PAC CONTROL USER'S GUIDE

Form 1700-080623—June 2008

OPTO 22

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
Form 1700-080623—June 2008

Copyright © 2008 Opto 22.
All rights reserved.
Printed in the United States of America.

The information in this manual has been checked carefully and is believed to be accurate; however, Opto 22 assumes no responsibility for possible inaccuracies or omissions. Specifications are subject to change without notice.

Opto 22 warrants all of its products to be free from defects in material or workmanship for 30 months from the manufacturing date code. This warranty is limited to the original cost of the unit only and does not cover installation, labor, or any other contingent costs. Opto 22 I/O modules and solid-state relays with date codes of 1/96 or later are guaranteed for life. This lifetime warranty excludes reed relay, SNAP serial communication modules, SNAP PID modules, and modules that contain mechanical contacts or switches. Opto 22 does not warrant any product, components, or parts not manufactured by Opto 22; for these items, the warranty from the original manufacturer applies. These products include, but are not limited to, OptoTerminal-G70, OptoTerminal-G75, and Sony Ericsson GT-48; see the product data sheet for specific warranty information. Refer to Opto 22 form number 1042 for complete warranty information.

Cyrano, Opto 22 FactoryFloor, Optomux, and Pamux are registered trademarks of Opto 22. Generation 4, ioControl, ioDisplay, ioManager, ioProject, ioUtilities, *mistic*, Nvio, Nvio.net Web Portal, OptoConnect, OptoControl, OptoDataLink, OptoDisplay, OptoOPCServer, OptoScript, OptoServer, OptoTerminal, OptoUtilities, PAC Control, PAC Display, PAC Manager, PAC Project, SNAP Ethernet I/O, SNAP I/O, SNAP OEM I/O, SNAP PAC System, SNAP Simple I/O, SNAP Ultimate I/O, and SNAP Wireless LAN I/O are trademarks of Opto 22.

ActiveX, JScript, Microsoft, MS-DOS, VBScript, Visual Basic, Visual C++, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries. Linux is a registered trademark of Linus Torvalds. Unicenter is a registered trademark of Computer Associates International, Inc. ARCNET is a registered trademark of Datapoint Corporation. Modbus is a registered trademark of Schneider Electric. Wiegand is a registered trademark of Sensor Engineering Corporation. Nokia, Nokia M2M Platform, Nokia M2M Gateway Software, and Nokia 31 GSM Connectivity Terminal are trademarks or registered trademarks of Nokia Corporation. Sony is a trademark of Sony Corporation. Ericsson is a trademark of Telefonaktiebolaget LM Ericsson.

All other brand or product names are trademarks or registered trademarks of their respective companies or organizations.

Table of Contents

Chapter 1: Welcome to PAC Control 1

Introduction 1

About this Guide 1

Choosing Documentation 3

Document Conventions 3

Documents and Online Help 4

Product Support 4

Installing PAC Control 5

System Requirements 5

Installation Requirements 5

Compatible Control Engines and I/O Units 6

Important Note on Disk Drives 6

Chapter 2: PAC Control Tutorial 7

Introduction 7

In this Chapter 7

Opening the Strategy 8

Saving the Strategy 9

Examining the Strategy 11

The Strategy Tree 11

Docking the Strategy Tree 12

Opening a Chart 13

Opening a Block 15

Adding a Command 18

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 35
Using a Watch Window 37
Closing the Strategy and Exiting 44
What's Next? 44

Chapter 3: What Is PAC Control? 45

Introduction 45
In this Chapter 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 Special-Purpose I/O Modules 48
Key Features 49
PAC Control Terminology 50
action block 50
analog point 50
blocks 50
condition block 50
continue block 50
digital point 51
external value 51
flowcharts 51
input point 51
instructions (commands) 52
internal value 52
multitasking 52
OptoScript block 53
output point 53
pointer 53
strategy 53

variables 53
PAC Control Main Window 55
Status Bar 55
Mode 55
Toolbars 56
Moving Toolbars 57
Hiding and Showing Toolbars 57
Strategy Tree 58
Windows and Dialog Boxes in PAC Control 59
Using Tabs to View Open Windows 59
Docking Windows 60
Zooming in a Chart or Subroutine Window 61
Redrawing a Chart or Subroutine Window 63
Changing Column Width in a Dialog Box 63
Sorting Data in a Dialog Box 64
Customizing PAC Control for Your Needs 65
Setting Decimal, Binary, or Hex Display Mode 65
Setting Hex String View 65
Customizing Toolbars 66
Choosing Toolbars for Your Screen Resolution 66
Moving Toolbars 67
Moving and Deleting Buttons 68
Creating Your Own Toolbar 69
Setting Up Applications to Launch from PAC Control 70
Online Help 71

Chapter 4: Designing Your Strategy 73

Introduction 73
In this Chapter 73
Steps to Design 73
Solving the Problem 74
Defining the Problem 74
Designing a Logical Sequence of Steps to Solve the Problem 75
Testing the Steps 76
Simple Sprinkler System Process Flowchart 76
Building the Strategy 77
Configuring Hardware 77

Determining and Configuring Variables 77
Creating PAC Control Charts and Adding Instructions 78
Compiling and Debugging the Strategy 81
Using and Improving the Strategy 81
Basic Rules 81
Chart Guidelines 82
Naming Conventions 82
Instruction Examples.... 83
Creating Messages to Display On Screen 84
Error Handling 85
Counting 86
Using a Count Variable for Repetitive Actions 87
Programming Case Statements 88
Using a Timer 89
Using a Flag Lock 90
Flag Lock Timeout Values 91
Pointers and Indexing 93
Without Pointers 93
With Pointers and Indexing 94
Using OptoScript 94
Optimizing Throughput 95
Understanding PAC Control Multitasking 95
Host Task 95
Optimizing PC to Control Engine Throughput 95
Increasing Host Task Frequency 96
Increasing Efficiencies in Your Strategy 96
Ensuring Data Freshness for PAC Display 96
Optimizing Control Engine to I/O Throughput 97
Using I/O Unit Commands 97
Handling I/O Errors Efficiently 97

Chapter 5: Working with Control Engines 99

Introduction 99
In this Chapter 99
Configuring Control Engines 99
Defining a Control Engine on Your PC 100
Control Engine Configuration Dialog Box 102
Associating the Control Engine with Your Strategy 102

Using Network Segmenting in PAC Control 103
Using Ethernet Link Redundancy in PAC Control 104
System Architecture for Ethernet Link Redundancy 104
Ethernet Link Redundancy 105
Ethernet Link, Computer, and Software Redundancy 106
Ethernet Link Redundancy with Serial I/O Units 107
Configuring Ethernet Link Redundancy 108
Using Strategies with Link Redundancy 108
Changing or Deleting a Control Engine 108
Changing a Control Engine's Definition 108
Changing the Control Engine that Receives the Downloaded Strategy 109
Removing a Control Engine's Association with a Strategy 109
Deleting a Control Engine from Your PC 110
Inspecting Control Engines and the Queue 110
Inspecting Control Engines in Debug Mode 110
Viewing the Message Queue 112
Message Queue Information 114
Inspecting Control Engines from the PAC Terminal Utility 115
Downloading Files to the Control Engine 115
Archiving Strategies 115
Archiving to the Control Engine 116
Restoring Archived Strategies from the Control Engine 117
Downloading Files Without Opening PAC Control 119

Chapter 6: Working with I/O 121

Introduction 121
In this Chapter 121
Choosing a Configuration Tool 121
Importing I/O Configuration into PAC Control 123
Copying I/O Configurations 123
Creating the Configuration Export File 123
Importing the Configuration File 124
Adding an I/O Unit 124
Add I/O Unit Dialog Box 125
Changing the Baud Rate for Serial I/O Units 126
Changing the Baud Rate and Mode for Serial I/O Units 127
Tuning the I/O Unit Timeout Value for Ethernet I/O Units 128

Changing Configured I/O Units 129
Deleting Configured I/O Units 129
Adding I/O Points 129
Adding a Digital I/O Point 130
Add Digital Point Dialog Box 132
Adding an Analog I/O Point 133
Add Analog Point Dialog Box 136
Configuring a Serial Module 137
Changing Point Configuration 138
Moving a Configured I/O Point 138
Copying a Configured I/O Point 139
Changing a Configured I/O Point 141
Deleting a Configured I/O Point 141
Configuring PID Loops 142
PIDs and Strategies 142
Adding a PID Loop 143
Add PID Loop Dialog Box 144
Changing a PID Loop 148
Deleting a PID Loop 149
Inspecting I/O in Debug Mode 152
Inspecting I/O Units and Saving Settings to Flash 152
Inspecting Digital I/O Points 153
Inspecting Analog I/O Points 155
Inspecting and Tuning PID Loops 156
Inspecting a PID 156
Determining System Lag 157
Tuning a PID Loop 162
Using Watch Windows for Monitoring 168
Creating a Watch Window 169
Opening an Existing Watch Window 170
Working in Watch Windows 170

Chapter 7: Working with Strategies 173

Introduction 173
In this Chapter 173
Creating a New Strategy 173

Opening a Strategy 174
Opening an Existing Strategy 174
Opening an ioControl Strategy 174
Opening a Recently Used Strategy 174
Loading a Strategy or Mode at Startup 174
Opening Strategies in PAC Control Basic and PAC Control Professional 175
Opening an OptoControl Strategy 175
Saving and Closing 175
Saving the Strategy and All Charts 175
Saving the Strategy and Some Charts 175
Saving the Strategy to a New Name 176
Saving Before Debugging 176
Closing a Strategy 176
Saving a Strategy to Flash 177
Saving to Flash Once 177
Saving to Flash on Every Download 177
Archiving Strategies 178
Archiving to the Computer 178
Archiving to the Control Engine 178
Compiling and Downloading 179
Compiling and Downloading in One Step 179
Switching the Strategy 181
Compiling without Downloading 182
Compiling the Active Chart or Subroutine 182
Compiling Changes Only 182
Compiling the Entire Strategy 182
Downloading Only 182
Background Downloading 183
Downloading Without Using PAC Control 185
Creating the Control Engine Download (.cdf) File 185
Downloading the .cdf File using PAC Terminal 185
Using Command Lines or a DOS Batch File to Perform PAC Terminal Tasks 187
Changing Download Compression 187
Running a Strategy Manually 188
Running a Strategy Automatically (Autorun) 188
Protecting a Running Strategy 189
Stopping a Strategy 189

Debugging 189
Choosing Debug Level 189
Changing Debugger Speed 190
Pausing a Chart or Subroutine 190
Stepping Through a Chart or Subroutine 191
Single Stepping 191
Auto Stepping 193
Setting and Removing Breakpoints 194
Managing Multiple Breakpoints 195
Interpreting Elapsed Times 195
Viewing and Printing 196
Viewing Strategy Filename and Path 197
Viewing an Individual Chart or Subroutine 197
Viewing All Charts in a Strategy 197
Printing Chart or Subroutine Graphics 198
Setting Up the Page 198
Previewing a Flowchart Printout 199
Printing One Chart or Subroutine 200
Printing All Charts in a Strategy 201
Viewing and Printing Strategy or Subroutine Commands 201
Viewing and Printing Strategy or Subroutine Elements 202
Viewing and Printing a Cross Reference 203
Viewing and Printing a Bill of Materials 204
Viewing and Printing I/O Point Mappings 205
Searching and Replacing 207
Searching 207
Replacing 208
Legacy Options 209
Existing Strategies 209
New Strategies 209
Enabling Legacy Options 210
Enable Ethernet, Ultimate, and Simple I/O Units and Commands 210
Enable High-Density Digital Module Commands 210
Enable *mistic* I/O Units and Commands 211

Chapter 8: Working with Flowcharts 213

Introduction 213

In this Chapter 213
Creating a New Chart 213
Working with Chart Elements 214
What's In a Chart? 214
Using the Drawing Toolbar 215
Changing the Appearance of Elements in a Chart Window 215
Configure Chart Properties Dialog Box 216
Changing Existing Elements to Match New Defaults 217
Drawing Blocks 218
Naming Blocks 218
Renaming Blocks 219
Connecting Blocks 219
Action Blocks and OptoScript Blocks 219
Condition Blocks 221
Adding Text 222
Editing Text 222
Selecting Elements 222
Moving Elements 223
Moving Elements in Front of or Behind Other Elements (Changing Z-Order) 223
Cutting, Copying, and Pasting Elements 224
Deleting Elements 224
Changing Element Color and Size 224
Resizing Blocks or Text Blocks 225
Changing Block Colors 225
Changing Text 225
Changing an Element Back to the Defaults 225
Opening, Saving, and Closing Charts 226
Opening a Chart 226
Saving a Chart 226
Closing a Chart 226
Copying, Renaming, and Deleting Charts 226
Copying a Chart 226
Renaming a Chart 227
Deleting a Chart 228
Printing Charts 228
Exporting and Importing Charts 228
Exporting a Chart 228

Importing a Chart 229

Chapter 9: Using Variables and Commands 231

Introduction 231
In this Chapter 231
About Variables 231
Types of Data in a Variable 232
Variables in PAC Control 233
Table Variables 234
Persistent Data 235
Literals 235
Adding Variables 235
Add Variable Dialog Box 236
Adding Tables 239
Adding Table Variables 239
Setting Initial Values in Tables During Strategy Download 240
Creating the Initialization File 240
Text Examples 241
Downloading the Initialization File 242
Changing a Configured Variable 244
Deleting a Variable 244
Viewing Variables in Debug Mode 244
Viewing Numeric, String, and Communication Handle Variables 244
Viewing Pointer Variables 246
Viewing Numeric and String Tables 247
Viewing Pointer Tables 248
Adding Commands 249
Changing a Command 253
Deleting a Command 253
Permanently Deleting a Command 253
Commenting Out a Command 254
Cutting or Copying a Command 254
Pasting a Command 255
Configuring a Continue Block 255
Viewing and Printing Chart Instructions 256

Chapter 10: Programming with Commands 257

Introduction 257
In this Chapter 257
Analog Point Commands 258
Offset and Gain Commands 258
Minimum/Maximum Values 259
Analog Points and OptoScript Code 259
Chart Commands 260
About the Task Queue 260
Communication Commands 261
Communication Handles 261
Using TCP Communication Handles 262
Incoming and Outgoing Communication 262
TCP Communication Handle Examples 264
Using Flowcharts to Control TCP/IP Communication 265
Ethernet Connections and Ports 267
Using the Control Engine's File System 268
Working with Files in Your Strategy 269
Moving Files via FTP 273
FTP Communication Handle Examples 273
Using FTP Communication Handles in Your Strategy 273
Retrieving a Directory Listing 274
Using Serial Communication Handles to Communicate with Serial Devices 276
Serial Communication Handle Examples 276
Using Serial Communication Handles in Your Strategy 277
Control Engine Commands 278
Commands Relating to Permanent Storage 278
Digital Point Commands 279
States, Latches, and Counters 280
Latches 280
Counters 280
Quadrature Counters 281
Totalizers 281
Pulses 281
IVAL and XVAL 282
Simulation and Test: The "Real" Use for XVAL and IVAL 282
Additional Commands to Use with Standard Digital Points 282
Standard Digital Points and OptoScript Code 283

Error Handling Commands 283
I/O Unit Commands 284
Commands for Ethernet Link Redundancy 284
Table Commands 285
I/O Unit—Event Message Commands 286
I/O Unit—Memory Map Commands 287
I/O Unit—Scratch Pad Commands 288
Logical Commands 290
Understanding Logical Commands 290
Logical True and Logical False 291
Mathematical Commands 291
Using Integers 291
Controlling Rounding 292
Using Floats 292
Float Issues and Examples 292
Helpful Links for More Information 294
Mixing and Converting Integers and Floats 294
Miscellaneous Commands 295
Comment Commands 295
PID—Ethernet Commands 296
What is a PID? 296
PID Loops on I/O Units 297
Algorithm Choices (PID) 297
Pointer Commands 299
Understanding Pointers 299
Advantages of Using Pointers 300
Referencing Objects with Pointers 300
Simulation Commands 301
String Commands 301
Using Strings 302
String Length and Width 302
Using Numeric Tables as an Alternative to Strings 303
Strings and Multitasking 303
Adding Control Characters to a String 303
Sample String Variable 304
Sample String Table 304

String Data Extraction Examples 305
Find Substring in String: Example 1 305
Find Substring in String: Example 2 305
String Building Example 305
Move String 305
Append Character to String 306
Append String to String 306
Append Character to String 306
Comparison to Visual Basic and C 307
Convert-to-String Commands 308
ASCII Table 309
Time/Date Commands 310
Timing Commands 310
Delay Commands 311
Using Timers 311
Down Timer Operation 311
Up Timer Operation 312

Chapter 11: Using OptoScript 313

Introduction 313
In this Chapter 313
About OptoScript 314
When To Use OptoScript 314
For Math Expressions 315
For String Handling 316
For Complex Loops 318
For Case Statements 319
For Conditions 320
For Combining Expressions, Operators, and Conditions 321
OptoScript Functions and Commands 322
Standard and OptoScript Commands 322
Using I/O in OptoScript 322
OptoScript Syntax 323
More About Syntax with Commands 324
Variable Name Conventions 325
Using Numeric Literals 326
Making Assignments to Numeric Variables 327

Using Strings 327
Working with Pointers 328
Working with Tables 329
OptoScript Expressions and Operators 330
Using Mathematical Expressions 331
Using Comparison Operators 331
Using Logical Operators 332
Using Bitwise Operators 333
Precedence 333
OptoScript Control Structures 333
If Statements 334
Switch or Case Statements 334
While Loops 335
Repeat Loops 336
For Loops 336
Using the OptoScript Editor 337
Troubleshooting "Unable To Find" Errors 340
Troubleshooting Syntax Errors 341
Debugging Strategies with OptoScript 342

Chapter 12: Using Subroutines 343

Introduction 343
In this Chapter 343
About Subroutines 343
Data Types for Subroutines 344
Creating Subroutines 345
Tips for Subroutines 345
Drawing the Flowchart 345
Configuring Subroutine Parameters 347
Configured Parameters Example 349
Adding Commands and Local Variables 350
Compiling and Saving the Subroutine 351
Using Subroutines 351
Including a Subroutine in a Strategy 351
Adding a Subroutine Instruction 352
Debugging Subroutines 354

Viewing Subroutines 355
Viewing All Subroutines in a Strategy 355
Printing Subroutines 355

Appendix A: Troubleshooting 357

How to Begin Troubleshooting 357
1. Read Any Error Message Box 357
2. Check Communication with the Control Engine 358
3. Check the Message Queue 358
4. Check Status Codes in Your Strategy 358
5. Call Product Support 358
Strategy Problems 358
If You Cannot Delete an Item 358
If You Have Memory Problems 359
Archiving Strategies to Battery-Backed RAM 360
Do You Use Online Mode? 360
Checking Communication with the Control Engine 360
Resolving Communication Problems 361
Matching PAC Control Configuration to the Real World 361
Resolving TCP/IP Cannot Connect Errors (-412) 361
Pinging the Control Engine 361
Other Troubleshooting Tools 363
Checking Detailed Communication Information Using PAC Message Viewer 363
Checking File Versions for Opto 22 Software 365
Problems with Permissions in Windows 2000 366

Appendix B: Errors and Messages 367

Introduction 367
Types of Errors 367
PAC Control Errors 367
Queue Messages 367
Using Queue Messages 368
Status Codes 369
Naming Conflicts 369
List of Common Messages 369

Appendix C: PAC Control Files 375

Introduction 375
Files Related to a Strategy 375
Files Associated with a Subroutine 376
Files in the PAC Control Directory 376

Appendix D: Sample Strategy 377
Introduction 377
Factory Schematic 377
Description of the Process 378
Dough Vessel 378
Chip Hopper 378
Oven 378
Inspection Station 378
Conveyor 378
Emergency Stops 378
Required I/O 379
Analog I/O 379
Digital I/O 379

Appendix E: OptoScript Language Reference 381
Introduction 381
OptoScript Comparison with Standard Programming Languages 381
Function Comparison 382
Variable Comparison 386
Notes to Experienced Programmers 386
Variable Database and Other Surprises 386
PAC Control's Target Audience 386
Language Syntax 387
OptoScript Lexical Reference 388
Token Syntax Legend 388
Literals and Names 388
Keywords (Reserved Words) 389
Operators 389
Comments 390
OptoScript Grammar Syntax Reference 391

Index 397

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 Ethernet link redundancy to controllers and I/O units, supports additional features in SNAP PAC I/O units (such as ramping and pulsing), and offers additional data types in subroutines. PAC Control Professional also provides a migration path for Opto 22 FactoryFloor™ customers by adding support for serial-based *mistic*® I/O units and by including a conversion utility to help customers move older OptoControl™ strategies to PAC Control. PAC Control Professional includes about 500 commands. For information on features supported by a specific controller, see the controller documentation.

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. Features that are available only in PAC Control Professional are marked with .

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

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

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

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

Chapter 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.

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

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

Chapter 5, "Working with Control Engines"—How to configure and communicate with control engines.

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

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

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

Chapter 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.

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

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

Chapter 12, "Using Subroutines"—How to use subroutines to streamline your strategy development.

Appendix A, "Troubleshooting"—Tips for resolving communication problems and other difficulties you may encounter.

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

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

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

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

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

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 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.

form 1For information on what we mean by "legacy" and how to migrate from an older system to a SNAP PAC system, see form 1688, *SNAP PAC System Migration Technical Note*. For information on how to enable legacy functionality in PAC Control for a specific strategy, see "Legacy Options" on page 209.

Document Conventions

The following conventions are used in this document:

- The Pro icon next to text indicates that a feature is available only in PAC Control Professional, not in PAC Control Basic.
- 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 names appear in small capital letters. (Example: "Press SHIFT.")
- 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.

Documents and Online Help

To help you understand and use PAC Control systems, the following resources are provided:

- **Online Help** is available in PAC Control and in most of the utility applications. To open online Help, choose Help→Contents and Index in any screen.
- ***PAC Control User's Guide*** shows how to install and use PAC Control.
- ***PAC Control Command Reference*** contains detailed information about each command (instruction) available in PAC Control.
- A **quick reference card**, located in the front pocket of the *PAC Control Command Reference*, lists all PAC Control commands plus their OptoScript™ code equivalents and arguments.
- ***PAC Manager User's Guide*** and other guides provided with specific hardware help you install, configure, and use controllers and I/O units.

Online versions (Adobe® Acrobat® format) of PAC Control documents are provided on the CD that came with your controller or purchase of Professional software and are also available from the Help menu in PAC Control. To view a document, select Help→Manuals, and then choose a document from the submenu.

When you purchase PAC Control Professional or PAC Project Professional, you also receive a complete set of printed documents.

Resources are also available on the Opto 22 Web site at www.opto22.com. You can conveniently access the Web site using the Help menu in PAC Control. Select Help→Opto 22 on the Web, and then select an online resource from the submenu.

Product Support

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

Phone:	800-TEK-OPTO (835-6786) 951-695-3080 (Hours are 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, be prepared to provide the following information about your system to the Product Support engineer:

- Software and version being used
- Firmware versions
- PC configuration (type of processor, speed, memory, and operating system)
- A complete description of your hardware and operating systems, including:
  - type of power supply
  - types of I/O units installed
  - third-party devices installed (for example, barcode readers)
- Specific error messages seen.

Installing PAC Control

PAC Control installation is easy and quick. Insert the CD containing PAC Control in your CD-ROM drive, and the installation wizard should appear. If the wizard does not appear, start Windows Explorer and navigate to your CD-ROM drive. Double-click Setup.exe to begin installation.

NOTE: If you run PAC Project applications in Microsoft Windows XP, make sure to use the Windows Classic theme. Otherwise, a Microsoft bug with how themes are handled may cause the system to crash.

If you have trouble installing PAC Control, contact Opto 22 Product Support at 800-835-6786 or 951-695-3080.

System Requirements

Installation Requirements

Here's what you need to install and run PAC Control:

- A computer with at least the minimum processor required for your version of Microsoft Windows (1 GHz Pentium®-class or better recommended) and Ethernet capability
- VGA or higher resolution monitor (Super VGA recommended). Minimum size: 800x600 with small fonts.
- Mouse or other pointing device
- Installed Windows printer (optional)
- Microsoft Windows XP (with Service Pack 2) or Windows 2000® (with Service Pack 4) workstation operating system. Microsoft Windows server and 64-bit versions of Windows workstation operating systems are not supported.
- At least 512 MB RAM

  If you are using PAC Display, please note its requirements in the *PAC Display User's Guide*.

- Available hard disk space: at least 29 MB for PAC Control Basic or at least 30 MB for PAC Control Professional
- Compatible control engine and I/O unit(s), as shown in the following section.

Compatible Control Engines and I/O Units

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

Using this control engine	PAC Control Basic supports these I/O units	PAC Control Professional supports these I/O units
SNAP PAC S-series controller	SNAP-PAC-R1 SNAP-PAC-R2 SNAP-PAC-EB1 SNAP-PAC-EB2 SNAP-PAC-SB1 SNAP-PAC-SB2	SNAP-PAC-R1 SNAP-PAC-R2 SNAP-PAC-EB1 SNAP-PAC-EB2 SNAP-PAC-SB1 SNAP-PAC-SB2
SNAP PAC R-series controller	SNAP-PAC-R1 SNAP-PAC-R2 SNAP-PAC-EB1 SNAP-PAC-EB2	SNAP-PAC-R1 SNAP-PAC-R2 SNAP-PAC-EB1 SNAP-PAC-EB2

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 floppy disks, key chain USB drives, and memory cards, their use is not recommended. They are better suited for transferring files rather than directly accessing them.

PAC Control Tutorial

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. (PAC Control Basic is shown in the graphics.)

The best way to use the tutorial is to sit down at your computer and follow it through. All you need is PAC Control and access to a control engine. Even if you can't access a control engine at the moment, you can still do everything up to the point of downloading your strategy.

In this Chapter

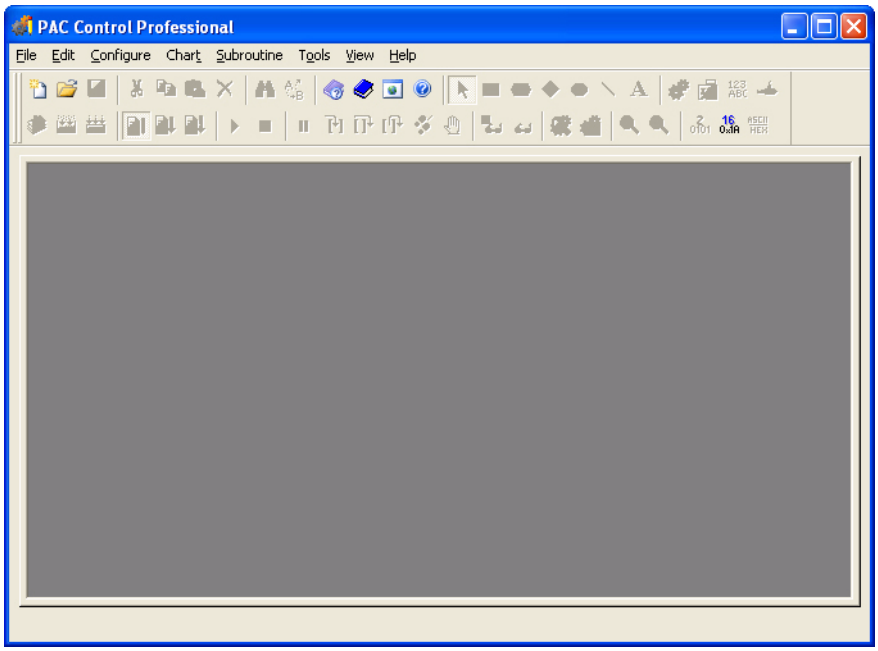
Opening the Strategy 8
Saving the Strategy 9
Examining the Strategy 11
Opening a Chart 13
Opening a Block 15
Adding a Command 18
Configuring a Control Engine 23
Compiling the Strategy 26
Running the Strategy 27
Compiling and Downloading the Change 35
Using a Watch Window 37
Closing the Strategy and Exiting 44
What's Next? 44

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 377, but for now, let's just open and explore it.

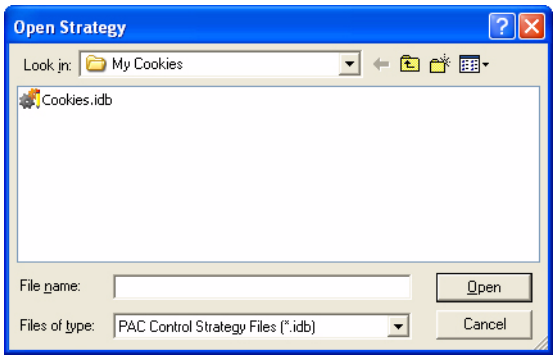
1. Start PAC Control by clicking the Start button and selecting Programs→Opto 22→PAC Project→PAC Control.

   The PAC Control main window opens.



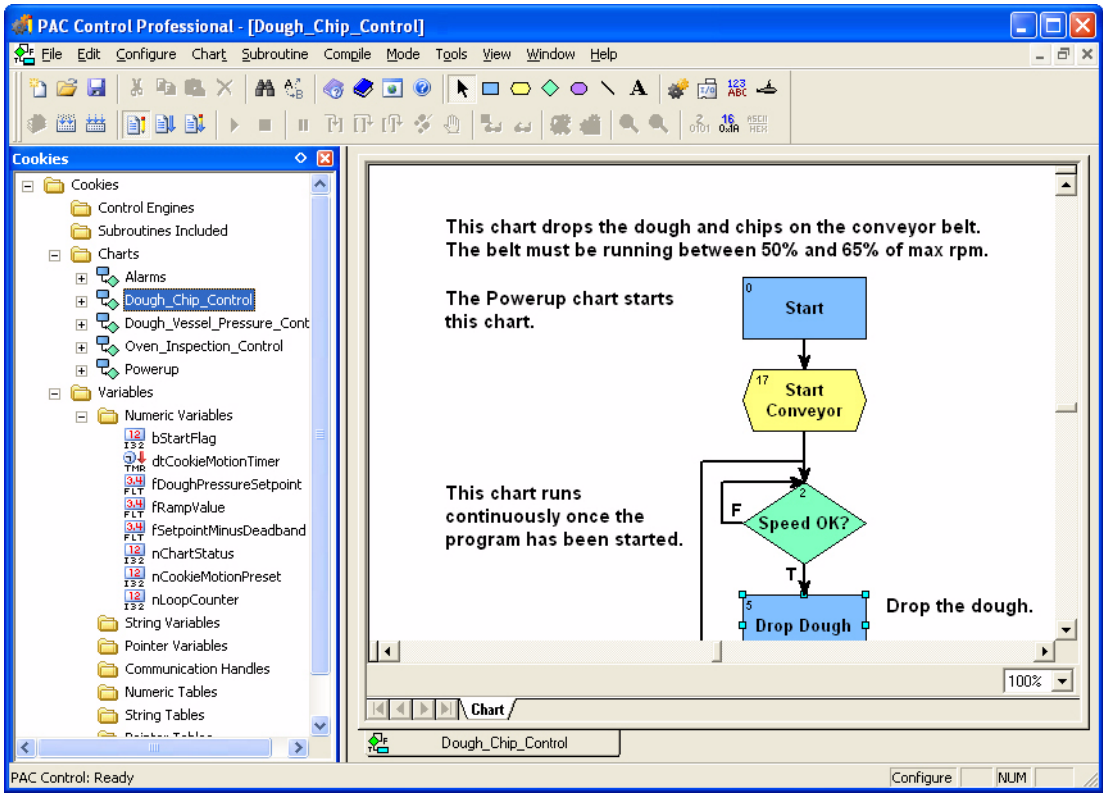
2. Click the Open Strategy button on the toolbar, or choose File→Open Strategy.
3. In the Open Strategy dialog box, navigate to PAC Project\Control Pro (or Basic) Examples.
4. In the Examples directory, double-click the ioCookies subdirectory to open it.

   The strategy file Cookies.idb appears.



5. Double-click the Cookies.idb file to open it.

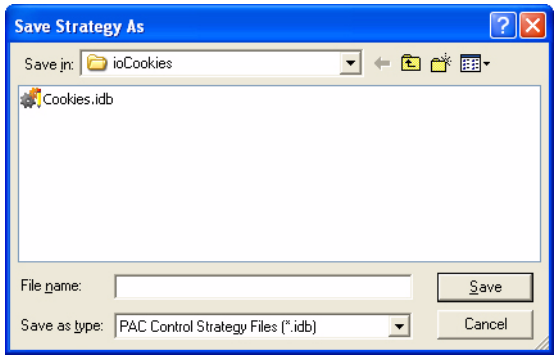
The Cookies strategy opens and the PAC Control window now shows the Cookies strategy (yours may look somewhat different).



Saving the Strategy

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

1. Select File→Save Strategy As.

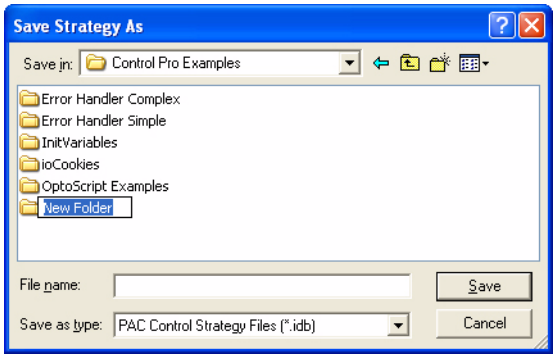


Since each PAC Control strategy must be located in its own directory, we cannot save the strategy to a new name in its current location.

2. Click the Up One Level button to move up to the Examples directory.

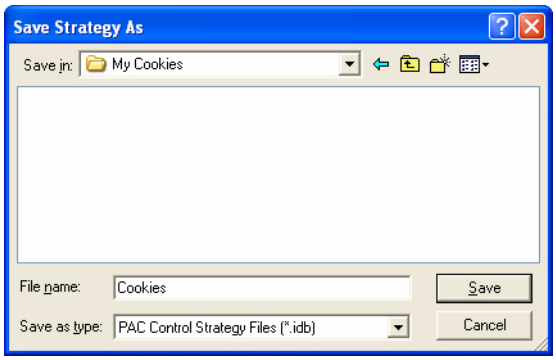
3. Click the Create New Folder button .

   The new folder appears in the list.



4. Type `My Cookies` to replace New Folder. Double-click the folder to open it.
5. Click in the File name field and type `Cookies`.

   The dialog box now looks like this:



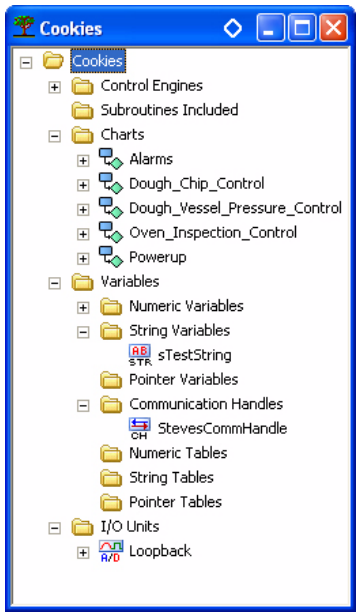
6. Click Save.

   The strategy is saved as Cookies in the My Cookies directory.

Examining the Strategy

Briefly, our cookie factory includes a tank of pre-mixed cookie dough, a tank of chocolate chips, an oven, a visual inspection station, a conveyor belt, and some 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 alarms built in to stop the process when necessary.

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



The Strategy Tree

As with any window in PAC Control, you can move the Strategy Tree window by clicking and dragging the title bar, you can minimize, maximize, or dock it by clicking buttons at the right of the title bar, or you can reshape it by dragging any edge in any direction.

However, the Strategy Tree window is unique in that it must remain open, since closing it is equivalent to closing the strategy.

The Strategy Tree works like Windows Explorer: you can expand and collapse folders to show or hide what is in them. A quick look at the tree reveals that our 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 also provides shortcuts to many common PAC Control activities, for example, opening flowcharts.

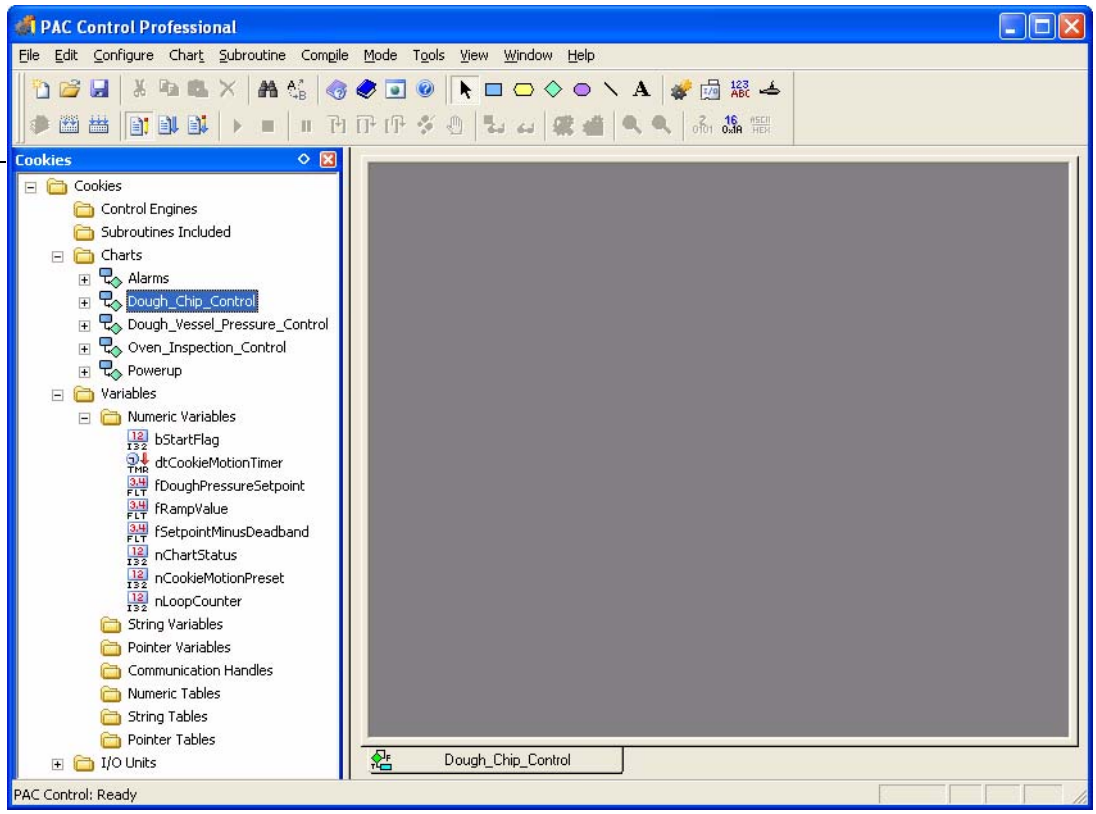
Docking the Strategy Tree

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 always visible, you can dock it in a separate frame.

1. Click the docking icon 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



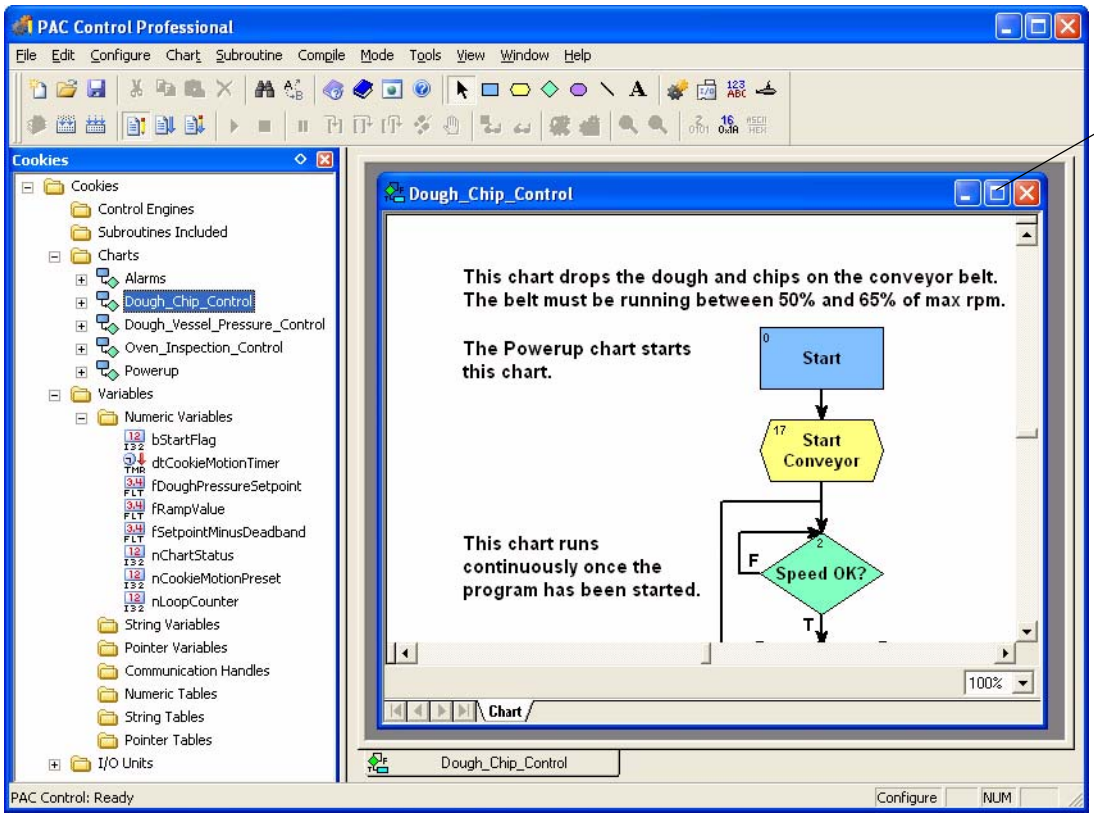
2. To change the width of the Strategy Tree's frame, move your mouse over the right side of the frame. When the cursor changes, click and drag the side to make the frame wider or narrower.

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.

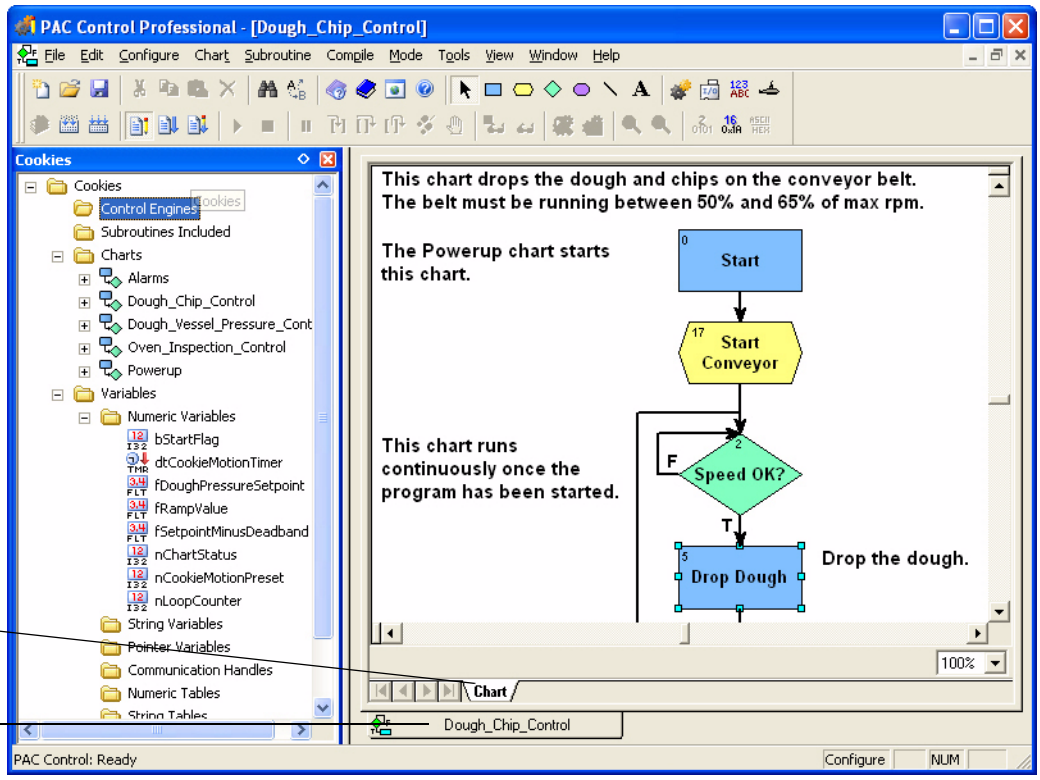
1. Double-click the Dough_Chip_Control chart name on the Strategy Tree. (You can also open a chart by selecting Chart→Open and navigating to the chart name.)

   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.

   The chart now covers the whole frame. Notice 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.



White tab shows this is a chart.

Gray tab shows chart name.

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 is not 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.

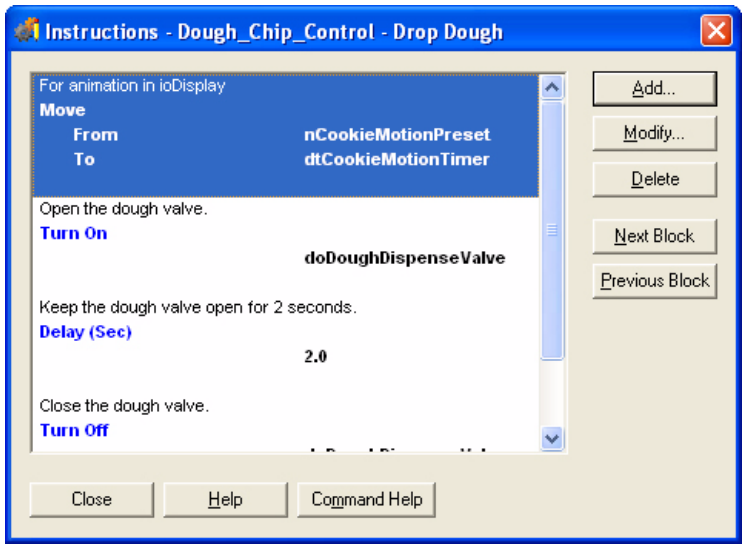
Connections link the blocks together and show how the program logic flows. Action blocks exit through just one connection, since they always go in one direction. Condition blocks exit through two connections, one for a true evaluation and the other for a false evaluation.

Opening a Block

Let's see what's in a block.

1. Double-click the Drop Dough block.

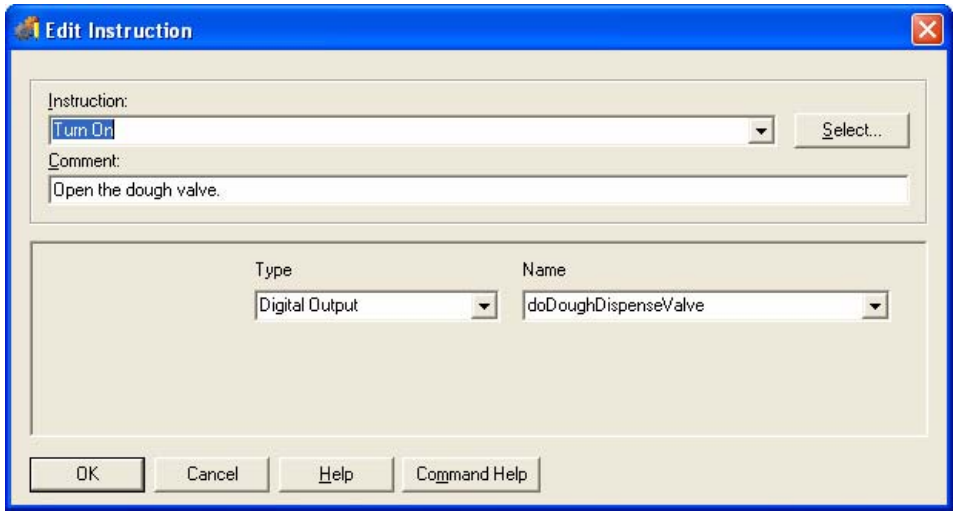
   The Instructions dialog box appears.



   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.



   Here we see the details of the command, which simply turns on the digital output Dough_Dispense_Valve. In other words, it opens the valve.

3. Close the Edit Instruction dialog box by clicking OK or Cancel.
4. Back in the Drop Dough block's Instructions dialog box, move your cursor to the bottom right edge of the dialog box.

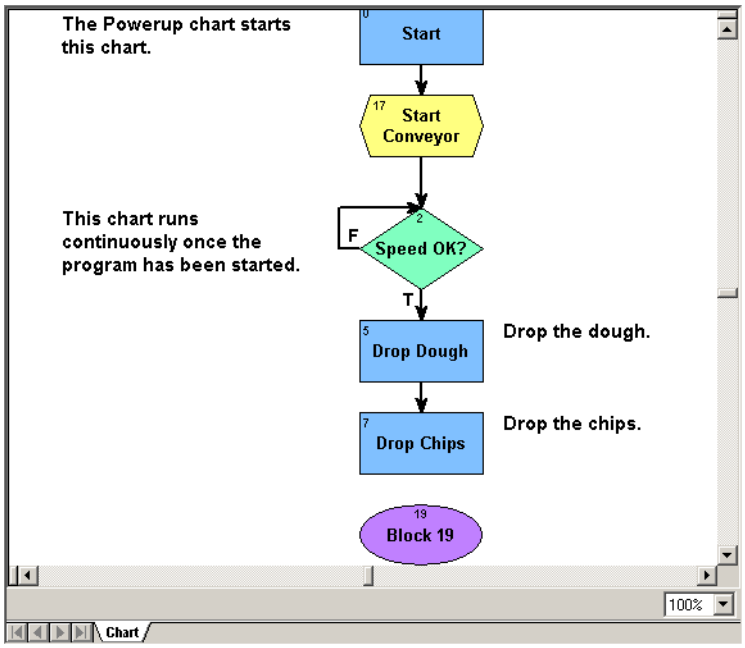
   When your cursor changes to a bidirectional arrow, you can resize this dialog box by clicking and dragging any edge.
5. Close the Instructions - Dough_Chip_Control - Drop Dough dialog box by clicking Close or by pressing ESC.

   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.
6. Select the connection line by clicking it at a bend or junction point, and delete it by pressing DELETE. Click the down arrow at the bottom right of the chart window once to scroll down a little.
7. Now click the continue block tool button in the toolbar.

   When you bring the mouse back into the chart area, an oval outline appears, representing the new continue block.
8. Position the oval about half an inch below the Drop Chips block, and click your mouse button once.

   A continue block appears with the default name Block 19. (The number on yours may be different.) If you move the mouse again, a new oval outline follows. Deactivate the continue block tool by clicking the right mouse button or by pressing ESC.

   Your screen should now look like this:



9. Connect the Drop Chips block to this new block by clicking the connection tool ＼ in the toolbar. Click once in the block the connection is coming from, Drop Chips.

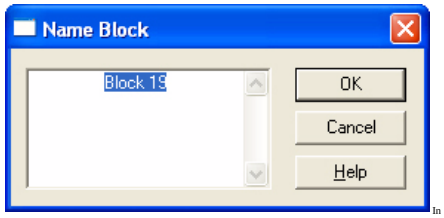
If you move your cursor around the screen, you see a connection following your movements. If you move the cursor to the top of Block 19, the connection becomes a short line from the bottom center of Drop Chips to the top center of Block 19.

10. Click inside the upper part of Block 19 to complete the connection. Click the right mouse button or press ESC to release the tool, returning your cursor to an arrow.

Let's name Block 19 something more descriptive.

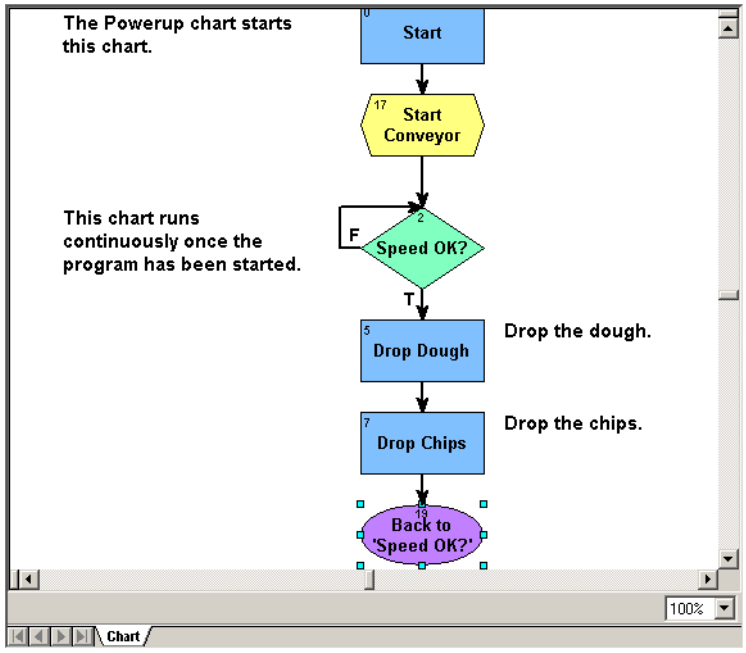
11. Click Block 19 once to select it, then right-click it and select Name from its pop-up menu.

The Name Block dialog box appears.



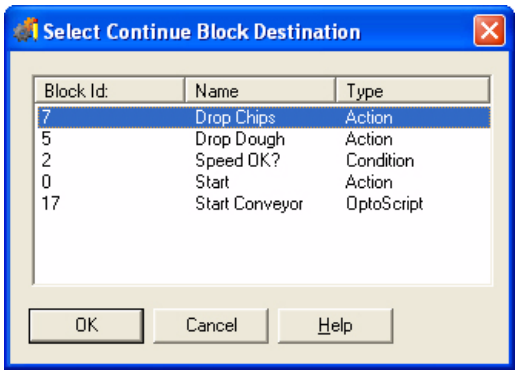
12. Type `Back to 'Speed OK?'` right over the selected text, then click OK.

The chart now looks like this:



Now let's give the continue block its instructions.

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



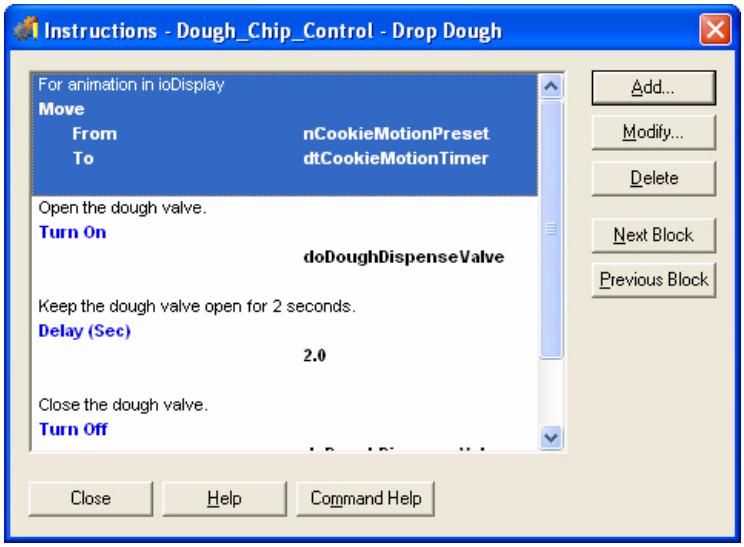
14. Select the Speed OK? block by clicking it once, and then click OK.

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, or instruction, to a block in the Dough_Chip_Control chart. We'll add the command so we can keep track of the number of cookies we produce.

1. Double-click the Drop Dough block.

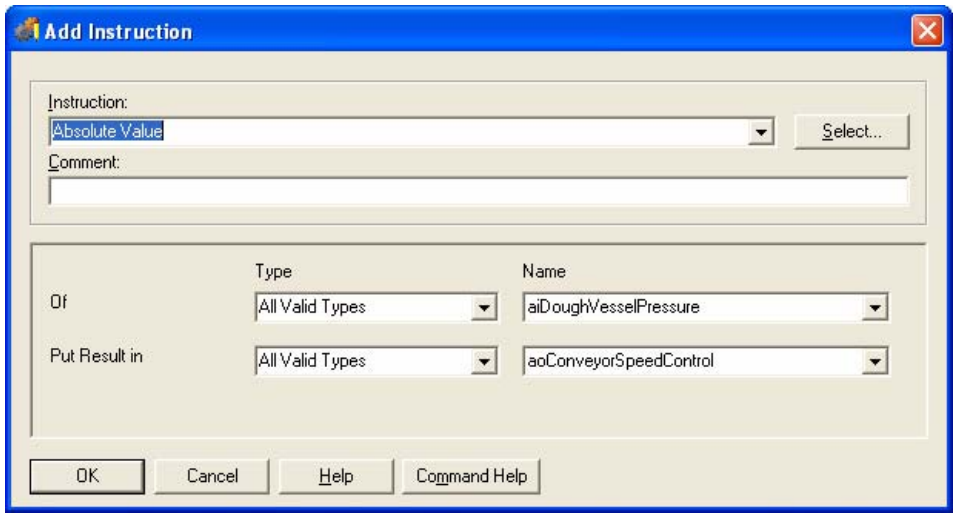


We'll add the new instruction (command) between the Turn On and Delay (Sec) commands.

2. Click anywhere on Delay (Sec) to highlight this command.

This highlight indicates the position where the next command is added.

3. Click Add to open the Add Instruction dialog box.

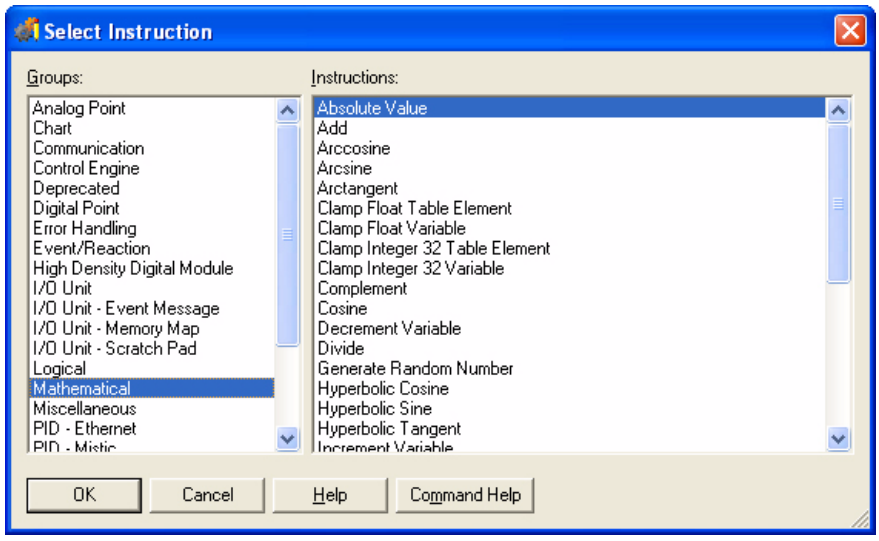


If you knew the exact name of the command to enter, you could type it over Absolute Value (which is first in the alphabetical list of commands). As you typed, the first command that matched the pattern you entered would be filled in automatically. For example, if you really wanted to enter Absolute Value, you would start typing `abs`.

Another way to add a command is to click the down arrow just to the left of the Select button. You could scroll through the resulting list of commands to find the right one.

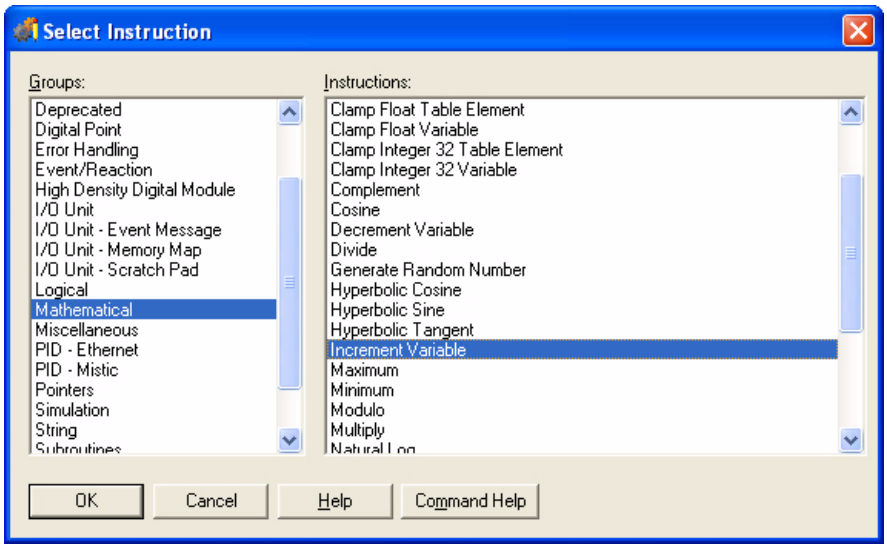
The third way to enter a command is the one we'll use here.

4. Click Select.



All PAC Control command groups are listed on the left, and commands in the highlighted group are listed on the right. The command we want has something to do with increasing a counter. Counting sounds like math, so let's try the Mathematical group.

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



Notice that if you need information about any command, you can click the Command Help button.

6. Click OK and this command is entered in the Add Instruction dialog box.

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

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

8. Next, click the arrow in the Type field, the one that currently reads All Valid Types.

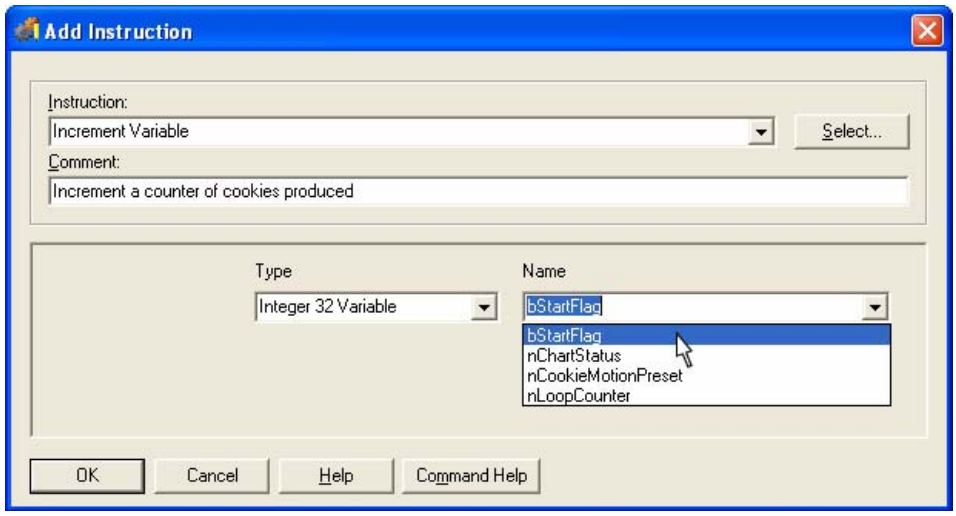
   This list shows what those valid types are: a float variable, an integer 32 variable, and an integer 64 variable.

9. Counters are integers, so select Integer 32 Variable.

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

10. Click the arrow in the Name field, which currently reads bStartFlag.

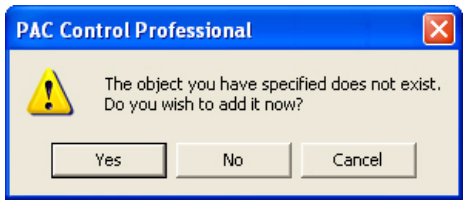
    The drop-down list shows all variables currently configured as integer 32 variables:



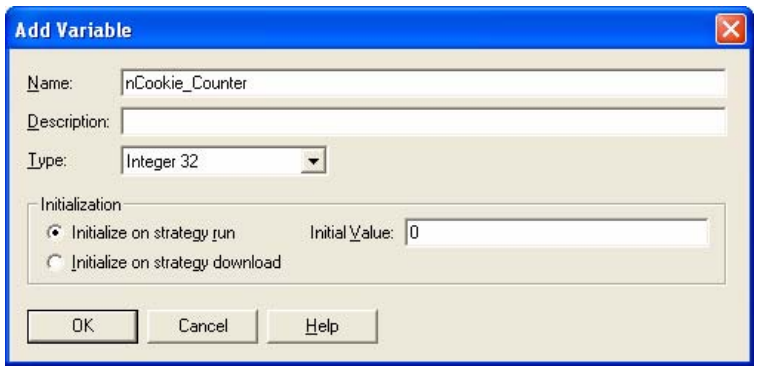
nCookie_Counter is not in the list, because we never needed it before, so we never created it. We'll create it now using what we call on-the-fly configuration.

11. Highlight bStartFlag and type the new variable name, nCookie_Counter, right over it.

As soon as you try to do something else, such as click OK to close the dialog box, the following message appears.



12. Click Yes.

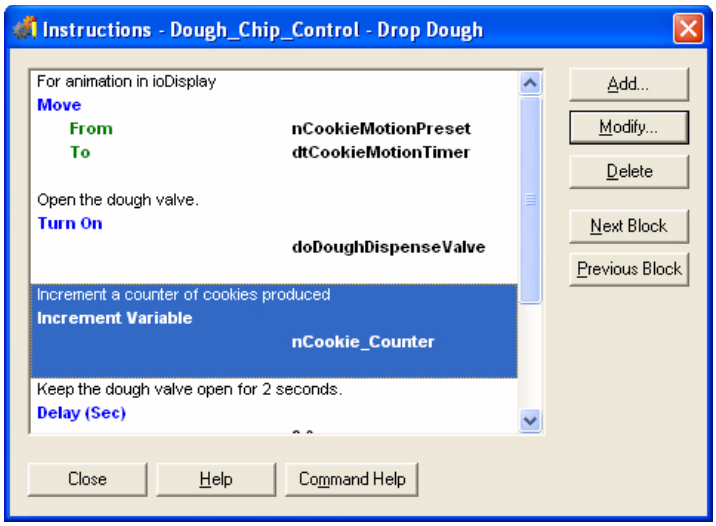


Notice that the name (nCookie_Counter) and type (Integer 32) have already been filled in.

13. Add a description, if you wish. Leave the initial value at zero, which is the default. Then click OK to accept the data and close the dialog box.

14. In the Add Instruction dialog box, click OK.

The new instruction appears in the Instructions window for Drop Dough.



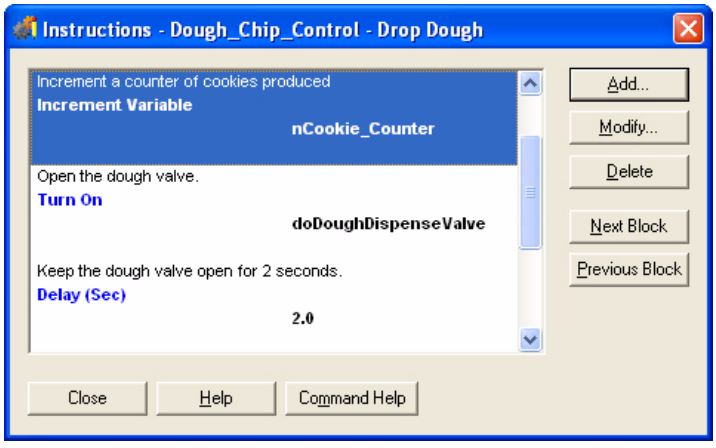
But maybe it makes more sense to start the counter at the beginning of the process, rather than in the middle, after some dough has already been dropped.

15. With the Increment Variable command highlighted, press the right mouse button and select Cut from the pop-up menu.

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

16. Now click Turn On to highlight it. Click the right mouse button and select Paste from the pop-up menu.

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



17. Click Close to return to the chart.

You've just added a cookie counter. On the Strategy Tree, open the Numeric Variables folder in the Variables folder. The new numeric variable is there.

18. Save the changes you've made by clicking the Save Strategy button on the toolbar. Click OK to save.

    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

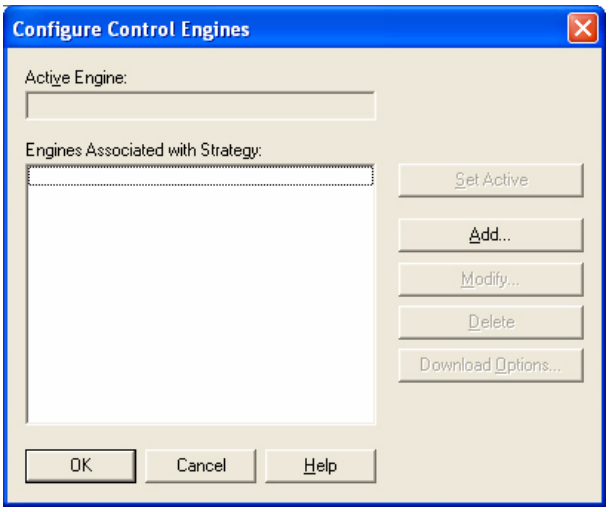
Up to this point, we've been able to play around in PAC Control without hardware. Now it's time to configure a control engine.

1. If you have a PAC Control-compatible control engine 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 control engines and I/O units, see page 6. For instructions to load firmware, see Opto 22 form 1704, the *PAC Manager User's Guide*.

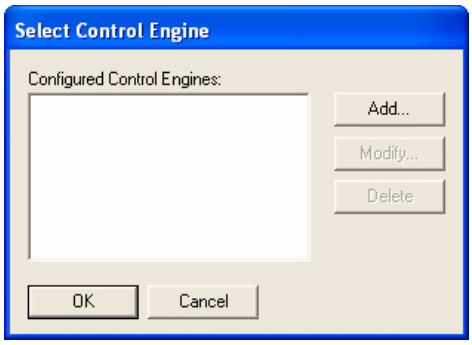
2. Turn the I/O unit off and then back on again.
3. Double-click the Control Engines folder on the Strategy Tree, or click the Configure Control Engine button on the toolbar.

   The Configure Control Engines dialog box appears.



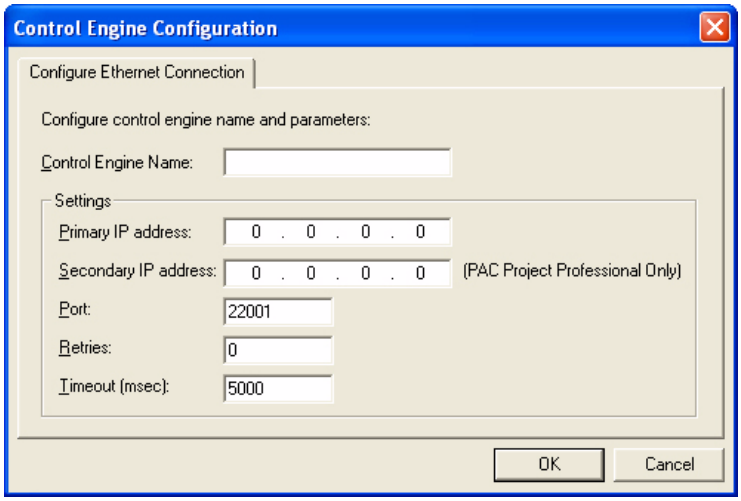
   Since we haven't configured a control engine yet, there are no control engines in the list.

4. Click Add.



5. Click Add again to add a control engine.

   The Control Engine Configuration dialog box appears.



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.

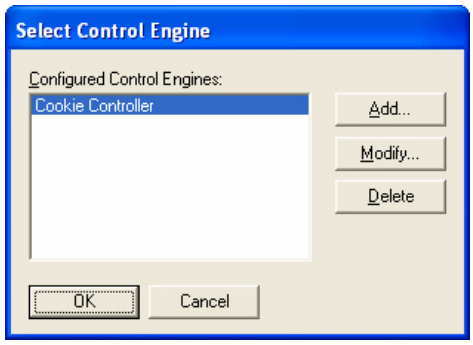
7. Enter the control engine's IP address.

   On a hardware control engine such as a SNAP PAC controller or brain, the IP address assigned to the device 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 Opto 22 form 1704, the *PAC Manager User's Guide,* for configuration instructions.

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.

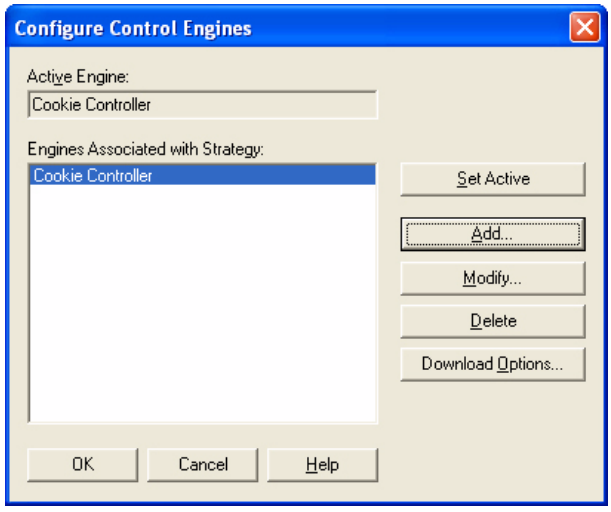
8. Make sure that you have not changed the values in the Port, Retries, and Timeout fields, and then click OK.

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



9. 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.

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

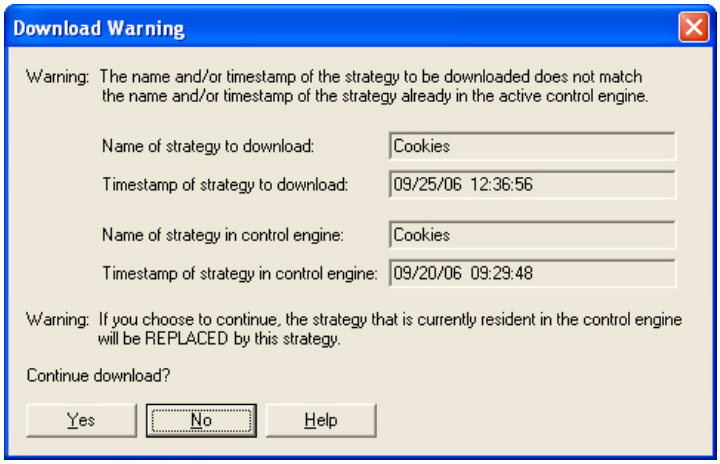
On the Strategy Tree, the new control engine appears as the first entry in the Control Engines folder.

Compiling the Strategy

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

1. Click the Debug mode button 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.

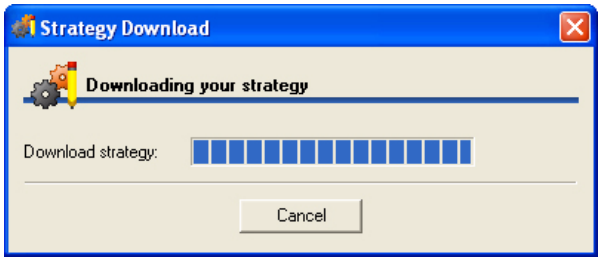
   You may see a Download Warning message like this one:



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

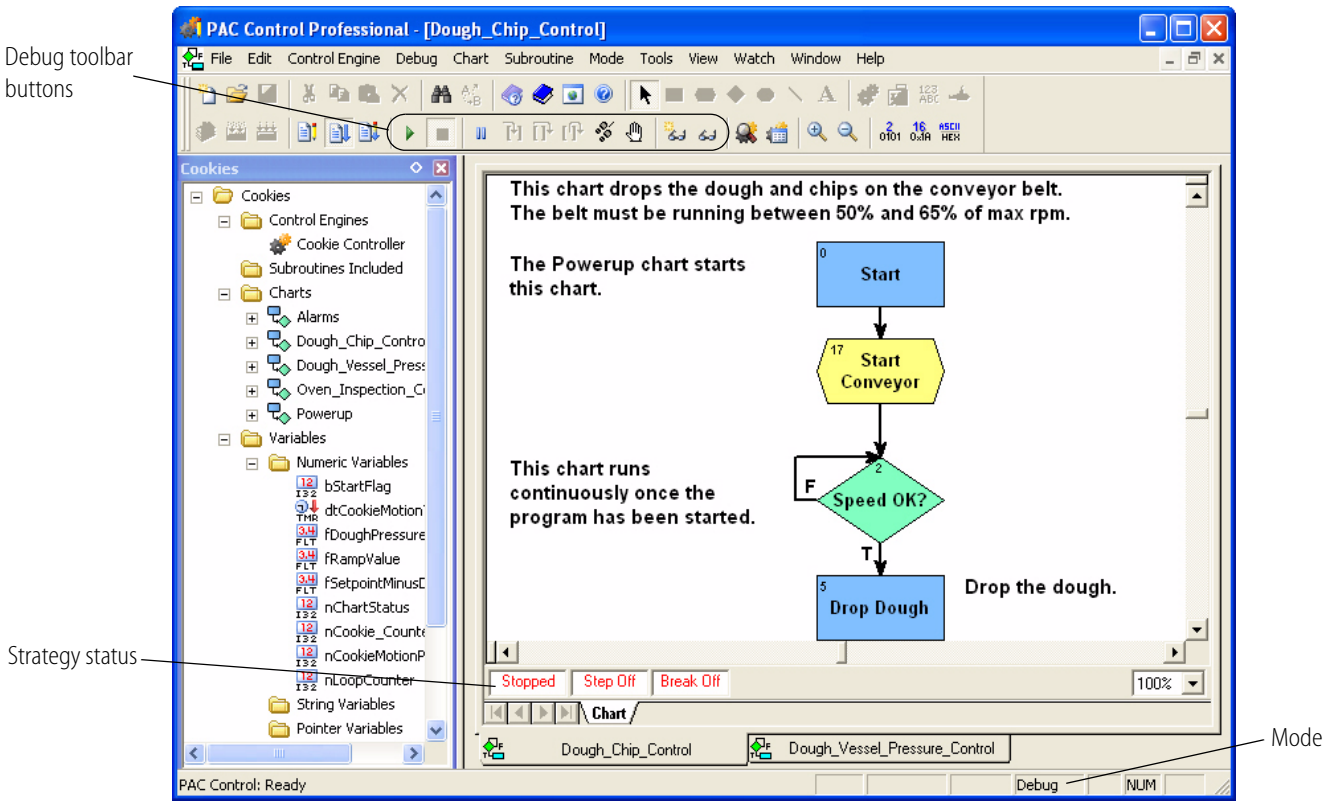
4. Click Yes to proceed.

   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:

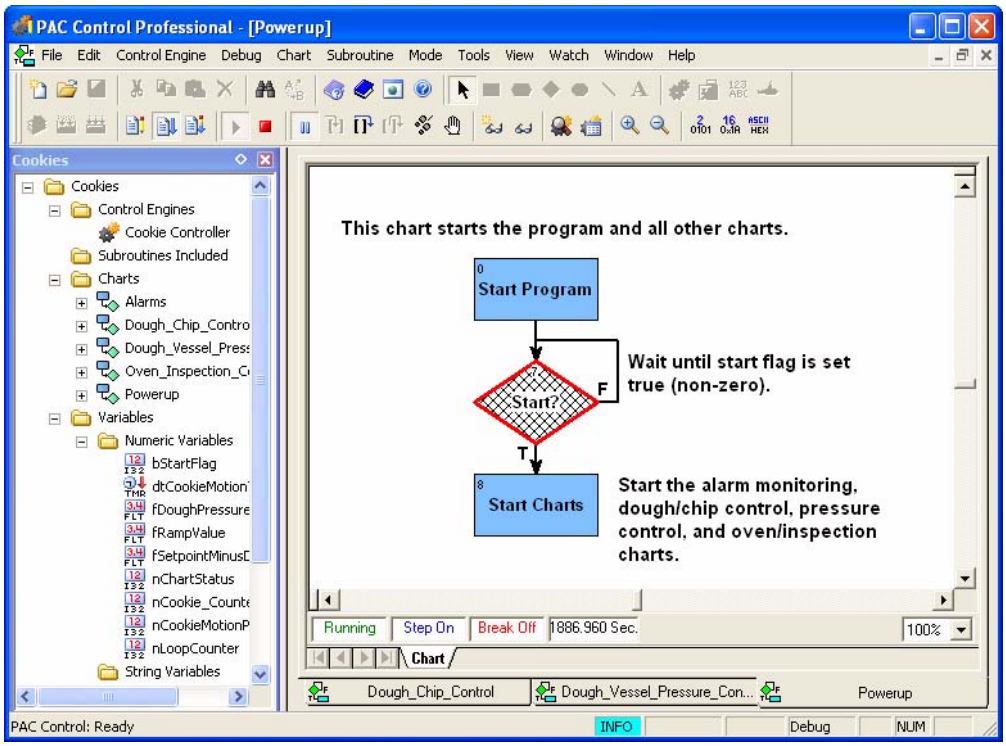


Running the Strategy

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 button .

   At the bottom of the chart window, the word Stopped changes to Running. Let's try pausing the program to see where we are.
3. Click the Pause button .

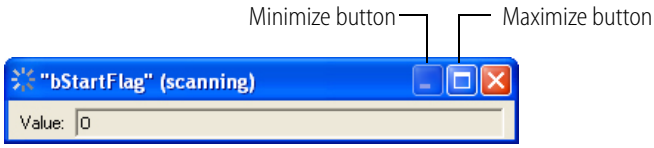


The hatch marks on the Start? block indicate that this command block was about to be executed when we clicked Pause. Apparently the program isn't getting past this block. Notice that the False exit routes right back to the top of the Start? block, while the True exit moves on to Start Charts. We can 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.

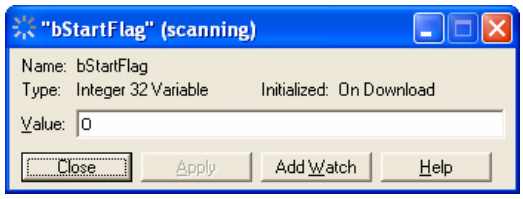
4. Double-click the bStartFlag variable on the Strategy Tree.

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.

5. 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.

6. 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.

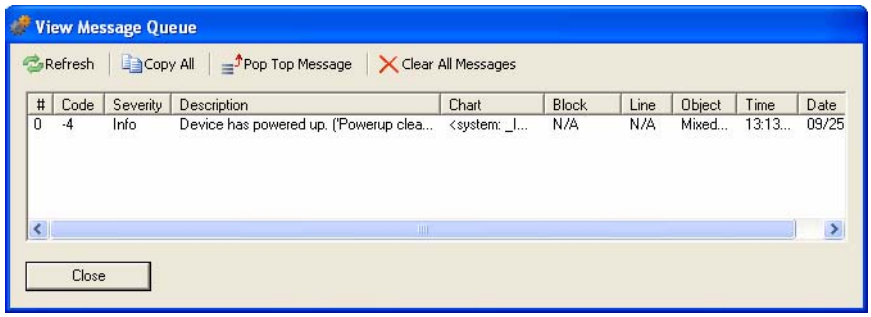
7. Click Apply to implement the change. Click the Minimize button in the bStartFlag View Variable dialog box and move it out of the way. Click the Powerup tab to make the Powerup chart the active window.

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 to view the View Message Queue dialog box.



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.

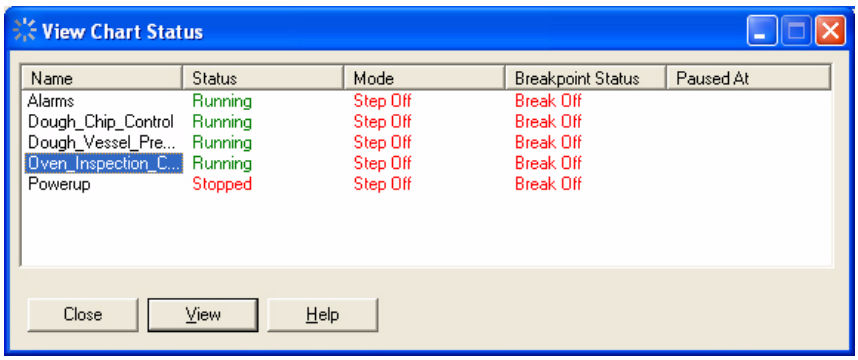
   In the View Variable dialog box, the bStartFlag value reverts to zero, because the Start Charts block set the flag back to zero.

3. Close the bStartFlag View Variable dialog box. In the Powerup chart window, click the Pause 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.



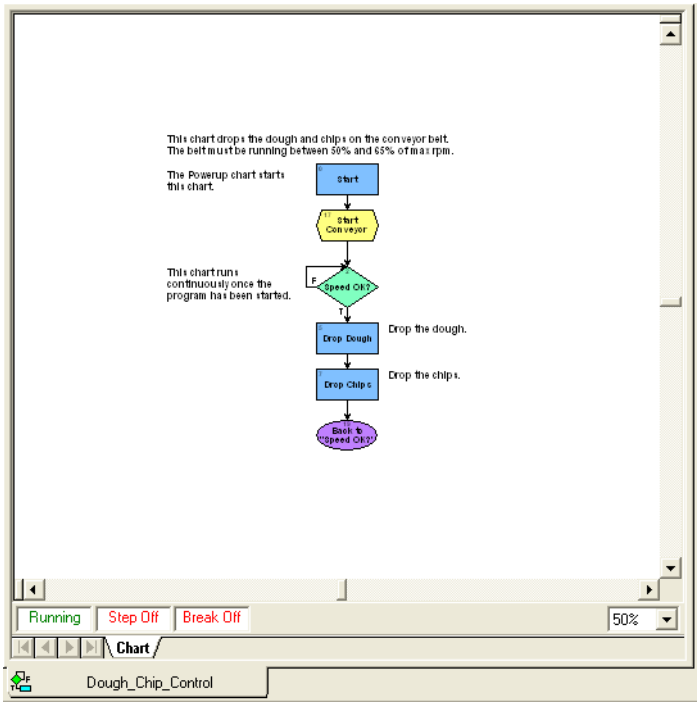
   This dialog box shows us all the charts in our strategy. As you can see, four of the charts are running and one (Powerup) is stopped.

   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 click the Zoom Out button on the toolbar to view the window at a smaller percentage. 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:



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 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 button is clicked, the chart status indicates Step On.

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

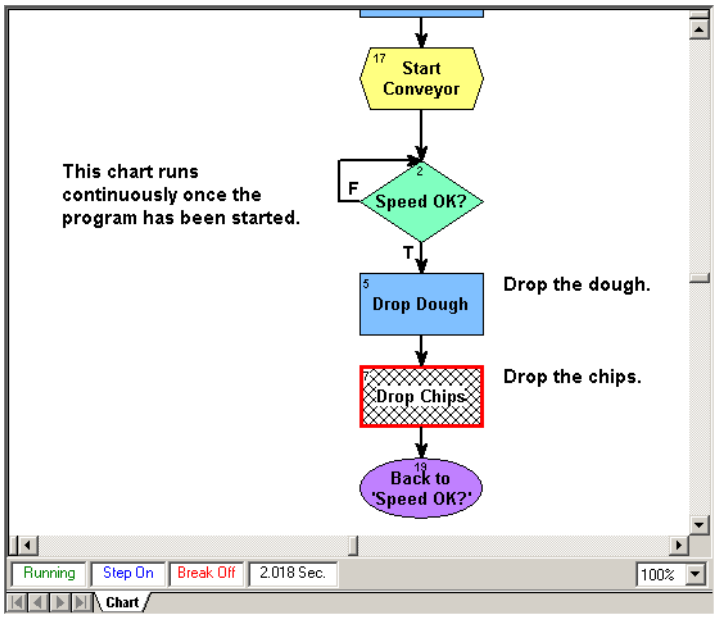
Step Off now appears in the chart status bar.

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:



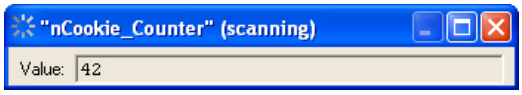
Notice that in the PAC Control toolbar, the Run and Auto Step Chart buttons are depressed. The chart status bar shows us the chart is running and in Step Auto mode. The time indicator to the right of Break Off shows the time it took for the most recent block to execute.

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.



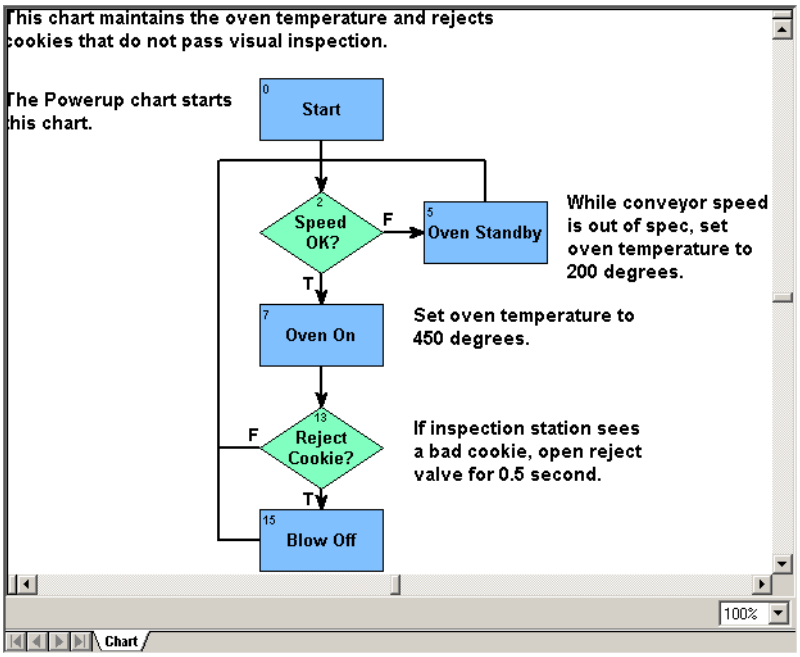
   The Value field should increase every time a cookie is produced, adding to the total number of cookies produced since the strategy run began. The nCookie_Counter above shows this figure as 42. (Yours may be different.)

   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.

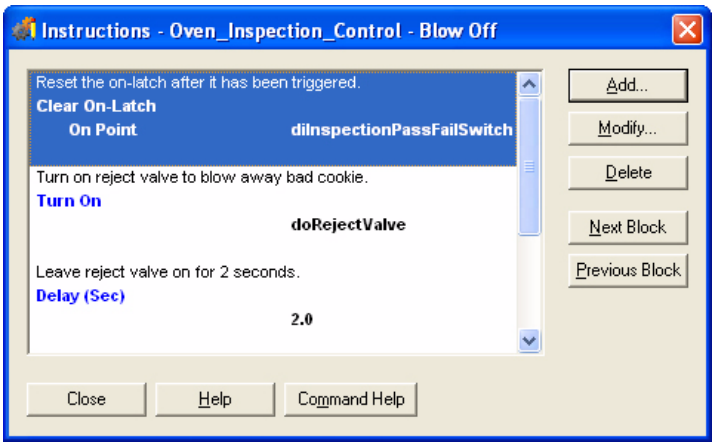
5. Double-click the Oven_Inspection_Control chart on the Strategy Tree to open it.

   The chart window looks something like this:



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:

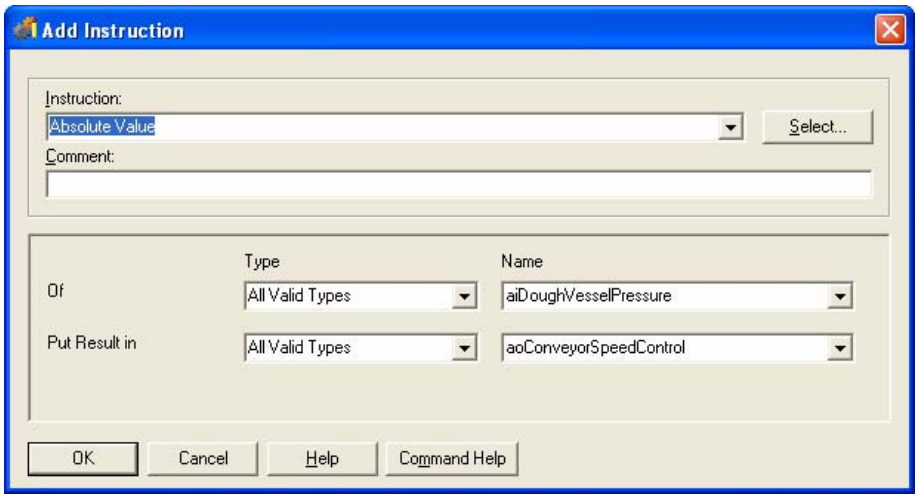


This command block is executed only when a bad cookie has been found. This block first resets the on-latch triggered by the bad cookie, so that the next cookie won't be marked bad, too. The block then turns on the reject valve. The valve stays on for two seconds before being shut off. Let's decrement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



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.

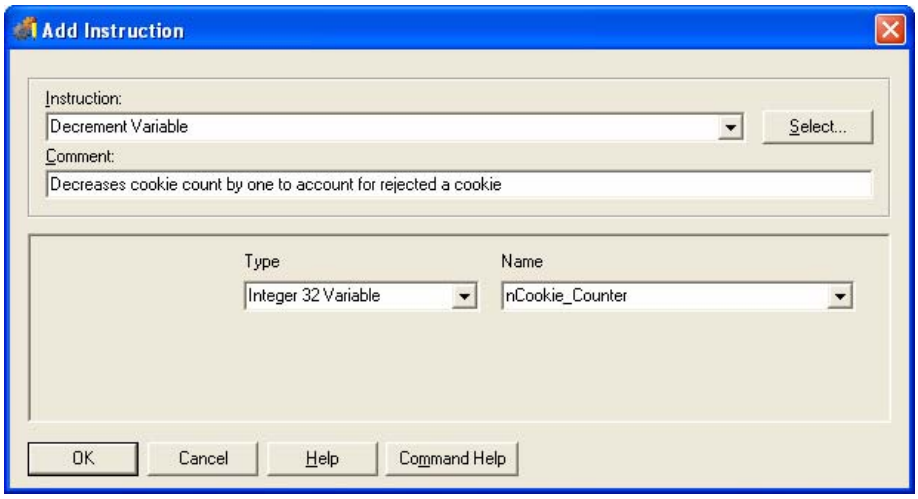
9. Type `dec` in the Instruction field.

   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. Type in 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.

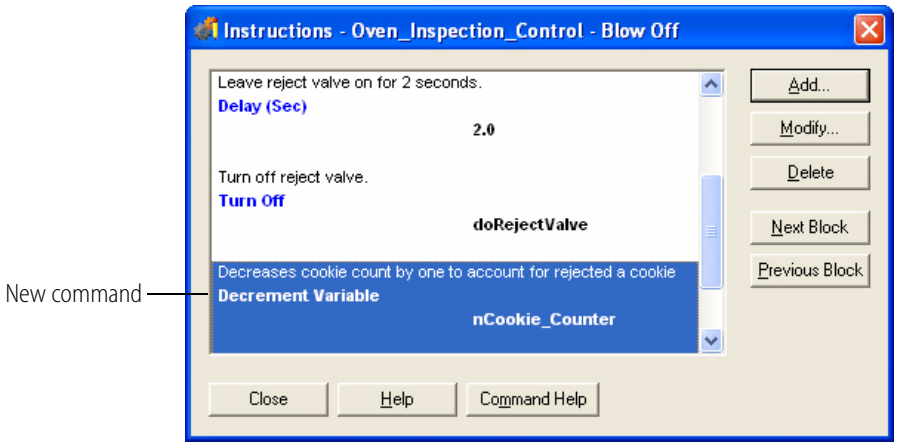
    An outlined button means that pressing the space bar or ENTER is equivalent to clicking the button.

    Now the dialog box looks like this:



14. Press ENTER.

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



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 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 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 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.

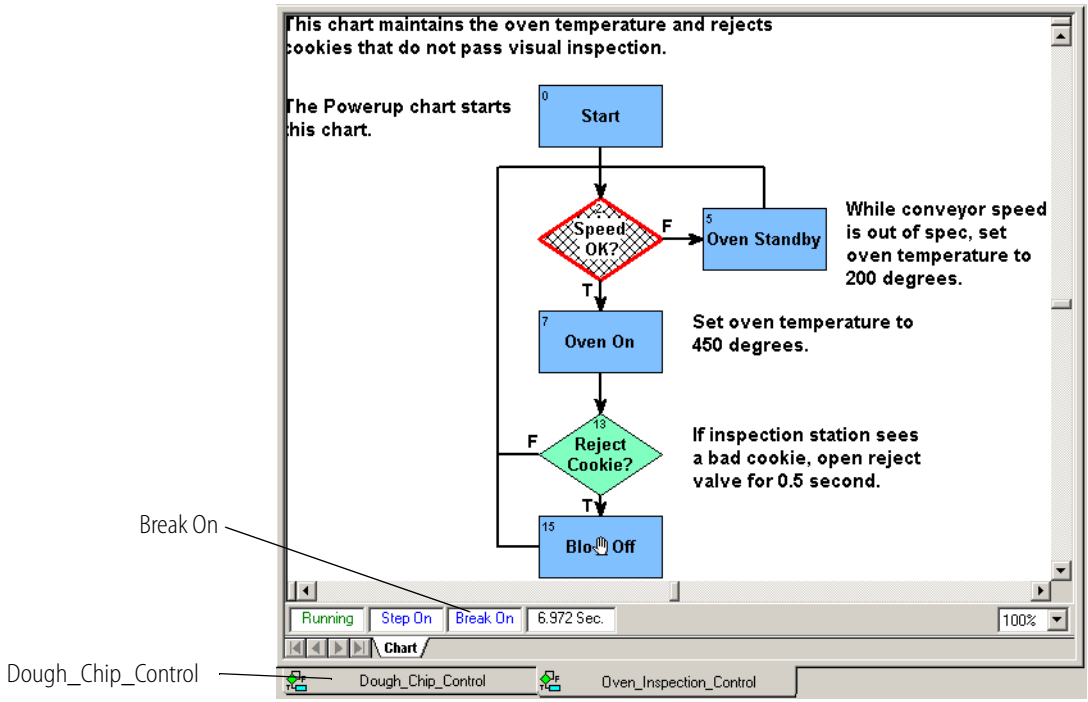
7. Click the Pause button.

Step Off changes to Step On.

8. Click the Breakpoint Tool button and click once on the Blow Off block.

   A breakpoint hand appears on the block.

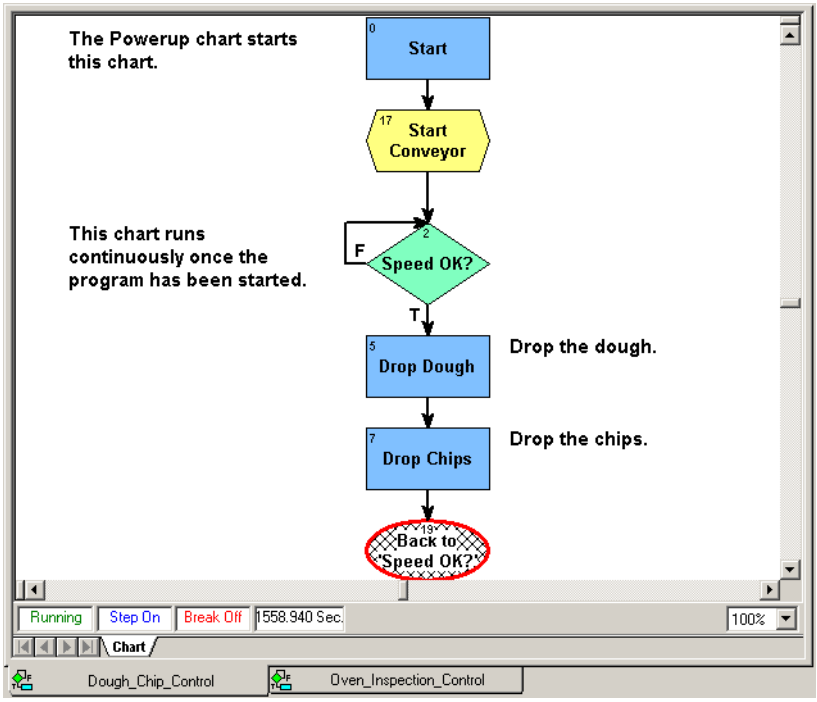
9. Click the right mouse button or press ESC to release the tool. Notice that Break On now appears in the chart status bar.



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

    The chart appears.

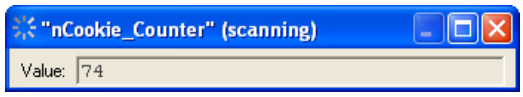
11. Click the Pause button to pause the chart.



Using a Watch Window

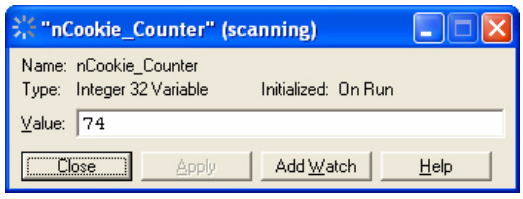
To see clearly what we're doing, we'll create a watch window to monitor cookie production.

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

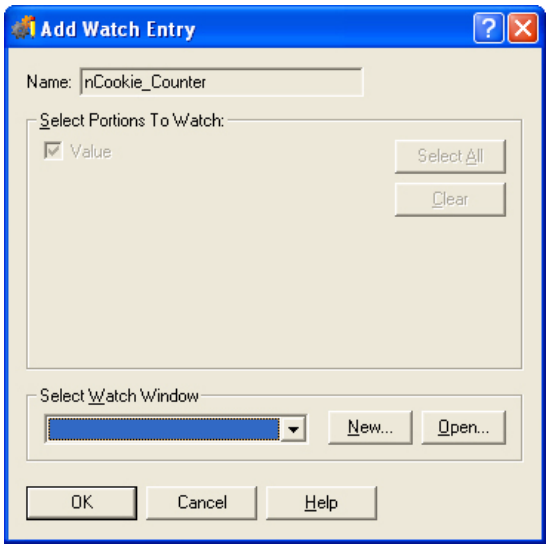


The value is frozen at some number, such as the 74 shown above. Since the counter is no longer increasing, we can see that cookie production has temporarily stopped.

2. Maximize the dialog box.



3. Click Add Watch.



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

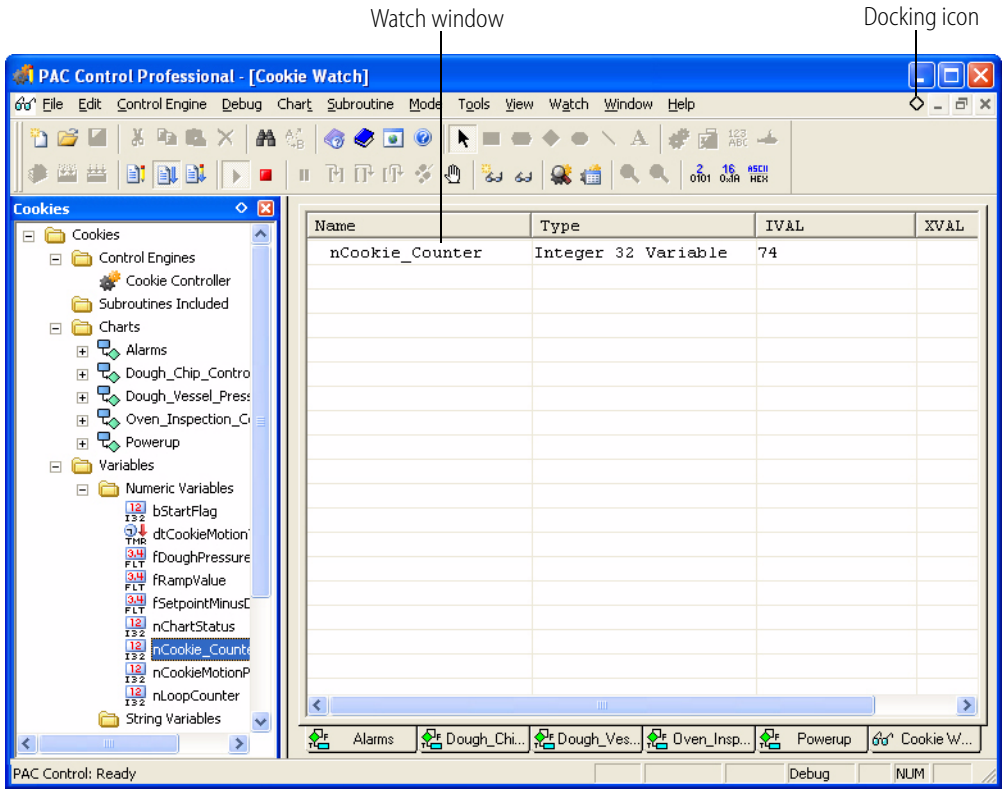
4. Click New.

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.

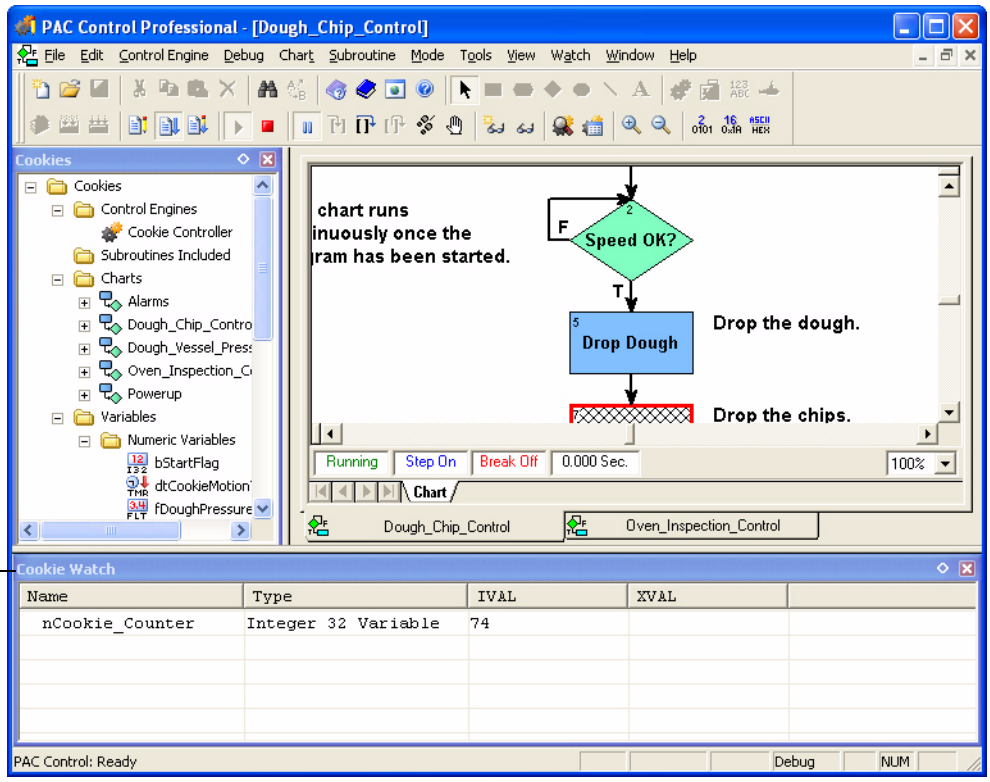
   The new watch window looks something like this:



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 icon in the upper-right corner.

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

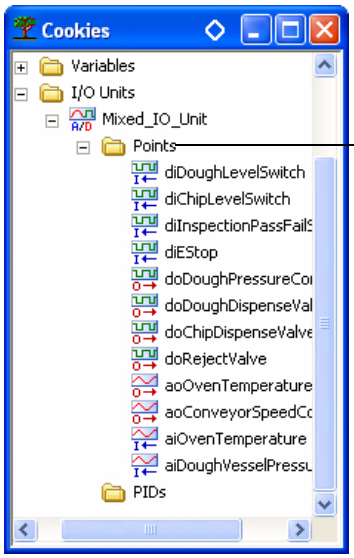


Now we'll trip the latch that signals a bad cookie.

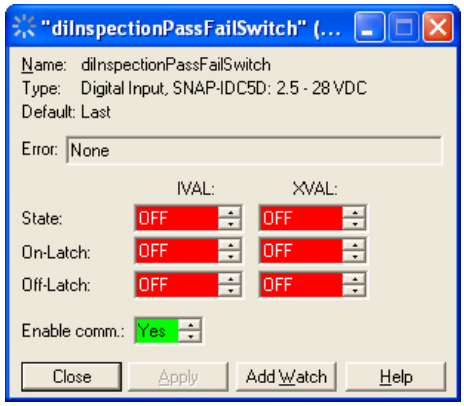
8. On the Strategy Tree, under the I/O Units folder at the bottom of the window, expand Mixed_IO_Unit by clicking the plus sign at the left of the name.

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. In the minimized dialog box, click the maximize button.

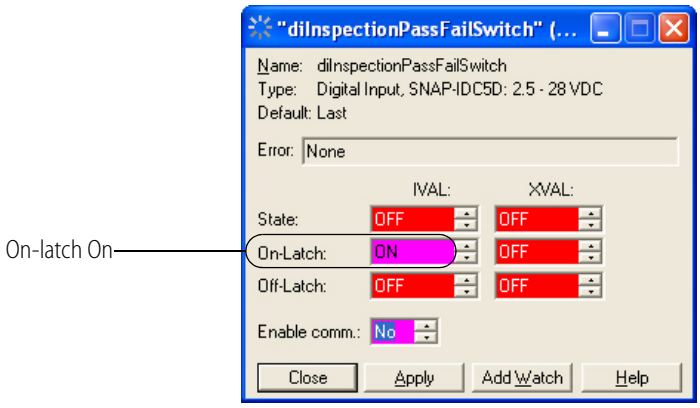


NOTE: Don't worry if the red error message "I/O unit not configured" or XVAL values appear in this dialog box. This occurs because the strategy is configured for sample hardware that probably doesn't correspond to your actual I/O units and modules. PAC Control can't locate this hardware. That's okay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

11. 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:



12. Click Apply.

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

13. Click Add Watch.

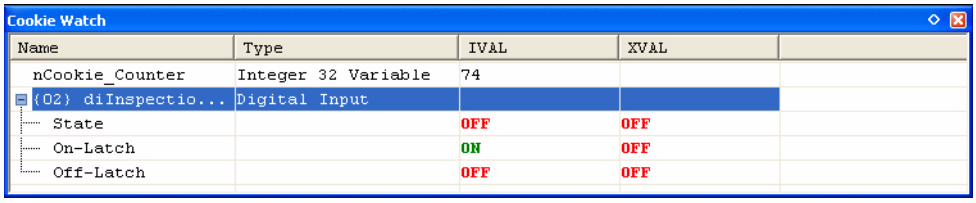
We'll add the variable to our watch window, so we can see what happens.

14. 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.

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

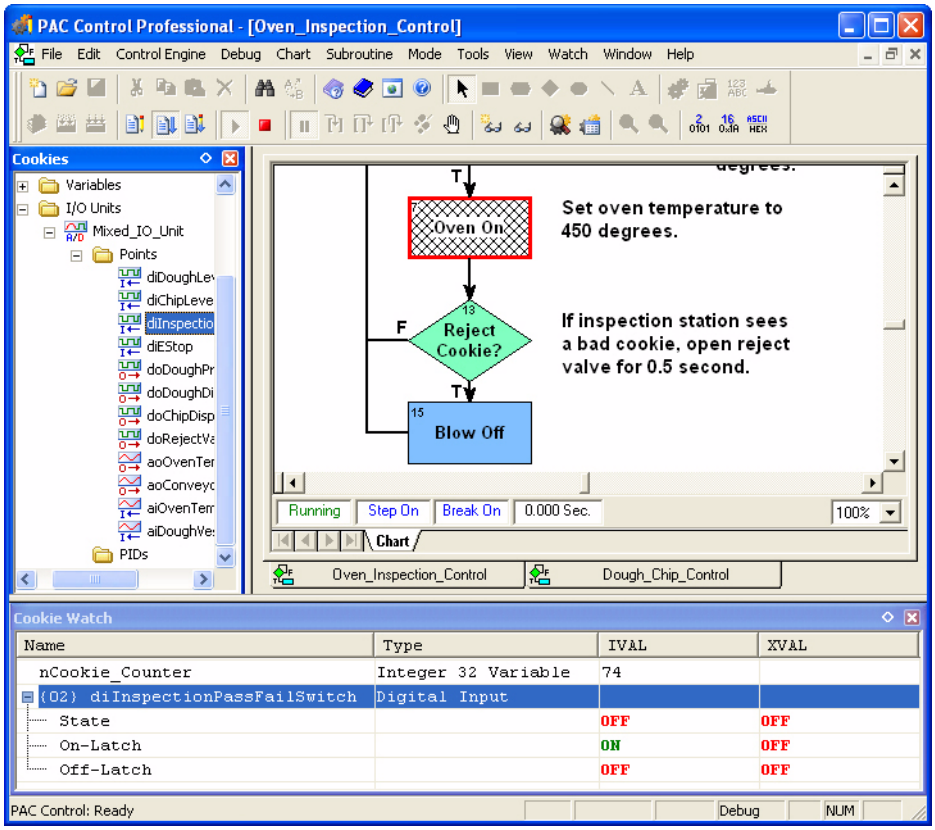
Your screen may show only part of the words.

16. 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.



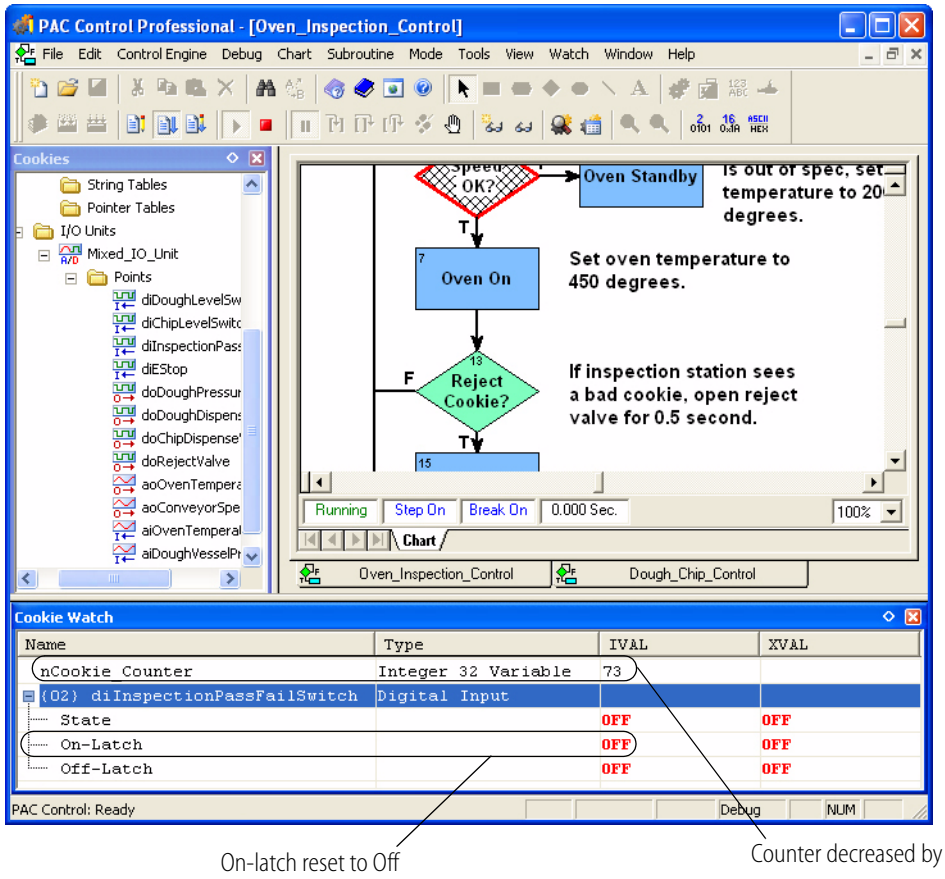
17. Click in the Oven_Inspection_Control chart to make it active, and move the scroll bars until you can see the Blow Off block at the bottom.

Your window now looks something like this:



18. 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.



19. 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.

20. Click the Dough_Chip_Control chart tab. Click the Pause 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 button in the toolbar.

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

2. 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.

What Is PAC Control?

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

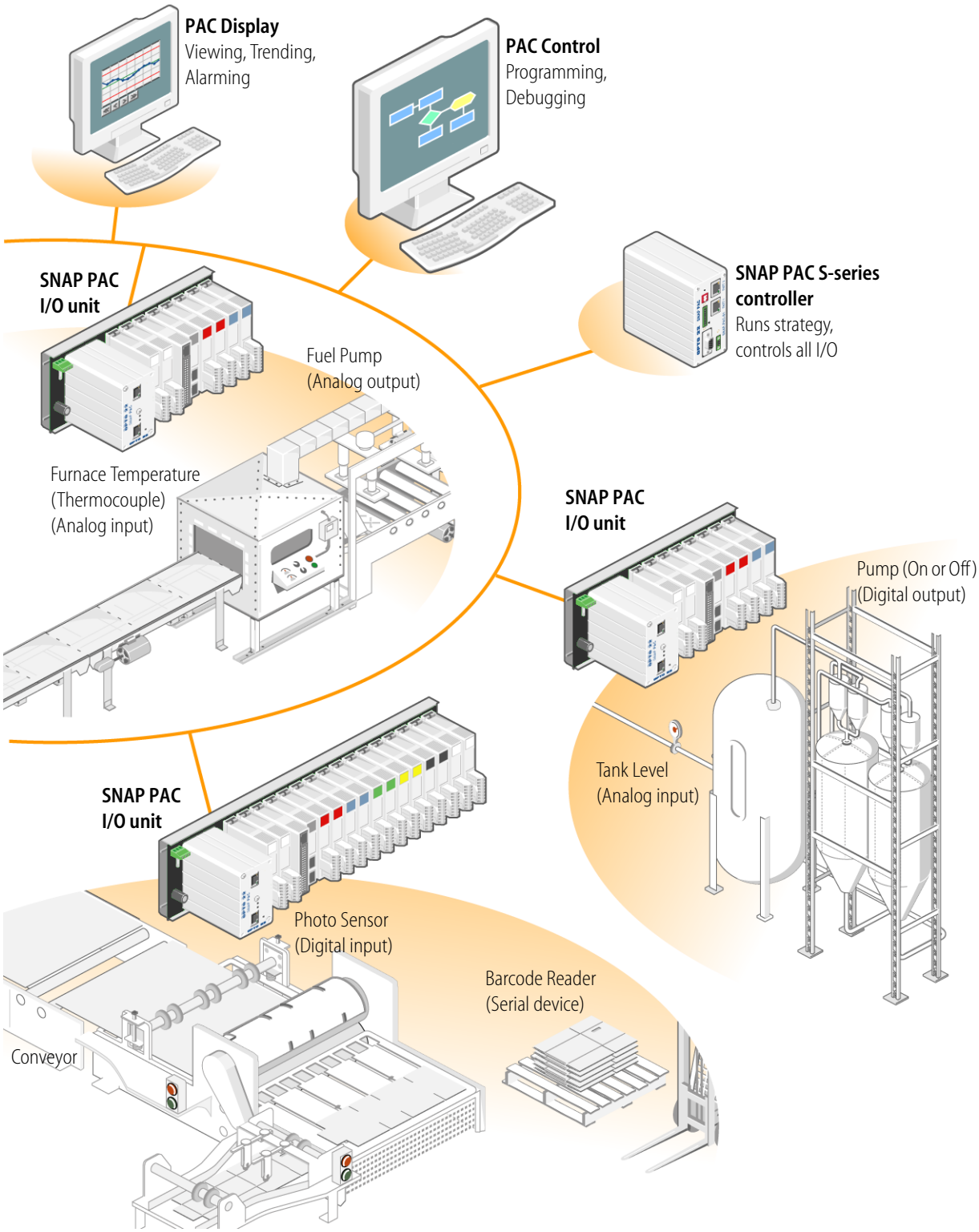
About PAC Control 45
General Control Concepts 47
PAC Control Terminology 50
PAC Control Main Window 55
Windows and Dialog Boxes in PAC Control 59
Customizing PAC Control for Your Needs 65
Online Help 71

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

These processors are programmable electronic components that provide the intelligence required for automation. In an Opto 22 PAC Control system, the processor is called a *control engine*, and it is built into the Opto 22 controller.

Using PAC Control, you create a set of instructions (a software program) that tells the control engine how every aspect of a process should work. Using PAC Control, you download the software program to a Opto 22 SNAP PAC controller, and the controller runs it as a stand-alone application. Since the instruction set is stored in the processor's electronic memory, much as a small computer would store it, the PC can be turned off or used for other operations while the control engine in the processor runs the program. And the instructions can be easily modified when necessary.

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 Special-Purpose I/O Modules

In the PAC Control system, some I/O modules do not contain standard analog or digital input/output points, but are used for special purposes.

SNAP serial communication modules provide two channels of serial data. Through these channels, data can be sent to and received from a serial device. In the Control System Example on page 46, the barcode reader is an example of a serial device.

SNAP high-density digital (HDD) modules provide 32 channels of digital input or 32 channels of digital output in one module. Any of the digital input or output devices shown in the diagram could be wired to HDD modules rather than standard 4-channel digital modules.

Key Features

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

- "Copying I/O Configurations" on page 123

  A configuration file allows you to 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

  Using the two independent Ethernet network interfaces on a SNAP PAC controller, you can 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.

- "Persistent Data" on page 235

  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 299

  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 Tables During Strategy Download" on page 240

  When you are adding table variables in PAC Control, you can set all table elements to one initial value. Or, you 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 343

  You can use subroutines that are independent from strategies, but that can be called from any 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 142

  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, 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

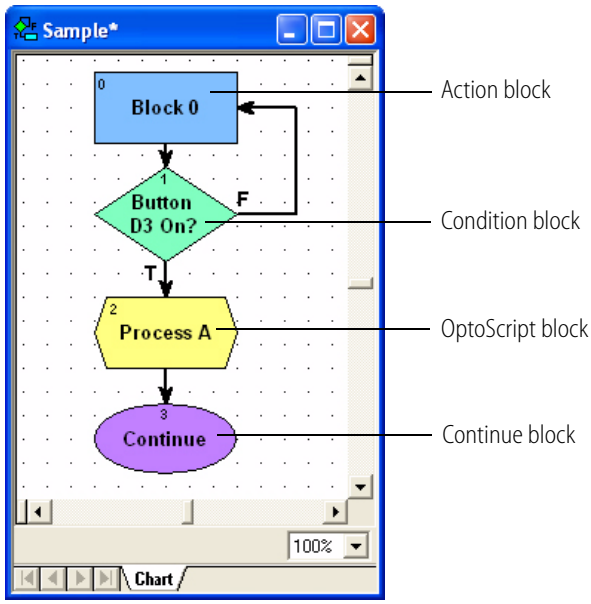
Analog points have a range of possible values. Temperature and pressure are examples of analog information. Temperature might be any number in a range, -2 or 31.65 or 70.1 or many other possible numbers. Analog points can be either inputs (such as a tank level sensor) or outputs (such as a fuel pump).

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.

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" on page 52.)

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.

- 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.

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 OptoScripts.
- **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.

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 22 SNAP PAC R-series 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. 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 236, you can use it wherever the object itself would be used.

strategy

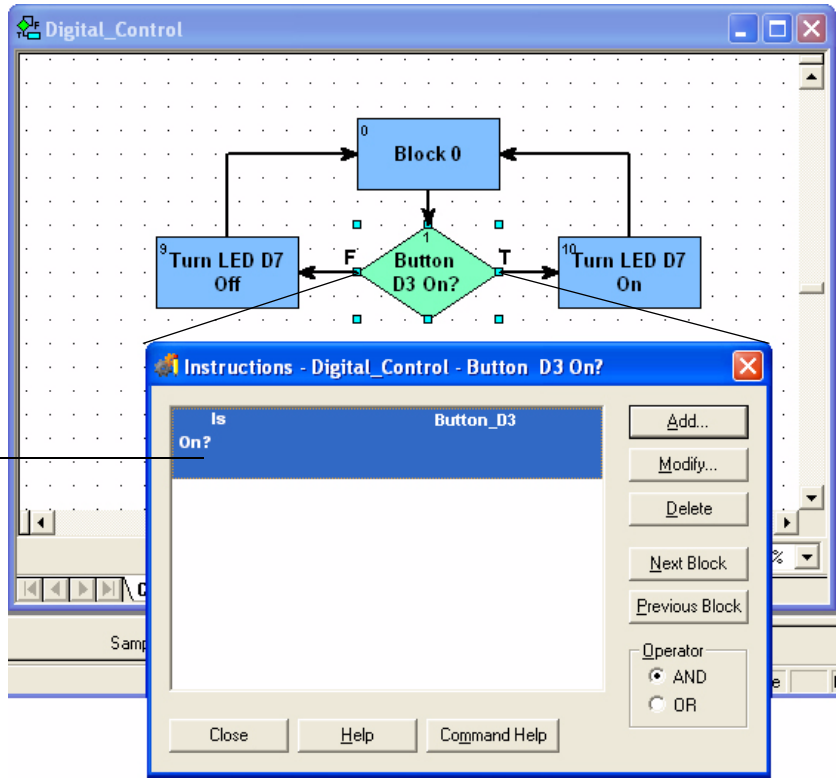
The software program you create using PAC Control Basic or PAC Control Professional. The strategy includes all the definitions and instructions (commands) necessary to control the process that one Opto 22 controller handles. Since most control processes are complex, the strategy typically consists of 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, pointer, or communication handle. When you create the variable, you designate the type of data it contains.

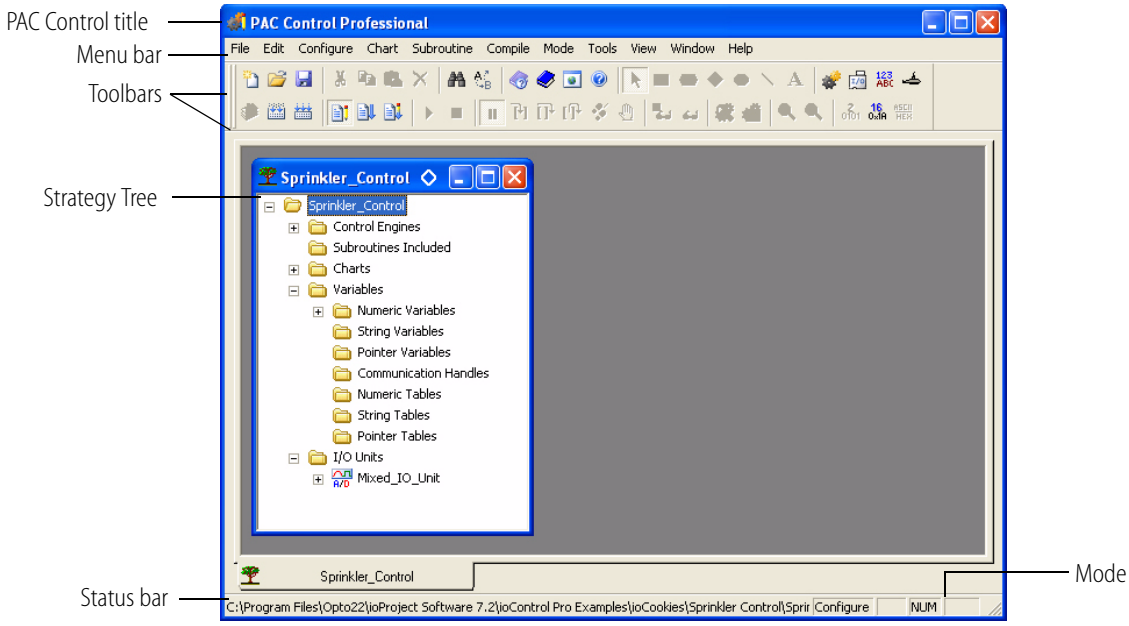
The instruction shown below is for the condition block, *Button D3 On?* This block contains only one instruction. As you look at the chart, you can see that the answer to the question in the instruction determines whether the process flows to *Turn LED D7 On* or to *Turn LED D7 Off*.



Instruction for the block named Button D3 On?

PAC Control Main Window

With a strategy open, the main window in PAC Control Basic and PAC Control Pro looks 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.

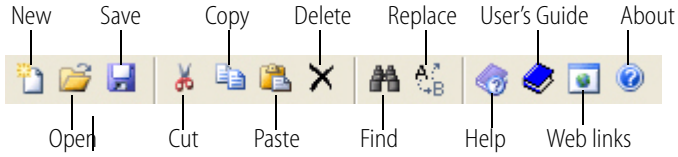
To change modes, choose the mode you want from the Mode menu, or click its button in the toolbar.

Toolbars

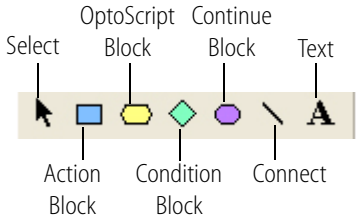
Toolbars give you shortcuts for doing many things that also appear on menus. Toolbars in PAC Control include standard Windows buttons like New and Help as well as special buttons for PAC Control functions. Like the menu bar, the tools that you can use depend on which mode you're in. Tools that don't apply to a mode are grayed out.

The following toolbars are standard in PAC Control. To change the buttons in them or to create your own toolbar, see "Customizing Toolbars" on page 66.

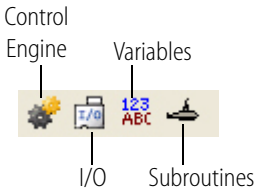
File Toolbar



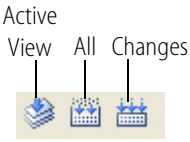
Drawing Toolbar



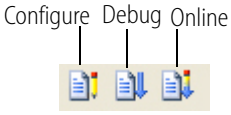
Configure Toolbar



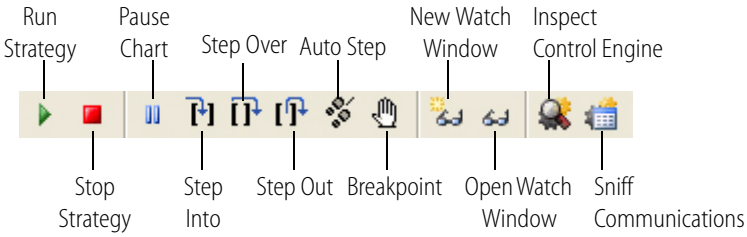
Compile Toolbar



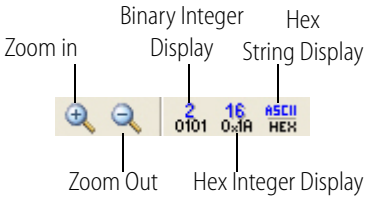
Mode Toolbar



Debug Toolbar



View Toolbar

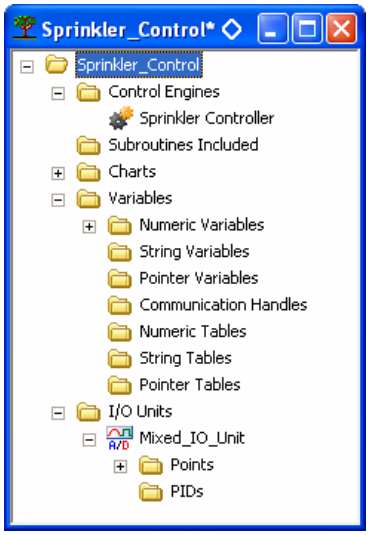


Moving Toolbars

You can move toolbars to the place where they are most convenient for you. To move a toolbar, click and drag it with the mouse to the location where you want it to be.

Hiding and Showing Toolbars

To hide or show a toolbar, choose View→Toolbars. In the dialog box, click to put a check mark in the box next to the toolbar you want to show. To hide a toolbar, click the box to remove the check mark.



Strategy Tree

The Strategy Tree (shown at left) 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		Integer 32 Table
	Chart		Integer 64 Table
	Subroutine		String Table
	Integer 32 Variable		Pointer Table
	Integer 64 Variable		Digital I/O Unit
	Float Variable		Mixed I/O Unit
	Down Timer Variable		Analog Input Point
	Up Timer Variable		Digital Input Point
	String Variable		Analog Output Point
	Communication Handle		Digital Output Point
	Float Table		PID Loop
	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 do these things.

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

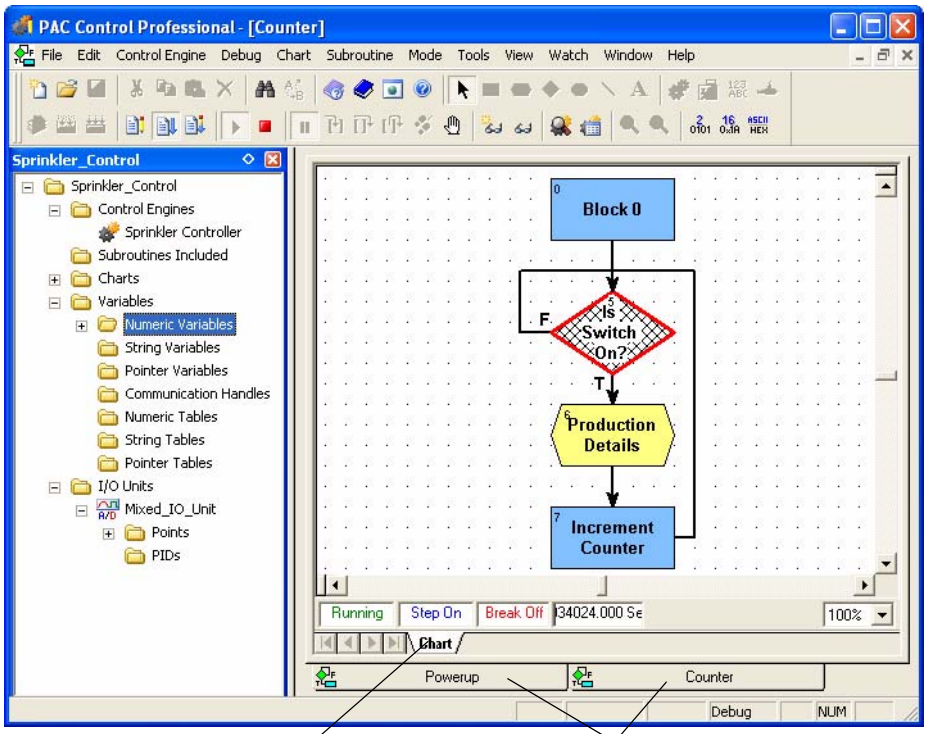
"Redrawing a Chart or Subroutine Window" on page 63

"Changing Column Width in a Dialog Box" on page 63

"Sorting Data in a Dialog Box" on page 64

Using Tabs to View Open Windows

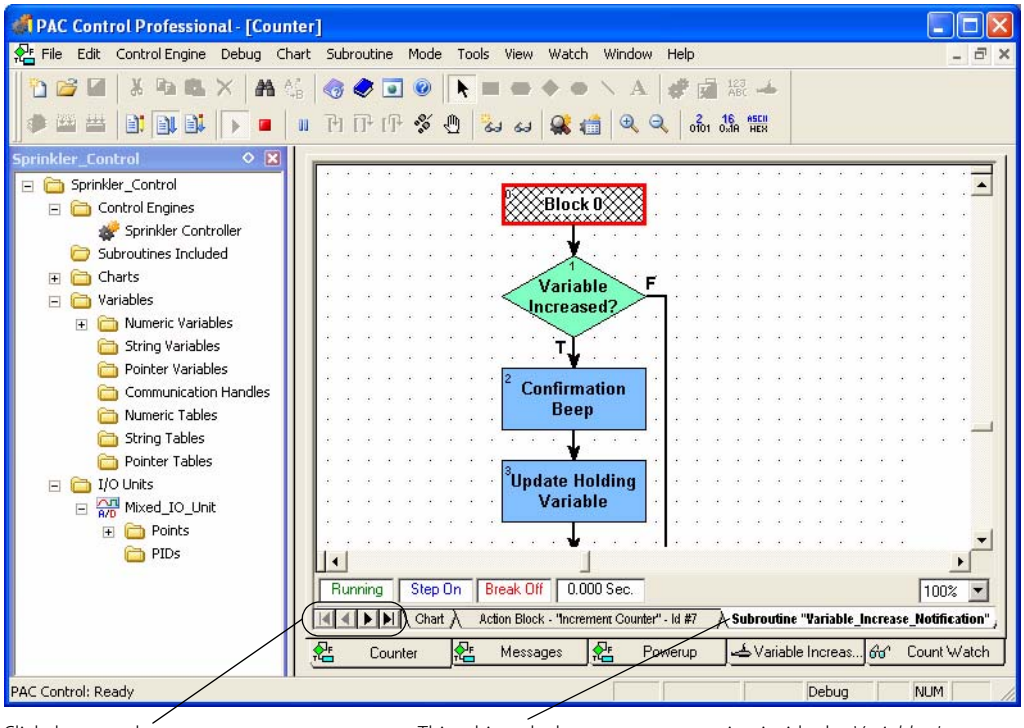
When multiple windows are open—especially if they are maximized—it can be difficult to know where you are, and windows can become lost behind each other. However, you can click the tabs at the bottom of the main PAC Control window to move among chart windows, subroutine windows, watch windows, and blocks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



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.

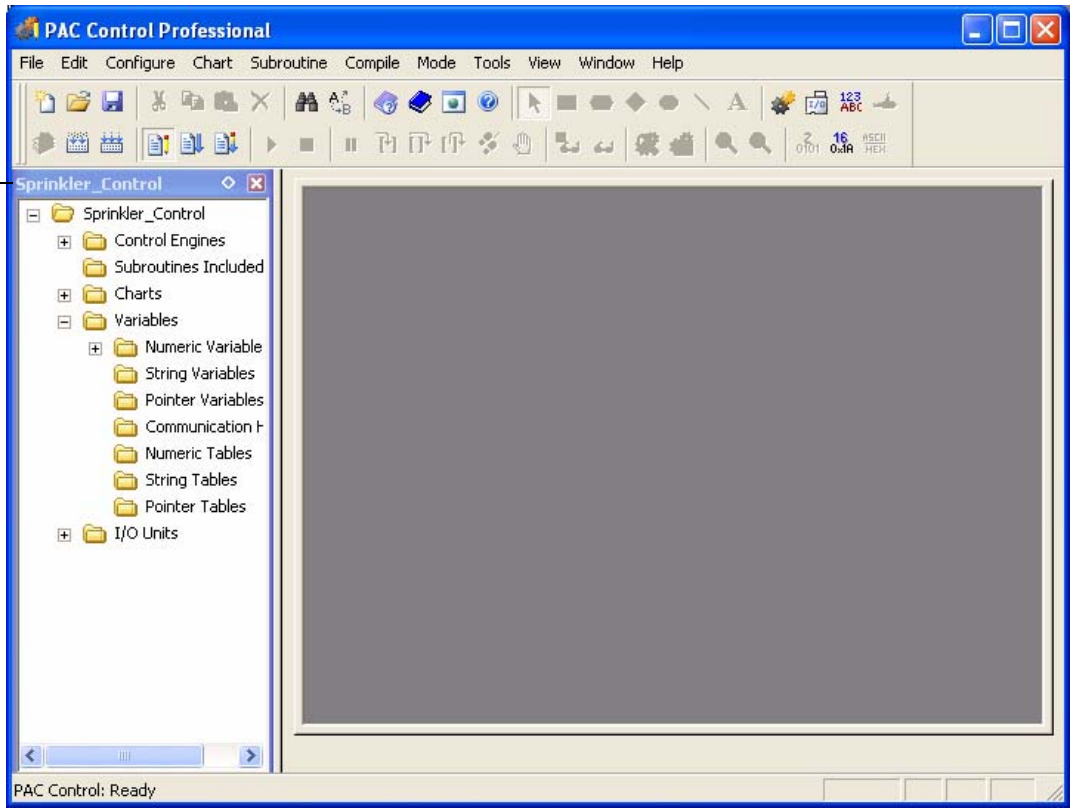
This white tab shows you are stepping inside the *Variable_Increase_Notification* subroutine, which was called by the *Increment Counter* action block on the gray tab at left. The gray *Chart* tab farther left shows the chart this action block is in. Click any tab to see how you got to where you are.

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 icon in the window's title bar.

  The window moves to its own frame. (Compare the following figure with the one on page 55 to see the difference in the Strategy Tree window.)



- 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 icon in its title bar.

Zooming in a Chart or Subroutine Window

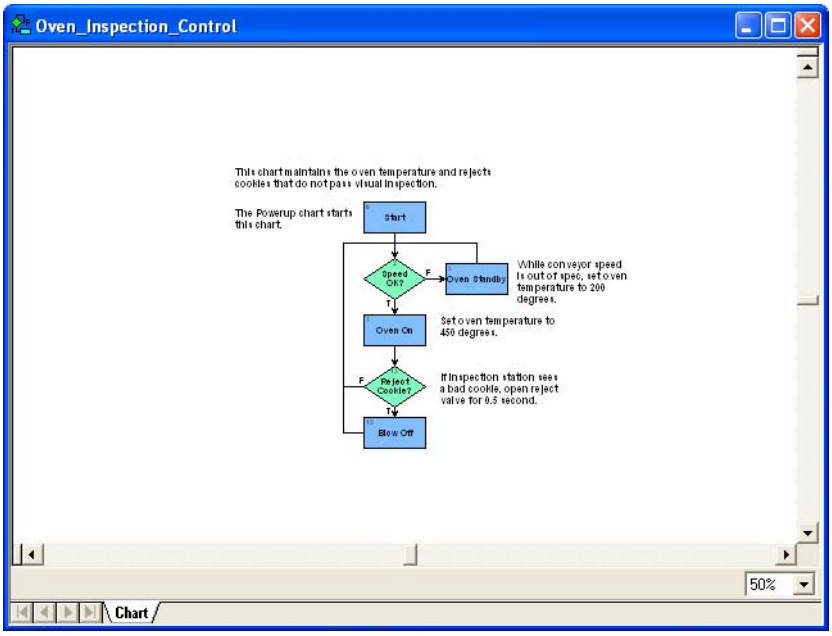
You can change the scale of any chart or subroutine window by using the zoom feature. You can view elements at seven sizes, from one-eighth to eight times the normal size. There are several ways to zoom:

- Hold down CTRL key and move the mouse wheel up and down.
- Click the Zoom In or Zoom Out buttons on the toolbar.
- 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 (to zoom in at twice the size), Out (to zoom out at half the size), or Reset (to return the zoom to 100 percent).
- From the Window menu, choose Zoom In, Zoom Out, or Zoom Reset.

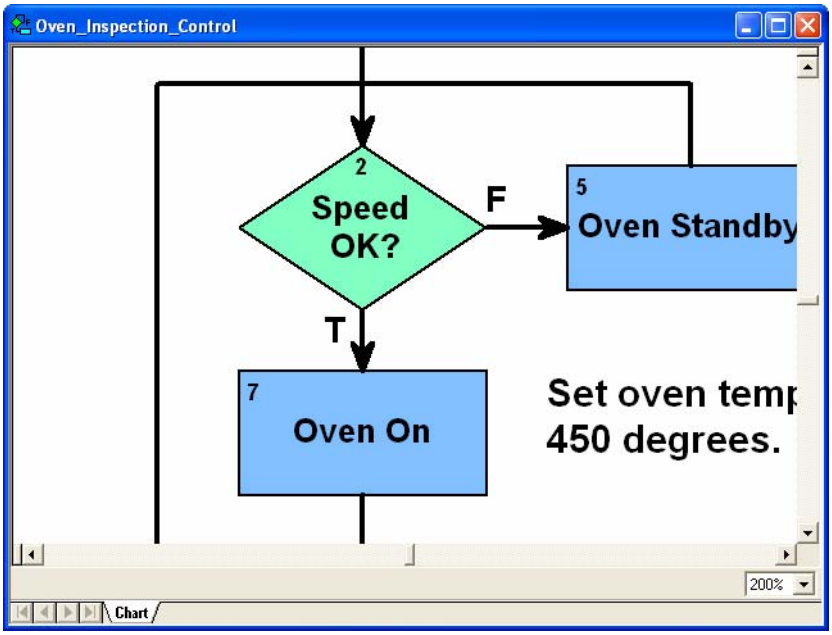
- To pick the zoom percentage you want, click the Zoom field at the bottom right of the window and select 12.5 percent, 25 percent, 50 percent, 100 percent (the default), 200 percent, 400 percent, or 800 percent.

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

Here is a window at 50 percent zoom:



The same window at 200 percent zoom looks like this:

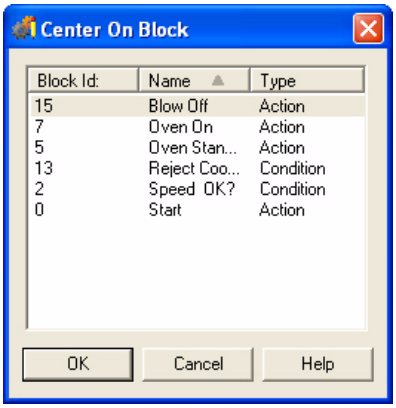


Redrawing a Chart or Subroutine Window

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 popup menu.

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



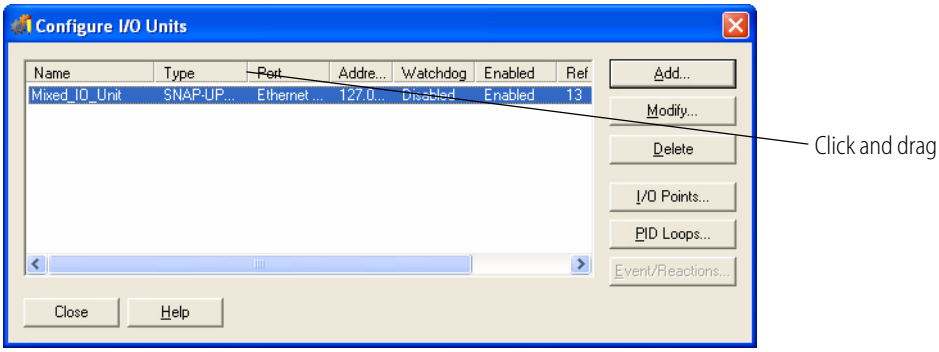
2. Double-click the block you want, or click it once and click OK.

   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.

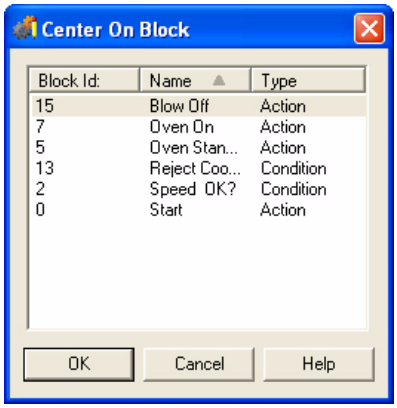
- To widen or narrow a column, click the right edge of the column label and drag it horizontally.



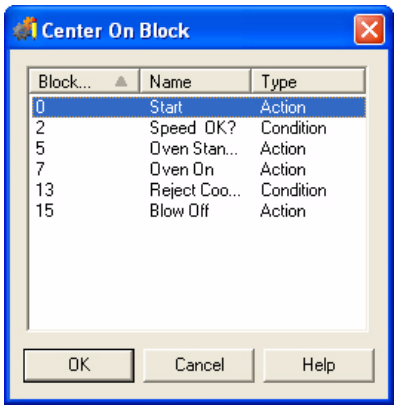
- To resize a column to the exact width of its longest entry, double-click the line that separates the column from the one on its right.

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.



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



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

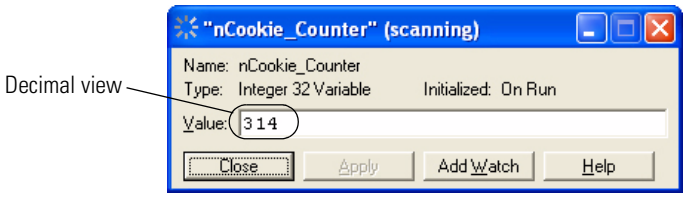
Some dialog boxes don't allow custom sorting. If you click a column label in these dialog boxes, nothing happens.

Customizing PAC Control for Your Needs

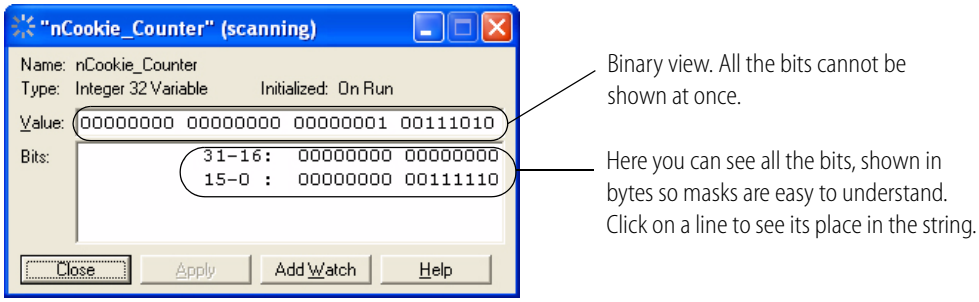
Setting Decimal, Binary, or Hex Display Mode

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 view is available in all modes: Configure, Debug, and Online.

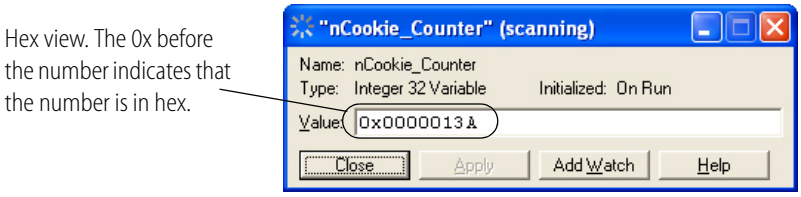
The default view is in decimal notation, as shown in the following figure.



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



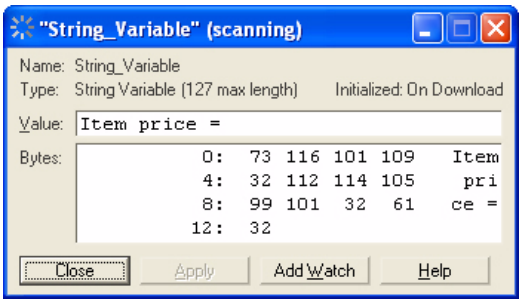
To view integers and integer tables in hex view, click the Hex Integer Display button on the toolbar or choose View→Hex Integer Display. Here's how the integers appear in hex:



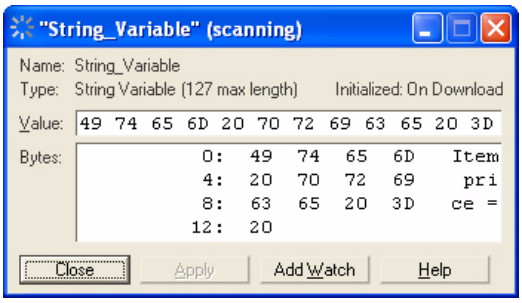
To return to decimal view, click the toolbar button again or choose the same item again from the View menu.

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:



To change to hex notation, click the Hex String Display button on the toolbar ASCII HEX or choose View→Hex String Display. Here's how the string appears in hex:



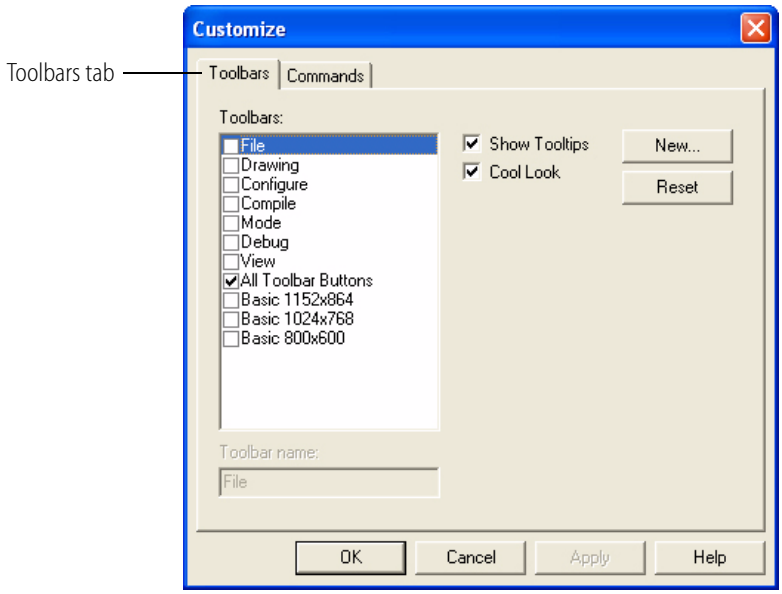
Customizing Toolbars

You can customize toolbars in PAC Control for the way you work. You can choose toolbars to match your screen resolution. You can move toolbars to another place in the window. You can move a button into another position or toolbar, or delete it if you don't use it. You can even create your own toolbar with just the buttons you use.

Choosing Toolbars for Your Screen Resolution

You can choose toolbars to match your screen resolution and place the most frequently used toolbar buttons all on one line. To do so, follow these steps:

1. Choose View→Toolbars.
2. In the Customize dialog box, make sure the Toolbars tab is open.



3. Click to place a check mark next to the toolbar for your current screen resolution. Click OK.

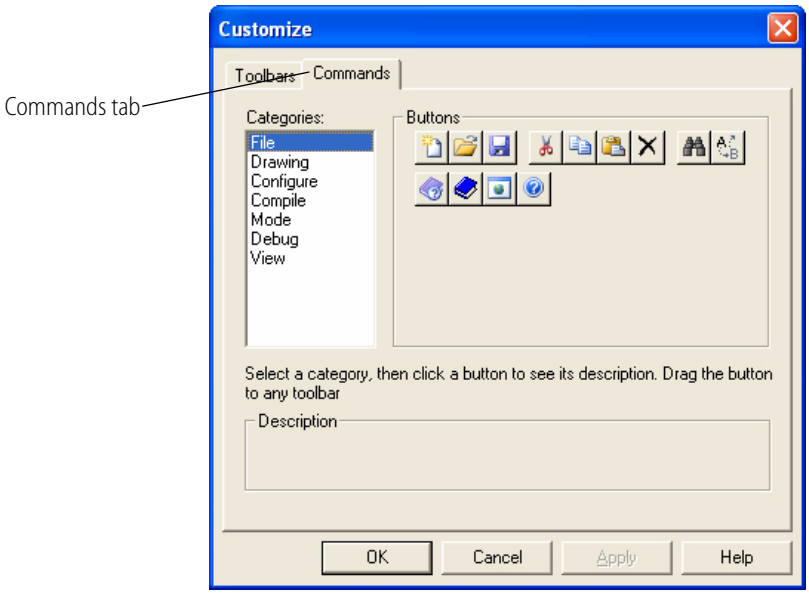
   The most commonly used tools appear in a single line in the main window. If you want to change the tools, see "Moving and Deleting Buttons" on page 68.

Moving Toolbars

To move a toolbar, click on its edge to outline the whole toolbar. Then drag the toolbar where you want it. You can place it at the top, bottom, left, or right of the main window or of any docked window.

Moving and Deleting Buttons

1. Choose View→Toolbars.
2. In the Customize dialog box, click the Commands tab.



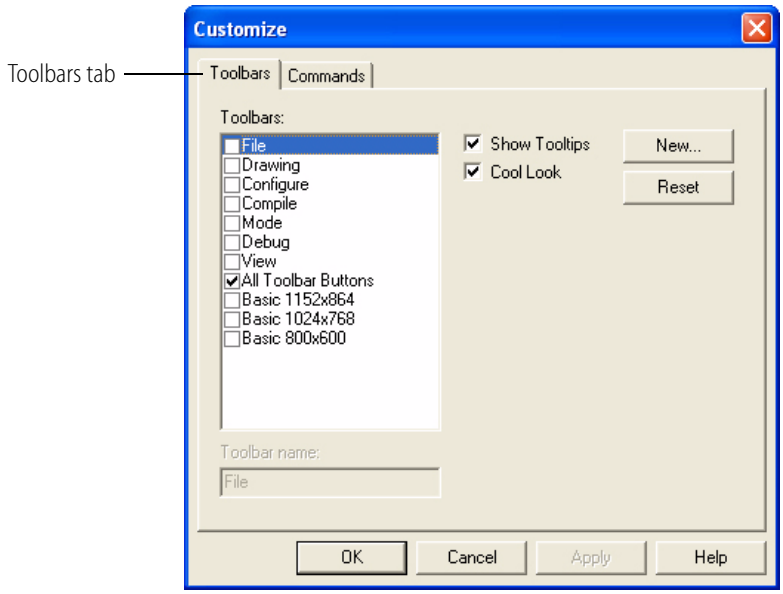
3. In the Categories list, click the name of the toolbar you want to move the button from.

   The buttons in that toolbar appear in the Buttons area.

- To change a button's position in its current toolbar, click the button in the PAC Control main window (not in the dialog box) and drag it to the position you want.
- To move a button to another toolbar, click the button either in the dialog box or in the main window, and drag it to the toolbar in the main window where you want it.
- To delete a button from a toolbar, click the button in the main window and drag it out of its toolbar.

Creating Your Own Toolbar

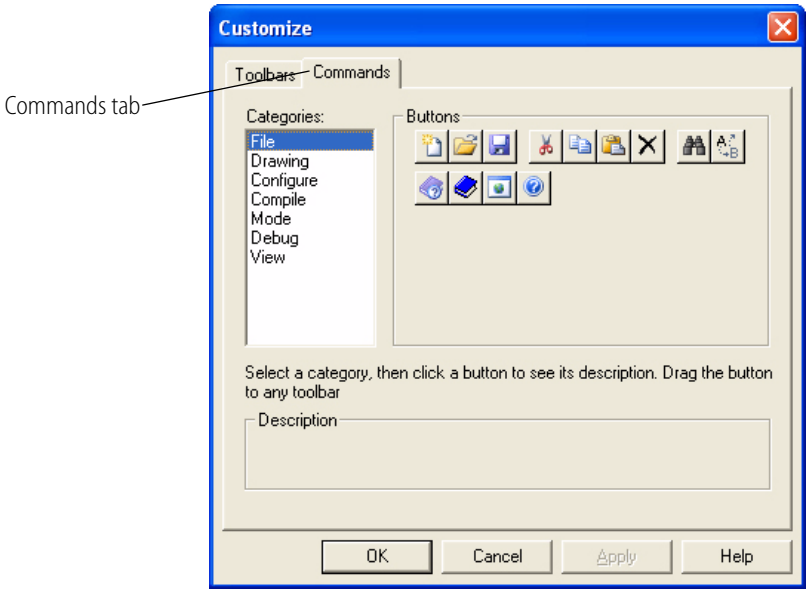
1. Choose View→Toolbars.
2. In the Customize dialog box, make sure the Toolbars tab is open.



3. Click New. Enter a name for the custom toolbar and click OK.

   A small box appears in the upper-left corner of the main window. You build your custom toolbar by placing the buttons you want in this box.

4. In the Customize dialog box, click the Commands tab.



5. In the Categories list, click the name of the standard toolbar that contains the button you want to place in the custom toolbar.

The buttons in that toolbar appear in the Buttons area.

6. Click the button you want and drag it to its place in the small gray box. Repeat until you have all the buttons you want in the custom toolbar.
7. When you have finished building the custom toolbar, click it to outline it, and drag it into position in the PAC Control main window.
8. When you have finished customizing toolbars, click OK in the Customize dialog box.

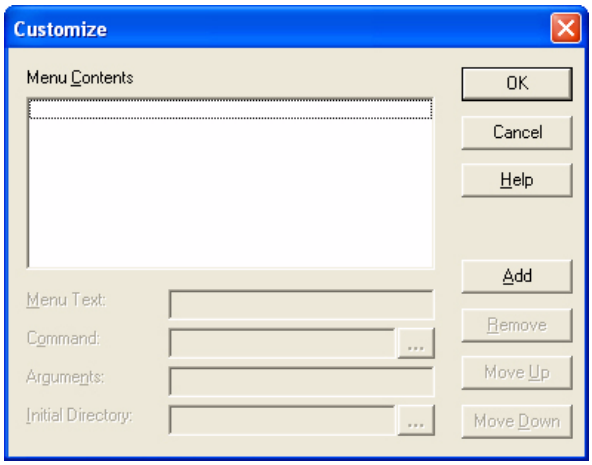
Setting Up Applications to Launch from PAC Control

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.
3. In the Menu Text field, type the name of the application as you want it to appear in the Tools menu.
4. In the Command field, type the path for the application's executable file, or click the browse button ... and navigate to the file.
5. (Optional) In the Arguments field, type any necessary command line parameters.
6. (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.

7. 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.
8. When you have finished adding applications, click OK.

   You return to the PAC Control main window, and the applications you've added now appear in the Tools menu.

Online Help

To open online Help, choose Help→Help Topics, 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.

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 an instruction (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, choose PAC Control Options→Show/Hide Instruction Type Information.)

To open online copies of PAC Control manuals and quick reference cards, choose 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 by choosing Help→Opto 22 on the Web.

Designing Your Strategy

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 Chapter 10, "Programming with Commands," 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 209.

In this Chapter

Steps to Design 73
Basic Rules 81
Instruction Examples 83
Optimizing Throughput 95

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

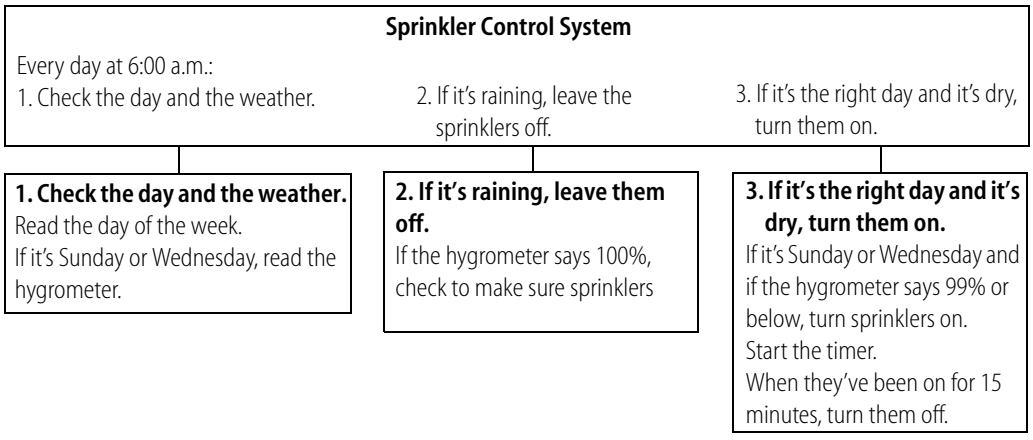
What do the outputs need to be? List all the outputs. Describe the results that are needed. Check for any outputs you need but don't have.

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.



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.

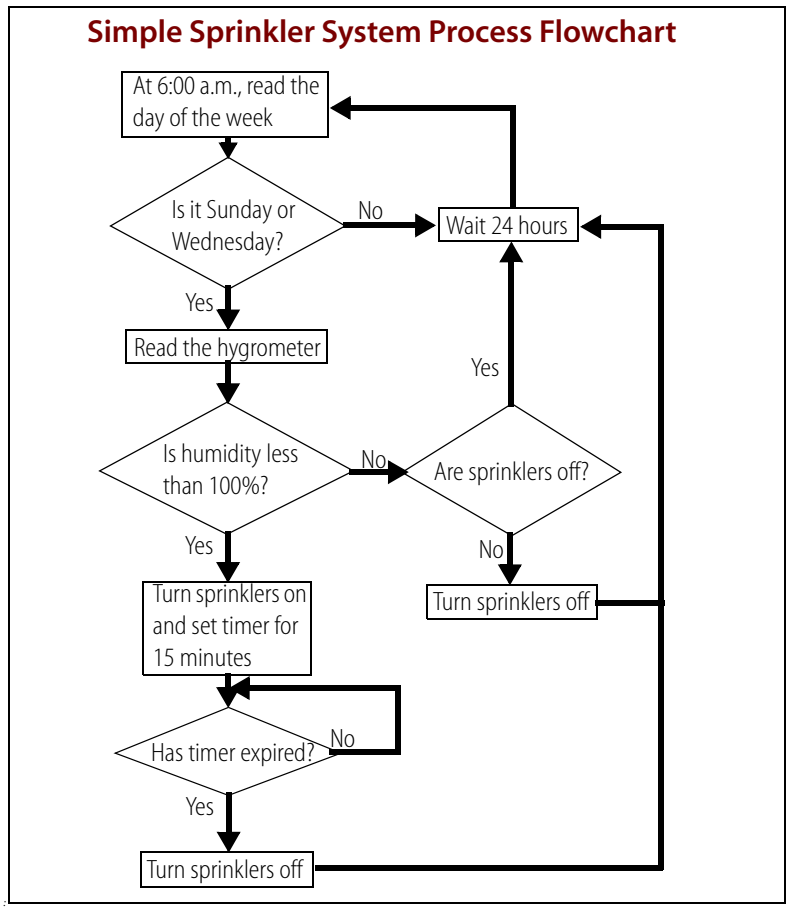
c. If the hour does not equal six, go back to step a.

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.



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

Variable Needed	Type	Name in PAC Control
Humidity	Numeric (float)	Humidity
Down timer	Timer	Timer

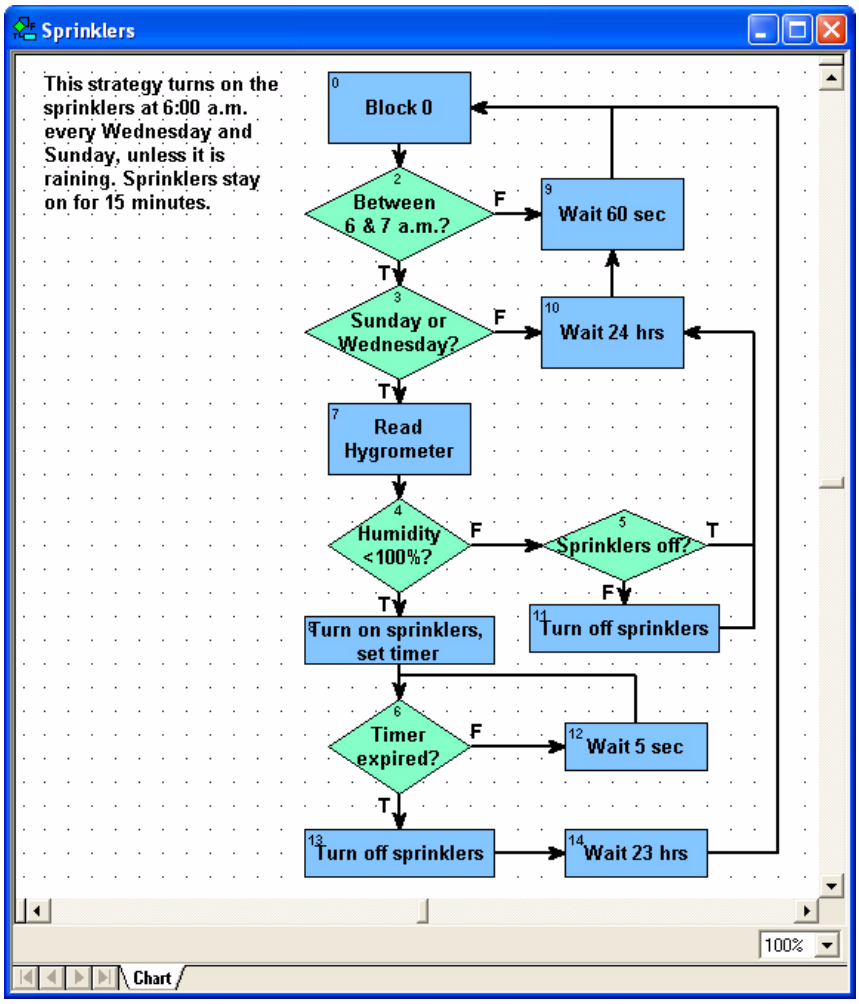
Creating PAC Control Charts and Adding Instructions

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.

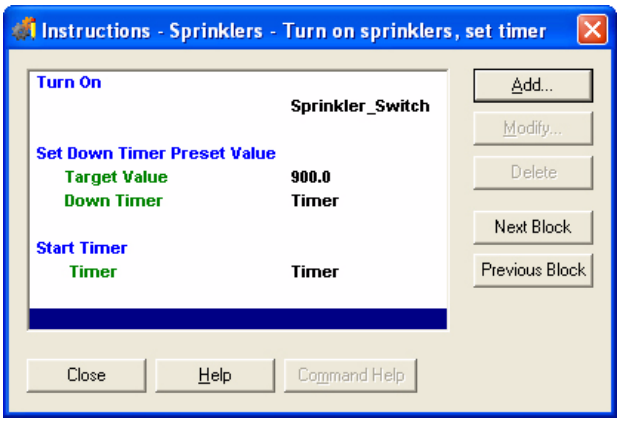
Even for a simple process like our sprinkler control, there is no single "correct" way to solve the control problem and build the strategy. Instead, there are many possible ways. The following chart shows one example:



Notice one of the differences between this chart and the human-level chart on page 76: 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.

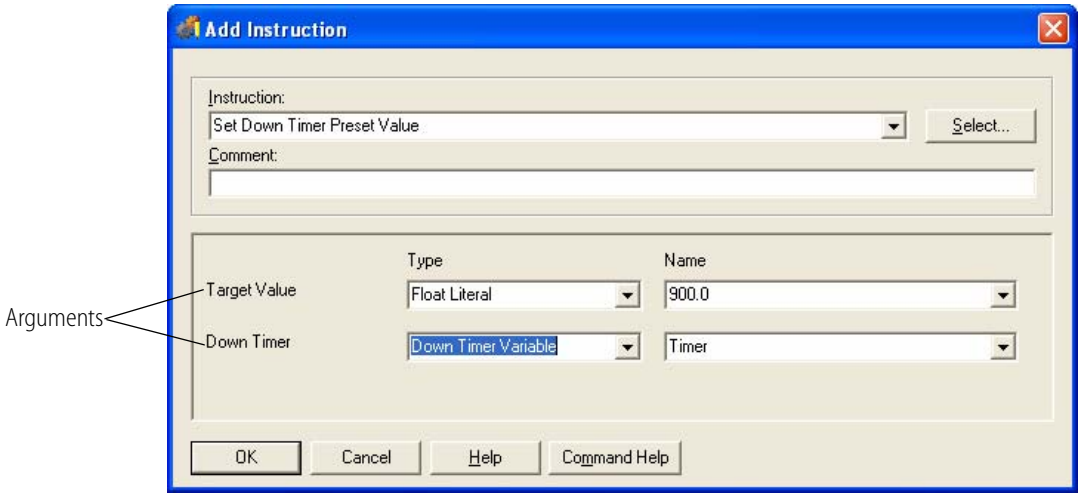
This sprinkler system is not time-critical. If the sprinklers turn on at a little past 6:00 a.m. or the grass is watered an extra few seconds, it doesn't matter. Delays give the control engine time to do other things. (For more information on delays and using control engine time effectively, see page 95.)

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":



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, 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.



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,") 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:

- 32 charts on a SNAP PAC S-series controller
- 16 charts on a SNAP PAC R-series controller

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 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. On a SNAP PAC R-series 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. (For more information, see "Optimizing Throughput" on page 95.)

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.
- 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 use Hungarian notation to indicate the type of variable, since the variable type is not always apparent in its name. See "Variable Name Conventions" on page 325 for a suggested list of notations.

Variable names might also include a way to distinguish variables used in a single chart or process from variables used by more than one chart or process. If your strategy is large, it's helpful to separate them. Variables that are used only in a table could include the table name in their name, for example Fail_Count_in_Config_Values. (Config_Values is the table name.)

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.)

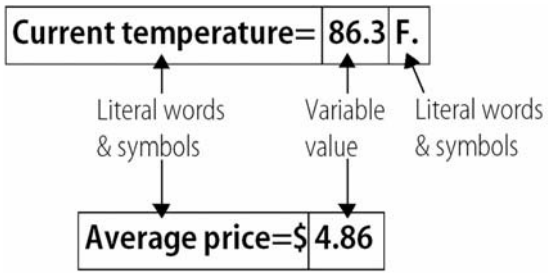
Instruction Examples

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.

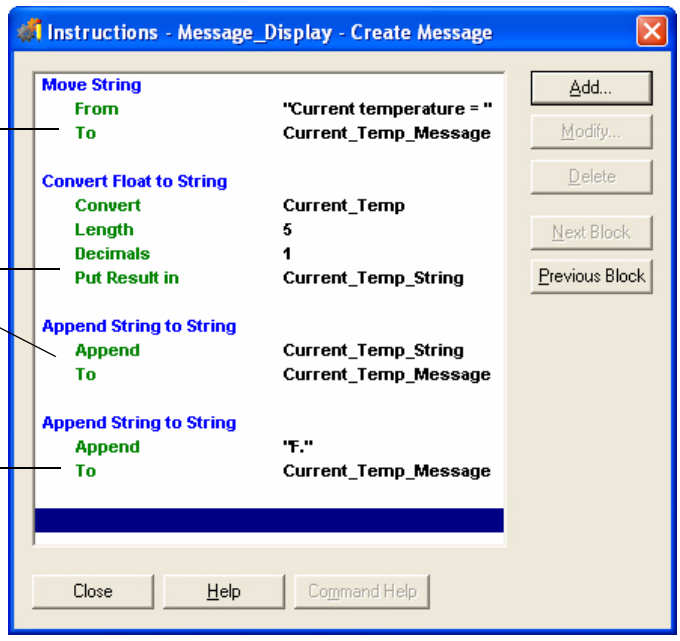
Creating Messages to Display On Screen

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.



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.

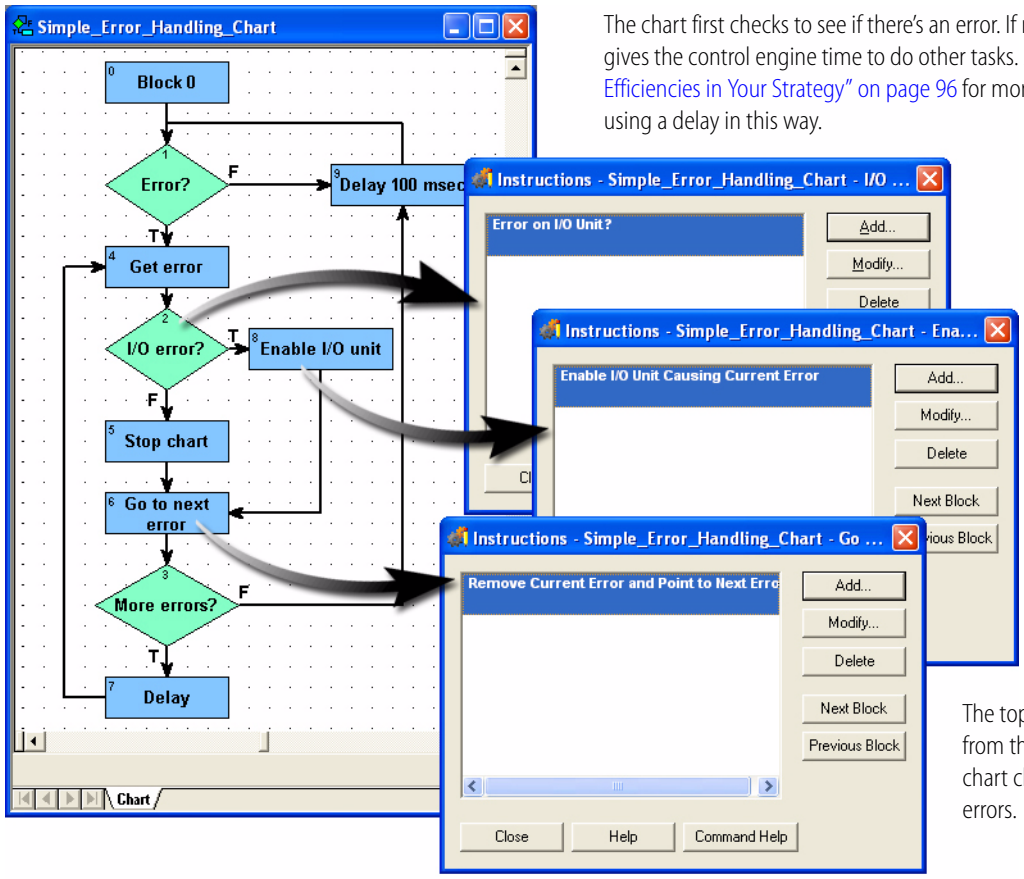
Error Handling

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.

You can easily build a simple error handling chart like the one shown below, which automatically checks errors and other messages and takes action based on the type of error.

This chart, for example, checks for I/O unit errors. Suppose a remote I/O unit loses power for a few seconds, and then the power comes back on. When the power is lost, communication to the unit is disabled. To enable communication again, you could stop the strategy and restart it, or you could enable communication to the unit in the debugger. With this error handling chart, however, you can automatically enable the I/O unit and restore communication while the strategy is running.

The chart checks the top error in the message queue. If it is an I/O unit error, communication to the unit is enabled. If it is not an I/O unit error, another action happens (another chart in the strategy is stopped). Two delays are built in: the first one to release the time slice if an error is not found, and the second one to allow the control engine time to restore communication with the I/O unit.



The chart first checks to see if there's an error. If not, a slight delay gives the control engine time to do other tasks. See "Increasing Efficiencies in Your Strategy" on page 96 for more information on using a delay in this way.

If the error is an I/O unit error, communication to the I/O unit that caused it is enabled.

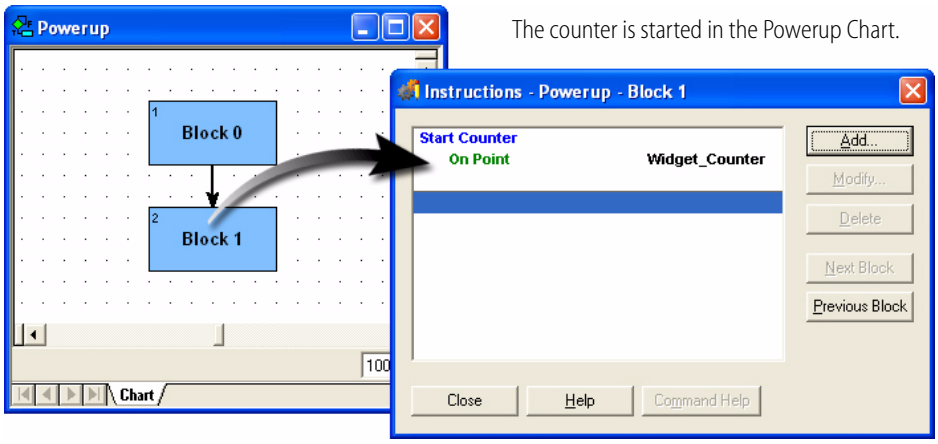
The top error is removed from the queue, and the chart checks for more errors.

The error chart shown on the previous page is designed to be started by the Powerup chart and to run continuously. If necessary, you might modify the chart to take different actions depending on the error code, using the command Get Error Code of Current Error. Or you might want to track problem I/O units with the command Get Name of I/O Unit Causing Current Error, and then save the I/O unit names to a string table. These and other commands used in error handling are listed on page 283.

Counting

If an I/O unit supports counting, any digital input on it can be used as a counter. Counters count the number of times the input changes from off to on. **You must start the counter before you can read it.** See "Counters" on page 280 for more information.

Counters are often started in the Powerup Chart. For example, suppose you wanted to count the number of widgets moving along a conveyor belt from manufacturing to shipping. You could start the counter in the Powerup Chart, like this:



The counter is started in the Powerup Chart.

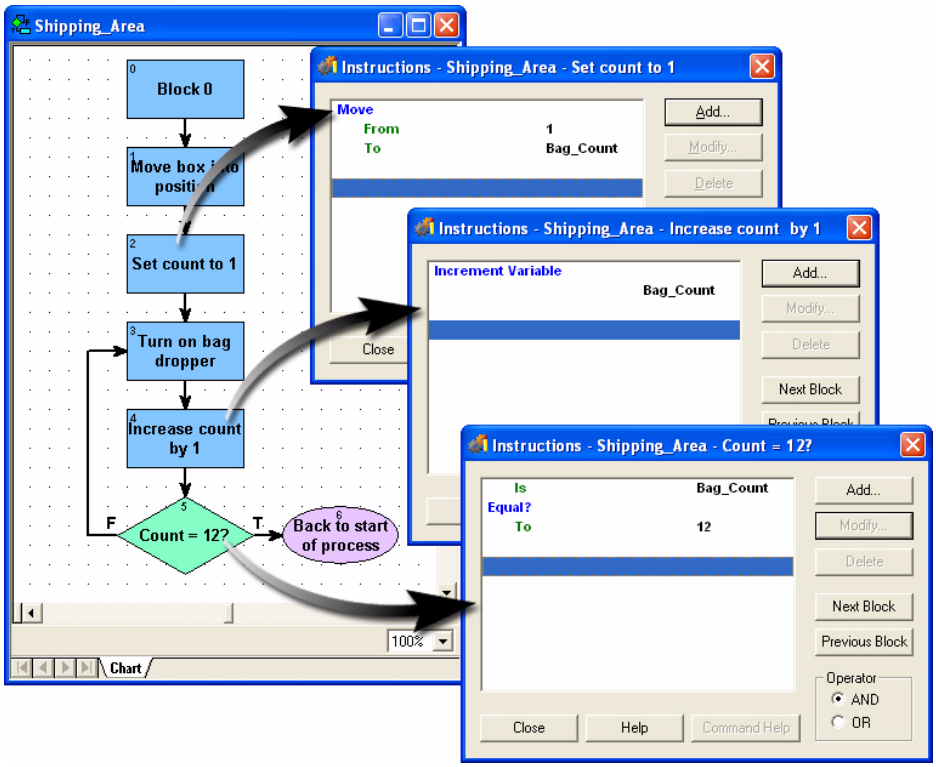
Once the counter is started, you can read its value at any time by using the command Get Counter.

If you want the counter to start over when you read its value—for example, to count the number of widgets in each 24-hour period—use the command Get & 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:



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:

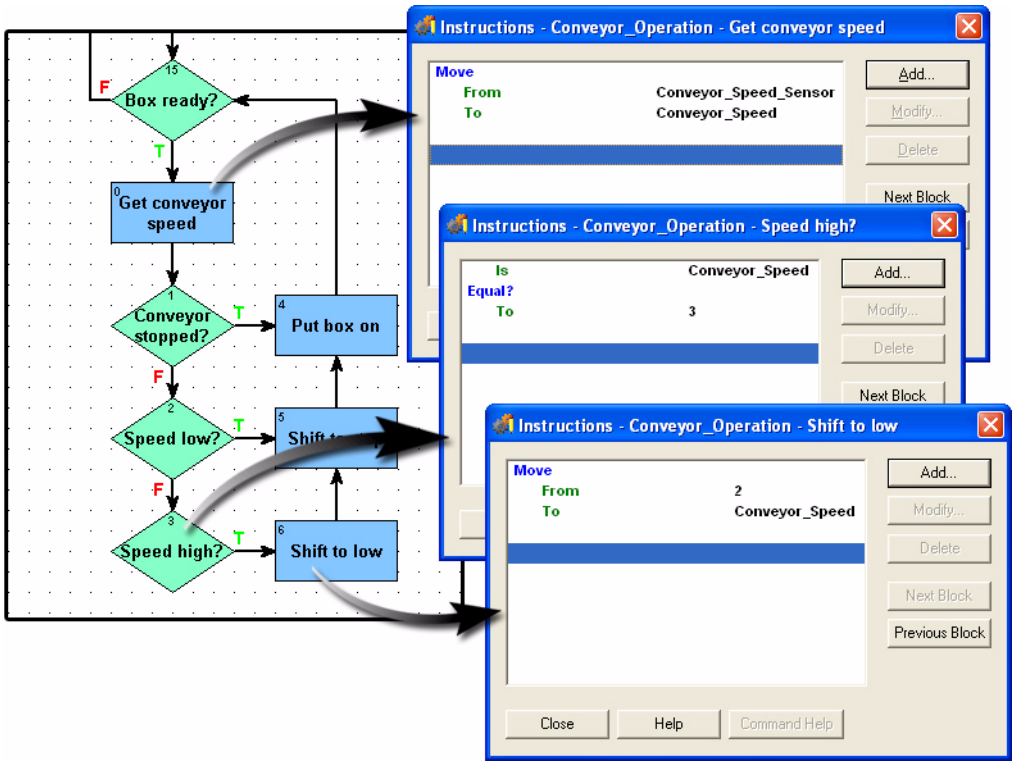
```
for Bag_Count = 1 to 12 step 1 //Count 12 items dropped into box
  Bag_Dropper = 1; //Turn on bag dropper
next
```

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; see Chapter 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 only be stopped from low speed, not from high speed. Here's a portion of a chart showing the case statements that control this process:



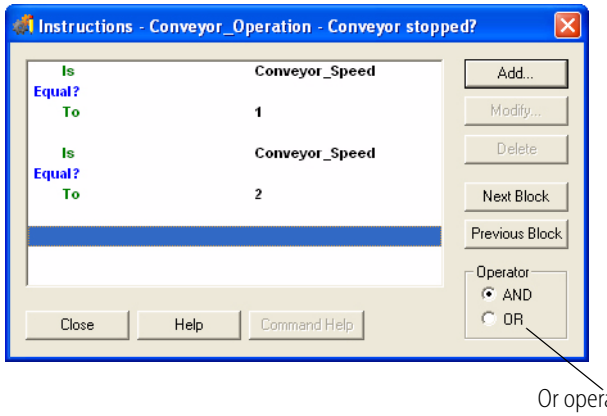
Conveyor speed is checked and moved (copied) into a variable.

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.

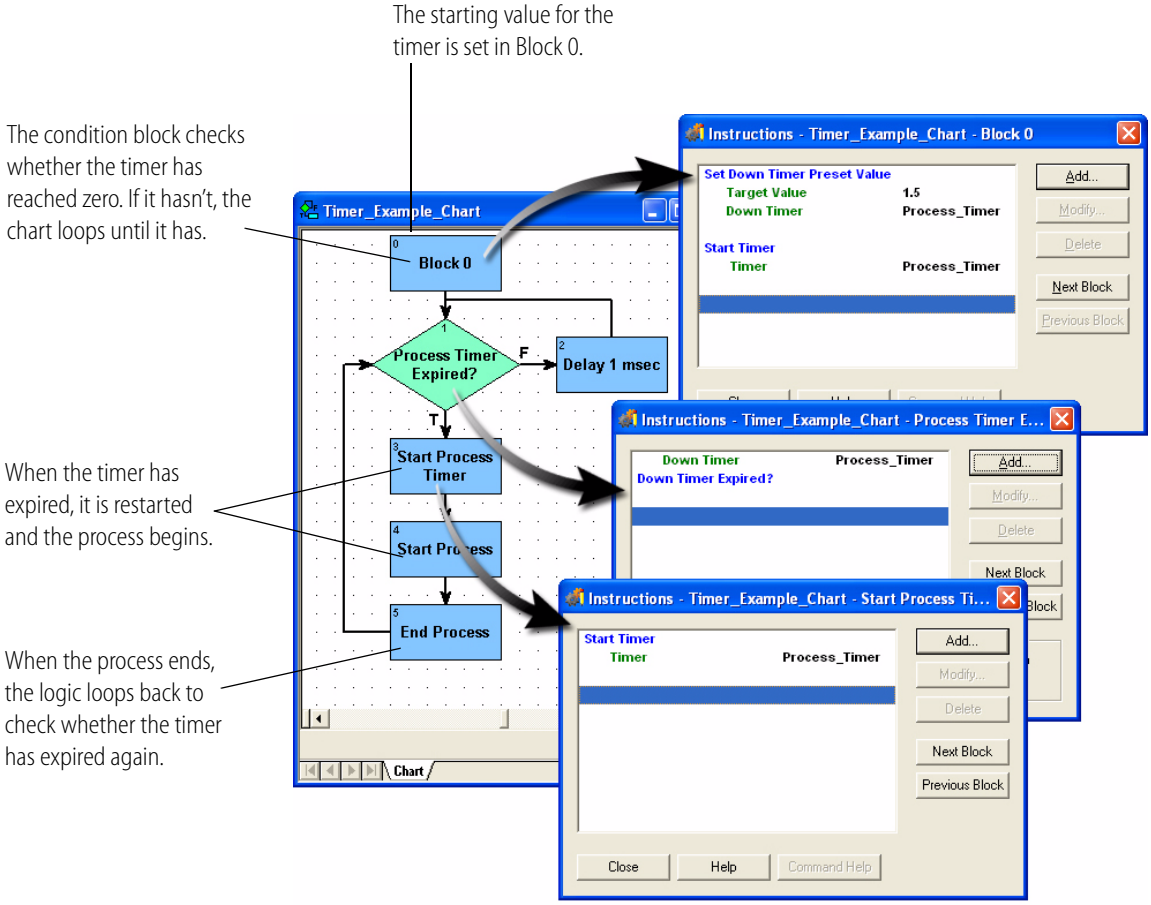


Using a Timer

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.



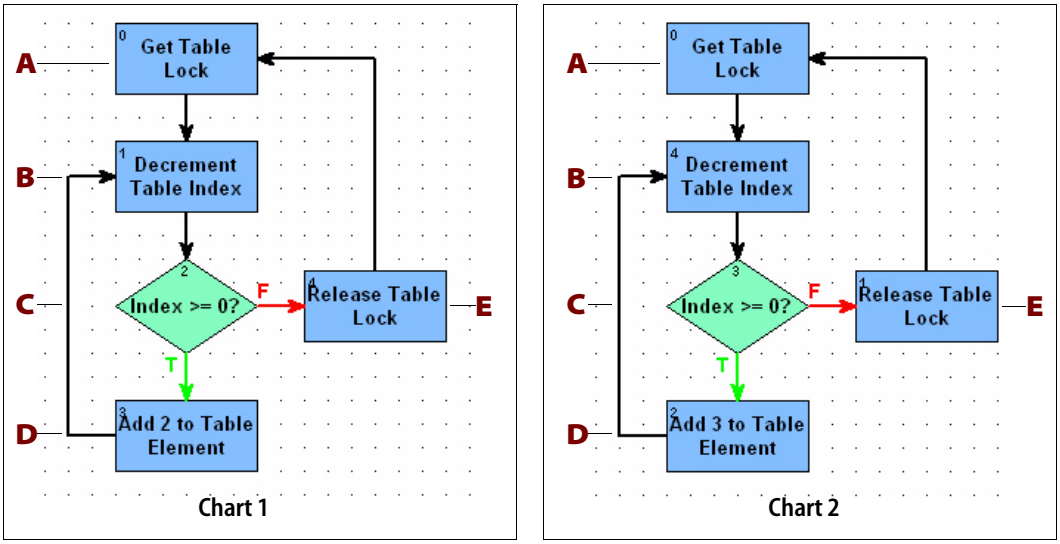
Timers can be tricky. For additional details, see "Using Timers" on page 311.

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.



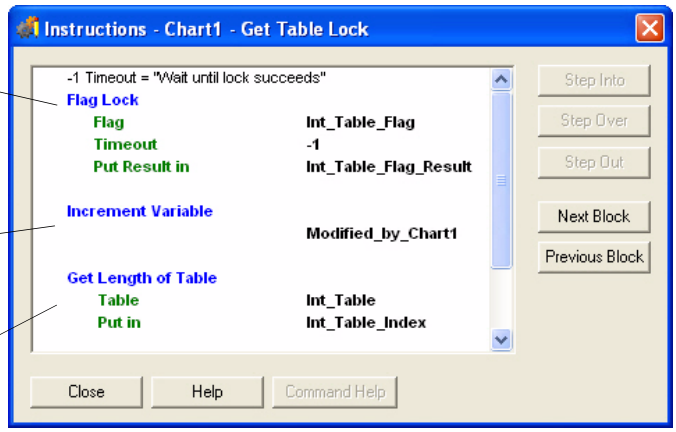
Here's what each block accomplishes:

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 91.

Flag lock will wait until Chart 2 unlocks the flag on the table. See also, "Flag Lock Time-out Values" on page 91

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



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 37.

Watch

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 only unlock 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.

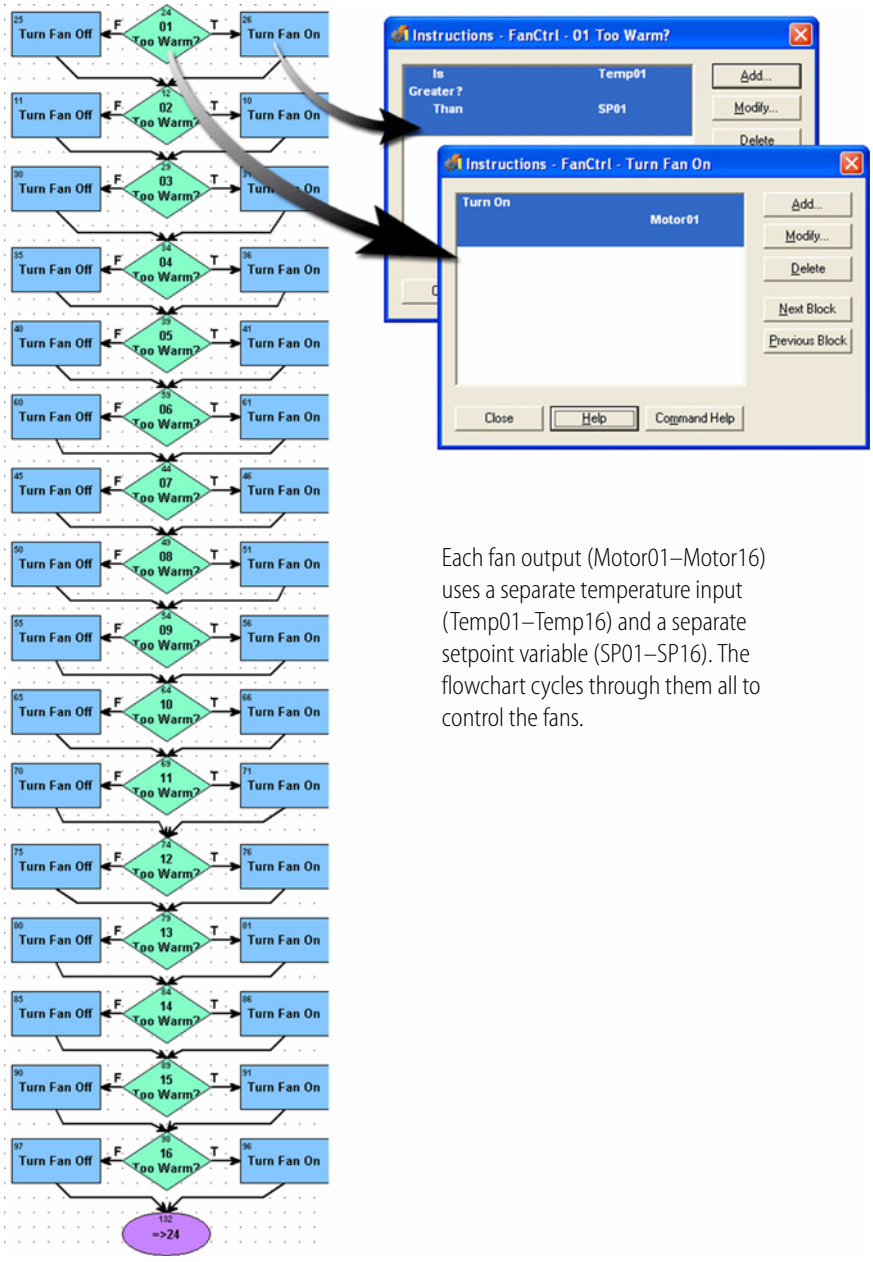
Timeout	Effect
0	Don't wait at all. Succeeds only if the flag is available to be locked immediately. Makes this a non-blocking call. If the flag is not already locked, a return of 0 (Success) is returned. If the flag is locked, a -17 (Already Locked) error is returned.
Value greater than 0	Wait maximum 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

This example shows 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.

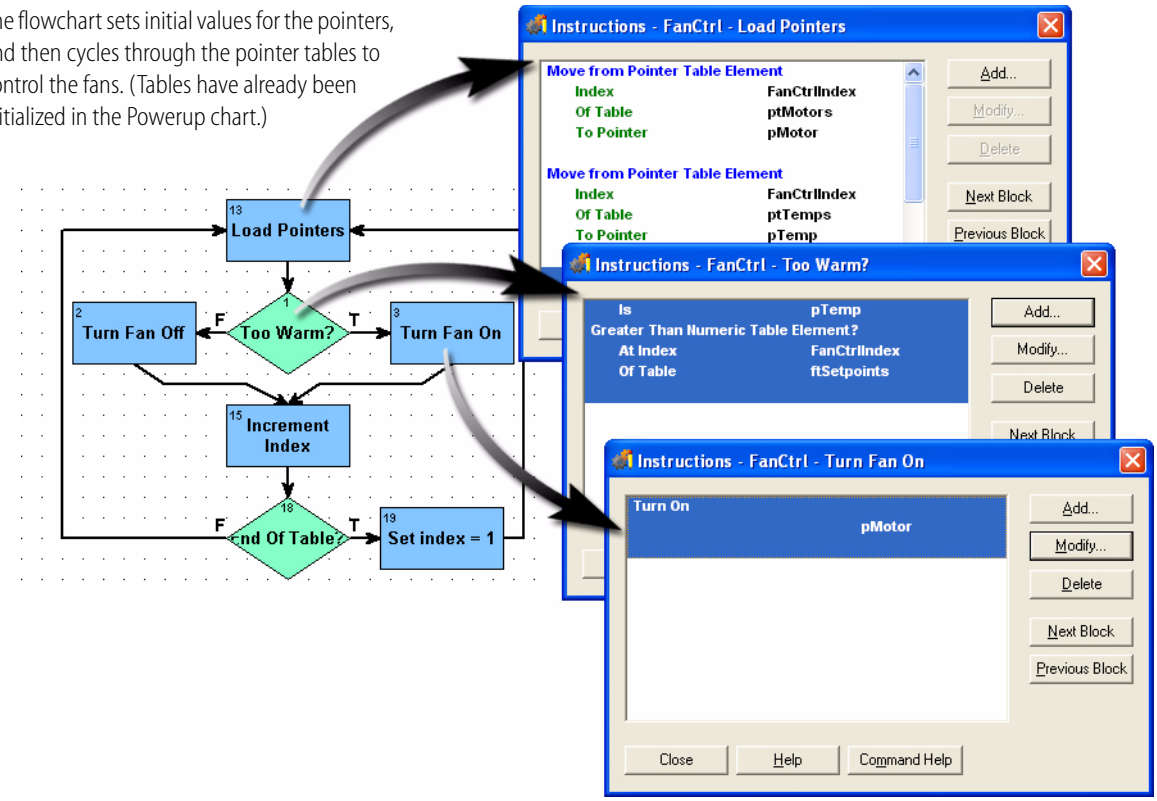


Each fan output (Motor01–Motor16) uses a separate temperature input (Temp01–Temp16) and a separate setpoint variable (SP01–SP16). The flowchart cycles through them all to control the fans.

With Pointers and Indexing

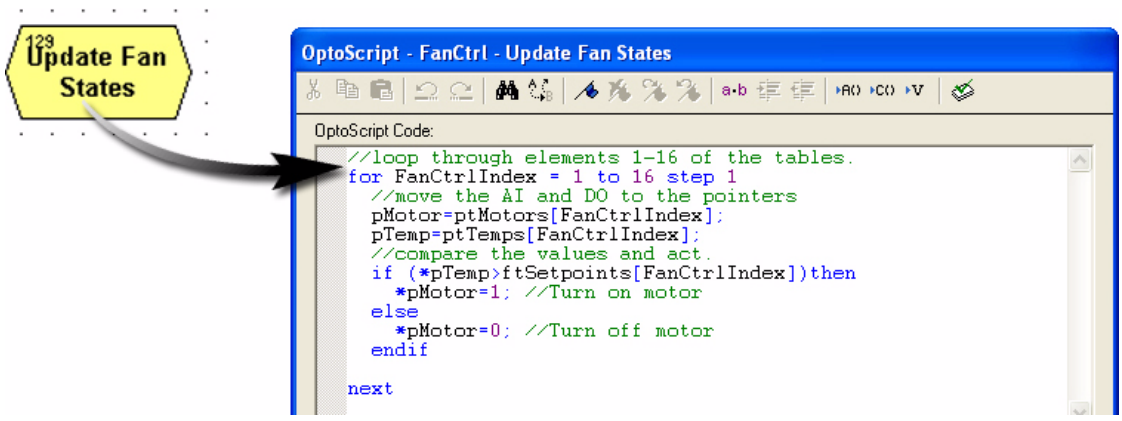
The second way uses pointers and indexing to accomplish the same result in a smaller space, using less control engine memory and fewer programming steps. The fan outputs and the temperature inputs are referred to by pointers. The setpoints, instead of requiring individual variables, are simply numbers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

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.)



Using OptoScript

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



Optimizing Throughput

See more information on throughput in Opto 22 form 1702, the *PAC Display User's Guide*.

Throughput can refer to communications between a PC and the control engine, or communications between the control engine and I/O. The following factors affect throughput for these two types of communication:

PC ↔ Control Engine	Control Engine ↔ I/O
Increasing host task frequency Using design efficiencies	Using I/O unit commands Using an I/O error-handling chart

In both types of communication, throughput is affected by control strategy design. The following sections discuss how to design your control strategy to maximize throughput.

Understanding PAC Control Multitasking

When SNAP controllers run PAC Control strategies, they can multitask, or run several tasks at once. The total number depends upon the controller you use:

- A SNAP PAC S-series controller can run up to 33 tasks (32 flowcharts plus the host task).
- A SNAP PAC R-series controller can run up to 17 tasks (16 charts plus the host task).

Total tasks include:

- Host task, which is a vehicle for communicating with a PC
- Powerup Chart
- Any other chart or subroutine in your strategy that is running or suspended. (A subroutine assumes the time slice of the chart that called it.)

The tasks are not actually run simultaneously; each task gets a small amount of time before the control engine moves to the next task in the queue. If a task is finished before its time is up, the control engine moves to the next task sooner. For example, a suspended chart uses no time, even though it is counted as a task.

Host Task

The host task functions as a slave, which means it never originates messages, but only responds to inquiries or commands. The host task runs by default, but it only uses time when there is communication to the controller.

The host task must be used to download new firmware to the control engine.

Optimizing PC to Control Engine Throughput

You can optimize throughput between the PC and the control engine by increasing the host task's frequency and by using design efficiencies in your strategy.

Increasing Host Task Frequency

If you minimize the number of tasks in the queue, the host task is done more frequently and all tasks get more time. To minimize tasks, combine processes so you have fewer charts. Here's how:

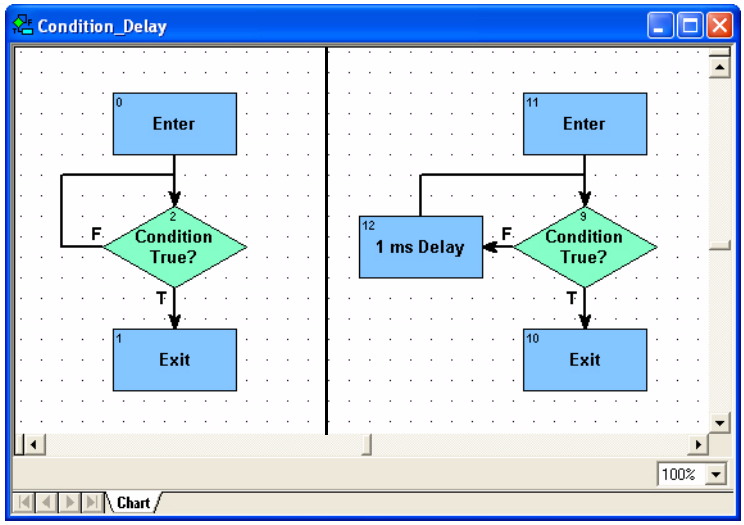
- Determine the processes that are time-critical. Put these processes in charts with looping logic that runs constantly.
- Use subroutines for any processes that recur within the strategy. A subroutine does not add an additional task; it assumes the time slice of the chart that called it. Make sure you check the status variable each time a subroutine is called to verify that the call was successful. See Chapter 12, "Using Subroutines," for more information.

Increasing Efficiencies in Your Strategy

A second way to optimize PC to Control Engine throughput is to increase efficiency in condition block loops.

If the condition block continually loops waiting for the condition to be true, the entire time slice for the chart is used up. However, you can build in a slight delay by using the command Delay (mSec) with a value of one. This command causes the chart to wait one millisecond before checking the condition again, and the rest of the chart's time slice is given up while it waits.

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



Ensuring Data Freshness for PAC Display

Remember to develop your PAC Control strategy in conjunction with your PAC Display project to help optimize performance. To ensure maximum throughput, the strategy should be coordinated with Refresh groups in PAC Display. For example, if PAC Display is using typical freshness values of one second, then the strategy should read all of the I/O at least once every second. (One efficient way to do so is to use I/O Unit commands. See the next section.)

Optimizing Control Engine to I/O Throughput

Throughput between the control engine and I/O is also affected by strategy design. You can take advantage of I/O unit commands to communicate with several I/O points at once, and you can use an error-handling chart to handle I/O errors efficiently.

Using I/O Unit Commands

I/O unit commands speed up communications between the control engine and I/O by using tables to read or write all points on an I/O unit at once, rather than reading or writing one point at a time. For example, using the command Move I/O Unit to Numeric Table is many times faster than using the Move command once for each point. The command Move Numeric Table to I/O Unit is many times faster than using Turn On or Turn Off for each point.

Handling I/O Errors Efficiently

If the control engine encounters an error when communicating to I/O, it disables communication to the I/O unit. Disabling communication ensures that control engine performance is maintained. If an I/O timeout error occurred and communication with the I/O unit was not disabled, throughput to the remaining I/O units would drop significantly while the control engine tried to communicate.

If you use an I/O error-handling chart, make sure there is a reasonable delay after each attempt to re-enable communication to the I/O unit. In addition, for debugging purposes it is helpful if the error-handling chart logs the error. Investigate and correct the root cause of any I/O unit communication error to maintain throughput.

Working with Control Engines

Introduction

This chapter shows you how to configure and work with control engines. See "Compatible Control Engines 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
Changing or Deleting a Control Engine 108
Inspecting Control Engines and the Queue 110
Downloading Files to the Control Engine 115

Configuring Control Engines

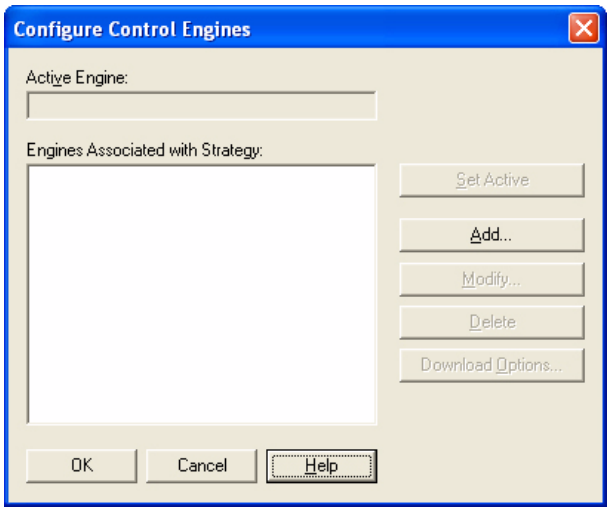
Before you can use a control engine to run a strategy, you must first define the control engine on your PC and then associate the control engine with your PAC Control strategy.

- 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 OS.) 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 102 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

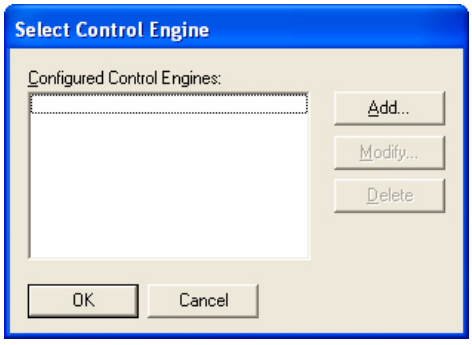
1. Choose one of the following:
   - **Using PAC Terminal:** Choose Start→Programs→Opto 22→PAC Project→Tools→PAC Terminal. From the Configure menu, choose Control Engines to open the Select Control Engine dialog box. Skip to step 3.
   - **Using PAC Control:** With a strategy open in Configure mode or Online mode, double-click the Control Engines folder on the Strategy Tree.
     You can also click the Configure Control Engines button in the toolbar, select Configure→Control Engines, or right-click an individual control engine on the Strategy Tree.

   The Configure Control Engines dialog box appears.



2. Click Add.

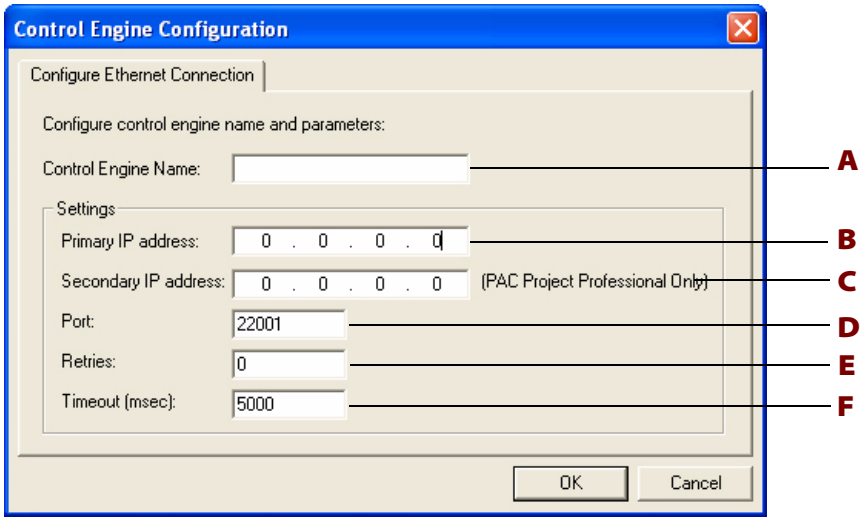
   The Select Control Engine dialog box appears.



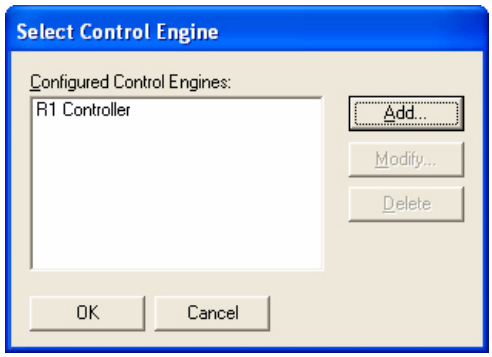
   This dialog box lists all the control engines configured on your system, whether or not they are associated with your strategy.

3. 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 click OK. Then skip to "Associating the Control Engine with Your Strategy" on page 102.
4. If the control engine you want is not in the list, click Add.

   The Control Engine Configuration dialog box appears.



5. Complete the fields as described in "Control Engine Configuration Dialog Box" below.
6. Click OK.
7. In the Select Control Engines dialog box, the control engine appears in the list.



8. In the Select Control Engine dialog box, highlight the control engine you want to associate with the strategy. Click OK. Continue with "Associating the Control Engine with Your Strategy" on page 102.

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—*IP Address* Enter the IP address of the control engine in decimal notation (for example, `192.9.200.24`).

C—*Second IP* (PAC Control Pro only) To configure Ethernet link redundancy, enter a secondary IP address. The secondary IP address can be the second Ethernet interface on a SNAP PAC controller, or a separate controller running a strategy designed to respond if the primary control engine is unavailable. Note that both IP addresses use the same port number. See "Configuring Ethernet Link Redundancy" on page 5-108 for more information.

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.

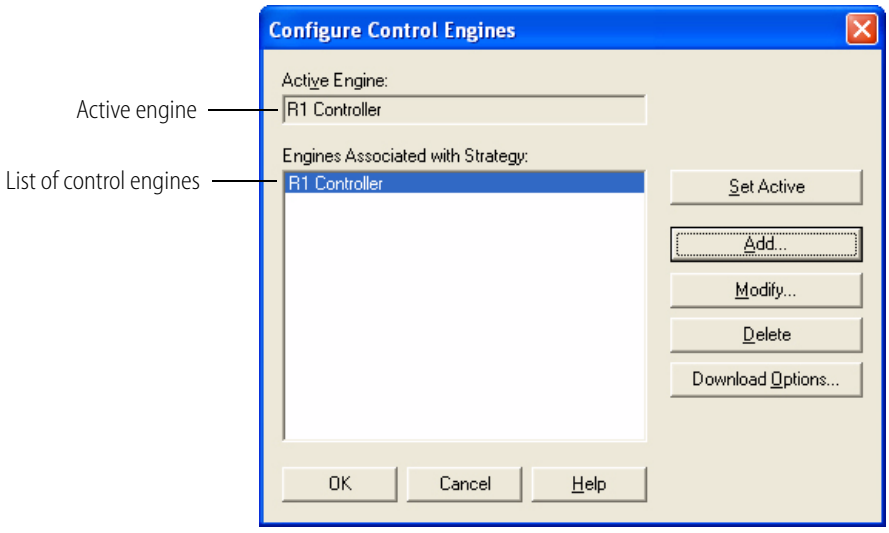
F—*Timeout* Enter the timeout value in milliseconds. Timeout value is the length of time PAC Control tries to establish communication through the port. If it fails, it tries again as many times as specified in D. Any positive integer is valid. For Ethernet, 3–5 seconds (3000–5000 milliseconds) is a good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

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



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 at any time. 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 I/O units are on the other.

To segment networks, start by assigning a secondary IP address in PAC Manager, following instructions in 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.

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.

Using Ethernet Link Redundancy in PAC Control

For Ethernet link redundancy between **PAC Control Professional** running on a PC and the control engine you can configure a secondary control engine IP address in case communication to the primary address fails. The secondary address can be the second Ethernet network interface on a SNAP PAC controller or a separate controller. If it is the second Ethernet network interface on a SNAP PAC controller, you must assign the secondary IP address in PAC Manager (using Status Write) in order to communicate with it. See the *PAC Manager User's Guide* for instructions.

NOTE: If you use a separate controller for link redundancy, be aware that coordinating strategy data on two controllers can be difficult, and no automatic method for doing so is available. Using a separate controller for monitoring over a redundant link is usually less difficult to coordinate than using a separate controller for control over a redundant link.

The two Ethernet interfaces on a SNAP PAC controller are ideal for adding redundant Ethernet network links to your control system. A PAC Display Professional project or OptoOPCServer, for example, can be set up to use the primary and secondary 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 client application will automatically shift to the secondary address; if the secondary address fails, then the primary address becomes the active address and it will automatically try the primary address again. If communication to the active address fails and the control engine is not able to switch to the other address, then communication to the I/O unit will become disabled. Communication is tested on each valid address that is enabled (any address other than 0.0.0.0) when a strategy is started or when communication to an I/O unit is changed from disabled to enabled.

NOTE: You can also manually change the address from within the client application. See the OptoOPCServer or PAC Display user's guide for details.

PAC Control Professional also provides for Ethernet link redundancy from the controller to SNAP PAC I/O units. You can configure a secondary I/O unit IP address in case communication with the primary address fails. The secondary address can be the second Ethernet network interface on a SNAP PAC R-series controller, or a separate I/O unit. (If you use a separate I/O unit, you may find that wiring is problematical.) See "Commands for Ethernet Link Redundancy" on page 284 for more information on primary and secondary I/O units.

System Architecture for Ethernet Link Redundancy

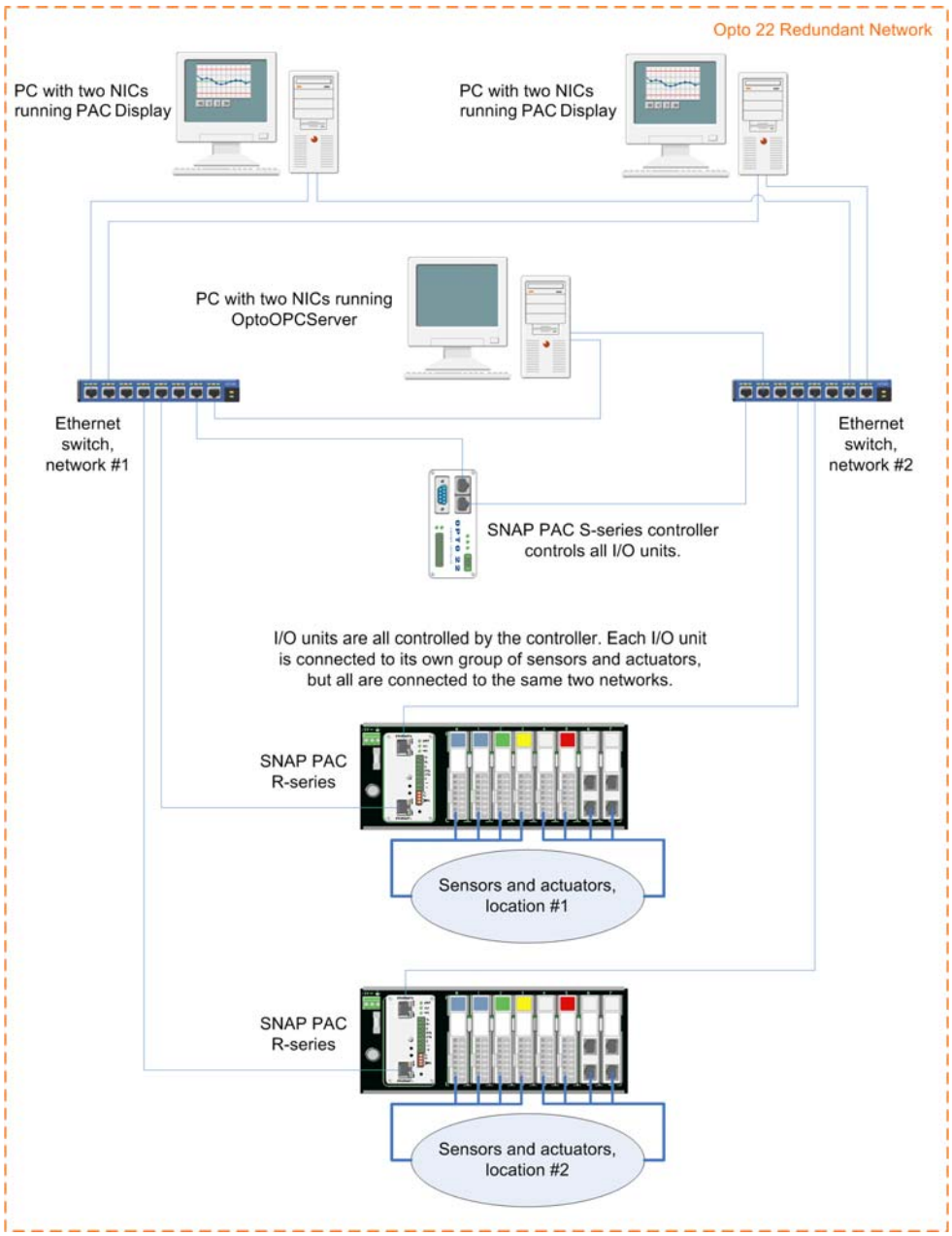
You can set up your system in several ways to take advantage of the link redundancy capability in PAC Control. The following diagrams show some of those ways. For additional information, see the user's guides for the SNAP PAC controllers. For details on how PAC Display and OptoOPCServer work with link redundancy, see their user's guides.

Ethernet Link Redundancy

In this example, the primary concern is that the Ethernet network may need maintenance or may fail, leaving the computer running OptoOPCServer, the PC running PAC Display Professional, 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).

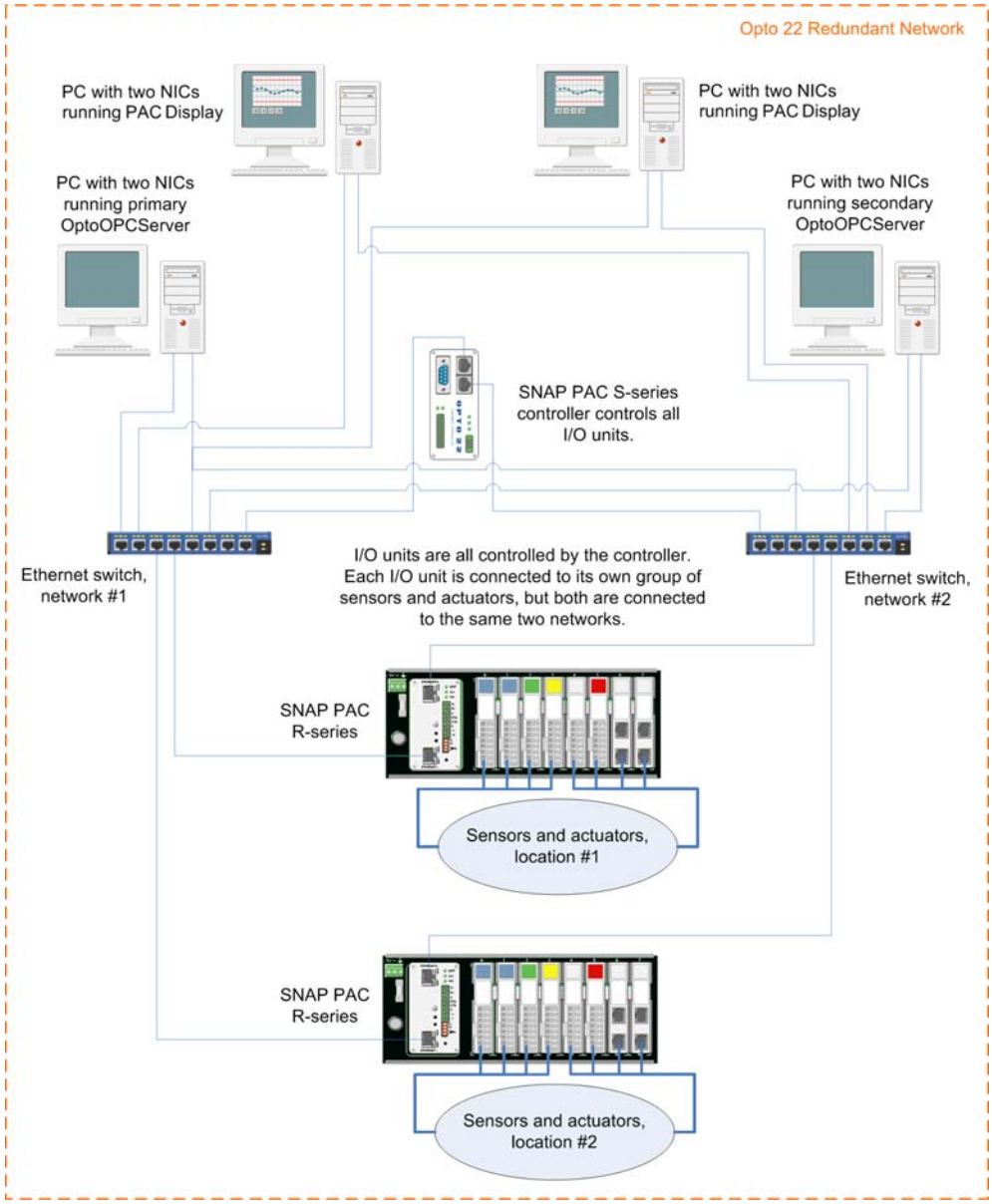
If one network needs maintenance, for example to replace a switch, you can safely shut it down using the Set Target Address commands in PAC Control and options in PAC Display Professional and OptoOPCServer.



Ethernet Link, Computer, and Software Redundancy

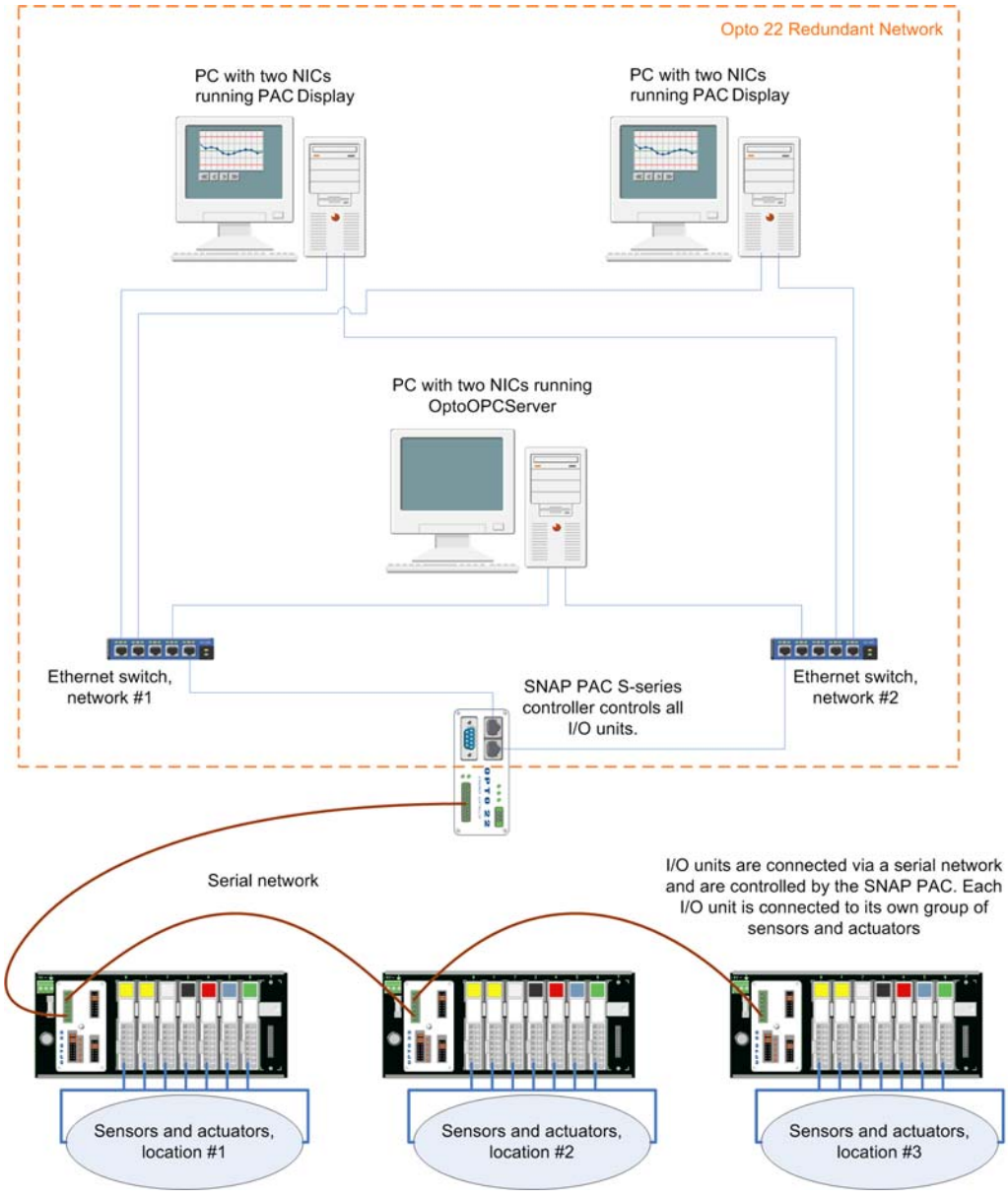
The second example of link redundancy illustrates concern not only about the stability of the Ethernet network, but also about the computers and the software run on them. Any of these things—the network, a computer, or the OptoOPCServer or PAC Display Professional software running on the computer—may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



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 and a separate serial control network. Again, note that the computers all have two NICs.



Configuring Ethernet Link Redundancy

Ethernet link redundancy to the control engine is easy to configure. First, make sure the secondary IP address has been assigned 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). It is not necessary to separately define the second controller on your PC.

To configure link redundancy from the control engine to I/O units, make sure the secondary IP address has been assigned in PAC Manager. Then enter both the primary and secondary IP addresses when you configure the I/O unit (see "Adding an I/O Unit" on page 124).

Using Strategies with Link Redundancy

Although clients of PAC Control strategies such as PAC Display and OptoOPCServer automatically shift to the secondary address if 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, you must be in Configure or Online mode. Follow the steps in "Changing the Control Engine that Receives the Downloaded Strategy" on page 109.

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.

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 109
- "Removing a Control Engine's Association with a Strategy" on page 109
- "Deleting a Control Engine from Your PC" on page 110

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 the PAC Terminal utility.

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:** Choose Start→Programs→Opto 22→PAC Project→Tools→PAC Terminal. From the Configure menu, choose Control Engine.

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.

Pro

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 the PAC Terminal utility. Deleting the control engine removes its definition only on the PC you are using.

1. Choose Start→Programs→Opto 22→PAC Project→Tools→PAC Terminal.
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 the PAC Terminal utility:

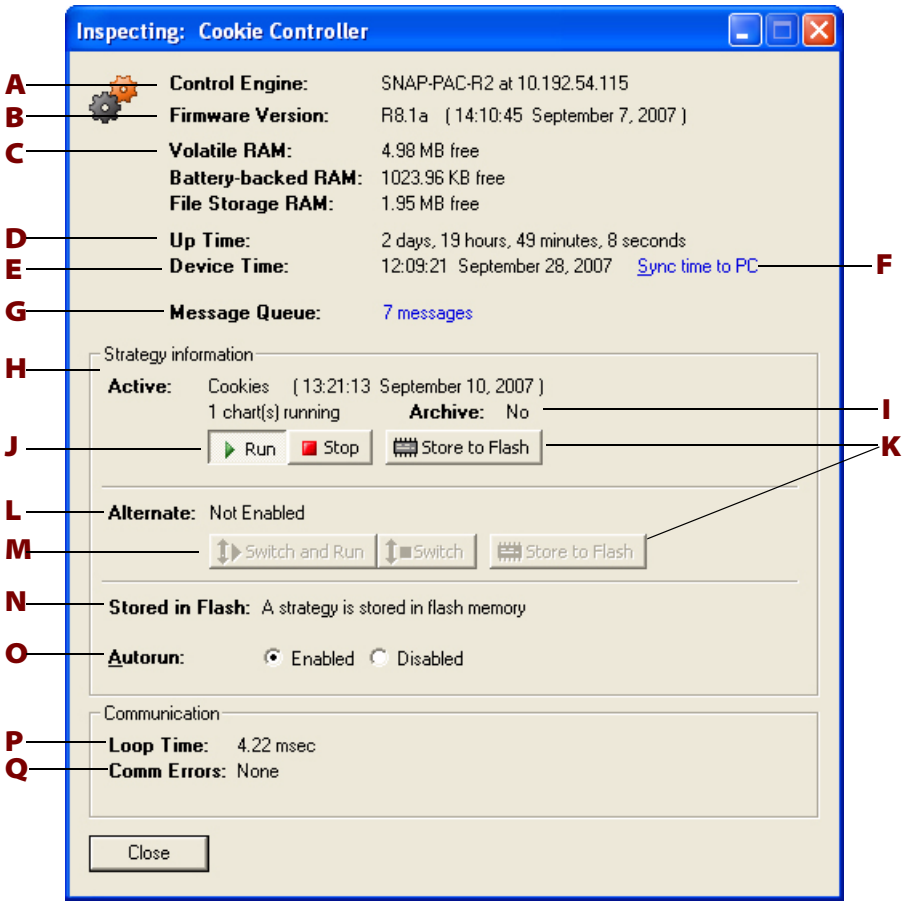
- "Inspecting Control Engines in Debug Mode" (below)
- "Viewing the Message Queue" on page 112
- "Inspecting Control Engines from the PAC Terminal Utility" on page 115

Inspecting Control Engines in Debug Mode

With the strategy running in Debug mode, click the Inspect Control Engine button in the toolbar.

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, or choose Control Engine→Inspect.

The Inspecting dialog box opens.



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 108 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 in the dictionary, 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.

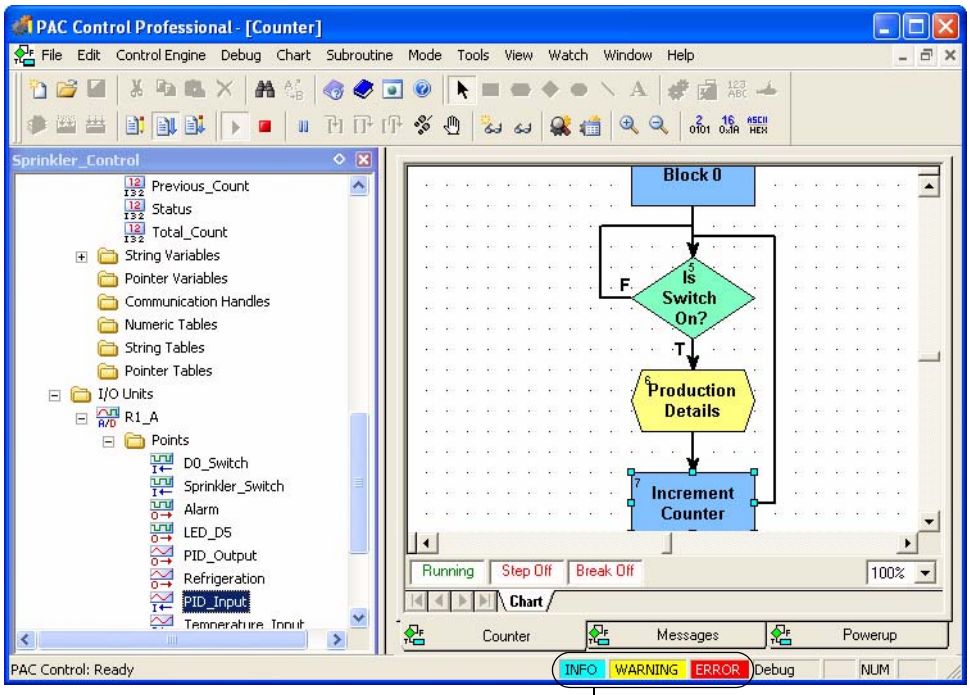
O—*Autorun* Click to indicate whether the strategy should automatically run when the control engine is restarted. The strategy must be stored to flash to autorun. See Store to Flash above"Saving a Strategy to Flash" on page 177.

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

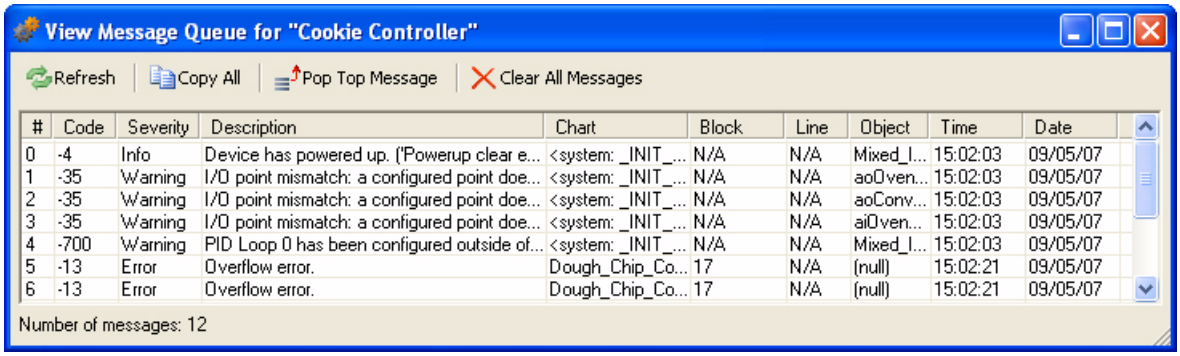
The message queue holds error, information, and warning messages. All may be helpful in troubleshooting. When a message is placed in the queue, a blue INFO, yellow WARNING, or red ERROR box appears in the PAC Control status bar, as shown in the following diagram.



Messages have been placed in the queue.

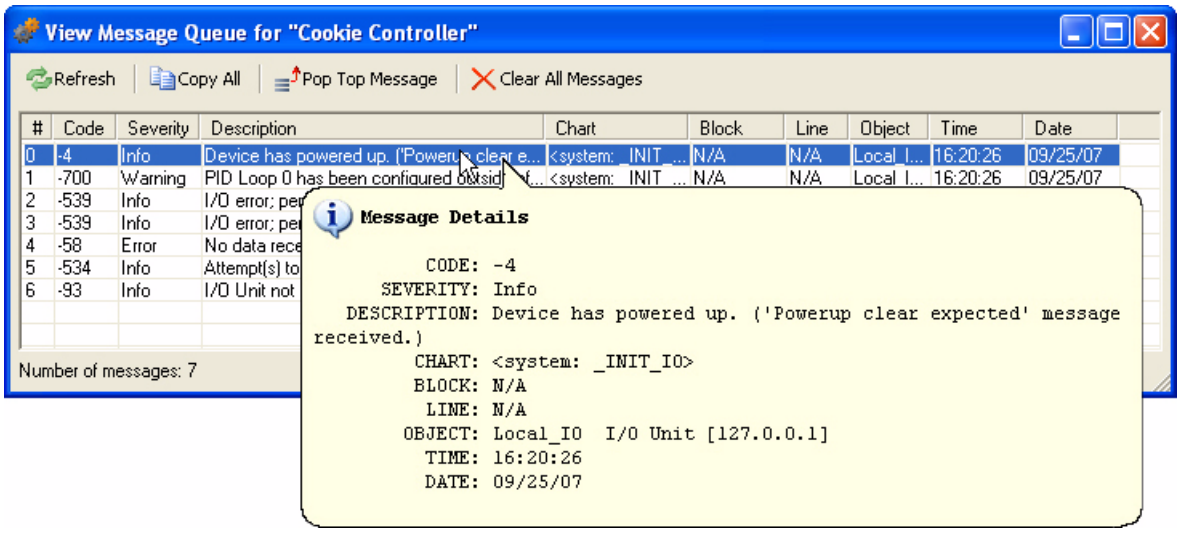
To see the message queue,

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

2. To see all the information in the column, drag the edge of a column heading . See "Message Queue Information" below.
3. To see all of the information for one message, double-click the message.



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 369) 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.

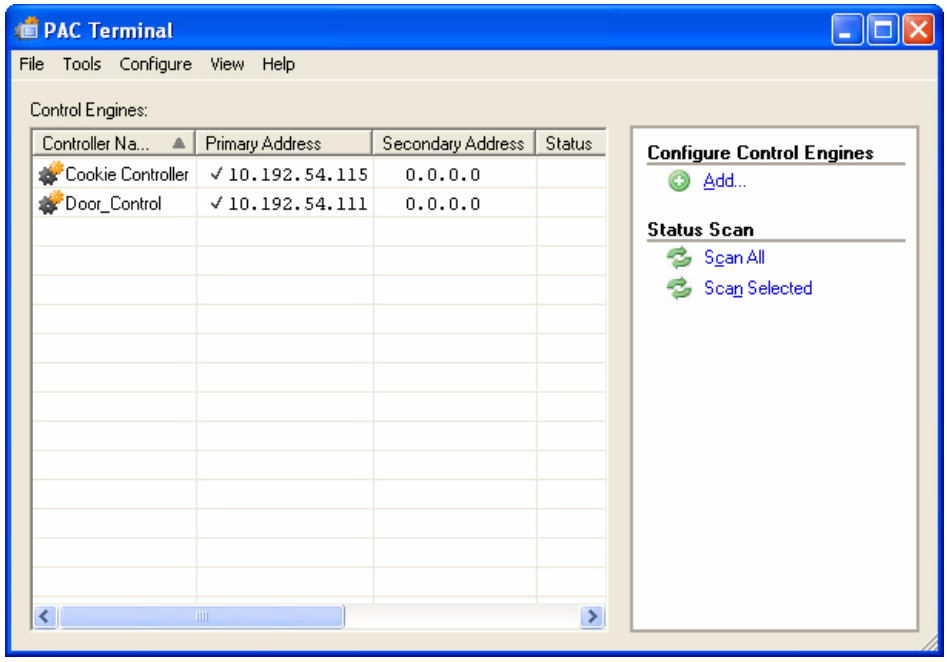
Time and Date When the error occurred.

Inspecting Control Engines from the PAC Terminal Utility

You can also inspect control engines from the PAC Terminal utility.

1. Click the Windows Start menu and select Programs→Opto22→PAC Project→Tools→PAC Terminal.

   The PAC Terminal window appears.



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 110.

Downloading Files to the Control Engine

This section discusses how to archive strategies and download Forth files directly related to a strategy, such as library or initialization files. For information on using the brain's file system to store data and manipulate it within your strategy, see page 268.

Archiving Strategies

Archiving strategies on the control engine provides a backup in case original strategy files on the computer are lost. 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.

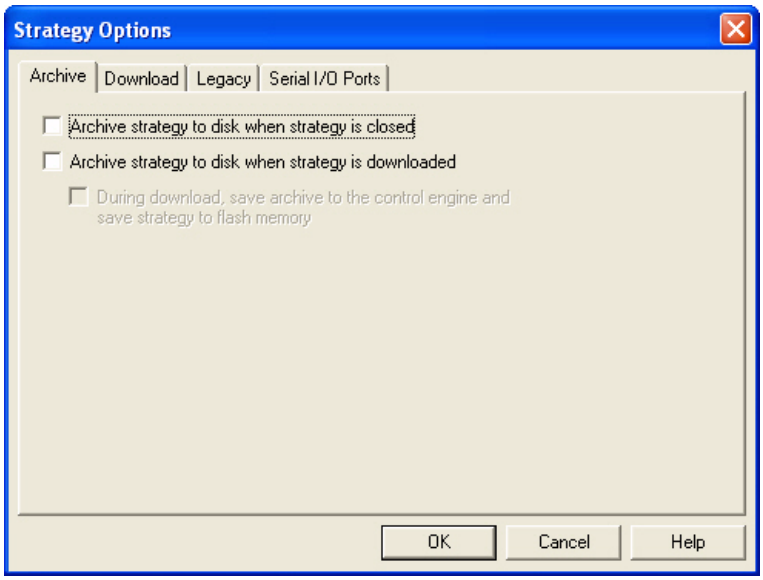
Archiving to the Control Engine

When you archive a strategy to the control engine, you are placing the zipped file in battery-backed RAM. If power to the control engine is lost, the archive is still there. 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.

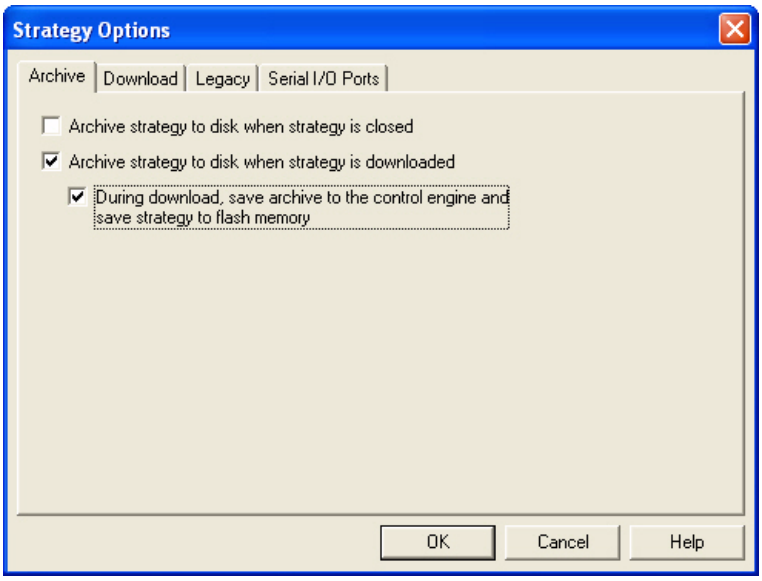
Make sure the control engine has sufficient memory available to store the archive file. Since only one strategy can be on the control engine at any time, only the latest archive for that strategy is on the control engine. Other archives are erased during strategy download.

Follow these instructions to archive a strategy to the control engine:

1. In PAC Control, choose File→Strategy Options.



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."



3. Click OK.

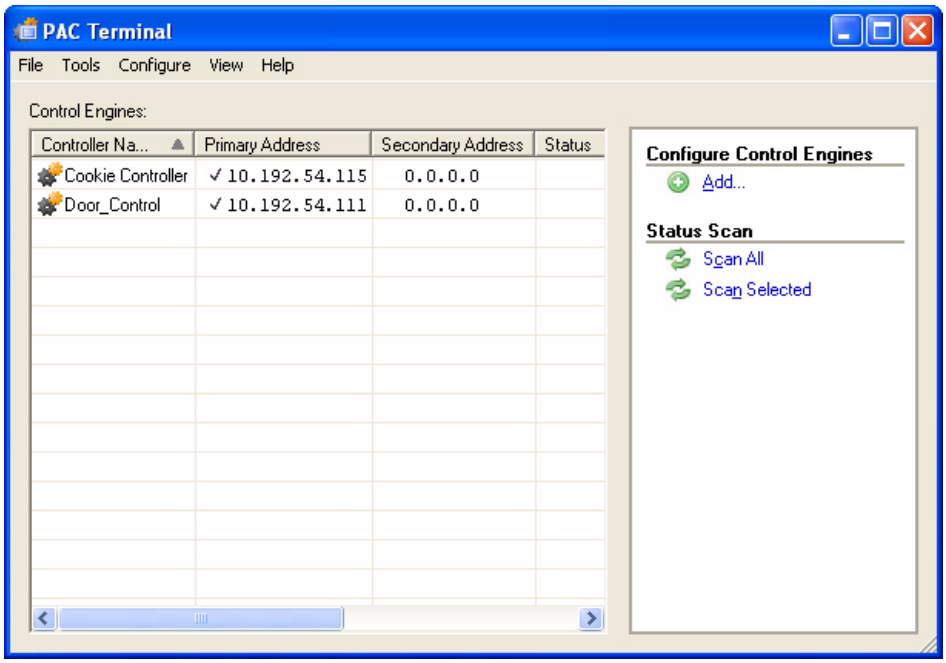
   The strategy will be archived to the computer and to the control engine when it is downloaded. Any archive already on the control engine will be replaced by the new archive. In addition, the strategy will be saved to flash memory, so it will still be available if power to the controller is turned off.

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. Click the Windows Start menu and choose Programs→Opto22→PAC Project→Tools→PAC Terminal.

   The PAC Terminal window appears.



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.

Downloading Files Without Opening PAC Control

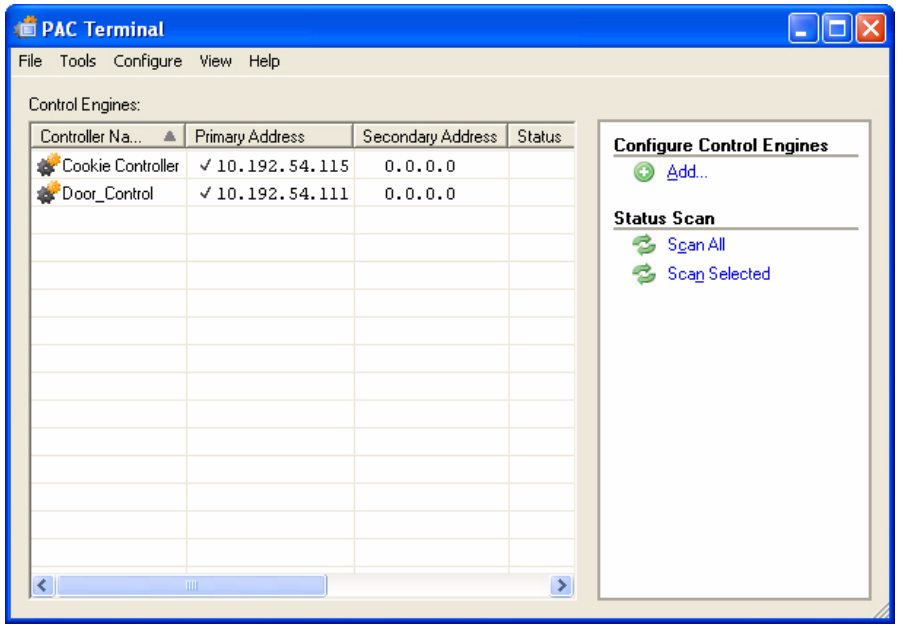
Using the PAC Terminal utility, you can download PAC Control strategies or Forth files related to the strategy, such as library or initialization files, directly to a control engine, without having to open each program.

(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 268.)

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 Tables During Strategy Download" on page 240.

1. Click the Windows Start menu and choose Programs→Opto22→PAC Project→Tools→PAC Terminal.

   The PAC Terminal window appears.



2. Right-click the control engine and choose Download→Forth Files 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.

Working with I/O

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.

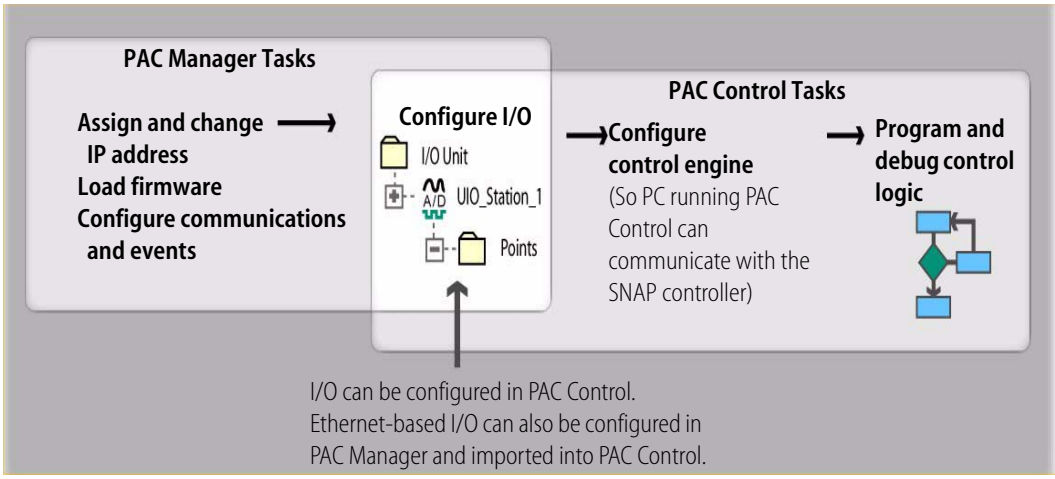
In this Chapter

Choosing a Configuration Tool 121
Adding an I/O Unit 124
Adding I/O Points 129
Configuring a Serial Module 137
Changing Point Configuration 138
Configuring PID Loops 142
Inspecting I/O in Debug Mode 152
Inspecting and Tuning PID Loops 156
Using Watch Windows for Monitoring 168

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 graphic on the next page compares their functions.



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.

For most I/O units, if you are already in PAC Control, configuration is easier there and you can use the loopback IP address for 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 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.

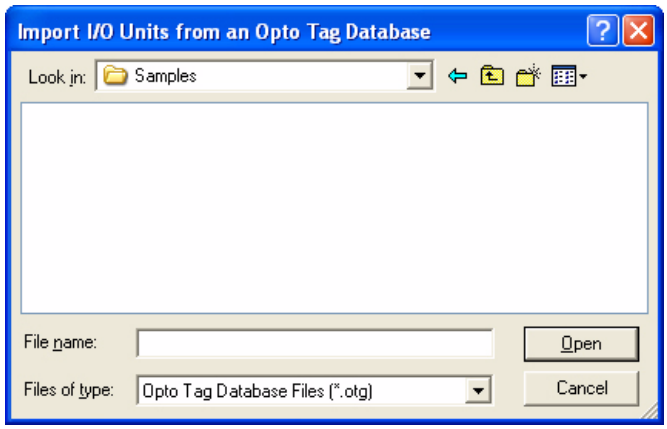
If you use PAC Manager, follow instructions in Opto 22 form 1704, the *PAC Manager User's Guide*. When you have finished configuration and saved the configuration file, you can import it into PAC Control following the steps in "Importing I/O Configuration into PAC Control" below.

If you use PAC Control, follow the steps beginning with "Adding an I/O Unit" on page 124.

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 124. To tune PID loops, see "Inspecting and Tuning PID Loops" on page 156.

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 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.

Adding an I/O Unit

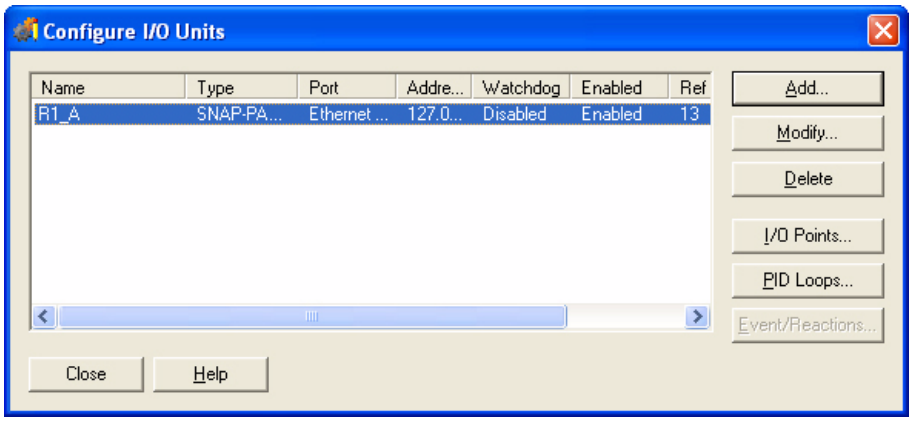
In PAC Control, the term *I/O unit* usually refers to a mounting rack with a brain or brain board and up to 16 I/O modules attached.

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.

1. Make sure the strategy is open and in Configure mode. On the Strategy Tree, double-click the I/O Units folder.

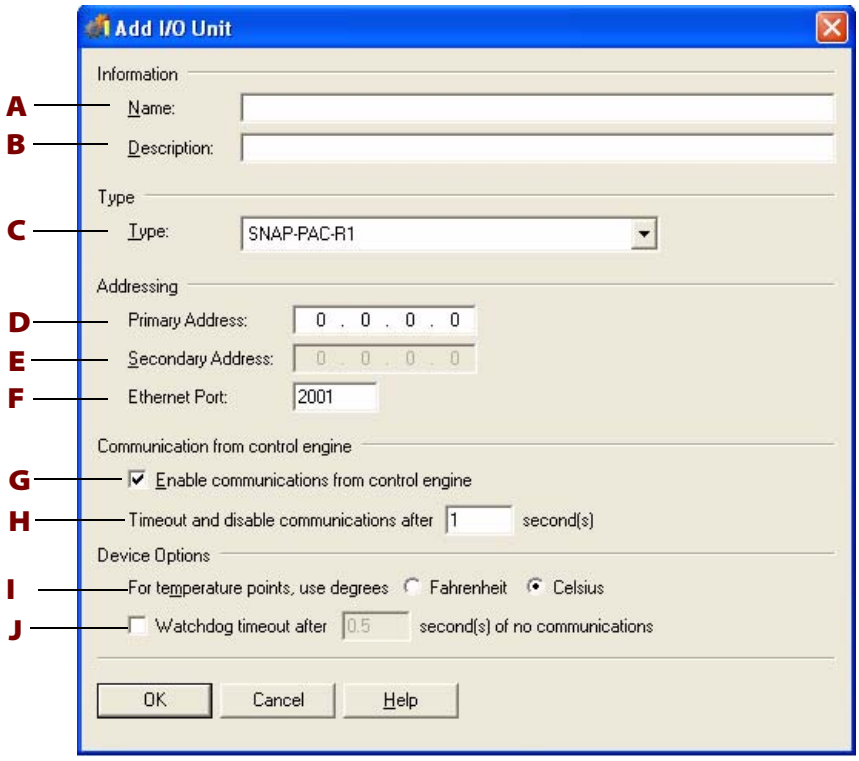
   You can also click the Configure I/O button on the toolbar, or select Configure→I/O.

   The Configure I/O Units dialog box opens, showing all configured I/O units.



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.



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 152.

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. Brain model numbers are listed in parentheses to help you determine which type to choose. 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), use `127.0.0.1` as the IP address. This is a loopback address; it 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—PAC Control Professional 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 binary/CRC settings on your hardware are required.

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 length of time the control engine should wait for a response when communicating with this I/O unit. Default is 1 second. **CAUTION:** For Ethernet, two retries are built in. 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 128.

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.

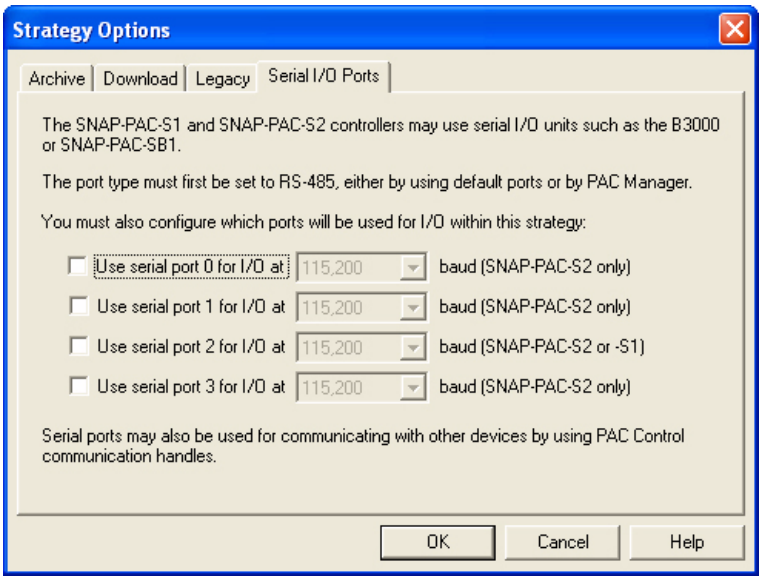
With a Watchdog, the I/O unit monitors activity on the port. If no communication is received for the specified interval, the unit assumes a Watchdog state. All selected outputs will then immediately go to a specified value, as configured in the Watchdog field of the Add Analog Point or Add Digital Point dialog boxes. (See "Adding I/O Points" on page 129.)

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 127.

If the default baud rate shown in the Add I/O Unit dialog box 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.

1. Make sure the strategy is open and in Configure mode.
2. Choose File→Strategy Options, then click the Serial I/O Ports tab.



3. 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

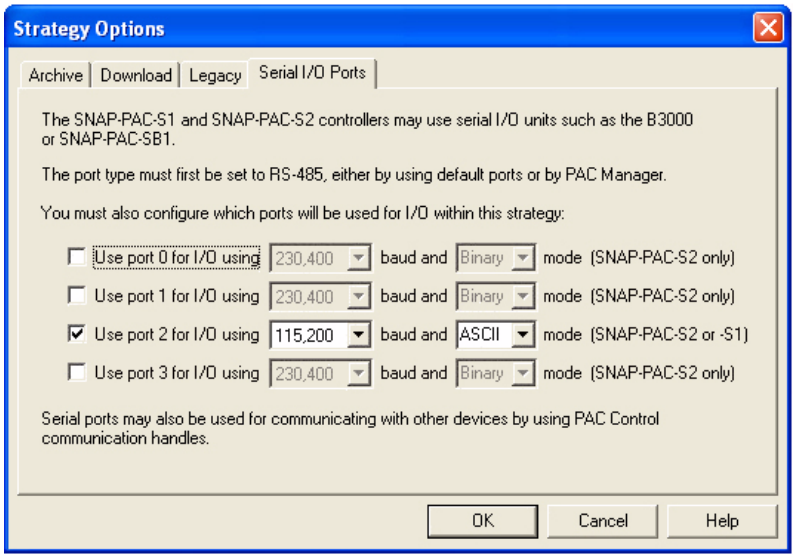
NOTE: This section is for PAC Control Professional users. If you have the Basic version, see "Changing the Baud Rate for Serial I/O Units" on page 126.

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. Choose File→Strategy Options, then click the Serial I/O Ports tab.



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 only support 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, R2, EB1, or 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 an 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 125.
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.

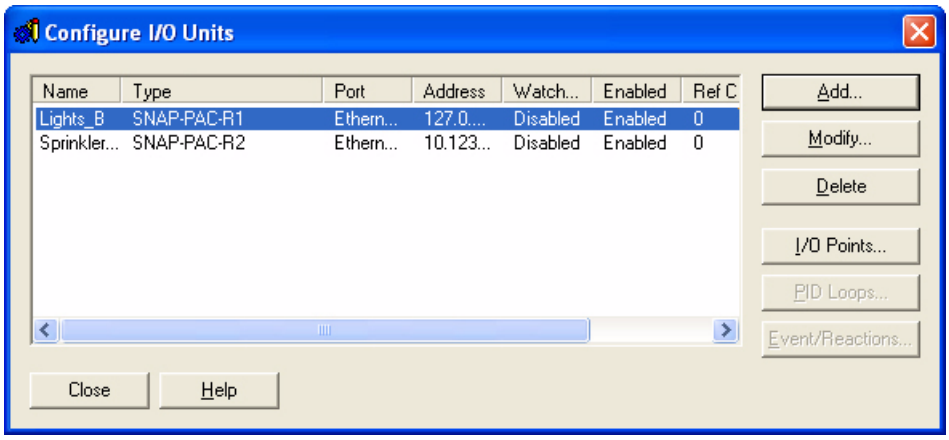
Adding I/O Points

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 124.

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.

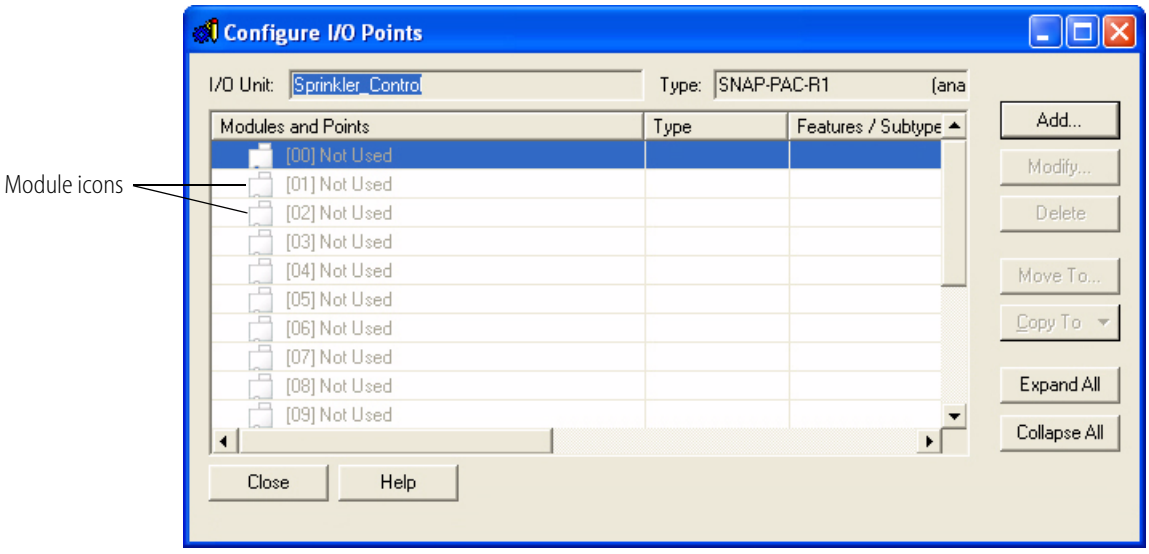


2. Highlight the I/O unit the points are on, and click I/O Points.

   The Configure I/O Points dialog box appears.

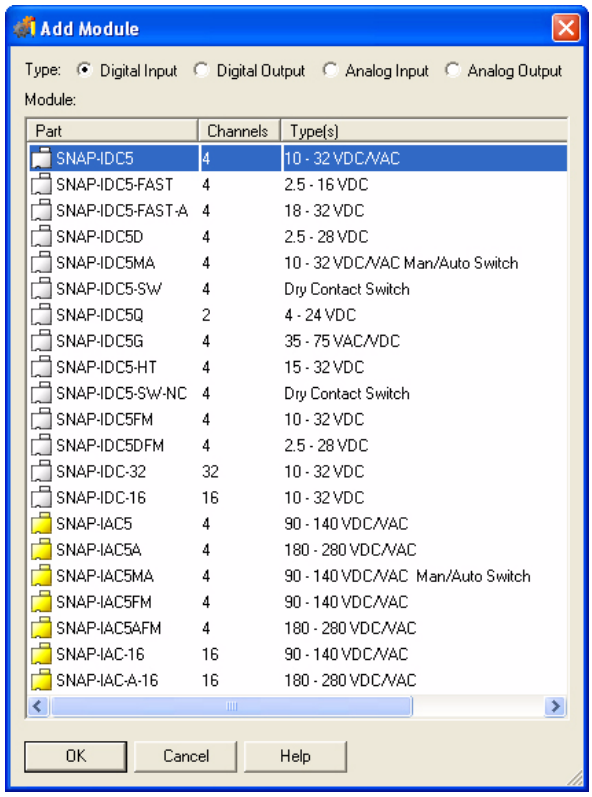
3. (Pro only) **If you are using a non-SNAP I/O unit (*not* including the B3000),** double-click the channel you want to use. Skip to step 8.

4. **If you are using a SNAP I/O unit (including the B3000),** notice the module icons in the dialog box.



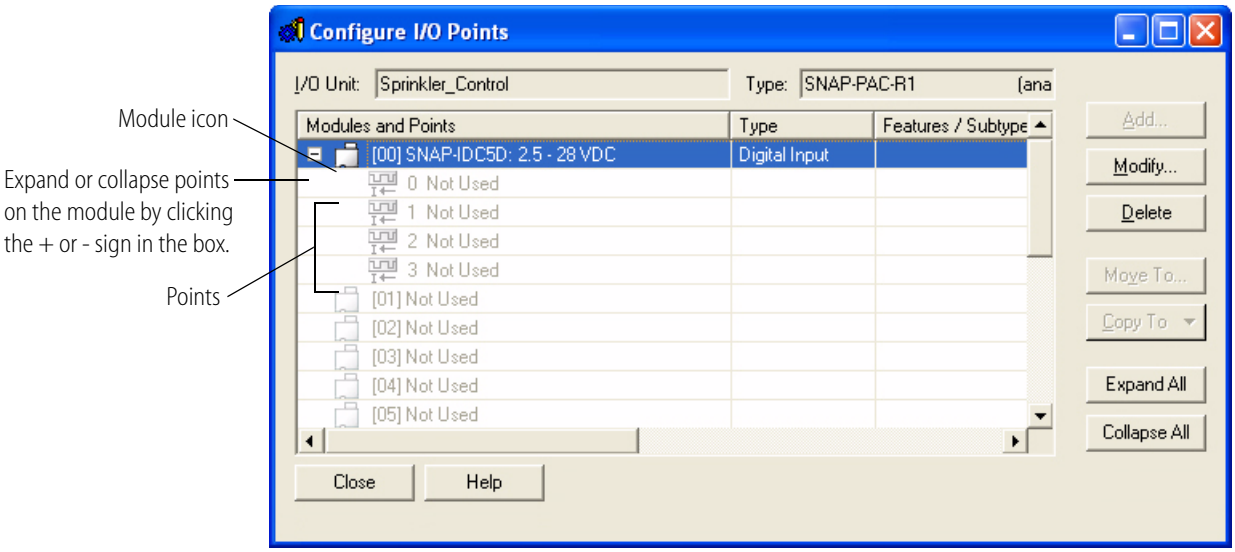
NOTE: This example shows a SNAP PAC I/O unit, so both digital and analog modules can be configured on the same rack.

5. Highlight the number that represents the module's position on the rack, then click Add.



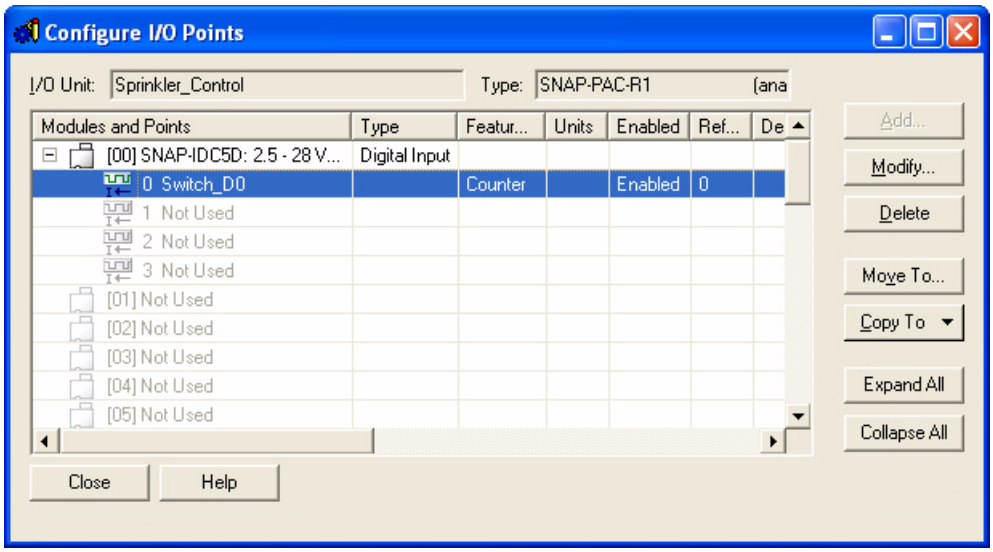
6. In the Add Module dialog box, choose the module type and then the exact module from the lists. Click OK.
7. In the Configure I/O Points dialog box, click the plus sign next to the new module to expand it.

   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.



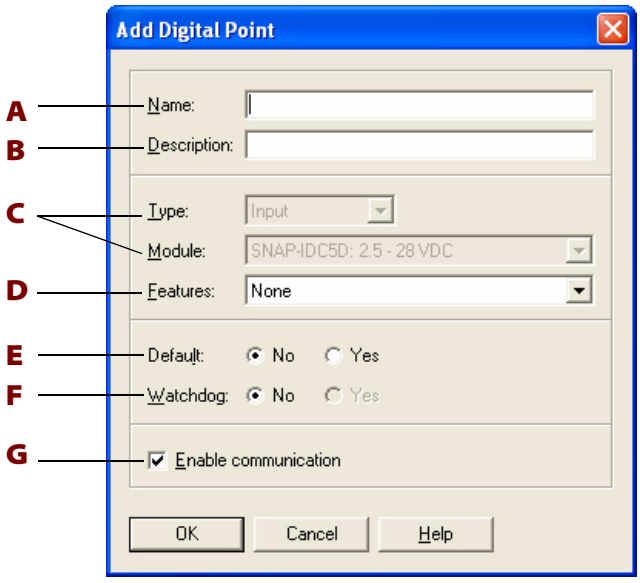
8. Highlight the point you want to configure and click Add.
9. Complete the fields as described in "Add Digital Point Dialog Box" below.
10. When you have completed the fields, click OK.

    The new point appears in the list. Here is an example of how it might look on a SNAP I/O unit:



NOTE: If you need to add several similar points, see "Copying a Configured I/O Point" on page 139.

Add Digital Point Dialog Box



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, 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.

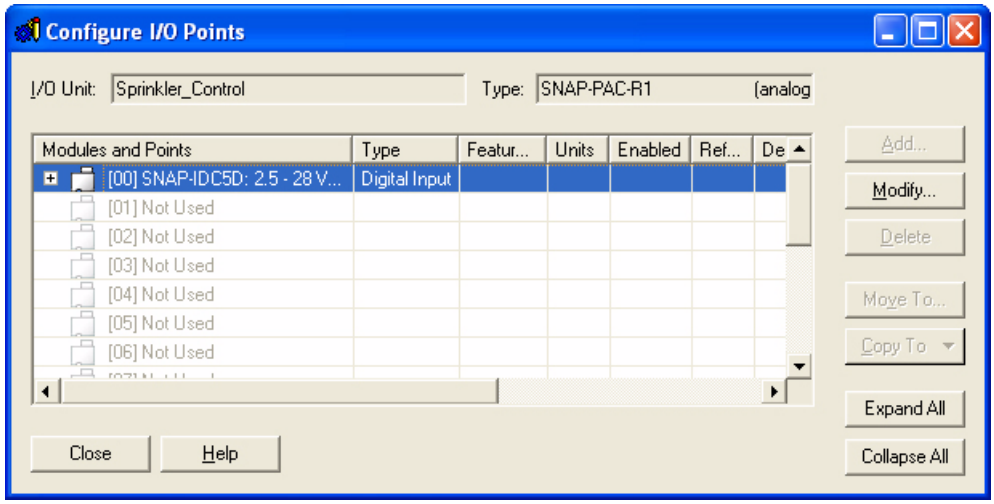
G—*Enable Communication* Select whether you want communication to this I/O point enabled or disabled on startup. Enabled is the default.

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.

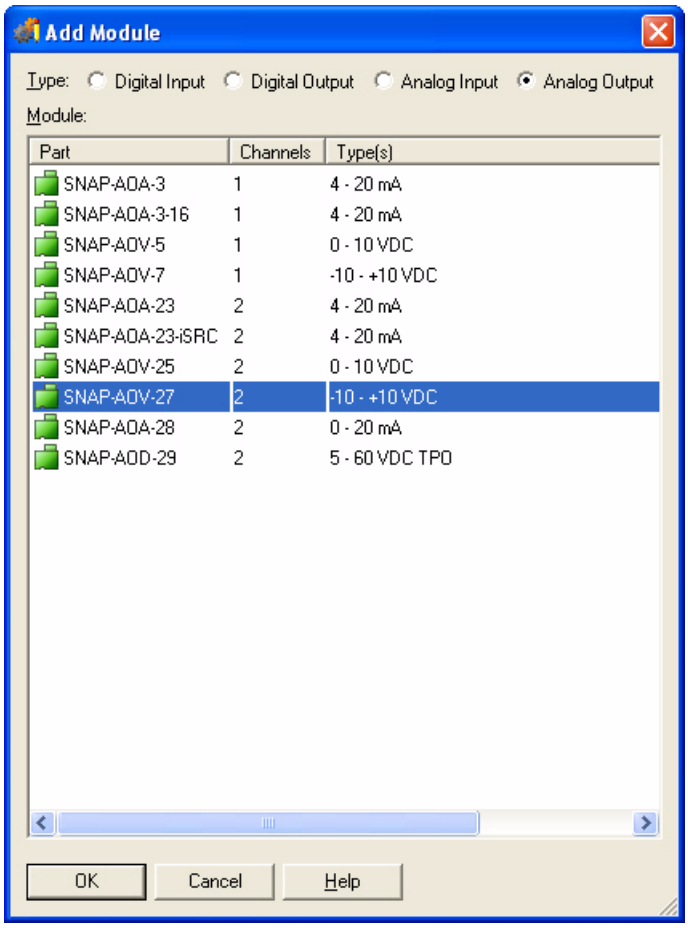
   The Configure I/O Points dialog box appears.

3. (Pro only) **If you are using a non-SNAP I/O unit (*not* including the B3000),** 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)**, notice the module icons in the Configure I/O Points dialog box.



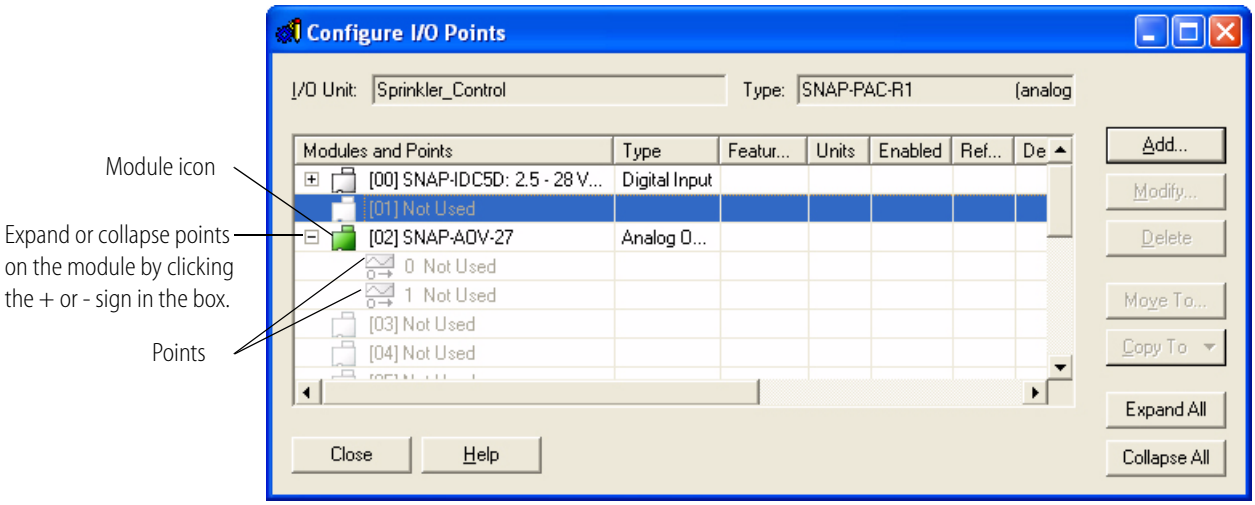
NOTE: In this example, a digital module has already been added in position zero. This example shows a SNAP PAC I/O unit, so both digital and analog modules can be configured on the same rack.

5. Highlight the number of the analog module's position on the rack, then click Add.



6. In the Add Module dialog box, choose the module type and then the exact module from the lists. Click OK.

7. 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.

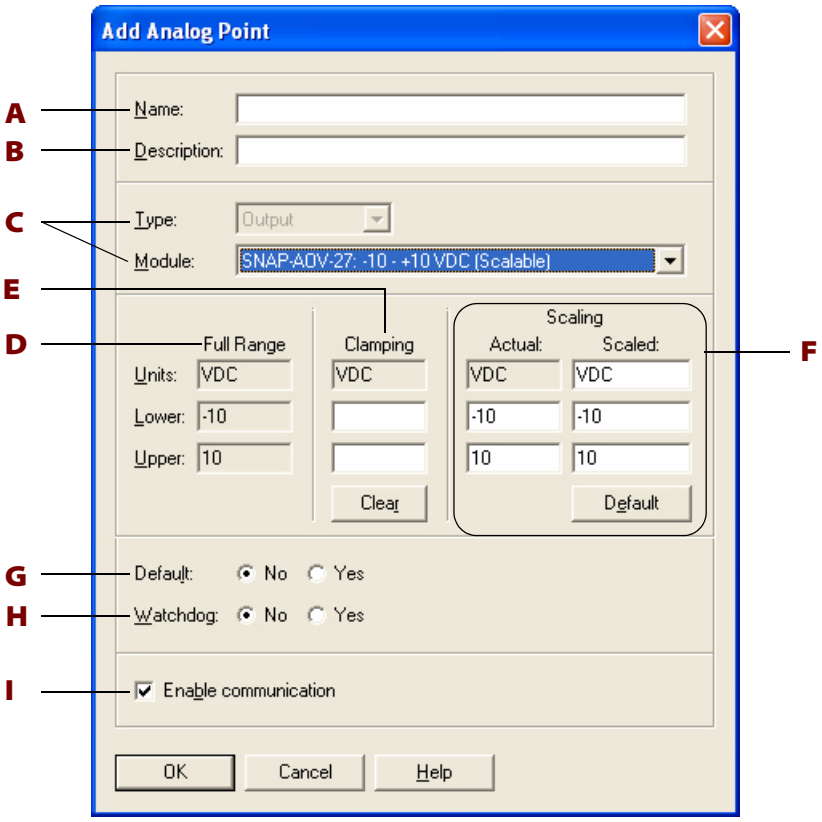


8. Highlight the point you want to configure and click Add.
9. Complete the fields as described in "Add Analog Point Dialog Box" below.
10. When you have completed the fields, click OK.

    The new point is added.

NOTE: If you need to add several similar points, see "Copying a Configured I/O Point" on page 139.

Add Analog Point Dialog Box



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.

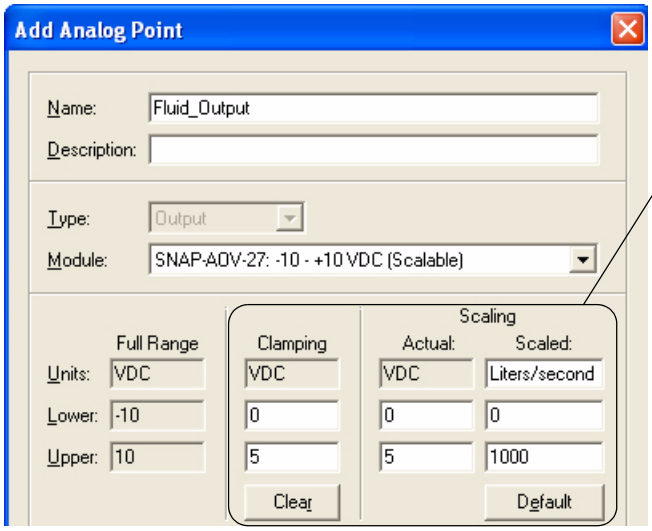
C—*Type/Module* For non-SNAP I/O units, choose the module type and the exact module from the drop-down lists.

For SNAP I/O units: Type and module are already filled in for you. You may be able to choose a different range or a scalable module from the drop-down list.

D—*Full Range* Full range and units for this module. If the module is scalable, use **F** to change scale.

E—*Clamping* (Outputs only—optional) Enter upper and lower clamp if necessary to limit output to the device attached to the point. If fields are left empty, no clamp is applied. To empty the fields, click Clear.

F—*Scaling* (Scalable modules only—optional) 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:



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 can be any floating point value as long as the upper value is higher than the lower value. Note that inputs typically have under-range and over-range capability, which means you can specify a lower or upper value beyond the standard value. Outputs do not have under-range or over-range capability.

To return the units and upper/lower values to the defaults for the module, click Default.

G—*Default* To set the initial values for this I/O point when the strategy is run, click Yes and define the value. To leave the internal/external values at their last state, click No.

If communication to the point is disabled, only the internal values (IVALs) are 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 point is disabled.

H—*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.

I—*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 module. See the *PAC Manager User's Guide* (Opto 22 form 1704) for instructions.

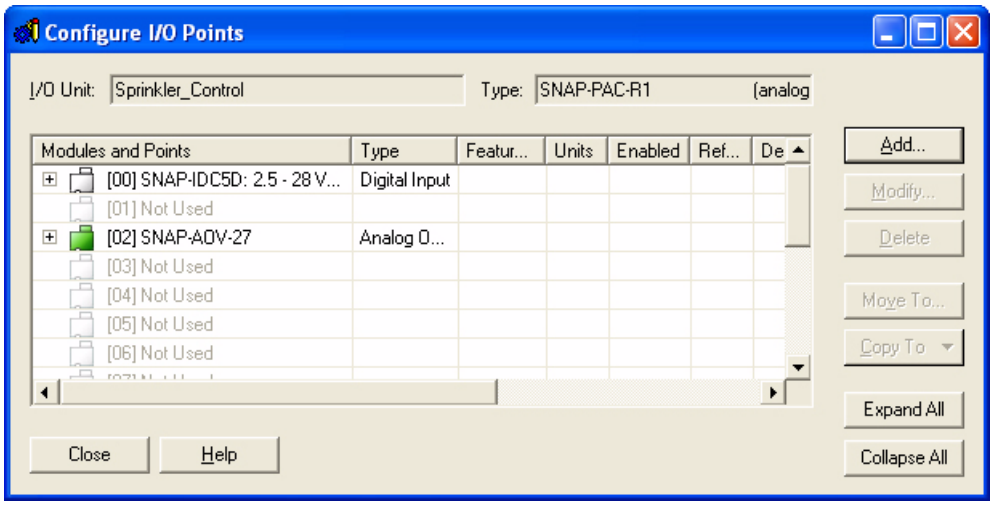
Changing Point Configuration

Moving a Configured I/O Point

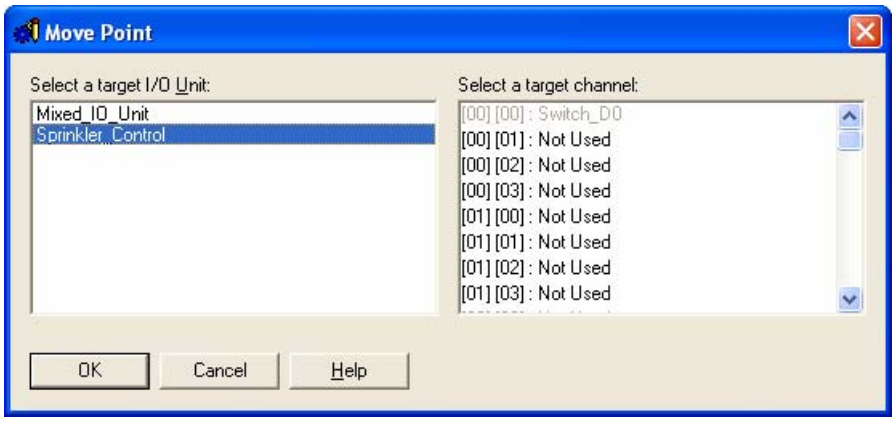
You can move most configured I/O points to an empty point on the same I/O unit or on a different unit.

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 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:



3. If necessary, expand the modules by clicking Expand All.
4. Highlight the point you want to move and click Move To.



5. In the Select a target channel area of the Move Point To dialog box, highlight the location you are moving the point to. Then click OK.

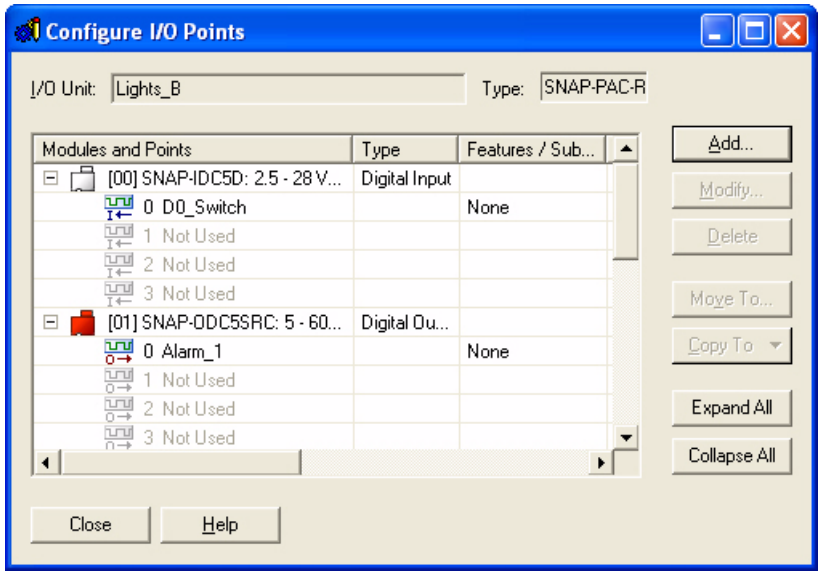
   You return to the Configure I/O Points dialog box, and the point has been moved.

Copying a Configured I/O Point

If you have several points 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 other point on the same or a different I/O unit.

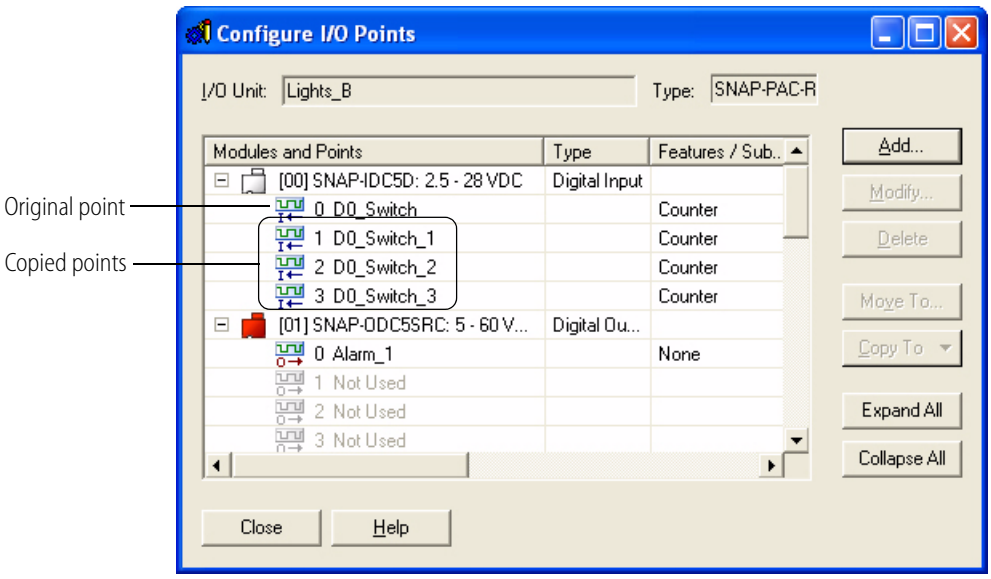
1. With the strategy open in Configure mode, double-click the I/O Units folder on the Strategy Tree to open the Configure I/O Points dialog box.
2. Click Expand All so you can see the points.



3. Highlight the point you want to copy and click the Copy To button. From the popup menu, choose one of the following:

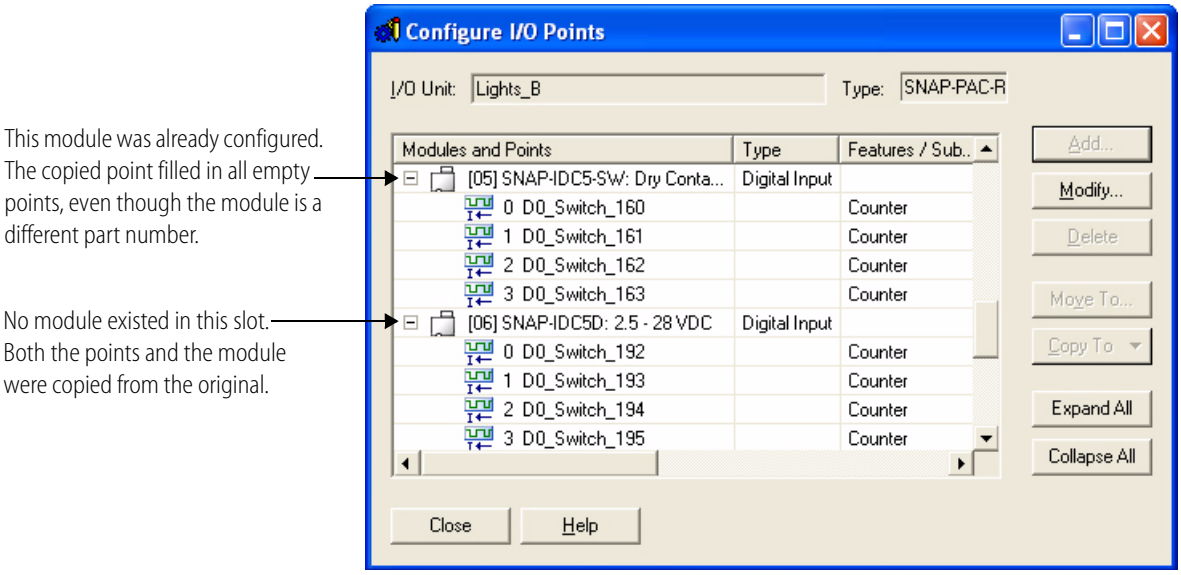
   To copy the point to fill empty points on the module, choose Fill In Module.

   The point is copied to the other empty points on the same module.



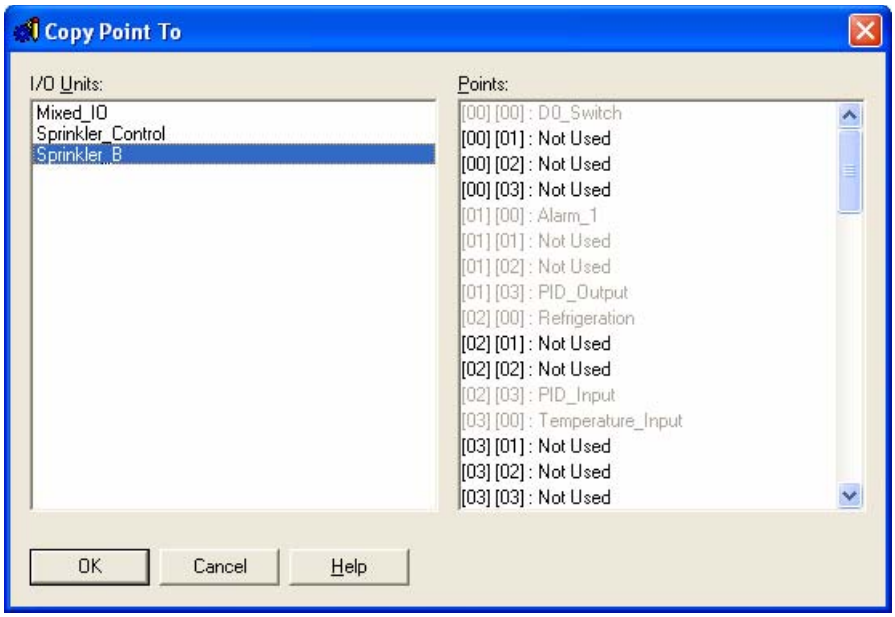
To copy the point to fill all empty points on the I/O unit, choose Fill In I/O Unit.

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 to all compatible slots without configured modules.



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.



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.

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 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, open PAC Manager and use Inspect mode to change the PID's algorithm to None.

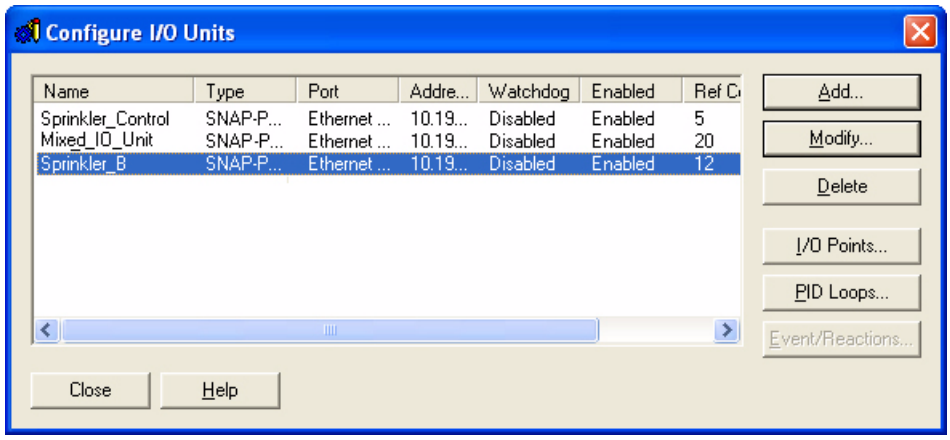
Each PID loop must be individually configured and tuned.

Configuration steps start in "Adding a PID Loop" on page 143, and tuning steps are described in "Inspecting and Tuning PID Loops" on page 156. For additional information, see "PID—Ethernet Commands" on page 296, and Opto 22 form 1641, *OptoTutorial: SNAP PAC PID*.

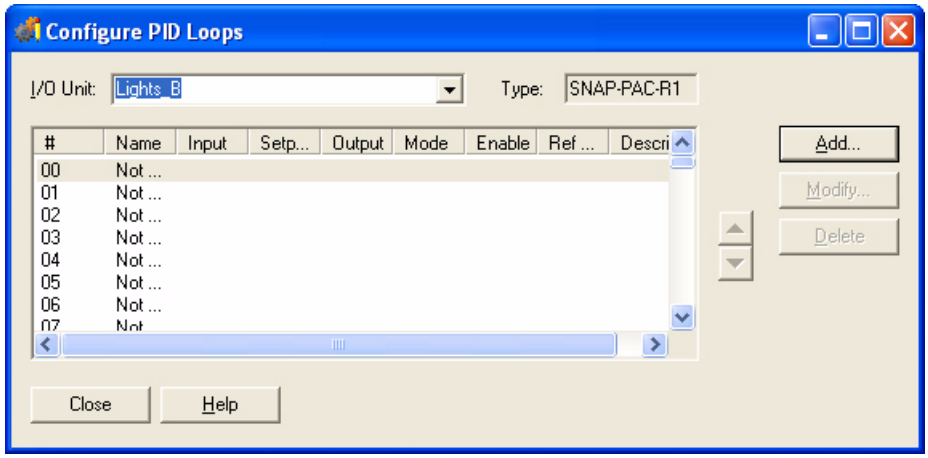
Adding a PID Loop

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.



2. Select the I/O unit the PID will be on, and click PID Loops.



3. Double-click the lowest unused number.

   The Add PID Loop Dialog appears.

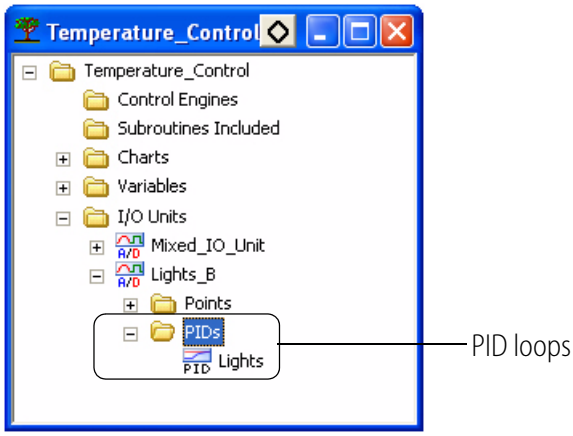
4. Complete the fields as described in "Add PID Loop Dialog Box" below.

5. Click OK.

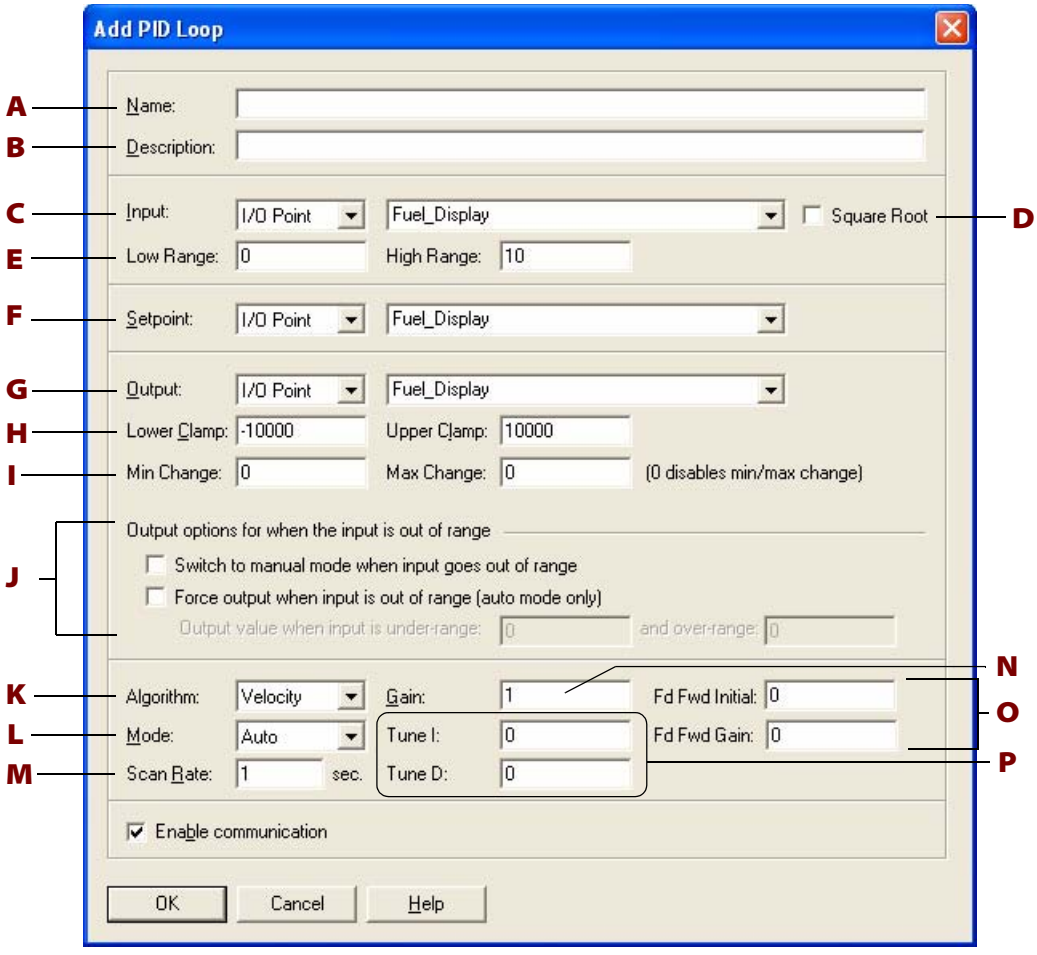
   The new PID appears in the Configure PID Loops dialog box.

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. If no boxes are checked, the PID will freeze output at the current value. 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.

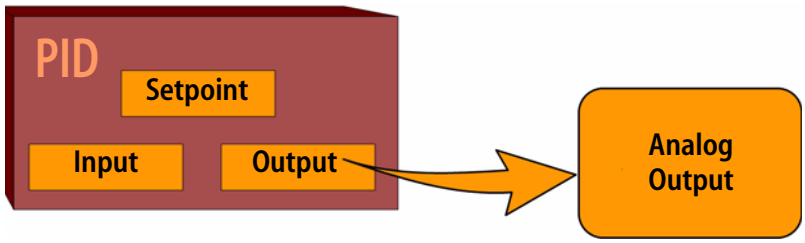
NOTE: If both boxes are checked (forced output and manual mode), the output will be forced and the PID put into manual mode; but if the PID is already in manual mode, the output will not be forced. (You can use the command Get PID Status Flags to determine current settings.)

K—*Algorithm* Choose algorithm: Velocity, ISA, Parallel, Interacting. For details on algorithms, see "Algorithm Choices (PID)" on page 297.

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.

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.)

Description (Optional) Enter a description of the PID.

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.

Options To set advanced options.

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.

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.

Input drop-down list (If I/O Point is selected) 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.

Square Root (If I/O Point is selected) Enable or disable square root extraction and averaging of the input value prior to the PID calculation. These options are disabled by default.

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.

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 and low-scale value. The value representing this full range will appear by default.

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.

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.

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.

Initial Value (If Host is selected) Specify the initial value for the setpoint. This value must be between the lower-clamp and upper-clamp values. The default is zero.

Lower Clamp, Setpoint (If Host is selected) Specify the lowest possible setpoint value. Default: -32,768.

Upper Clamp, Setpoint (If Host is selected) Specify the highest possible setpoint value. Default: 32,767.

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.

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.)

Derivitive 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.

Enable Communication Select whether you want communication to this PID loop enabled or disabled.

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.

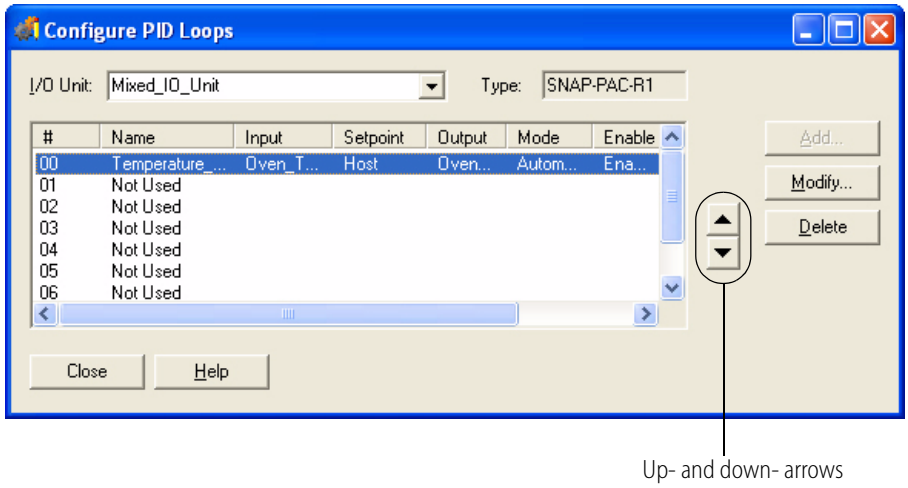
E—***Manual Mode/Reset*** If you want to force the PID loop to reset when entering manual mode, click Yes. This option stops all calculations, sets all process errors to zero, and resets the scan rate timer, leaving the output unchanged. If you want the PID calculation to continue, click No. (When you enter automatic mode, however, you cannot force a reset.)

Changing a PID Loop

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.



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" on page 143.

Deleting a PID Loop

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.

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.
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.

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
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.

Inspecting I/O in Debug Mode

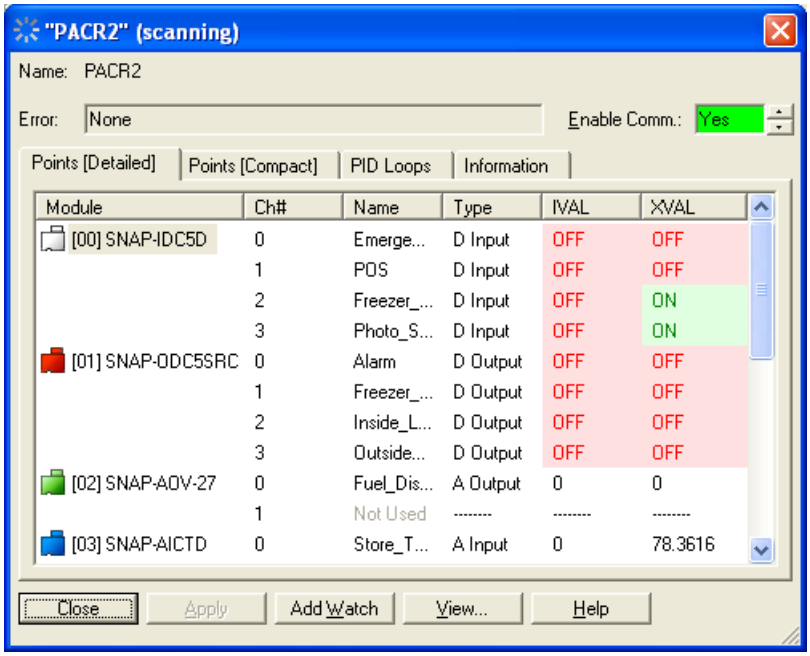
You may want to inspect or change I/O while you are running your strategy in Debug mode. This section shows how to view information about I/O and make changes while the strategy is running.

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 168.

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.



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 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.

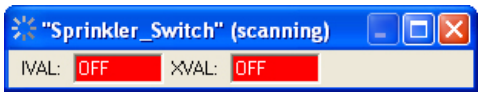
   Yes on a green background means enabled; No on a red background means disabled. If you change it, the background turns magenta until you click Apply.
5. To view an individual I/O point, highlight its name in the list.
   - To add an I/O element to a watch window, click Add Watch. See page 168.
   - To 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

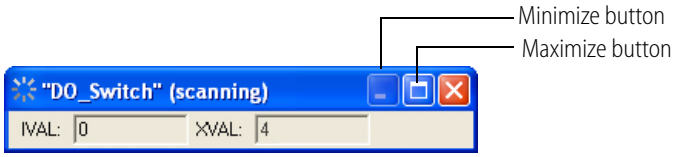
You can inspect a digital point's data, change its status, or set its internal values or external values in Debug mode. To monitor the point in a watch window, see page 168. To change the point, follow these steps.

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.

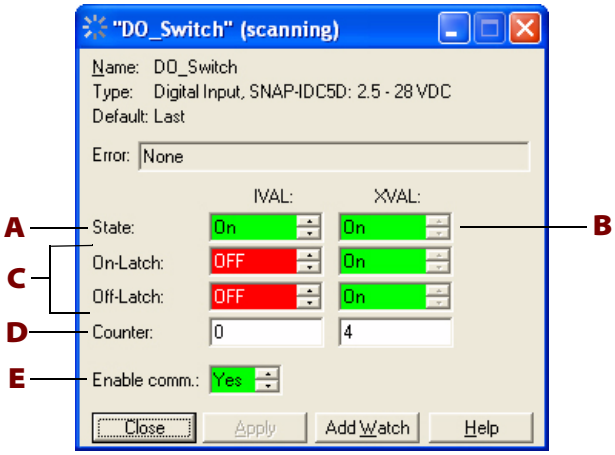


If the digital point is configured with a counter, the counter values appear instead of the point's status.



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.

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 Add Watch and see page 168.

Inspecting Analog I/O Points

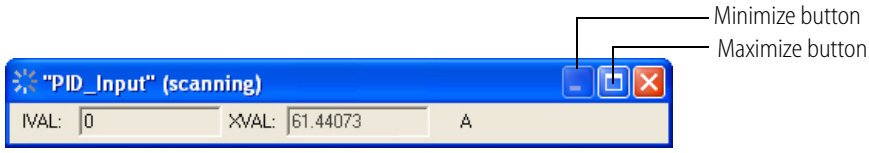
You can review an analog point's data, modify its status, or set its internal values or external values in Debug mode. To monitor the point in a watch window, see page 168. To change the point, follow these steps.

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.

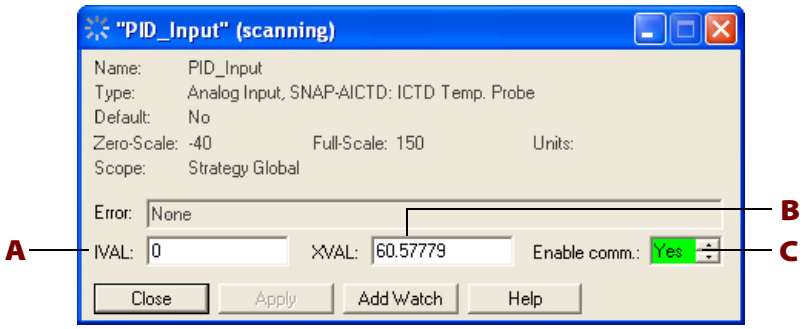
   – The *XVAL*, or external value, is the "real" or hardware value as seen by the I/O unit. This value is external to the control engine.

   – 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.



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 Add Watch and see page 168.

Name Name of the event/reaction.

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.

Unit The I/O unit on which the event/reaction is configured

Error Any communication error appears here with a red background.

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.

Event Parameters Parameters of the event, together with their internal and external values. You can change the internal or external values. If you do, click Apply.

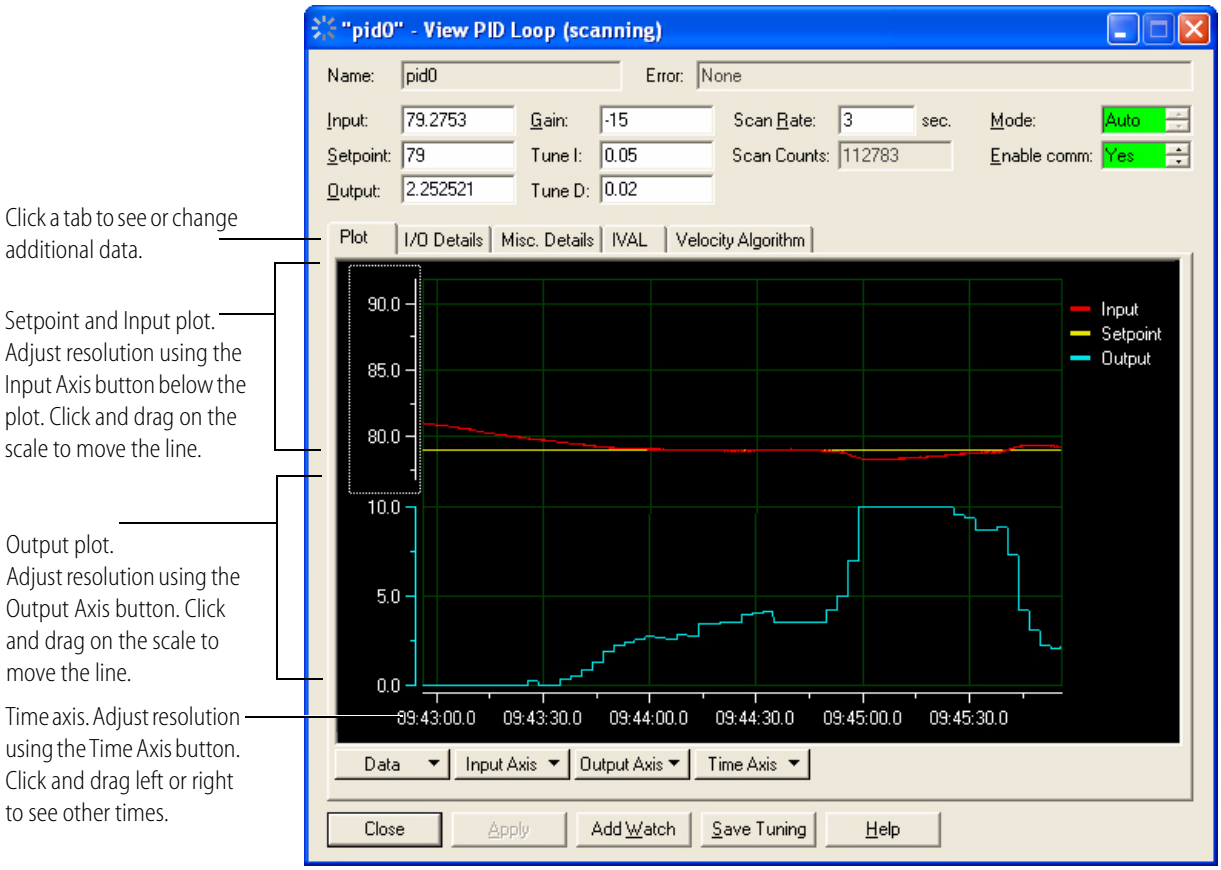
Reaction Parameters Parameters of the reaction, together with their internal and external values. Depending on the reaction type, other parameters that can appear here include an I/O point, an event to be triggered, or MOMO data. You can change any of the internal or external values. If you do, click Apply.

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. See also, Opto 22 form 1641, *OptoTutorial: SNAP PAC PID*, which we highly recommend. Form 1641 is available for download from our website at www.opto22.com.

Inspecting a PID

1. With the strategy running in Debug mode, double-click the PID on the Strategy Tree.

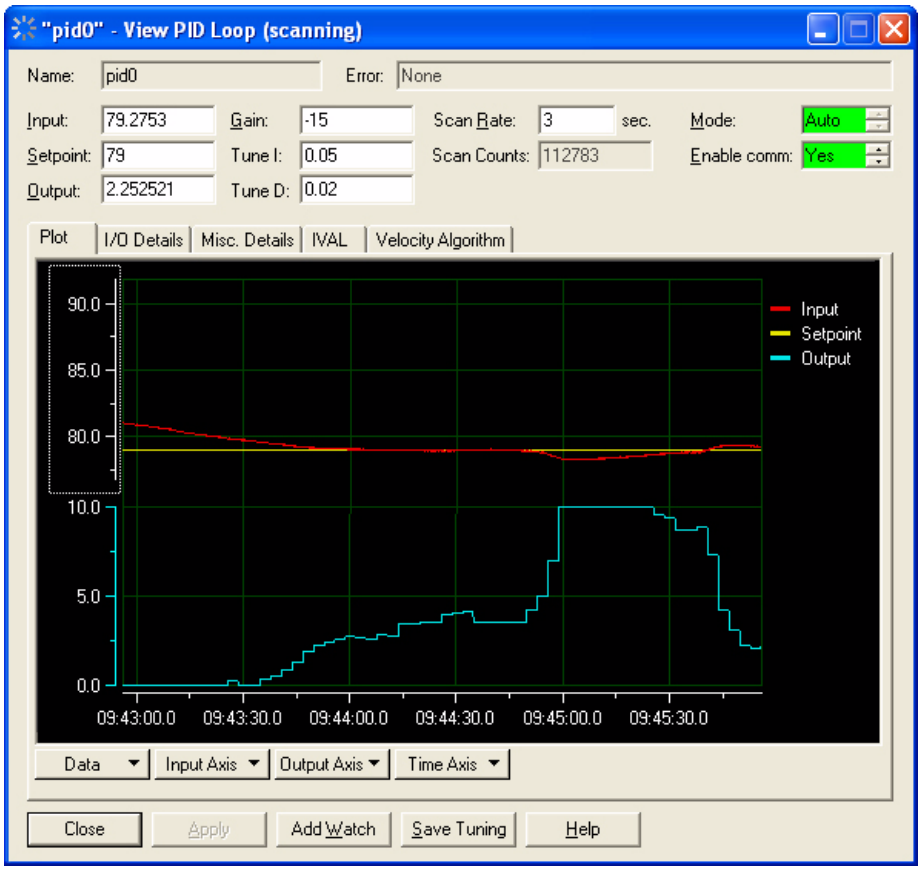


2. View or change PID parameters as necessary. Click the other tabs to see additional data. To tune the PID, see page 162.
3. To add the PID to a watch window, click Add Watch and see page 168.
4. To save, copy, or print the current plot, click the Data button and choose from the popup menu.
5. To save 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 Opto 22 form 1641, *OptoTutorial: SNAP PAC PID*, 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.



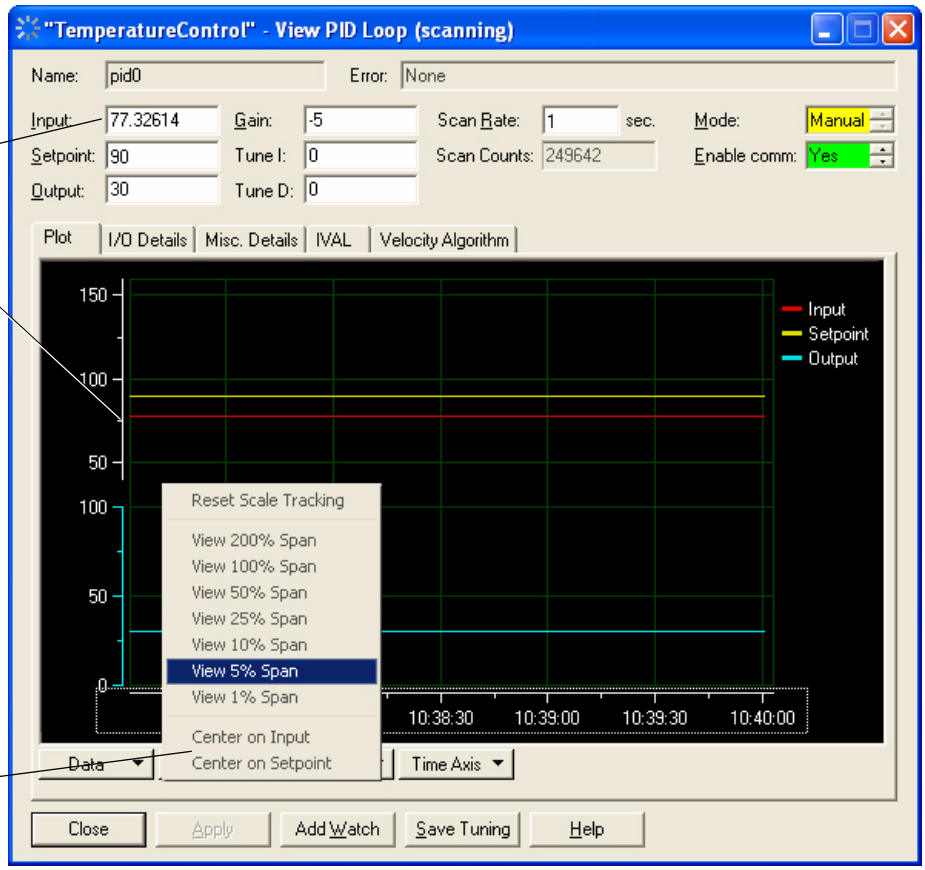
3. Set the Mode to Manual (if not set already) and click Apply.
4. In the Output field, type one of your two output settings and click Apply.

   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 popup 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.



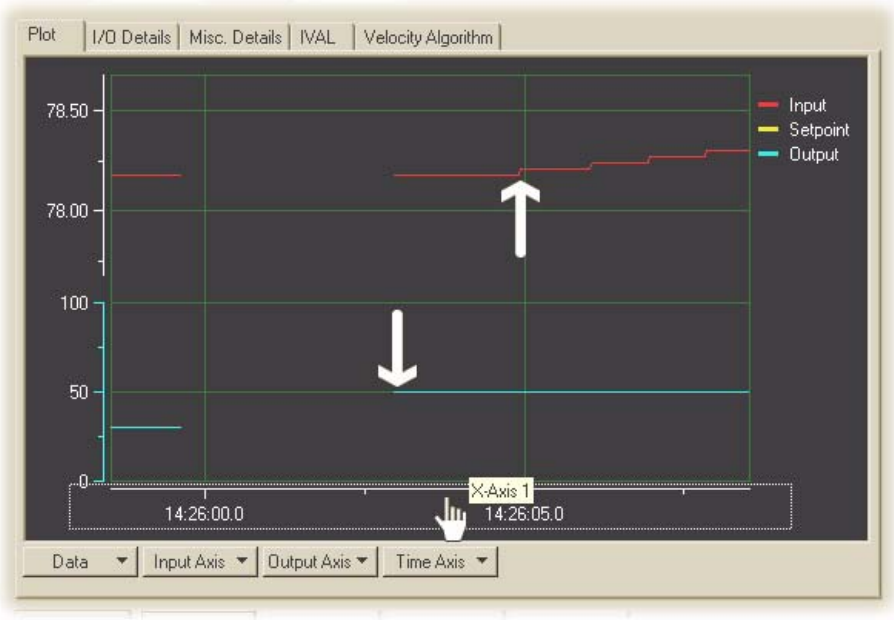
Adjust resolution of the Input Axis by clicking the Input Axis button.

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.

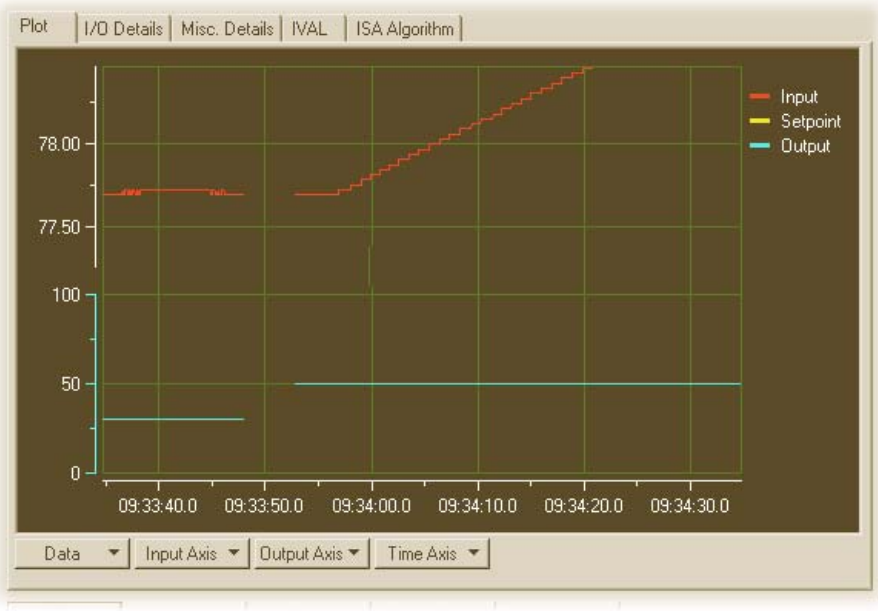
The Input axis (indicated by the upper white arrow added to this picture) begins to respond to the change in output (lower white arrow).



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.

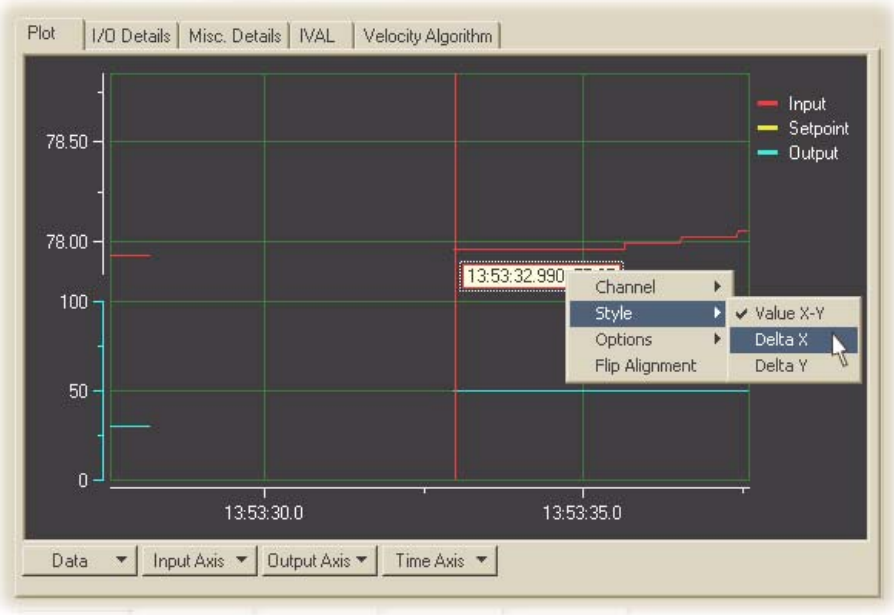
Both the time and input axes should display the point at which the Output changed, the lag, and the point the input changed. If not, adjust the Input axis and Time axis, until this information is displayed.



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

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 Opto 22 form 1641, *OptoTutorial: SNAP PAC PID*, for more detailed information. Form 1641 is available for download from our website at www.opto22.com.

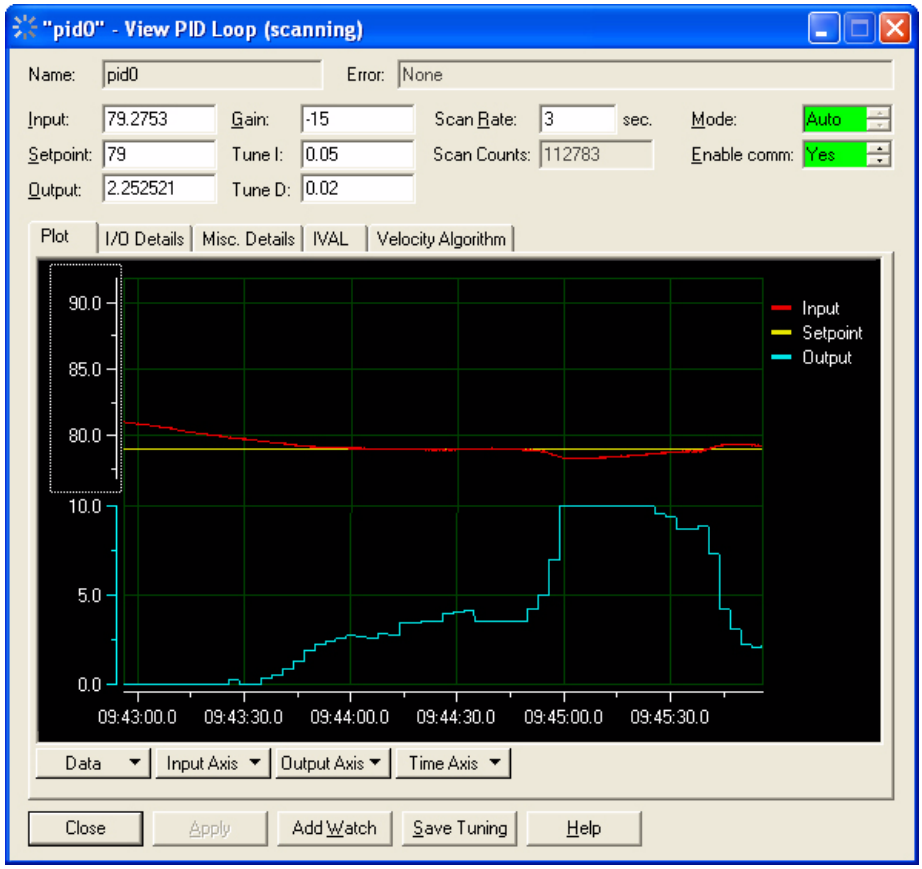
CAUTION: Before following these procedures, make sure you know the limits of the equipment being controlled and monitored by your PID loop. Also, make sure that these points are configured properly. Any values suggested in these steps are for example only and must be modified according to the capabilities 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.



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.)

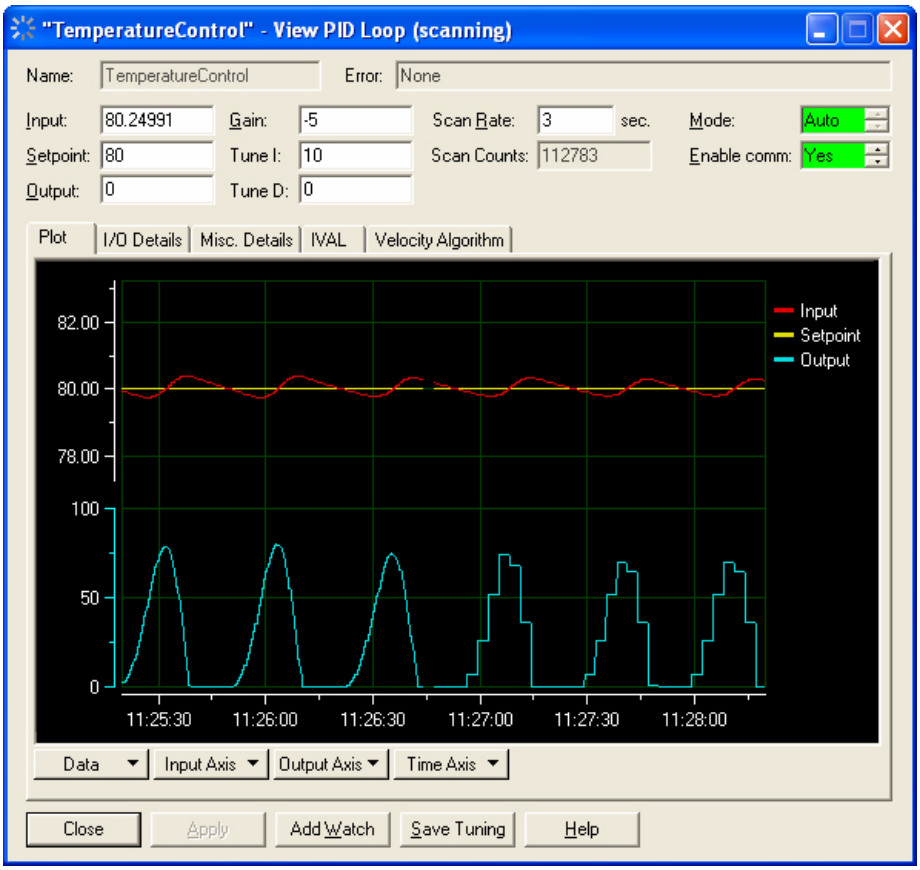
   Depending on the type of system, your PID may maintain a setpoint or respond to changes in setpoint. Experiment with setpoint changes again after tuning the P, I, and D constants.

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 popup 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 157). 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:

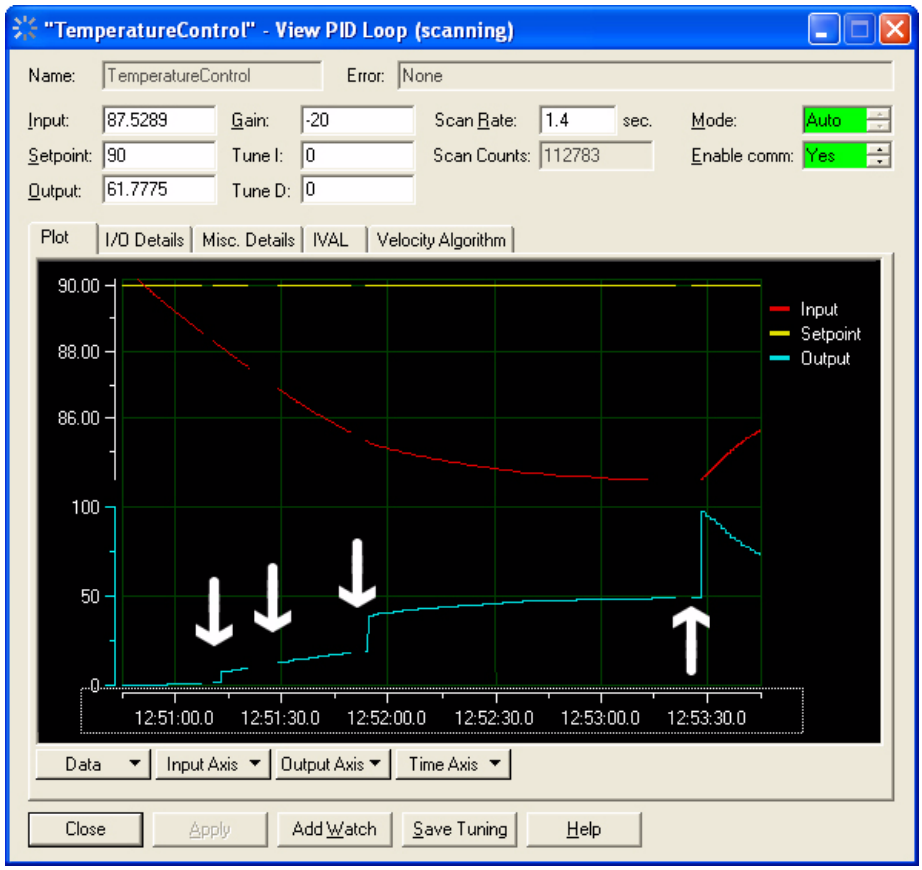
The lag for this system was determined to be about 2 seconds. The left half of the plot reflects a 0.5-second scan rate, while the right half shows a 3-second scan rate. Notice both scan rates have the same effect on the input; however, the 3-second scan rate is using less of the processor's resources.



8. Experiment with gain settings by typing a new value in the Gain field and clicking Apply.

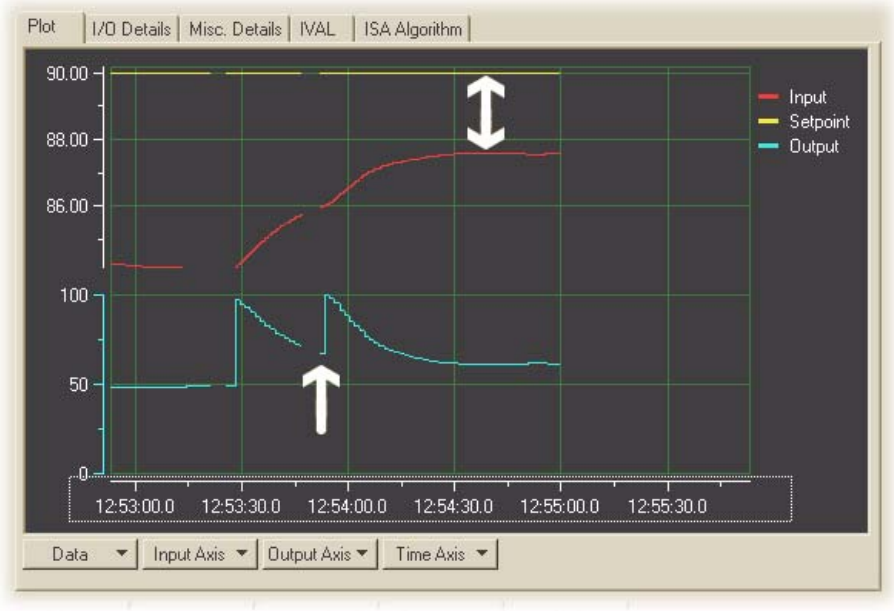
The easiest way to tune most PIDs is to experiment with the gain constant first. Try various gains to see how well the system stays at setpoint and responds to setpoint changes.

In the example below, the white arrows (added for the example), show where gain constants of -2, -5, -10, and -20 were applied:



In this example, a gain setting of -30 revealed an offset error:

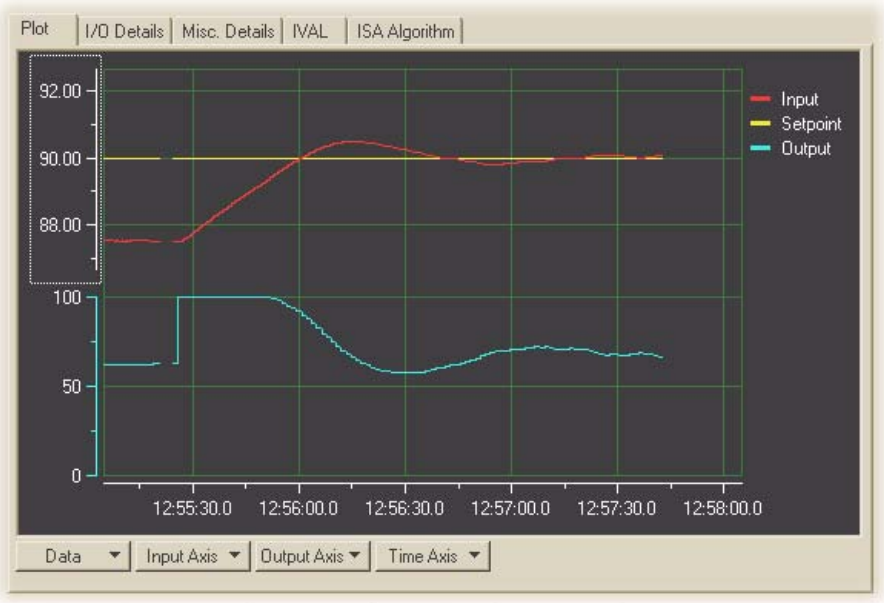
With only a gain constant applied, the input often stabilizes at an incorrect value. In this heating example, a gain setting of -30 drove the input close to the setpoint, but subsequent increases failed to eliminate the offset. It is time to try integral constants to eliminate the offset error.



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.

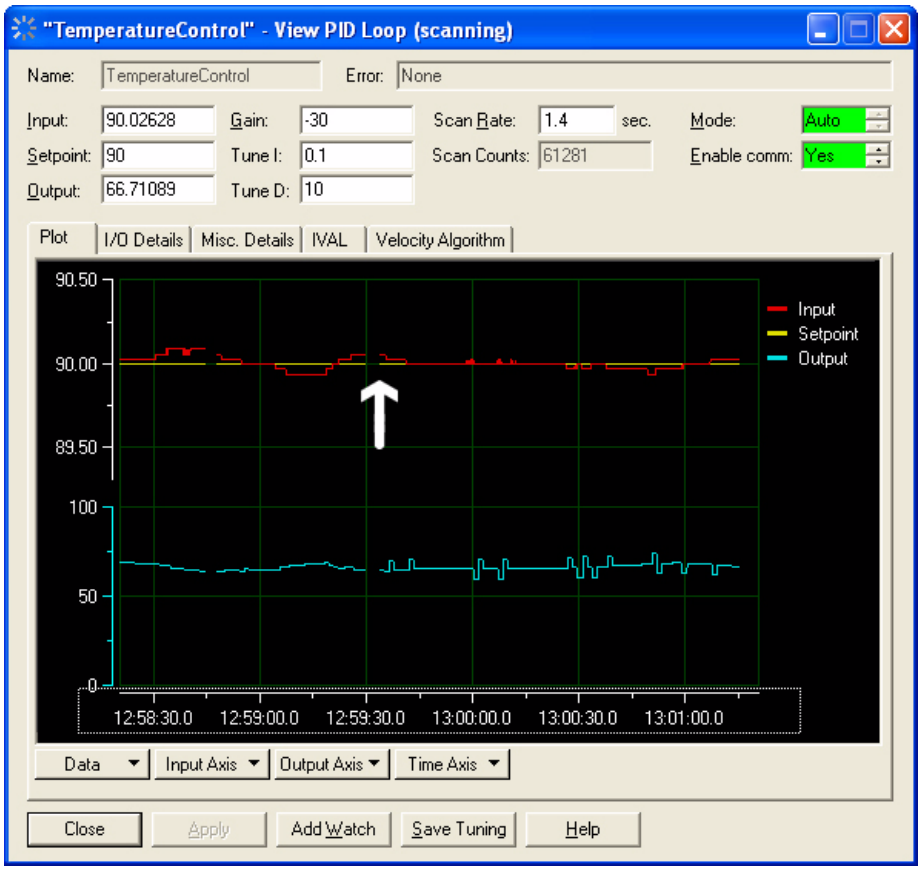


10. If derivative correction is needed, experiment with the derivative constant: in the Tune D field, type 1 and click Apply. (Your PID loop may require a larger number.)

In this example, a derivative of 10 makes a noticeable difference in keeping the input near the setpoint.

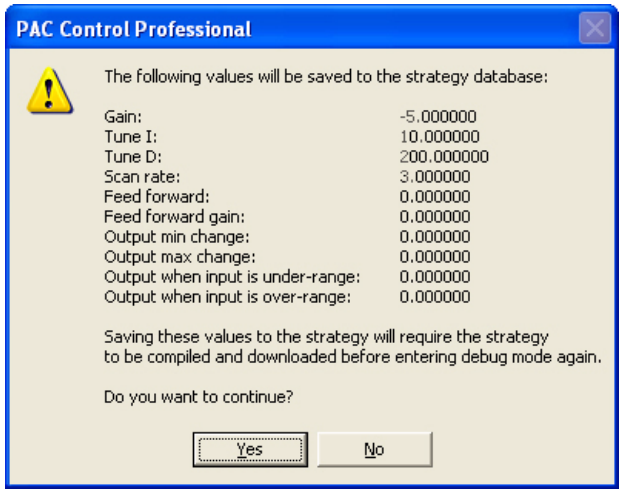
Many PID systems are effectively controlled with gain and integral constants only and no derivative constant. In this example, the gain and integral settings are maintaining the temperature at 0.06 from setpoint. To demonstrate the effect of the derivative constant, the resolution of the input axis was 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.



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.



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 152).

Input, Output, and Setpoint (current internal and external values) ***C***hange 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.

Gain, Integral, and Derivative (current internal and external values) ***Y***ou 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.

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).

PID execution mode ***A*** green background means Automatic, a yellow background means Manual. Click an arrow to change the mode; then click Apply.

Setpoint and Input plot Adjust resolution using the Input Axis button at right. 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.

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

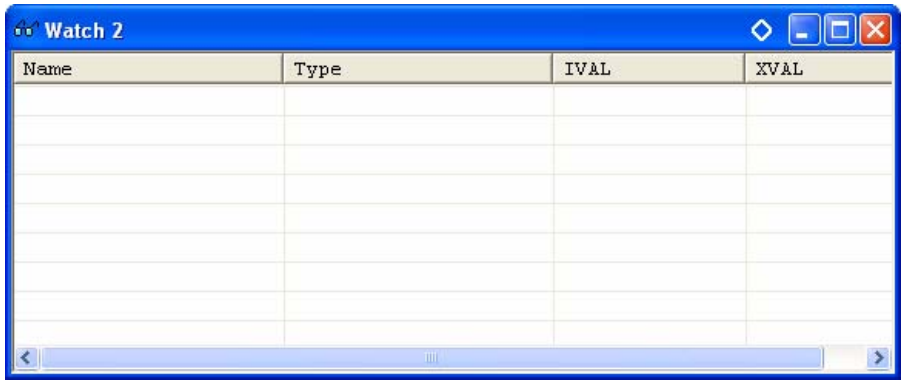
While the strategy is running, you can monitor several strategy elements at once in a watch window: I/O units, digital and analog points, PID loops, variables, even charts. You cannot monitor subroutine parameters or variables that are local to a subroutine in a watch window.

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, click the New Watch Window button in the toolbar or 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.

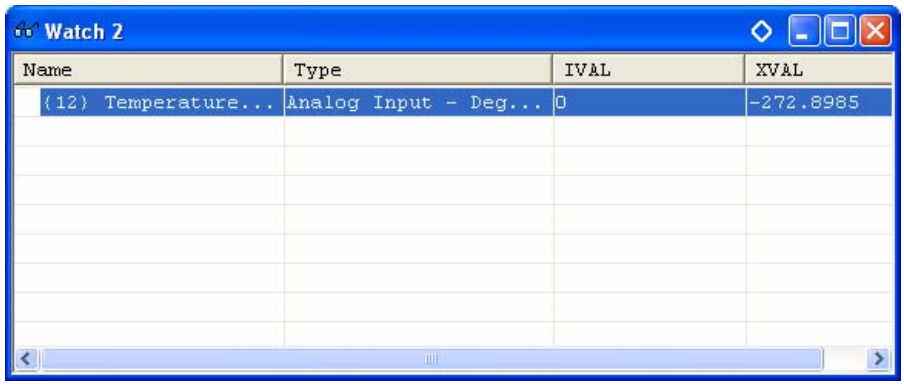
   The empty watch window appears.



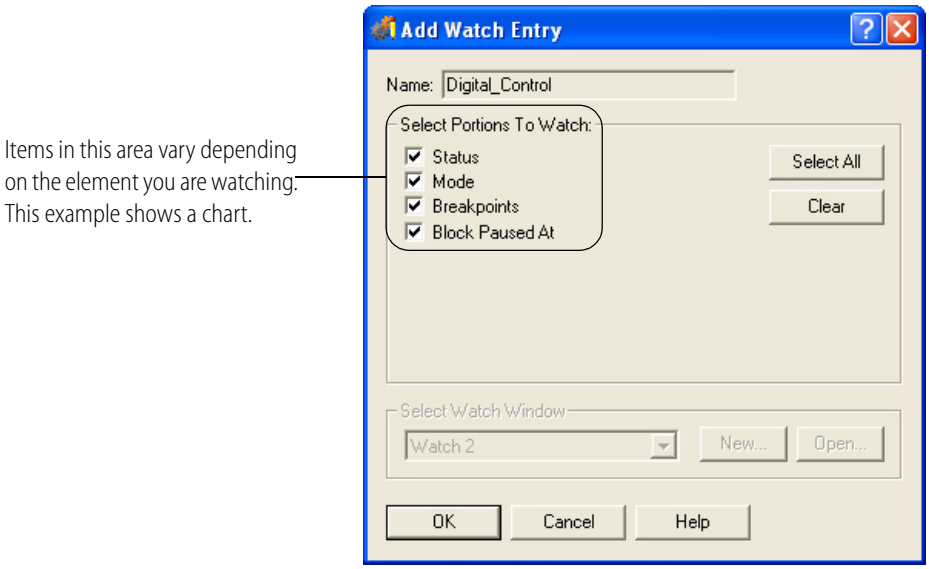
3. Add elements you want to watch in this window by clicking them on the Strategy Tree and dragging them into place in the watch window, or by right-clicking them and choosing Watch from the pop-up menu.

   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.



   For some elements, the Add Watch Entry dialog box appears, so you can specify what to watch.



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. Click the Open Watch Window button on the toolbar, or 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.

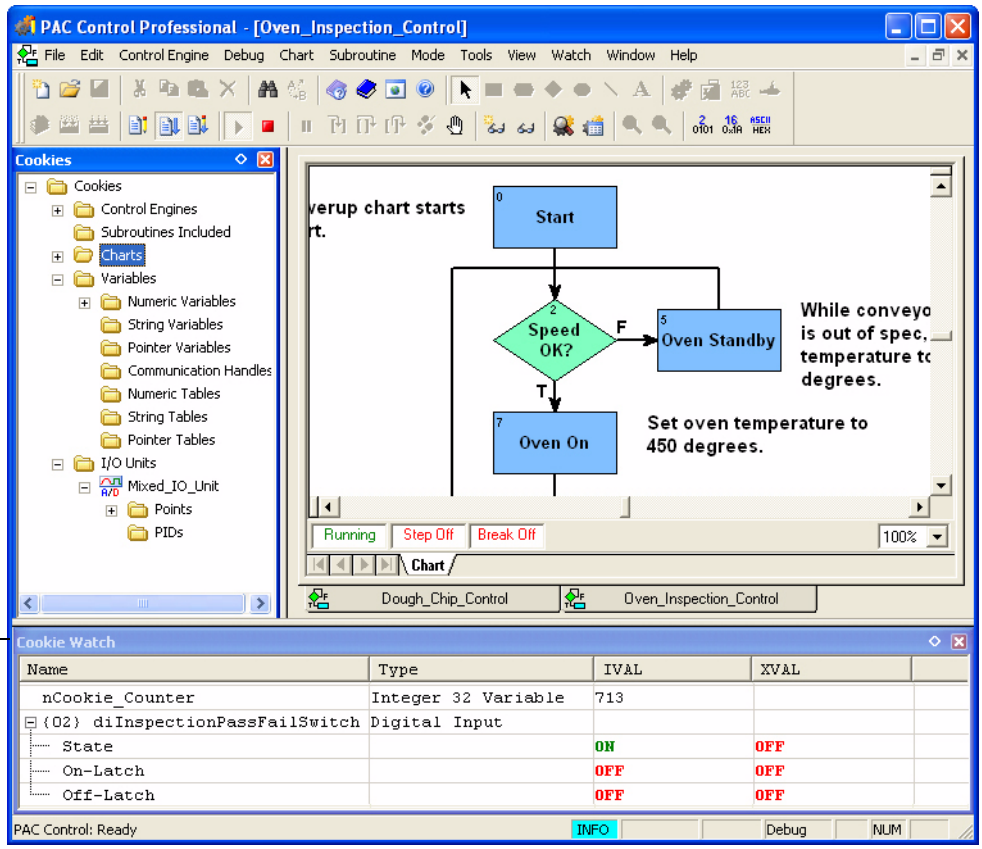
Working in Watch Windows

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.

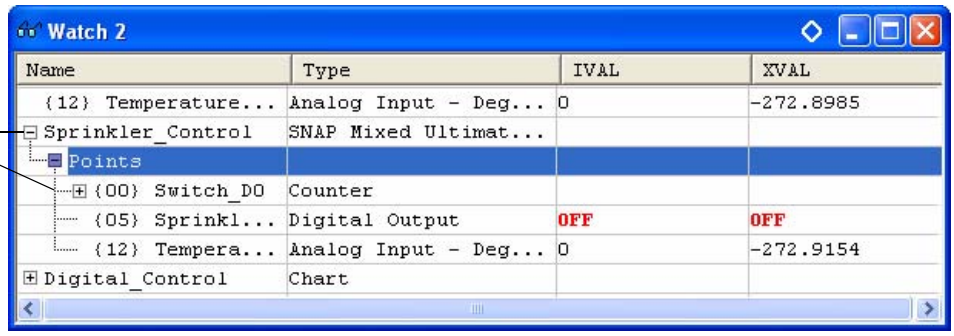
"Docked" watch window



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.

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 110

Working with Strategies

OPTO 22

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.....173
Opening a Strategy.....174
Saving and Closing.....175
Saving a Strategy to Flash.....177
Compiling and Downloading.....179
Running a Strategy Manually.....188
Debugging.....189
Viewing and Printing.....196
Searching and Replacing.....207
Legacy Options.....209

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. Having 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 Folder button 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 55. For programming information, see Chapter 4, "Designing Your Strategy." For steps to create charts, see Chapter 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 button 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 Basic and PAC Control Professional" 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 Basic and PAC Control Professional

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

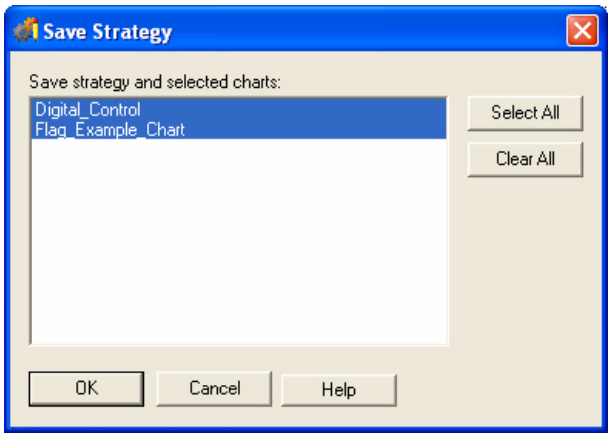
To save all your work quickly, choose File→Save All. The strategy and all modified charts and subroutines are saved.

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 button 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. In this example, two charts have been modified:



2. To save some charts and not others, press CTRL and click any charts you don't want to save.

   You can also click Clear All to select none of the charts, or click Select All to select all of the charts.

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.

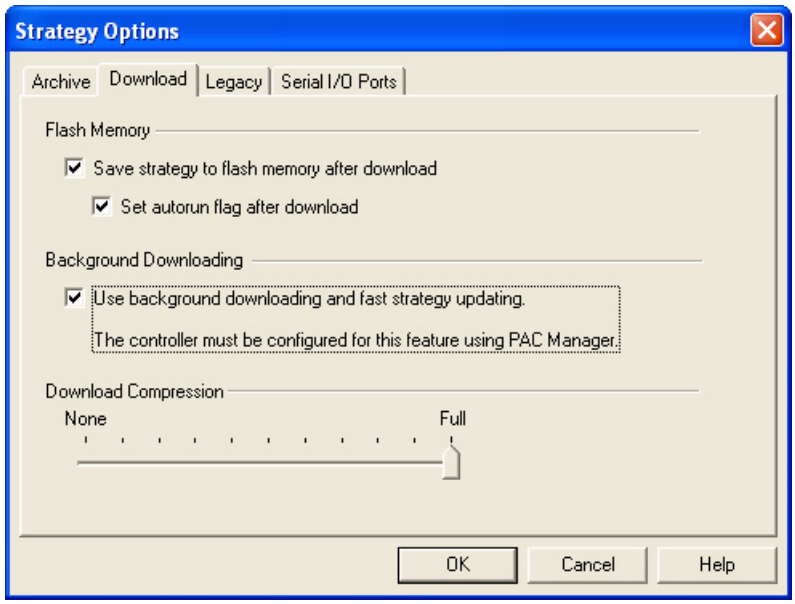
Saving to Flash Once

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.



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

Strategy archives help you track changes during development and provide a backup in case of a failure on the control engine or on the computer where the original files are kept. Archive files are date and time stamped, and zipped for compact storage. We recommend you archive both to the computer (during strategy development) and to the control engine (when the strategy is completed).

Archiving to the Computer

Archiving strategies to the computer is an excellent way to track changes over time and to produce a zipped file 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.

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.

Archiving to the Control Engine

When you archive a strategy to the control engine, you are placing the zipped file in battery-backed RAM. If power to the control engine is lost, the archive is still there. 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.

Make sure there is sufficient memory in the control engine for the archive file. Battery-backed RAM holds 256KB total; in addition to the archived strategy, it stores persistent variables and variables that are initialized on strategy download.

For steps for archiving to the control engine, see "Archiving to the Control Engine" on page 116.

Compiling and Downloading

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

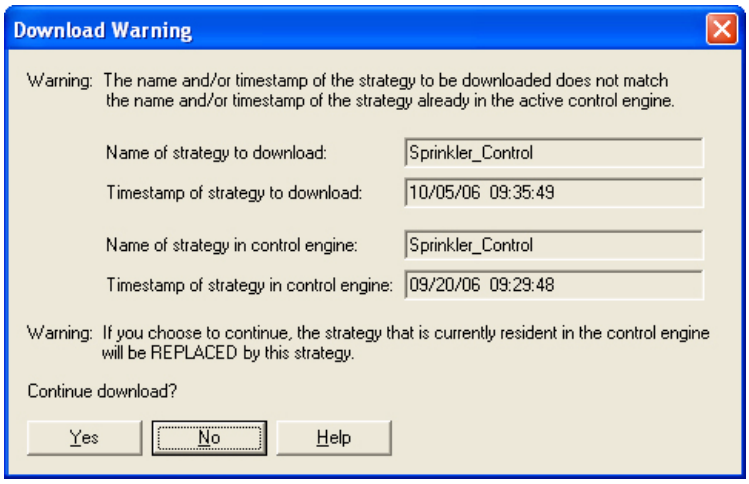
NOTE: If you are using Ethernet link redundancy in PAC Control Professional, make sure you are downloading through the IP address you want to use, either the primary or secondary address. See "Using Ethernet Link Redundancy in PAC Control" on page 104 for more information.

1. With the strategy open in PAC Control, click the Debug mode button 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.

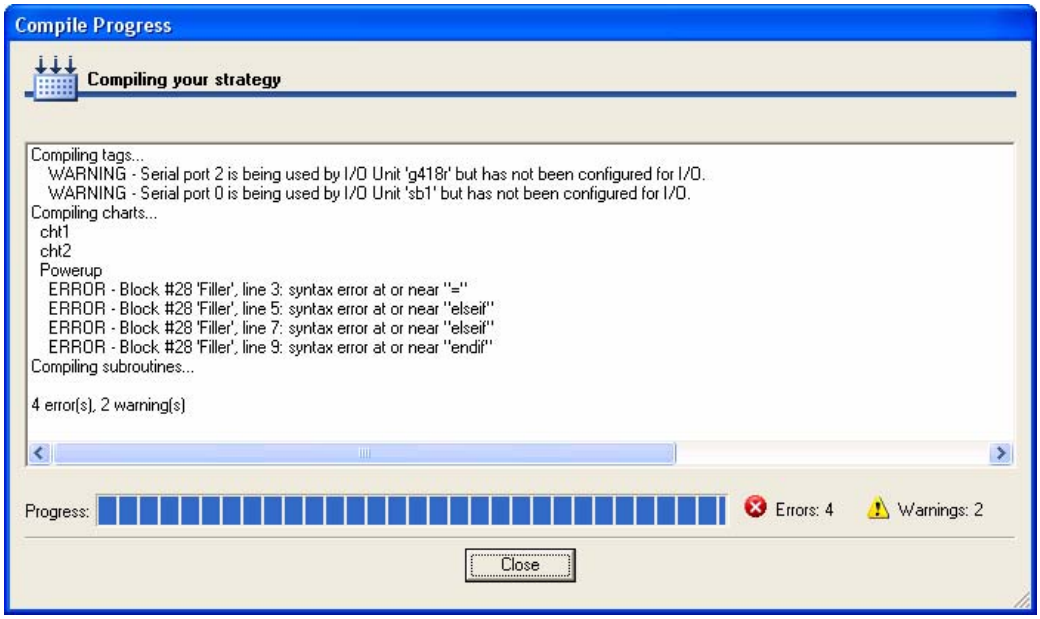
   A download warning message may appear.



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.

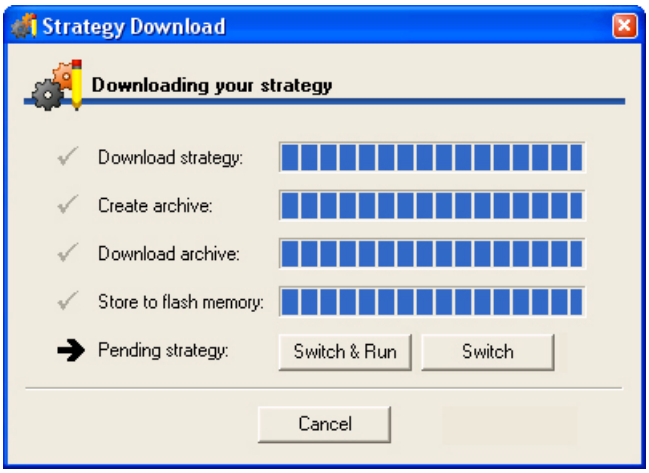
   As the strategy is compiled, the Compile Progress dialog box appears. This dialog may appear only briefly. However, if there are errors or warnings, the dialog box will stay open. You can copy the errors and warnings by clicking the right mouse button and selecting Copy All as Text.



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 183), 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 Appendix A, "Troubleshooting.")

After switching strategies, you can switch back to the original strategy by choosing 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 183), 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 183.

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 button on the toolbar, or choose Compile→Compile 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, in Configure mode, click the Compile Changes button on the toolbar, or 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 button 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.

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 it in PAC Manager. Also, one chart is used for the alternate download port, so there is one less chart that can run.

1. Open PAC Manager and do the following steps:
   a. Click the Inspect button to open the Inspect window.

b. In the IP Address field on the Inspect window, type the IP address for the SNAP PAC controller.

c. Click Status Write, then scroll down to Strategy Download Method.

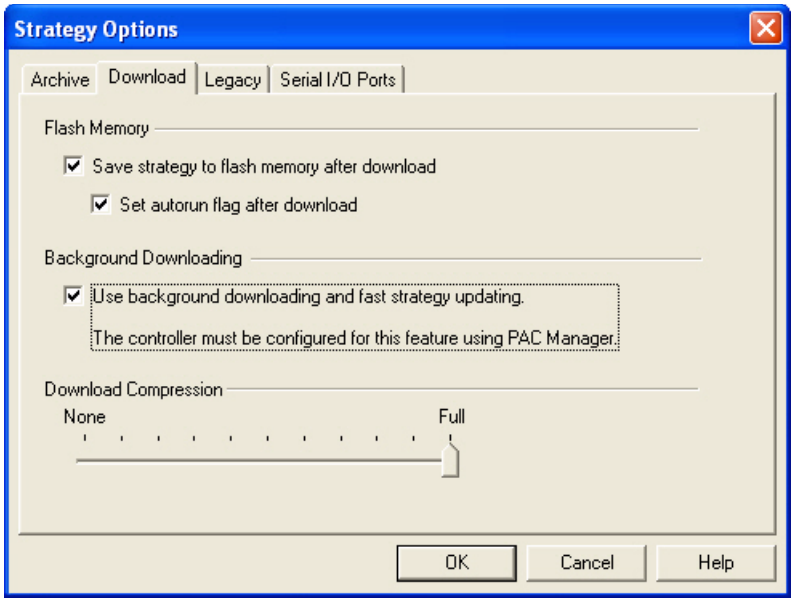
d. Click on the Value field for Strategy Download Method, select Background from the drop-down menu, then click Apply.

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 the *PAC Manager User's Guide*, form 1704.

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. On the Download tab under Flash Memory, 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."
7. Click OK.

PAC Control is now set to download the strategy in the background when you click the Debug mode button . Next, see "Compiling and Downloading" on page 179. See also, "Switching the Strategy" on page 181.

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 controllerusing 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 177. 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

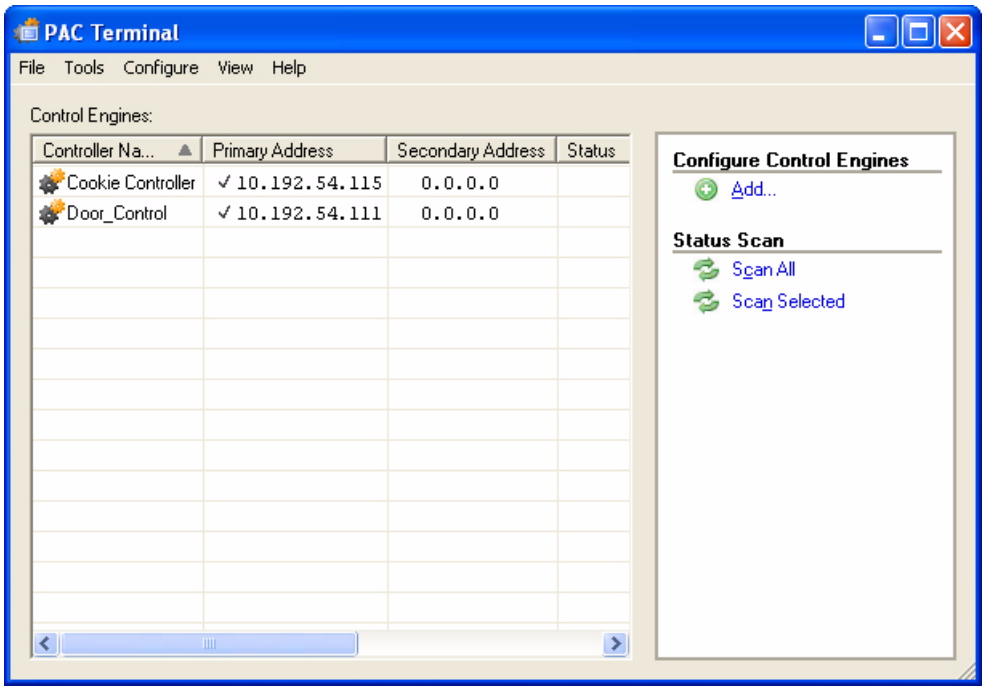
With the strategy open in PAC Control in Configure mode, right-click the name of the control engine in the Strategy Tree and choose Compile Control Engine Download File from the pop-up menu. (You can also choose Compile→Compile Control Engine Download File.)

The file is created in the same folder as the strategy, with a .cdf extension and a filename consisting of the strategy's name and the control engine's name (for example, MyStrategy.MyEngine.cdf).

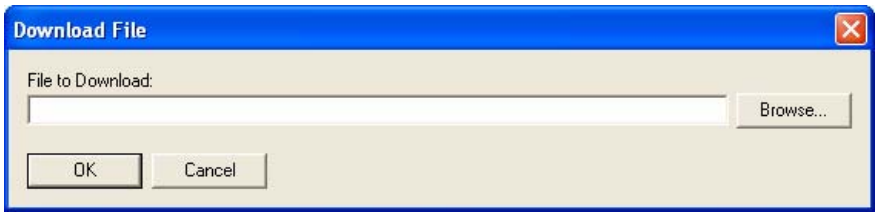
Once the control engine download file is created, it can be downloaded using either the graphical interface in PAC Terminal or the command line interface with a batch file you create.

Downloading the .cdf File using PAC Terminal

1. Click the Windows Start menu and choose Programs→Opto22→PAC Project→Tools→PAC Terminal.



2. Right-click the name of the control engine you want to download the file to.
3. In the pop-up menu, choose Download. In the submenu, choose Control Engine Download File.



4. Enter the path and filename of the .cdf file, or click the Browse button and navigate to it. When the filename appears in the File to Download field, click OK.

   The file is downloaded to the control engine, and a dialog box shows its progress.

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`

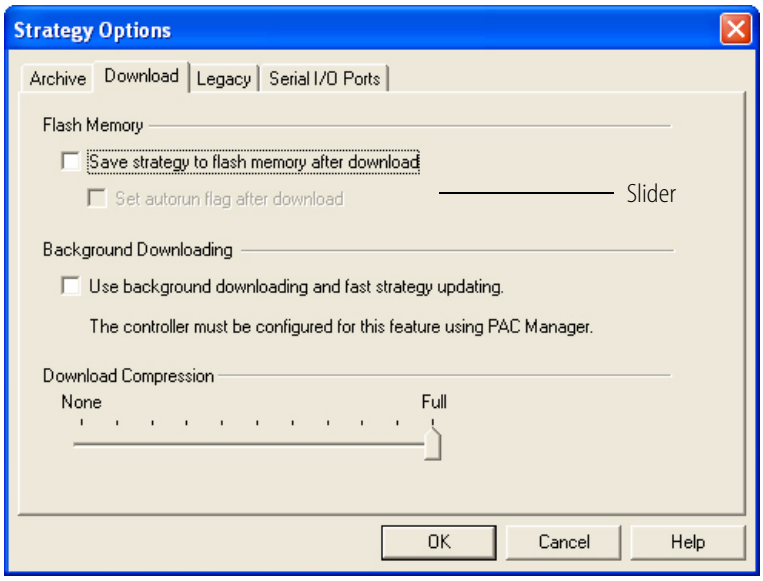
This example shows lines included in a batch file that will define the control engine, download the .cdf file to it, and then run the strategy:

```
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.



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 button (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 110.

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 177.
- 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 110 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 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 button (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 110.

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.

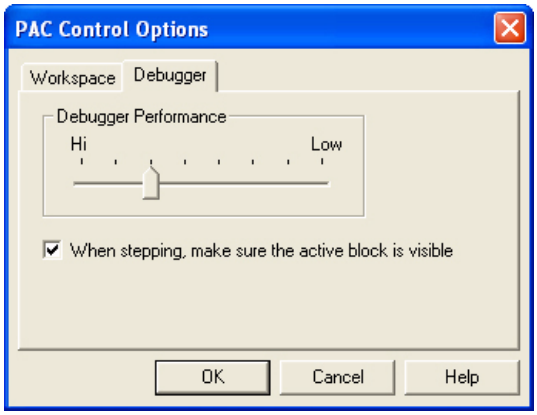
Changing Debugger Speed

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 MessageViewer (see page 363).

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.



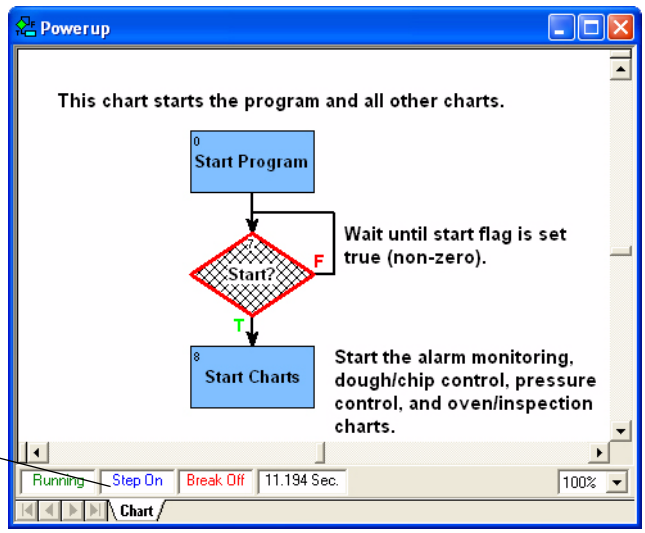
3. Click and drag the slider to the speed you want.

   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.

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 button , or press F7, or select Debug→Pause Chart or Debug→Pause Subroutine. Here's an example of a paused chart:



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.*

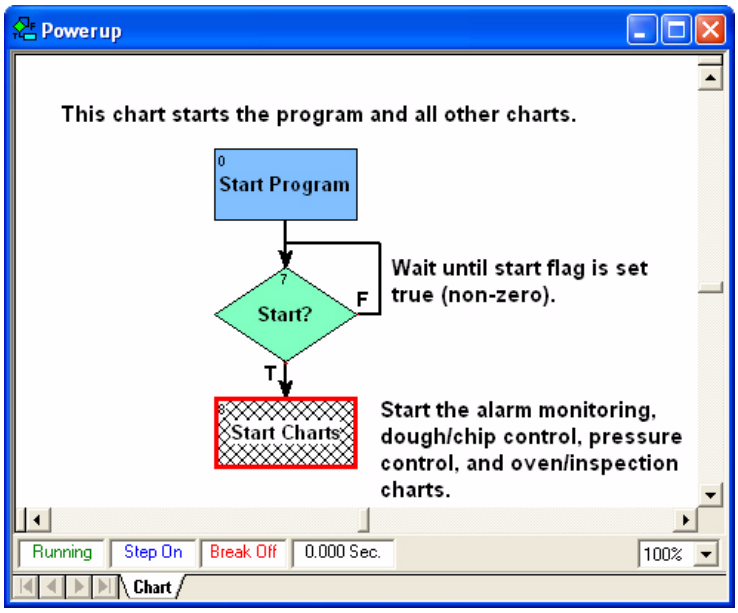
Single Stepping

When you are debugging a strategy, start by using the Step Over button to go through a chart one block at a time. The Step Over button 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 189) so you can step into blocks and execute one line at a time.

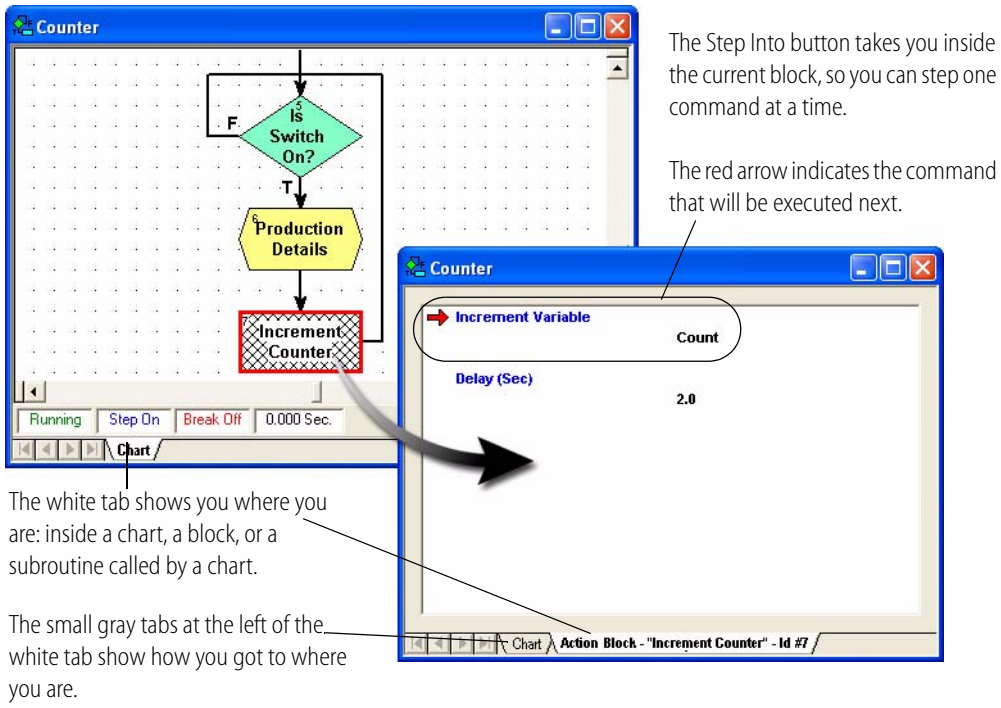
1. Pause the chart or subroutine to be stepped through by pressing the Pause Chart or Pause Subroutine button .

2. To step to the next command block, click the Step Over button (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 191. 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 189 for help.
4. If the chart or subroutine is not already paused, press the Pause Chart or Pause Subroutine button .
5. To step inside the block you are on (the one with the hatch marks), click the Step Into button (or press F11, or select Debug→Step Into).



6. To move from line to line, click either the Step Into or the Step Over button.

   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, unpauses the chart. In a subroutine, clicking Step Out takes you out of the subroutine.

Auto Stepping

Autostepping 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 button , 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 autostepping continues until you stop it by pressing the Auto Step button 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.

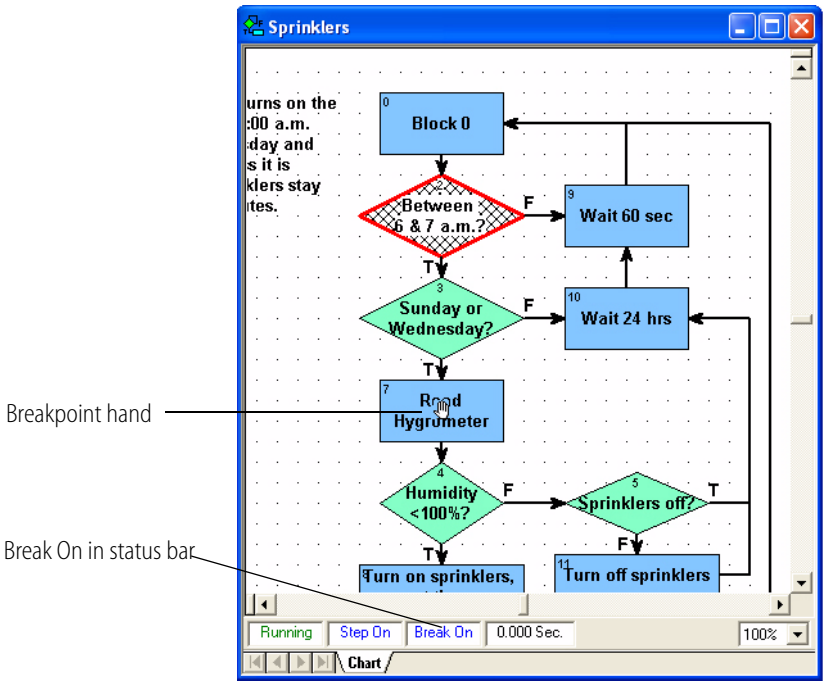
You can set a breakpoint at any block in any chart or subroutine, whether it is running or stopped, paused or auto stepped. The strategy does not need to be running. You can set up to 16 breakpoints in one chart or subroutine. However, you cannot set a breakpoint inside a block.

To set a breakpoint, follow these steps:

1. With the chart or subroutine open and in Debug mode, click the Breakpoint Tool button .

   The pointer turns into a hand.

2. Click the target block to mark it with the breakpoint hand.



   The breakpoint hand appears on the block and Break On appears in the status bar.

3. Click other blocks to set additional breakpoints, or click blocks currently marked with a hand to remove the breakpoint.

4. When you have finished marking or removing breakpoints, click the Breakpoint button again or click the right mouse button within the window.

   When the chart or subroutine runs, it pauses just before executing the breakpoint block. You can inspect variables or I/O points, disable strategy elements, change values, and so on to see the effect the block has.

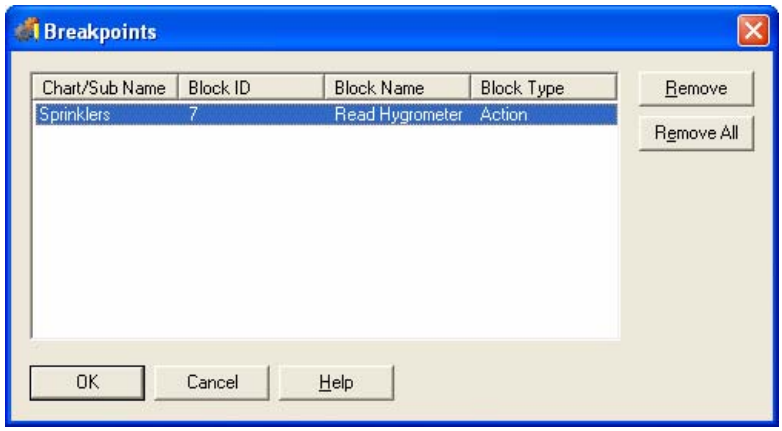
5. To single step past the breakpoint, click the Step Block or Step Line button. Or to run the chart or subroutine at full speed after the breakpoint, click the Pause Chart button.

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 194 to use the breakpoint tool.

1. Press CTRL + B or select Debug→Breakpoints.

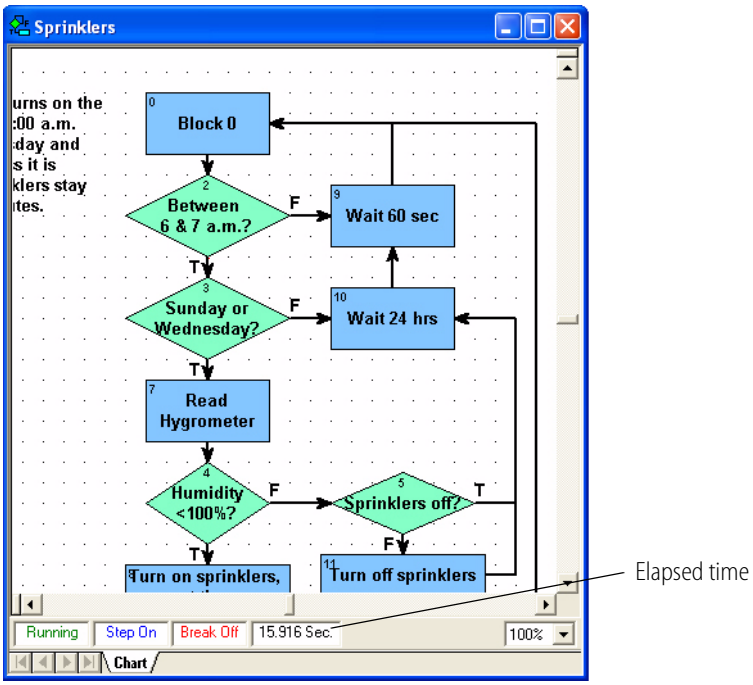
   The Breakpoints dialog box appears, showing all the breakpoints set in the strategy.



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 197
- "Viewing All Charts in a Strategy" on page 197
- "Printing Chart or Subroutine Graphics" on page 198
- "Viewing and Printing Strategy or Subroutine Commands" on page 201
- "Viewing and Printing Strategy or Subroutine Elements" on page 202

- "Viewing and Printing a Cross Reference" on page 203
- "Viewing and Printing a Bill of Materials" on page 204

For information on viewing and changing I/O units, see "Inspecting I/O in Debug Mode" on page 152. For variables, see "Viewing Variables in Debug Mode" on page 244.

Viewing Strategy Filename and Path

To see an open strategy's filename and path, choose File→Strategy Information. A dialog box appears showing the path and filename.

Viewing an Individual Chart or Subroutine

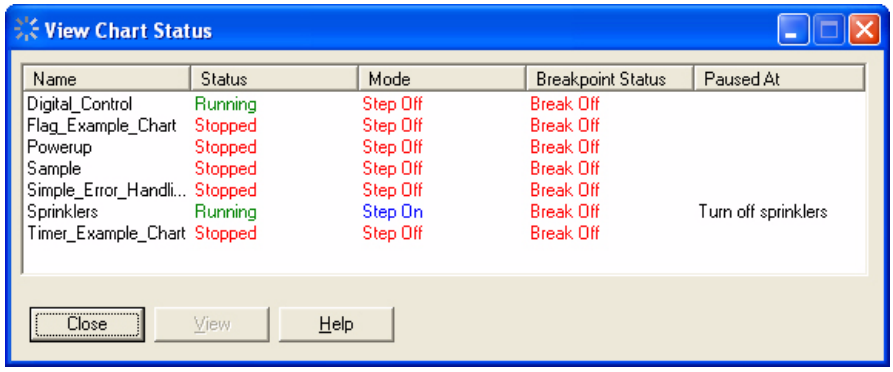
To view an individual chart or subroutine, double-click its name on the Strategy Tree, or choose Chart→Open or Subroutine→Open. You can open as many of these windows as you need. The names of open windows appear on tabs at the bottom of the PAC Control main window. Click a tab to bring its window into 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.

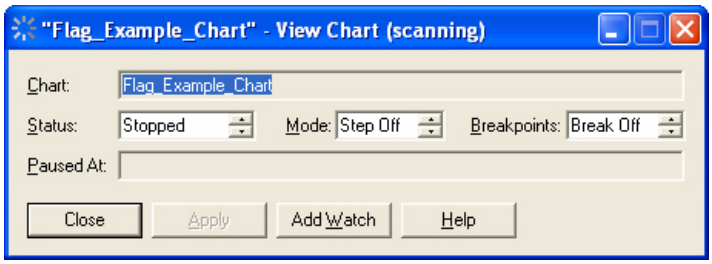
1. Make sure the strategy is open and in Debug mode. On the Strategy Tree, double-click the Charts folder.

   The View Chart Status dialog box appears, showing every chart in the strategy:



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:



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 168 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.

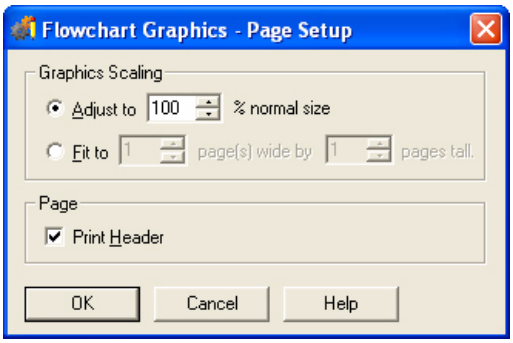
NOTE: If you have trouble printing graphics, set your printer for PostScript emulation.

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.



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.

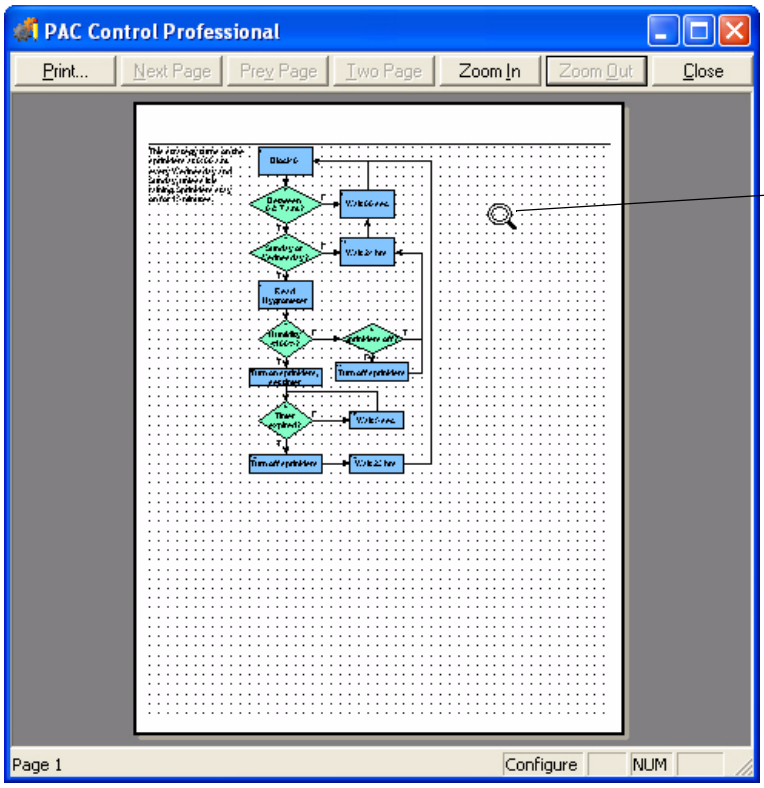
     If you choose one for each dimension, each chart prints to a single page. For each dimension, you can specify any integer between one and 255, but be careful. Selecting values of five and five, for example, would cause each chart to print five pages wide and five pages long, a total of 25 pages.
3. (Recommended) 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.



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 button to open the standard Windows Print dialog box. To change settings before printing, click Close and see "Setting Up the Page" on page 198.

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, enter the file name and location.

   Your chart or subroutine is printed.

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 198.

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.

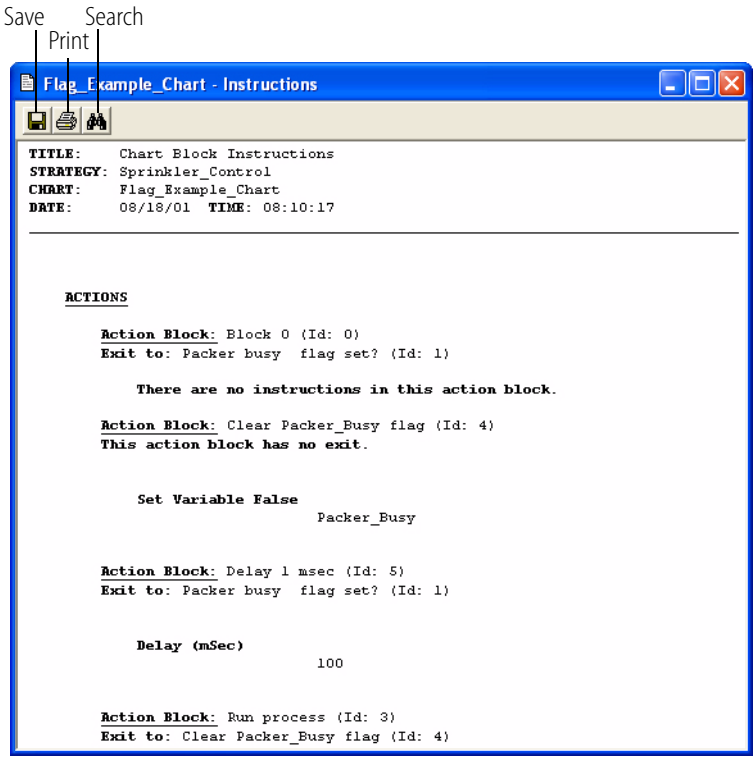
Viewing and Printing Strategy or Subroutine Commands

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.



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.

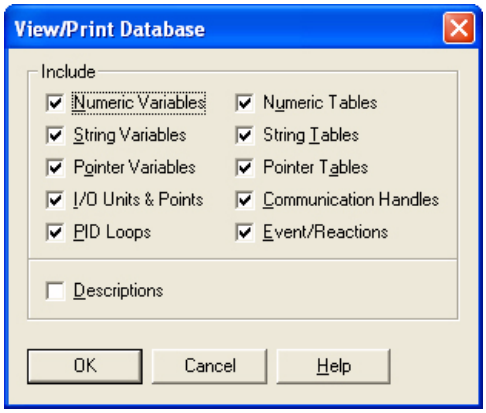
3. To print the data, click the print button on the toolbar. To save it to a text file, click the save button. To search the data, click the search button. When finished, close the window.

Viewing and Printing Strategy or Subroutine Elements

You must be in Configure mode.

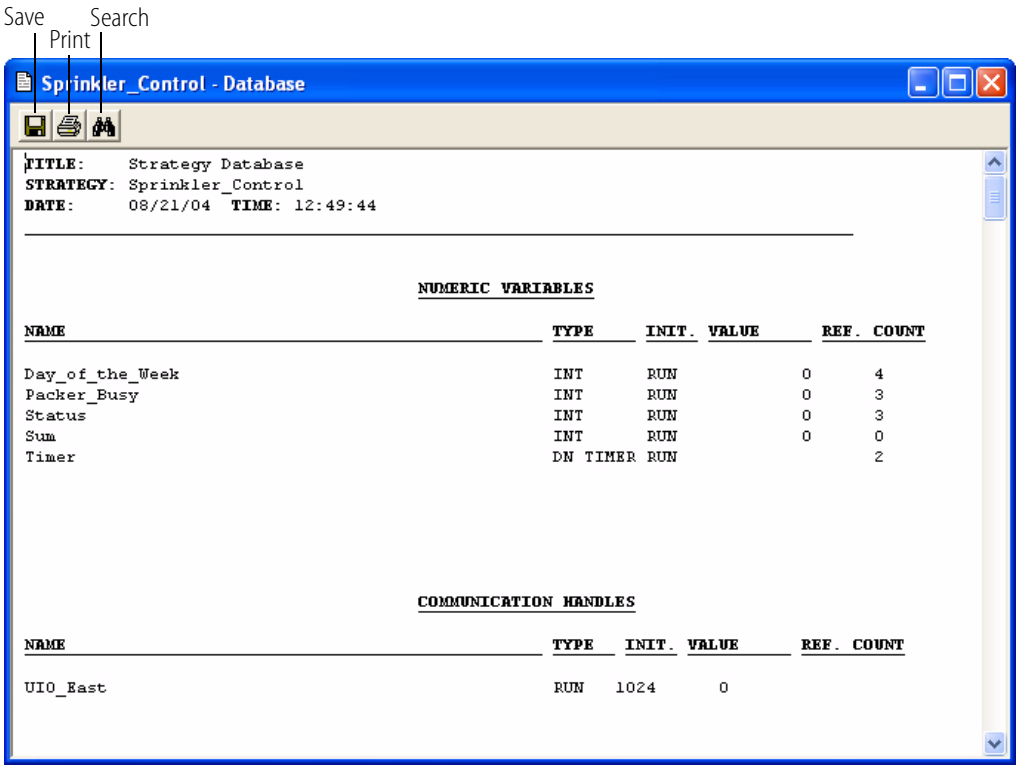
1. To view a summary of I/O elements and variables configured in a strategy, select File→View/Print→Database.
2. To view the same summary for a subroutine, open the subroutine window and select Subroutine→View/Print→Database.

   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.



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.

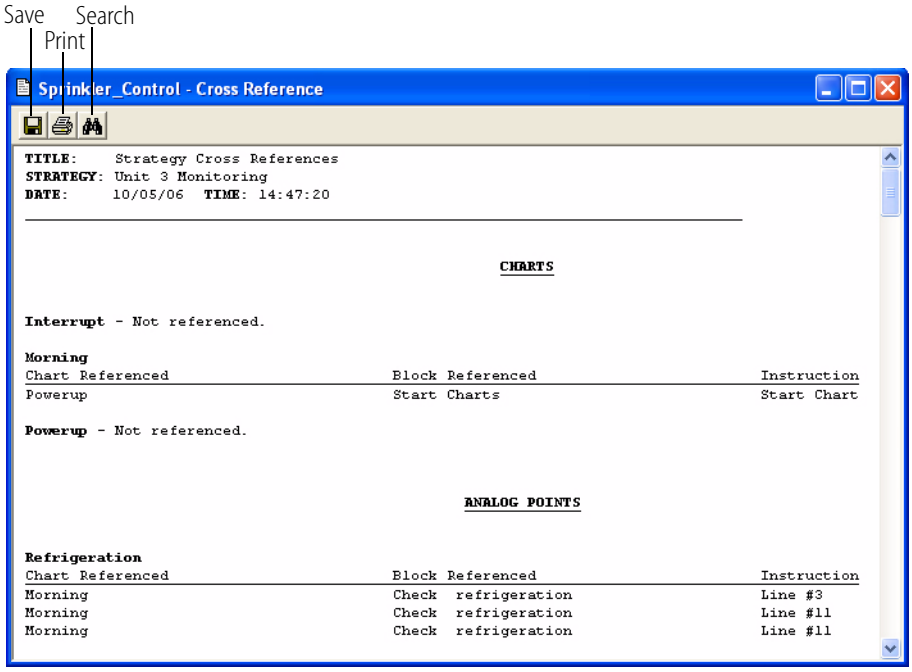
6. To print the data, click the print button on the toolbar. To save it to a text file, click the save button. To search the data, click the search button. When finished, close the window.

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.



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.

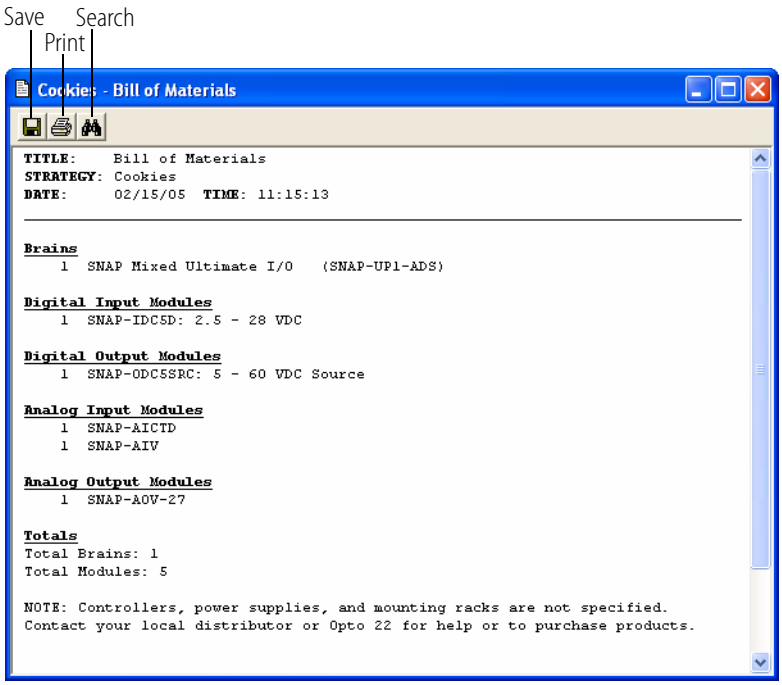
3. To print the data, click the print button on the toolbar. To save it to a text file, click the save button. To search the data, click the search button. When finished, close the window.

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.



You may need to resize the window and use the scroll bar to see all the data.

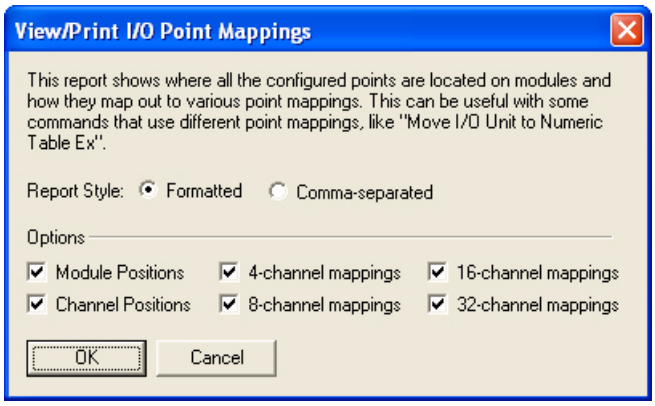
2. To print the data, click the print button on the toolbar. To save it to a text file, click the save button. To search the data, click the search button. When finished, close the window.

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.

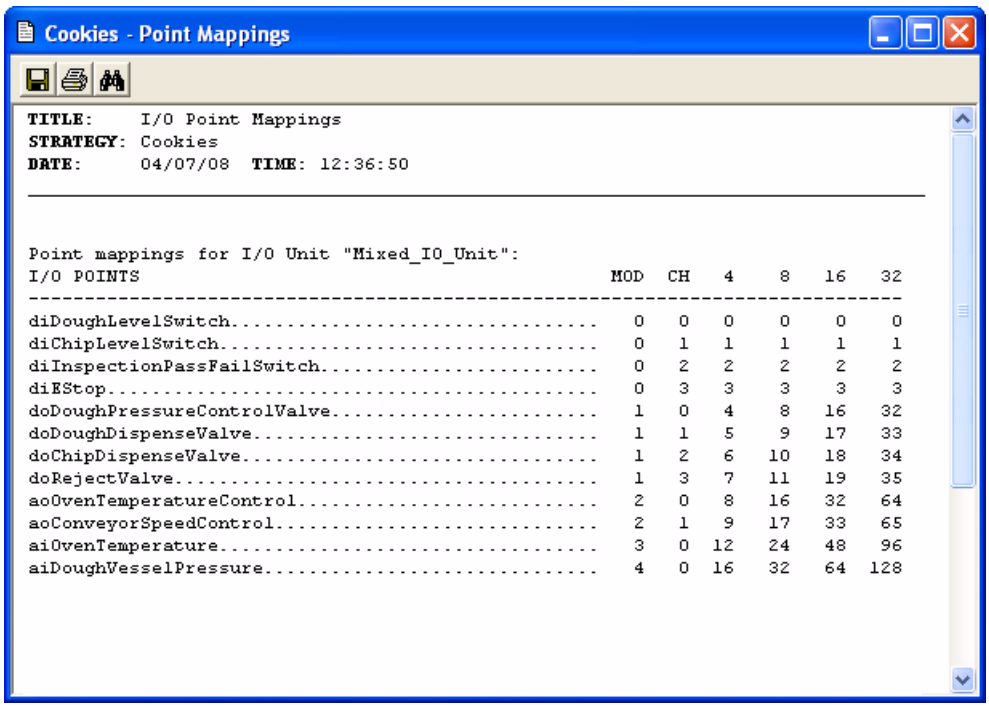
This report basically provides the same information provided in the appendix Table Index Offset Examples in the *PAC Control Command Reference*, form 1701, but for the actual points in the strategy.

1. To produce an I/O point mappings report for a strategy, open the strategy and select File→View/Print→I/O Point Mappings.



2. 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.



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.

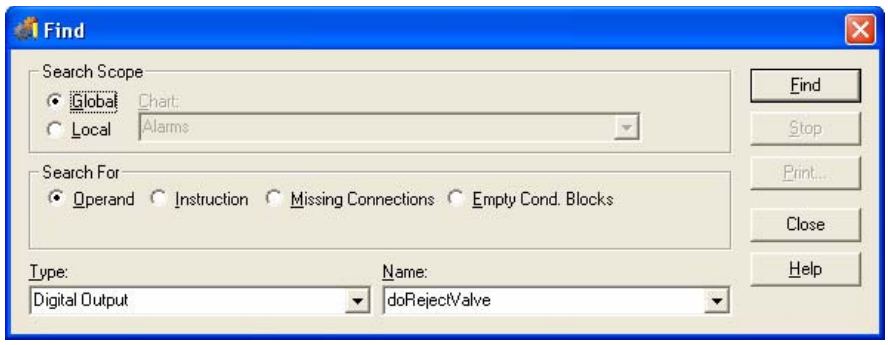
You can also replace instructions or operands with similar items.

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.

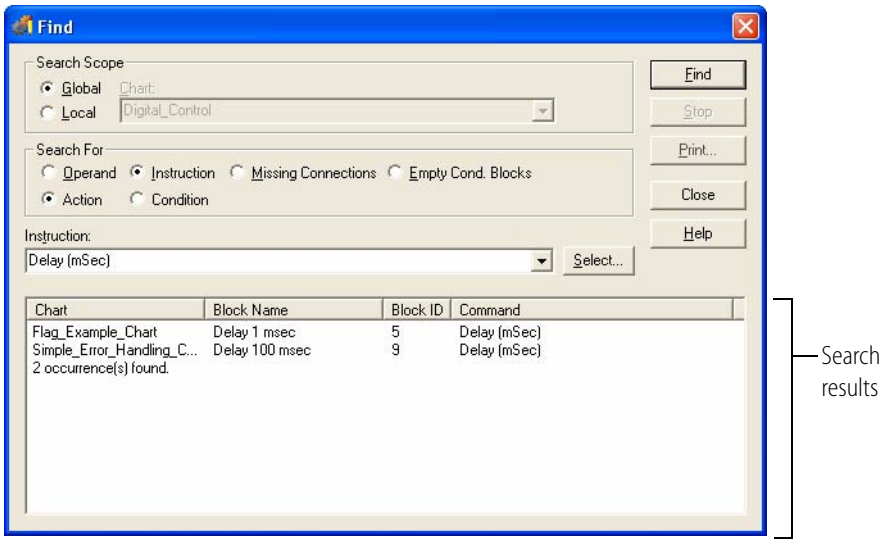


2. Under Search Scope, to search the entire strategy, click Global. To search one chart only, click Local and choose the chart name from the drop-down list.

   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.



For more information on any item in the search results, try double-clicking the item.

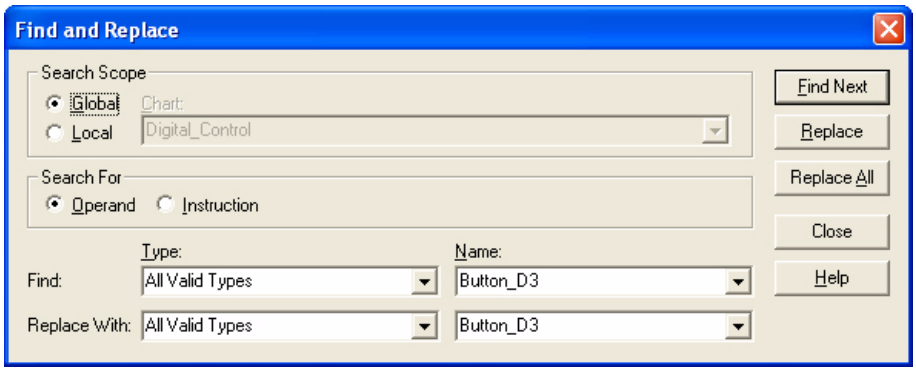
5. To save the search results to a file or to print them, click Print. In the window that opens, click the disk button to save or the printer button to print your search results.
6. When you have finished your search, close the Find dialog box.

Replacing

You can also replace any operand or instruction 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.



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.

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

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.

NOTE: If you enable legacy I/O capabilities in PAC Control, see the legacy versions of the PAC control guides for information on how to use them: forms 1710 and 1711.

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 now deprecated because they are not needed with SNAP PACs—such as High-Density Digital commands—will also be 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—will automatically be 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-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 *Clear 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 210).

   – Enable high-density digital commands (see page 210).

   – Enable mistic I/O units and commands (PAC Control Pro only; see page 211).
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 "deprecated" 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 types and commands will become available in the strategy:

I/O Unit Types	Other Types
G4D16R	Mistic PID Loop
G4D32RS	Digital Event/Reaction
G4A8R, G4RAX	Analog Event/Reaction
B200	Event/Reaction Group
B3000 (Digital)	
B3000 (Analog)	
B100	
SNAP-BRS	

Commands	
Clamp Mistic PID Output	Get & Restart Period
Clamp Mistic PID Setpoint	Get Event Latches
Clear All Event Latches	Get Frequency
Clear Event Latch	Get Mistic PID Control Word
Convert Mistic I/O Hex String to Float	Get Mistic PID D Term
Convert Number to Mistic I/O Hex String	Get Mistic PID I Term
Disable Communication to Event/Reaction	Get Mistic PID Input
Disable Communication to Mistic PID Loop	Get Mistic PID Mode
Disable Event/Reaction Group	Get Mistic PID Output Rate of Change
Disable Mistic PID Output Tracking in Manual Mode	Get Mistic PID Output
Disable Mistic PID Output	Get Mistic PID P Term
Disable Mistic PID Setpoint Tracking in Manual Mode	Get Mistic PID Scan Rate
Disable Scanning for All Events	Get Mistic PID Setpoint
Disable Scanning for Event	Get Off-Time Totalizer
Disable Scanning of Event/Reaction Group	Get On-Time Totalizer
Enable Communication to Event/Reaction	Get Period
Enable Communication to Mistic PID Loop	Get Period Measurement Complete Status
Enable Event/Reaction Group	IVAL Set Digital Binary [DEPRECATED]
Enable Mistic PID Output Tracking in Manual Mode	IVAL Set Frequency
Enable Mistic PID Output	IVAL Set Mistic PID Control Word
Enable Mistic PID Setpoint Tracking in Manual Mode	IVAL Set Mistic PID Process Term
Enable Scanning for All Events	IVAL Set Off-Totalizer
Enable Scanning for Event	IVAL Set On-Totalizer
Enable Scanning of Event/Reaction Group	IVAL Set Period
Event Occurred?	Mistic PID Loop Communication Enabled?
Event Occurring?	Read Event/Reaction Hold Buffer
Event Scanning Disabled?	Set Analog Totalizer Rate
Event Scanning Enabled?	Set Digital I/O Unit from MOMO Masks [DEPRECATED]
Event/Reaction Communication Enabled?	Set Mistic PID Control Word
Event/Reaction Group Communication Enabled?	Set Mistic PID D Term
Get Analog Filtered Value	Set Mistic PID I Term
Get Analog Square Root Filtered Value	Set Mistic PID Input
Get Analog Square Root Value	Set Mistic PID Mode to Auto
Get Analog Totalizer Value	Set Mistic PID Mode to Manual
Get & Clear Analog Filtered Value	Set Mistic PID Output Rate of Change
Get & Clear Analog Totalizer Value	Set Mistic PID P Term
Get & Clear Event Latches	Set Mistic PID Scan Rate
Get & Restart Off-Time Totalizer	Set Mistic PID Setpoint
Get & Restart On-Time Totalizer	Transmit/Receive Mistic I/O Hex String

Working with Flowcharts

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.

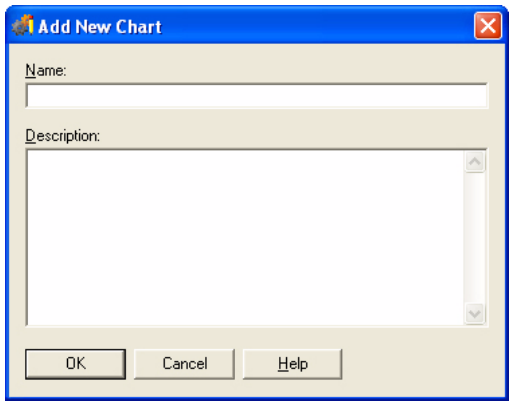
In this Chapter

Creating a New Chart.....213
Working with Chart Elements 214
Opening, Saving, and Closing Charts.....226
Copying, Renaming, and Deleting Charts 226
Printing Charts.....228
Exporting and Importing Charts.....228

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.



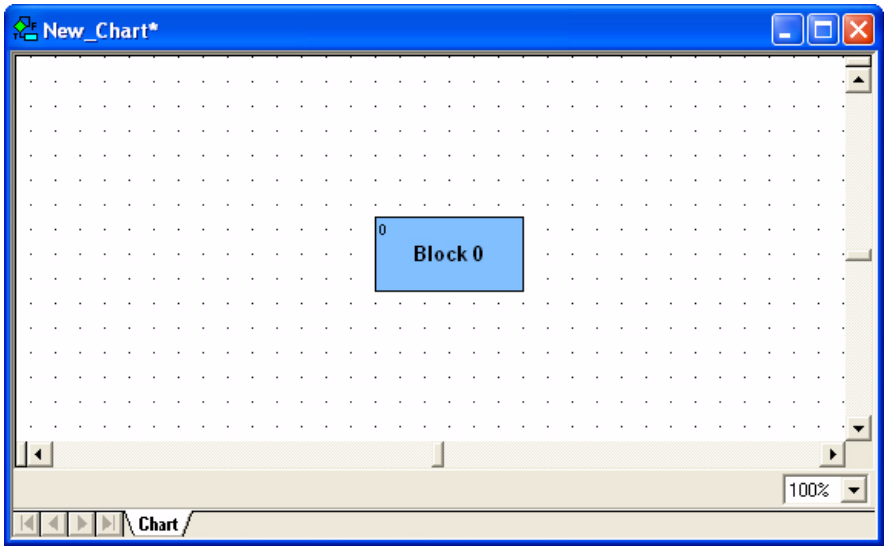
2. Enter a name for the new chart.

   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.

Working with Chart Elements

What's In a Chart?

Charts can contain four 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 Chapter 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 Chapter 11, "Using OptoScript," for more information. OptoScript blocks can have more than one entrance but only one exit.

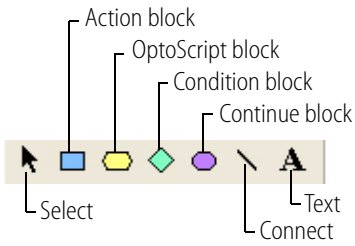
Continue Blocks are ovals that contain no commands, but simply route chart logic to a new location, such as to the top of a chart. These blocks help keep charts neat by avoiding awkward connections between two blocks that are far apart.

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 217 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.

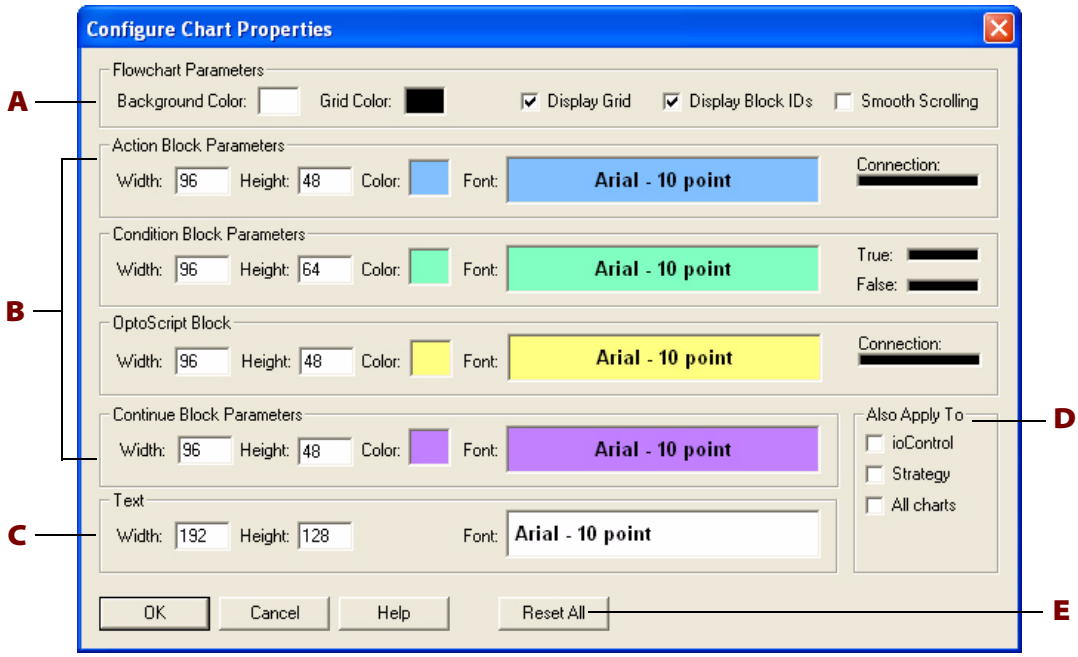
– To change all new charts in the open strategy only, choose File→Strategy Properties to open the Configure Strategy Properties dialog box.

– 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



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 elements 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.

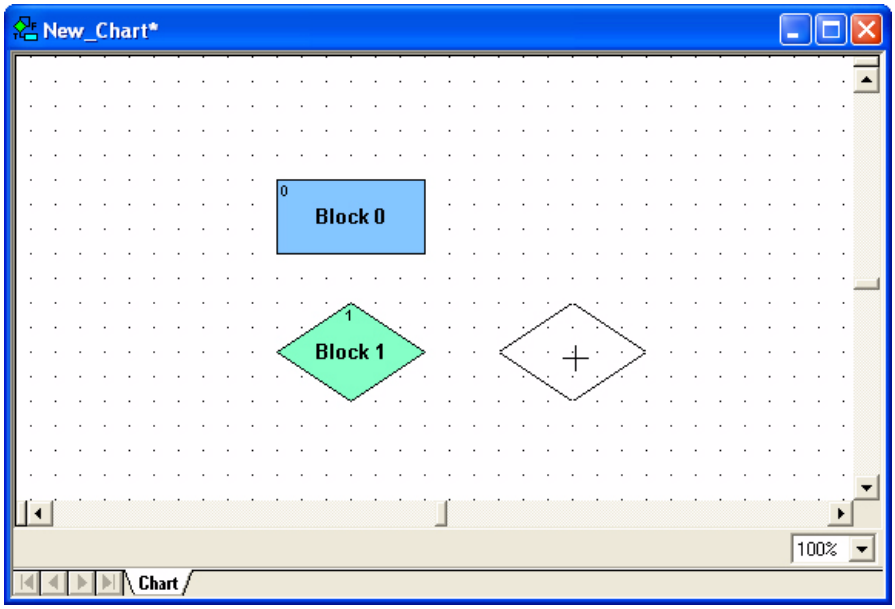
Drawing Blocks

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.

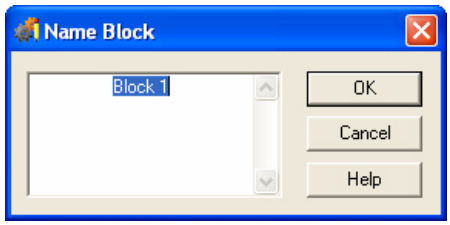
   The new block appears.



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.

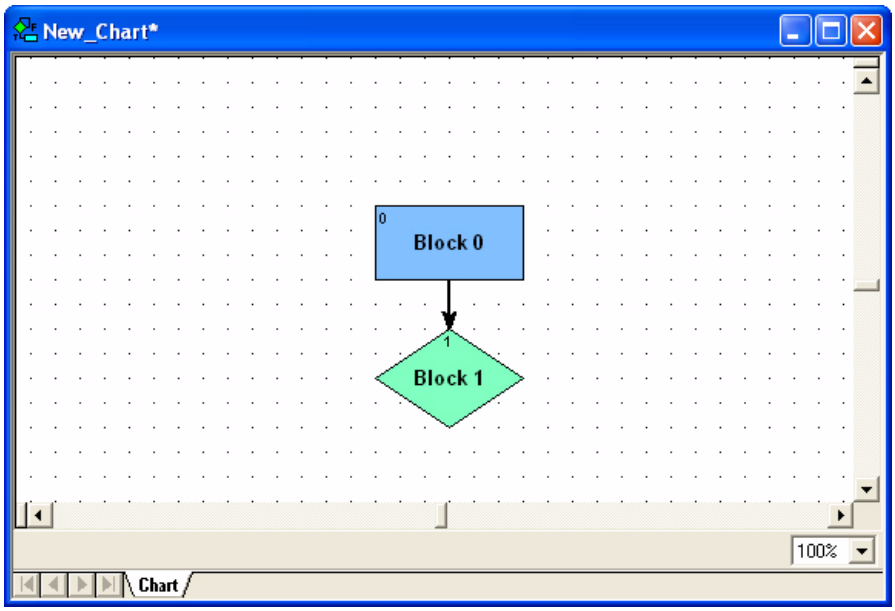
Connecting Blocks

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.

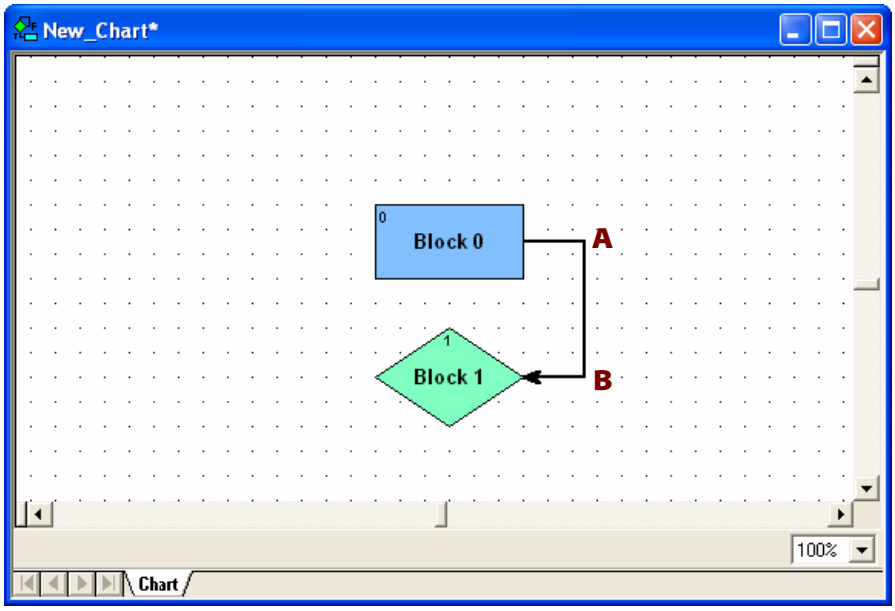
   Although you can click anywhere inside the blocks to make a connection, the connection is attached at the side closest to where you clicked. In the figure below, Block 0 is the source block and Block 1 is the destination block:



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:

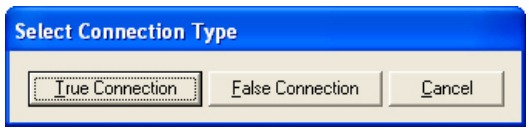


4. While you're still drawing a line, to delete an elbow you don't want, click the right mouse button once to undo it.

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.

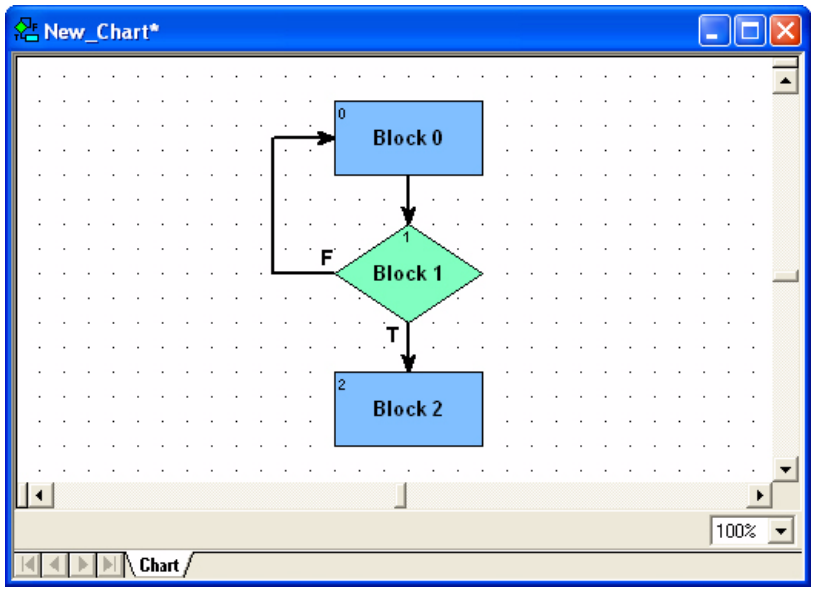


3. Indicate whether you are drawing the True connection or the False connection, and then click OK.
4. Click the destination block you chose (True or False).

   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 A.

   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 225.

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.

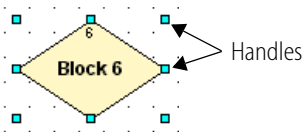
   The text block stays the same width but changes length to accommodate additional or deleted text. To change the size or shape of the text block, see "Resizing Blocks or Text Blocks" on page 225.

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.

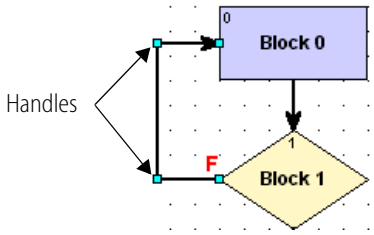
1. Click the Select tool.
2. To select an action, OptoScript, condition, continue, or text block, click the block.

   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. See the 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 choose 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, select Edit→Paste, or right-click anywhere on a chart and 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.

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 215.

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 215.

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 button 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.

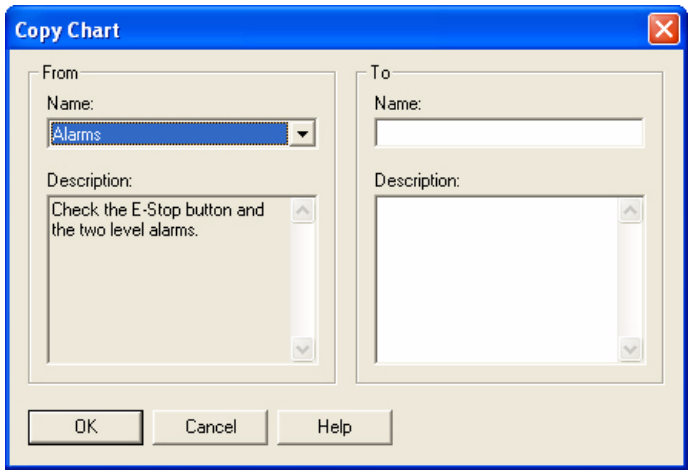
Copying, Renaming, and Deleting Charts

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 228.

1. With the strategy open and in Configure mode, select Chart→Copy.

   The Copy Chart dialog box appears.



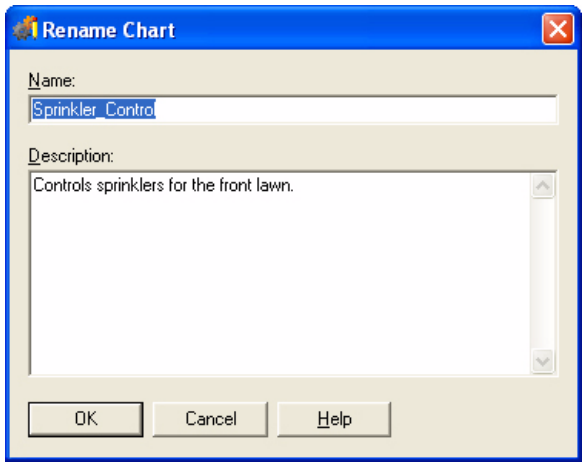
2. In the From field, choose the chart you want to copy from the drop-down list.
3. In the To field, enter a name for the new chart.

   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.



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 198. To print commands (instructions) for the chart, see "Viewing and Printing Strategy or Subroutine Commands" on page 201.

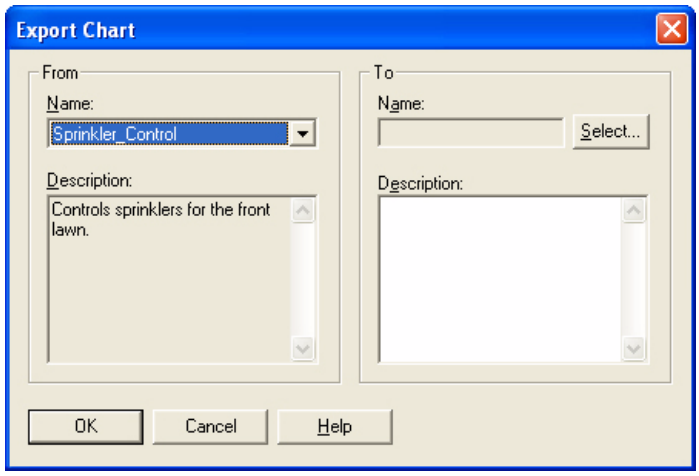
Exporting and Importing Charts

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.

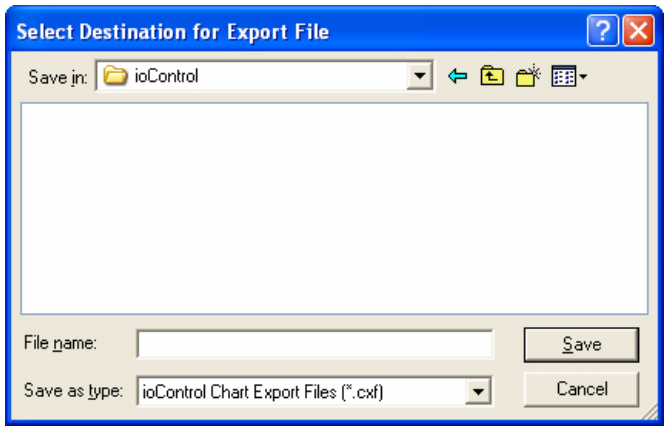
Exporting a Chart

1. With the strategy open and in Configure or Online mode, choose Chart→Export.

   The Export Chart dialog box appears.



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.



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.

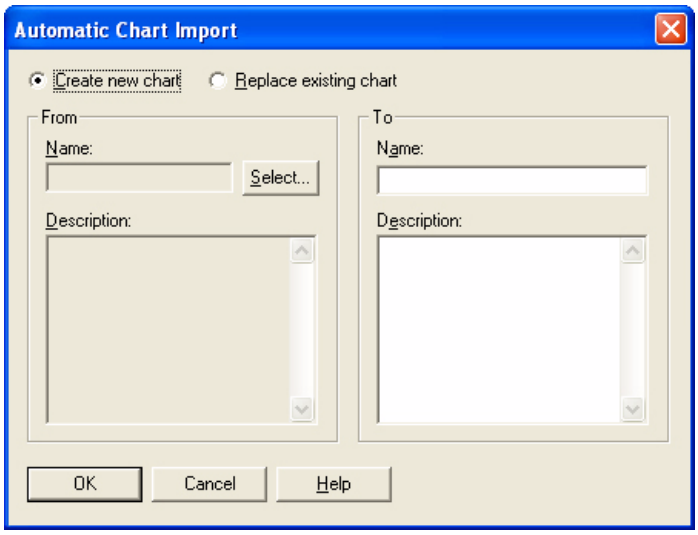
   You return to the Export Chart dialog box, which now shows the path and file name in the To section.
5. (Optional) Enter a description for the new chart.
6. Click OK.

   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

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, choose the chart to be replaced. 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.

Using Variables and Commands

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

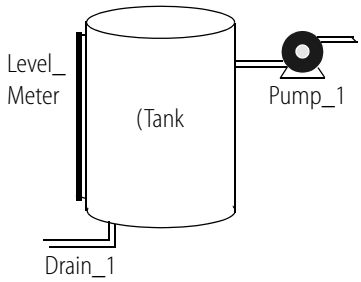
About Variables 231
Variables in PAC Control 233
Adding Variables 235
Adding Tables 239
Changing a Configured Variable 244
Viewing Variables in Debug Mode 244
Adding Commands 249
Changing a Command 253
Deleting a Command 253
Cutting or Copying a Command 254
Configuring a Continue Block 255
Viewing and Printing Chart Instructions 256

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. . 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 4.611686×10^{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 359.)

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. 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. 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.
- 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.

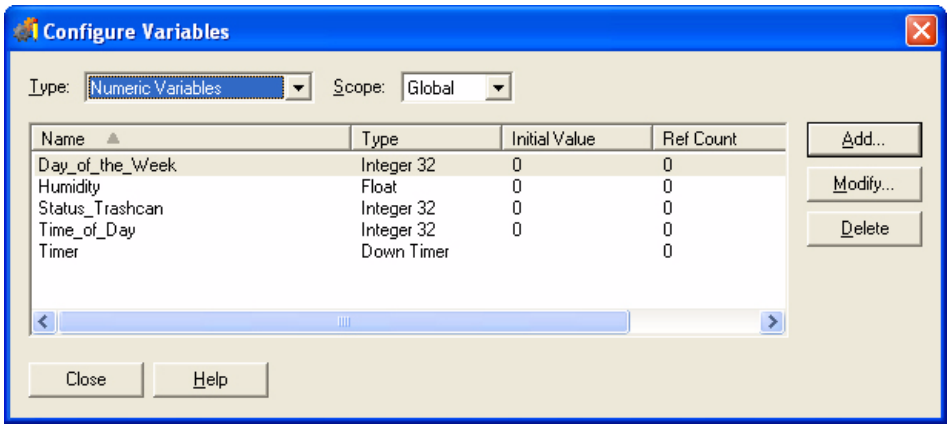
If you are using subroutines, see the information on literals in "Data Types for Subroutines" on page 344.

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 239.)

1. With the strategy or subroutine open in Configure mode, click the Configure Variables button on the toolbar or choose Configure→Variables.

   The Configure Variables dialog box opens.



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. To have data in the variable be persistent, select Persistent in the Scope drop-down list.

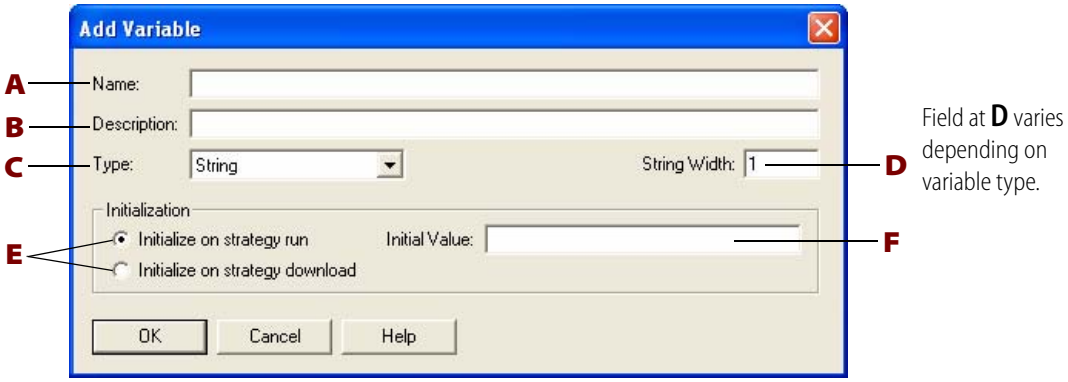
   You must choose Persistent as the scope *before* creating the variable; existing variables cannot be changed to be persistent. Pointer variables and timers cannot be persistent. Global is the default scope. For more information, see "Persistent Data" on page 235.
4. If you are adding the variable to a subroutine, select Subroutine in the Scope drop-down list.
5. To add a new variable, click Add.

   The Add Variable dialog box appears.
6. Complete the fields as described in "Add Variable Dialog Box" below.
7. 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 232.

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) To set the variable to the initial value (F) each time the strategy is run (either manually from the debugger or automatically via the autorun flag), click Initialize on Strategy Run.

To set the variable to the initial value (F) only when a strategy is downloaded, click Initialize on Strategy Download. 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. See "If You

Have Memory Problems" on page 359. To keep a variable's current value through both power cycles and strategy download, make the variable a persistent variable (see "Persistent Data" on page 235).

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. 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 261 for information and examples.

- If you are talking to 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 See Opto 22 form #1191, the *SNAP Serial Communication Module User's Guide*, for port numbers.
- If you are talking to another SNAP PAC brain, be aware of port numbers that are reserved for a specific protocol. For more infomration, see the section on Networking in form #1690, the *SNAP PAC Brains User's Guide*.
- 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.
- 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 intially, 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

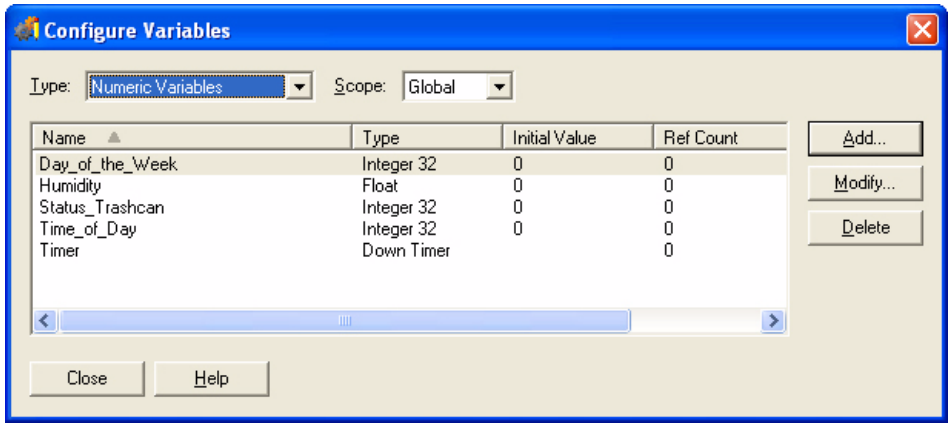
See the following topics to add numeric, string, and pointer tables:

- "Adding Table Variables" (below)
- "Setting Initial Values in Tables During Strategy Download" on page 240

Adding Table Variables

1. With the strategy or subroutine open in Configure mode, click the Configure Variables button 123 ABC on the toolbar or choose Configure→Variables.

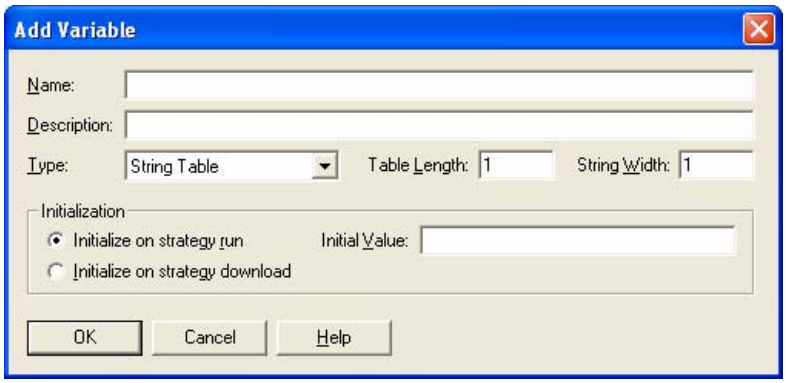
   The Configure Variables dialog box opens, listing all variables of one type in the strategy.



2. In the Type drop-down list, choose the type of table variable you want to add.
3. To have data in the table be persistent, select Persistent in the Scope drop-down list.

   Pointer tables cannot be persistent. Global is the default scope. For more information, see "Persistent Data" on page 235.
4. If you are adding the table variable to a subroutine, select Subroutine in the Scope drop-down list.
5. To add a new table variable, click Add.

   The Add Variable dialog box appears.



The figure above shows the dialog box as it appears for string tables. Fields are slightly different for other table variables.

6. Complete the fields as described in "Add Variable Dialog Box" on page 236.

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 Tables During Strategy Download."

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 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 included with PAC Control. 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
```

Download compression removes the new lines 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, 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.

Variables Example To set initial values of 123 for the variable INTEGER_VARIABLE, 456.789 for FLOAT_VARIABLE, "String Variable Test String" for STRING_VARIABLE, and to have pointer PTR_POINTER_VARIABLE point initially to INTEGER_VARIABLE, you would include the following text in the initialization file:

```
123 ^INTEGER_VARIABLE @!
456.789 ^FLOAT_VARIABLE @!
*STRING_VARIABLE $INN
String Variable Test String
^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
*	string
PTR_	pointer variable
~	I/O point
&	chart

Character	Object Type
}	float table or integer table
{	string table
PTBL_	pointer table
%	I/O unit

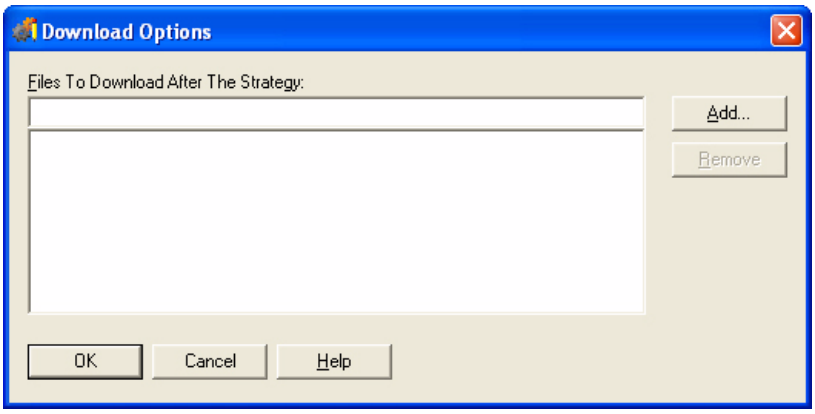
Saving the Initialization File When you have finished modifying the file, save it as a text file (.txt file extension) to your strategy directory. You can name the file anything you like. (You can also save it to any directory you like, but it is good practice to save each initialization file to the 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, click the Configure Control Engines button on the toolbar, 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.

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, click the Configure Variables button on the toolbar or 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 235 or "Adding Tables" on page 239.

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.

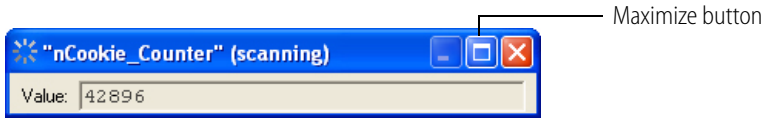
Viewing Variables in Debug Mode

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.

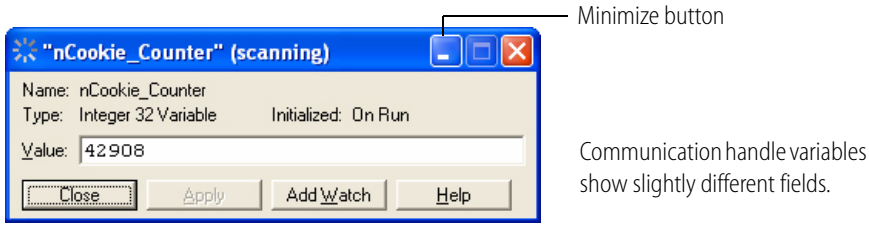
Viewing Numeric, String, and Communication Handle Variables

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 information, click the Maximize 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 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.

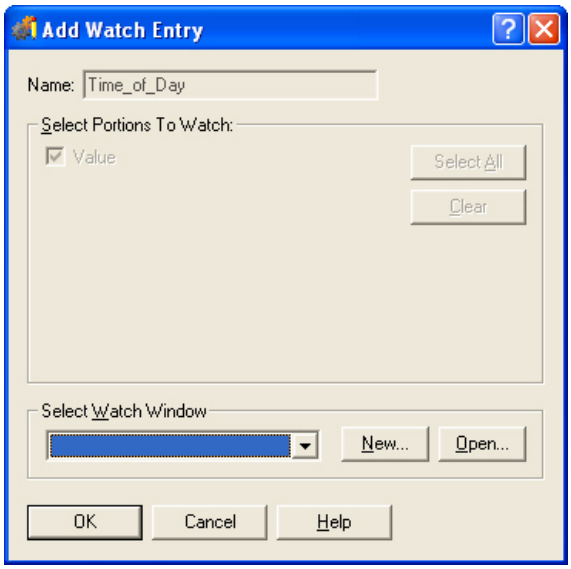
   For a string variable, if your change lengthens the string beyond its maximum width, the string is truncated to fit.

   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*.

4. To monitor the variable in a watch window, click Add 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.



5. Check the items you want to watch.

   Items to watch vary depending on the variable type.

6. 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 169.)

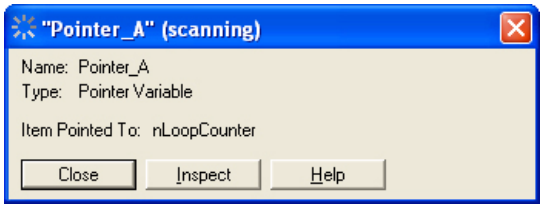
7. When the Add Watch Entry dialog box shows the correct items to watch and the watch window you want, click OK.

   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.



2. To view the status or value of the item pointed to, click the Inspect button.

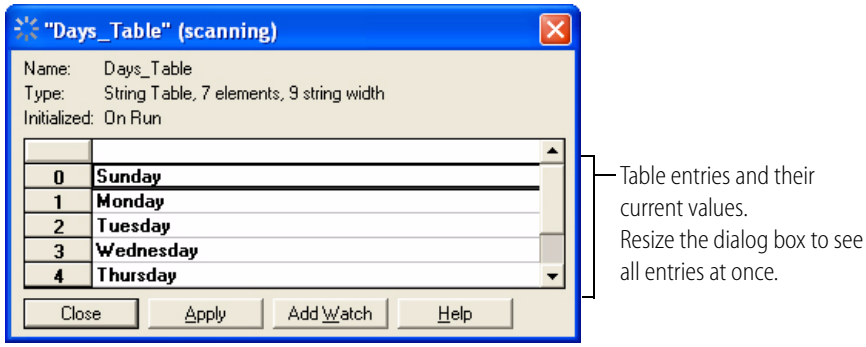
If you need help, follow the steps in "Viewing Numeric, String, and Communication Handle Variables" on page 244.

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.



   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 a table entry, click its index number, highlight the value, and type in a new value. Click Apply.
3. To monitor the table in a watch window, click Add Watch.

   The Add Watch Entry dialog box appears.

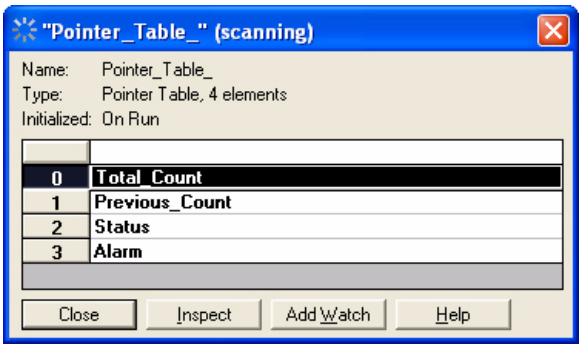
4. In the Add Watch Entry dialog box, do one of the following:
   - If the watch window you want to use to monitor the table variable is open, choose it from the Select Watch window drop-down list.
   - If the watch window you want is not open, click Open. Navigate to it and double-click it to open it.
   - If you want to monitor the variable in a new watch window, click New. (For help, see "Creating a Watch Window" on page 169.)
5. Select the indexes you want to watch.
6. 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.



2. To view the status or value of the item pointed to, highlight it in the table and click the Inspect button.

   If you need help, follow the steps in "Viewing Numeric and String Tables" on page 247.

Adding Commands

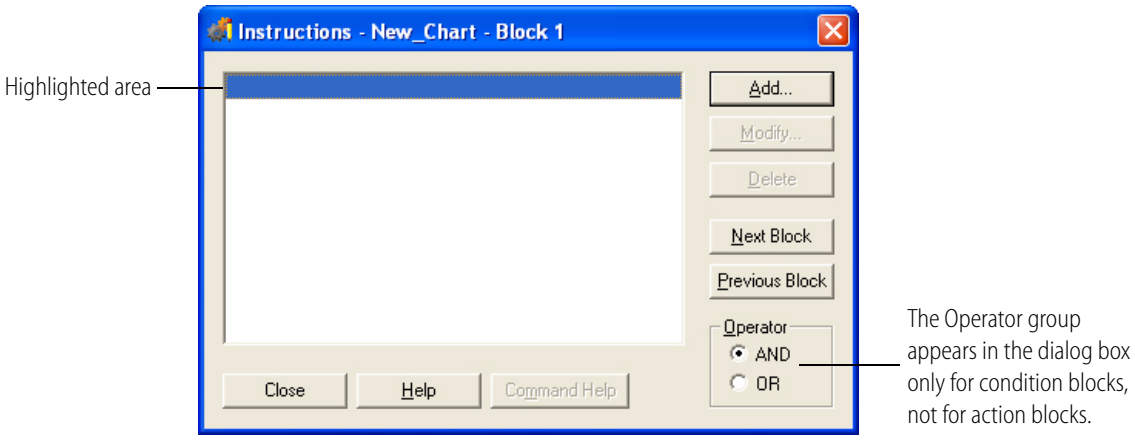
To make a block in a strategy flowchart do the work it's intended to do, you add one or more commands. Commands use the I/O points and variables you've already configured, as well as other elements in your strategy. A command, for example, might turn on a digital point, move a value to a variable, or 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 255.)

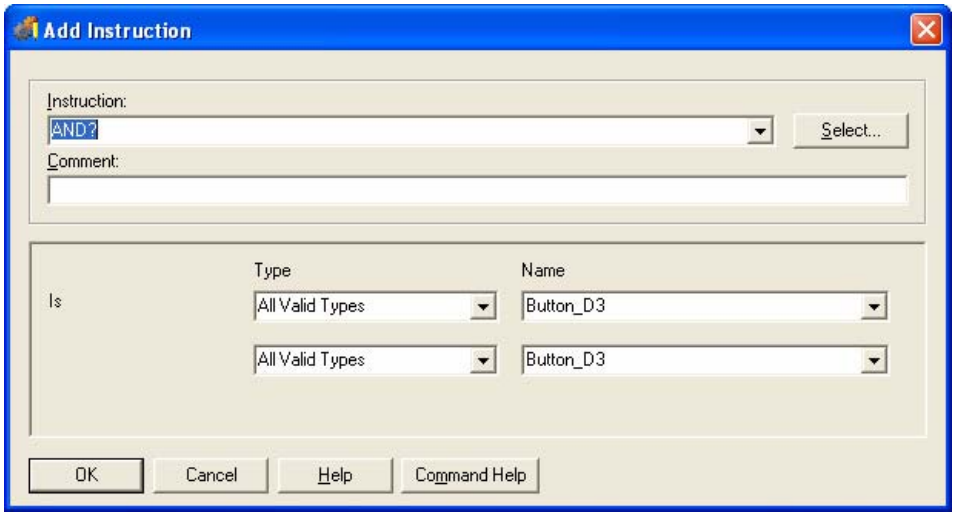
To add commands to an action or condition block, follow the steps below. (To add commands to an OptoScript block, see "OptoScript Functions and Commands" on page 322 and "Using the OptoScript Editor" on page 337.)

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.

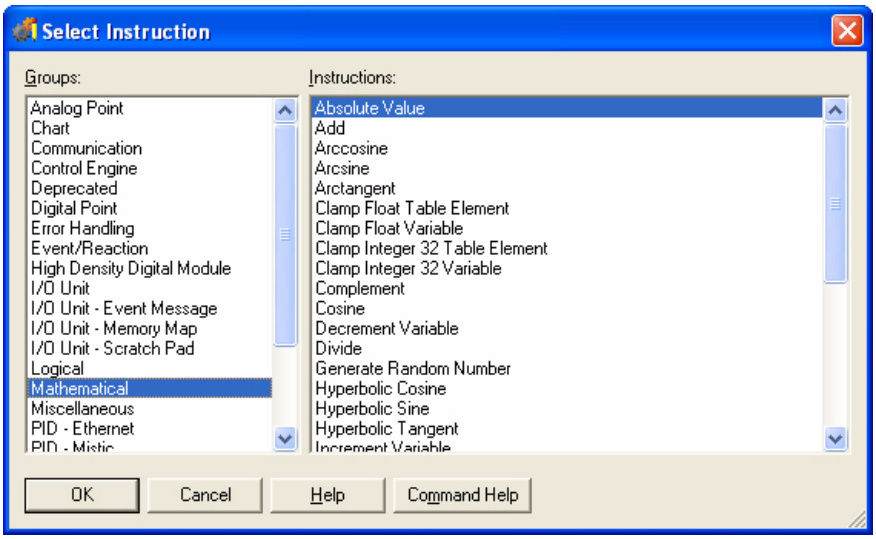
   The Instructions dialog box appears. The highlighted area shows where the new command will be added.



2. Click Add to open the Add Instruction dialog box.



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.
4. If you don't know the command name, click Select to open the following dialog box.



5. 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.

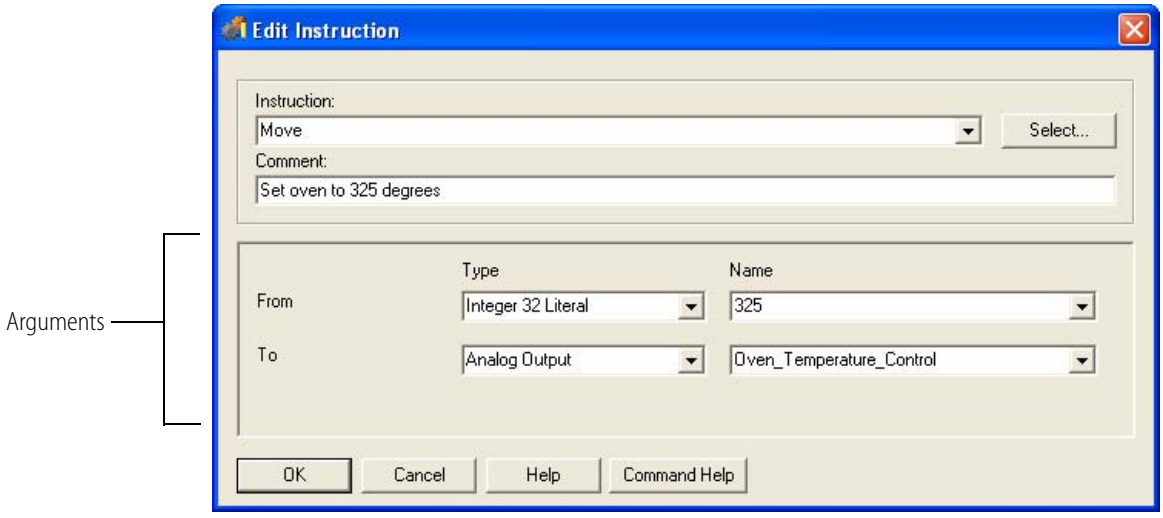
   For more information on any command, click the command name and click Command Help to open online help. You can also look up the command in the *PAC Control Command Reference*.

6. When the command you want is highlighted, click OK to return to the Add Instruction dialog box.

7. (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.

8. Complete each argument for the command by typing in the Type and Name fields or by choosing from the drop-down lists.



If the argument type is a literal (or constant), you must type it in.

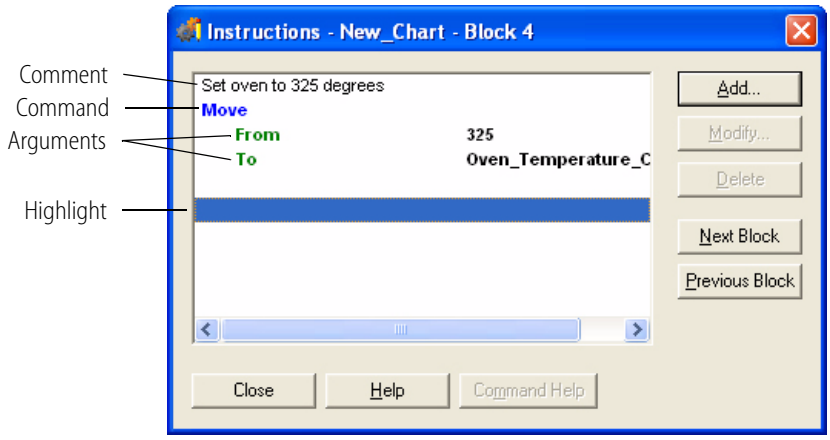
If you type in the name of an item that doesn't exist, for example a variable or I/O point, you are prompted to add it to the strategy.

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.

9. 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:



10. 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.

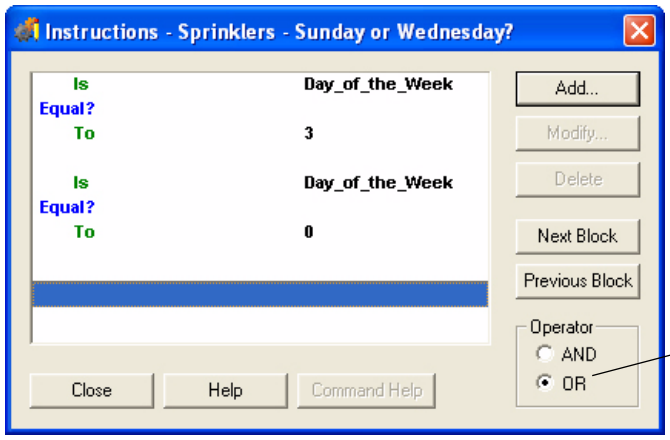
TIP: If you are adding commands to several blocks at once, you can quickly move from block to block in the chart by clicking the Next Block or Previous Block buttons in the Instructions dialog box.

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.

11. 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.



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 337.*

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 249.

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 337.*

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 337.

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 337.

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.

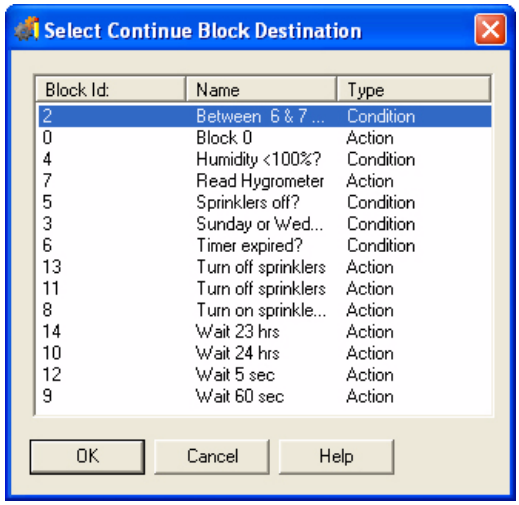
Configuring a Continue Block

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.



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 201.

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 the *PAC Control Command Reference*.

In this Chapter

Analog Point Commands 258
Chart Commands 260
Communication Commands 261
Control Engine Commands 278
Digital Point Commands 279
Error Handling Commands 283
I/O Unit Commands 284
I/O Unit—Event Message Commands 286
I/O Unit—Memory Map Commands 287
I/O Unit—Scratch Pad Commands 288
Logical Commands 290
Mathematical Commands 291
Miscellaneous Commands 295
PID—Ethernet Commands 296
Pointer Commands 299
Simulation Commands 301
String Commands 301
Time/Date Commands 310
Timing Commands 310

Analog Point Commands

The following commands are used with analog points:

Offset and Gain

Calculate & Set Analog Gain
Calculate & Set Analog Offset
Set Analog Gain
Set Analog Offset

Others

Set Analog Filter Weight
Set Analog Load Cell Fast Settle Level
Set Analog Load Cell Filter Weight
Set Analog TPO Period

Minimum/Maximum Values

Get Analog Maximum Value
Get Analog Minimum Value
Get & Clear Analog Maximum Value
Get & Clear Analog Minimum Value

Offset and Gain Commands

The easiest way to set offset and gain is to do so when you configure analog points in PAC Manager, using the Calibrate button in the Configure I/O Points dialog box.

You can also set offset and gain in PAC Control. If you already know the offset and gain for a point, you can use the commands Set Analog Offset and Set Analog Gain. If you do not know the offset and gain, you can use the commands Calculate & Set Analog Offset and Calculate & Set Analog Gain to have the brain calculate them. Calculate offset first, and then calculate gain.

By setting offset and gain, you make sure that values read are accurate.

Offset is the difference between the minimum input of an analog input point and the actual minimum signal received from a field device. For example, if a 4–20 mA input receives a minimum signal that is slightly off (not exactly 4 mA), the difference between the two minimums is the offset. Reading ± Offset = Actual Value. For example:

If minimum input =	4.000 mA
and zero-scale reading =	4.003 mA
then offset =	-0.003 mA

Gain is the difference in the full-scale reading, but expressed differently. Measured Value * Gain = Actual Value. For example:

If maximum input =	20.00 mA
and measured value =	20.50 mA
then gain =	0.9756097560976

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 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 322.

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 50.

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 "Increasing Efficiencies in Your Strategy" on page 96.

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 String
Clear Communication Receive Buffer	Receive String Table
Close Communication	Send Communication Handle Command
Communication Open?	Set Communication Handle Value
Get Available File Space	Set End-Of-Message Terminator
Get Communication Handle Value	Transfer N Characters
Get End-Of-Message Terminator	Transmit Character
Get Number of Characters Waiting	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 String
Receive Pointer Table	

* PAC Control Professional only; mistic I/O units only

Communication Handles

A communication handle is a variable in PAC Control that stores the parameters needed to connect to a specific entity; 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 the communication handle variable is a string consisting of the communication parameters separated by colons.

Typically, a communication handle is used to open communication, to transmit and receive data, and to close communication. For example, you might use the TCP communication handle to communicate with another device on the network via TCP/IP, or the FTP communication handle to send data from the brain to a file on a PC.

In many cases you'll define a communication handle variable and never change it, because the same parameters are always needed. However, in some cases it might be useful to change it. For example, if you use the same process to send serial data through two different serial communication modules, you can use Set Communication Handle Value to switch to the other module. Advanced users may also want to use pointers.

Set Communication Handle Value also allows you to retrieve the communication timeout value for some types of communication handles, and you can specify the communication timeout value in seconds.

Several types of communication handles are available:

TCP	For Ethernet communication For communication with serial modules	See page 262. See page 265.

File	For creating a file to be stored on the control engine, and reading or writing to a file stored on the control engine	See page 268.
FTP	For accessing files on the local file system or another FTP server, including transferring files between the two servers	See page 273.
Serial	For using the RS-232 connectors on a SNAP PAC controller to communicate with serial devices. (Not used for serial communication modules or serial-based I/O units.)	See page 276.

Using TCP Communication Handles

The TCP communication handle is used for most Ethernet communication. Any device on the network accessible via TCP/IP can be communicated with using communication commands. For example, data from a serial device can be received through a serial communication module, or communication can be established with a PC on the network.

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 288 for more information.

NOTE: Be sure to use a separate 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.

Incoming and Outgoing Communication

TCP communication is normally requested by the device that needs the data. For example, a PAC Control strategy running on SNAP PAC I/O unit A might need the value of a set of variables from another strategy running on SNAP PAC I/O unit B. Unit A would request communication using the command Open Outgoing Communication. Unit B would Listen for Incoming Communication and then Accept Incoming Communication to establish the connection. Once connected, data could be transmitted and received in both directions.

This scenario is similar to the way we use the telephone. If Joe Wong needs information from Saul Garcia, he requests communication by calling Saul on the phone (opening outgoing communication). If Saul is there (listening for incoming communication), he answers the phone (accepts incoming communication), and the connection is completed. Once connected, both Joe and Saul can talk and listen to transmit and receive information.

Depending on the situation, you may want to have both peers request communication and establish two connections between them. That way if one peer 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. A serial device such as a barcode reader attached to a serial communication module on a SNAP PAC I/O unit is an example. The serial device is incapable of requesting Ethernet communication; essentially it acts as a slave device and can only wait until it is asked for data. For serial devices attached to serial communication modules, the control engine needing the serial data must request communication using Open Outgoing Communication. (For serial devices attached directly to a serial port on a SNAP PAC controller or brain, see "Using Serial Communication Handles to Communicate with Serial Devices" on page 276.)

Outgoing Communication For *outgoing* communication (communication you request), the communication handle's value is in the format `tcp:<IP address>:<port>`. Note that `tcp` is all lowercase letters. The following table shows examples of communication handles for outgoing communication.

Outgoing Communication	Communication Handle Value Protocol:IP Address:Port
Ethernet/TCP - to another control engine	`tcp:10.192.56.185:22004`
Ethernet/TCP - to a serial communication module on another rack	`tcp:10.192.54.10:22506`

Ethernet/TCP - to a serial communication module on the same rack (uses SNAP PAC R-series IP address or loopback IP address)	`tcp:10.192.55.90:22511`
	or
	`tcp:127.0.0.1:22511`

Port Numbers If you are talking to other control engines or other peers on the network, you can use any port number that is not used by another device on the network. For example, do not use the following port numbers, which are used by Opto 22 devices for the purposes shown:

Port #	Purpose
21	FTP (file transfer protocol)
20	FTP (file transfer protocol) for get, send, and dir functions
502	Modbus/TCP hardware and software
2001	Main command processor, OptoMMP-based (also used by SNAP PAC controllers and brains)
161	SNMP-based enterprise management systems
22000 22001	Host port (for information on configuring the host port, see form 1704, the *PAC Manager User's Guide*)

Note that port numbers on the SNAP PAC controllers and brains are the default ports; some may have been changed for security reasons. (See the section on Security in form 1704, the *PAC Manager User's Guide*.)

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 http://www.iana.org/assignments/port-numbers to see ports that may apply to your devices on your network.

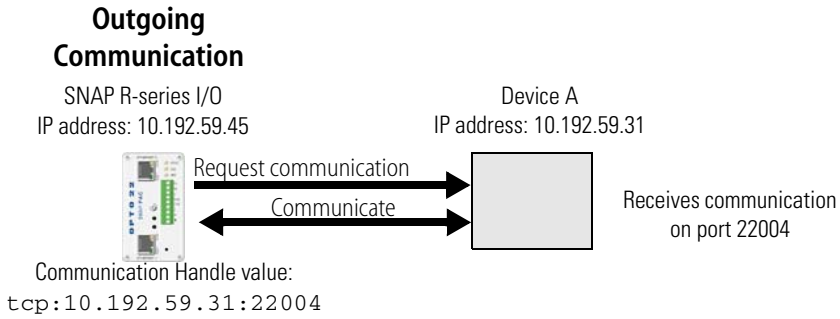
Incoming Communication For *incoming* communication (communication requested by another device), the communication handle value includes just the protocol and the port number: `Protocol:Port`. TCP automatically tracks senders so there is no mix-up in the data sent and received. Here are a couple of examples of communication handles for incoming communication:

Incoming Communication	Communication Handle Value
	Protocol:Port
Ethernet/TCP - from another control engine	`tcp:22004`
Ethernet/TCP - from another control engine	`tcp:22005`

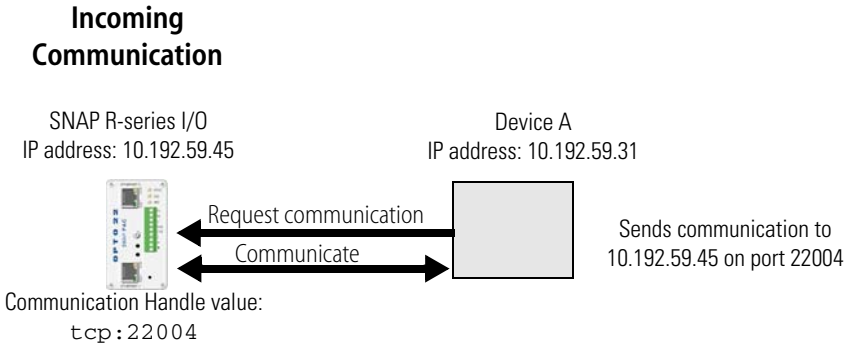
To add a communication handle, see the steps starting on page 235.

TCP Communication Handle Examples

The following diagram shows an example of a communication handle value for outgoing communication:



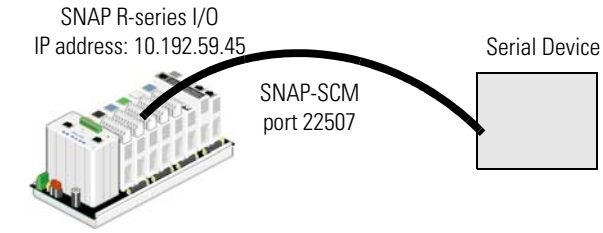
For incoming communication, the communication handle value could be:



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. For port number information, see 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.

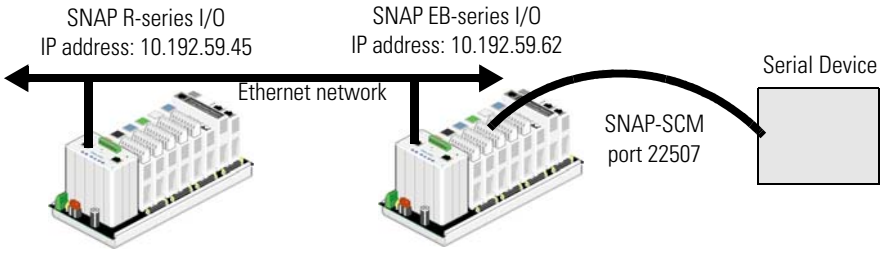
To a Serial Module on the Same Rack



Communication Handle value:
`tcp:127.0.0.1:22507`
or `tcp:10.192.59.45:22507`

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.

To a Serial Module on a Different Rack



Communication Handle value:
`tcp:10.192.59.62:22507`

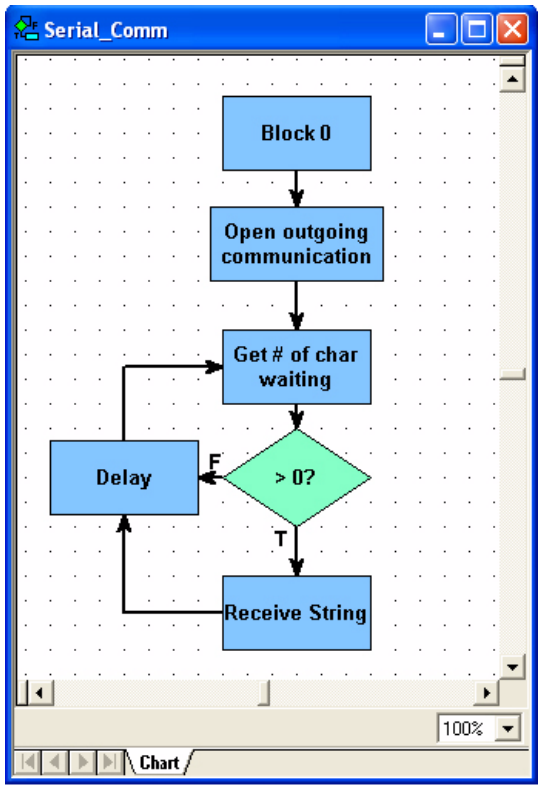
Using Flowcharts to Control TCP/IP Communication

When a control engine is communicating with another device using TCP/IP, it runs a PAC Control flowchart or charts to control communication.

- **For outgoing communication,** the flowchart uses the command Open Outgoing Communication to request a connection.
- **For incoming communication,** 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.

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 is done via TCP/IP. It can be done from the SNAP PAC R system itself, as in this example, or from another TCP/IP device on the network.

In this example, the R-series I/O unit listens for and accepts communication from the serial module using the communication handle:

```
tcp:127.0.0.1:22507
```

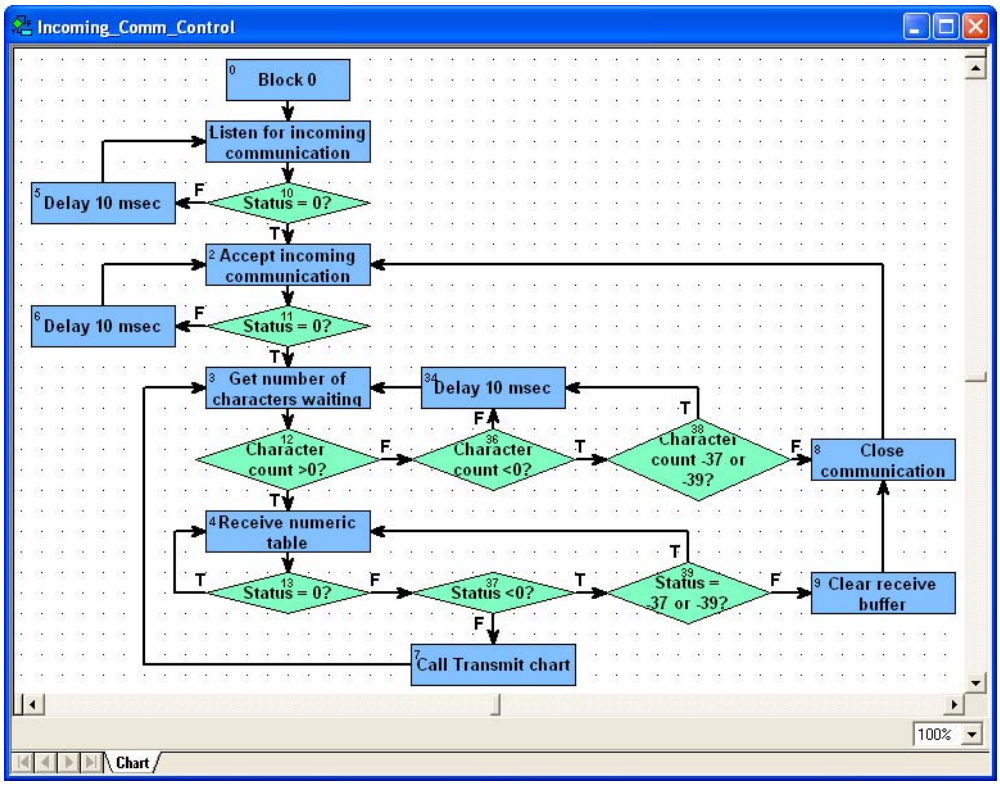
The loopback IP address is used, since the serial module is on the same rack.

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, 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, 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 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: http://www.iana.org/assignments/port-numbers

IMPORTANT: If you are using TCP/IP connections, consult commercially available texts on TCP/IP that discuss client/server architectures, so you'll understand how the protocol works and what to expect during communication.

Using the Control Engine's File System

The memory in SNAP PAC controllers includes a substantial area (about 4 MB on a SNAP-PAC-S1, or 2 MB on a SNAP PAC R-series controller) available for file storage. 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.

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.
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`

Keep the following limitations in mind as you use the file communication handle:

Maximum length for filenames 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 (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 278 for information on saving files to flash using PAC Control commands.

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.

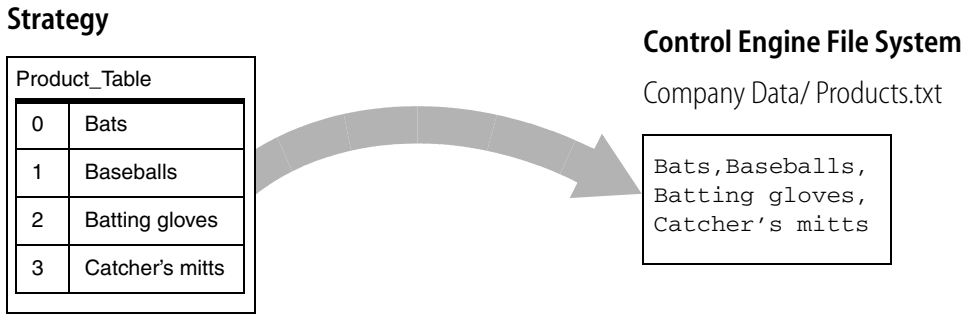
Commands to use with any file communication handle	Commands to use with file comm handles in Read mode	Commands to use with file comm handles in Write or Append mode
Close Communication Communication Open? Get Available File Space Get Number of Characters Waiting Open Outgoing Communication Send Communication 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`

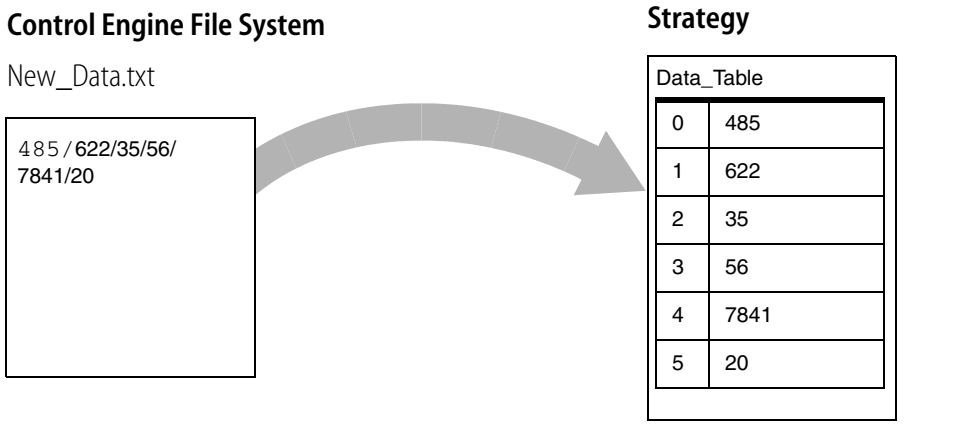
2. Use the condition Communication Open? to make sure the communication handle opened.
3. To put the data from the string table into a comma-delimited file (which is easy to open in database software), use the command Set End-Of-Message Terminator to indicate that a comma should be used as the delimiting character.
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.)
4. Use the command Receive String Table to receive the data from New_Data.txt directly into Data_Table.



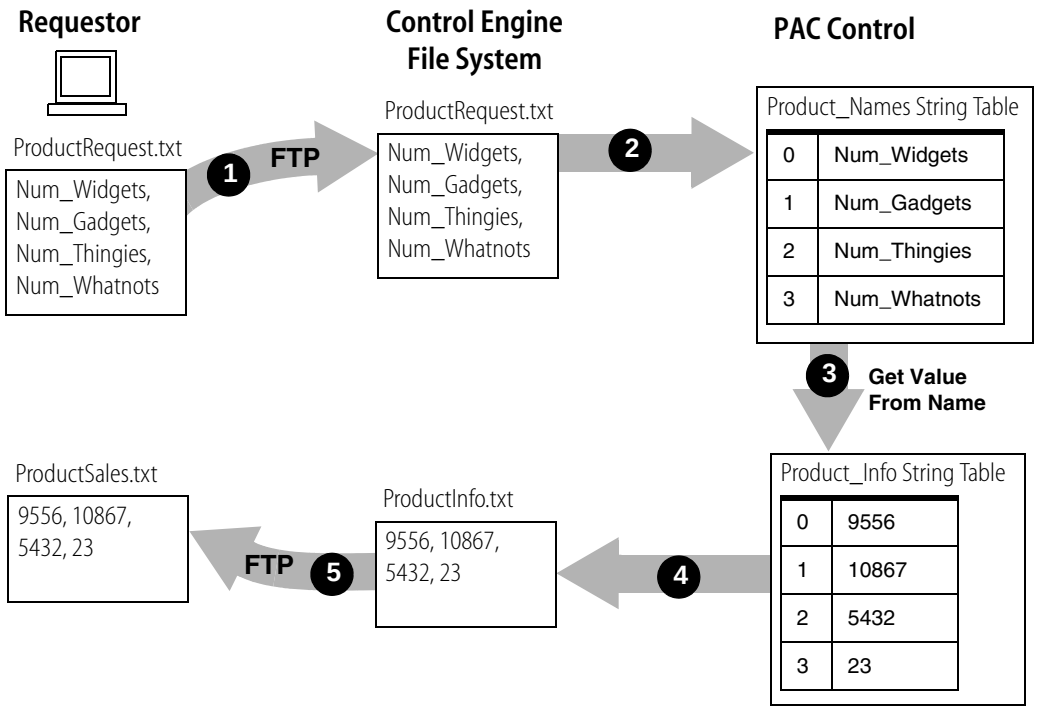
Notice the numbers used in this example. These are numbers represented as strings. For the purpose of storing and sending data, this is the simplest way to represent them. If you need to use them in calculations, however, you must first convert them to numeric values. You can do so in your PAC Control strategy by using a command such as Convert String to Float or Convert String to Integer 32. (See "String Commands" on page 301 for more information.)

5. 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 273 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],`
	`Product_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.

Moving Files via FTP

As explained in the previous section ("Using the Control Engine's File System," starting on page 268) 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.

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 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)

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 username 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 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.

Suppose you want to place the data into a file (HunkoData.txt) on the device; the data is currently in the file UseThisData.txt on the control engine. 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 filename and send the file in one step. The communication handle you send is:

   ```
   send:UseThisData.txt,HunkoData.txt
   ```

   If the remote filename already exists, it is overwritten.
4. When you have finished sending 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 filename 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 filename 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 271, 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. filename (remaining characters) and removes the 0D 0A left over on the end of the filename.
	`      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. 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).

First, the serial port must be set to None as in PAC Manager, as follows:

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 the *PAC Manager User's Guide*.

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 262*.

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 controllers.

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	SNAP-PAC-S1: 0, 1, or 2 SNAP-PAC-S2: 0, 1, 2, or 3
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

Setting	Valid Values
stop bits	1 or 2

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)

Method 2 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 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
```

NOTE: You must include at minimum the port number and at least one parameter. Otherwise, it will not work.

The keywords and possible values are:

Keyword	Valid Values
port	0-3, depending on controller
baud	230400*, 115200, 76800, 57600, 38400, 19200, 9600, 4800, 2400, 1200, or 300
parity	E,e,O,o,N,n,9
data	8 or 7
stop	1 or 2
mode	0,522(=I/O mode)
rts-cts	0,1*
timeout	float, seconds

* Enables symmetrical rts-cts flow control.

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 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 the *PAC Control Command Reference* or online Help for details on using specific commands.

Control Engine Commands

The following commands refer to the control engine:

- Calculate Strategy CRC
- Erase Files In Permanent Storage
- Get Available File Space
- Get Control Engine Address
- Get Control Engine Type
- Get Firmware Version
- Load Files From Permanent Storage
- Retrieve Strategy CRC
- Save Files To Permanent Storage

Commands Relating to Permanent Storage

The term "Permanent Storage" in the last three 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.

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 268. 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*

Clear Counter
Get Counter
Get & Clear Counter
Start Counter
Stop Counter

Period and Frequency*

Get Frequency
Get Period
Get Period Measurement Complete Status
Get & Restart Period
Set TPO Percent
Set TPO Period

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?

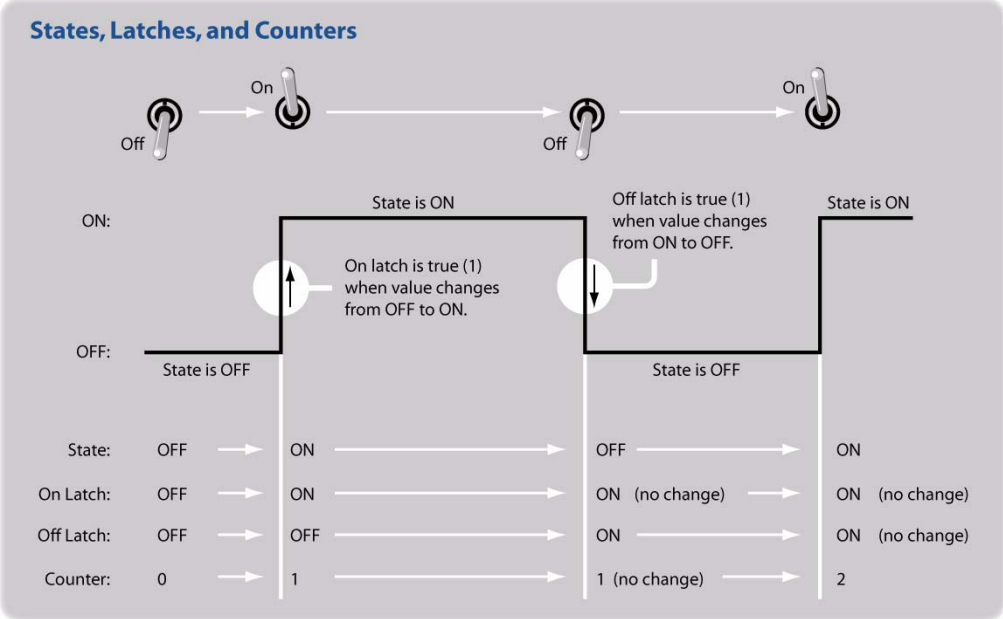
Pulses*

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
Get & Restart On-Pulse Measurement
Start Continuous Square Wave
Start Off-Pulse
Start On-Pulse

*Some digital point commands are available in PAC Control Professional only. Some commands are available only on some I/O units. For details, see specific information for each command in the *PAC Control Command Reference* or 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.



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; see the brain's and module's data sheets for specifications.

Before using a counter, you must configure the point as a counter. (See "Adding a Digital I/O Point" on page 130 or use PAC Manager.) The counter begins counting as soon as it is configured. You do not need to use the Start Counter command to start it.

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.

Counters start as soon as they are configured, the Start Counter command is only used after Stop Counter, and a positive value means that phase A leads phase B. See additional details in the *PAC Control Command Reference* or online Help.

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 129 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.

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 only when a strategy in the control engine reads or writes to an I/O point.

Do not be concerned if the IVAL does not match the XVAL. A mismatch just means that the program 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 301.)

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 322.

Error Handling Commands

The following commands refer to handling errors:

- Add Message to Queue
- Add User Error to Queue
- Add User I/O Unit Error to Queue
- Caused a Chart Error?
- Caused an I/O Unit Error?
- Clear All Errors
- Copy Current Error to String
- Disable I/O Unit Causing Current Error
- Enable I/O Unit Causing Current Error
- Error on I/O Unit?
- Error?
- Get Error Code of Current Error
- Get Error Count
- Get ID of Block Causing Current Error
- Get Line Causing Current Error
- Get Name of Chart Causing Current Error
- Get Name of I/O Unit Causing Current Error
- Get Severity of Current Error
- Remove Current Error and Point to Next Error
- Stop Chart on Error
- Suspend Chart on 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 communicatin 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 85. For more on the message queue, see "Queue Messages" on page 367.

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.

I/O Unit Commands

The following commands are used to communicate with an I/O unit, which controls a group of I/O points:

- Get I/O Unit as Binary Value
- Get Target Address State[1]
- I/O Unit Ready?
- IVAL Move Numeric Table to I/O Unit
- IVAL Move Numeric Table to I/O Unit Ex
- Move I/O Unit to Numeric Table
- Move I/O Unit to Numeric Table Ex
- Move Numeric Table to I/O Unit
- Move Numeric Table to I/O Unit Ex
- Set All Target Address States[1]
- Set I/O Unit Configured Flag
- Set Target Address State[1]
- 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 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 152**).*

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. 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 124. 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 "Miscellaneous Commands" on page 295, "Logical Commands" on page 290, "Mathematical Commands" on page 291, "Pointer Commands" on page 299, and "String Commands" on page 301.

I/O Unit—Event Message Commands

The following commands refer to event messages sent from SNAP PAC I/O units.

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 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 Memory Map
Read Numeric Table from I/O Memory Map
Read String from I/O Memory Map
Read String Table from I/O Memory Map
Write Number to I/O Memory Map
Write Numeric Table to I/O Memory Map
Write String Table to I/O Memory Map
Write String to I/O 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. 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 288). If you are changing event messages, use the I/O Unit—Event Message commands instead (page 286).

Before you use these commands with a SNAP-PAC-S1, you must have already configured an I/O unit to represent the controller. Configure the controller as a Generic Memory Map Device with the IP address of the control engine.

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 the *OptoMMP Protocol Guide* (Opto 22 form 1465) 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 PAC 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

Since these are I/O unit commands, when they are used for a SNAP-PAC-S1, you must have already configured an I/O unit to represent the controller. Configure the controller as a Generic Memory Map Device with the IP address of the control engine.

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 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 261), thus speeding up peer-to-peer communication.

The memory map Scratch Pad area supports four data types: bits, integer 32s, floats, and strings.

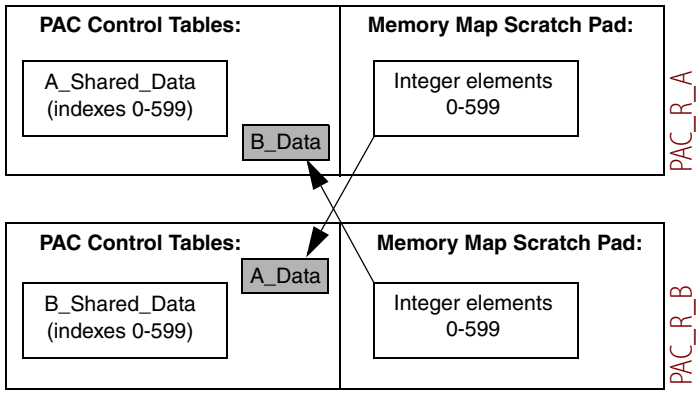
- For details on the Scratch Pad area, see Opto 22 form 1704, the *PAC Manager User's Guide*.
- For the complete memory map, see form 1465, the *OptoMMP Protocol Guide*.
- For details on using the Scratch Pad for peer-to-peer communication with a controller, see the controller's user's guide.

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

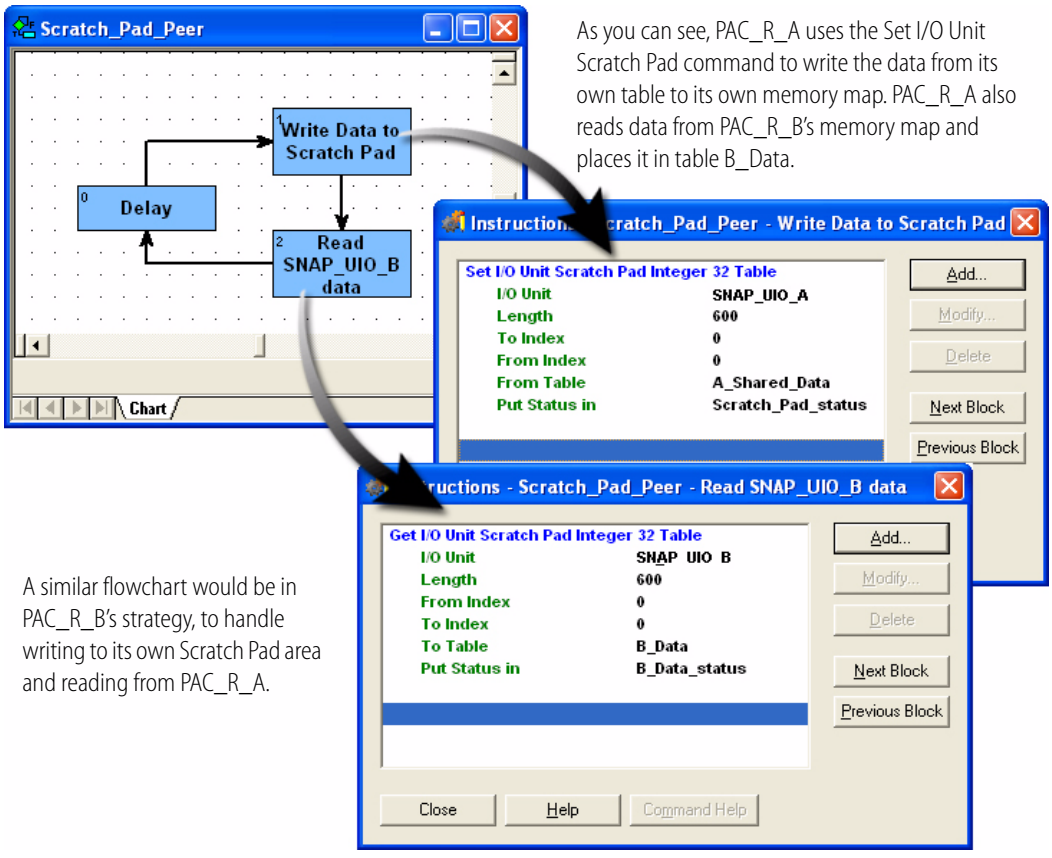
PAC_R_A writes data from its own PAC Control strategy table to its own Scratch Pad area, which SPAC_R_B can then read.



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.

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:

AND
AND?
Bit AND
Bit AND?
Bit Clear
Bit NOT
Bit NOT?
Bit Off in Numeric Table Element?
Bit Off?
Bit On in Numeric Table Element?
Bit On?
Bit OR
Bit OR?
Bit Rotate
Bit Set
Bit Shift
Bit Test
Bit XOR
Bit XOR?
Equal to Numeric Table Element?
Equal?
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?
Less Than Numeric Table Element?
Less Than or Equal to Numeric Table Element?
Less Than or Equal?
Less?
Make Integer 64
Move 32 bits
NOT
Not Equal to Numeric Table Element?
Not Equal?
NOT?
Numeric Table Element Bit Clear
Numeric Table Element Bit Set
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 332 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 291 for further information on integers and floats.

Mathematical Commands

The following commands perform mathematical functions:

Absolute Value
Add
Clamp Float Table Element
Clamp Float Variable
Clamp Integer 32 Table Element
Clamp Integer 32 Variable
Complement
Decrement Variable
Divide
Generate Random Number
Increment Variable
Maximum
Minimum
Modulo
Multiply
Natural Log
Raise e to Power
Raise to Power
Round
Seed Random Number
Square Root
Subtract
Truncate

Trigonometry

Arcsine
Arccosine
Arctangent
Cosine
Hyperbolic Cosine
Hyperbolic Sine
Hyperbolic TangentSine
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: $(-1)^{Sign} \times [1 + Significand/2^{23}] \times 2^{(Exponent-127)}$

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 7^{th} digit). Also, converting $5.554444e^{+8}$ back to an integer yields 555444352 (note the error starting in the 7^{th} 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 only keep track 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)
Comment (Single Line)
Flag Lock
Flag Unlock
Float Valid?
Generate Reverse CRC-16 on Table (32 bit)
Get Length of Table
Get Type From Name
Get Value From Name
Move
Move from Numeric Table Element
Move Numeric Table Element to Numeric Table
Move Numeric Table to Numeric Table
Move to Numeric Table Element
Move to Numeric Table ElementsShift 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 9 in "Using the OptoScript Editor" on page 337.

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 I/O units. 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 142. For more information about PID loops, 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 IntegralSet 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. Because PIDs run on the I/O side of the SNAP PAC R-series controller, not on the control side, these PIDs will keep running on the SNAP PAC R-series 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 (PID)" on page 297.
- 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, 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 (PID)

When you configure a PID loop in PAC Control, choose one of the following algorithms:

- Velocity
- ISA
- Parallel
- Interacting

The ISA, Parallel and Interacting algorithms are functionally equivalent; the only difference is the way the tuning constants are factored. The identical and differing equations for all algorithms are shown in the following sections.

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.
PvIn_1, PvIn_2	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
```

Velocity Algorithm

```
TermP = ( Err - Err_1 )
TermI = TuneI * ScanTime * Err
TermD = TuneD / ScanTime * ( PvIn - 2 * PvIn_1 + PvIn_2 )
ΔOutput = Span * Gain * ( ΔTermP + ΔTermI + ΔTermD )
```

Non-velocity Algorithms

These equations were 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.

Equations common to all but the velocity algorithm:

```
Integral = Integral + Err
TermP = Err
TermI = TuneI * ScanTime * Integral
TermD = (TuneD / ScanTime) * ( PvIn - PvIn_1 )
```

"Ideal" or ISA Algorithm

```
Output = Span * Gain * ( TermP + TermI + TermD )
```

"Parallel" Algorithm:

```
Output = Span * ( Gain * TermP + TermI + TermD )
```

"Interacting" Algorithm:

```
Output = Span * Gain * ( TermP + TermI ) * ( 1 + TermD )
```

Pointer Commands

The following commands are used with pointers:

Clear Pointer	Move to Pointer
Clear Pointer Table Element	Move to Pointer Table
Get Pointer From Name	Pointer Equal to NULL?
Move from Pointer Table Element	Pointer Table Element Equal to NULL?

See also:

"Understanding Pointers"

"Advantages of Using Pointers"

"Referencing Objects with Pointers"

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-R2 SNAP-PAC-EB1 SNAP-PAC-EB2 SNAP-PAC-SB1 SNAP-PAC-SB2	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 93.

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?
Communication to All I/O Units Enabled?
Disable Communication to All I/O Points
Disable Communication to All I/O Units
Disable Communication to I/O Unit
Disable Communication to PID Loop
Disable Communication to Point
Enable Communication to All I/O Points
Enable Communication to All I/O Units
Enable Communication to I/O Unit
Enable Communication to Mistic PID Loop[1]
Enable Communication to PID Loop
Enable Communication to Point
I/O Point Communication Enabled?
I/O Unit Communication Enabled?
IVAL Set Analog Point
IVAL Set Counter
IVAL Set I/O Unit from MOMO Masks
IVAL Set Off-Latch
IVAL Set Off-Pulse[1]
IVAL Set On-Latch
IVAL Set On-Pulse[1]
IVAL Set TPO Percent[1]
IVAL Set TPO Period[1]
IVAL Turn Off
IVAL Turn On
PID Loop Communication Enabled?

[1] PAC Control Professional 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 *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
Generate Forward CCITT on String
Generate Forward CRC-16 on String
Generate Reverse CCITT 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
Set Nth Character
String Equal to String Table Element?
String Equal?
Test Equal Strings
Verify Checksum on String
Verify Forward CCITT on String
Verify Forward CRC-16 on String

Find Substring in String	Verify Reverse CCITT on String
Generate Checksum on String	Verify Reverse CRC-16 on String

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 Chapter 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

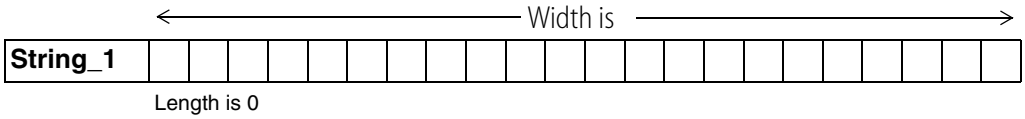
You can input most control characters in a string by typing a backslash (\ ) followed by the two-character hex value of the character. For example, to add an ACK (CTRL+F) character, enter \06 as part of the string.

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.

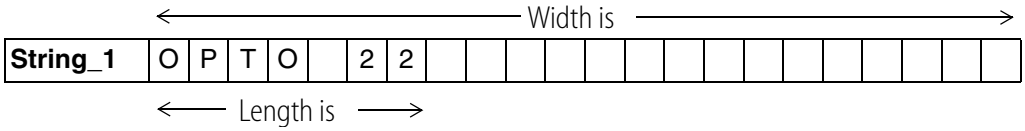
To input a single backslash in a string, type in a double backslash (\\ ).

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

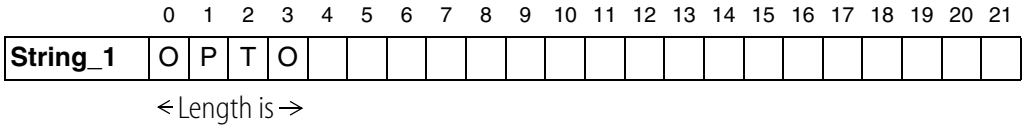
Strings are assembled using commands Move String, Append Character to String, and Append String to String. Consider the following original string and the examples that follow:

	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*

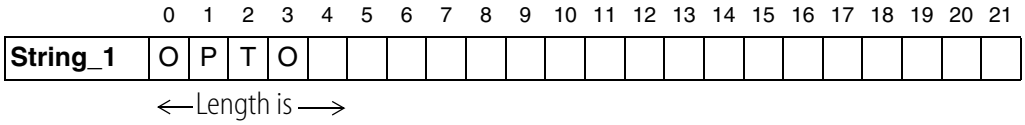
Results in (note that Move String erased the previous contents of the string):



Append Character to String

From	32	*integer literal (represents a space)*
To	String_1	*string variable*

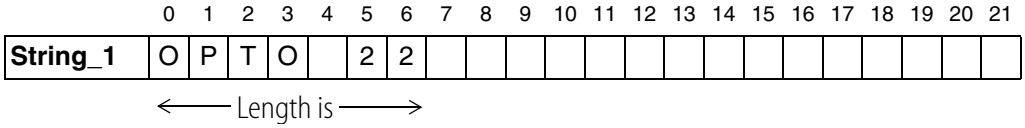
Results in (note the space character in position 4):



Append String to String

From	22	*string literal*
To	String_1	*string variable*

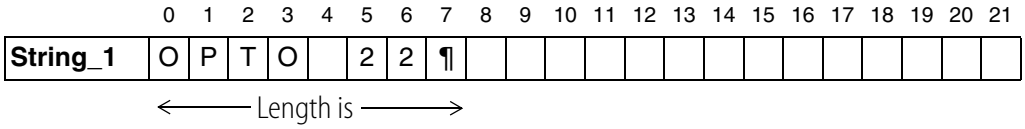
Results in:



Append Character to String

From	13	*integer literal (carriage return)*
To	String_1	*string variable*

Results in:



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 Appendix 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 a 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

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	<

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	\

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	|

Dec	Hex	CC	Char
29	1D	^]	GS
30	1E	^^	RS
31	1F	^_	US

Dec	Hex	Char
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
125	7D	}
126	7E	~
127	7F	DEL

Time/Date Commands

The following commands refer to time, dates, and days:

Dates and Days	**Time**
Copy Date to String (DD/MM/YYYY)	Copy Time to String
Copy Date to String (MM/DD/YYYY)	Get Hours
Get Day	Get Minutes
Get Day of Week	Get Seconds
Get Julian Day	Get Seconds Since Midnight
Get Month	Get System Time
Get Year	Set Hours
Set Date	Set Minutes
Set Day	Set Seconds
Set Month	Set Time
Set Year	

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.

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?
Pause Timer	
Set Down Timer Preset Value	**Delays**
Set Up Timer Target Value	Delay (mSec)
Start Timer	Delay (Sec)

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 "Increasing Efficiencies in Your Strategy" on page 96.

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.

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 in this chapter, see the Control Basic Examples directory on your hard drive for a fully annotated example strategy. OptoScript information is also provided in 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 386.

In this Chapter

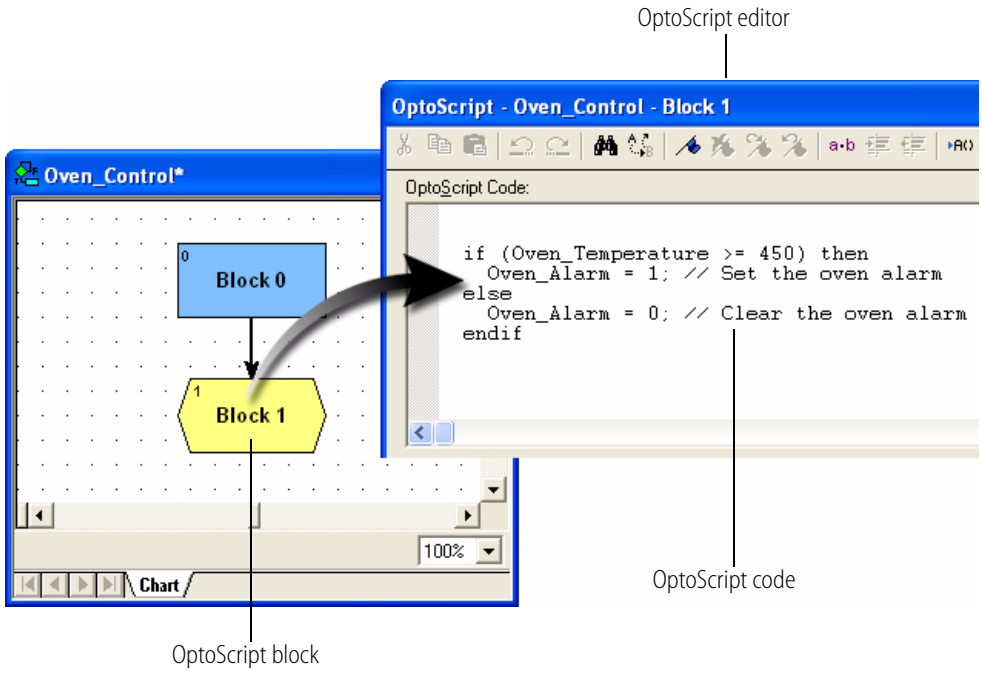
About OptoScript 314
When To Use OptoScript 314
OptoScript Functions and Commands 322
OptoScript Syntax 323
OptoScript Data Types and Variables 325
OptoScript Expressions and Operators 330
OptoScript Control Structures 333
Using the OptoScript Editor 337
Troubleshooting "Unable To Find" Errors 340
Troubleshooting Syntax Errors 340
Debugging Strategies with OptoScript 342

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:



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
- for string handling
- for complex loops
- for case statements
- for conditions
- for combining math expressions, loops, and conditions.

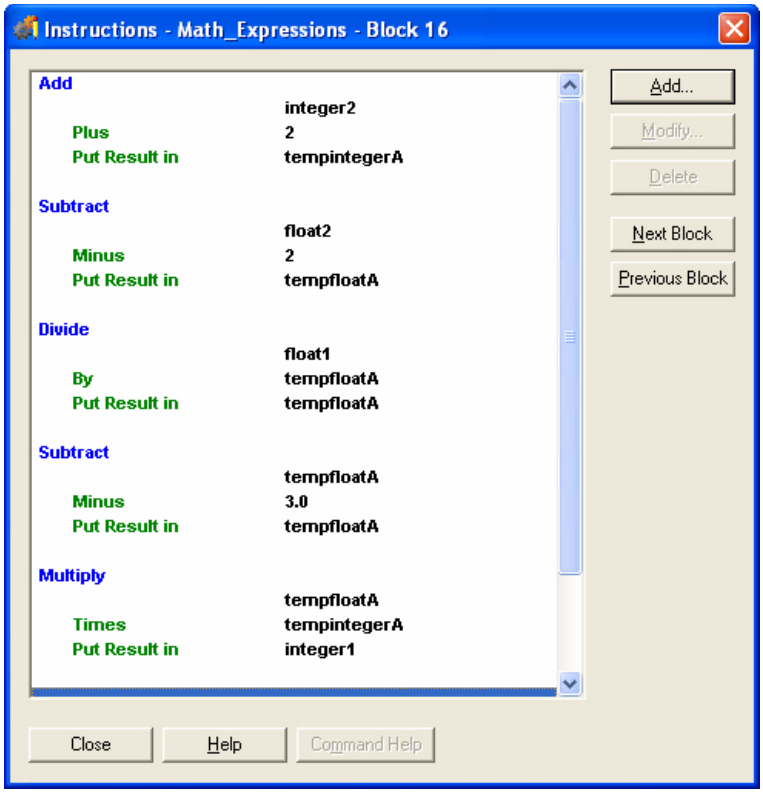
For Math Expressions

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.

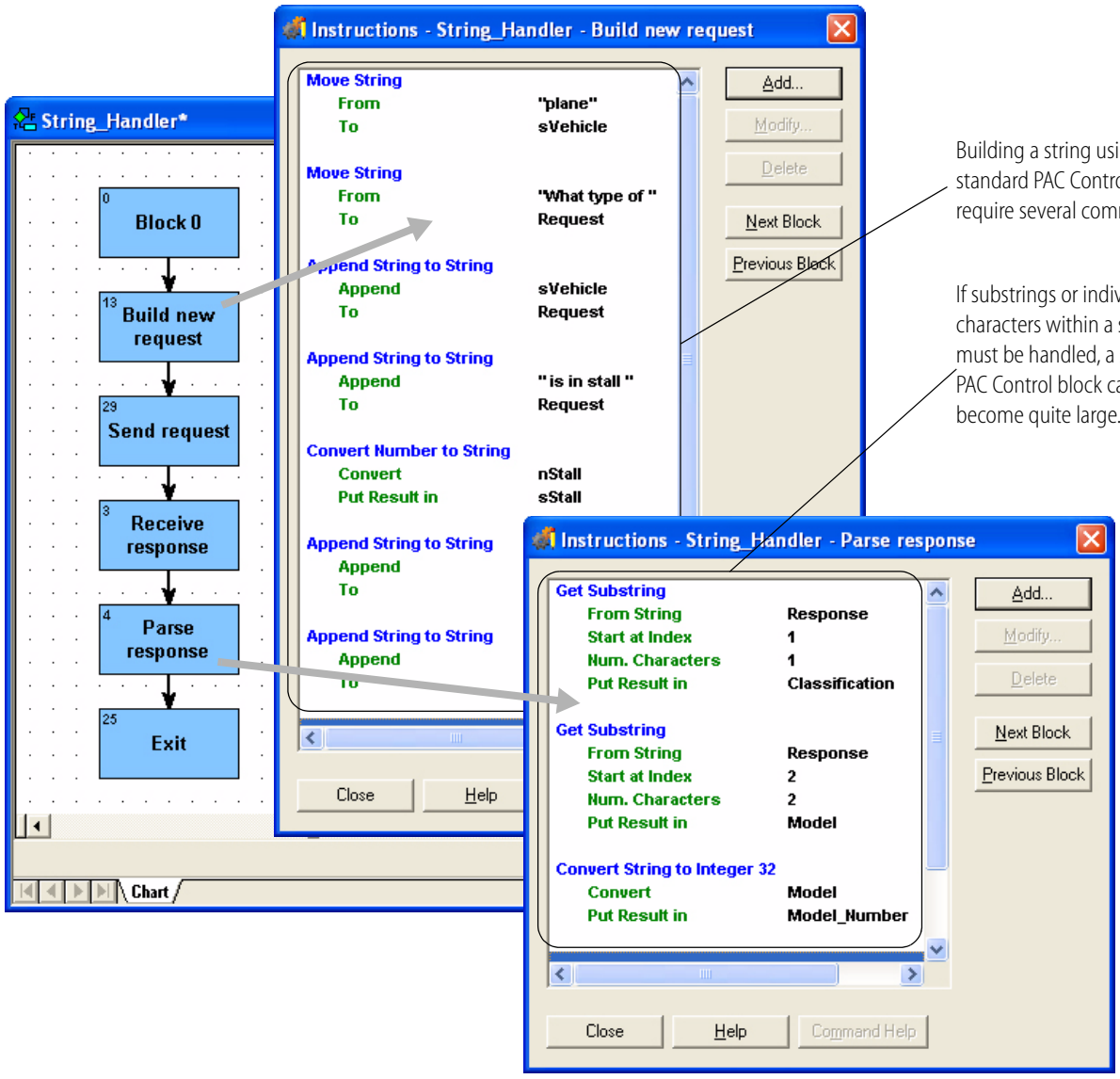


As you can see, the OptoScript version of this math expression is not only simpler to create, but also easier to understand once created.

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.

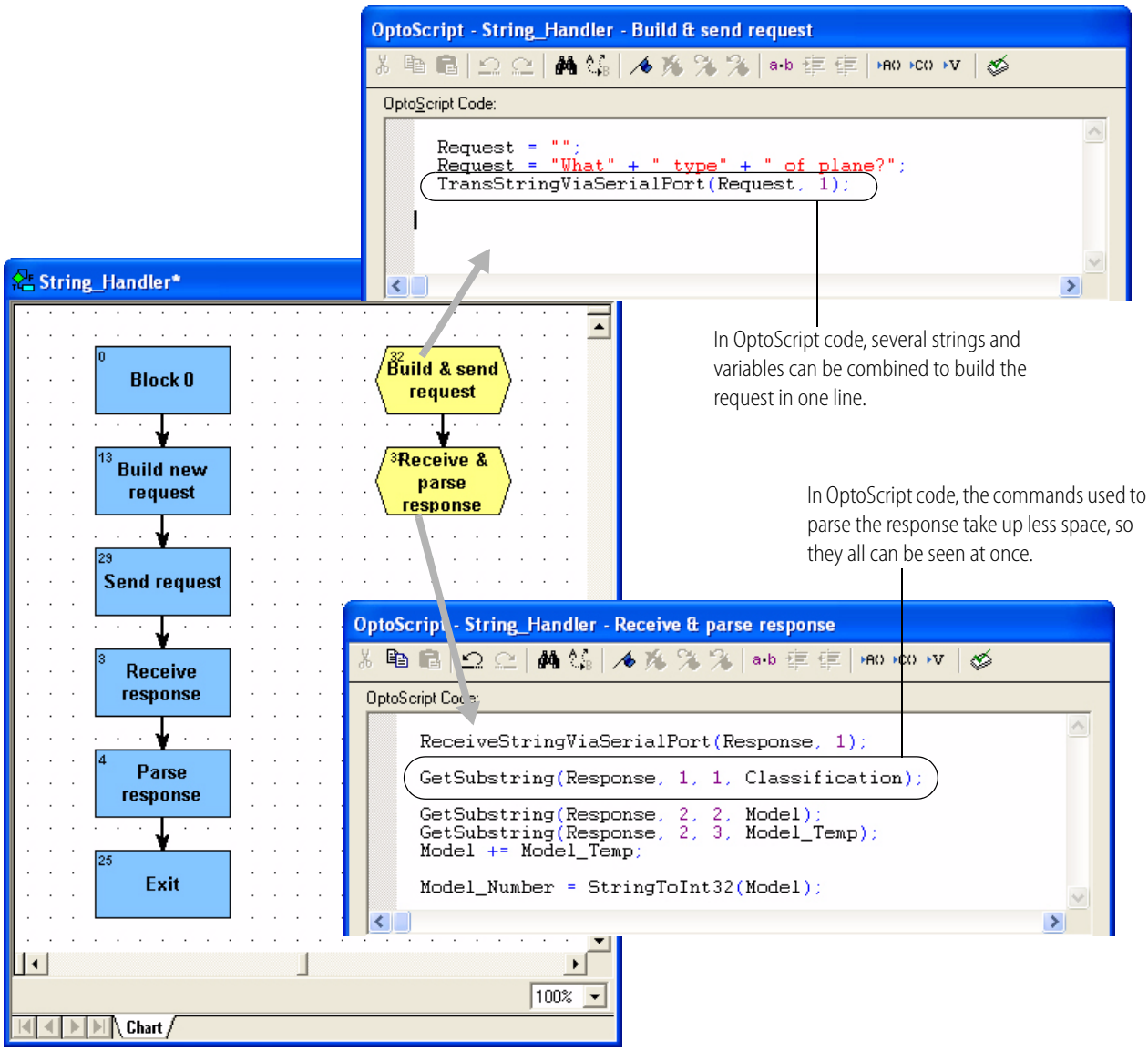
The following figure shows a flowchart designed to send the string request, "What type of plane?" and parse the response, "F14," into a classification (F) and a model number (14). Compare these blocks and instructions with the ones on the following page, done in OptoScript.



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.

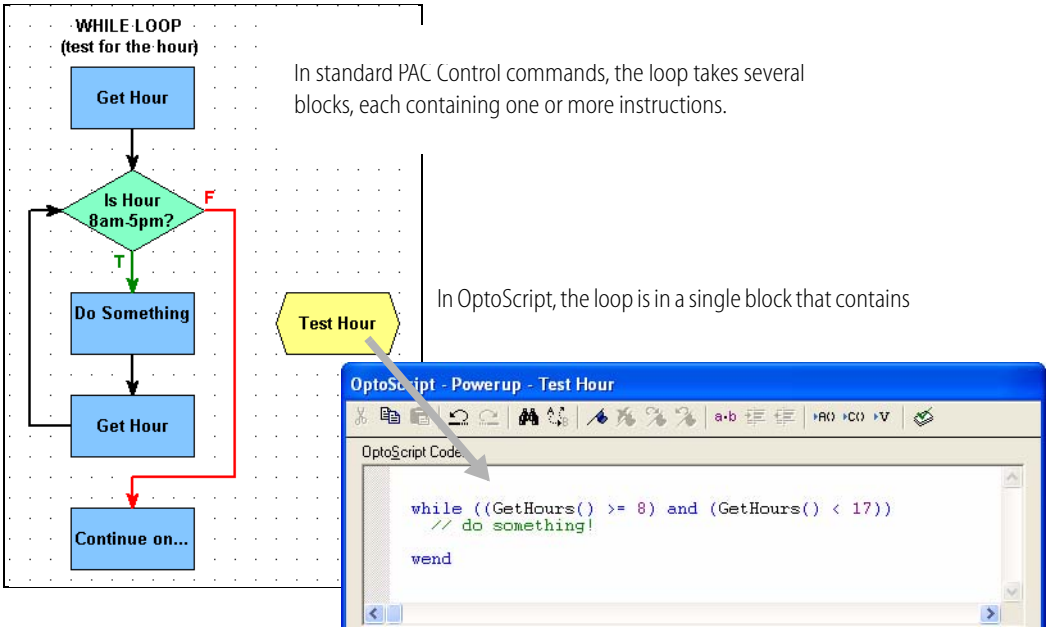


For Complex Loops

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.

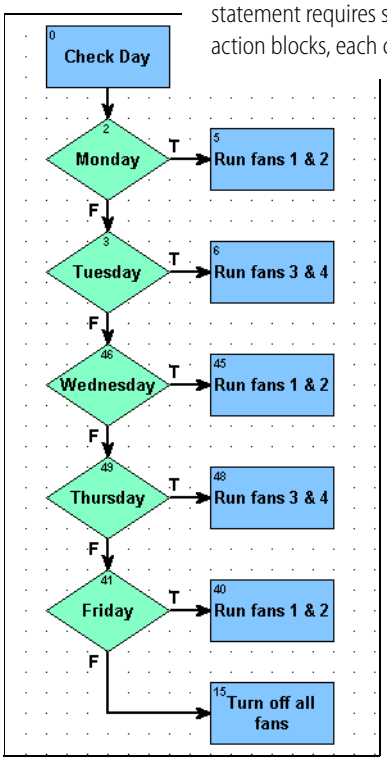
Below is an example of a *while* loop as it would appear in standard flowchart commands, contrasted with the way it could be handled in an OptoScript block.



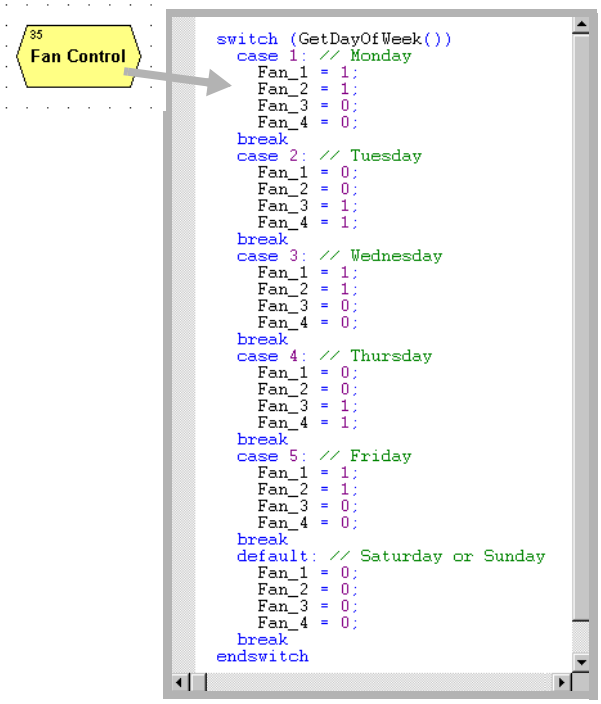
For Case Statements

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.



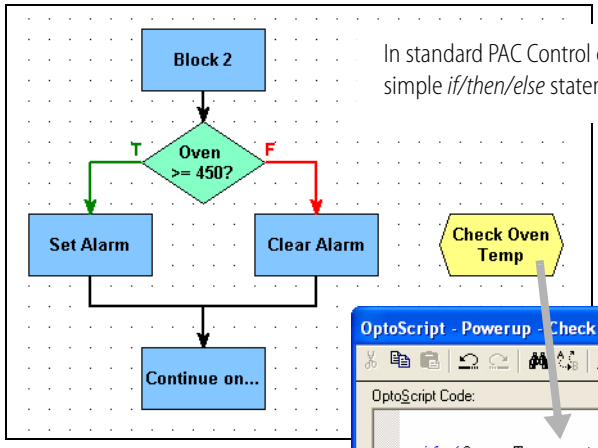
In OptoScript, the code is all in one block.



Using OptoScript for case statements saves space in the flowchart and lets you see all the possible cases in one dialog box.

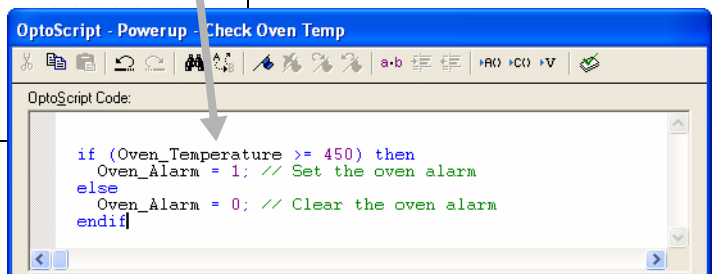
For Conditions

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:

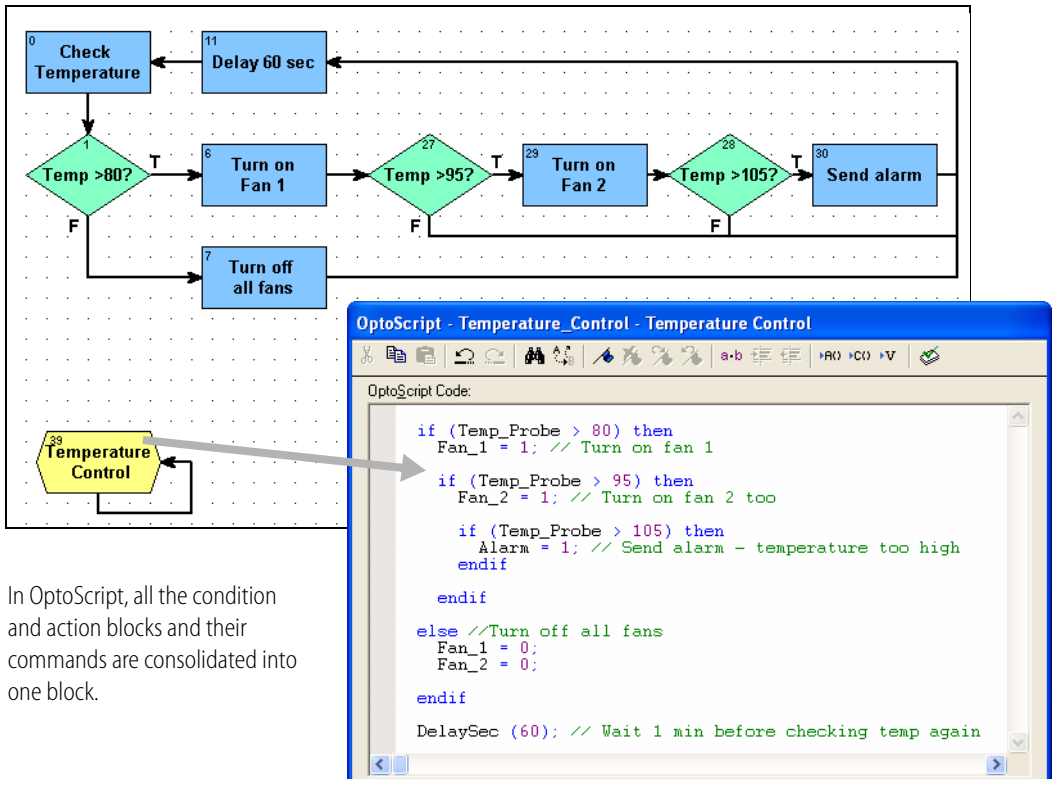


In standard PAC Control commands, even a simple *if/then/else* statement requires three

In OptoScript, a single block contains the statement.



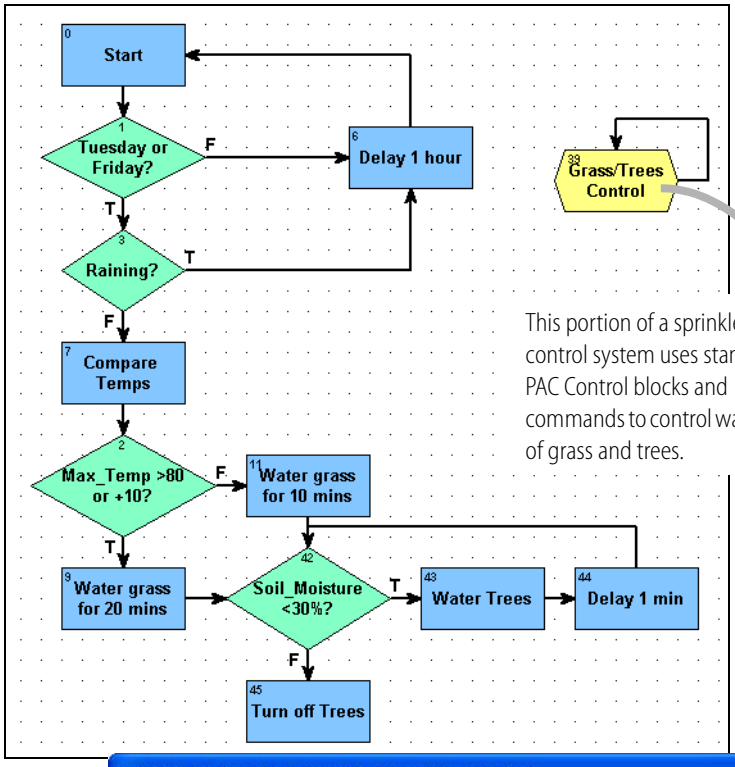
OptoScript is even more useful for more complex conditions, such as the following:



In OptoScript, all the condition and action blocks and their commands are consolidated into one block.

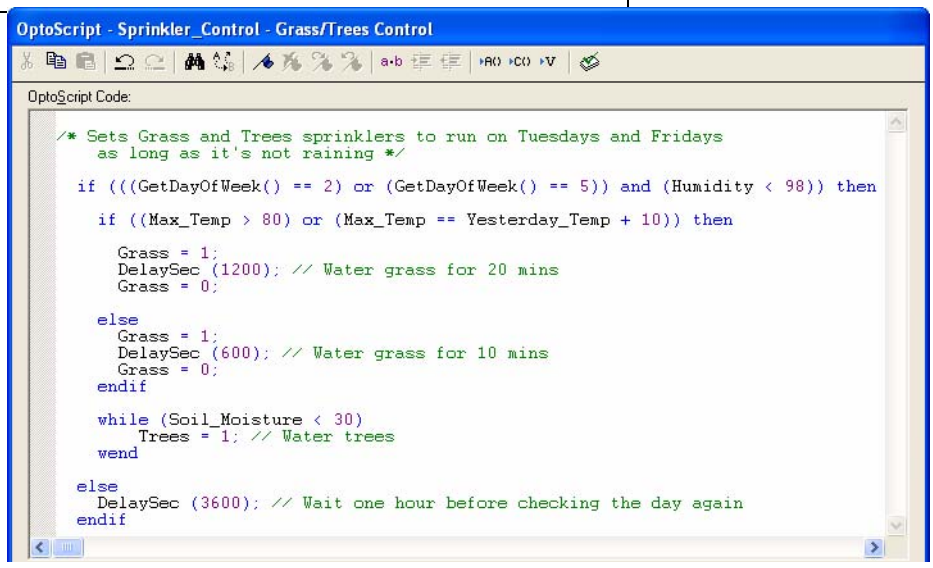
For Combining Expressions, Operators, and Conditions

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.



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:

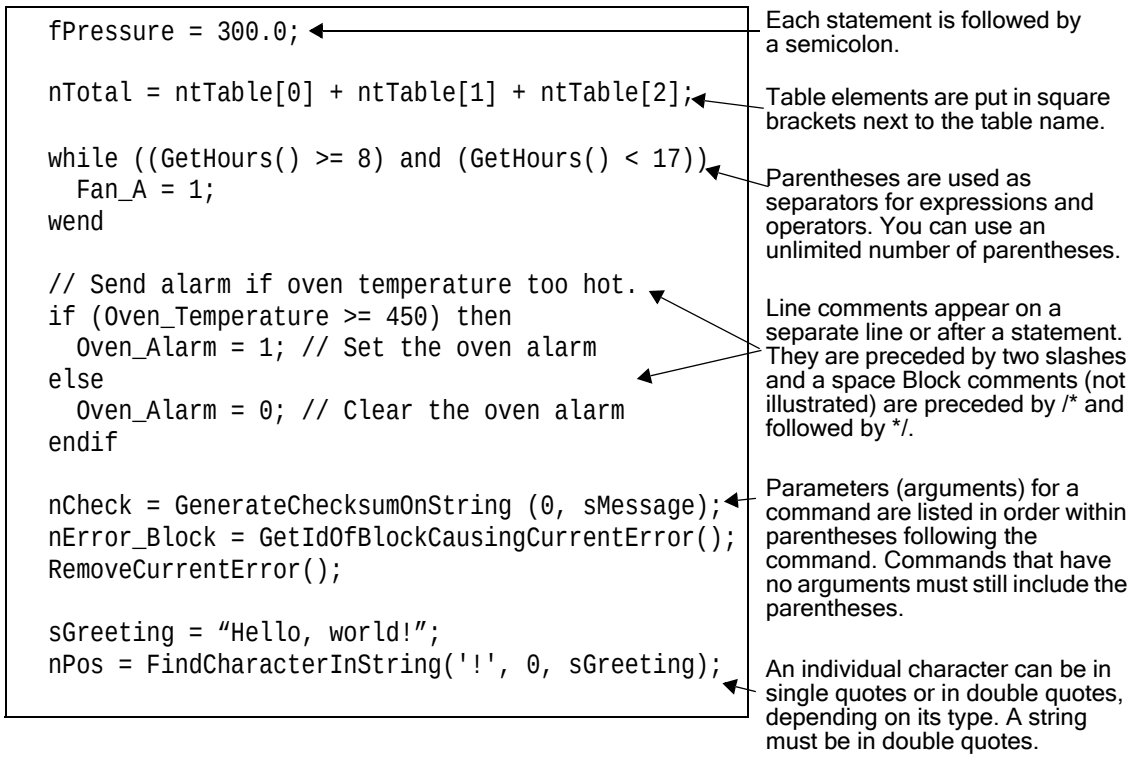
```
LED01 = Switch_A;
Proportional_Valve = fPressure_Control
```

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. Indentation is not required, but is used for clarity.



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, as shown below:

```
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 Chapter 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 example, a variable named *Month* might be either a string or an integer.

An easy way to avoid this confusion is to use Hungarian notation—that is, to place letters indicating variable type at the beginning of the name. For example, *sMonth* would indicate a string; *nMonth* would indicate an integer. 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

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 Float assignments:

```
f1 = 3.0;
f2 = f1
```

Simple Integer 64 assignments:

```
nn1 = 2i64;
nn2 = nn1;
```

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

Pointers can be tricky, but they are powerful tools. For more information on using pointers, see “Pointer Commands” on page 299.

For the following examples, assume that:

```
n1 = 5;
f1 = 9.2;
s1 = "test 123";
```

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;
```

To see if a pointer is pointing to something, use the comparison operator == (see page 331) 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
```

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;
```

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;
```

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 334) 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;
```

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 page 389 for 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:

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 only takes a value 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 389.

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 334
- "While Loops" on page 335
- "Repeat Loops" on page 336
- "For Loops" on page 336

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:

```
if (n1) then
  f1 = 2.0;
endif
```

Complex logical operations can also be used:

```
if ((n1 > 3) and (not n1 == 6)) then
  f1 = 2.0;
  f2 = 6.5;
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
```

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
```

An optional `else` statement can be added:

```
if (n1 > 3) then
  f1 = 2.0;
  f2 = 6.5;
else
  f3 = 8.8;
endif
```

If statements can be nested. Each `if` requires an `endif`:

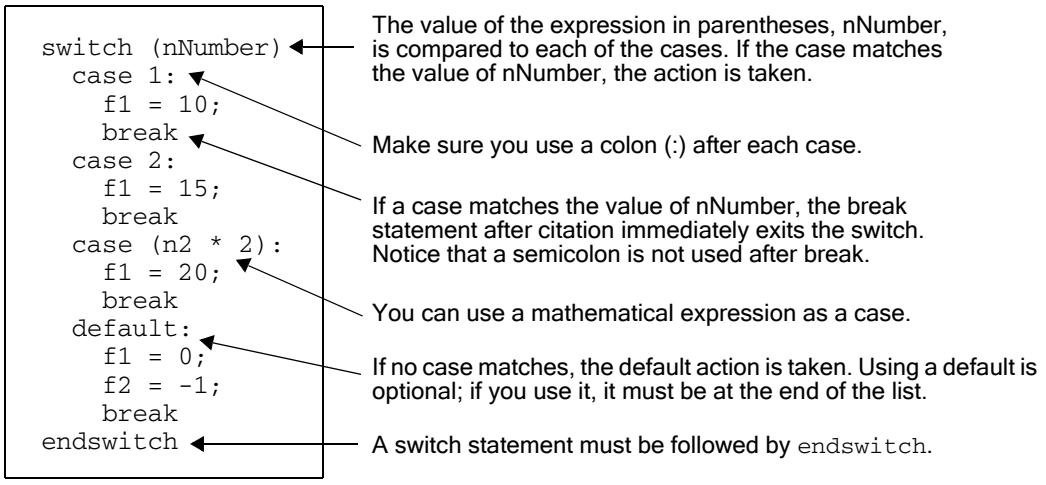
```
if (n1 > 3) then
  f1 = 2.0;
  f2 = 6.5;

  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.

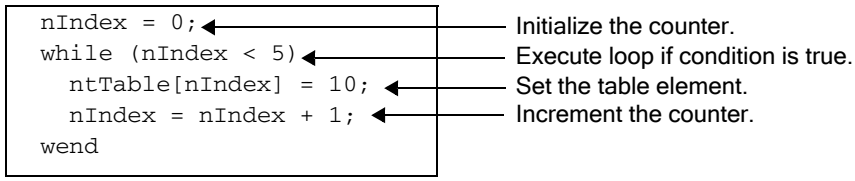
Here's an example of a switch statement.



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:



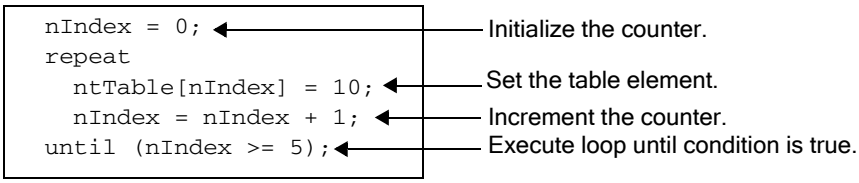
While loops can be nested and can contain other kinds of program statements. Each `while` needs a matching `wend` at the end. For example:

```
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.



Repeat loops can be nested and can contain other kinds of program statements. Each `repeat` statement needs a matching `until` statement at the end.

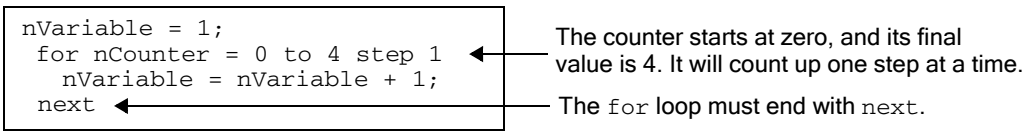
For Loops

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:



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:

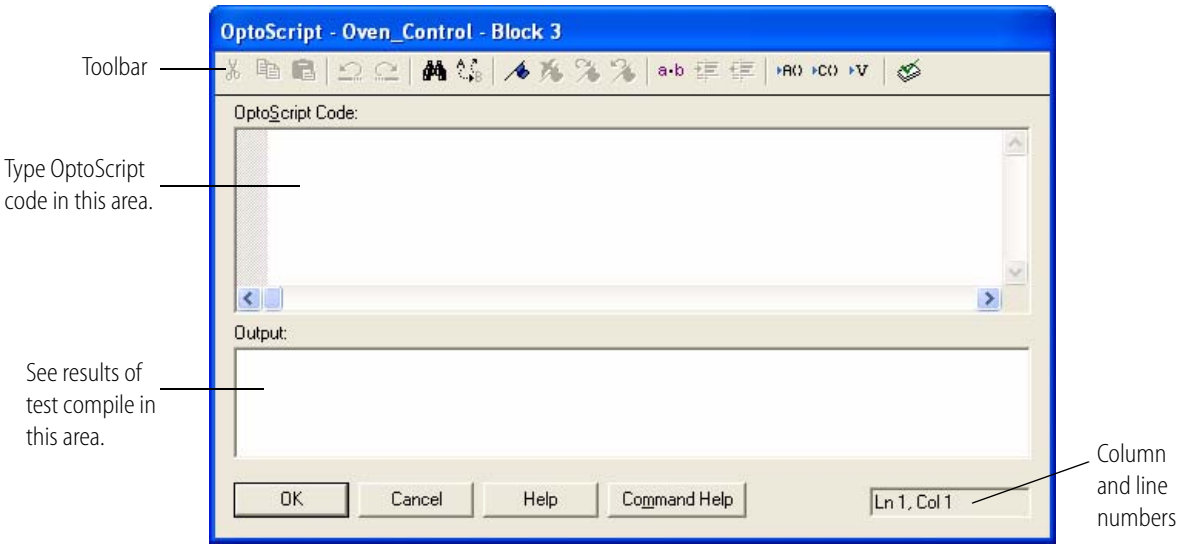
```
nSide = 5;
for nLength = 0 to (nSide * 3) step 1
  nSide = 1;
next
```

For loops can be nested and can contain other types of statements. Each `for` requires a `next` at the end.

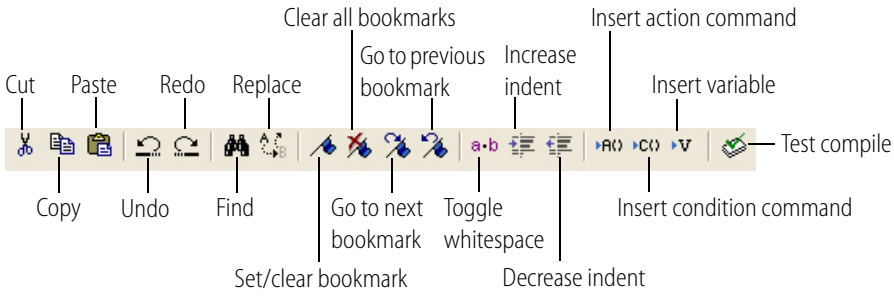
Using the OptoScript Editor

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 Chapter 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.



The toolbar includes the following buttons:



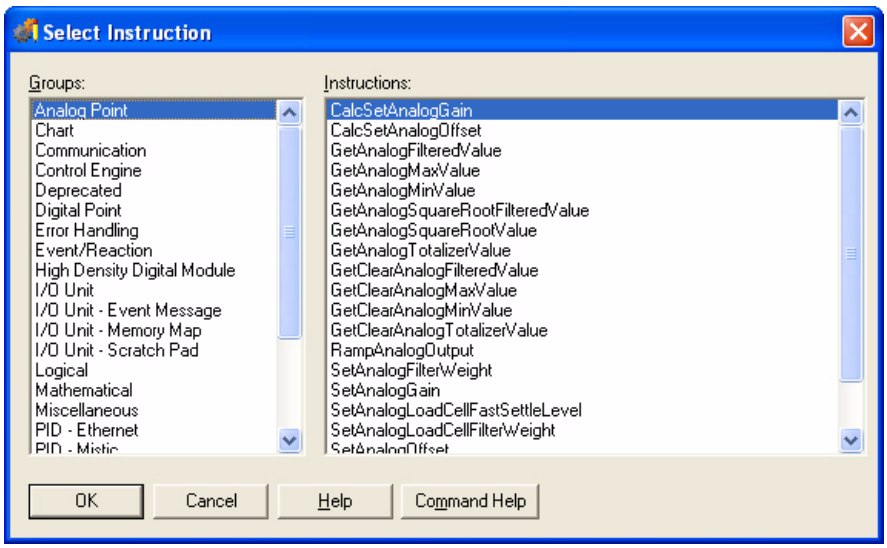
2. Begin typing OptoScript code in the top area.

   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 Action Command or Insert Condition Command button in the toolbar.



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.

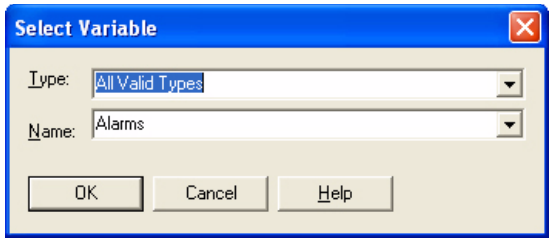
For information on any command, highlight it and click Command Help, or just double-click the command name.

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 Appendix E, "OptoScript Command Equivalents," for more information.

5. Click OK.

   The command appears in the OptoScript code.

6. 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 8. If you're not sure of the item's name, click the Insert Variable button in the toolbar.



7. From the Type drop-down list in the Select Variable dialog box, choose the type of item you want to use. From the Name drop-down list, choose the item. Click OK.

   The item appears in the code.

8. 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.

9. 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:

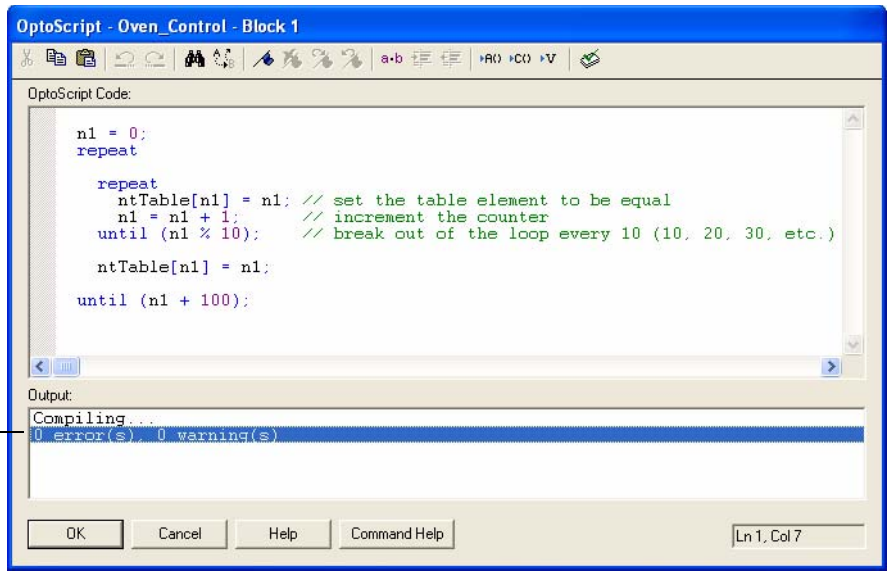
```
/* This is a block comment that goes
        beyond one line. */
```

10. 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.

11. 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:



Results after code is compiled

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.

12. 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

See also, "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 Command Equivalents," 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.

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 330.

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 333 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 325 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 189.

Using Subroutines

Introduction

This chapter shows you how to create and use subroutines.

In this Chapter

About Subroutines.....343
Creating Subroutines.....345
Using Subroutines.....351
Viewing Subroutines.....355
Printing Subroutines.....355

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. (You cannot call a subroutine from another subroutine, however.)

Like charts, subroutines start at one block and proceed 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 the same time.

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). Other types can be passed into the subroutine through literals, however; see below for more information.
- Numeric tables
- String variables
- String literals
- String tables
- Communication handles

In PAC Control Professional, 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 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. If you are familiar with other programming languages, literals are similar to "passed by value" parameters, while variables are like "passed by reference" parameters.

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.

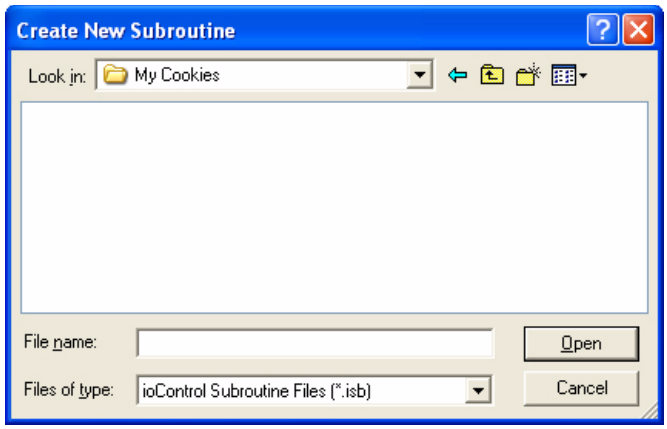
A Put Status In parameter appears automatically in every subroutine. This parameter is used to let you know whether the subroutine was called successfully, in the same way that function commands return a status. Make sure that you always check the status code after calling a subroutine. Subroutine status codes are:

0	Success
-67	Out of memory
-69	Null object error. Make sure you are not passing a pointer that points to null.
-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.



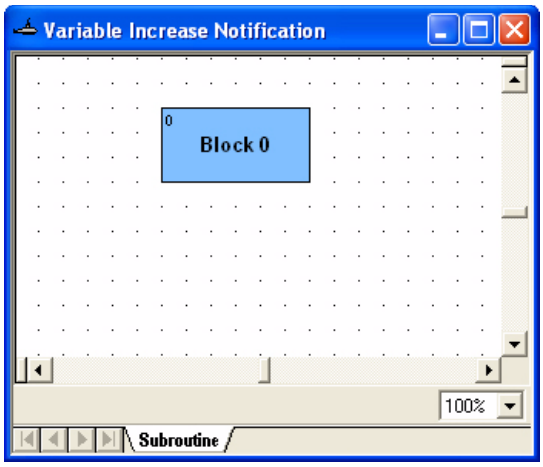
2. Enter a subroutine name.

   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").

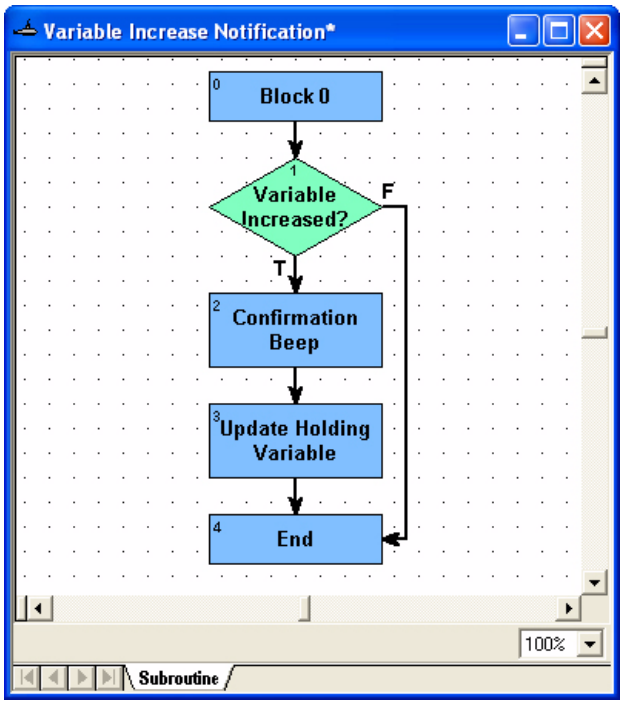
3. Navigate to the directory where you want to store the subroutine and click Open.

   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 224.

5. Save the subroutine by selecting Subroutine→Save.

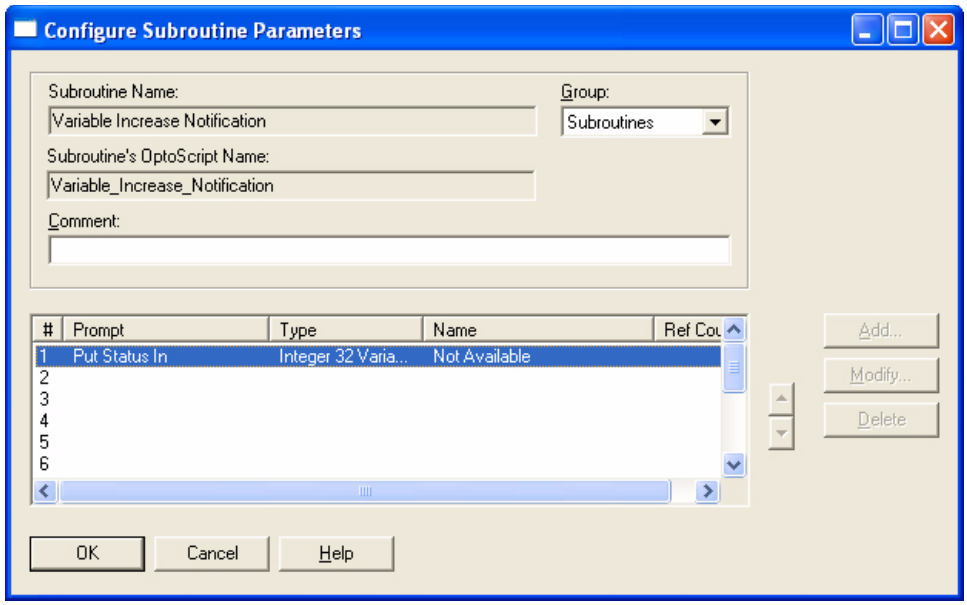
Configuring Subroutine Parameters

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.



   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; 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.

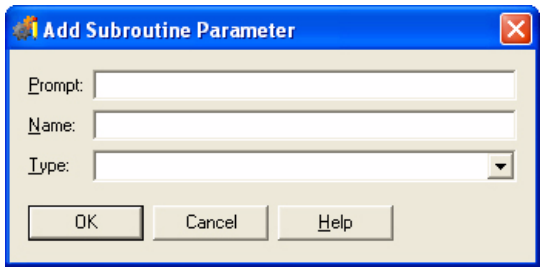
IMPORTANT: *Make sure you always check returned status codes for all subroutines. Subroutine status codes and their meanings are:*

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 (future use)

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 349 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.



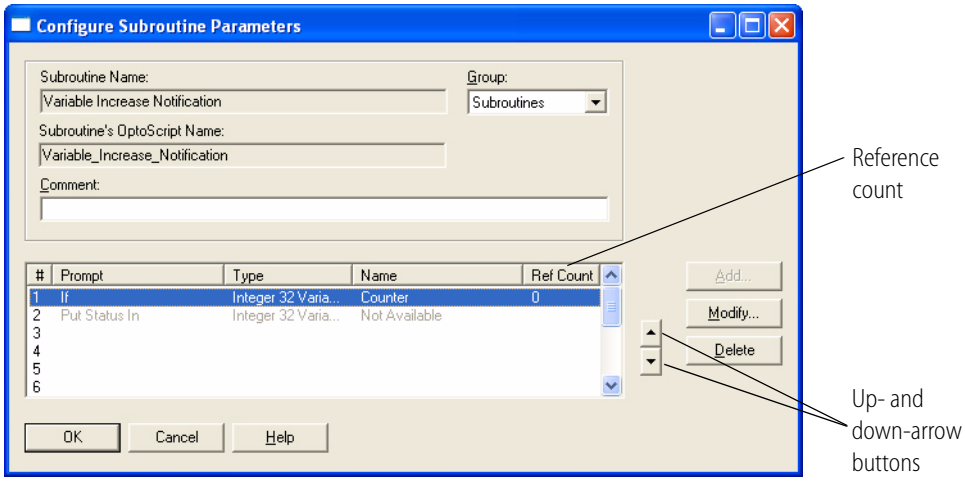
b. In the Prompt field, enter the prompt text you want to show in the Add Instruction dialog box in the strategy. This field is limited to 16 characters.

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.



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.

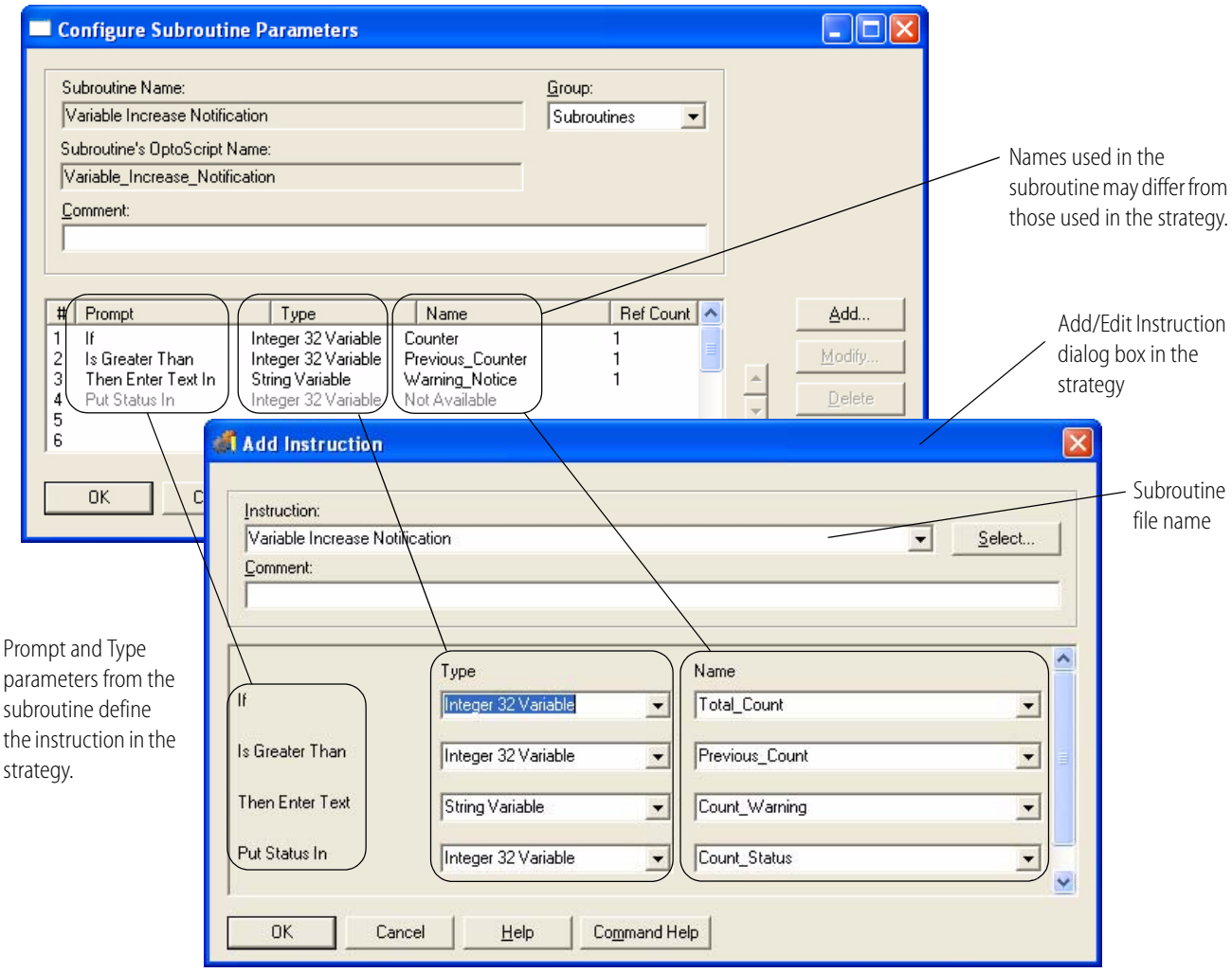
NOTE: You cannot delete the Put Status In parameter or change its order in the list, and you cannot delete a parameter that has a reference count greater than zero (indicating that it is used in the subroutine). Also, if you add or delete parameters after including a subroutine in a strategy, you may 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:



Adding Commands and Local Variables

Adding commands (instructions) to subroutines is exactly like adding instructions to charts. For help, see "Adding Commands" on page 249. 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. Adding variables to subroutines is also like adding variables to charts. For help, see "Adding Variables" on page 235.

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

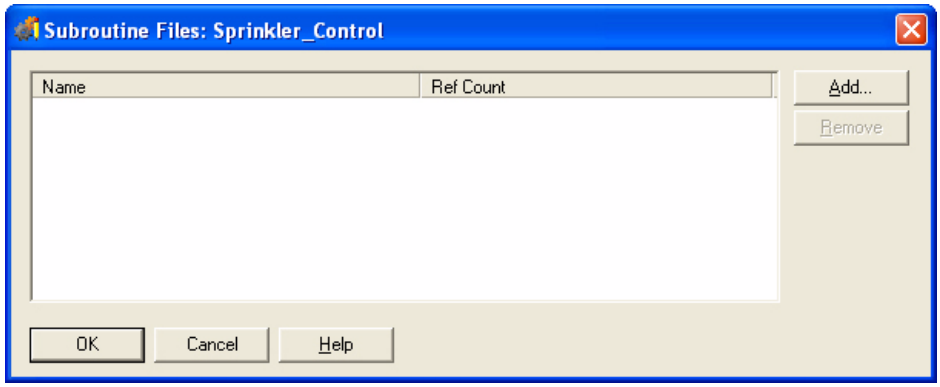
To use a subroutine in a strategy, you include it in the strategy and then add it as a command (instruction) so it can be called from a chart.

Including a Subroutine in a Strategy

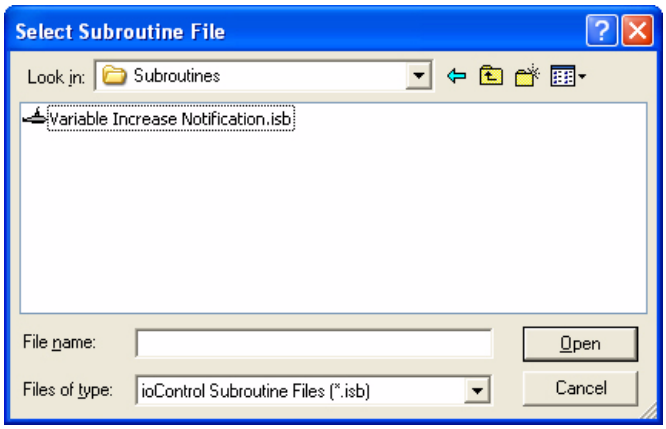
Since subroutines are independent of strategies, you must include the subroutine in the strategy before you can use it.

1. With the strategy open in Configure mode, double-click the Subroutines Included folder on the Strategy Tree (or click the Include Subroutines button on the toolbar, or select Configure→Subroutine Included).

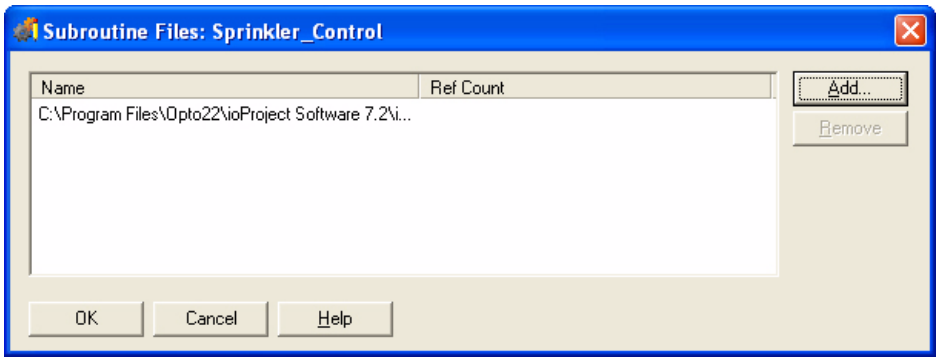
   The Subroutine Files dialog box appears, listing all subroutines currently included in the strategy. The example below shows no subroutines currently included.



2. Click Add.



3. Navigate to the directory containing the subroutine you want to add and double-click the subroutine.



4. When the full path to the subroutine appears in the Subroutine Files dialog box, click OK.

   The new subroutine appears in the Strategy Tree in the Subroutines Included folder.

Adding a Subroutine Instruction

You use a subroutine just like a PAC Control command: by adding the subroutine instruction to a block in the chart.

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 347.

1. With the strategy open in Configure mode, open the chart that will use the subroutine.
2. Double-click the block that will call the subroutine.

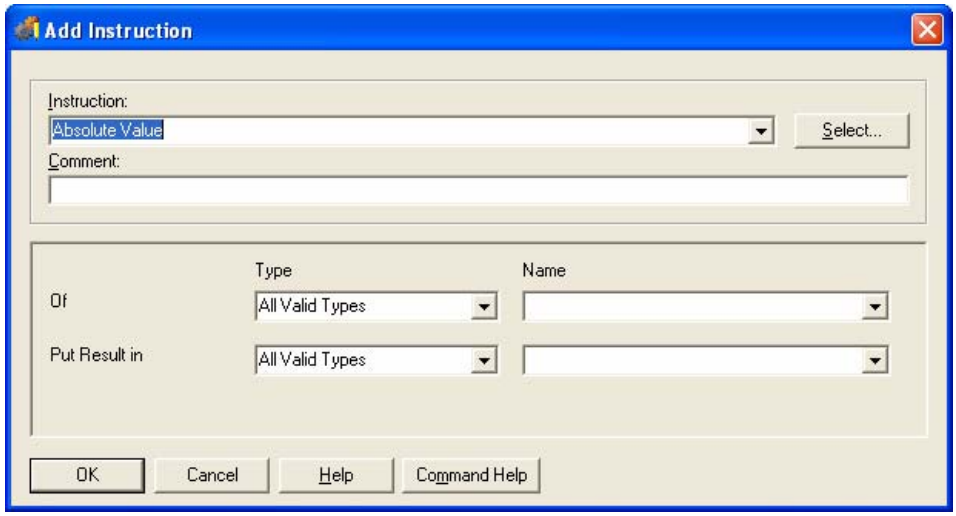
   If it is an OptoScript block, list parameters (arguments) in order within parentheses, the same way you enter parameters for PAC Control commands (see "OptoScript Syntax" on page 323). Be sure to use the subroutine's OptoScript name, using underscores instead of spaces. The Put Status In return value can be consumed by a variable (as shown below) such as 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 337 .

3. In the Instructions dialog box, click where you want the instruction to be placed, and then click Add.

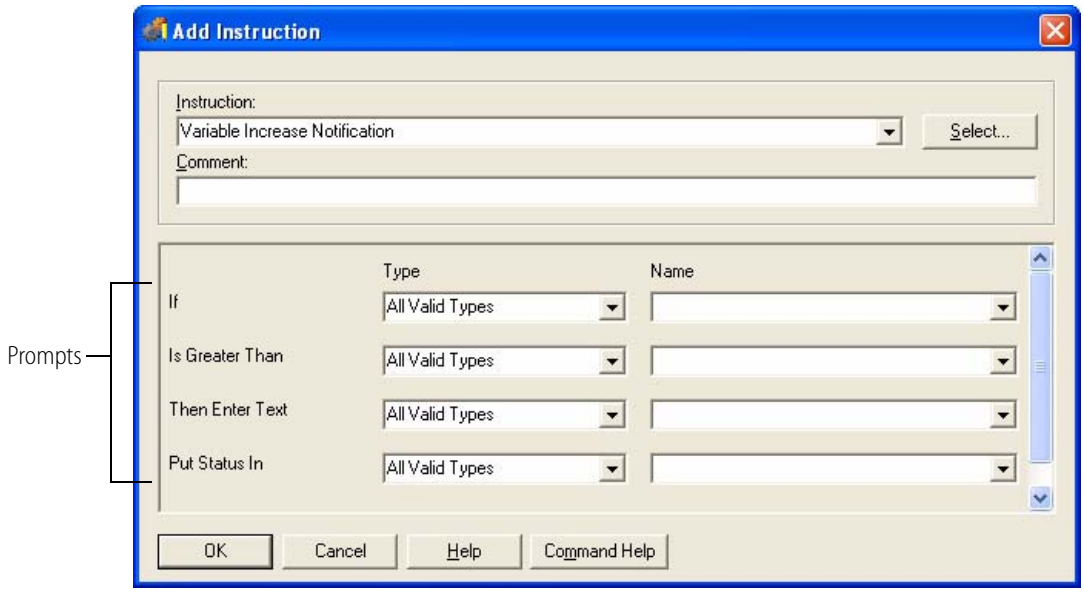
   The Add Instruction dialog box appears.



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 group when configure the Subroutine's parameters. See "Configuring Subroutine Parameters" on page 347.

   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.



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 is now set up to call the subroutine.

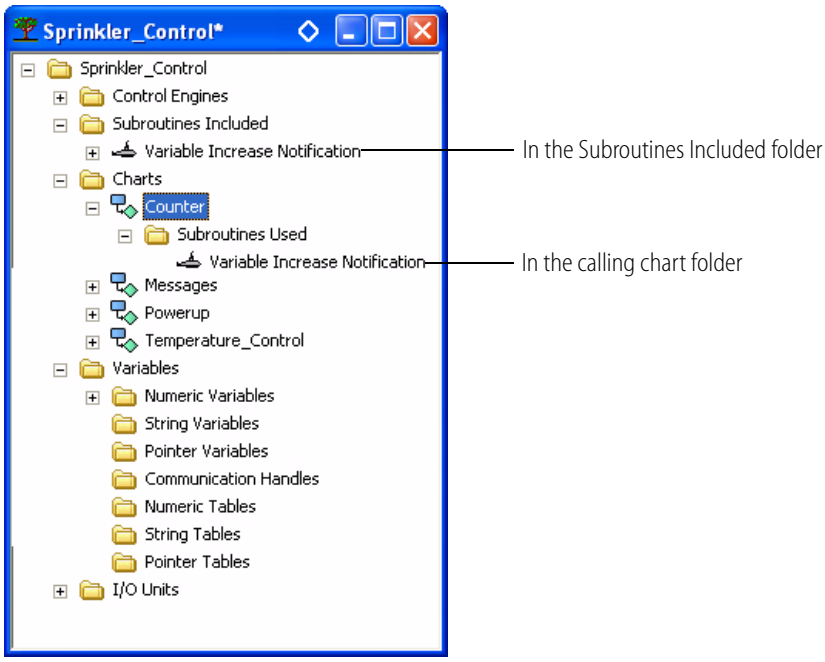
Debugging Subroutines

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 189). 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 194 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 that calls it.



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, filename, 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 198. To view and print instructions in the subroutine's blocks, see "Viewing and Printing Strategy or Subroutine Elements" on page 202.

Troubleshooting

This appendix provides general tips for resolving problems you may encounter while running PAC Control or communicating with your hardware. If you are running Windows NT or Windows 2000 and encounter problems with permissions, see page 366.

For information about types of errors and lists of error codes, see Appendix B, "Errors and Messages."

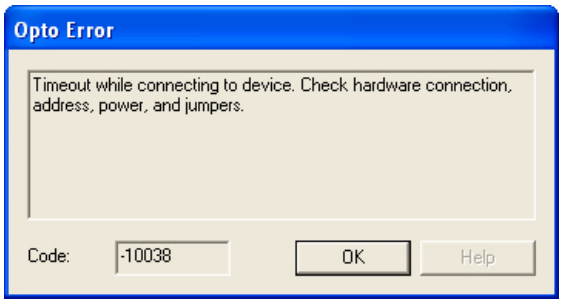
Also check troubleshooting information in your controller's user guide. See also, Opto 22 form 1690, the *SNAP PAC Brain User's Guide*form 1.

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:



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 360.

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 110. In the "List of Common Messages" on page 369, 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 361. 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.

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 369 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 207.

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. Click in the Strategy Tree to make it the active window.
2. Press CTRL+R.
3. Choose Compile→Compile All.
4. Choose File→Save All.

   The strategy database is rebuilt and the reference counts should be correct.

If You Have Memory Problems

Control engine memory is allocated as shown in the following table. You can see the total amount of RAM and the amount of battery-backed RAM available for use by inspecting the control engine in Debug mode (see page 110).

	SNAP-PAC-S1	SNAP PAC R-series
Total memory (RAM)	32 MB	16 MB
Memory used for control	32 MB	10 MB
Memory available for strategy and variables	16 MB	5 MB
Total battery-backed RAM	8 MB	2 MB
Battery-backed RAM for control*	8 MB	1 MB

* Stores persistent variables, variables initialized on download, autorun flag, and strategy archive. Note that strategies are not automatically saved in battery-backed RAM. Save your strategy to flash memory so it will be available if power is lost to the control engine. See "Saving a Strategy to Flash" on page 177.

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.

If you are using subroutines, use the minimum number of variables and size of tables that the process requires. Also, less memory is used if only one chart in the strategy calls subroutines than if multiple charts call subroutines.

Since the battery-backed RAM contains variables initialized on download, if you have a large number of these on a smaller controller, you can run out of persistent RAM. To avoid this problem, use as few persistent variables as possible and initialize all other variables on strategy run.

Although you can archive the strategy, the currently running strategy is not stored in battery-backed RAM. To make sure the strategy will run after a power loss, save the strategy to flash memory after downloading it. See page 177 for information on saving to flash.

Archiving Strategies to Battery-Backed RAM

Strategies are archived to battery-backed RAM, which is limited to 256KB. In addition to an archived strategy, battery-backed RAM holds persistent variables and variables that are initiated on download. If you have an unusually large strategy or large numbers of persistent variables or variables that are initialized on download, you may not have sufficient space for an archived strategy. See page 178 for more information on archiving.

Do You Use Online Mode?

If you frequently use Online mode to change your strategy, you may find you are having memory problems. 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, these additional chart copies begin to take up memory.

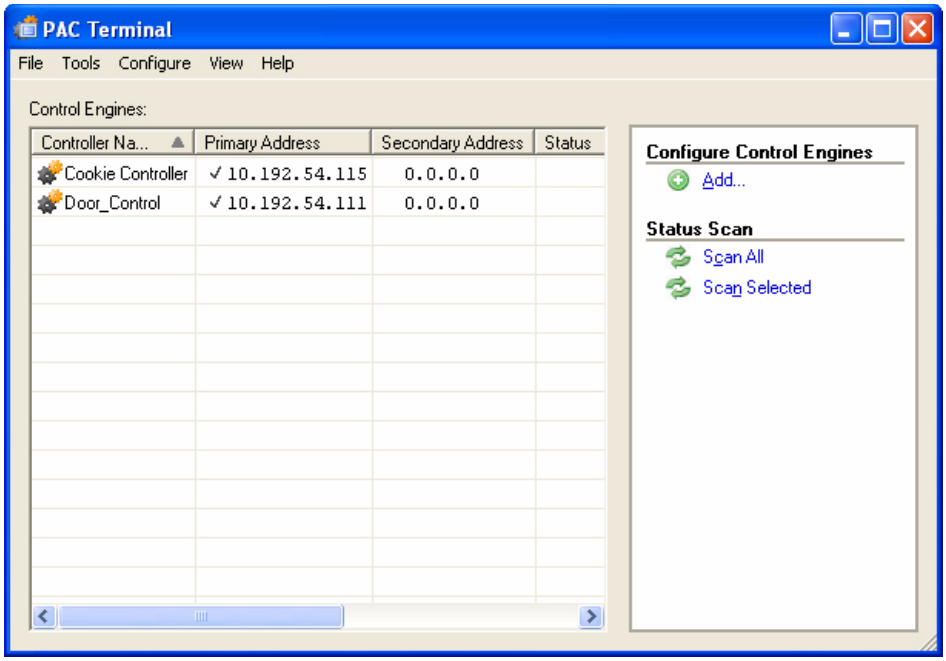
To avoid memory problems, stop the strategy after making several online changes. Completely compile and download the strategy, and old chart copies will be cleared from memory.

Checking Communication with the Control Engine

You can test communication with the control engine by using the PAC Terminal utility.

1. From the Start menu, choose Programs→Opto 22→PAC Project→Tools→PAC Terminal.

   The PAC Terminal window appears, showing all control engines configured on your system:



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 110.

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.

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 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 PING protocol.

Choose Start→Programs→MS-DOS Prompt. At the prompt, type:

```
ping [control engine's IP address]
```

(For example, type `ping 10.192.54.40`.)

If data is returned from the control engine, it 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.

If you are running Windows 95 or Windows 98, follow these steps:

a. Choose Start→Settings→Control Panel and double-click Network.

b. Highlight the adapter in the list. Make sure both NetBEUI and TCP/IP appear just below the name of the adapter. Click Properties.

c. Highlight TCP/IP and click Properties. Verify that the IP address and subnet mask are appropriate for your network.

If you are running Windows NT, follow these steps:

d. Choose Start→Settings→Control Panel and double-click Network.

e. Click the Protocols tab. Make sure both NetBEUI and TCP/IP are listed. Highlight TCP/IP and click Properties.

f. Highlight the adapter name in the list. Verify that the IP address and subnet mask are appropriate for your network.

If you are running Windows 2000, follow these steps:

g. Choose Start→Settings→Control Panel and double-click Network and Dialup Connections.

h. Right-click your network card and choose Properties from the pop-up menu. Make sure that TCP/IP is present and checked.

i. Highlight TCP/IP and click Properties. Verify that the IP address and subnet mask 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

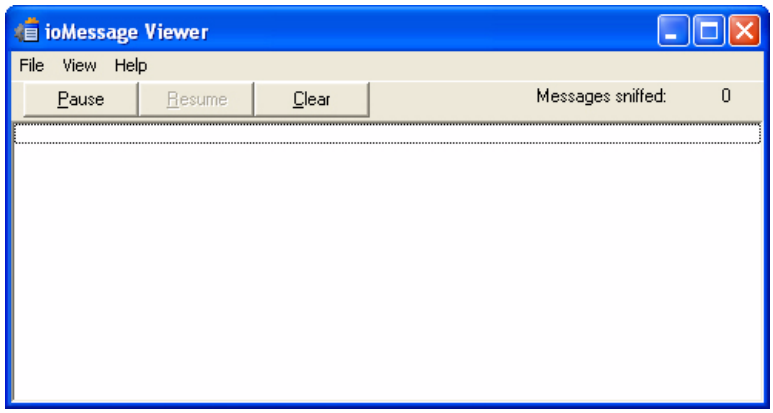
Checking Detailed Communication Information Using PAC Message Viewer

For detailed information about each communication transaction, use the PAC Message Viewer utility.

1. In the Start menu, choose Programs→Opto 22→PAC Project→Tools→PAC MessageViewer.

   You can also start PAC Message Viewer from PAC Terminal by choosing Tools→Start PAC Message Viewer, or from PAC Control in Debug mode by choosing Debug→Sniff Communication.

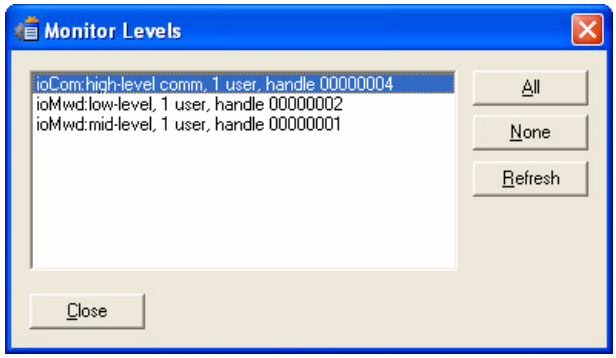
   The PAC Message Viewer window appears.



   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.

2. 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.



3. 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.

4. 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.

5. View or edit the file as needed, and then close it.

   You return to the PAC Message Viewer window.

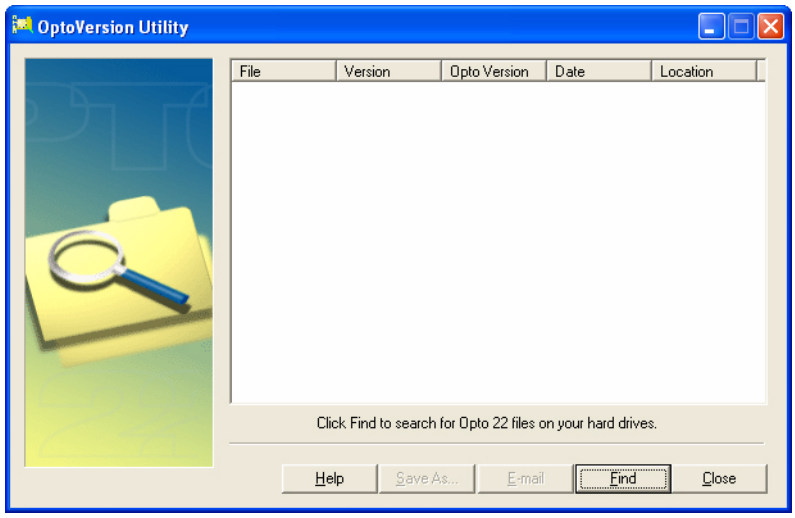
6. To resume logging, choose File→Log to File.
7. To rename the log file or change its location, choose File→Select Log File. Navigate to the location where you want to save the file and enter a name. Click OK.
8. When you have finished monitoring communication, close the PAC Message Viewer window.

   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. From the Start menu, choose Programs→Opto 22→PAC Project→Tools→OptoVersion.



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 2000

When you set up controllers on a computer running the Microsoft Windows 2000 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.

WARNING: *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. From the Windows Start menu, select Run.

   The Run dialog box appears.

2. Enter the following command in the Open field and press ENTER:

   ```
   regedt32
   ```

   NOTE: This is NOT regedit.exe, which is a similar tool.

   The Registry Editor main window appears with several open windows inside it.

3. Select the HKEY_LOCAL_MACHINE window to make it active.
4. Double-click the Software folder in the HKEY_LOCAL_MACHINE window.
5. Select the Opto22 folder.
6. In the Security menu, choose Permissions.

   The Registry Key Permissions dialog box opens. Make sure that "Opto22" appears next to Registry Key at the top of the window.

7. Click Add.
8. In the Select Users, Computers, or Groups dialog box, select the name of the appropriate group or domain from the Look In drop-down list.
9. In the Name list, select the name of the user or group that should have control engine access and then click Add.
10. If it is not already selected, check "Full Control" in the Permission area. Make sure "Allow inheritable permissions from parent to propagate to this object" is checked.
11. Click OK.
12. Select Registry→Exit to close the Registry Editor.
13. Restart the computer.

The user or group you added can now use control engines without having administrator access.

Errors and Messages

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 369.

See "Troubleshooting" on page 357 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.

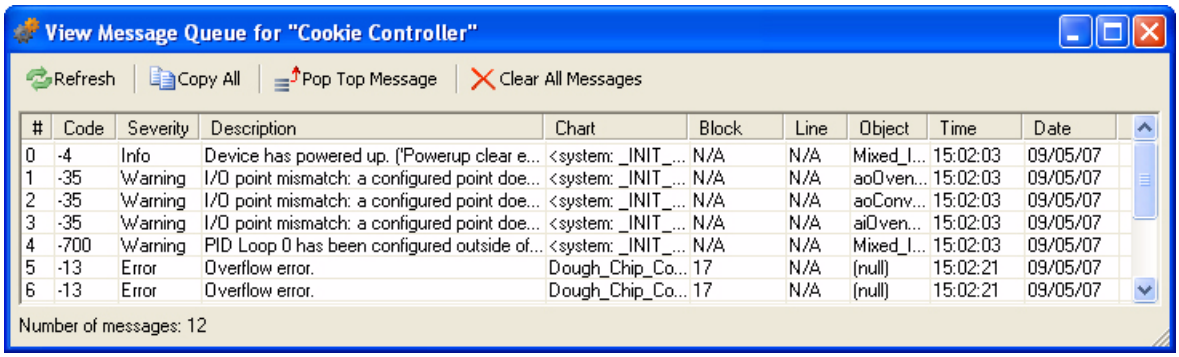
PAC Control Errors

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 "Inspecting Control Engines and the Queue" on page 110.) 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

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 283 and "Handling I/O Errors Efficiently" on page 97.

#	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.		

#	Description	Possible Cause	Q?	I/O?
-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 125.		
-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
-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. 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 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	

#	Description	Possible Cause	Q?	I/O?
-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
-35	I/O Point mismatch	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.		

#	Description	Possible Cause	Q?	I/O?
-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 page 4.		
-61	Dictionary full error. PAC Control engine dictionary is full and no more 'words' can be defined. Clear PAC Display words using PAC Display or PAC Terminal, if appropriate.	Attempted to create a variable, command, or similar item when there's no room left. Or large string or numeric tables are being created in the strategy.		
-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 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.		
-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
-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 361.		X
-417	Cannot open file	File does not exist or filename 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.	Ethernet "accept" attempted, but no more sessions are available (32 total).	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
-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
-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
-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	
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.		

PAC Control Files

Introduction

This appendix lists all of the PAC Control file types and special files. You can use this information as a reference to determine what types of files are present in your PAC Control project directory when you're looking through your PAC Control or project directory.

Files Related to a Strategy

<strategy>.idb	PAC Control strategy database
<strategy>.crn	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 185)
<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 page 178 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

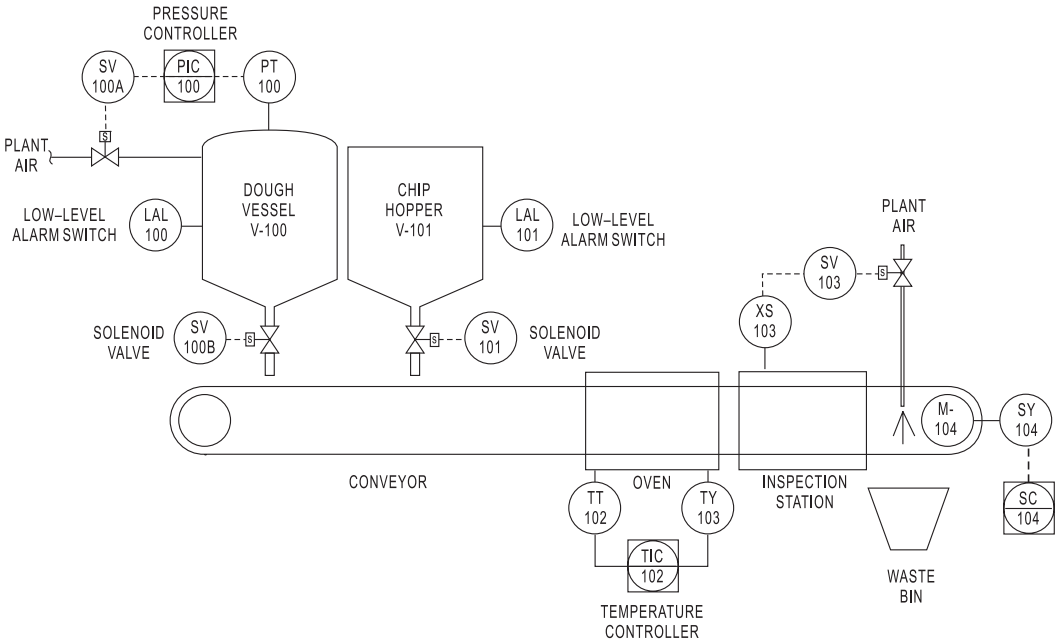
Sample Strategy

Introduction

Chapter 2, "PAC Control Tutorial," 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), signalling 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

OptoScript Language Reference

OPTO 22

Introduction

This appendix includes the following reference information about the OptoScript language:

OptoScript Comparison with Standard Programming Languages 381
Notes to Experienced Programmers 386
OptoScript Lexical Reference 388
OptoScript Grammar Syntax Reference 391

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 Chapter 11, "Using OptoScript," and Appendix E, "OptoScript Command Equivalents."

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.

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 only exist 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	*:	any character
(parenthesis):	content is treated as a unit	*letter*:	any character a through z, upper or lower case
[brackets]:	set of possible items		
opt subscript:	item is optional	*digit*:	one of [0 1 2 3 4 5 6 7 8 9]
no subscript:	item is not allowed	*non-zero-digit*	one of [1 2 3 4 5 6 7 8 9]
0+ subscript:	0 or more items may be chosen	*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]
1 subscript:	one item must be chosen		
1+ subscript:	one or more items must be chosen		

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
then
else
elseif
endif
	for
	to
	step
	next
		switch
		endswitch
		case
		break
		default
			while
			do
			wend
			
			repeat
			until
				Chr
				
				null

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* **;**

Index

A

action block
  definition 214
action command
  definition 52
active address 104, 284
adding
  analog point 133
  command 249
    in OptoScript 338
  digital point 130
  I/O unit 124
  PID loop 143
  pointer variable to watch window 246
  table variable 239
  table variable to watch window 247, 248
  text to chart 222
  variable 235
  variable to watch window 244
algorithms for PID loops 297
analog
  adding I/O point 133
  changing I/O while strategy is running 155
  copying I/O point 139
  I/O unit commands 284
  minimum/maximum value 258, 259
  offset and gain commands (instructions) 258
  point commands 258, 322
  point, definition 48
  TPO 258
applications
  launching from PAC Control 70
archiving strategies
  restoring archives 117
  to computer 178
  to the control engine 116, 178
argument 80, 251
  in OptoScript 324
ASCII, decimal and hex values of characters 309
auto stepping through a chart 191, 193
automation 47
autorun 188

B

background downloading 181, 183
baud rate and mode for serial I/O units (Pro) 127
baud rate for serial I/O units 126
bill of materials 204, 205
bit commands 290
bitwise operator, in OptoScript 333
block
  block 0 214
  changing color or size 215, 224
  connecting blocks 219
  definition 214
  deleting 224
  drawing 218
  finding in a chart 63
  moving 223
  naming 83, 218
  OptoScript 314
  selecting 222
  stepping through 191
BOM 204, 205
bookmark 339
breakpoints in a chart 194

C

calling a subroutine 352
case statement 319, 334
  programming example 88
changing
  active control engine 109
  analog I/O while strategy is running 155
  baud rate and mode for serial I/O units (Pro) 127
  baud rate for serial I/O units 126
  block color or size 215, 224
  chart background 215
  chart name or description 227
  column width in a dialog box 63
  command (instruction) 253
  connection line color or size 224
  control engine definition 108
  debug level 189
  debugger speed 190
  digital I/O while strategy is running 153
  I/O points 141
  I/O unit configuration 129
  I/O while strategy is running 152
  numeric variable 244
  PID loop 148
  pointer table 248
  pointer variable 246
  scale of a chart 61
  strategy download compression 187
  string table 247
  string variable 244
  text color or size 215, 224
  text in chart 222
  variable 244
chart
  adding text to 222
  auto stepping 191, 193
  breakpoints 194
  calling a subroutine 352
  changing background 215
  changing name or description 227
  changing scale 61
  closing 226
  commands 260
  contents 214
  copying 226
  creating 213
  definition 51
  deleting 228
  designing 78
  exporting 228
  finding a block 63
  flow-through logic 81
  guidelines for design 82
  importing 229
  loop logic 81
  looping 318
  monitoring in watch window 168
  naming 83, 227
  opening 226
  pausing 190
  printing commands (instructions) 201
  printing graphics 198
  replacing elements 208
  saving 175, 226
  searching 207
  status 51, 197
  stepping through 191
  viewing 59, 197
  zooming in or out 61
chart element
  changing color or size 224
  cutting, copying, and pasting 224
  moving 223
  selecting 222
closing
  chart 226
  strategy 175, 176
command
  adding 249
  adding in OptoScript 338
  arguments 80
  changing 253
  commenting out 254
  cutting, copying, and pasting 254, 255
  definition 52
  deleting 253
  deleting temporarily 254
  for continue block 255
  groups
    analog point commands 258

chart commands 260
communication commands 261
control engine commands 278
error handling commands 283
I/O unit—event message commands 286
I/O unit—scratch pad commands 288
I/Ounit—memory map commands 287
logical commands 290
mathematical commands 291
miscellaneous commands 295
PID—Ethernet commands 296
pointer commands 299
simulation commands 301
string commands 301
time/date commands 310
timing commands 310
in OptoScript 322
function 325
procedure 324
printing 201
comment 254, 295
in OptoScript 339, 390
communication
Ethernet link redundancy 104
I/O memory map commands (instructions) 287
I/O unit—event message commands (instructions) 286
I/O unit—scratch pad commands 288
incoming 264
link redundancy 284
optimizing throughput 95
outgoing 263
PC and control engine 99
peer-to-peer 262, 288
TCP/IP 261, 262
with serial communication module 262, 265, 266
communication commands 261
communication handle
definition 52, 233, 261
examples
File 269
FTP 273
serial 276
TCP 264, 265
File 268
FTP 273
port numbers 263
serial 276
serial communication modules 265
TCP 261, 262
types 261
viewing 244
comparison operators 331
Compile toolbar 57
compiling
copy errors and warnings 180
strategy 26, 179
strategy without downloading 182
subroutine 351
condition block
definition 214
increasing efficiency in loops 96
condition command, definition 52
condition in OptoScript 320
Configure mode, definition 55
Configure toolbar 56
configuring
continue block 255
control engine 99
I/O points 129
I/O unit 121
PID loop 142, 297
serial module 137
subroutine parameters 347
table variables 239
variables 235
connection line
changing color or size 215, 224
definition 215
deleting 224
drawing 219
moving 223
selecting 222
constant, *See* literal
continue block
command (instruction) 255
configuring 255
definition 215
control characters, adding to string 303

control concepts 47
control engine
  active control engine 109
  archiving strategies 115
  changing definition 108
  changing the active control engine 109
  commands (instructions) 278
  configuring 99
  definition 47
  deleting from strategy 110
  download file, creating 185
  inspecting 110
  primary and secondary addresses 104
  removing association with strategy 109
  restoring archived strategies 117
  testing communication 360
  viewing message queue 112
control network
  segmenting from company network 103
control structures, in OptoScript 333
control system example 46
controller, *See* control engine
converting a strategy 369
converting IEEE float 292
copying
  block 224
  chart 226
  command (instruction) 254, 255
  configured I/O point 139
  connection line 224
  I/O configuration 123
  text block 224
count variable programming example 87
counter
  commands (instructions) 280
  programming example 86
CRC commands 278
creating
  flowchart 213
  PID loop 142
  strategy 173
  subroutine 345
  toolbar 69
  watch window 169
  See also changing
cross reference window 203
customizing toolbars 66
cutting
  block 224
  command (instruction) 254, 255
  connection line 224
  text block 224

D

data
  IEEE float format 292
data format
  IEEE float 292
data types in subroutines 344
data, sorting, in a dialog box 64
date commands 310
Debug mode
  definition 55
  inspecting control engine 110
  inspecting I/O 152
Debug toolbar 57
debugging
  changing speed 190
  choosing level 189
  strategy 189
  subroutines 354
  when using OptoScript 342
delay
  commands 310
  using in condition block loops 96
deleting
  chart 228
  chart elements 224
  command (instruction) 253
  command, temporarily 254
  control engine association with strategy 109
  control engine from strategy 110
  I/O point 141
  I/O unit 129
  PID loop 149
  toolbar buttons 68
  variable 244
designing
  basic rules for strategies 81
  for faster throughput 95

steps for 73
dialog box
sizing columns 63
sorting columns 64
digital
adding digital point 130
changing I/O while strategy is running 153
copying I/O point 139
counter 280
I/O unit commands 284
latches 280
point commands 322
point, definition 48
pulse commands (instructions) 281
quadrature counter 281
totalizer 281
directory listing 274
disabling I/O 282
docking
Strategy Tree 60
watch window 170
down timers 311
downloading
files using FTP 273
files using PAC Terminal 119
initialization file for table variable 242
strategy 179, 182
changing compression 187
without using PAC Control 185
drawing toolbar 56, 215
duplicate object names 369

E

editing, *See* changing
EEPROM, *See* flash
elapsed time 195
error
cannot delete item 358
copy compile errors 180
queue 112, 367
status codes 369
TCP/IP cannot connect 361
types 367
See also troubleshooting
error handling
commands (instructions) 283
programming example 85
Ethernet
connections 267
link redundancy 104
commands for 284
ports 267
Ethernet I/O units and commands 210
example
case statements 88, 319
complex operations 321
control system 46
counter 86
creating messages to display on screen 84
error handling chart 85
File communication handle 269
flag lock 90
FTP communication handle 273
if statements 320
loops 318
mathematical expressions 315
pointers (indexing) 93
repetitive actions 87
serial communication handle 276
string building 305, 316
string data extraction 305
string formats 304
string table 304
subroutine parameters 349
TCP communication handle 264
timer 89
exiting, *See* closing
exporting
chart 228
I/O configuration 123

F

File communication handle 268
examples 269
file system on the brain
moving files via FTP 273
saving files to flash 278
file toolbar 56
finding a block in a chart 63
flag lock example 90

flash memory 152
saving files to 278
saving strategy to 177
float
IEEE 292
floating point
converting to integer 294
converting to string 308
definition 51
in logic 291
variable, definition 232
flowchart, *See* chart
flow-through logic 81
for loop 318, 336
format
File communication handle 268
FTP communication handle 273
IEEE float 292
serial communication handle 276
TCP communication handle 263
FTP
directory listing 274
FTP communication handle 273
full debug, definition 189
functions in OptoScript 322

G

gain
and offset 258
in PID loops 162
grammar in OptoScript 391

H

hardware requirements for PAC Control 5
help
See also troubleshooting
help, online 71
hex display mode 65
hiding toolbars 57
high-density digital
enabling commands 210
high-density points 285
host task 95
definition 52
increasing frequency 96
Hungarian notation 325

I

I/O
changing configured point 141
configuring 129
copying configured point 139
deleting configured point 141
disabling 282
exporting configuration 123
importing configuration 124
in strategy design 75, 77
monitoring in watch window 168
moving configured point 138
naming points 83
point numbers 124
point, definition 47
saving configuration 152
segmenting I/O from host network 103
using in OptoScript 322
viewing all in a strategy 202, 203
I/O unit
adding 124
changing 129
commands 97, 284, 286, 287, 288
configuring 121
definition 121
deleting 129
exporting configuration 123
importing configuration 124
monitoring in watch window 168
point numbers 124
IEEE float 292
if statements
in OptoScript 334
if/then statements 320
importing
chart 229
I/O configuration 124
info, viewing 112
initialization file 119, 240
creating 240
downloading 242
input point

definition 47
disabling 282
input/output, *See* I/O
inspecting
control engine 110
control engine using PAC Terminal 115
I/O 152
I/O using watch window 168
PID loop 156
See also viewing
installing PAC Control 5
instruction, *See* command
integer
converting to float 294
definition 51, 291
in logic 291
variable, definition 233
interacting algorithm for PID 297
IP address
loopback 263, 265
primary and secondary 104
ISA algorithm for PID 297
IVAL 282

L

lag time, for PID 157
latch
commands (instructions) 280
launching, *See* opening
legacy options, enabling 209
link redundancy 284
literal
constant 235
in OptoScript 388
load last mode at startup 174
load last strategy at startup 174
local subroutine parameter 344
local variable 344
log file 376
logic, in charts 81, 96
logical commands (instructions) 290
logical operator in OptoScript 332
logical true and false 291
loop logic 81
loopback address 263, 265
looping 318, 335, 336

M

mathematical commands (instructions) 291
mathematical expressions 315
in OptoScript 331
memory 359
memory map commands 287
message on screen, programming example 84
message queue 112
minimal debug, definition 189
minimum/maximum value 258, 259
mistic I/O units and commands 211
mode
Configure, definition of 55
Debug, definition of 55
Online, definition of 56
mode toolbar 57
modifying, *See* changing
monitoring, *See* inspecting *and* viewing
moving
block 223
connection line 223
files via FTP 273
I/O point 138
text block 223
to another window or chart 59
toolbar 57, 67
toolbar buttons 68
multitasking 52, 95
and strings 303

N

naming
block 83, 218, 219
chart 83, 227
conventions 82, 325
I/O points 83
variables 83, 325
naming conflicts 369
network
segmenting control network 103
network redundancy 104
number, converting to string 308

numeric literals in OptoScript 326
numeric table
  adding 239
  as alternative to strings 303
numeric variable
  adding 235
  in OptoScript 327

O

offset and gain commands (instructions) 258
offset, definition 258
online help 71
Online mode
  avoiding memory problems 56
  definition 56
opening
  applications from PAC Control 70
  chart 226
  strategy 174
  watch window 170
operator
  bitwise, in OptoScript 333
  comparison, in OptoScript 331
  in standard commands (AND/OR) 252
  logical, in OptoScript 332
  order of precedence 389
OptoScript
  bitwise operators 333
  block example 314
  block, definition 215
  bookmark 339
  case statements 319, 334
  command types 324
  commands 322
  comments 390
  communication handle 271
  comparison operators 331
  comparison with other languages 381
  complex loops 318
  conditions 320
  control structures 333
  debugging strategies 342
  definition 313
  editor 337
  for loop 336
  functions 322
  if statements 334
  language reference 381
  literals 388
  logical operator 332
  math expressions 315, 331
  notes for programmers 386
  numeric literals 326
  numeric variables 327
  pointers 328
  precedence for operators 389
  repeat loop 336
  string handling 316
  strings 327
  switch statements 334
  syntax 323, 387, 391
  tables 329
  toolbar 338
  troubleshooting 340
  types of commands 324
  when to use 314
  while loop 335
OptoVersion utility 365
order of precedence for operators 389
output point
  definition 47
  disabling 282

P

PAC Control
  customizing 66
  definition 45
  designing a strategy 73
  directory, list of files in 376
  errors 367
  files, list of 375
  installing 5
  main window 55
  mode 55
  opening other applications 70
  programming 73, 313
  system requirements 5
PAC Display 52
PAC Manager 121
PAC Message Viewer utility 363

PAC Terminal utility
downloading files 119
inspecting control engines 115
testing communication with control engine 360
PAC Utilities 115, 119, 360, 363, 365
page setup for printing graphics 198
parallel algorithm for PID 297
parameters, subroutine 343, 347
passed-in subroutine parameters 344
pasting
block 224
command (instruction) 254, 255
connection line 224
text block 224
pausing
chart 190
peer-to-peer communication 262, 288
permanent storage, *See* flash memory
permissions in Windows 2000 366
persistent data 235
persistent RAM 359
PID loop
algorithms 297
changing 148
configuring 142, 297
definition 296
deleting 149
determining system lag 157
inspecting 156
tuning 162
PID—Ethernet commands 296
pointer
adding 235
commands 299, 328
definition 52, 233, 299
in OptoScript 328
programming example (indexing) 93
pointer table 300
adding 239
port
Ethernet 267
peer-to-peer communication 263
Powerup chart 51
uses 82
precedence for operators 389
primary and secondary addresses
for control engine 104
for I/O unit 126
printing
bill of materials 204, 205
chart commands (instructions) 201
chart graphics 198
chart instructions 256
page setup 198
subroutine commands 201
problems, *see* troubleshooting
Product Support 4
programming
comparing OptoScript with other languages 381
examples 83
in PAC Control 73
pulse
commands (instructions) 281

Q

quadrature counter commands (instructions) 281
queue
checking messages 112, 367
quotation marks in strings 327

R

RAM 359
redundant communication links 104, 284
repeat loop 318, 336
repetitive actions programming example 87
replacing elements 208
retrieving a directory listing 274
RTS/CTS 277
running a strategy 27
automatically 188
manually 188

S

saving
chart 175, 226
configurations to flash memory 152
files to flash memory 278

strategy 175
strategy to flash 177
subroutine 351
saving I/O unit configuration 152
scratch pad commands 288
searching 207
secondary address 104, 126
segmenting control network 103
selecting
block or text block 222
connection line 222
serial communication handle 276
serial communication module
communicating with 262, 265, 266
configuring 137
serial devices, communicating with 276
serial I/O unit
changing baud rate 126
changing baud rate and mode (Pro) 127
setpoint, for PID 142
setting hex display mode 65
showing toolbars 57
Simple I/O units and commands 210
simulation commands 301
sizing columns in a dialog box 63
SNAP-PAC-S2
RTS/CTS 277
software, launching from PAC Control 70
sorting columns in a dialog box 64
starting a strategy 188
status codes 369
step into 191
step out 191
step over 191
stepping
inside blocks 192
one block at a time 191
through a chart 191
within a block 189
stopping
strategy 189
strategy
archiving
to computer 178
to control engine 116, 178
closing 175, 176
compiling 26, 179
compiling without downloading 182
creating 173
debugging 189
debugging when using OptoScript 342
definition 53, 173
deleting control engine 110
designing 73, 81
designing for faster throughput 95
disassociating control engine 109
downloading 179, 182
downloading, changing compression 187
downloading, without using PAC Control 185
files 375
including subroutine 351
opening 174
replacing elements 208
restoring archive from control engine 117
running 27
automatically 188
manually 188
saving 175
saving to flash EEPROM 177
searching 207
stopping 189
viewing all operands 203
viewing variables and I/O 202
Strategy Tree
definition 58
docking 60
icons 58
string
adding control characters 303
building, example of 305
commands 301, 327
equivalents in Visual Basic and C 307
convert-to-string commands 308
data extraction, example of 305
definition 51
examples 304, 327
in OptoScript 327
length and width 302
quotation marks 327
variable, adding 235

variable, definition of 233
string handling 316
string table
adding 239
example 304
subroutine
adding commands (instructions) 350
adding local variables 350
auto stepping 193
breakpoints 194
calling from a chart 352
compiling 351
configuring parameters 347
configuring parameters, example 349
creating 345
data types 344
debugging 354
definition 343
including in strategy 351
list of files in 376
parameters 343
pausing 190
printing commands 201
printing graphics 198
saving 351
searching 207
stepping through 191
tips 345
viewing 355
viewing all operands 203
viewing variables 202
Switch & Run 181
switch statement 319, 334
switching strategies 181, 183
syntax
common errors, in OptoScript 341
in OptoScript 323, 387, 391
system requirements for PAC Control 5

T

table
commands 285, 295
in OptoScript 329
numeric table as alternative to strings 303
table variable 234
adding 239
changing 247, 248
configuring 239
initializing during strategy download 240
monitoring in a watch window 247, 248
optimizing throughput 97
tabs, using to view windows 59
target address 284
task queue 95, 260
TCP communication handle 261, 262
TCP/IP cannot connect error, resolving 361
text block
deleting 224
moving 223
selecting 222
text, changing color or size 215, 224
throughput, optimizing 95
time/date commands (instructions) 310
timer
commands (instructions) 310, 311
definition 51
down timer 311
programming example 89
up timer 312
variable, definition 233
timing a process, commands used for 310
toolbar 56
buttons, moving and deleting 68
customizing 66, 69
drawing 215
hiding 57
matching to screen resolution 66
moving 57, 67
OptoScript 338
showing 57
totalizer commands (instructions) 281
TPO 258
trigonometry commands (instructions) 291
troubleshooting
how to begin 357
in OptoScript 340
memory problems 359
memory problems from Online mode changes 56
PAC Message Viewer utility 363

Product Support 4
Windows 2000 permissions 366
tuning
PID loop 162
types of OptoScript commands 324

U

Ultimate I/O units and commands 210
up timers 312

V

variable
adding 235
changing 244, 246
configuring 235
definition 53, 231
deleting 244
in strategy design 77
literal (constant) 235
monitoring in a watch window 168, 244, 246
naming 83, 325
OptoScript compared to other languages 386
persisent data in 235
pointer, definition 299
table 234
types of data 232
types, in PAC Control 233
viewing all in a strategy 202, 203
velocity algorithm for PID 297
version, checking with OptoVersion 365
View toolbar 57
subroutine
viewingviewing, *See also* inspecting
viewing
all operands 203
all variables and I/O 202
another window or chart 59
bill of materials 204, 205
chart 197
chart instructions 256
communication handle variable 244
message queue 112
numeric table variable 247
numeric variable 244
PID loop 156
pointer table variable 248
pointer variable 246
string table changing
numeric table 247
string variable 244
subroutine 355

W

warning
copy compile warnings 180
viewing 112
watch window
creating 169
docking 60, 170
monitoring pointer variable 246
monitoring table variable 247, 248
monitoring variable 244
opening 170
while loop 318, 335
Windows 2000
permissions 366

X

XVAL 282

Z

zooming in or out 61
z-order, changing 223