window automatically opens, and you can continue to step through blocks or lines inside the subroutine. As you step through blocks, subroutines, and charts, the tabs at the bottom of the window let you know where you are (see "Using Tabs to View Open Windows" on page 59).

You can also set breakpoints on any subroutine block as needed. See "Setting and Removing Breakpoints" on page 215 for more information.

Viewing Subroutines

Since subroutines appear on the Strategy Tree, it is easy to view information about them. A subroutine appears in two places on the Strategy Tree: in the Subroutines Included folder and in the folder for the chart or subroutine that calls it.


Cookies
Cookies
Control Engines
Subroutines Included
Check_Day_Time
Variable_Increase_Notification
Passed Parameters
Variables
Subroutines Used
Check_Day_Time
Charts
Alarms
Dough_Chip_Control
Subroutines Used
Variable_Increase_Notification
Dough_Vessel_Pressure_Control
Oven_Inspection_Control
Powerup
Variables
I/O Units
Next
Prev
This subroutine appears in the Subroutines Included folder and also under the subroutine that calls it.
This subroutine appears in the Subroutines Included folder and also in the folder for the calling chart (Dough_Chip_Control).

You can view, add, and change variables in a subroutine from the Strategy Tree, just as you would for a chart.

Viewing All Subroutines in a Strategy

To see all the subroutines in a strategy, double-click the Subroutines Included folder on the Strategy Tree. All subroutines associated with the strategy are listed in the Subroutine Files dialog box. Click and drag the right side of the box to see all the information. The path, file name, and reference count (how many times the strategy refers to the subroutine) are shown for each subroutine.

Printing Subroutines

For steps to print a subroutine's graphics, see "Printing Chart or Subroutine Graphics" on page 221. To view and print instructions in the subroutine's blocks, see "Viewing and Printing Strategy or Subroutine Elements" on page 224.

A: Troubleshooting

OPTO 22

This appendix provides general tips for resolving problems you may encounter while running PAC Control or communicating with your hardware. If you encounter problems with permissions in Windows, see page 421.

For information about types of errors and lists of error codes, see "B: Errors and Messages."

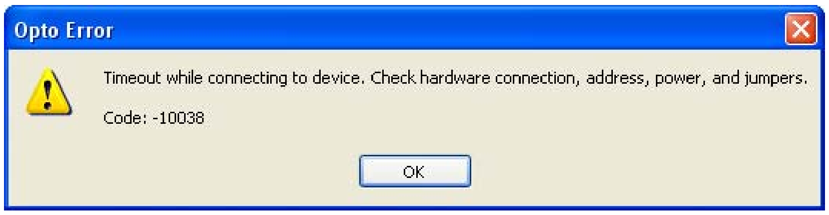
Also check troubleshooting information in your controller's user guide. See also form 1690, the *SNAP PAC Brain User's Guide*, and the troubleshooting appendix in form 1460, the *SNAP Ethernet-Based I/O Units User's Guide*.

How to Begin Troubleshooting

You've built your strategy, but now you get errors when you try to download it, or it won't run properly. How do you begin to figure out what's wrong? The problem may be in communication with the control engine, in communication between the control engine and I/O, in a command, or in the strategy logic. Following are some steps to help you discover the cause.

1. Read Any Error Message Box

If an error message box appears on the computer running PAC Control, it's probably a PAC Control error. Here's an example of a PAC Control error message:



Timeout while connecting to device. Check hardware connection, address, power, and jumpers.

Code: -10038

2. Check Communication with the Control Engine

If there is no error message box, or the error indicates that there may be a communication problem, check whether the PC running PAC Control is communicating with the control engine. See "Checking Communication with the Control Engine" on page 416.

3. Check the Message Queue

If communication with the control engine is OK, check the message queue. To open the queue, see "Inspecting Control Engines and the Queue" on page 111. In the "List of Common Messages" on page 425, look up any errors you find in the queue. Errors are listed in numerical order. Queue errors may indicate problems with a command or with communication to I/O. Check the possible causes for help in fixing problems.

- For help with a command, look up details about the command in the *PAC Control Command Reference* or online help.
- For help with communication to I/O, see "Resolving Communication Problems" on page 417. Many of these suggestions apply to I/O as well as to control engines.

4. Check Status Codes in Your Strategy

If all is well up to this point, double-check Status Codes in your strategy. Status Codes are responses to a command that appear in a variable within your PAC Control strategy or as returned values in OptoScript. Your strategy should routinely check status codes and adjust logic as necessary to respond. It's especially important to check status codes when you use a command that starts a chart or subroutine, to make sure it started successfully.

Status codes may indicate problems with a command or communication to I/O, or they may indicate a problem in the strategy logic. See "List of Common Messages" on page 425 for more information. Again, look at the possible causes for help in fixing problems.

5. Call Product Support

If you cannot find the help you need in this book, the *PAC Control Command Reference*, or the *SNAP PAC Brains User's Guide*, call Opto 22 Product Support. See "Product Support" on page 4 for contact information.

Strategy Problems

If You Cannot Delete an Item

Sometimes when you try to delete an item in a strategy—a variable, a chart, an I/O unit or point—you receive a message saying "You cannot delete an item with a reference count greater than zero." This message means you cannot delete the item because other elements in the strategy use it.

You can use Find to locate all references to the item you want to delete. For help in using Find, see "Searching" on page 229.

Sometimes the reference counts can become incorrect due to cutting and pasting variables or importing charts into a strategy. If a reference count appears to be incorrect, you can rebuild the strategy database by following these steps:

1. With the strategy open in Config mode, choose Tools > Regenerate Reference Counts.
2. Choose Compile > Compile All.

3. Choose File > Save All.

The strategy database is rebuilt and the reference counts should be correct. If they are not, you may need to choose Tools > Regenerate Reference Counts more than once.

If You Have Memory Problems

Control engine memory is allocated as shown in detail in form 1646, the *SNAP PAC Memory Usage Technical Note*.

On your controller, you can see the amount of RAM and the amount of battery-backed RAM available for use by inspecting the control engine in Debug mode (see page 111). Battery-backed RAM stores persistent variables, variables initialized on download, autorun flag, and optionally, an archive of the strategy. Note that strategies are not saved in battery-backed RAM. Save your strategy to flash memory so it will be available if the control engine is rebooted. See "Saving a Strategy to Flash" on page 200.

Check Strings and Tables

In general, if you experience memory problems, you can reduce the amount of memory needed by checking strings and tables for lengths and widths that are longer than necessary.

Since the battery-backed RAM contains persistent variables and variables initialized on download and optionally stores a strategy archive, you may need to adjust the length of tables or the width of strings and string tables.

The currently running strategy is not stored in battery-backed RAM. To make sure the strategy will run after a control engine reboot, save the strategy to flash memory after downloading it. See page 200 for information on saving to flash.

Subroutines and Memory

When you use subroutines, use the minimum number of variables and the minimum size of tables that the process requires. Here are some additional things to keep in mind:

- Less memory is used if only one chart in the strategy calls subroutines than if multiple charts call subroutines.
- Using nested subroutines (that is, subroutines that call other subroutines) uses more memory. (To use nested subroutines, you need SNAP PAC firmware R9.5a or higher.)

Make sure your strategy logic checks the value of the "Put status in" variable when starting a chart or calling a subroutine to make sure they start correctly. When a chart or subroutine starts, if there is not enough memory available to process it and all the subroutines called within its logic, you will receive an error in the "Put status in" variable for that command.

Archiving Strategies to Battery-Backed RAM

Strategies can optionally be stored in battery-backed RAM on the control engine. In addition to an archived strategy, battery-backed RAM holds persistent variables and variables that are initiated on download. If you have a large strategy, large tables, or wide strings or string tables, you may not have sufficient space for an archived strategy. Strategies can optionally be stored in battery-backed RAM on the control engine. See page 201 for more information on archiving.

Do You Use Online Mode?

If you use Online mode to change your strategy, you may run out of memory. When you change a chart in Online mode, a new copy of that chart is downloaded to the control engine, causing additional memory to be used.

To avoid memory issues, it may be necessary to periodically do a full download of the strategy.

Checking Communication with the Control Engine

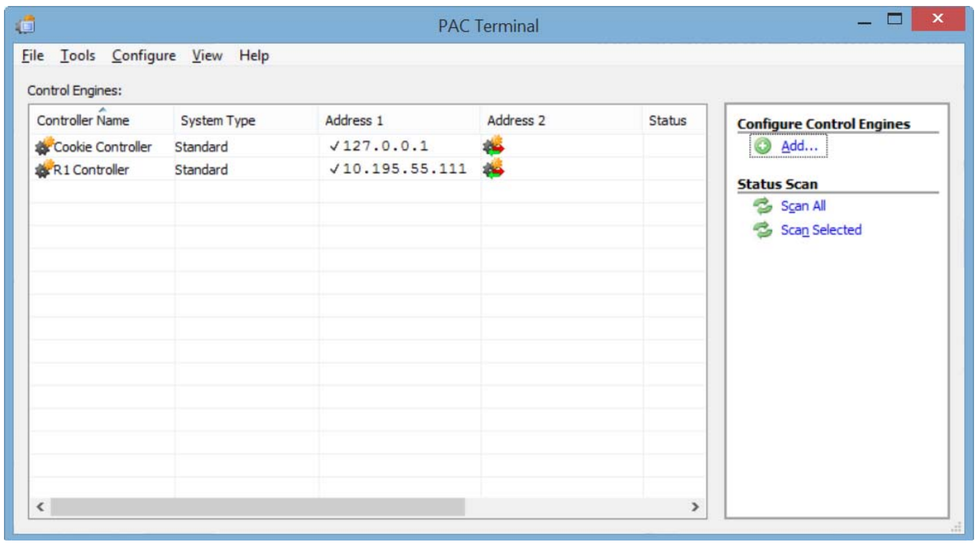
You can test communication with the control engine by using PAC Terminal.

1. Start PAC Terminal as follows:

– In Windows 7 and Windows Vista, press the Windows Start key, and then click Programs > Opto 22 > PAC Project 9.6 > Tools > PAC Terminal.

– In Windows 10 and Windows 8.1, press the Windows Start key, type **`PAC Terminal 9.6`** and then press the Enter key.

The PAC Terminal window appears, showing all control engines configured on your system:



Controller Name	System Type	Address 1	Address 2	Status
Cookie Controller	Standard	✓127.0.0.1		
R1 Controller	Standard	✓10.195.55.111		

2. If no control engine is listed, configure one by choosing Configure > Control Engine and following directions on the screen. See "Configuring Control Engines" on page 99 for help.

3. To verify that a control engine in the list is communicating, double-click the control engine's name.

The Comm. Loop Time (communication time) in the Inspect Control Engine dialog box indicates how long it takes to gather the information in the dialog box and is a good relative indicator of communication time.

This dialog box also shows the status of the current strategy and any errors in communication with the control engine. For further explanation, see "Inspecting Control Engines and the Queue" on page 111.

4. If you receive an error indicating a communication problem, go on to the next section.

Resolving Communication Problems

Matching PAC Control Configuration to the Real World

I/O unit and point configuration in PAC Control must match actual I/O units and points with which the control engine is communicating. See brain and I/O module data sheets for specifications and information, and see "Adding an I/O Unit" on page 130 for help configuring I/O in PAC Control.

Resolving TCP/IP Cannot Connect Errors (-412)

Many problems with Ethernet connections return a TCP/IP Cannot Connect error. Cannot connect errors are probably the most common communication problem with control engines. They indicate that a TCP/IP connection could not be made to the control engine within the specified time interval.

If you receive this error, first check the following:

- Make sure the control engine has been turned on.
- Verify that the correct IP address appears for the control engine.
- Make sure your control engine has been assigned a valid IP address. These controllers and brains come from the factory with a default IP address of 0.0.0.0, which is invalid. For help in assigning an IP address, see Opto 22 form 1704, the *PAC Manager User's Guide*.
- Make sure you have up-to-date drivers installed on your computer's Network Interface Card (NIC). Contact your system administrator or the manufacturer of the card for help.
- If problems persist, you can increase the length of time before a timeout occurs. Choose Configure > Control Engines and change the Timeout (mSec) field to a larger number.

Pinging the Control Engine

If you still cannot communicate with the control engine after you have checked these items, try to reach it using the Windows PING command.

1. To ping a control engine, open a Windows Command Prompt:
   - For Windows 7, Windows Vista, and Windows XP, click the Windows Start menu, and then and then click Programs > Command Prompt.
   - For Windows 10 or Windows 8.1, press the Windows Start key, type **`cmd`** and then press the Enter key.
2. At the prompt, type **`ping`** and the control engine's IP address, and then press Enter.

   Example: `ping 10.192.54.40`

If data is returned from the control engine, the control engine can be found on the network.

If the PING command cannot be found—Verify that the PC has TCP/IP bound to and configured on the network adapter in Windows' Network and Sharing Center. To open Windows Network and Sharing Center:

1. Open Windows Control Panel.
   - If Control Panel is set to **View by: Category**, click the **View by** drop-down list, and then click Large icons.
   - In the Large icons view, click the Network and Sharing Center icon.

2. On the left side of the Network and Sharing Center window, click Change adapter settings.
3. In the Network Connections window, right-click your network connection, and then click Properties on the pop-up menu.
4. On the network properties' Networking tab, make sure Internet Protocol Version 4 (TCP/IPv4) is present and checked.
5. Click Internet Protocol Version 4 (TCP/IPv4) to highlight it, and then click the Properties button. If the IP address is not automatically obtained, verify that the IP address, subnet mask, and default gateway are appropriate for your network.

If you see the message "Destination host route not defined," the control engine probably has an IP address and subnet mask that are incompatible with those on the computer. Subnetwork numbers and netmasks must be identical on the control engine and the computer.

If you see the message "No response from host," check the following:

- Are the computer and control engine correctly connected? Is the control engine turned on?
- Are the IP address and subnet mask on the control engine compatible with those on the computer?

If your host computer has more than one Ethernet card, check your route table to make sure packets are routed to the correct adapter card.

If all else fails, connect the PC and the control engine using an Ethernet crossover cable, and retest the connection.

If you still cannot ping the control engine, contact Product Support. (See page 4.)

Other Troubleshooting Tools

Checking Detailed Communication Information Using PAC Message Viewer

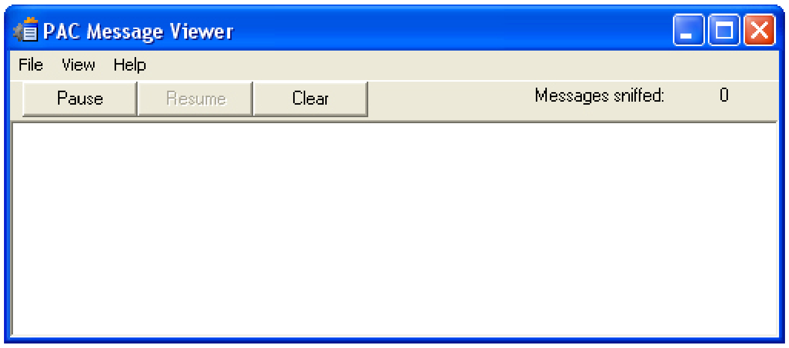
For detailed information about each communication transaction, use the PAC Message Viewer tool. To open PAC Message Viewer:

- In Windows 7 and Windows Vista, press the Windows Start key, and then click Programs > Opto 22 > PAC Project 9.6 > Tools > PAC Message Viewer.
- In Windows 10 and Windows 8.1, press the Windows Start key, type **`PAC Message Viewer 9.6`** and then press the Enter key.

You can also start PAC Message Viewer from:

- PAC Terminal's menu bar, by clicking Tools > Start PAC Message Viewer
- PAC Control in Debug mode by clicking Debug > Sniff Communication.

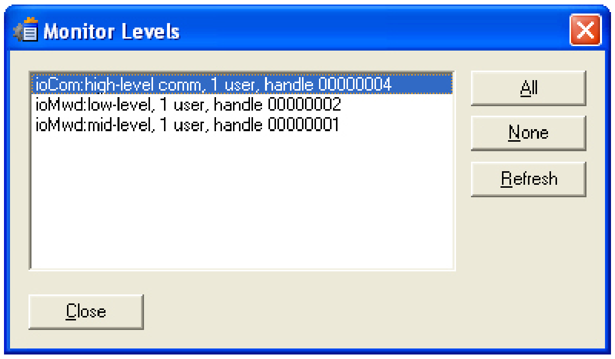
The PAC Message Viewer window appears.



In most cases the main window is blank, indicating that no messages are being monitored between the PC and the active control engine. In some cases, for example when PAC Control launches PAC Message Viewer, messages should appear immediately.

1. To start monitoring or change the level of monitoring, choose View > Monitor Levels.

   The Monitor Levels dialog box lists all the possible levels to monitor. You can click Refresh to make sure the list is up to date.



ioCom:high-level comm, 1 user, handle 00000004
ioMwd:low-level, 1 user, handle 00000002
ioMwd:mid-level, 1 user, handle 00000001

2. Highlight one or more of the monitor levels in the list, and click Close.

   You return to the PAC Message Viewer window, where the changes you made are reflected at once. To stop monitoring, click Pause. To start monitoring again, click Resume. To erase all messages from the window, click Clear.

   By default, communication messages in PAC Message Viewer are automatically saved to a log file named IOSNIF.LOG. You can toggle saving on and off by choosing File > Log to File.

   Also by default, messages are temporarily stored in system cache memory before being saved to the log file. If you are having trouble with system crashes and need to capture messages just before a crash, however, you can choose File > Flush File Always to send messages directly to the log file.

3. To view or edit the log file, choose File > Edit Log File.

   The file opens in a text editor. Logging is turned off when you open the file.

4. View or edit the file as needed, and then close it.

   You return to the PAC Message Viewer window.

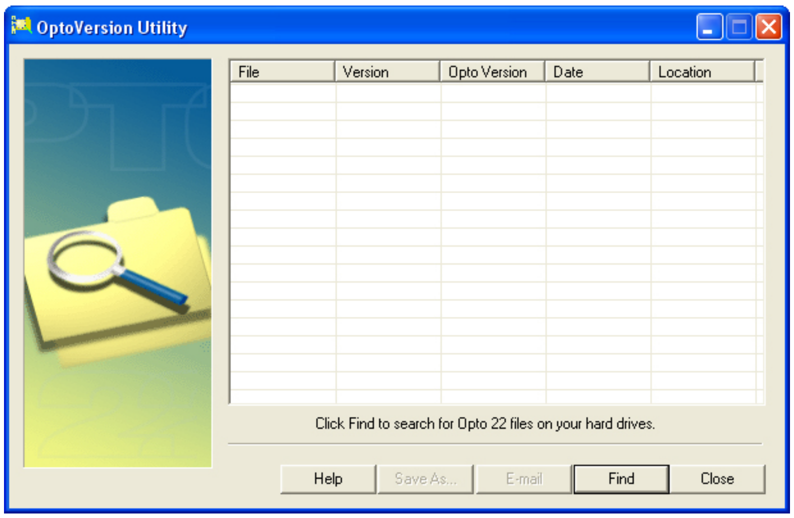
5. To resume logging, choose File > Log to File.
6. To rename the log file or change its location, choose File > Select Log File. Navigate to the location where you want to save the file and enter a name. Click OK.
7. When you have finished monitoring communication, close the PAC Message Viewer window.

   If you leave it open, it will normally appear on top of other running programs. If you don't want it to appear on top of other programs, choose View > Always on Top to toggle that option.

Checking File Versions for Opto 22 Software

Sometimes problems may be caused by older or misplaced files. Product Support may ask you to run OptoVersion to check the versions and paths of your Opto 22 .dll and .exe files. Here's how:

1. Start OptoVersion as follows:
   - In Windows 7 and Windows Vista, press the Windows Start key, and then click Programs > Opto 22 > PAC Project 9.6 > Tools > OptoVersion.
   - In Windows 10 and Windows 8.1, press the Windows Start key, type **`OptoVersion 9.6`** and then press the Enter key.



Click Find to search for Opto 22 files on your hard drives.

2. In the OptoVersion window, click Find.

   The utility searches your hard drive and prints a list of Opto-related files found.
3. To see more information on any file, double-click its name. To sort the list in a different order, click any column heading.
4. To email the information to Opto 22 Product Support, click E-mail.

   The utility saves the list to a file named Version.bd in the same directory that contains OptoVersion.exe. If you use Microsoft Outlook as your email program, a new message automatically appears addressed to Product Support, with the version file attached.
5. If you use Microsoft Outlook, add comments to the new message and click Send.
6. If you use another email program, attach the Version.bd file to an email message and address the message to support@opto22.com, along with an explanation of the problem you're experiencing.

7. To save the file, click Save As. Give the file a name and save it in the location you want.

   OptoVersion also creates a tab-delimited file with the same file extension and in the same directory. This file has the same name you gave it but with `_Delimited` added. For example, if you name the saved file `Opto_software.txt`, the tab-delimited file is named `Opto_software_Delimited.txt`. This file can be opened in Microsoft Excel or other programs to easily sort and view its contents.

Problems with Permissions in Windows

When you set up controllers on a computer running the Microsoft Windows operating system, typically you are using the computer with top-level "administrator" privileges. If someone later uses this same computer to run PAC Control or PAC Display, but logs in to the computer with lower-level, non-administrator privileges, the application may not recognize control engines that have been previously configured.

If this problem occurs, you can modify the permissions to let specific users access previously configured control engines without having administrator access. This is done using the Registry Editor utility. Follow the steps below.

***CAUTION:** Use the Windows Registry Editor carefully. It is strongly recommended that you make a backup copy of your Windows Registry before continuing with this procedure. Without a backup copy, if you delete the wrong properties and cannot return the Registry to its original state, application and system files can become unusable and will have to be reinstalled.*

1. Log on as an administrator on the computer where PAC Control or PAC Display is installed.
2. On your keyboard, press the Windows Start key, type **`run`** and then press Enter. The Run dialog box appears.
3. In the Open field, type **`regedt32`** and then press Enter:

   NOTE: This is NOT regedit.exe, which is a similar tool.

   If a User Account Control window appears, click Yes to continue.

   The Registry Editor window appears.
4. In the left pane, double-click the HKEY_LOCAL_MACHINE folder to expand it.
5. In the HKEY_LOCAL_MACHINE folder, double-click the SOFTWARE folder to expand it.
6. Navigate to the Opto 22 folder by following the instructions for your system type.
   – For 32-bit operating systems, in the SOFTWARE folder, click the Opto22 folder to highlight it.
   – For 64-bit operating systems:
     – In the SOFTWARE folder, double-click the WOW6432Node folder to expand it.
     – In the WOW6432Node folder, click the Opto22 folder to highlight it.
7. With the Opto22 folder highlighted, click Edit > Permissions on the Registry Editor menu bar.

   The Permissions for Opto22 dialog box opens.
8. Click Add.
9. In the Select Users, Computers, or Groups dialog box, type the name of the user or group that should have control engine access, and then click OK.

   The Permissions for Opto22 dialog box reappears.

10. If it is not already selected, check the "Full Control—Allow" box in the Permissions area.
11. Click OK to save your changes and close the dialog box.
12. Click File > Exit to close the Registry Editor.

The added users will need to restart their computers for the new permissions to take effect. Then they will be able to use control engines without having administrator access.

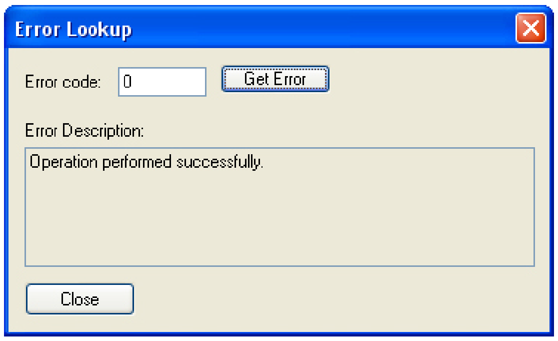
B: Errors and Messages

OPTO 22

Introduction

This appendix discusses errors and messages you may see in PAC Control and their possible causes. Errors and messages may appear with text only or with a negative number and text. The more common errors and messages are listed in this chapter in numeric order, starting on page 425.

To look up an error in PAC Control, select Help > Error Lookup. In the Error Lookup dialog box, enter the error code, and then click Get Error.



Error code: 0

Error Description:

Operation performed successfully.

See also, "A: Troubleshooting" on page 413 for additional help in resolving errors.

Types of Errors

As you work in PAC Control, you may see three types of errors:

- PAC Control Errors appear in dialog boxes on the computer screen.
- Queue Messages (both errors and other messages) appear in the control engine's message queue.
- Status Codes appear in variables or as returned values in OptoScript.

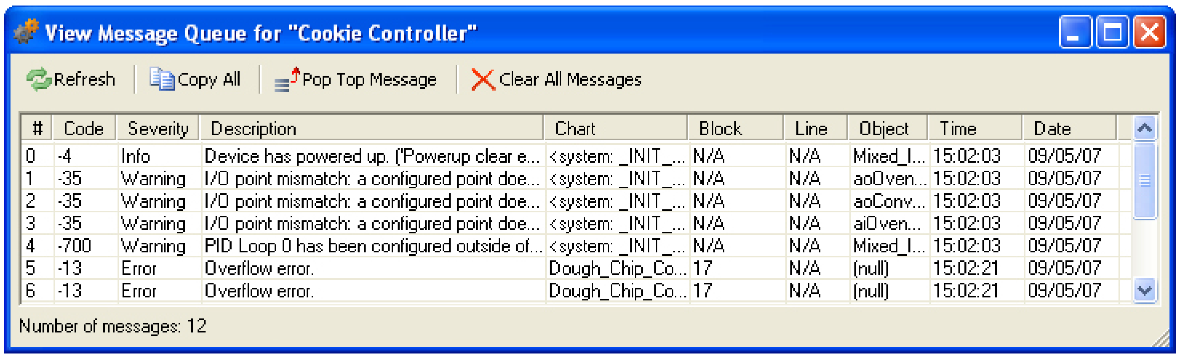
PAC Control Errors

PAC Control errors indicate a problem within PAC Control that may have been reported by the control engine or may have occurred before control engine communication.

PAC Control errors appear in dialog boxes on the computer running PAC Control. Some of these errors appear as numbers, some as text, and some show both numbers and text. An example of a PAC Control error is "Timeout. No response from device." Another example is "TCP/IP: Cannot connect error" with an error code of -412.

Queue Messages

Queue messages indicate an error or other message during strategy operation, and they appear in the PAC Control message queue. (For information on viewing the queue, see "Viewing the Message Queue" on page 114.) Here's an example of a message queue:



#	Code	Severity	Description	Chart	Block	Line	Object	Time	Date
0	-4	Info	Device has powered up. ('Powerup clear e...	<system: _INIT_...	N/A	N/A	Mixed_I...	15:02:03	09/05/07
1	-35	Warning	I/O point mismatch: a configured point doe...	<system: _INIT_...	N/A	N/A	aoOven...	15:02:03	09/05/07
2	-35	Warning	I/O point mismatch: a configured point doe...	<system: _INIT_...	N/A	N/A	aoConv...	15:02:03	09/05/07
3	-35	Warning	I/O point mismatch: a configured point doe...	<system: _INIT_...	N/A	N/A	aiOven...	15:02:03	09/05/07
4	-700	Warning	PID Loop 0 has been configured outside of...	<system: _INIT_...	N/A	N/A	Mixed_I...	15:02:03	09/05/07
5	-13	Error	Overflow error.	Dough_Chip_Co...	17	N/A	(null)	15:02:21	09/05/07
6	-13	Error	Overflow error.	Dough_Chip_Co...	17	N/A	(null)	15:02:21	09/05/07

Number of messages: 12

This queue shows several types of messages that you might see. To see all the information in a column, drag the edge of the column heading to the right.

Code. Queue errors generated by the system are shown as negative numbers in the Code column. For example, if you specify a table index that is greater than the number of elements in the table, an error -12, "Invalid table index," appears, as in message #3 above. Common queue errors for each command are listed in the *PAC Control Command Reference* and in online help.

If the Code column indicates *User*, the error is one you have placed in the queue using the command Add Message to Queue. User messages can help with troubleshooting. Message #2 above is an example of a message the user placed in the Temperature_Control chart.

Severity. The Severity column indicates the type of message: information, warning, or error.

Chart, Block, Line, Object. If a PAC Control command in the strategy caused an error, the chart name, block number, and line number (if you are in Full Debug mode) where the command appears are listed. Message #3 above is an example: an invalid table index was used in block 19 of the Temperature_Control chart. The Object column shows the table name.

If an error did not occur in a strategy chart, the Chart column shows <system>. Messages 1 and 2 occurred when the strategy was unable to initialize an I/O unit, so the Chart column shows <system: _INIT_IO>. Block and Line do not apply, but the Object column shows the name and IP address of the I/O unit.

If an error was caused by a subroutine, the Chart column shows the name of the chart that calls the subroutine, and the Block column shows the name of the subroutine and the block number where the error occurred, in the format `<subroutine name>.<block number>`. Messages 4, 5, and 6 are examples; these errors occurred in block 1 of the subroutine Variable_Increase_Notification, which was called by the Temperature_Control chart.

Using Queue Messages

If a block number is listed for the error, look in that block in the strategy to find the PAC Control command that caused the error. The easiest way to find a block is to open the chart or subroutine, then choose Center on Block from the View menu. You can click the Block column to sort the blocks by number and locate the one with the problem.

To see which line within a block is causing the error, in PAC Control Configure mode, choose Configure > Full Debug. When the error appears in the queue, it will include the line number of the command as well as the block ID.

Status Codes

Status Codes indicate the success or failure of a PAC Control command (instruction), and they are reported by the control engine to a variable in your PAC Control strategy or as a returned value in OptoScript. The status code is either zero (indicating a successful command) or a negative number (indicating an error).

For example, suppose you use the command Transmit Numeric Table. You create a variable named Transmit_Status to put the status of the command in. In Transmit_Status you receive either a zero, indicating that the table was successfully transmitted, or you receive a negative number such as -37 or -42.

Status codes that may be returned are listed for each command in the *PAC Control Command Reference* and in online help.

Naming Conflicts

During various procedures, PAC Control checks for duplicate names of objects such as variables, charts, subroutines, strategy variables, OptoScript instruction names, and reserved keywords. If a naming conflict is found, you may need to rename an object or perform some other housekeeping procedures. When converting to a newer PAC Project, make sure to update PAC Control strategies first and then PAC Display projects so that PAC Display projects are updated based on changes to the strategy database.

List of Common Messages

The following messages may appear in PAC Control. They are listed in numeric order.

If an X appears in the Q? column, the code number appears in the message queue. If an X appears in the I/O? column, the message is an I/O unit error and may appear either in the message queue, as a status code in a variable, or both. For more information on handling I/O unit errors, see "Error Handling Commands" on page 331 and form 1776, *Optimizing PAC Project System Performance*.

#	Description	Possible Cause	Q?	I/O?
0	Operation performed successfully.	NOT AN ERROR. Indicates the command was successful.		
-1	Undefined command.	An unknown command was sent to the PAC Control engine.		
-2	Checksum or CRC mismatch.	When comparing DVFs (Data Verification Fields), a mismatch occurred. Examples of DVFs include checksum and CRC.		
-3	Buffer overrun or invalid length error.	In string manipulations, a string was requested that is longer than the string it will be put into, or the destination string length <= 0.	X	X
-4	Device has powered up. ('Powerup clear expected' message received.)	NOT AN ERROR. The device has been turned off and then on again since the last communication, and is now ready.	X	
-5	Operation failed.	An attempt to store the strategy to flash failed, or an attempt to do something with a chart failed (like call, continue, suspend, start, initialize threads).		
-6	Data field error.	Invalid year entered (must be between 2000 and 2099), or invalid data read from memory when attempting to read the strategy from flash memory.	X	
-7	Watchdog timeout has occurred.	See "Add I/O Unit Dialog Box" on page 131.		
-8	Invalid data.	Invalid data read when attempting to read a strategy from flash, or an invalid character number was passed to a string function.		
-10	Invalid port number.	Valid range for Ethernet is 0–65535. For a communication handle, serial port format may be incorrect.		X
-11	Could not send data.	A Transmit or Transfer command fails when using a comm handle. For example, an attempt is made to send a file to a remote ftp server that has gone off-line.		
-12	Invalid table index.	Used an index greater than the number of elements in the table.	X	X

#	Description	Possible Cause	Q?	I/O?
-13	Overflow error.	Typically, a math result too big to fit in the value passed, while doing ***number conversion functions*** such as converting a float value to engineering units, a float to an unsigned integer, a float to an integer, a 64-bit integer to a 32-bit integer, a floating point value to a 64-bit integer, a floating point value to an unsigned 64-bit integer, an ASCII value to a float. NOTE: In OptoScript, an If Statement expects an 32-bit number and will automatically convert any other type (such as 64-bit integer or float value) to a 32-bit number. To avoid this overflow error, a slight change to the code can avoid this problem. For example: if (your_int_64_here) then... becomes: if (your_int_64_here <> 0) then... Also ***math functions*** such as multiply, exponentiation, hyperbolic sine, hyperbolic cosine, function x to the y, add, subtract, modulo, negate, move/assign. ***Time/date functions***: setting a month not in the range 1–12, a day not in the range 1–31, hours not in 0–23, minutes 0–59, seconds 0–59. Number entered on host port (for example, through PAC Terminal) was too big for data types.	X	
-14	Invalid number.	Math resulted in an invalid number (like infinity or an imaginary number): natural log of the floating point number, square root, arc sine, arc cosine of float, function x to the y (with negative x).	X	
-15	Cannot divide by zero.	Attempted to divide a number by zero.	X	
-16	Bus error.	Contact Product Support. See Opto 22 page 4.		
-17	Port already locked on PAC Control engine.	Attempted to lock a connection that's already locked.		X
-20	Device busy. May be in use by another user or another application.	A resource is already acquired by another task or process.	X	
-21	Had to relock host port in 'QUIT'.	Host port needed relocking.		
-23	Destination string too short.	In string manipulations, a string was requested that is longer than the string it will be put into, or destination string length <= 0.	X	X
-25	Port not locked.	Attempted to transmit or receive on a connection that wasn't locked, or to unlock a connection that wasn't locked.		X
-26	Unknown response from device.	OptoMMP-based protocol packet returned by the device was invalid.		X
-29	Wrong object type. Most likely caused by moving a pointer table element to a pointer of the wrong type.	An object was passed to a command that doesn't handle that object type.		
-34	Invalid I/O command or invalid memory location.	Contact Product Support. See page 4.		X

#	Description	Possible Cause	Q?	I/O?
-35	I/O Point mismatch. The Object field specifies the point name, board name, module number, and point number. For example: BadPoint AIARMS Point Analog (R1: 2, 1) Point name Board Module Point	A point is incorrectly configured (for example, an output point is configured as an input point), or the point type is not supported by the I/O Unit's firmware. Check www.opto22.com for firmware updates.	X	X
-36	Invalid command or feature not implemented.	Feature not yet implemented for this hardware/command combination, or command may not apply to the type of communication handle you are using.		X
-37	Timeout on lock.	Unable to lock a resource (such as a variable) for exclusive writing within the timeout period.		X
-38	Timeout on send.	Unable to send communication in the timeout period.		
-39	Timeout on receive.	Unable to receive communication in the timeout period.		
-42	Invalid limit (on string index, task state, priority, etc.).	A character number greater than the length of the string was used (or the string had a zero length), or an invalid value was passed when setting the state of a chart (running, etc.)		
-44	String too short.	String less than 8 characters used to read time (“hh:mm:ss”), or not 8 or 10 characters for a date, or zero length on a string function.	X	
-45	Null string.	Attempted to use an uninitialized string.		
-46	Invalid string.	String not 8 characters long when setting time, or invalid format when setting date, or invalid communication handle string or comm handle command (such as a missing colon).	X	
-47	Invalid connection. Device drivers might be missing or not loaded/running.	Attempted to open an already open connection.		X
-49	No more connections are available. Maximum number of connections already in use.	No more sessions available for Ethernet.		X
-50	Open connection timeout. Could not establish connection within the timeout period.	Unable to open a connection in time.		X
-52	Invalid connection—not opened.	Attempted to close a connection that wasn’t opened. Communication handle may have been closed by a previous command that failed.		X
-57	String not found.	Substring not found in the string being searched.		
-58	No data received. or Character not found.	Attempted to read an empty buffer (or a connection with no characters waiting); or the I/O unit may be turned off or unreachable; or when searching a string for a particular character, the character wasn’t found.		X
-59	Could not receive data.	Command may not apply to the type of communication handle you are using; for example, Receive commands cannot be used with ftp comm handles. (Use the *get* option with Send Communication Handle Command instead.)		
-60	Empty stack error. PAC Control engine attempted to perform an operation that expected data on the PAC Control engine stack.	Contact Product Support. See Opto 22 page 4.		

#	Description	Possible Cause	Q?	I/O?
-61	Dictionary full error. PAC Control engine dictionary is full and no more 'words' can be defined.	A number of things can reduce the amount of dictionary space, such as: • Too many blocks or variables in a strategy • Too many large tables • Using background downloads, which cuts available memory in half		
-62	Stack full error. PAC Control engine stack has grown too big.	The PAC Control engine stack is full.		
-64	Execute-only error. A command or 'word' was encountered that cannot be used when compiling.	Contact Product Support. See Opto 22 page 4.		
-66	Requested item in protected dictionary	Attempted to remove a strategy when there was no strategy in the control engine.		
-67	Out of memory. To minimize the size of your strategy, reduce the number and size of variables (especially tables). You can also shrink your strategy by using subroutines to perform common tasks	No room left to create any variables or save any data on the stack. Or an attempt was made to save the strategy to flash, without enough room in flash to save it.	X	
-69	Invalid parameter (null pointer) passed to command.	Attempted to use an uninitialized pointer, or a null pointer was received by a command.	X	
-70	Not enough data supplied.	Table index given is larger than the size of the table.		X
-71	Out of persistent memory. If applicable, check length of tables.	Too many persistent variables, variables initialized on download, or too large a strategy archive to fit in battery-backed RAM.		
-72	Subroutine nesting too deep	Reduce the number of layers of subroutines called.		
-73	Invalid subroutine or parameters.	In configure mode, choose Edit > Find, select Global and Operand, and then click Find to search for objects with the same name as the subroutine. Rename any objects that you find.		
-93	Communication to I/O unit not enabled. Previous communication failure may have disabled communication to the unit automatically. Re-enable it and try again.	Communication to the I/O unit may have been disabled by a communication failure that happened earlier.		X
-95	Stack not empty.	Contact Product Support. See Opto 22 page 4.		
-94	Invalid event error	An attempt to enable an event interrupt on a null entry in the event/reaction table. An illegal reaction command specified.		X
-103	Port could not be unlocked. Task attempting to unlock the port does not match the task with the current lock on that port.	Attempted to unlock a connection that was locked by a different task.		X
-203	Driver could not be found or loaded.	Communication command didn't find the driver described in the communication handle. Make sure driver name is not misspelled (for example, tcp must be lower case).		X
-407	File not found.	Attempted to save a strategy to flash, but there was no strategy in RAM to save.		
-412	TCP/IP: Cannot connect error	Ethernet "connect" failed. See page 417.		X
-417	Cannot open file	File does not exist or file name may be incorrect.		

#	Description	Possible Cause	Q?	I/O?
-430	Invalid data range. Verify high value is greater than low value.	Invalid data passed to driver (for example, a point number larger than the maximum number of points on the rack).		X
-433	Object/device already locked.	Couldn't set the state of a chart (running, suspended, etc.) because it's already locked by something else.		
-437	No acceptable socket interface found.	An Ethernet "accept" or "open" was attempted, but no more sessions are available.	X	
-438	Could not create socket.	This error is generated when the controller has run out of available TCP sockets (a limited resource). This is most likely caused when the strategy is overaggressive when trying to establish TCP communications to other devices. When sockets are opened and fail or are closed, there is a time period that must expire before the socket resource may be reused. Sockets that are rapidly opened and closed will cause all available socket resources to enter this time period. When this occurs, any attempt to open an outgoing communication will fail with this error code because there are no available sockets to use for this attempt. When an error occurs when trying to open a TCP communication handle, the strategy should back off for a significant amount of time before reattempting. Subsequent failures should have additional delays applied. These delays slow the consumption of TCP resources.		X
-440	Could not bind socket.	Another device, such as the I/O on the brain, is already using the port. If you are using the Modbus toolkit, make sure to use PAC Manager to set the brain's Modbus port to a number other than 502. Try 0.		
-442	Could not accept on socket.	Ethernet “accept” failed.		X
-443	Could not receive on socket.	Ethernet “receive” failed.		X
-444	Could not send on socket.	Ethernet “send” failed.		X
-446	FTP: Login failed.	Incorrect user name or password. The maximum number of login attempts on the server has been exceeded.		
-531	Buffer full.	Attempted to write to a full buffer. For a serial communication handle, data is being sent faster than the serial port can send and buffer it. Use a faster baud rate or a delay between Transfer/Transmit commands.		X
-534	Attempts to communicate with I/O unit failed.	I/O unit may have lost power or network connection.	X	X
-539	I/O error; performing retry	I/O unit may have lost power or network connection. When communicating with I/O Units, retries are logged in the message queue.	X	X

#	Description	Possible Cause	Q?	I/O?
-612	Unable to enqueue sync command	(Applicable only when redundant controller support is enabled.) The synchronization queue contains more than one write command for the same object. In a redundant strategy, write commands (for example, Turn On and Turn Off) are queued until the controller encounters a sync block. When the sync block is executed, the queued commands are processed. New write commands are then queued until the next sync block is executed. A -612 message indicates that an object has been sent more than one write command without a sync block between them. The command that triggered the -612 message can be in any chart in the strategy. To resolve the issue: • Add a sync block between any blocks that send write commands to the same object. • When writing to the same object, put the write commands in the same transactional chart. (This will make it easier to debug -612 errors that are caused when a chart sends a write command to an object in a different chart.)	X	X
-700	PID Loop has been configured outside of this strategy and could conflict with this strategy's logic.	You are trying to download a new strategy, but a PID loop is currently running on the brain. Open PAC Manager and turn off the loop in Inspect mode by changing its algorithm to None.		
-8607	Invalid protocol.	Attempted to set a port to an unknown mode.		X
-8608	Port initialization failed.	While starting up a chart or task, the default host port could not be created.	X	

#	Description	Possible Cause	Q?	I/O?
-8612	Old response to new command	Each command sent by the PAC controller has a sequence ID. Each response from an I/O unit includes the sequence ID of the command it is responding to. This allows the controller to verify that a response from an I/O unit matches the most recent command sent to that I/O unit. An out-of-sync error would occur (and be posted in the PAC controller's message queue) if the response received by the controller is for a previous command rather than the current command. It usually occurs like this: Controller sends a command and waits the timeout period for the response. If no response is received, it sends a re-transmit of the same command, but with a new sequence ID. The controller receives a response to the original command, so the sequence ID does not match the current sequence ID. Possible problems: • **Network problems**. Out-of-sync errors typically occur when communication to the I/O unit is via a network with high latency or inconsistent latency, such as wireless Ethernet, radio modem, cell modem, etc. This usually does not occur on a wired network because normal response times are much shorter than the default timeout period for I/O Units (typically 1 Sec). Note: If an Ethernet packet from a controller to an I/O unit gets dropped, you would get a -539 error instead of an out-of-sync error. • **The timeout interval is too short**. Increase the timeout value to reduce out-of-sync errors.	X	
All -10,000 and -11,000 errors	[Various descriptions]	Socket or network problems. Check cables and connections to control engine; cycle power to control engine.		

C: PAC Control Files

OPTO 22

Introduction

This appendix lists all of the PAC Control file types and special files. You can use this information to determine what types of files are present in your PAC Control project directory.

Files Related to a Strategy

<strategy>.idb	PAC Control strategy database
<strategy>.crf	Controller Run File (compiled file that is sent to the control engine)
<strategy>.crn	Intermediate run file (compiled file that is sent to the control engine)
<strategy>.crn1	Intermediate run file (component of the run file)
<strategy>.crn2	Intermediate run file (component of the run file)
<strategy>.crn3	Intermediate run file (component of the run file)
<strategy>.inc	Initialization data for variables with "Init on Download" option (component of the run file)
<strategy>.inf	Strategy configuration information
<strategy>.$idb	Temporary PAC Control strategy database file
<strategy>.lidb	PAC Control strategy database lock file
<strategy>.per	Persistent variable definitions
<strategy>.<control engine>.cdf	Control engine download file for special circumstances (see page 207)
<chart name>.cht	Chart
<chart name>.ccd	Compiled chart code (component of the run file)
<chart name>.con	Online compiled chart code
<chart name>.cxf	Exported chart file
<filename>.wth	Watch window file (you name the file)
<filename>.otg	Exported I/O configuration file (you name the file)
<strategy.date.time>.zip	Strategy archive file (automatically named; see "Archiving Strategies on the Computer" on page 201 for file name formats)

Files Associated with a Subroutine

<subroutine name>.isb	Subroutine
<subroutine name>.ini	Subroutine configuration information
<subroutine name>.isc	Compiled subroutine (component of the run file)
<subroutine name>.lisb	Subroutine lock file

Files in the PAC Control Directory

<xxx>.io.def	Object definition files (commands, I/O points, and I/O units). You must not modify these files.
ioCtrl.exe	PAC Control executable file
PAC Control.cnt	PAC Control help contents file
PAC Control.GID	PAC Control help support file (created when you launch the help file)
PAC Control.hlp	PAC Control help file
PAC ControlCommands.cnt	Commands help contents file
PAC ControlCommands.GID	Commands help support file (created when you launch the help file)
PAC ControlCommands.hlp	Commands help file
ioCtrlTools.dat	File that lists software applications you've configured in the Tools menu to launch from PAC Control
ioSnif.log	PAC Message Viewer log file
OptoScriptTemp.txt	A temporary file
Readme.txt	README text file containing information about PAC Control

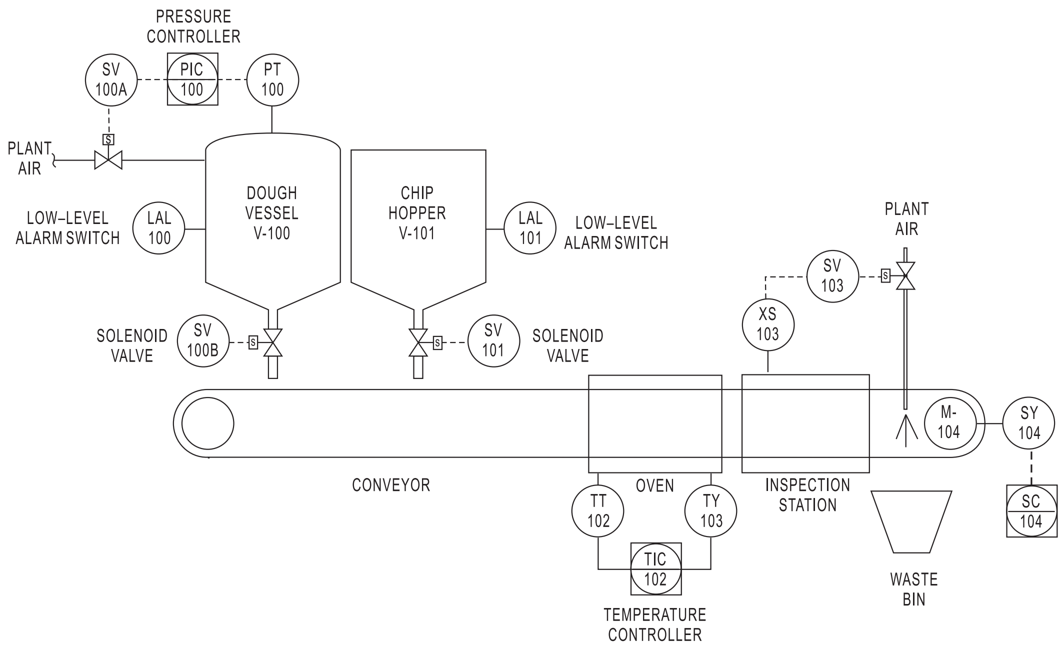
D: Sample Strategy

Introduction

Chapter 2 introduced the Cookies strategy, a sample project used to illustrate how PAC Control works. Although this strategy is based on a mythical factory, you may want to know more about the factory, its process, and its hardware. This appendix gives you that information.

Factory Schematic

The following schematic drawing summarizes the cookie factory:



Description of the Process

Dough Vessel

The first station in our process is the dough vessel. This tank contains a pre-made cookie dough mix.

Dough is dispensed onto the conveyor belt through a valve (SV-100B) at the bottom of the vessel. The dough, being somewhat viscous, must be kept under low pressure to dispense properly. To monitor the pressure, we have included a pressure transmitter (PT-100) in the vessel. Our control engine (a SNAP PAC R-series I/O system) maintains the vessel pressure through a plant air valve (SV-100A).

The vessel also includes a level switch (LAL-100) to tell us when the dough level is low. When it is, the process is halted so that an operator can refill the vessel.

Chip Hopper

The chip hopper supplies chocolate chips. A chip dispenser valve (SV-101) controls the number of chips dropped on each cookie. Like the dough vessel, this tank also includes a level switch (LAL-101) to stop the system when the chip hopper needs refilling.

Oven

After the dough and chips have been dropped onto the conveyor, the conveyor sends the cookie into the oven, and the oven bakes it.

Inspection Station

Our freshly baked cookies then move to the inspection station, where someone inspects them. If the cookie does not fall within normal tolerances— for example, it doesn't have enough chips or is shaped oddly—the inspector closes a switch (XS-103), signaling the bad cookie. A valve (SV-103) then opens to allow plant air to blow the reject cookie into a waste bin.

If the cookie passes the inspection, it moves on to packaging and shipping.

Conveyor

The conveyor and its motor continuously move the cookies from the dough vessel to the inspection station. The conveyor speed is controlled through an analog output (SY-104) from a speed controller (SC-104).

Emergency Stops

Wired at key locations around our bakery are emergency-stop buttons. If something goes wrong with the process, an operator can press any of these E-STOP buttons.

The buttons are wired in series and are normally closed, so pressing any E-STOP button breaks the circuit. One digital input can monitor all the buttons. The system can be restarted by resetting the button.

Required I/O

Here's the list of analog and digital I/O modules required for the cookie factory:

Analog I/O

Name	Description	Type	Module	Range
PT-100	Dough Vessel Pressure	Input	SNAP-AIV (–10 to +10 VDC)	0–15 psig
TT-102	Oven Temperature	Input	SNAP-AICTD (ICTD)	-50–350°C
TY-102	Oven Temperature Control	Output	SNAP-AOV-27 (–10 to +10 VDC)	0–100%
SY-104	Conveyor Speed Control	Output	SNAP-AOV-27 (–10 to +10 VDC)	0–100%

Digital I/O

Name	Description	Type	Module	States
SV-100A	Pressure Control Valve	Output	SNAP-ODC5SRC (5–60 VDC)	0=Closed 1=Open
SV-100B	Dough Dispense Valve	Output	SNAP-ODC5SRC (5–60 VDC)	0=Closed 1=Open
LAL-100	Dough Level Alarm	Input	SNAP-IDC5D (2.5–28 VDC)	0=OK 1=Low
SV-101	Chip Dispense Valve	Output	SNAP-ODC5SRC (5–60 VDC)	0=Closed 1=Open
LAL-101	Chip Level Alarm	Input	SNAP-IDC5D (2.5–28 VDC)	0=OK 1=Low
XS-103	Inspection Signal	Input	SNAP-IDC5D (2.5–28 VDC)	0=OK 1=Reject
SV-103	Reject Valve	Output	SNAP-ODC5SRC (5–60 VDC)	0=Closed 1=Open
XS-105	Emergency Stop	Input	SNAP-IDC5D (2.5–28 VDC)	0=Stop 1=OK

E: OptoScript Language Reference

Introduction

This appendix includes the following reference information about the OptoScript language:

OptoScript Comparison with Standard Programming Languages 439
Notes to Experienced Programmers. 444
OptoScript Lexical Reference 446
OptoScript Grammar Syntax Reference. 449

OptoScript Comparison with Standard Programming Languages

The tables on the following pages compare OptoScript functions and variables to those available in Pascal, BASIC, and C. For more information on using OptoScript, see "11: Using OptoScript," on page 369.

General Notes:

1. The BASIC column is based on Microsoft's Visual Basic language.
2. The Pascal column is based on Borland's ObjectPascal language.
3. The use of logical statements in BASIC and Pascal is significantly different than in OptoScript and C. BASIC and Pascal have a Boolean data type; OptoScript and C use integers. OptoScript and C treat a zero value as false and a non-zero value as true.
4. In OptoScript, you cannot use a break type of command in a loop.

OptoScript can test only one case in a switch at a time; other languages can test more than one.

Function Comparison

	OptoScript	BASIC[1]	C	Pascal[2]
integer 32 (decimal)	123	123	123	123
integer 32 (hexadecimal)	0x123ABC	&H123ABC	0x123ABC	&123ABC
integer 64 (decimal)	123i64	Not available	123LL or 123i64	123
integer 64 (hexadecimal)	0x123ABCi64	Not available	0x123ABCLL or 0x123ABCi64	&123ABC
float (standard)	12.34	12.34	12.34	12.34
float (scientific)	12.34E+17	1.234E+18	1.234E+17	12.34E+17
string	"hello"	"hello"	"hello"	'hello'
character	65 'A'	Not available	65 'A'	'A'
block comment	/* */	Not available	/* */	{ }
line comment	//	'	//	//
numeric assignment	n = 3;	n = 3	n = 3;	n := 3;

numeric table assignment	t[0] = 1;
	t[i] = 2;
	t[i+1] = 3;
		t(0) = 1
		t(i) = 2
		t(i + 1) = 3
			t[0] = 1;
			t[i] = 2;
			t[i+1] = 3;
				t[0] := 1;
				t[i] := 2;
				t[i + 1] := 3;

numeric expressions	i = (f * 2.0) + t[3];
	t[4] = n + ((x – y) * z);
		i = (f * 2.0) + t(3)
		t(4) = n + ((x – y) * z)
			i = (f * 2.0) + t[3];
			t[4] = n + ((x – y) * z);
				i := (f * 2.0) + t[3];
				t[4] := n + ((x – y) * z);

string assignment	s = "hello";
	s = s2;
		s = "hello"
		s = s2
			strcpy(s, "hello");
			strcpy(s, s2);
				s := 'hello';
				s := s2;

string table assignment	st[0] = s;
	st[1] = "hello";
	st[1+i] = st[5];
		st(0) = s
		st(1) = "hello"
		st(1 + i) = st(5)
			strcpy(st[0], s);
			strcpy(st[1],"hello");
			strcpy(st[1+i], st[5]);
				st[0] := s;
				st[1] := 'hello';
				st[1+i] := st[5];

string characters	n = s[0];
	s[0] = 'A';
		Not available	n = s[0];
			s[0] = 'A';
				s[0] := 'A';

	OptoScript	BASIC[1]	C	Pascal[2]

string expressions	s = "hello" + s2 + s3;
	s = "hello" + Chr(n);
		s = "hello" + s2 + s3;
		s = "hello" + Chr(n)
			strcpy(s, "hello");
			strcat(s, s2);
			sprintf(s, "hello%c", n);
				s := 'hello' + s2 + s3;
				s := 'hello' + Chr(n);

equal	x == y	x = y	x == y	x = y
not equal	x <> y	x <> y	x != y	x <> y
less than	x < y	x < y	x < y	x < y
less than or equal	x <= y	x <= y	x <= y	x <= y
greater than	x > y	x > y	x > y	x > y
greater than or equal	x >= y	x <= y	x >= y	x <= y
logical OR (See Note 3)	x or y	(x <> 0) Or (y <> 0) a Or b (Booleans only)	x || y	(x <> 0) Or (y <> 0) a Or b (Booleans only)
logical AND	x and y	(x <> 0) And (y <> 0) x And y (Booleans only)	x && y	(x <> 0) And (y <> 0) a And b (Booleans only)
logical XOR	x xor y	(x <> 0) Xor (y <> 0) x Xor y (Booleans only)	Not available	(x <> 0) Xor (y <> 0) a Xor b (Booleans only)
logical NOT	not x	Not (x <> 0) Not b (Booleans only)	!x	Not (x <> 0) Not b (Booleans only)
logical expressions	(x >= 0) and (y == 3) not ((x>0)or(not y==3))	(x >= 0) And (y = 3) Not((x > 0) Or (y <> 3))	(x >= 0) && (y == 3) !((x > 0) || (y != 3)	(x >= 0) and (y = 3) not((x > 0) or (y <> 3))
bitwise NOT	bitnot x	Not x	~x	not x
bitwise OR (See Note 3)	x bitor y	x Or y	x | y	x or y
bitwise AND	x bitand y	x And y	x & y	x and y
bitwise XOR	x bitxor y	x Xor y	x ^ y	x xor y
bitwise shift left	x << y	Not available	x << y	x shl y
bitwise shift right	x >> y	Not available	x >> y	x shr y

bitwise expressions	i = x bitand 0x0000FFFF;
	i = x << (i * 2);
		i = x And &H0000FFFF
		Not available
			i = x & 0x0000FFFF;
			i = x << (i * 2);
				i := x and &0000FFFF;
				i := x shl (i * 2);

	OptoScript	BASIC[1]	C	Pascal[2]

if statement	if (i == 2) then
	  // do something
	endif
		If (i == 2) Then
		  // do something
		End If
			if (i == 2)
			{
			  // do something
			}
				if (i = 2) then
				begin
				  // do something
				end

if/else statement	if (i == 2) then
	  // do something
	else
	  // do something else
	endif
		If (i == 2) Then
		  // do something
		Else
		  // do something else
		End If
			if (i == 2)
			{
			  // do something
			}
			else
			{
			  // do something else
			}
				if (i = 2) then
				  begin
				    // do something
				  end
				else
				  begin
				    // do something else
				  end

if/else if statement	if (i == 2) then
	  // do something
	elseif (i >= 5) then
	  // do something else
	else
	  // do something else
	endif
		If (i = 2)
		  // do something
		ElseIf (i >= 5) Then
		  // do something else
		Else
		  // do something else
		End If
			if (i == 2)
			{
			  // do something
			}
			else if (i >= 5)
			{
			  // do something else
			}
			else
			{
			  // do something else
			}
				if (i = 2)
				  begin
				    // do something
				  end
				else if (i >= 5)
				  begin
				    // do something else
				  end
				else
				  begin
				    // do something else
				  end

for loop (See Note 4)	for i = 0 to 5 step 1
	  MyTable[i] = i * 2;
	next
		For i = 0 To 5 Step 1
		  MyTable(i) = i * 2
		Next
			for (i = 0; i < 5 ; i++)
			{
			  MyTable[i] = i * 2;
			}
				for i := 0 to 5 do
				begin
				  MyTable[i] := i * 2;
				end

while loop (See Note 4)	while (i < 5)
	  MyTable[i] = i * 2;
	  i = i + 1;
	wend
		While (i < 5)
		  MyTable(i) = i * 2
		  i = i + 1
		Wend
			while (i < 5)
			{
			  MyTable[i] = i * 2;
			  i = i + 1;
			}
				while (i < 5) do
				begin
				  MyTable[i] := i * 2;
				  i := i + 1;
				end

repeat loop (See Note 4)	repeat
	  MyTable[i] = i * 2;
	  i = i + 1;
	until (i > 5);
		Do
		  MyTable(i) = i * 2
		  i = i + 1
		Loop Until (i > 5)
			do
			{
			  MyTable[i] = i * 2;
			  i = i + 1;
			} while !(i > 5);
				repeat
				  MyTable[i] := i * 2;
				  i = i + 1;
				until (i > 5);

	OptoScript	BASIC[1]	C	Pascal[2]

case (See Note 5)	switch (i)
	  case 1:
	    f = 2.0 * x;
	    break
	  case z:
	    f = 2.0 * y;
	    break
	  default:
	    f = 2.0 * z;
	    break
	endswitch
		Select Case (i)
		  Case 1
		    f = 2.0 * x
		  Case z
		    f = 2.0 * y
		  Case Else
		    f = 2.0 * z;
		End Select
			switch (i)
			{
			  case 1:
			    f = 2.0 * x;
			    break;
			//case z: NOT ALLOWED IN C
			// f = 2.0 * y;
			// break;
			  default:
			    f = 2.0 * z;
			    break;
			}
				case i of
				  1:
				    f := 2.0 * x;
				  2:
				    f := 2.0 * y;
				  else
				    f := 2.0 * z;
				end;

Notes:

1. Based on Microsoft's Visual Basic language.
2. Based on Borland's ObjectPascal language.
3. The use of logical statements in BASIC and Pascal is significantly different than in OptoScript and C. BASIC and Pascal have a Boolean data type; OptoScript and C use integers. OptoScript and C treat a zero value as false and a non-zero value as true.
4. OptoScript cannot have a break type of command in a loop as is common with other lanuages.
5. OptoScript can test only one case at a time.

Variable Comparison

Variable Name	PAC Control Type	BASIC Example	C Example	Pascal Example
n	integer 32	Dim n as Long	long n;	n: Integer;
nn	integer 64	Not available	LONGLONG d;	d: Int64;
f	float	Dim f as Single	float f;	f: Single;
s	string	Dim as String	char s[128];	s: ShortString;
p	pointer	not available	long * pn;	pn: ^Integer;
nt	integer 32 table	Dim nt(10) as Long	long i[10];	nt: array[0..9] of Integer;
ft	float 32 table	Dim ft(10) as Single	float f[10];	ft: array[0..9] of Single
st	string table	Dim st(10) as String	char s[10][128];	st: array[0..9] of ShortString;
pt	pointer table	not available	void * pt[10];	pt: array[0..9] of ^Integer;

Notes to Experienced Programmers

Experienced programmers, especially those who are new to PAC Control, may be interested in the following notes.

Variable Database and Other Surprises

PAC Control maintains a database of all declared variables—a notable difference from common procedural languages. Variables are not declared in the programming code, but in the PAC Control tag database. This is a basic concept of PAC Control and how it ties in with PAC Display, but may seem odd to experienced programmers using PAC Control for the first time. Also, all variables and objects are global. Local variables do not exist in PAC Control in the way they do in most procedural languages. Subroutines in PAC Control contain "local" variables, but those local variables apply throughout that subroutine.

Most languages allow you to return from a function before it ends, but OptoScript does not. The same effect can be achieved in other ways, however, such as introducing tests into the code. (Some people argue that this limitation produces better programming, because each function has only one exit point.)

PAC Control's Target Audience

Because PAC Control is based on OptoControl, which was conceived as a simple programming tool for non-programmers, it is designed to be relatively foolproof. Even though OptoScript provides advanced functionality, this philosophy also influenced the design of OptoScript. OptoScript exists only inside OptoScript blocks, which can exist only inside a flowchart. Flowcharts are the basis of PAC Control.

Even an experienced programmer may want to think twice before using OptoScript blocks extensively. Many programmers like PAC Control's simplicity not for themselves but for the field technicians and maintenance personnel who will use it in the field. While you could write an entire chart (or conceivably an entire strategy) in one block, doing so would eliminate most of the advantages of using PAC Control. Consider limiting your use of OptoScript to math operations, complex string manipulation, and other logic most suited to scripting, so you retain PAC Control’s advantages for non-programmers.

Language Syntax

In the same way, OptoScript syntax is meant to be simple enough for a beginner to understand but also easy for an experienced programmer to learn quickly.

Some programmers may wonder why OptoScript is not modeled after just one existing language, such as BASIC or C. Instead, one can clearly see influences from Pascal, BASIC, and C. PAC Control's target audience is one reason; internal consistency with PAC Control commands and the capabilities and limitations of Opto 22 control engines are another.

Some aspects of OptoScript were designed to be consistent with PAC Control commands. For instance, the *bitwise and* operator was named *bitand* in OptoScript because there is a command in PAC Control named *Bit AND*.

OptoScript provides all the functionality of Opto 22 control engines but is subject to their limitations. For instance, OptoScript provides some convenient ways of working with strings, but only to a certain point.

For example, in an assignment statement, strings can be added together like this:

```
strDate = strMonth + "/" + strDay + "/" + strYear;
```

It would certainly be nice to use the same kind of string addition in a procedure call:

```
TransmitString(strMonth + "/" + strDay + "/" + strYear, nPort);
```

However, due to the current abilities of control engines, this type of string addition inside a function call is not possible.

OptoScript Lexical Reference

Token Syntax Legend

Tokens are the smallest units that the OptoScript compiler processes.

bold character:	specific character
(parenthesis):	content is treated as a unit
[brackets]:	set of possible items
opt subscript:	item is optional
no subscript:	item is not allowed
0+ subscript:	0 or more items may be chosen
1 subscript:	one item must be chosen
1+ subscript:	one or more items must be chosen

*:	any character
letter:	any character a through z, upper or lower case
digit:	one of [0 1 2 3 4 5 6 7 8 9]
non-zero-digit	one of [1 2 3 4 5 6 7 8 9]
hex-digit:	one of [0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f]

Literals and Names

Token Name	Token Syntax	Comments
Int32Literal	`0[xX]`$_1$`hex-digit`$_{1+}$	Hexadecimal Integer 32-bit Good examples: `0x12AB, 0X12aB, 0x0` Bad examples: `x123ABC, 0123ABC`
	`0` `non-zero-digit`$_1$ `digit`$_{0+}$	Decimal Integer 32-bit Good examples: `0, 123, 7890` Bad examples: `123ABC`
	`'[*'`$_{no}$`]`$_1$`'`	Single Character Good examples: `'A', '1', ' '` Bad examples: `'''`
Int64Literal	`0[xX]`$_1$`hex-digit`$_1$`+i64`	Hexadecimal Integer 64 Good examples: `0x12ABi64, 0X12aBi64, 0x0i64` Bad examples: `x123ABCi64, 0123ABCi64`

	`0i64`
	`digit`$_1$`i64`
	`non-zero-digit`$_{0+}$`i64`
		Decimal Integer 64
		Good examples: `0i64, 123i64, 7890i64`
		Bad examples: `123ABCi64`

FloatLiteral	`digit`$_{0+}$ `. digit`$_{1+}$ `([Ee]`$_1$ `[+-]`$_{opt}$ `digit`$_{1+}$`)`$_{opt}$	Float Literal Good examples: `1.0, 2.3, 1.2e6, 1.2e-6, .1, .2e6` Bad examples: `1., 1.e7, 1e7`
StringLiteral	`"[*"`$_{no}$`]`$_{0+}$`"`	String Literal. Confined to single line. Good examples: `"abc def"` Bad examples: `"abc"def"`

Token Name	Token Syntax	Comments
NumericVariable StringVariable ChartVariable DigIoUnitVariable MixedIoUnitVariable PointerVariable NumericTable StringTable PointerTable CommunicationHandle	`[letter]`$_1$ `[letter digit _ ]`$_{0+}$	A letter followed by mix of letters, digits, and underscores. The name must be found in the PAC Control database. Good examples: `MyInt, MyInt2, My_Int_3` Bad examples: `_MyInt, 0MyInt`
CommandProcedure CommandProcedureNoArgs CommandFunction CommandFunctionNoArgs	`[letter]`$_1$ `[letter digit _ ]`$_{0+}$	A letter followed by mix of letters, digits, and underscores. The name must be a built-in command or subroutine. Good examples: `Sine, Sine2, Sine_3` Bad examples: `_Sine, 0Sine`

Keywords (Reserved Words)

if	for	switch	while	Chr
then	to	endswitch	do	
else	step	case	wend	null
elseif	next	break		
endif		default	repeat	
			until	

Operators

The following table lists operators in order of highest to lowest precedence:

Operator	Name/Meaning	Comments
-	negation	
`not`	logical not	
`bitnot`	bitwise not	
*	multiplication	
/	division	
%	modulo division	
-	subtraction	
+	addition	
+=	string append assignment	
<<	bitwise left shift	
>>	bitwise right shift	

Operator	Name/Meaning	Comments
`==`	equality	
`<>`	non-equality	
`<`	less than	
`<=`	less than or equal to	
`>`	greater than	
`>=`	greater than or equal to	
`bitand`	bitwise and	
`bitor`	bitwise or	
`bitxor`	bitwise exclusive or	
`and`	logical and	
`or`	logical or	
`xor`	logical exclusive or	
`not`	logical not	
`(  )`	parentheses	no precedence
`[  ]`	brackets	no precedence
`:`	colon	no precedence
`;`	semi-colon	no precedence
`,`	comma separator	no precedence
`=`	assignment	no precedence
`&`	address of	no precedence

Comments

OptoScript has two kinds of comments: single line and block.

Single line comments are indicated by two slashes, followed by any sequence of characters, until the end of the line.

Examples:

```
i = a + b; // this is a comment
i = a + b; //determine i by adding a and b together
// i = a + b; // This whole line is commented out
```

Block comments are indicated by a slash and an asterisk (`/*`), followed by any sequence of characters, and ending with an asterisk and a slash (`*/`). This type of comment may span multiple lines. Block comments may not be nested.

Examples:

```
i = a + b; /*determine i by adding a and b together*/
i = a + b; /* determine i by adding
```

```
a and b together */
/* i = a + b; // determine i by adding a and b together */
```

OptoScript Grammar Syntax Reference

Tokens are in regular type.
Keywords and operators are in **bold** type.
Syntax Rules are in *italic* type.

Program
> *StatementList*

StatementList
> *Statement*
> *StatementList Statement*

Statement
> *AssignmentStatement*
> *StrAssignmentStatement*
> *PtrAssignmentStatement*
> *ProcedureCommand*
> *FunctionCommand ;*
> *ConditionStatement*
> *ForStatement*
> *WhileStatement*
> *RepeatStatement*
> *SwitchStatement*

ProcedureCommand
> CommandProcedureNoArgs *( )* **;**
> CommandProcedure **(***ArgumentList* **) ;**

FunctionCommand
> CommandFunctionNoArgs **( )**
> CommandFunction **(***ArgumentList* **)**

ArgumentList
> *NumericExp*
> *ArgumentList , NumericExp*
> *StrIdentifier*
> *ArgumentList , StrIdentifier*
> *ObjVarIdentifier*
> *ArgumentList , ObjVarIdentifier*

NumericExp
> **(** *NumericExp* **)**
> *NumericExp*
> *LogicalExp*
> *LogicalUnaryExp*

> *AdditiveExp*
> *MultiplicativeExp*
> *BitwiseExp*
> *NumIdentifier*
> *NumericLiteral*
> *FunctionCommand*

LogicalExp
> *NumericExp* **and** *NumericExp*
> *NumericExp* **or** *NumericExp*
> *NumericExp* **xor** *NumericExp*
> *NumericExp* **==** *NumericExp*
> *NumericExp* <> *NumericExp*
> *NumericExp* **<** *NumericExp*
> *NumericExp* **<=** *NumericExp*
> *NumericExp* **>** *NumericExp*
> *NumericExp* **>=** *NumericExp*
> *StrIdentifier* **==** *StrIdentifier*
> PointerVariable **== null**
> **null ==** PointerVariable
> **null ==** PointerTable **[** *NumericExp* **]**
> PointerTable **[** *NumericExp* **] == null**

AdditiveExp
> *NumericExp* **+** *NumericExp*
> *NumericExp* **-** *NumericExp*

MultiplicativeExp
> *NumericExp* ***** *NumericExp*
> *NumericExp* **/** *NumericExp*
> *NumericExp* **%** *NumericExp*

NotNumExp
> *ObjVarIdentifier*
> *StrIdentifier*

BitwiseExp
> **bitnot** *NumericExp*
> *NumericExp* **bitand** *NumericExp*
> *NumericExp* **bitor** *NumericExp*
> *NumericExp* **bitxor** *NumericExp*
> *NumericExp* **<<** *NumericExp*
> *NumericExp* **>>** *NumericExp*

AssignmentStatement
> NumericVariable **=** *NumericExp* **;**
> NumericTable **[** *NumericExp* **] =** *NumericExp* **;**
> StringVariable **[** *NumericExp* **] =** *NumericExp* **;**

PtrAssignmentStatement
> PointerVariable **=** *PointableIdentifier* **;**

> PointerVariable **=** PointerTable **[** *NumericExp* **]** **;**
> PointerTable **[** *NumericExp* **]** **=** *PointableIdentifier* **;**

PointableIdentifier
> **null**
> &StringVariable
> &*NumVarIdentifier*
> &*ObjVarIdentifier*

StrAssignmentStatement
> StringVariable **=** *StrExp* **;**
> StringTable **[** *NumericExp* **]** **=** *StrExp* **;**
> StringVariable **+=** *StrIdentifier* **;**
> StringVariable **+=** **Chr (** *NumericExp* **) ;**

StrExp
> *StrAdditiveExp*
> *StrIdentifier*
> **Chr (** *NumericExp* **)**

StrAdditiveExp
> *StrExp* **+** *StrExp*

StrIdentifier
> StringVariable
> StringLiteral
> StringTable **[** *NumericExp* **]**

NumIdentifier
> *NumVarIdentifier*
> *NumTableIdentifier*
> *StringCharIdentifier*

NumVarIdentifier
> NumericVariable

ObjVarIdentifier
> ChartVariable
> DigIoUnitVariable
> MixedIoUnitVariable
> *TableIdentifier*
> *CommunicationHandle*

NumTableIdentifier
> NumericTable **[** *NumericExp* **]**

TableIdentifier
> NumericTable
> StringTable

StringCharIdentifier
> StringVariable **[** *NumericExp* **]**

NumericLiteral
> Integer32Literal

> Integer64Literal
> FloatLiteral

LogicalUnaryExp
> **not** *NumericExp*

ConditionStatement
```
> IfStatement
        StatementListOrEmpty
     EndifStatement
> IfStatement
        StatementListOrEmpty
     ElseStatement
        StatementListOrEmpty
     EndifStatement
> IfStatement
        StatementListOrEmpty
     ElseIfList
     EndifStatement
> IfStatement
        StatementListOrEmpty
     ElseIfList
     ElseStatement
        StatementListOrEmpty
     EndifStatement
```

IfStatement
> **if (** *NumericExp* **) then**

ElseStatement
> **else**

ElseIfStatement
> **elseif (** *NumericExp* **) then**
        StatementListOrEmpty

ElseIfList
> *ElseIfStatement*
> *ElseIfList ElseIfStatement*

EndifStatement
> **endif**

CaseList
> *CaseStatement*
> *CaseStatement DefaultStatement*
> *CaseList CaseStatement*
> *CaseList CaseStatement DefaultStatement*

DefaultStatement
> **default :**
        StatementListOrEmpty

    break

CaseStatement
> **case** *NumericExp* **:**
        StatementListOrEmpty
    break

SwitchStatement
> **switch (** *NumericExp* **)**
        CaseList
    endswitch

ForStatement
> **for** NumericVariable **=** *NumericExp* **to** *NumericExp* **step** *NumericExp*
        StatementListOrEmpty
    next

WhileStatement
> **while (** *NumericExp* **)**
        StatementListOrEmpty
    wend

RepeatStatement
> **repeat**
        StatementListOrEmpty
    until *NumericExp* **;**

F: Legacy High-Density Digital Module Commands

All SNAP high-density digital module commands are fully supported by the SNAP PAC System using regular digital point commands. However, three of our older HDD modules can also be used with analog-capable SNAP Ultimate, SNAP Ethernet, and SNAP Simple I/O units, as long as you use the older "legacy" commands. If you are using SNAP-ODC-32-SNK, SNAP-ODC-32-SRC, or SNAP-IDC-32 HDD modules with these older I/O units, see "Legacy Options" on page 232.

The legacy HDD commands are as follows:

States
Get HDD Module States
Get All HDD Module States
Set HDD Module from MOMO Masks
Turn On HDD Module Point
Turn Off HDD Module Point

Counters
Get HDD Module Counters
Get & Clear HDD Module Counter
Get & Clear HDD Module Counters

Latches
Get HDD Module On-Latches
Get HDD Module Off-Latches
Get All HDD Module On-Latches
Get All HDD Module Off-Latches
Clear HDD Module On-Latches
Clear HDD Module Off-Latches
Get & Clear HDD Module On-Latches
Get & Clear HDD Module Off-Latches
Get & Clear All HDD Module On-Latches
Get & Clear All HDD Module Off-Latches

About High-Density Digital Modules

SNAP high-density digital modules can be used on I/O units with the following brains:

SNAP-PAC-R1	SNAP-PAC-EB1
SNAP-PAC-R1-B	SNAP-PAC-EB2
SNAP-PAC-R2	SNAP-B3000-ENET
SNAP-UP1-ADS	SNAP-ENET-RTC
SNAP-UP1-M64	SNAP-ENET-S64

High-density modules cannot be used with digital-only brains, because they communicate with the brain as an analog or special-purpose module communicates.

Comparing SNAP High-Density and Standard Digital Modules

SNAP high-density digital (HDD) modules are different in several other ways from standard SNAP digital input and output modules:

- Standard digital modules contain four points per module; high-density digital modules contain 32 points per module.
- The points on a standard digital input module are isolated from each other. Points on high-density input modules are in four groups of eight points; groups are isolated from each other, but points within a group are not isolated.
- Standard digital modules have LEDs for each point visible on the module's top; high-density modules do not have LEDs.
- Standard digital modules can be placed only in digital slots on the mounting rack; high-density digital modules can be placed anywhere, even in slots marked "Analog Only."
- The turn-on/turn-off time is faster for standard digital modules than for high-density modules. The update time (time for data to pass from the module to the brain) is also faster for standard digital modules and is determined by the speed of the module itself. In high-density modules, update time depends on the brain's analog scanner and is affected by the number of modules on the rack and how busy the brain is with Ethernet communication. (For more specific information on turn-on/turn-off and update times, see the data sheets for standard and high-density modules.) You may find that inserting Delay commands in your strategy provides more accurate results, especially with counters.
- **IMPORTANT:** Each point on a standard digital module is given a name when configured in PAC Control or PAC Manager, and your strategy refers to the point by its name. Points on high-density modules do not have individual names. HDD modules do not require configuration, so their points do not appear in the Configure I/O Points dialog box nor on the Strategy Tree. Because HDD points do not have names, most PAC Control commands use bitmasks to read or write to them.
- Points on HDD modules cannot be disabled in Debug mode for simulating inputs and outputs.
- Counting is done differently. See "Counting on High-Density Digital Modules" below for details.

Counting on High-Density Digital Modules

On standard SNAP digital input modules, any point can be configured as a counter because counting is done on the brain. We refer to it as "high-speed" counting because it can be very fast, depending on the speed of the module.

On high-density SNAP digital modules, however, the module itself does the counting, so no configuration is necessary. The module uses a 16-bit counter, but the brain used with the module accumulates counts to 32 bits by periodically getting and clearing the module's counts and adding each new count to what it already has for each point. Update time varies based on the number of modules on the rack and Ethernet communication demands on the brain. When using PAC Control's signed 32-bit variables, one bit of the 32 is used for the integer's sign (+/–), so counts over 2 billion may not be accurate.

Because counting is done in the module rather than in the brain, you can get counts for high-density digital modules used with SNAP-UP1-M64 and SNAP-ENET-S64 brains, which do not support high-speed counting.

Counting on HDD modules is at 0–50 Hz at a 50% duty cycle. This rate is useful for applications that require counting but not at high speeds—for example, rotating shafts, flow meters that generate pulses, and electrical meters tuned to slower speeds.

Using HDD Module Counters

Counters on HDD modules are always active; you do not need to configure a counter before using it. Because counters are always active, you should clear a counter before using it to make sure the count accurately reflects your current task.

Counters are returned as 32-bit integers, either in a variable for one point, or in an integer 32 table for all points on a module. The table must contain 32 elements beyond the starting index, since there are 32 points on the module.

For more information about specific commands, see Opto 22 form 1701, the *PAC Control Command Reference*, or PAC Control Command Help.

Using HDD Module Commands

As shown in the following table, HDD module commands work either with an individual point, with all points on one module, or with all HDD modules on the I/O unit.

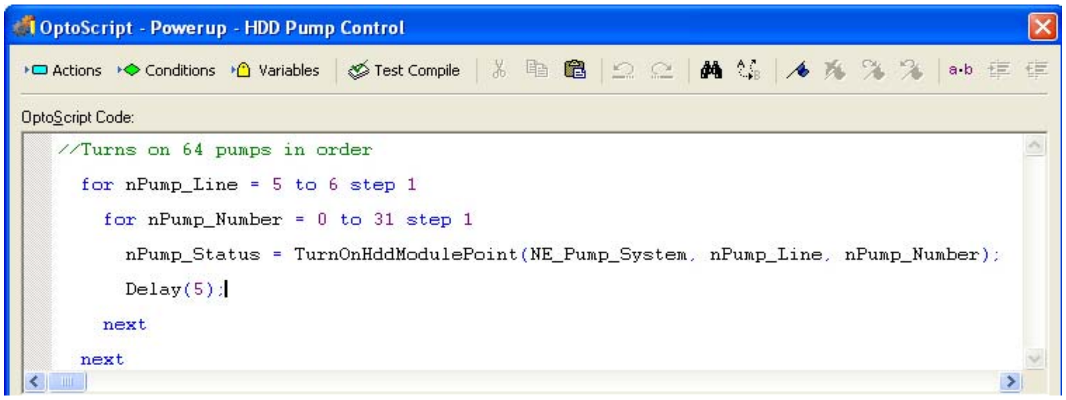
Purpose	Command name	Reads or writes to:		
		One point	One module	All modules
Reading states and latches and clearing latches	Clear HDD Module Off-Latches		X	
	Clear HDD Module On-Latches		X	
	Get & Clear HDD Module Off-Latches		X	
	Get & Clear HDD Module On-Latches			
	Get & Clear All HDD Module Off-latches		X	X
	Get & Clear All HDD Module On-latches			X
	Get All HDD Module States			X
	Get All HDD Module Off-Latches			X
	Get All HDD Module On-Latches			X
	Get HDD Module States		X	
	Get HDD Module Off-Latches		X	
	Get HDD Module On-Latches		X	
Reading and clearing counters	Get HDD Module Counters		X	
	Get & Clear HDD Module Counters		X	
	Get & Clear HDD Module Counter	X		
Writing to points	Set HDD Module from MOMO masks			
	Turn On HDD Module Point	X	X	
	Turn Off HDD Module Point	X		

Individual Point

A few HDD module commands, such as Turn On HDD Module Point and Get & Clear HDD Module Counter, address an individual point by module and point number. Remember that Opto 22 modules and points start at zero.

HDD points do not have names. If you need to refer to a specific point or a group of points by name, however, you can create a PAC Control variable and place data in it.

Creating a variable for a group of points can be very useful in a large installation with many similar inputs or outputs. For example, suppose output modules 5 and 6 on the NE_Pump_System I/O unit each control 32 pumps. You could use an integer 32 variable called nPump_Line for the module number and another integer 32 variable called nPump_Number for the point number. The variables make it easy to loop through all pumps to turn them on, as shown in the following OptoScript example.



```
//Turns on 64 pumps in order
  for nPump_Line = 5 to 6 step 1
    for nPump_Number = 0 to 31 step 1
      nPump_Status = TurnOnHddModulePoint(NE_Pump_System, nPump_Line, nPump_Number);
      Delay(5);
    next
  next
```

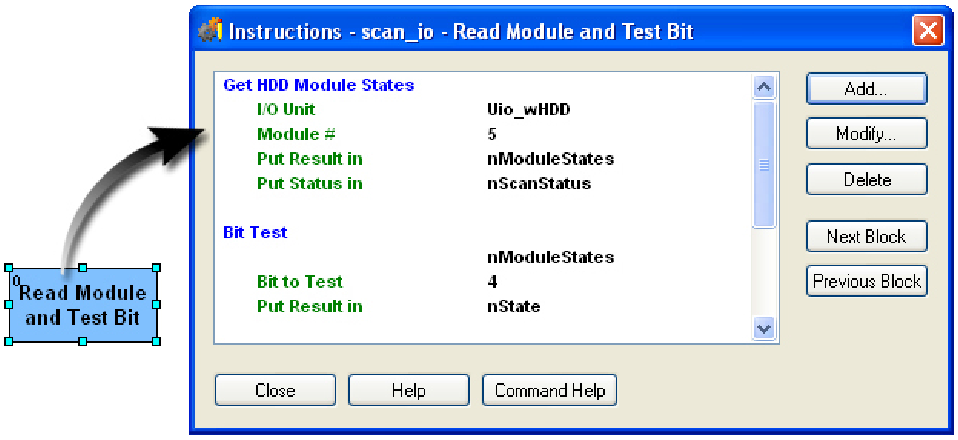
All Points on a Module

Several commands, such as Get HDD Module States, work with all points on a module at once by using a bitmask. The least significant bit corresponds to point zero. A bit with a value of 1 indicates that the corresponding point is on; a bit with a value of 0 indicates the point is off.

Because HDD points on a module are represented and controlled through bitmasks, Logical Commands such as Bit Test and Bit Clear are useful for manipulating the bits.

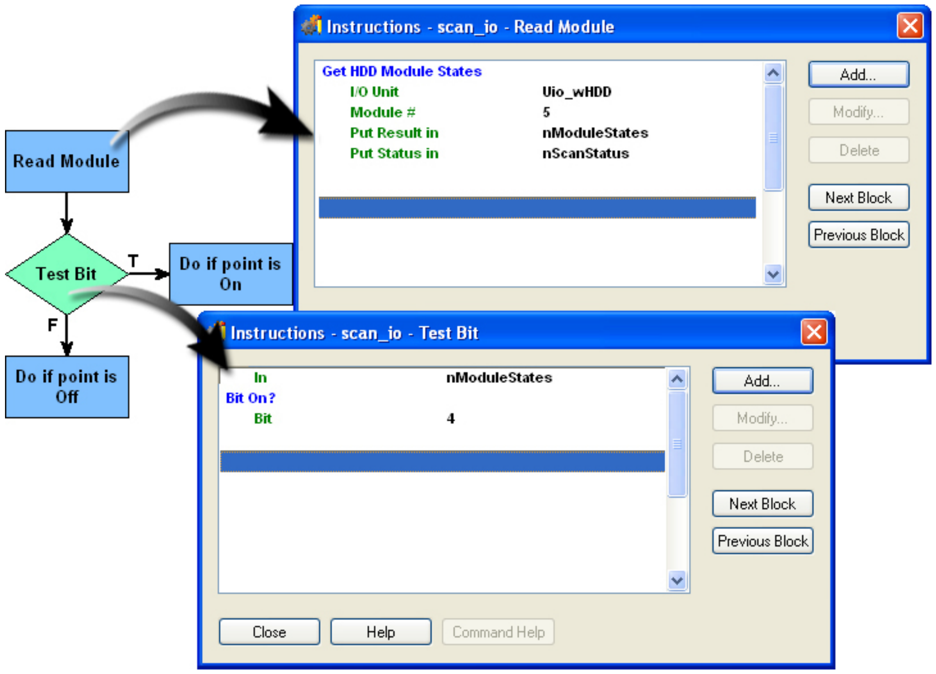
The following example reads the states of all points on the module in rack position 5 and places the bitmask in an integer variable, nModuleStates.

To read point 4, which corresponds to bit 4, you can use Bit Test. The result of the bit test is placed in the variable nState. If nState = 1, the point is on; if nState = 0, the point is off.



```
Get HDD Module States
    I/O Unit            Uio_wHDD
    Module #            5
    Put Result in       nModuleStates
    Put Status in       nScanStatus

Bit Test
                        nModuleStates
    Bit to Test         4
    Put Result in       nState
```

You can place the bit's value in a variable and test it as in the example above, or you can use the bit to direct flowchart logic. In the example below, the Bit On? command tests the bit and exits true if the bit is 1, false if the bit is 0.



```
Get HDD Module States
    I/O Unit            Uio_wHDD
    Module #            5
    Put Result in       nModuleStates
    Put Status in       nScanStatus
```

```
    In                  nModuleStates
Bit On?
    Bit                 4
```

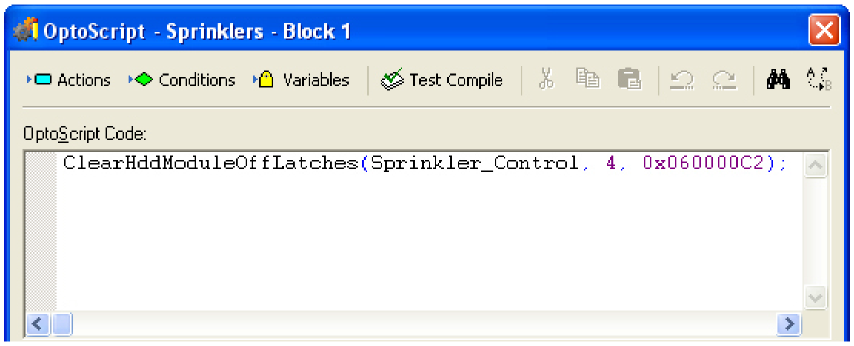
Here's another example of a command that works with all points on the module at once. Suppose you want to reset the off-latches on a few points on the module in rack position 4—specifically, points 1, 6, 7, 25, and 26. Using Clear HDD Module Off-Latches, you indicate the I/O unit, the module number (4), and then use a bitmask to determine the off-latches to clear. The bitmask would be:

```
00000110000000000000000011000010
```

As you can see in the illustration below, the 1 bits in the mask will affect their respective points, so off-latches for point numbers 1, 6, 7, 25, and 26 will be cleared. (To save space, only the first 8 and last 8 off-latches are shown.)

	Point Number	31	30	29	28	27	26	25	24	> > >	7	6	5	4	3	2	1	0
Bit Mask	Binary	0	0	0	0	0	1	1	0	> > >	1	1	0	0	0	0	1	0
	Hex	0				6				> > >	C				2			

Here is this example shown in OptoScript code:

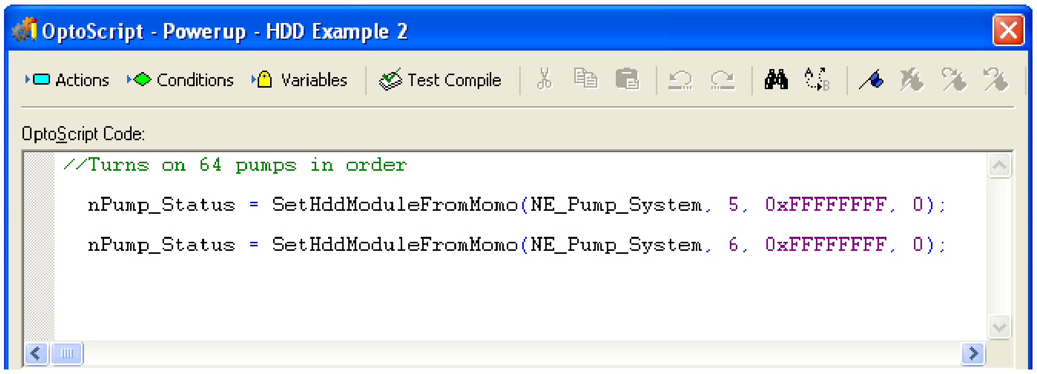

OptoScript - Sprinklers - Block 1

OptoScript Code:

```
ClearHddModuleOffLatches(Sprinkler_Control, 4, 0x060000C2);
```

NOTE: In standard PAC Control commands, you can enter the bitmask in binary or in hex, depending on the integer display you've chosen under the View menu. In OptoScript code, use hex. In either case, don't use decimal to represent masks, since PAC Control's 32-bit integers use the most significant digits as the sign (+/–).

As a final example of working with all points on a module at once, here is a rewritten version of the pump control script shown on page 458. In this example, the pumps controlled by HDD Output modules 5 and 6 on the I/O unit NE_Pump_System are turned on by the Set HDD Module From MOMO Masks command. Notice that this command turns on all the pumps controlled by a single module simultaneously, rather than one after the other.


OptoScript - Powerup - HDD Example 2

OptoScript Code:

```
//Turns on 64 pumps in order
  nPump_Status = SetHddModuleFromMomo(NE_Pump_System, 5, 0xFFFFFFFF, 0);
  nPump_Status = SetHddModuleFromMomo(NE_Pump_System, 6, 0xFFFFFFFF, 0);
```

All HDD Modules on the I/O Unit

Commands with the word "All" in their title—such as Get All HDD Module States—work with all HDD modules on one I/O unit at once. They do so using a table; each element in the table contains a bitmask representing the data for one module.

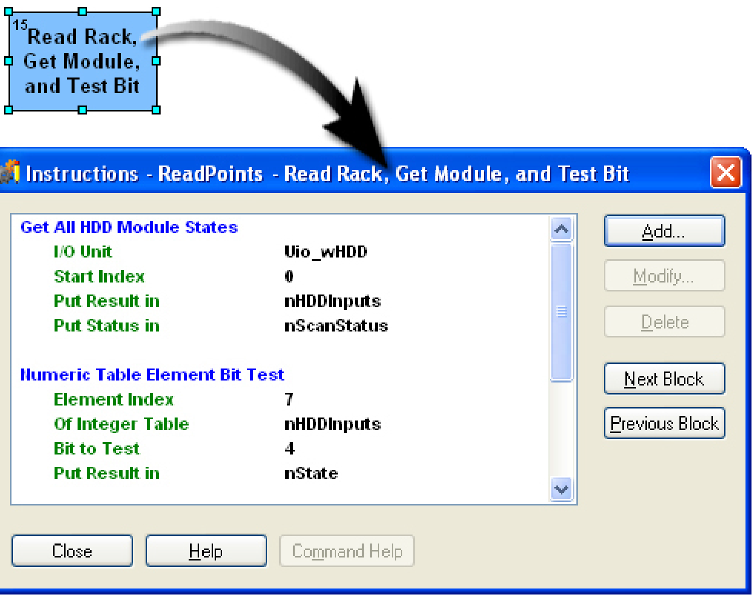
For example, to read the states of all points on all HDD modules on one I/O unit, you would use the command Get All HDD Module States. If the I/O unit consists of an 8-module rack filled with HDD modules, the table would contain data such as the following.

Index	Value (Bitmask)	Description
0	00000001000101000000000110010000	Each index contains the status data for the HDD module in the corresponding position on the rack. A value of 1 indicates that the point is on; a value of 0 indicates that it is off. The least significant bit in the mask corresponds to point zero on the module.
1	01100001010001110000001010110010	
2	00000000000010000100010000000111	In this example, index 2, which contains the status of all points on the module in slot 2, shows that points 0, 1, 2, 10, 14, and 19 are on. All other points on the module are off.
3	00100000011000000010010001000100	
4	01100001010001110000001010110010	
5	00001110000100001100100000001001	
6	10000000110000001110000000100100	
7	00110000011100001111100000000001	
8	0000000000000000000000000000000000	The remainder of the table is zero-filled, since there are no more modules.
↓	↓	

Returned data is only for HDD modules. If the rack contained a standard digital module in position 3 and a serial module in position 4, elements 3 and 4 in the table would be zero-filled.

To use the data in the table, you can manipulate the table using commands such as Shift Numeric Table Elements and Numeric Table Element Bit Set.

For example, the first command in the block shown below places data for all modules on the rack in the table nHDDInputs, starting at table element zero. The second command tests bit 4 in table element 7 (which corresponds to point 4 on the module in position 7) and puts the result of the test into the variable nState.


15
Read Rack, Get Module, and Test Bit
Instructions - ReadPoints - Read Rack, Get Module, and Test Bit
Add...
Modify...
Delete
Next Block
Previous Block
Close
Help
Command Help

Get All HDD Module States

I/O Unit	Uio_wHDD
Start Index	0
Put Result in	nHDDInputs
Put Status in	nScanStatus

Numeric Table Element Bit Test

Element Index	7
Of Integer Table	nHDDInputs
Bit to Test	4
Put Result in	nState

If you need to work with the data for only one module in the table, you can use the command Move From Numeric Table Element to move the module's data from the table into an integer variable. Then you can use bit commands within the integer.

Index

OPTO 22

A

action block
    definition, 238
action command
    definition, 51
active address, 335
adding
    analog point, 139
    command, 270
        in OptoScript, 393
    communication handle, 257
    digital point, 136
    event/reaction, 162
    mistic PID loop, 155
    numeric variable, 257
    PID loop, 151
    pointer, 257
    pointer variable to watch window, 268
    string variable, 257
    table variable, 260
    table variable to watch window, 268, 270
    text to chart, 245
    variable, 257
    variable to watch window, 266
alarm enunciation, 333
algorithms for PID loops, 347
analog
    adding I/O point, 139
    biasing, 333
    changing I/O while strategy is running, 175
    copying I/O point, 147
    I/O unit commands, 334
    minimum/maximum value, 278, 279
    offset and gain commands (instructions), 278
    point commands, 278, 378
    point, definition, 48
    totalizer, 279
    TPO, 278
applications
    launching from PAC Control, 67
archiving strategies
    restoring archives, 118
    to computer, 201
    to control engine, 116
argument, 83, 272
    in OptoScript, 380
ASCII, decimal and hex values of characters, 366
auto stepping through a chart, 213, 215
automation, 47
autorun, 210

B

background downloading, 204, 206
baud rate and mode for serial I/O units (Pro), 133
baud rate for serial I/O units, 133
bill of materials, 226, 227
bit commands, 340
bitwise operator, in OptoScript, 388
block
    block 0, 238
    changing color or size, 239, 248
    connecting blocks, 243
    definition, 238
    deleting, 247
    drawing, 241
    finding in a chart, 63
    moving, 246
    naming, 86, 242
    OptoScript, 370
    selecting, 246
    stepping through, 213
BOM, 226, 227
bookmark, 395

breakpoints in a chart, 215

C

CA root certificate, 316
calling a subroutine, 409
case statement, 375, 389
    programming example, 92
certificate, 316
changing
    active control engine, 110
    analog I/O while strategy is running, 175
    baud rate and mode for serial I/O units (Pro), 133
    baud rate for serial I/O units, 133
    block color or size, 239, 248
    chart background, 239
    chart name or description, 250
    column width in a dialog box, 64
    command (instruction), 274
    connection line color or size, 248
    control engine definition, 109
    debug level, 211
    debugger speed, 212
    digital I/O while strategy is running, 173
    event/reaction, 170
    I/O points, 150
    I/O unit configuration, 135
    I/O while strategy is running, 171
    numeric variable, 266
    PID loop, 161
    pointer table, 270
    pointer variable, 268
    scale of a chart, 61
    strategy download compression, 209
    string table, 268
    string variable, 266
    text color or size, 239, 248
    text in chart, 245
    variable, 265
chart
    adding text to, 245
    auto stepping, 213, 215
    breakpoints, 215
    calling a subroutine, 409
    changing background, 239
    changing name or description, 250
    changing scale, 61
    closing, 249
    commands, 279
    contents, 238
    copying, 249
    creating, 237
    definition, 50
    deleting, 250
    designing, 81
    exporting, 251
    finding a block, 63
    flow-through logic, 84
    guidelines for design, 85
    importing, 252
    loop logic, 84
    looping, 373
    monitoring in watch window, 193
    naming, 86, 250
    opening, 248
    pausing, 212
    printing commands (instructions), 223
    printing graphics, 221
    replacing elements, 230
    saving, 199, 249
    searching, 228
    status, 50, 220
    stepping through, 213
    viewing, 59, 219
    zooming in or out, 61
chart element
    changing color or size, 248
    cutting, copying, and pasting, 247
    moving, 246
    selecting, 246
client, 284
client/server architecture, 291
closing
    chart, 249
    strategy, 199, 200
command
    adding, 270
    adding in OptoScript, 393
    arguments, 83
    changing, 274
    commenting out, 275
    cutting, copying, and pasting, 275, 276
    definition, 51
    deleting, 274
    deleting temporarily, 275
    for continue block, 276
    groups
        analog point commands, 278
        chart commands, 279
        communication commands, 280
        control engine commands, 327

error handling commands, 331
high-density digital module commands, 455
I/O unit—event message commands, 336
I/O unit—memory map commands, 337
I/O unit—scratch pad commands, 338
logical commands, 340
mathematical commands, 342
miscellaneous commands, 345
PID—Ethernet commands, 345
PID—mistic, 349
pointer commands, 355
simulation commands, 357
string commands, 358
time/date commands, 366
timing commands, 367
in OptoScript, 378
function, 380
procedure, 380
printing, 223
comment, 275, 345
in OptoScript, 395, 448
communication
Ethernet link redundancy, 104, 335
I/O memory map commands (instructions), 337
I/O unit—event message commands (instructions), 336
I/O unit—scratch pad commands, 338
incoming, 287
outgoing, 284
overhead, reducing by using event/reactions, 333
PC and control engine, 99
peer-to-peer, 284, 338
TCP/IP, 282
with serial communication module, 284, 288, 290
communication commands, 280
communication handle
adding, 257
definition, 54, 255, 281
examples
File, 303
FTP, 308
serial, 312
TCP, 288, 289
File, 301
FTP, 308
port numbers, 286
serial, 312
serial communication modules, 288
TCP, 282
types, 282
viewing, 266
comparison operators, 386
compiling
copy errors and warnings, 203
strategy, 26, 202
strategy without downloading, 205
subroutine, 407
condition block
definition, 238
condition command, definition, 51
condition in OptoScript, 376
configuration mode toolbar, 56
Configure mode, definition, 55
configuring
continue block, 276
control engine, 99
event/reaction, 162
I/O points, 136
I/O unit, 123
mistic PID loop, 155, 349
PID loop, 150, 347
serial module, 144
subroutine parameters, 403
table variables, 260
variables, 257
connection line
changing color or size, 239, 248
definition, 239
deleting, 247
drawing, 243
moving, 246
selecting, 246
constant, *See* literal
continue block
command (instruction), 276
configuring, 276
definition, 239
control characters, adding to string, 361
control engine
active control engine, 110
archiving strategies, 116
changing definition, 109
changing the active control engine, 110
commands (instructions), 327
configuring, 99
definition, 47
deleting from strategy, 111

download file, creating, 208
hostname, 102
inspecting, 111
IP address, 102
moving to another PC, 119
removing association with strategy, 110
restoring archived strategies, 118
testing communication, 416
viewing message queue, 114
control network
segmenting from company network, 103
control structures, in OptoScript, 388
controller, *See* control engine
converting a strategy, 425
converting IEEE float, 343
copying
block, 247
chart, 249
command (instruction), 275, 276
configured I/O point, 147
connection line, 247
I/O configuration, 125
text block, 247
count variable programming example, 91
counter
commands (instructions), 329
CRC commands, 327
creating
flowchart, 237
mistic PID loop, 155
PID loop, 150
strategy, 197
subroutine, 401
watch window, 193
See also changing
cross reference window, 226
cutting
block, 247
command (instruction), 275, 276
connection line, 247
text block, 247

D

data
IEEE float format, 343
data format
IEEE float, 343
data types in subroutines, 400
data, sorting, in a dialog box, 65
date commands, 366
Debug mode
definition, 55
inspecting control engine, 111
inspecting I/O, 171
debug mode toolbar, 57
debugging
changing speed, 212
choosing level, 211
follow stepping, 212
strategy, 211
subroutines, 411
when using OptoScript, 398
delay
commands, 367
deleting
chart, 250
chart elements, 247
command (instruction), 274
command, temporarily, 275
control engine association with strategy, 110
control engine from strategy, 111
event/reaction, 171
I/O point, 150
I/O unit, 135
PID loop, 161
variable, 265
designing
basic rules for strategies, 84
steps for, 77
dialog box
sizing columns, 64
sorting columns, 65
digital
adding digital point, 136
changing I/O while strategy is running, 173
copying I/O point, 147
counter, 329
high-density module commands, 455
I/O unit commands, 334
latches, 328
MOMO event or reaction, 166
point commands, 378
point, definition, 47
pulse commands (instructions), 330
quadrature counter, 329
totalizer, 329
directory listing, 311
disabling I/O, 330
docking
Strategy Tree, 60
watch window, 195

down timers, 368
downloading
  files using FTP, 308
  files using PAC Terminal, 120
  initialization file for table variable, 264
  strategy, 202, 205
    changing compression, 209
    without using PAC Control, 207
drawing toolbar, 239
drum sequencers, using event/reactions for, 333
duplicate object names, 425

E

editing, *See* changing
EEPROM, *See* flash
elapsed time, 218
email, 315, 323
error
  cannot delete item, 414
  copy compile errors, 203
  queue, 114, 424
  status codes, 425
  TCP/IP cannot connect, 417
  types, 423
  See also troubleshooting
error handling
  commands (instructions), 331
  programming example, 90
error lookup, 423
Ethernet
  connections, 291
  link redundancy, 104
    commands for, 335
  ports, 291
Ethernet I/O units and commands, 233
event/reaction
  adding, 162
  changing, 170
  configuration example, 168
  configuring, 162
  deleting, 171
  groups, 168
  inspecting, 177
  Interrupt chart, 88
  MOMO, 166, 178
example
  case statements, 92, 375
  complex operations, 377
  creating messages to display on screen, 87
  error handling chart, 90
  event/reaction, 168
  File communication handle, 303
  flag lock, 94
  FTP communication handle, 308
  if statements, 376
  Interrupt chart, 88
  loops, 373
  mathematical expressions, 371
  mistic PID loop configuration, 160
  pointers (indexing), 96
  repetitive actions, 91
  serial communication handle, 312
  string building, 363, 371
  string data extraction, 362
  string formats, 361
  string table, 361
  subroutine parameters, 405
  TCP communication handle, 288
  timer, 93
exiting, *See* closing
exporting
  chart, 251
  I/O configuration, 125

F

File communication handle, 301
  examples, 303
file system on the brain
  moving files via FTP, 308
  saving files to flash, 327
finding a block in a chart, 63
flag lock example, 94
flash memory, 172
  saving files to, 327
  saving strategy to, 200
float
  IEEE, 343
floating point
  converting to integer, 345
  converting to string, 365
  definition, 53
  in logic, 341
  variable, definition, 254
flowchart, *See* chart
flow-through logic, 84
follow debugger stepping, 212
for loop, 373, 391
format
  File communication handle, 301
  FTP communication handle, 308

IEEE float, 343
serial communication handle, 313
TCP communication handle, 284
FTP
directory listing, 311
FTP communication handle, 308
full debug, definition, 211
functions in OptoScript, 378

G

gain
and offset, 278
in PID loops, 185
grammar in OptoScript, 449

H

handshake, 284
hardware requirements for PAC Control, 5
help
See also troubleshooting
help, online, 76
hex display mode, 65
high-density digital
commands, 455
counting, 456
enabling commands, 233
latches, 455
high-density points, 336
host task
definition, 52
hostname, control engine, 102
Hungarian notation, 86, 381

I

I/O
changing configured point, 150
configuring, 136
copying configured point, 147
deleting configured point, 150
disabling, 330
exporting configuration, 125
importing configuration, 125
in strategy design, 78, 80
monitoring in watch window, 193
moving configured point, 146
naming points, 87
point numbers, 126, 130
point, definition, 47
saving configuration, 172
segmenting I/O from host network, 103
using in OptoScript, 378
viewing all in a strategy, 224, 226
I/O unit
changing, 135
commands, 334, 336, 337, 338
compatibility chart, 126
configuring, 123
definition, 123
deleting, 135
exporting configuration, 125
importing configuration, 125
mistic, 127
monitoring in watch window, 193
point numbers, 126, 130
serial, 127
IEEE float, 343
if statements
in OptoScript, 389
if/then statements, 376
importing
chart, 252
I/O configuration, 125
info, viewing, 114
initialization file, 120, 261
creating, 262
downloading, 264
input point
definition, 47
disabling, 330
input/output, *See* I/O
inspecting
control engine, 111
control engine using PAC Terminal, 116
event/reactions, 177
I/O, 171, 172
I/O using watch window, 193
mistic PID, 191
PID loop, 179
See also viewing
installing PAC Control, 5
instruction, *See* command
integer
converting to float, 345
definition, 53, 342
in logic, 341
variable, definition, 254
interacting algorithm for PID, 347
Interrupt chart

example, 88
IO Enabler, 332
IP address
  control engine, 102
  loopback, 131, 285, 289
  primary and secondary, 131
ISA algorithm for PID, 347
IVAL, 330

K

keyboard shortcuts, 76

L

lag time, for PID, 180
latch, 455
  commands (instructions), 328
launching, *See* opening
legacy options, enabling, 232
link redundancy, 105, 335
  SoftPAC controller, 105
literal
  constant, 257
  in OptoScript, 446
load last mode at startup, 198
load last strategy at startup, 198
local subroutine parameter, 400
local variable, 400
log file, 434
logic, in charts, 84
logical commands (instructions), 340
logical operator in OptoScript, 387
logical true and false, 341
look up an error, 423
loop logic, 84
loopback address, 131, 285, 289
looping, 373, 390, 391

M

mathematical commands (instructions), 342
mathematical expressions, 371
  in OptoScript, 386
memory, 415
memory map commands, 337
message on screen, programming example, 87
message queue, 114
microSD card, 303, 308
minimal debug, definition, 211
minimum/maximum value, 278, 279
mistic I/O unit, 127
  addressing points, 128, 129
mistic I/O units and commands, 234
mistic PID commands, 349
mode
  Configure, definition of, 55
  Debug, definition of, 55
  Online, definition of, 55
modifying, *See* changing
MOMO, 166, 178
monitoring, *See* inspecting *and* viewing
mounting rack
  B3000 serial, 128
  compatibility, 126
  G4, B100, B200 serial mistic brains, 129
  SNAP-BRS, 128
moving
  block, 246
  connection line, 246
  control engine definitions, 119
  files via FTP, 308
  I/O point, 146
  text block, 246
  to another window or chart, 59
moving control engines, 119
multitasking, 52
  and strings, 360
must on/must off, *See* MOMO

N

naming
  block, 86, 242
  chart, 86, 250
  conventions, 85, 86, 381
  I/O points, 87
  variables, 86, 381
naming conflicts, 425
network
  segmenting control network, 103
network redundancy, 104
number, converting to string, 365
numeric literals in OptoScript, 381
numeric table
  adding, 260
  as alternative to strings, 360
numeric variable
  adding, 257
  in OptoScript, 381

O

offset and gain commands (instructions), 278
online help, 76
Online mode
  avoiding memory problems, 55
  definition, 55
online mode toolbar, 56
opening
  applications from PAC Control, 67
  chart, 248
  strategy, 198
  watch window, 194
operator
  bitwise, in OptoScript, 388
  comparison, in OptoScript, 386
  in standard commands (AND/OR), 273
  logical, in OptoScript, 387
  order of precedence, 447
OptoScript
  bitwise operators, 388
  block example, 370
  block, definition, 238
  bookmark, 395
  case statements, 375, 389
  command types, 380
  commands, 378
  comments, 448
  communication handle, 305
  comparison operators, 386
  comparison with other languages, 439
  complex loops, 373
  conditions, 376
  control structures, 388
  debugging strategies, 398
  definition, 369
  editor, 392
  for loop, 391
  functions, 378
  if statements, 389
  language reference, 439
  literals, 446
  logical operator, 387
  math expressions, 371, 386
  notes for programmers, 444
  numeric literals, 381
  numeric variables, 381
  pointers, 383
  precedence for operators, 447
  repeat loop, 391
  string handling, 371
  strings, 381
  switch statements, 389
  syntax, 379, 445, 449
  tables, 385
  toolbar, 393
  troubleshooting, 396
  types of commands, 380
  when to use, 370
  while loop, 390
OptoVersion utility, 420
order of precedence for operators, 447
output point
  definition, 47
  disabling, 330

P

PAC Control
  definition, 45
  designing a strategy, 77
  directory, list of files in, 434
  errors, 423
  files, list of, 433
  installing, 5
  main window, 54
  mode, 55
  opening other applications, 67
  programming, 77, 369
  system requirements, 5
PAC Display, 52
PAC Manager, 123
PAC Message Viewer utility, 418
PAC Terminal utility
  downloading files, 120
  inspecting control engines, 116
  testing communication with control engine, 416
PAC Utilities, 116, 120, 416, 418, 420
page setup for printing graphics, 221
panning, 62
parallel algorithm for PID, 347
parameters, subroutine, 399, 403
passed-in subroutine parameters, 400
pasting
  block, 247
  command (instruction), 275, 276
  connection line, 247
  text block, 247
pausing
  chart, 212
peer-to-peer communication, 255, 284, 291,

338
permanent storage, *See* flash memory
permissions in Windows, 421
persistent data, 256
persistent RAM, 415
PID loop, 349
  adding mistic, 155
  algorithms, 347
  changing, 161
  configuring, 150, 347
  definition, 346, 349
  deleting, 161
  determining system lag, 180
  inspecting, 179
  inspecting mistic, 191
  mistic, 349
  mistic configuration example, 160
  tuning, 185
  tuning mistic, 191
PID—Ethernet commands, 345
PID—mistic commands, 349
pointer
  adding, 257
  commands, 355, 383
  definition, 54, 255, 356
  in OptoScript, 383
  programming example (indexing), 96
pointer table, 357
  adding, 260
port
  Ethernet, 291
  peer-to-peer communication, 286
Powerup chart, 51
  uses, 85
powerup sequencing, 333
precedence for operators, 447
primary and secondary addresses
  for I/O unit, 132
printing
  bill of materials, 226, 227
  chart commands (instructions), 223
  chart graphics, 221
  chart instructions, 276
  page setup, 221
  subroutine commands, 223
problems, *see* troubleshooting
Product Support, 4
programming
  comparing OptoScript with other languages, 439
  examples, 87
  in PAC Control, 77
pulse
  commands (instructions), 330

Q

quadrature counter commands (instructions), 329
queue
  checking messages, 114, 424
quotation marks in strings, 381

R

rack for I/O modules
  B3000 serial brains, 128
  compatibility, 126
  G4, B100, B200 serial mistic brains, 129
  SNAP-BRS, 128
RAM, 415
redundant communication links, 104, 335
redundant controllers, 109
repeat loop, 373, 391
repetitive actions programming example, 91
replacing elements, 230
resolution, timer, 368
retrieving a directory listing, 311
root certificate, 316
RTS/CTS, 313
running a strategy, 27
  automatically, 210
  manually, 210

S

saving
  chart, 199, 249
  configurations to flash memory, 172
  files to flash memory, 327
  strategy, 199
  strategy to flash, 200
  subroutine, 407
saving I/O unit configuration, 172
scanning event/reactions in a group, 168
scratch pad commands, 338
searching, 228
secondary address, 132
security certificate, 316
Security type, 315
segmenting control network, 103

selecting
    block or text block, 246
    connection line, 246
Send Email
    timeouts, 323, 326
send email, 315, 323
serial communication handle, 312
serial communication module
    communicating with, 284, 288, 290
    configuring, 144
serial devices, communicating with, 312
serial I/O unit, 127
    changing baud rate, 133
    changing baud rate and mode (Pro), 133
server, 284
setpoint, for PID, 150
setting hex display mode, 65
shortcuts, keyboard, 76
Simple I/O units and commands, 233
simulation commands, 357
sizing columns in a dialog box, 64
SNAP PAC redundancy option, 109
SNAP-PAC-S2
    RTS/CTS, 313
SoftPAC
    adding a control engine, 25
    archiving strategies, 116
    compatible I/O units, 6
    file storage, 301
    link redundancy, 105
    saving files to flash, 327
software, launching from PAC Control, 67
sorting columns in a dialog box, 65
SSL (Secure Sockets Layer), 315
starting a strategy, 210
status codes, 425
step into, 213
step out, 213
step over, 213
stepping
    inside blocks, 214
    one block at a time, 213
    through a chart, 213
    within a block, 211
stopping
    strategy, 211
strategy
    archiving
        to computer, 201
        to control engine, 116
    closing, 199, 200
    compiling, 26, 202
    compiling without downloading, 205
    creating, 197
    debugging, 211
    debugging when using OptoScript, 398
    definition, 53, 197
    deleting control engine, 111
    designing, 77, 84
    disassociating control engine, 110
    downloading, 202, 205
    downloading, changing compression, 209
    downloading, without using PAC Control, 207
    files, 433
    including subroutine, 407
    opening, 198
    replacing elements, 230
    restoring archive from control engine, 118
    running, 27
        automatically, 210
        manually, 210
    saving, 199
    saving to flash EEPROM, 200
    searching, 228
    starting automatically, 265
    stopping, 211
    transfer to another PC, 119
    viewing all operands, 226
    viewing variables and I/O, 224
Strategy Tree
    definition, 57
    docking, 60
    icons, 58
string
    adding control characters, 361
    building, example of, 363
    commands, 358, 381
        equivalents in Visual Basic and C, 364
    convert-to-string commands, 365
    data extraction, example of, 362
    definition, 54
    examples, 361, 381
    in OptoScript, 381
    length and width, 360
    quotation marks, 381
    variable, adding, 257
    variable, definition of, 255
string handling, 371
string table
    adding, 260
    example, 361
subroutine

adding commands (instructions), 406
adding local variables, 406
auto stepping, 215
breakpoints, 215
calling from a chart, 409
compiling, 407
configuring parameters, 403
configuring parameters, example, 405
creating, 401
data types, 400
debugging, 411
definition, 399
including in strategy, 407
list of files in, 434
parameters, 399
pausing, 212
printing commands, 223
printing graphics, 221
saving, 407
searching, 228
stepping through, 213
tips, 401
viewing, 411
viewing all operands, 226
viewing variables, 224
Switch & Run, 204
switch statement, 375, 389
switching strategies, 204, 206
sync block
definition, 239
syntax
common errors, in OptoScript, 398
in OptoScript, 379, 445, 449
system requirements for PAC Control, 5

T

table
commands, 335, 345
in OptoScript, 385
numeric table as alternative to strings, 360
table variable, 255
adding, 260
changing, 268, 270
configuring, 260
initializing during strategy download, 261
monitoring in a watch window, 268, 270
tabs, using to view windows, 59
target address, 335
task queue, 280
TCP communication handle, 282
TCP/IP cannot connect error, resolving, 417
text block
deleting, 247
moving, 246
selecting, 246
text, changing color or size, 239, 248
time/date commands (instructions), 366
timeouts
Send Email, 323, 326
timer
commands (instructions), 367, 368
definition, 53
down timer, 368
programming example, 93
resolution, 368
up timer, 368
variable, definition, 255
timing a process, commands used for, 366
TLS (Transport Layer Security), 315
toolbar, 55
drawing, 239
OptoScript, 393
toolbars
configuration mode and online mode, 56
debug mode, 57
totalizer commands (instructions), 279, 329
TPO, 278
transfer control engine, 119
trigonometry commands (instructions), 342
troubleshooting
how to begin, 413
in OptoScript, 396
memory problems, 415
memory problems from Online mode changes, 55
PAC Message Viewer utility, 418
Product Support, 4
Windows permissions, 421
tuning
mistic PID loop, 191
PID loop, 185
types of OptoScript commands, 380

U

UDP communication handle, 292
Ultimate I/O units and commands, 233
up timers, 368

V

variable
  adding, 257
  changing, 265, 266, 268
  configuring, 257
  definition, 53, 253
  deleting, 265
  in strategy design, 81
  literal (constant), 257
  monitoring in a watch window, 193, 266, 268
  naming, 86, 381
  OptoScript compared to other languages, 444
  persistent data in, 256
  pointer, definition, 356
  table, 255
  types of data, 254
  types, in PAC Control, 255
  viewing all in a strategy, 224, 226
velocity algorithm for PID, 347
version, checking with OptoVersion, 420
subroutine
  viewingviewing, *See also* inspecting
viewing
  all operands, 226
  all variables and I/O, 224
  another window or chart, 59
  bill of materials, 226, 227
  chart, 219
  chart instructions, 276
  communication handle variable, 266
  event/reactions, 177
  message queue, 114
  mistic PID, 191
  numeric table variable, 268
  numeric variable, 266
  PID loop, 179
  pointer table variable, 270
  pointer variable, 268
  string table changing
    numeric table, 268
  string variable, 266
  subroutine, 411

W

warning
  copy compile warnings, 203
  viewing, 114
watch window
  creating, 193
  docking, 60, 195
  monitoring pointer variable, 268
  monitoring table variable, 268, 270
  monitoring variable, 266
  opening, 194
while loop, 373, 390
Windows
  permissions, 421

X

XVAL, 330

Z

zooming in or out, 61
z-order, changing, 247